Numéro d'ordre NNT : 2022LYSES002

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON

opérée au sein de
l'**Université Jean Monnet de Saint-Étienne**

École doctorale N°488
Sciences, Ingénierie, Santé

Spécialité : **Sciences et Technologies de l'Information et de la Communication - Image, Vision, Signal**

Soutenue publiquement, le 18/01/2022 par :

Jules Rio

Deep Learning methodologies for denoising periodic signals

Méthodes d'apprentissage profond pour le débruitage de signaux périodiques

Devant le jury composé de :

Caroline CHAUX	*Chargée de Recherche HDR,Université Aix-Marseille*	Rapporteuse
Vicente ZARZOSO	*Professeur, Université Côte d'Azur*	Rapporteur
Jean-Christophe PESQUET	*Professeur, CentraleSupélec, Université Paris-Saclay*	Président
Abdourrahmane ATTO	*Maitre de Conférences HDR, Université Savoie-Mont-Blanc*	Examinateur
Olivier ALATA	*Professeur, Université Jean Monnet, Saint-Etienne*	Directeur
Christophe DUCOTTET	*Professeur, Université Jean Monnet, Saint-Etienne*	Co-directeur
Fabien MOMEY	*Maître de Conférences, Université Jean Monnet, Saint-Etienne*	Co-encadrant

Jules Rio
Deep Learning methodologies for denoising periodic signals
Rapporteurs: Caroline Chaux et Vicente Zarzoso
Président : Jean-Christophe Pesquet Examinateurs : Abdourrahmane Atto
Superviseurs: Olivier Alata
Co-encadrants : Christophe Ducottet et Fabien Momey

Université Jean Monnet
Laboratoire Hubert Curien
Image Science & Computer Vision
Spatio-temporal Data Analysis
18 Rue du Professeur Benoît Lauras
42023 Saint-Étienne

Acknowledgements

First of all, I would like to thank my supervisor Olivier Alata, my co-director Christophe Ducottet and my adviser Fabien Momey, for their guidance and their trust during these three years, both in terms of scientific requirements and of research methodology. I would also like to thank Olivier (again), as well as Julian Tugaut, for giving me the opportunity to teach at Télécom Saint-Étienne.

I also want to thank Caroline Chaux and Vicente Zarzoso for accepting to review my PhD, as well as Abdourrahmane Atto for accepting to be a member of my thesis jury and Jean-Christophe Pesquet for accepting to be the president of my thesis jury. I am also grateful for all the advices they gave me both in their reports and during my defense.

I would also like to thank everyone in the IMOTEP project for being at the origin of my work but even more for helping me understand the concepts of physics that were needed for my thesis. In particular, I want to thank the collaborators from IREIS, and especially Johnny Dufils and Étienne Macron, for providing both their signals and their expertise.

I also want to thank everyone I have met in the lab for the great atmosphere. Among all those persons, I would like to have a special thank for Thomas, Léo and Paul for the "meetings" outside of the lab.

Even though that was not the best time to be a member of the association, but I would also like to thank the other members of the ASEC "Bureau" Aubin, Damien (and Paul, again). We could not organize as many events as we would have wanted, but at least, the association survived a pandemic!

Lastly, I would like to thank my parents, and more generally, my family. Finding the motivation for my thesis has not always been easy, especially with the lockdowns, and I would probably not have gone that far without their support.

Résumé de la thèse

Introduction et contexte : Le projet IMOTEP

Ces dernières années, les enjeux environnementaux ont mis en lumière la nécessité de réduire les émissions de dioxyde de carbone (CO_2). De nombreuses solutions sont étudiées pour réduire l'impact de la production d'énergie d'une part, et réduire les besoins énergétiques d'autre part. Selon un rapport publié en février 2019 [1], le transport routier représenterait à lui seul 17.94% des émissions de CO_2. Plusieurs solutions sont proposées, incluant notamment le développement des transports publics et l'utilisation d'énergies renouvelables. Cette deuxième approche peut être connectée au développement de véhicules électriques. L'utilisation de véhicules électriques implique toutefois une capacité à produire en grande quantité et de façon peu polluante de l'électricité, et pose également problème pour l'utilisation de certains matériaux comme le lithium, nécessaire à la fabrication des batteries. En conséquence, les recherches pour l'amélioration des moteurs à combustion interne restent indispensables. Dans ces moteurs, une grande partie du carburant est utilisée pour compenser des pertes d'énergie liées aux frottements. Cette part d'énergie perdue était estimée à un tiers de l'énergie totale en 2012 [2], et encore à un cinquième en 2019 [3]. Ce même rapport de 2019 estime qu'il serait encore possible de réduire de 40% les pertes liées au frottement, et pointe la faible efficacité actuelle des moteurs à combustion.

Dans ce contexte, le projet IMOTEP (Innovation MOTEur Propre) vise à optimiser les traitements de surface et l'utilisation des lubrifiants dans les moteurs à combustion interne. Pour cela, une première étape passe par l'analyse des pertes liées aux frottements. Des capteurs piézoélectriques sont donc utilisés pour mesurer les forces de frottement à certains points du moteur. Toutefois, en raison des imperfections des capteurs, des vibrations des mécanismes et d'autres phénomènes aléatoires, les signaux acquis sont fortement pollués et difficilement exploitables de façon directe. Afin de faciliter l'exploitation de ces signaux, une tâche de prétraitement appelée "débruitage" est indispensable pour supprimer ou réduire ces diverses pollutions.

Dans cette thèse, nous cherchons à exploiter les propriétés spécifiques des signaux étudiés afin d'améliorer les techniques de débruitage qui existent actuellement. Les propriétés principales des signaux étudiées sont ici :

- la périodicité, c'est à dire qu'un même phénomène se reproduit dans le temps;
- la présence de discontinuités;
- la présence de zones lisses entre ces discontinuités.

Dans notre travail, nous nous intéresserons à des signaux de frottement visqueux directement liés au projet IMOTEP, mais également à des électrocardiogrammes pour assurer la généralisation de notre travail. Nous proposerons aussi quelques éléments de débruitage de séquences d'images issues de microscopie électronique.

Chapitre 1 : La tâche de débruitage

La tâche de débruitage consiste à extraire un signal $\mathbf{e}$ d'un signal $\mathbf{y}$ issu d'une transformation du signal $\mathbf{e}$. Cette transformation peut prendre plusieurs formes. Elle peut être déterministe en appliquant une fonction, par exemple un filtre convolutif, au signal d'intérêt. Dans nos études, nous considérerons toutefois que cette transformation se fait de façon additive, par superposition d'une composante $\mathbf{n}$ indépendante du signal d'intérêt.

Cette tâche peut aussi être associée à la séparation de sources, qui consiste à supposer que le signal acquis est une superposition de plusieurs signaux d'intérêt et à isoler chacun de ces signaux d'intérêt. Cette tâche plus générale a donné lieu à de nombreux travaux qui peuvent être réutilisés pour le débruitage. En particulier, le débruitage peut être vu comme une séparation de sources avec deux sources : le signal d'intérêt et le bruit.

Dans ce chapitre, nous rappelons les différentes méthodes qui ont été utilisées pour résoudre les tâches de débruitage et de séparation de sources. Nous commençons par des méthodes conventionnelles, comme les méthodes à base d'ondelettes ou de moyennes non locales. Après cela, nous nous intéressons plus particulièrement aux méthodes basées sur les réseaux de neurones, qui ont pris une place particulièrement importante dans la recherche ces dernières années. Après une rapide introduction sur des éléments généraux concernant l'apprentissage profond, nous évoquons les différents types d'architectures et leur application dans le cadre du débruitage et de la séparation de sources. Nous détaillons plus particulièrement les réseaux convolutifs, qui occuperont une place importante de notre travail, en évoquant les différentes propriétés de ces réseaux. Nous insistons notamment

sur la notion de champ réceptif qui joue un rôle important dans ces réseaux et définit le contexte pris en compte pour le débruitage de chaque échantillon d'un signal.

Enfin, nous évoquons certaines des métriques qui sont classiquement utilisées pour évaluer la qualité d'un débruitage.

Chapitre 2 : Analyse des signaux de frottement et débruitage par ajustement de modèle

Dans ce chapitre, nous effectuons une analyse des signaux afin de confirmer les propriétés périodiques, puis nous cherchons à ajuster des modèles paramétriques de frottement visqueux sur les signaux acquis.

Dans un premier temps, nous effectuons une analyse des signaux afin de vérifier les hypothèses de cyclostationnarité, c'est-à-dire de périodicité de leurs statistiques d'ordre 1 et 2. Rigoureusement, la confirmation des hypothèses effectuées nécessiterait l'application de tests statistiques. Toutefois, ces tests ne sont pas toujours simples à mettre en place, et l'objectif de la thèse est de proposer des méthodes qui restent valables dans des situations présentant de légères variations par rapport aux conditions théoriques. Nous nous contentons donc dans ce chapitre d'observations qualitatives sur ces statistiques d'ordre 1 et 2. Pour les statistiques d'ordre 1, nous étudions la moyenne des différentes périodes du signal et observons les zones pour lesquelles de fortes fluctuations apparaissent d'une période à l'autre. La variance est particulièrement élevée au niveau des discontinuités. Ceci n'est pas surprenant, d'une part parce que ces zones correspondent à un changement de direction du mécanisme, et donc à une possible instabilité, mais surtout parce qu'un léger décalage temporel entre deux périodes peut suffire à obtenir une forte différence lorsque l'une des périodes a déjà franchi la discontinuité et l'autre pas encore. Dans l'ensemble, on observe toutefois une cohérence avec les hypothèses de périodicité puisque la variance reste très faible en dehors des discontinuités. Pour l'ordre 2, nous effectuons des approximations de l'autocorrélation pour plusieurs valeurs de décalage temporel. Les observations sont assez similaires à celles faites à l'ordre 1. Il est toutefois difficile de conclure sur une cyclostationnarité à l'ordre 2 à partir d'un nombre limité de valeurs du décalage temporel. Comme l'ordre 2 aura moins d'impact sur les méthodes construites dans la suite de la thèse, nous estimons que ces premiers éléments suffisent pour notre cas.

Après cette première analyse, nous cherchons à ajuster des modèles de frottement sur les courbes acquises. Les modèles que nous utilisons dépendent de la vélocité au point de contact, et il nous faut donc obtenir des valeurs de cette vélocité. Nous disposons en plus des mesures de frottement de mesures de position. Une première approche pour obtenir la

vélocité serait alors d'appliquer un filtre dérivateur sur ces mesures de position. Toutefois, les mesures contiennent un bruit. Ce bruit est léger, mais la fréquence d'échantillonnage f_e est élevée et par conséquent, un filtre simple comme le filtre de réponse $H(z) = f_e(1 - z^{-1})$ risquerait de générer des fluctuations très importantes. Plutôt que de chercher à construire un filtre robuste, nous remarquons avec un calcul détaillé en Annexe C que le signal de position peut être approximé par un signal sinusoïdal. Nous décidons alors d'ajuster un modèle sinusoïdal sur les mesures de position, puis de dériver ce modèle pour obtenir les valeurs de la vélocité. Nous initialisons les paramètres de la sinusoïde à partir de la transformée du signal et de son amplitude crête-à-crête. Nous optimisons alors successivement les différents paramètres à partir de l'algorithme de Levenberg-Marquardt [4, 5, 6] puis nous les optimisons de façon jointe. Nous répétons alors cette procédure pour optimiser encore le résultat obtenu, et montrons que le modèle ainsi obtenu approxime bien les signaux acquis.

Comme nous disposons maintenant des valeurs de la vélocité, nous pouvons finalement effectuer l'ajustement des modèles de frottement visqueux. Nous comparons ici deux modèles [7, 8]. La procédure d'ajustement se fait là encore par une optimisation individuelle de chaque paramètre dans un premier temps, puis par une optimisation jointe dans un deuxième temps, avec répétitions pour affiner le modèle obtenu. Un algorithme à région de confiance est cette fois utilisé plutôt que l'algorithme de Levenberg-Marquardt. Les deux modèles parviennent à représenter correctement la forme générale des signaux, mais ignorent certains phénomènes, notamment un hystérésis entre la phase d'accélération et celle de décélération. Ils négligent également le lissage lié à la fonction de transfert du capteur au niveau des discontinuités. La prise en compte de ces effets dans les modèles n'étant pas facile, et la méthode utilisée étant difficilement généralisable à d'autres types de signaux, nous décidons d'orienter la suite de la thèse vers des méthodes plus adaptatives, basées sur l'utilisation de réseaux de neurones. Les modèles obtenus dans ce chapitre permettront toutefois la génération de données d'évaluation avec un comportement proche de celui des signaux de frottement réels.

Chapitre 3 : Utilisation d'une régularisation inspirée de la variation totale pour l'entraînement de réseaux WaveNet

Dans ce chapitre, nous proposons une première approche pour débruiter les signaux à partir de réseaux de neurones convolutifs. Nous utilisons pour cela des variations du réseau proposé dans [9], en jouant sur le nombre de stacks, de canaux par couches, et de couches

par stack. Ces variations nous permettent de jouer à la fois sur le champ réceptif et sur le nombre de paramètres à apprendre.

Ne disposant pas de vérité de terrain pour les signaux de frottement fournis, nous générons des signaux synthétiques qui serviront à entraîner les réseaux et à évaluer leurs performances. Pour l'entraînement, les signaux utilisés sont définis simplement en sommant un signal carré et une sinusoïde pour constituer la vérité de terrain, et un bruit blanc est ajouté pour le signal bruité. Les signaux d'évaluation sont définis selon le même modèle ou en se basant sur les modèles obtenus dans le chapitre précédent. Nous testons par ailleurs différents types de bruits.

Par rapport à la méthode originale [9], nous définissons également une nouvelle fonction de coût. Celle-ci est définie en ajoutant au terme de reconstruction un terme de régularisation par variation totale, destiné à éviter de surapprendre les discontinuités.

Les résultats obtenus confirment qu'il est possible d'obtenir des résultats satisfaisants sans utiliser une architecture aussi lourde que l'architecture originale. Par ailleurs, en observant les résultats obtenus en utilisant des bruits colorés, en particulier avec un contenu basse fréquence, on constate l'intérêt du terme de régularisation qui facilite la généralisation sur ce type de bruit. Des observations qualitatives sur des débruitages de signaux réels montrent aussi que la majorité du bruit est correctement supprimée, et que la régularisation aide à l'obtention d'un signal lisse. Il est toutefois possible qu'une régularisation élevée supprime non seulement les bruits, mais aussi certains phénomènes ayant un intérêt physique.

Nous effectuons aussi des expériences sur d'autres types de signaux, pour lesquels on compare notre méthodologie avec une méthodologie issue de l'état de l'art [10]. On constate l'intérêt de notre méthode, qui reste compétitive voire obtient de meilleurs résultats que la méthode de référence utilisée, et permet surtout une forte réduction du temps de calcul en inférence.

Enfin, nous appliquons les réseaux appris dans ce chapitre à des séquences d'images de microscopie électronique. Pour cela, nous isolons les signaux issus des lignes ou des colonnes de chaque image, ou considérons l'évolution temporelle d'un pixel. Nous montrons que cette troisième approche permet une meilleure suppression du bruit et améliore la stabilité du résultat en comparaison aux deux autres méthodes. Un bruit lié au balayage du composant lors de l'acquisition est toutefois toujours présent et indique que l'utilisation des dimensions spatiales pourrait être indispensable pour un débruitage complet.

Chapitre 4 : Architectures P1D pour exploiter la périodicité dans les réseaux de neurones

La méthode présentée dans le chapitre 3 a permis l'obtention de résultats intéressants et parvient à préserver les discontinuités. Toutefois, la périodicité n'est pas exploitée. Nous proposons donc une nouvelle méthode, basée sur un redimensionnement des données, afin de prendre en compte cette propriété.

Le redimensionnement proposé consiste à découper le signal en un ensemble de sous-signaux de longueurs égales à la période. Les différents sous-signaux sont alors mis en parallèle dans une grille. Ainsi, le signal est transformé en une image 2D. Cette méthode présente l'avantage d'assurer l'obtention d'un signal stationnaire le long de la nouvelle dimension et de blanchir le bruit, toujours le long de cette deuxième dimension.

Nous proposons deux façons d'exploiter la grille ainsi créée. La première méthode, qui est la moins coûteuse, consiste à effectuer une moyenne glissante directement sur la grille périodique, permettant ainsi une réduction du bruit. Le signal est alors recomposé sous sa forme 1D puis débruité par un réseau qui peut être l'un de ceux définis dans le chapitre 3. Nous appelons cette procédure la procédure MA (pour *Moving Average*). L'autre méthode consiste à modifier certaines des convolutions 1D du réseau convolutif en convolutions 2D, afin de prendre en compte la nouvelle dimension. Cette méthode a l'avantage d'être plus adaptable à différents phénomènes, mais augmente le nombre de paramètres. Nous appelons cette méthode P1D.

Dans des cas théoriques où le signal est exactement périodique et la fréquence fondamentale exactement connue, les deux méthodes améliorent grandement le résultat comparativement aux réseaux 1D, y compris avec des bruits colorés. La prise en compte de fluctuations de la périodicité durant l'entraînement des modèles P1D aboutit à une perte de performance dans ce contexte théorique, mais permet toujours une amélioration par rapport aux réseaux 1D. De plus, cette prise en compte de fluctuations devient indispensable dans des cas plus réalistes, où les différentes lignes de la grille ne contiennent pas exactement la même phase. Cette observation est également confirmée visuellement sur des signaux réels. Les modèles P1D semblent aussi exhiber des phénomènes locaux qui se reproduisent périodiquement et qui étaient ignorés par les modèles 1D. Ces résultats sont valables à la fois en utilisant WaveNet, mais aussi Wave-U-Net [11], un autre réseau convolutif.

Nous appliquons également les modèles P1D à des électrocardiogrammes pollués par du bruit blanc. Pour cette expérience, il est possible d'entraîner avec des signaux ayant les mêmes propriétés que les signaux que l'on souhaite débruiter, puisqu'une base de données existante [12] peut être exploitée. On constate alors que dans ces conditions, les modèles

P1D permettent une légère amélioration des résultats par rapport aux modèles 1D, tout en étant capables de s'adapter aux fortes variabilités de périodicité présents dans ces signaux.

Chapitre 5 : Méthodes d'ensemble pour améliorer la robustesse des modèles

Dans le chapitre 4, on a montré avec les expériences sur les électrocardiogrammes que lorsque les données adéquates étaient disponibles pour l'entraînement, les modèles P1D étaient déjà capables d'obtenir des résultats satisfaisants sur des données réelles. Toutefois, les résultats sur les autres données ont montré que dans d'autres cas, cette généralisation aux cas réalistes n'était pas assurée. Nous cherchons dans ce chapitre à définir des méthodes permettant d'améliorer cette généralisation, y compris lorsqu'il n'est pas possible d'apprendre avec des données réelles.

Après une courte introduction sur les méthodes d'ensemble, nous proposons une méthode pour combiner les résultats de plusieurs réseaux P1D et 1D à travers un nouveau réseau de post-traitement. Les réseaux combinés sont appris dans un premier temps, puis le réseau de post-traitement est appris seul, en prenant en entrée les sorties des réseaux P1D et 1D.

Concernant les réseaux P1D et 1D, nous essayons aussi de voir comment l'incorporation de bruits colorés durant l'entraînement permet d'améliorer la généralisation à ce type de bruit en évaluation.

Nous menons ensuite une étude approfondie du comportement des différents modèles 1D, P1D et ensemblistes face à différentes configurations, définies par le niveau de bruit initial, la couleur du bruit, la fréquence du signal et les différences de phase entre les lignes de la grille. Cette étude permet de montrer que si les méthodes ensemblistes obtiennent assez rarement les meilleurs résultats, elles gagnent en robustesse face à des données pour lesquelles des décalages importants entre les lignes de la grille apparaissent. Par ailleurs, ces modèles se comportent particulièrement bien lorsque le rapport signal à bruit se situe avant débruitage dans une zone allant de -10 à -5dB, qui correspond relativement bien aux niveaux de bruits qu'on peut être amené à rencontrer avec des signaux réels. Enfin, les modèles ensemblistes se comportent bien avec les signaux réels, en étant capables de fournir un bon compromis entre la préservation des phénomènes locaux et l'obtention d'un signal lisse.

Ainsi, bien que les modèles ensemblistes ne soient pas encore significativement meilleurs que tous les modèles P1D, ils offrent déjà une robustesse intéressante, justifiant l'approfondissement de ces méthodes.

Chapitre 6 : Conclusion et perspectives

Dans cette thèse, nous avons proposé plusieurs approches pour le débruitage de signaux périodiques. Nous avons commencé par proposer une approche par ajustement de modèle, permettant de constater la nécessité d'aller vers des méthodes plus adaptables. Nous avons pour cela exploité des approches d'apprentissage profond, en commençant par ignorer la périodicité, en proposant une nouvelle fonction de coût pour limiter les risques de surapprentissage des discontinuités présentes dans les signaux, ce qui a aussi permis une meilleure généralisation à différents types de bruits. Les chapitres suivants ont porté sur la prise en compte de la périodicité, au travers notamment d'un redimensionnement des données, puis sur l'amélioration de la robustesse des modèles ainsi obtenus lorsqu'aucune donnée réaliste n'est disponible pour l'entraînement. Pour améliorer cette robustesse, nous avons proposé une approche par méthode d'ensemble, dont les résultats, bien qu'encore perfectibles, sont déjà intéressants.

Parmi les pistes possibles pour prolonger le travail effectué, on peut citer :

- Un entraînement non supervisé afin d'entraîner les différents modèles avec des données réelles, même lorsqu'aucune vérité de terrain n'est pas disponible. Cela pourrait être fait pour tout l'entraînement, mais aussi pour affiner un réseau préalablement entraîné avec des données synthétiques.

- L'utilisation de réseaux antagonistes, notamment pour fournir une régularisation plus robuste que celle proposée dans le chapitre 3. Quelques expériences ont déjà été effectuées dans ce sens, mais les premiers résultats n'étaient pas bons. Un approfondissement pourrait malgré tout être envisagé.

- Une extension de la méthode par pré-traitement effectué pour la grille pourrait être envisagée en proposant une recherche du phénomène d'intérêt dans une zone limitée, de façon similaire à une approche par patches.

- Un travail supplémentaire est encore nécessaire concernant les méthodes d'ensemble.

- L'utilisation de réseaux bayésiens ou de prédiction de variance pour quantifier l'assurance du réseau quant à sa prédiction.

- L'extension vers une séparation de sources. En effet, nous avons ici considéré uniquement un bruit additif, mais les expériences ont montré la présence d'autres phénomènes, qu'il pourrait être intéressant d'isoler.

Contents

List of abbreviations

a.e.	Almost Everywhere
ANN	Artificial Neural Networks
CNN	Convolutional Neural Networks
CO_2	Carbon dioxyd
DFT	Discrete Fourier Transform
DL	Deep learning
DTFT	Discrete Time Fourier Transform
ECG	Electrocardiogram
e.g.	*Exempli Gratia* (For example)
EMD	Empirical Mode Decomposition
FCN	Fully convolutional Networks
GAN	Generative Adversarial Networks
FCNN	Fully Connected Neural Networks
FIB	Focused Ion Beam
FIR	Finite Impulse Response
HEF	Hydromécanique Et Frottements (Hydrodynamics and friction
i.e.	*Id Est* (That is to say)
IIR	Infintie Impulse Response
IMOTEP	Innoveation MOTEur Propre
IREIS	Institut de Recherche En Ingénierie des Surfaces
LSTM	Long short-term memory
LTI	Linear, Time-Invariant
MAE	Mean Absolute Error
MSE	Mean Squared Error
NLM	Non-Local Means
NMF	Non-negative Matrix Factorization
PSS	Piston-Skirt Segment
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Networks
rpm	Revolutions per minute

SEM	Scanning Electron Microscopy
SGD	Stochastic Gradient Descent
SNR	Signal-to-Noise-Ratio
SNR_{seg}	Segmental Signal-to-Noise-Ratio
STFT	Short-Time Fourier Transform
tanh	Hyperbolic tangent
TV	Total Variation

List of notations

$*$	Convolution product
f_e	Sampling frequency
$\mathbf{I}$	Identity matrix
$\odot$	Elementwise multiplication
T_e	Sampling period, $T_e = \frac{1}{f_e}$

Comments on notations

In this work, normal lower case letters will be used for designing scalar values, bold capital letters will be used to denote matrices (e.g. $\mathbf{X}$), bold lower case letters will be used to denote vectors (e.g. $\mathbf{x}$). Random signals will be written as normal capital letters (e.g. X) (this notation might also be used for some specific scalars when no confusion is possible) and their realizations will be assimilated to (possibly infinite) vectors (and will therefore be written in bold lower case letters. For a random signal X, the autocorrelation (resp. autocovariance) function will be written R_{XX} (resp. C_{XX}) and the spectral density will be written S_X. The Discrete Fourier Transform of a signal $\mathbf{x}$ will be written as $\mathcal{F}(\mathbf{x})$ and the Discrete Time Fourier Transform will be written as $\mathcal{F}_d(\mathbf{x})$.

Indexes will be used for accessing the elements of vectors, matrices or signals. For example, $\mathbf{x}_n$ denotes the scalar element of vector $\mathbf{x}$ located at position n, $\mathbf{X}_{i,j}$ denotes the scalar element at line i, column j of matrix $\mathbf{X}$ and X_n denotes the random variable at time n of random signal X. For vectors or finite signals, we will number the indexes from 0, meaning that a signal of length N will be defined for indexes $0, 1, \ldots, N-1$.

For matrices, the notation $\mathbf{X}_{i,.}$ will be used to denote the i-th line and the notation $\mathbf{X}_{.,j}$ will be used to denote the j-th column.

For a vector (or finite signal) $\mathbf{x}$, the L^k norm will be written $\|\mathbf{x}\|_k$ and is defined as:

$$\|\mathbf{x}\|_k = \left(\sum_{n=0}^{N-1} |\mathbf{x}_n|^k \right)^{1/k}. \tag{0.1}$$

This definition can be extended to signals of two or higher dimensions.

For a scalar s and a matrix $\mathbf{X}$, $\frac{s}{\mathbf{X}}$ is the matrix where the element at line i and column j is defined as $\frac{s}{\mathbf{X}_{i,j}}$. Similarly, $s + \mathbf{X}$ is the matrix where the element at line i and column j is $s + \mathbf{X}_{i,j}$. The notation $\mathbf{X}^T$ denotes the transpose of the matrix $\mathbf{X}$. These notations also apply if the matrix $\mathbf{X}$ is replaced by a vector $\mathbf{x}$.

When working with complex numbers, $\bar{x}$ will denote the conjugate of the scalar x. For a vector $\mathbf{x}$, the vector $\bar{\mathbf{x}}$ is the vector with all elements of $\mathbf{x}$ replaced by their conjugate. The same notation will be used for matrices.

For a function f, the notation $\boldsymbol{f}$ will be used to denote the signal obtained by applying the function on the full time domain (for example, if the time domain is $0, 1, \ldots, N-1$, then $\boldsymbol{f}$ is the vector $(f(0), f(1), \ldots, f(N-1))^T$).

When performing a prediction or estimation, the estimated value will be written with a hat. For example, $\hat{\mathbf{x}}$ is an estimation of the signal $\mathbf{x}$ and $\hat{w}$ is an estimation of the scalar w. For a random signal X, the notation $\hat{X}$ will be associated to the estimator.

When no precision is given, indexing starts at 0 (for matrices or vectors).

For talking about convolution kernels, we will always mention the size of the kernels as $K_1 \times K_2$, even when the networks will be applied with only 1D data. When using 1D data, we will use $K_1 = 1$ (i.e., we will consider that the signals are 2D data with a first dimension of size 1 and the entire signal contained along the second dimension).

Introduction and context : the IMOTEP project

Presentation of the context

In recent years, concerns about global warming have raised attention on the need to reduce carbon dioxide emissions. Multiple solutions are studied for both limiting the impact of the energy production, through renewable energies for example, or reducing our consumption, i.e., reusing the materials of old objects instead of using new materials, recycling, expanding lifetime of objects, ... In a report published in February 2019 [1] and based on data from the International Energy Agency, Saeed Solaymani claimed that almost a quarter of the global emissions were due to the transport sector, and that the road transport itself were responsible for 17.94% of CO_2 emissions. These statistics show how important it is to reduce the emissions of road transport. The report of Solaymani proposes various solutions, changing from one country to another, including a better access to public transport or use of renewable energy. This second approach can be linked to a now well known solution: using electric cars instead of cars using internal combustion engines. However, this solution has its own disadvantages and still requires a clean way to produce a large quantity of electricity and would require a huge amount of battery components such as lithium. Therefore, improvements of internal combustion engines are still a major issue. In such engines, a large part of the consumption is due to friction losses. In 2012, [2] estimated that compensation of friction losses represents one third of the total fuel consumption. They also assumed that it would be possible to save up to 65% of these losses. In 2019, [3] estimated that the losses were still responsible of around one fifth of the consumption, and that it would still be possible to save up to 40% of this energy. As a comparison with electric motors, they pointed out that the energy efficiency of internal combustion engines was of 21.5% (which means that only 21.5% of the total energy consumption is used for moving the car, the rest being due to thermal and friction losses), against 77% for electric motors.

Within this context, the IMOTEP (Innovation MOTEur Propre) project aims to improve both surface finishing and lubricants to optimize friction in internal combustion engines. To do

so, it is first required to properly analyze the friction losses in the engine. For that, sensors, in our case piezoelectric sensors, are used to acquire the friction efforts at some parts of the engine. However, due to defaults of the sensors, vibrations of the mechanisms and random phenomena, the acquired signal is highly polluted by what is called a noise, and it is therefore extremely hard to perform a direct analysis. For making the analysis easier, a common solution is to denoise the signal, which means that a preprocessing is performed to remove, or at least reduce, the pollutions from the signal.

This thesis aims to exploit some of the properties of the friction signals in order to improve existing denoising methods. For validating the developed methods, we will also use other kinds of signals having similar properties. As it will be further explained in the next sections, the main properties of these signals can be summarized as :

- the signals are periodic, i.e. they contain repetition of the same pattern over time. Actually, due to the noise and other random phenomena, it would be more rigorous to speak of cyclostationary signals. Cyclostationarity and associated notions are detailed in Appendix A;

- the signals contain some areas with a fast evolution of the observed value, which could even be seen as discontinuities;

- the signals contain some areas where the value evolves slowly, or even remains close to a constant.

Our work consists in proposing new methods exploiting these three properties and improving robustness to specific areas, especially the discontinuities, which are generally hard to process efficiently in denoising tasks.

Processed signals

Viscous friction signals

Real friction signals used in this study are provided by IREIS (HEF group)[1], and originate from two benches. The most realistic of these two benches is a piston-skirt segment (PSS) tribometer. In this bench, a piston slides into a cylinder (*skirt*) covered with lubricants, or in our case motor oils, resulting in viscous friction. A view of such a system can be seen in Figure 0.1.

[1] `http://www.ireis.fr/en/`

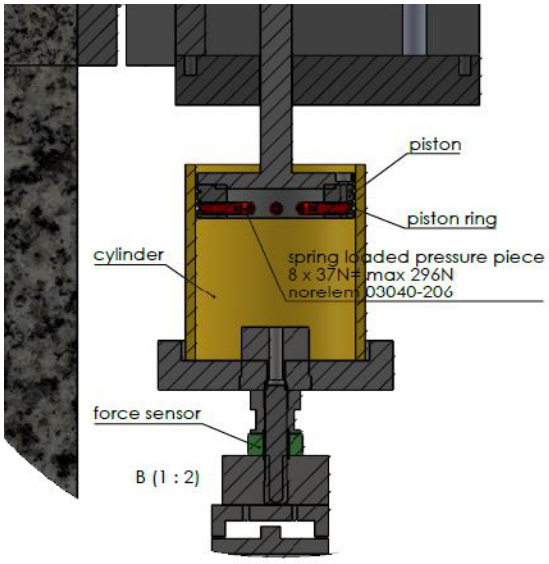


Figure 0.1: Representation of a PSS tribometer (illustration provided by IREIS)

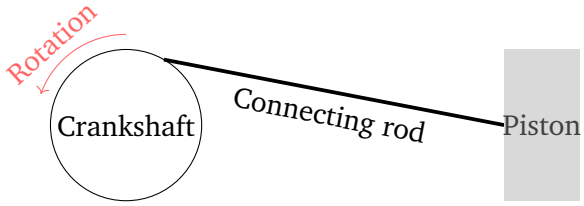


Figure 0.2: Illustration of the system for generating the motion of the piston

The motion of the piston is obtained through a crankshaft system. As illustrated in Figure 0.2, the kinetic energy of the crankshaft is transfered to the piston through a connecting rod.

Note that the movement of the piston can be done both forward and backward along the same direction, resulting in a reciprocating linear motion. Depending on the rotation speed of the crankshaft, the speed of the piston can vary a lot. Because of this, the system can be in three different viscous friction states. Assuming that the viscosity and load remain the same, these states depend on the speed :

- **The boundary lubrication** is obtained at low speeds, when the piston is in contact with the cylinder;
- **The mixed lubrication** is an intermediate state. The contact of the piston with the cylinder is reduced, which implies a reduction of the friction efforts;
- **The hydrodynamic lubrication**. There are no more friction efforts due to the contact with the cylinder, but the friction efforts due to the lubricant start increasing.

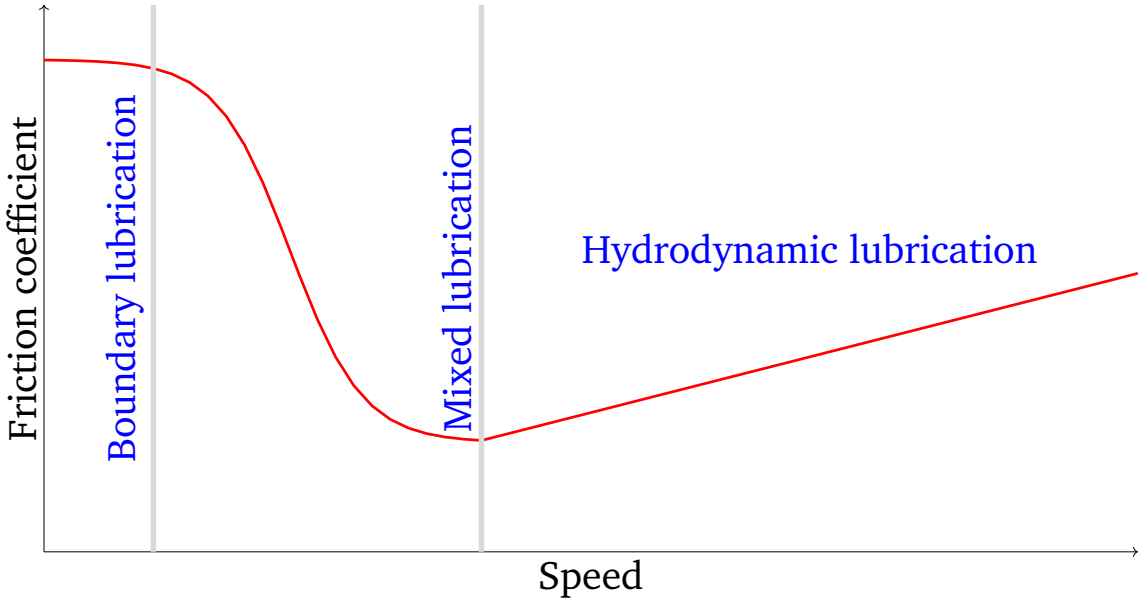


Figure 0.3: Schematic representation of the Stribeck curve, displaying the different lubrication states of a viscous contact

These three states result in the "Stribeck curve" [13, 14], named after Richard Stribeck, who was the first to mention these phenomena [15, 16]. A schematic representation of this curve is shown in Figure 0.3.

This system is close to some parts of internal combustion engines and is therefore ideal for the study. Unfortunately, it was unavailable for the main part of the thesis. In order to perform some analysis despite the unavailability of this system, a simpler bench was built to approximate a similar behavior.

This second system also uses a crankshaft system where the crankshaft is an eccentric to generate friction between a plan against a cylinder. However the contact is now at the external part of the cylinder instead of being at the internal part. As we are studying viscous friction, lubricant is added on the plan. The obtained mechanism, a representation of which is given in Figure 0.4, has a similar behavior as the PSS tribometer, as the motion is still a linear reciprocating motion. However, the speeds reached by this system are far below the ones of the PSS tribometer, which should reach speeds up to 6000 rpm, while this new system is limited to 750 rounds per minute. To obtain similar viscous friction states as with the PSS tribometer, other lubricants are used to change the viscosity parameters.

Figure 0.5 shows one of the acquired signals with a low rotation speed (120 rounds per minute). The periodicity is clearly observable in the obtained signal, as well as discontinuities. In these signals, the discontinuities appear when the motion changes its direction, changing the sign of the friction force and therefore of the friction coefficient. Here, the noise level remains quite low and it is likely that even some basic denoising methods, such

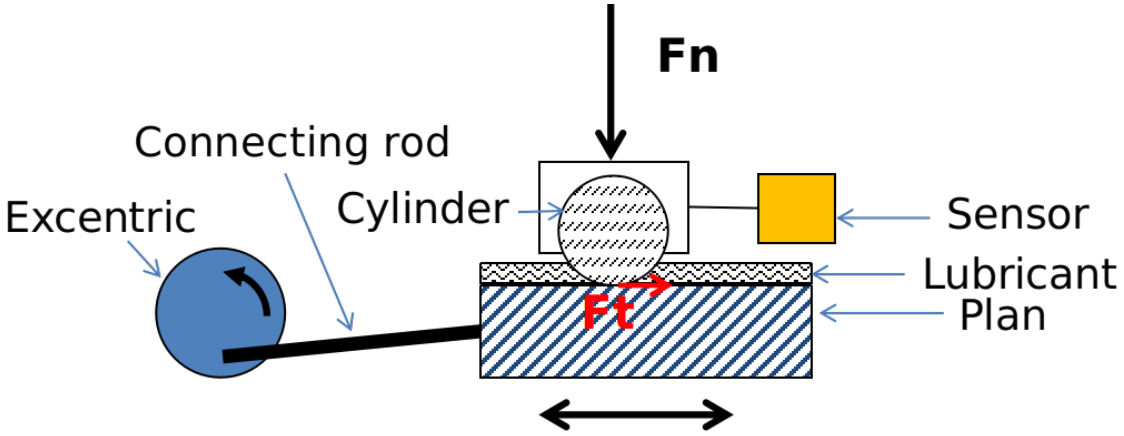


Figure 0.4: Schematic representation simulating the behavior of the PSS (illustration provided by IREIS)

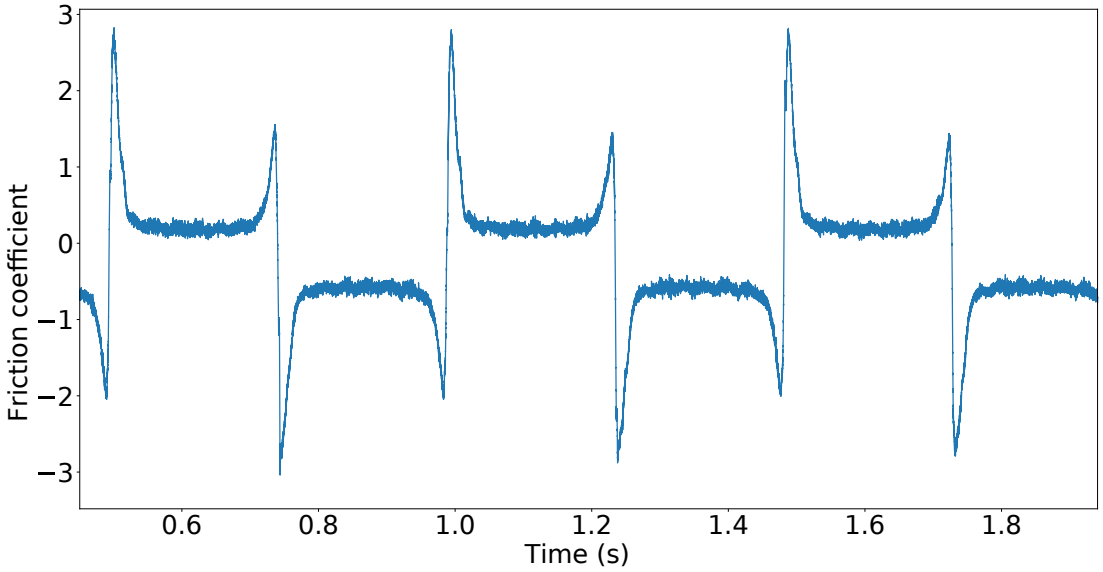


Figure 0.5: Example of a friction signal acquired by IREIS, with a low rotation speed (120 rpm)

as a moving average, could obtain satisfactory results, with the only problem of preserving the discontinuities.

Figure 0.6 shows another signal, obtained with a rotation speed of 750 rounds per minute. While the periodicity is still easily observable, the noise level is now much higher than in the previous signal (here, it seems that one of the origins of this noise is the instability of the bench, which results in vibrations). In this case, it becomes necessary to use more robust methods to perform a satisfactory denoising.

For the two tribological systems, recordings are made with a fixed rotation speed setpoint for the crankshaft, which explains this periodic movement of the plan (or the piston in the case of the PSS tribometer), except for some random variations. This motion will be further

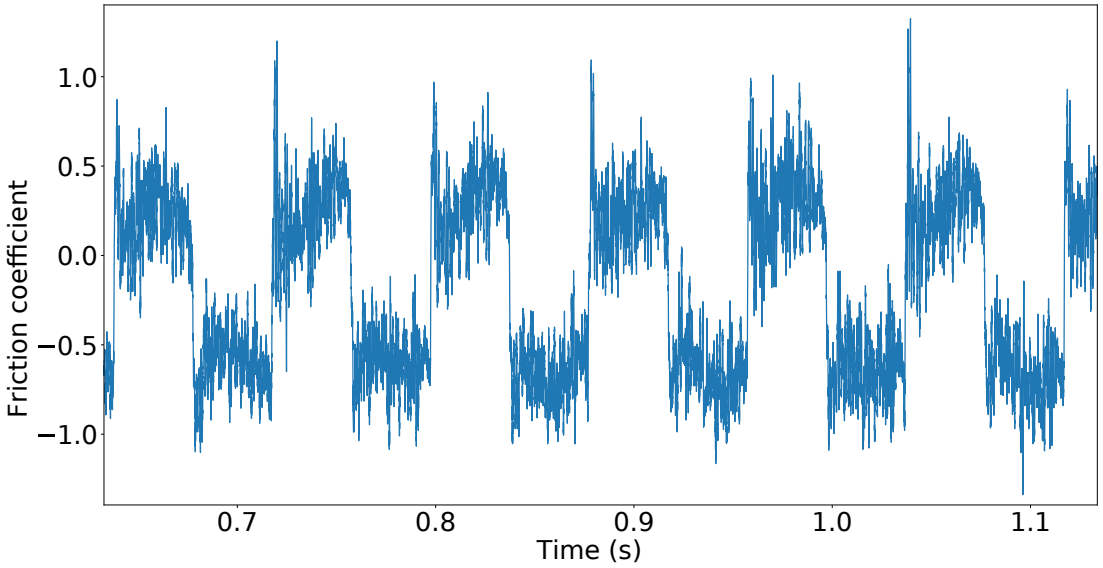


Figure 0.6: Example of a friction signal acquired by IREIS, with a high rotation speed (750 rpm)

studied in Section 2.2, and we will see that it can be approximated with a sine wave. The signals of the position of the plan during the recording are also provided by IREIS.

Electrocardiograms (ECG)

As we are aiming at methods that can be generalized to several kinds of signals, we will apply some of the developed methods on electrocardiograms (ECG). ECG represent the electric activity of the heart and are acquired through electrodes located on standard positions of the body surface. The electric activity that is acquired is linked to the cardiac rhythm and will therefore contain some periodic patterns. As for friction signals, it is not rigorous to speak of periodicity and the term of cyclostationarity would be more adequate due to the noise. Additionally, contrary to the friction signals where the fundamental frequency remained close to the setpoint frequency, implying limited variations around the expected frequency, the heart rhythm can have some large variations. This is especially true outside of effort, where each beat might have a different length, while the frequency tends to stabilize when doing an effort. This phenomenon is called Heart Rate Variability.

Figure 0.7 shows an electrocardiogram. This one was generated via a synthetic model obtained with the same software as in [17], which allowed both obtaining a stable fundamental frequency and the absence of noise. In real signals, such observations would not be possible and even the most advanced denoising techniques cannot completely remove the noise. This synthetic signal has the advantage of well illustrating the kind of phenomena appearing in ECG signals. As for friction signals, a periodic pattern appears as well as

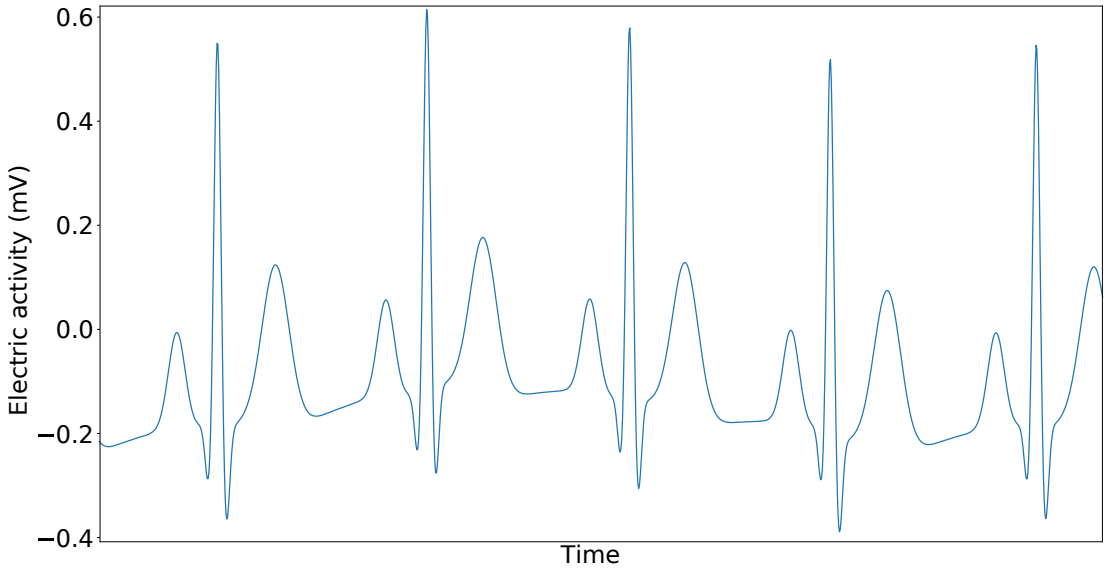


Figure 0.7: Representation of a synthetic ECG signal

some fast variations, even though, contrary to friction signals, those fast variations are not discontinuities.

We can also observe more precisely the period of an ECG (Figure 0.8), which lets us distinguish five local extrema. These local extrema, usually called P, Q, R, S and T, are representative of an ECG signal. Their presence or absence, their magnitude, their temporal regularity, their separating intervals, etc., . . . can provide some information about the health of the patient.

The interest of studying ECG signals is also the existence of ECG databases [12, 18] containing real clean (or almost) ECG signals. An example of such "clean" ECG signal is shown in Figure 0.9 and clearly shows that there are still some pollutions in the signal.

Some databases [19] also provide realistic noise signals that can appear in such signals. Three examples of typical noises are given in Figure 0.10. One of the remarkable properties of these noise signals, which might also imply some difficulties in denoising real ECG signals, is that they sometimes tend to have long time-effects (they have low frequencies components), and their effects might cover several periods of the ECG signal.

Having these databases of real signals will be extremely helpful for assessing the performance of our methods on real data with a fair idea of the performance and the possible limitations.

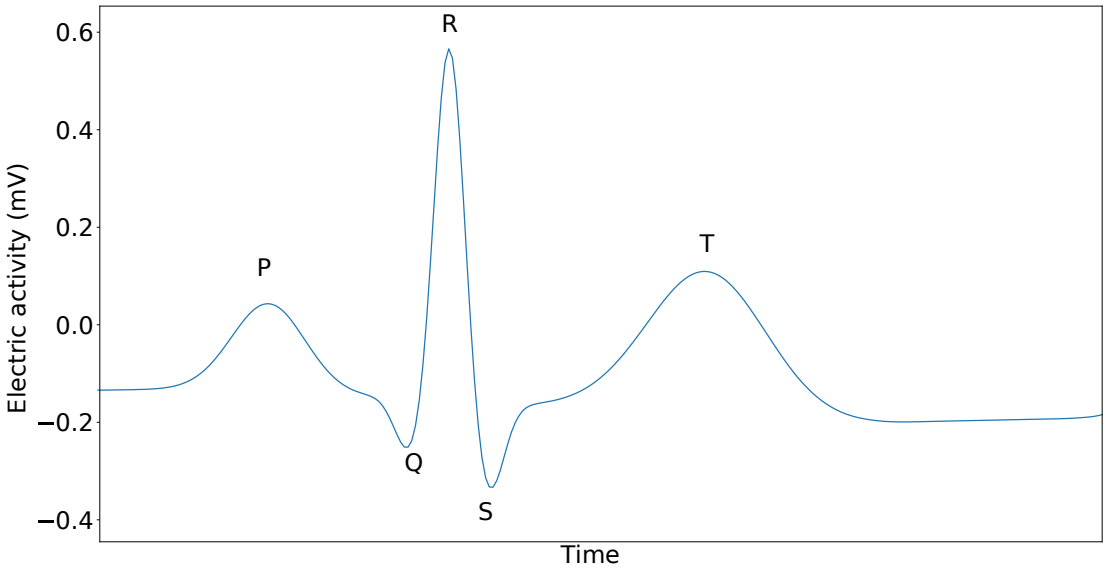


Figure 0.8: One period of an ECG signal, illustrating the typical aspect of the PQRST points

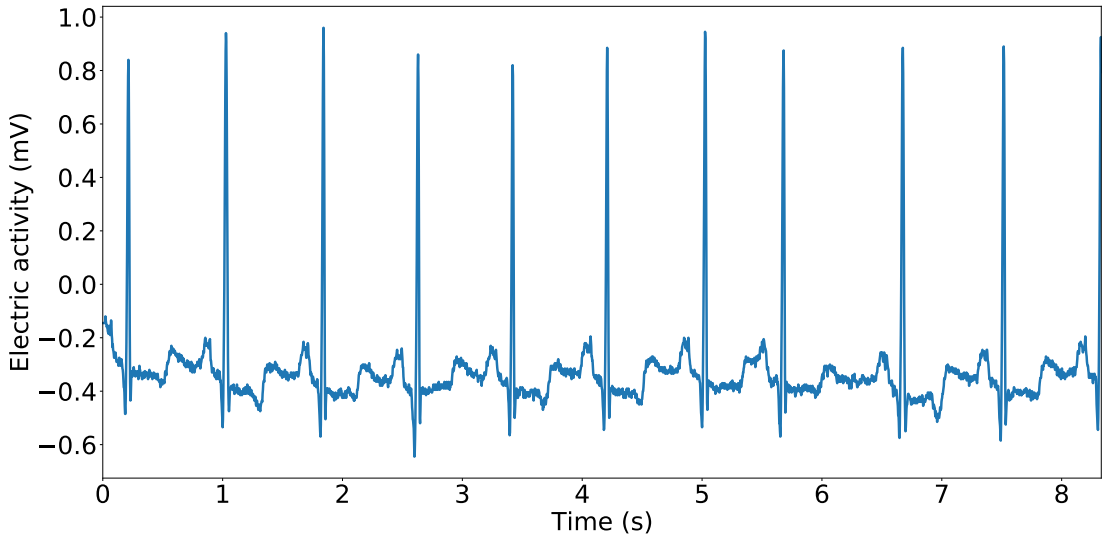


Figure 0.9: Real ECG signal (from [12])

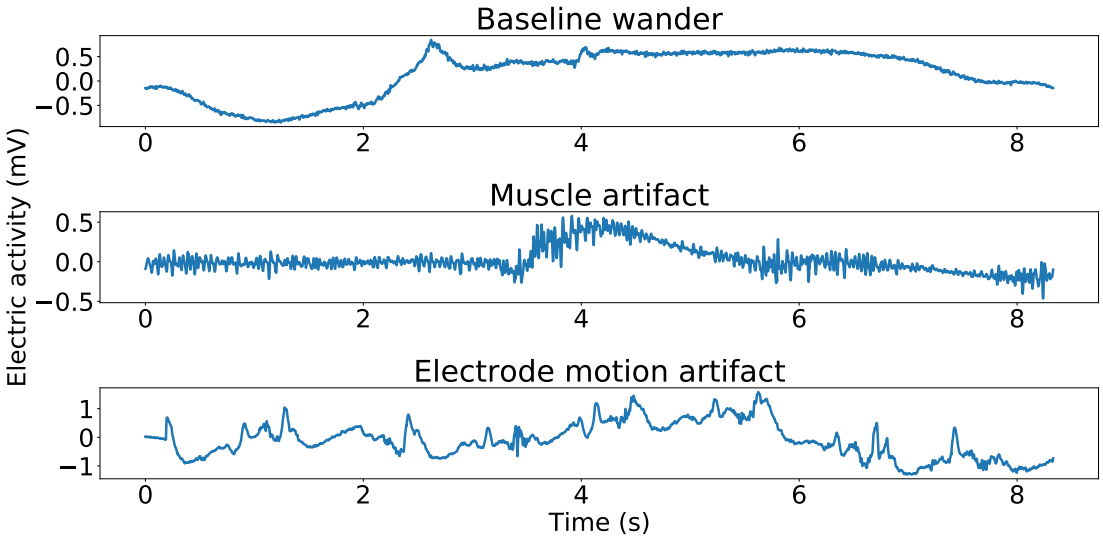


Figure 0.10: Typical noise signals in ECG records (from [19])

FIB/SEM sequences

Focused Ion Beam (FIB) and Scanning Electron Microscope (SEM) images are obtained by scanning a surface with a beam of ions (resp. electrons). The collisions of the particles with the matter produce various signals that can provide useful information about the matter. More details about the methodology employed for acquiring these images and some possible applications can be found in [20, 21] for FIB images and in [22] for SEM images.

In our case, we performed some applications on FIB images registered during the processing of some semiconductors. As the semiconductors are being processed, some textures corresponding to a certain physical composition appear. One of the images is shown in Figure 0.11, with modified colors for enhancing the contrast. Only the central part is being processed. Here, it is quite clear that directly exploiting such an image will not be easy as it contains a lot of pollutions. However, it is already possible to distinguish some of the patterns appearing, with some squares in the central part, separated by a constant distance. Here again, this constant distance can be assimilated to a (2D) periodicity.

Common property of these signals : cyclostationarity

As explained previously, the studied signals all have periodic properties. We can write it properly by saying that for a signal $\mathbf{y}$, there is a value $T > 0$, called the "period", verifying :

$$\forall n \in \mathbb{Z}, \mathbf{y}_{n+T} = \mathbf{y}_n. \tag{0.2}$$

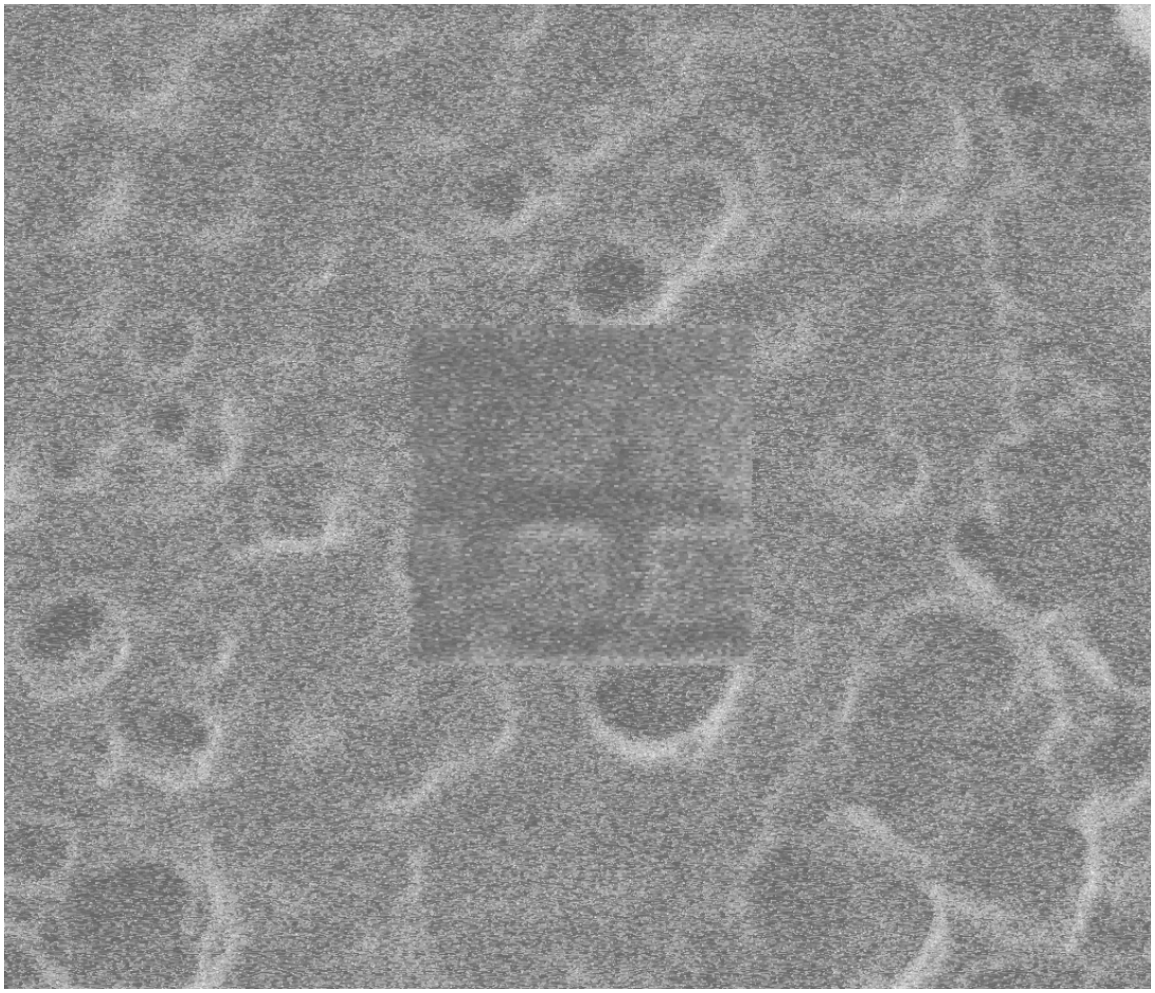


Figure 0.11: An image acquired via Focused Ion Beam microscopy.
Only the central square is being processed during the recording.

In real cases, a signal $\mathbf{x}$ is actually the sum of the clean and deterministic signal $\mathbf{y}$ and a noisy signal $\mathbf{e}$. Even if we assume that the clean signal is exactly periodic, it is clear that making such an assumption is not possible concerning the noisy part. As the noise is a random phenomenon, it is possible to consider the signal $\mathbf{e}$ as a realization of the random signal N. Then, the signal $\mathbf{x}$ is the realization of a random signal X:

$$\forall n \in \mathbb{Z}, X_n = \mathbf{y}_n + N_n. \tag{0.3}$$

It is then possible to compute the probabilistic moments of the random variables composing this signal. For example, with the expectation, i.e., the first order moment, we obtain :

$$\forall n \in \mathbb{Z}, \mathbb{E}\left[X_n\right] = \mathbf{y}_n + \mathbb{E}\left[N_n\right]. \tag{0.4}$$

If we assume that the noise is centered, i.e. that its average value is zero, then we obtain:

$$\forall n \in \mathbb{Z}, \mathbb{E}\left[X_n\right] = \mathbf{y}_n. \tag{0.5}$$

If the signal $\mathbf{y}$ is periodic, the signal X will have a periodic expectation. Random signals having a periodic expected value are called first-order wide-sense cyclostationary signals. Note that having a centered noise added to a periodic deterministic signal is a sufficient condition for obtaining this property, but it is not a necessary condition. In our case, first-order cyclostationarity will be the most studied cyclostationarity. However, we will also study partially second-order cyclostationarity in the wide sense. The definition of a wide-sense cyclostationary signals up to second order can be formulated with the existence of a period T such as :

$$\begin{cases} \forall n \in \mathbb{Z}, \mathbb{E}\left[X_{n+T}\right] = \mathbb{E}\left[X_n\right] \\ \forall n \in \mathbb{Z}, \forall \tau, \mathbb{E}\left[X_n X_{n+\tau}\right] = \mathbb{E}\left[X_{n+T} X_{n+T+\tau}\right]. \end{cases} \tag{0.6}$$

For such a signal, we say that the signal contains the cyclic frequency $\frac{1}{T}$. Note that the cyclic frequency is not unique. Indeed, if $T > 0$ verifies Equation (0.6) conditions, then any multiple of T will also verify these conditions. It is also possible that some signals have two different cyclic periods, without one of them being an integer multiple of the other. An easy example of such signal would be a constant signal, where any value of T would verify Equation (0.6) conditions.

For real signals, it is also impossible in most cases to obtain well-defined cyclostationarity. It would therefore be more rigorous to extend the defined properties to the notions of almost-cyclostationarity. In [23], further details about cyclostationarity, including various orders or

notions of almost-cyclostationarity are given. For this reason, the methods we will propose throughout this thesis should be able to adapt to small variations of the periodicity.

Outline of this thesis

This thesis is organized in six chapters.

In Chapter 1, we furnish the required background regarding existing methodologies for denoising, especially deep learning based methodologies. In this chapter, knowledge of basic concepts of signal processing and machine learning are required. The Appendix A provides some of these basic concepts for signal processing, and the Appendix B provides some of the basic concepts for Machine Learning.

In Chapter 2, we propose an analysis of the friction signals to confirm the assumptions about periodicity. We also propose a model fitting approach to perform a first denoising on friction signals.

In Chapter 3, we propose a first deep learning based approach for denoising signals in our case. We compensate the lack of labeled data by training with synthetic data. We also use regularization to ensure generalization to various kinds of noise. The proposed models are designed to ensure robustness to various settings rather than being trained for a specific setting.

In Chapter 4, we propose two methods for considering the periodicity within deep neural networks. Both methods rely on a reshaping of the data. One of the proposed methods consists on applying a preprocessing on the signals then using the same models as in Chapter 3. In the other method, we leverage the neural networks to directly exploit the new shape of the data.

In Chapter 5, we work on improving the robustness of the proposed methods to various contexts. For that, we propose to improve the variety of the generated training data. We also propose an ensemble combination of methods.

Lastly, Chapter 6 concludes this thesis with possible future works.

List of publications and presentations

International conferences papers

[24] J. Rio et al. “WaveNet based architectures for denoising periodic discontinuous signals and application to friction signals”. In: *2020 EUSIPCO*. IEEE. 2021, pp. 1580–1584
[25] J. Rio et al. “Leveraging end-to-end denoisers for denoising periodic signals”. In: *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE. 2021

Journal paper (to be submitted)

[26] J. Rio et al. “Towards a unified framework for denoising periodic signals”. In: *To be defined* (To be submitted in 2022)

Presentation in international conference

[27] J. Rio et al. “Wavenet-Based Architectures Adapted to the Denoising of FIB/SEM Image Sequences”. In: *2021 IEEE 17th International Conference on Automation Science and Engineering*. IEEE. 2021

1 Background : The task of denoising

1.1 Noise in the signal

During the analysis of the signal, several phenomenons have to be considered. While some parts will be relevant for the desired application, some others might be unwanted and might make the analysis or exploitation harder. These unwanted parts of the signal are called noise. Noise can have several origins.

The first origin of noise occurs even before the acquisition, it is directly a part of the original signal. It might be due to vibrations of the mechanisms in friction signals, surrounding conversations in audio signals, motions of the patient in ECG signal, ... This noise is generally an additive noise, in the sense that it does not modify the signal of interest but is added to it. The result of the presence of additive noise is shown in Figure 1.1. It clearly shows that the phenomena having a low magnitude, in this case the oscillations between the discontinuities, become much harder to observe with the presence of noise.

The other origins of noise are linked to the acquisition. The first problem appearing during the acquisition is due to the conversion from an analog signal to a digital signal (A-to-D conversion). There are two reasons for this. The first one is that the acquisition will be done with a certain temporal resolution (the sampling period); instead of continuously acquiring the signal, two acquired samples will be separated by a time T_e. The resulting sample will generally be dependent on the value at the time it was acquired, but also of all values that occurred since the last acquired sample. Additionally to this temporal resolution, digital acquisitions also imply a resolution for the possible values for one sample. Every digital tool has a fixed amount of possible values linked to the amount of bits used for coding one value, which implies that each acquired sample will be registered with one of these values, generally the one that is the closest to the real value, instead of its real value. An illustration of an acquisition of a sine wave is shown in Figure 1.2. The resulting noise is called quantization noise.

Lastly, electronic devices, such as sensors, tend to modify the shape of the acquired signal. When acquiring a sample at a certain time, previous samples generally have an impact to the output value, which might result in a different value and in a delay before obtaining the correct value. An illustration of these phenomena is shown in Figure 1.3. Here, we took a

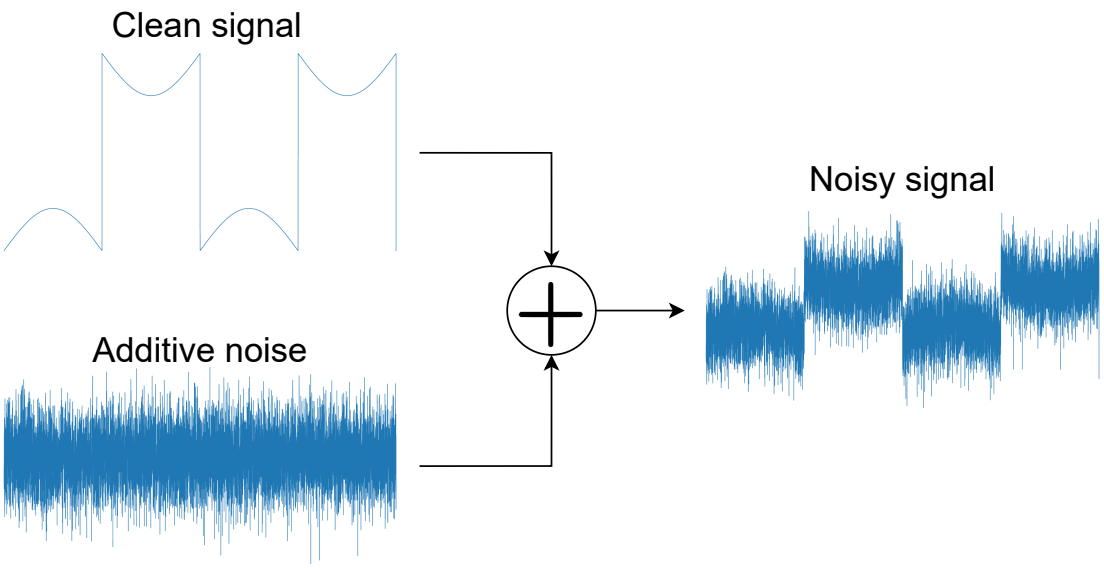


Figure 1.1: Representation of an additive noise added to a signal of interest

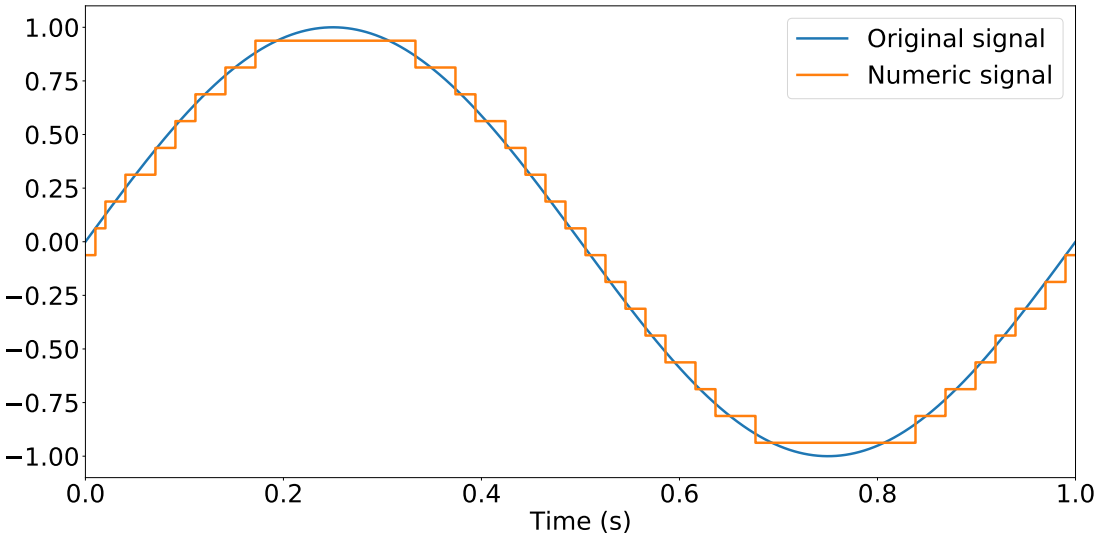


Figure 1.2: Illustration of the loss of information due to A-to-D conversion

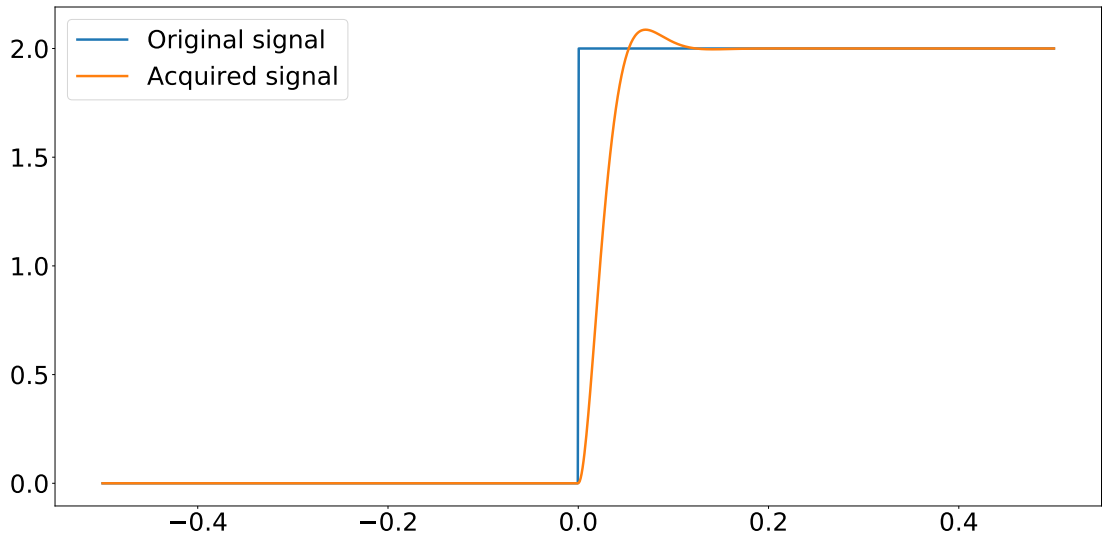


Figure 1.3: Illustration of the delay and modification due to the sensor

simple step function on which we applied a second order low-pass Butterworth filter [28]. This example clearly emphasizes the delay and the kind of oscillations that can be obtained due to the sensor acquisition.

Generally, for an input signal $\mathbf{u}$, the output $\mathbf{y}$ of the sensor verifies:

$$\forall n \in \mathbb{Z}, \mathbf{y}_n + \sum_{k=1}^{N} a_k \mathbf{y}_{n-k} = \sum_{k=0}^{M} b_k \mathbf{u}_{n-k} \tag{1.1}$$

with $a_1, \ldots, a_N, b_0, \ldots, b_M$ some coefficients. The resulting noise is therefore generally denoted as a convolutional noise. With finite acquisitions (we suppose that the values are null before and after the acquisition), we are interested only in samples from 0 to n_0 and we have :

$$\begin{aligned} &\mathbf{y}_0 = b_0 \mathbf{u}_0 \\ &\mathbf{y}_1 = b_0 \mathbf{u}_1 + b_1 \mathbf{u}_0 - a_1 \mathbf{y}_0 = b_0 \mathbf{u}_1 + (b_1 - a_1 b_0) \mathbf{u}_0 \\ &\ldots \end{aligned} \tag{1.2}$$

As a result, we have the relation :

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_0 \\ \vdots \\ \mathbf{y}_{n_0} \end{pmatrix} = \begin{pmatrix} b_0 & 0 & \ldots & \ldots & 0 \\ b_1 - a_1 b_0 & b_0 & \ldots & 0 & \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \mathbf{u}_0 \\ \vdots \\ \mathbf{u}_{n_0} \end{pmatrix} = \mathbf{H}\mathbf{u} \tag{1.3}$$

where $\mathbf{H}$ will be a lower triangular matrix with all elements of its diagonal equal to $\mathbf{b}_0$. The convolutional noise from the sensor can then be considered as an additive noise $\mathbf{u} - \mathbf{H}\mathbf{u} = (\mathbf{I} - \mathbf{H})\mathbf{u}$, even though its processing would generally be different, as additive

noise is generally random whereas convolutional noise will be deterministic. Let us notice that we could consider the noise due to the A-to-D conversion as an additive noise with the same logic.

In real acquisitions, all of these sources of noise contribute simultaneously, which also implies that not only the signal, but also the additive noise is being affected by the A-to-D conversion and the sensor transfer function. For our signals, we will assume that the temporal resolution, as well as the amount of bits used will be sufficient for neglecting the effects of the A-to-D conversion. Following our previous remarks, in most of our experiments, we will process all noise as being an additive noises.

Let us notice that when considering a signal with its strict definition, i.e., an application of $\mathbb{R}^N$ to $\mathbb{R}^M$, it is possible to extend all elements mentioned here to images or videos. While we focus mostly on 1D signal denoising, the applications on FIB and SEM image sequences are a 3D application (the two dimensions of the gray-scale image and the temporal dimension). Additionally, the methods developed in the last years (mostly methods based on Deep Learning) have largely benefited of studies for image denoising, which could then be adapted to 1D signal denoising.

1.2 Presentation of the task

As the analysis or exploitation of a signal is made harder by the presence of noise, numerous tasks might benefit from a preprocessing of the signal, consisting of removing the noise. In most realistic cases, completely removing the noise is not possible, and the preprocessing consists in reducing it sufficiently so that it does not affect the analysis anymore. This reduction of the noise is called denoising. The next sections will focus on detailing source separation, another task close to denoising which has generated a large number of studies in the recent years, and recall some of the methods that have been proposed for these two tasks.

1.3 An extension : Source separation

Source separation is a task close to denoising, where the objective is not to remove a part of the signal, but to consider that the signal contains S sources of interest :

$$\mathbf{y} = \sum_{i=1}^{S} \mathbf{s}^{(i)} \tag{1.4}$$

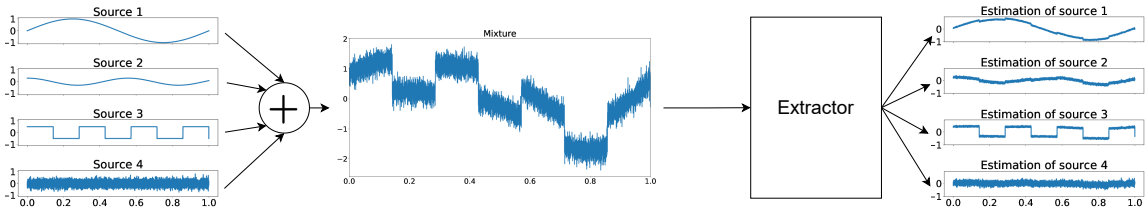


Figure 1.4: Illustration of source separation with a single model

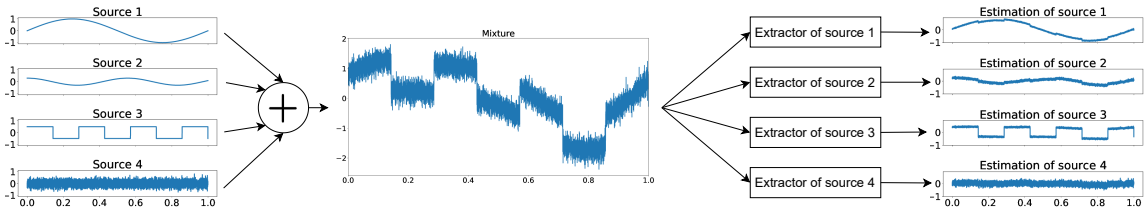


Figure 1.5: Illustration of source separation with one model per source

with $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(S)}$ the sources of interest. The objective of the task is then to isolate each contribution $\mathbf{s}^{(i)}$, that is, instead of returning one signal, the task should output as many signals as sources in the mixture signal $\mathbf{y}$. Audio source separation [29] is a very illustrative example of source separation, with two major applications :

Music source separation The objective is to isolate the contribution of each instrument (including voice);

The cocktail party problem Several conversations occur at the same time and are recorded in a single signal. The objective of source separation is then to isolate each conversation or each speaker.

Here, the relation with denoising is quite clear. Denoising can be seen as a specific case of source separation where the signal contains only two sources : the source of interest and the noise. Additionally, while some methods try to design a single model for extracting all sources at the same time (process illustrated in Figure 1.4), other methods will rather design one specific model for each source (Figure 1.5). As explained in Section 1.1, the definition of noise can vary from one application to another. Therefore, by considering the source of interest $\mathbf{s}^{(1)}$ as the source of interest and the sum of the other sources as the noise, it is possible to extract the source in a similar way as if we were performing denoising. It is then possible to extract other sources in a similar way. Therefore, a lot of methods designed for denoising can be adapted for source separation, and most interestingly in our case, methods designed for source separation should be easily adaptable for denoising.

As for denoising, source separation will never completely isolate one source and other sources will still have a contribution on each source estimation. This is the reason why the discontinuities of source 3 or white noise of source 4 are still present in the estimation of

sources 1 and 2 in Figure 1.4 and Figure 1.5. Note that these illustrations are not a real source separation result, they are a generated example of the kind of phenomena that would occur when performing source separation.

1.4 Review of the literature : conventional techniques

In this section, we recall the main techniques used for denoising and source separation outside of Deep Learning based approaches.

1.4.1 Filtering techniques

One solution for removing the noise of a signal is to design a filter, i.e., a process that removes unwanted components from a signal. This definition could apply to most techniques that we will describe in this review of the state-of-the-art. Here, we introduce some techniques that we will not further develop afterwards. We should first recall some of the definitions for filters.

A filter is said to be linear if its response $\mathbf{y}$ to an input $\mathbf{x} = \lambda\mathbf{x}^{(1)} + \mathbf{x}^{(2)}$ verifies:

$$\mathbf{y} = \lambda\mathbf{y}^{(1)} + \mathbf{y}^{(2)} \tag{1.5}$$

with $\mathbf{y}^{(1)}$ (resp. $\mathbf{y}^{(2)}$) the response to the system when the input is $\mathbf{x}^{(1)}$ (resp. $\mathbf{x}^{(2)}$) and λ a scalar, and this independently of λ, $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$.

A filter is said to be time-invariant if a shift of Δ in the input signal results in a shift of Δ in the response, i.e. that if $\mathbf{y}$ is the response to $\mathbf{x}$ and if $\mathbf{x}_{\Delta,\cdot}$ is the signal defined as:

$$\forall n \in \mathbb{Z}, \mathbf{x}_{\Delta,n} = \mathbf{x}_{n-\Delta} \tag{1.6}$$

then the system is said to be time-invariant if its output $\mathbf{y}_{\Delta,\cdot}$ to $\mathbf{x}_{\Delta,\cdot}$ verifies:

$$\mathbf{y}_{\Delta,\cdot} = \mathbf{y}_{n-\Delta}. \tag{1.7}$$

A system is said to be causal if it depends only on the past, anti-causal if it depends only on the future and non-causal if it depends on both the past and the future.

The impulse response of a system is defined as the response to an impulsion defined as:

$$\delta : n \to \delta_n = \begin{cases} 1 \text{ if } n = 0 \\ 0 \text{ otherwise.} \end{cases} \tag{1.8}$$

A system is said to be a Finite Impulse Response (FIR) system if the temporal support of its impulse response is finite, and an Infinite Impulse Response (IIR) otherwise.

The easiest to define filters are linear, time-invariant filters (LTI). For an input $\mathbf{x}$, their response $\mathbf{y}$ can be defined as:

$$\mathbf{y}_n = \sum_{k=1}^{N} \mathbf{a}_k \mathbf{y}_{n-k} + \sum_{l=0}^{M} \mathbf{b}_l \mathbf{x}_{n-l} \tag{1.9}$$

with $\mathbf{a}$ and $\mathbf{b}$ two vectors of coefficients. This definition corresponds to causal systems, it would be possible to change the indexes of the second sum to obtain different systems. If $N = 0$, then the system is a FIR system, otherwise, it is an IIR system. These systems, despite being simple to define, can potentially obtain extremely various frequency behaviors. A simple moving average filter ($N = 0$, $\forall 0 \leq l \leq M, \mathbf{b}_l = \frac{1}{M+1}$) will result in a low-pass filter that is efficient for removing high-frequency components. It has for example been used for denoising in [30]. However, LTI filters will not differentiate the noise and the signal of interest if they both have components in the stop-band of the filter.

Among more robust linear filtering methods, the Kalman filter [31] and its extensions have obtained large success and have been applied to various kinds of denoising [32, 33, 34, 35].

1.4.2 Wavelet transform

Wavelet transform is similar to the Fourier transform in the sense that its purpose is to represent a function as a sum of predefined components. From this first definition, Fourier transform could actually be seen as a special case of Wavelet transform.

For performing a wavelet transform, the first step is to define a "mother" wavelet, i.e., a function $\psi : t \to \psi(t)$ that will be used for creating a family of wavelets.

While it would technically be possible to use almost any oscillating function as a mother wavelet (as long as the infinite sum is convergent), the wavelet transform generally denotes cases with a mother wavelet equal to zero a.e.. Additionally, mother wavelets are generally centered ($\int_{-\infty}^{\infty} \psi(t)dt = 0$) and are normalized ($\int_{-\infty}^{\infty} |\psi(t)|^2 \, dt = 1$).

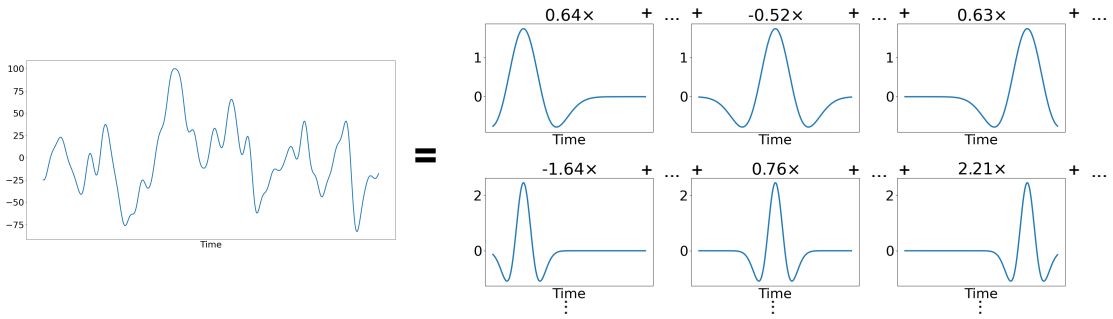


Figure 1.6: Example of a wavelet decomposition

Each wavelet is obtained by applying a scale a and a shift b to the mother wavelet:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right). \tag{1.10}$$

As it is not possible to compute an infinite amount of values, the parameter a can generally take values defined as 2^p and $b = ka\tau$ [36, 37], with τ a constant, generally 1, defining the resolution and k an integer, so the wavelets can be redefined with:

$$\psi_{p,k}(t) = \frac{1}{\sqrt{2^p}}\psi\left(\frac{t-k\tau}{2^p}\right). \tag{1.11}$$

A discrete temporal signal $\mathbf{x}$ will then be obtained as:

$$\mathbf{x} = \sum_{(p,k)\in\mathbb{Z}^2} \mathbf{A}_{p,k}\boldsymbol{\psi_{p,k}} = \sum_{(p,k)\in\mathbb{Z}^2} \left(\sum_{n=-\infty}^{\infty} \mathbf{x}_n \overline{\psi_{p,k}(nT_e)}\right)\boldsymbol{\psi_{p,k}} \tag{1.12}$$

with $\mathbf{A}_{p,k} = \langle \mathbf{x}, \boldsymbol{\psi_{p,k}} \rangle$ the wavelet coefficients. Let us notice that this equation corresponds to the case of the orthogonal wavelets. A more general approach using biorthogonal wavelets would allow several multiresolution analyses for the same signal.

Let us also notice, that in practice, it is not possible to compute the infinite amount of coefficients that would be required in Equation (1.12). With a finite acquisition, it is possible to retrieve all the information with a finite amount of coefficients, but it is also possible to chose an amount of scales to observe and use a complementary function, called the scaling function, to represent the information contained by other scales [37]. The scaling function is also used to design a filter bank to compute the coefficients.

An example of Wavelet transform is provided in Figure 1.6.

Wavelets can be quite convenient for denoising. Generally, components associated to the noise will have a small magnitude, and a solution for denoising is then to apply a threshold

to the wavelet transform. It is possible to apply a hard thresholding (components lower than the threshold are removed, other components are unchanged) or a soft thresholding (components below the threshold are removed, other components are reduced in magnitude by the threshold). Many studies have used wavelets for denoising [38, 39, 40, 41, 42, 43]. As for many methods, wavelets can also be applied in images [44, 45, 46]. Generally, wavelets are quite effective for preserving discontinuities but tend to create oscillations around them.

1.4.3 Total variation based methods

For a signal $\mathbf{x}$ of length N, the total variation (TV) is defined as:

$$TV(\mathbf{x}) = \sum_{n=1}^{N-1} |\mathbf{x}_n - \mathbf{x}_{n-1}| \,. \tag{1.13}$$

Generally, clean signals have a low total variation as they contain mostly smooth parts. Therefore, a high total variation is often due to the presence of noise. Total variation can then be used to penalize a signal containing too many variations which would be linked to noise. It is for example possible to extract an approximate of a clean signal $\mathbf{y}$ from a noisy signal by solving the problem:

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \mathbf{x}) + \gamma TV(\mathbf{u}) \tag{1.14}$$

with γ a weight and $\mathcal{L}$ a reconstruction term (for example $\|.\|_2$ or its square) called the data-fidelity term. The TV term is meant to promote piecewise-constant solutions, whereas the data-fidelity term is meant to promote unchanged signals. The balance between the two terms will then promote solutions with small variations outside discontinuities. Using a too high value of γ will remove more contributions of the noise but might also remove some smooth components from the signal of interest. Using a too small value will preserve the signal but might also be ineffective in removing the noise.

Several methods have been proposed to solve this problem. As well as for wavelet denoising, it has been applied to signal denoising [47, 48, 49, 50] and image denoising [47, 51, 52, 53]. While total variation based methods avoid the problem of creating oscillations near discontinuities, they tend to create staircase functions and smooth parts are therefore not well approximated, even when balancing the penalty and data-fidelity term.

To overcome this issue, certain methods proposed combinations of total variation with other approaches, such as wavelets [54, 55], curvelets (an extension of wavelets) [56] or shearlets (another extension of wavelets) [57].

1.4.4 Non-local means

While most methods denoise a sample based on surrounding samples, non-local means based methods consist in looking for some areas (patches) in the signal that are similar to the area around the sample. It is then possible to denoise the sample by using not only the local context, but also all similar patches, which extends the amount of data and reduces the variance of the remaining noise. While it would be ideal to observe all samples, it is not always possible to do so. Some signals might be too long, especially with high sampling frequencies, to be observed entirely in a reasonable time. As a search for relevant patches has to be performed for each sample of the signal, and considering that there are as many patches as samples in the observed area (one patch is generally defined as a ball around a sample), the algorithm would have a quadratic complexity. To limit the complexity, methods used in signal denoising will most of the time focus on a neighborhood, so that the search of patches relevant for one sample can be done in constant time, and the global complexity is linear. Even though it would be possible to use some clustering methods for reducing the complexity, it is likely that longer signals will contain more different patterns. Therefore, more clusters would be needed. Additionally, most time series have mostly short or middle-term correlations (even though theoretical periodic signals might have infinite time correlations, the properties tend to change over time in real signals), and observing the full signal is therefore not relevant anyway.

To find the relevant patches, the first step is to define a distance. Then, a search can be performed for finding the relevant patches for each sample, or a clustering can be used. The advantage of the first method is that it can adapt to the specificities of each sample, but has a high computational cost. Clustering based methods might result in less adaptability as two patches in the same cluster might be dissimilar if they are both far from the centroid, but it reduces the computational cost and the full image can more easily be used. Some methods also start by applying a clustering, then look for the relevant patches for one sample within this sample's cluster [58]. After obtaining the patches based on the similarity metric, the rest of the method consists in combining them to have a "collaborative" denoising. The easiest method would be to perform an average of all samples in the associated patches weighted based on the similarity [59, 60, 61, 62, 63]. Other methods apply a denoising algorithm on the entire group obtained with the patch around the sample and the patches found to be relevant. In image applications, it is for example the case of the BM3D algorithm [64], which is still often used.

An illustration of the general process for non-local means is provided in Figure 1.7.

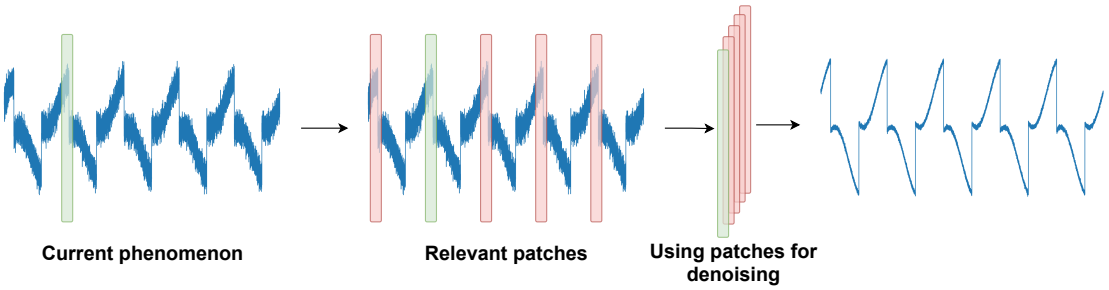


Figure 1.7: Illustration of the process used in non-local means. The green area is denoised by exploiting the similar areas, shown in red.

1.4.5 Non-negative matrix factorization

Nonnegative matrix factorization (NMF) is a method that has largely been used in source separation tasks. Most of the time, it is not applied directly on the raw signal, but on a 2D transformation of this signal, such as a magnitude spectrogram (see Section A.7). Other 2D representations are possible, at the condition that they contain only positive (or null) values.

The objective of the task is to extract each source $\mathbf{s}^{(i)}$ from a mixture $\mathbf{x} = \sum_{i=0}^{S-1} \mathbf{s}^{(i)}$. The short-time Fourier transform of the mixture $\mathbf{x}$ verifies:

$$STFT(\mathbf{x}) = \sum_{i=0}^{S-1} STFT(\mathbf{s}^{(i)}). \tag{1.15}$$

Generally, in such methods, the phase is assumed to be the same for all sources, and as a consequence, for the mixture. Therefore, the STFT of $\mathbf{x}$ can be written as a pointwise multiplication of a magnitude spectrogram $\mathbf{X}$ and a phase spectrogram $\mathbf{\Phi}$. The STFT of each source $\mathbf{s}^{(i)}$ is then the pointwise multiplication of a magnitude spectrogram $\mathbf{S}^{(i)}$ and $\mathbf{\Phi}$, which results in:

$$\mathbf{X}\mathbf{\Phi} = \sum_{i=0}^{S-1} \mathbf{S}^{(i)}\mathbf{\Phi}. \tag{1.16}$$

Therefore, due to the assumption of a shared phase, the linearity of the STFT also implies the linearity of the magnitude spectrogram:

$$\mathbf{X} = \sum_{i=0}^{S-1} \mathbf{S}^{(i)}. \tag{1.17}$$

The objective of NMF is to rewrite the spectrogram $\mathbf{X}$ as the product of a dictionary matrix $\mathbf{W}$ and an activation matrix $\mathbf{H}$. The dictionary matrix is an ensemble of possible phenomena (each column of the matrix $\mathbf{W}$ is one component) that can occur in the signal, while the

activation matrix is meant to decide when, and with which magnitude, each component occurs in the signal.

Generally, the dimensions of $\mathbf{W}$ and $\mathbf{H}$ are such that the total amount of values in $\mathbf{W}$ and $\mathbf{H}$ is less than the amount of values in $\mathbf{X}$, implying a dimensionality reduction. If $\mathbf{X}$ has F lines (F frequency bins) and T columns (T time ranges), then the dimension of $\mathbf{W}$ will be $F \times K$ and the dimension of $\mathbf{H}$ will be $K \times T$, such that $(F + T)K < FT$.

Once the matrices $\mathbf{W}$ and $\mathbf{H}$ are computed, each component has to be associated with one of the source. A factorization of the magnitude spectrogram $\mathbf{S}^{(i)} = \mathbf{W}^{(i)}\mathbf{H}^{(i)}$ can then be obtained by extracting the corresponding columns of $\mathbf{W}$ and lines of $\mathbf{H}$. The STFT of source i is then obtained by multiplying by the phase and the temporal signal is obtained through inverse transform of the STFT.

Some methods first perform a training, where a global dictionary is learned for each source. In this case, the association of the components with a source is straightforward. For test signals, the global dictionary is obtained by concatenating the dictionaries for all sources and only the activation matrix is updated. This process has been used in [65, 66]. Other methods apply directly on test signals without previous training. This is the case of [67], even though a supervised method is used for associating a component with a source.

Several algorithms can be used to obtain the estimation of the matrices $\mathbf{H}$ and $\mathbf{W}$. A review is given in [68].

1.4.6 Curve fitting

Some methods mentioned in this section can be seen as curve fitting, which consists in approximating a set of measurements with a simpler curve. Here, we must differentiate parametric and nonparametric curve fitting. Parametric curve fitting consists in approximating the measurements with a predefined model that contains some parameters. A well known case of curve fitting is linear regression [69], which consists of approximating the measurements with a linear function. In this case, the parameters to optimize are the slope and the y-intercept. Properly written, parametric curve fitting consists of approximating a set of observations $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=0,\dots,N-1}$ where the input $\mathbf{x}^{(i)}$ is associated to the output $\mathbf{y}^{(i)}$ with a model f defined as the sum of predefined basis functions:

$$f(\mathbf{x}) = \sum_{m=0}^{M-1} f_m(\mathbf{x}, \mathbf{a}^{(m)}) \tag{1.18}$$

where the $\mathbf{a}^{(m)}$ are the parameters to estimate. For two different values m_1 and m_2, $\mathbf{a}^{(m_1)}$ and $\mathbf{a}^{(m_2)}$ can have different dimensions, but the dimensions are previously known. The values of the parameters are obtained by minimizing a distance between the predictions of the model $f(\mathbf{x}^{(i)})$ and the measurements $\mathbf{y}^{(i)}$. Let us notice that the functions can map a value from a set to another set, and it is therefore not necessary that the input and the output of the model have the same dimension. It is also not necessary that the vectors $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ are multidimensional, they can contain only one value (in this case, we could replace them by scalars). For example, a lot of applications are performed with only the time value as an input. As the model is fixed before performing fitting, the amount of basis functions (M in Equation (1.18)) cannot change during the process. Parametric methods are generally quite easy to compute, but their results are limited by the quality of the model. In most situations, the model can only give an approximation of what the real signal should be, either because it does not consider some variables that have an impact on the signal, or because the model does not match exactly the real relation between input variables and outputs.

For nonparametric curve fitting, some models are still based on a set of predefined functions, but it is not decided beforehand which of the possible basis functions will be used for the model. For example, it would be possible to keep computing the scales of the wavelet Transform of a signal until a good level of reconstruction has been obtained. The spline smoothing [70], multivariate adaptive regression spline [71] and kernel regression [72] are other examples of nonparametric methods relying on existing functions.

Nonparametric curve fitting can also denote methods where no assumptions are made for a set of basis functions. In these methods, each sample is estimated without relying on a predefined model. This is the case for example of TV based methods, when not combined with other kind of methods (which is the case for example in [47, 48, 49, 50]).

1.4.7 Other methods

Other methods have been used for performing source separation or denoising.

Independent Component Analysis (ICA) is used to rewrite a signal as a sum of independent components. As for the dictionary in NMF, the components are adapted to the processed signal instead of using a predefined set of functions, as it would be done with some methods, such as wavelet transform. The first implementations assumed that the recordings contained at least as much channels as sources to retrieve [73, 74]. It has then been extended to over-complete ICA, where the amount of components is higher than the amount of channels [75], or even for single-channel signals [76, 77]. [77] points out that cases with lower independence might result in a need of additional information to perform the task.

Empirical Mode Decomposition [78] aims to decompose the signal on a set of functions. Contrary to the Fourier transform or the wavelet transform, the set is not predefined but obtained from the modes (local extrema) of the signal. The functions of the set can be any function that has as many extrema as zero crossings and a symmetric envelope (the upper envelope links all local maxima, the lower envelope links all local minima). In [79], it was used for denoising ECG signals, removing even the baseline wander. In [80], empirical mode decomposition was combined with independent component analysis to perform source separation.

Hidden Markov Models are models that aim to retrieve an unavailable signal based on the observation of another signal that is linked to the target signal. For example in denoising, the noisy signal would be the observation and the clean signal would be the hidden signal. For a target random signal Y and an observation X, the idea is to first define a probability density $f_T(a,b) = \mathbb{P}(Y_{n+1} = a | Y_n = b)$ mapping the temporal dependencies of the target signal and a probability density $f_C(x,y) = \mathbb{P}(X_n = x | Y_n = y)$ that maps the dependencies between the target signal and the observation (both densities are assumed to remain the same for all instants), as well as a density for the first instant $f_0(y) = \mathbb{P}(Y_0 = y_0)$. Then, for a specific observation $\mathbf{x}$, the objective is to find the sequence $\mathbf{y}$ that is the most likely to be at the origin of $\mathbf{x}$ (using for example the Viterbi algorithm [81]). In [82], one hidden Markov model is used for each speaker in a source separation task. A similar approach is used in [83] for separating speech and noise.

1.5 Review of the literature : Deep Learning

In the last years, conventional techniques have been progressively replaced by deep learning methods, which generally offer competitive or better performances, require few hypotheses and are time-efficient at inference time. In this section, we first recall some general concepts of Deep Learning, then we introduce some specific kinds of architectures, with bibliographical elements on their use for denoising and source separation. We will focus mostly on Fully Convolutional Networks as we will use this kind of architecture throughout the thesis. Some reviews about Deep Learning are given in [84, 85] for a general overview and [86] for speech separation.

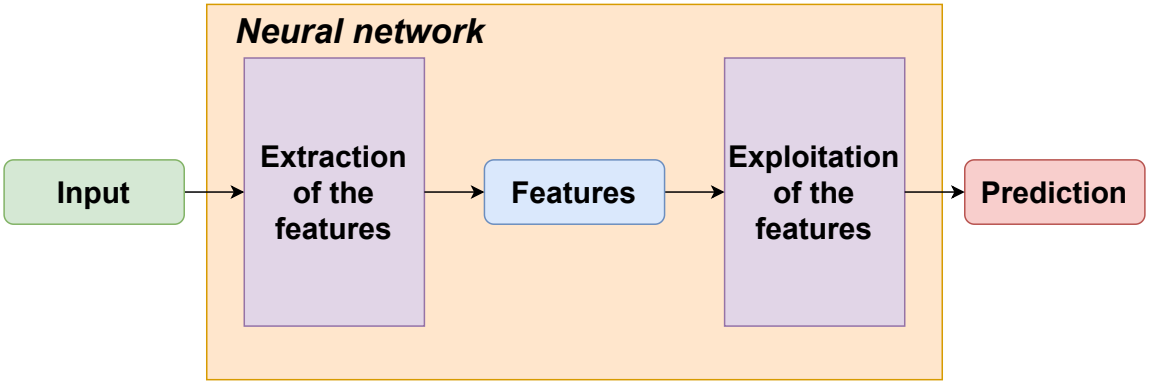


Figure 1.8: Overview of an Artificial Neural Network

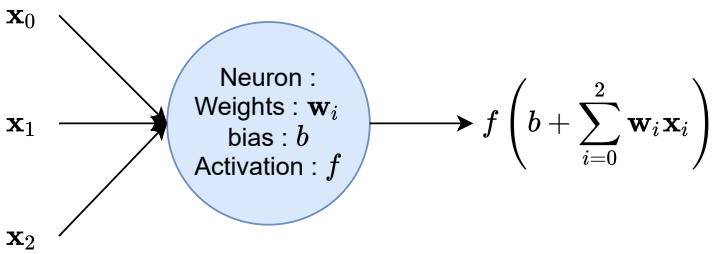


Figure 1.9: Example of a neuron

1.5.1 A short introduction to Deep Learning

Required background

Deep Learning is a specific branch of Machine Learning. It generally uses algorithms called artificial neural networks (ANN) to solve various tasks. ANN are a succession of operations applied on the data, resulting in a filter able to represent extremely complex functions. It is generally possible to decompose a neural network as a feature extractor followed by an exploitation of the extracted features, as shown in Figure 1.8. Neural networks can contain a huge amount of trainable parameters (sometimes tens of millions) that are generally obtained by using the same methods as for Machine Learning in general. However, trust-region algorithms are rarely used and stochastic gradient descent based methods such as Adam [87] and RMSProp [88] are preferred.

The first implementations of neural networks used neurons. A neuron is an operator that first applies a linear combination of all inputs and adds a bias, then applies a nonlinear function, called activation. An example of neuron is given in Figure 1.9 for an input with three elements.

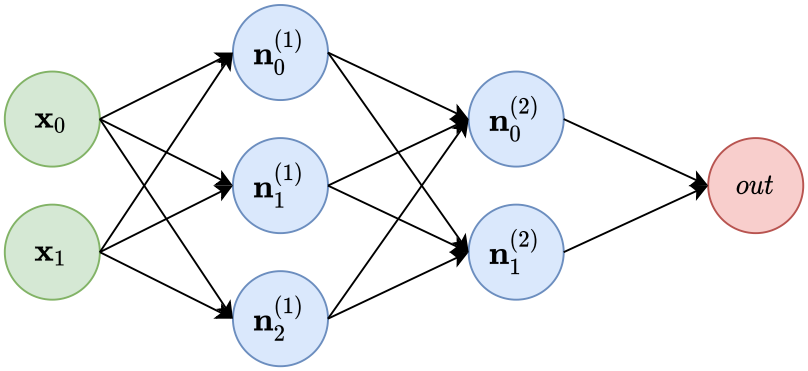


Figure 1.10: Example of a fully-connected neural network
The input layer is shown in green, the output layer is shown in red

In general, the weights and bias of a neuron are trainable parameters, that will change during optimization of the architecture, whereas the activation function is fixed within the architecture. Among some common activation functions, we could mention:

Rectified Linear Unit (ReLU) $f(x) = \max(0, x)$;

Leaky ReLU $f(x) = \begin{cases} x \text{ if } x > 0 \\ \alpha x \text{ otherwise} \end{cases}$ with $\alpha \in \mathbb{R}^+$ fixed before training;

Parametric ReLU It is one of the activation functions that is impacted during training, applying the same operation as the Leaky ReLU, but the parameter α is trained;

Sigmoid $f(x) = \frac{1}{1+e^{-x}}$;

Hyperbolic tangent (tanh) $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$;

Softmax For this activation, all inputs are considered to compute all outputs, instead of applying a function to the result of the linear combination of all inputs as for other activation functions. For N input value $\mathbf{x}_i$, each output value is obtained as $\mathbf{y}_i = \frac{e^{\mathbf{x}_i}}{\sum_{j=0}^{N-1} e^{\mathbf{x}_j}}$. This activation is generally used for computing the output of a classification network, as it transforms the input values into a vector of output probabilities.

In a neural network, several neurons are put in parallel to obtain several representations based on the same input. The outputs of all these parallel neurons are then given as an input for a new set of parallel neurons. Each set of parallel outputs of neurons is called a "layer". The inputs are generally denoted as the "input layer", the last layer is denoted as the "output layer" and the in-between layers are denoted as "hidden layers". A representation of a neural network with two hidden layers is given in Figure 1.10.

Fully connected neural networks (FCNN)

When using only classic neurons, the resulting architectures are named fully connected neural networks (FCNN). They obtained satisfactory results in various applications, but have three major drawbacks. First, they are variant to translation; if a phenomenon is always observed at the same position of the signal during training, then the network will learn that this phenomenon can only occur at a specific position of the signal as the data are processed without consideration of the temporal or spatial correlations. Second, they generally require a huge amount of parameters; this is a consequence of their variance to translation, which implies that several layers would have to be used for detecting the same phenomenon at different positions, but also the fact that each neuron uses one parameter per input. Lastly, FCNN cannot adapt to the shape of the input; a network is designed for a specific shape and the input signal has to match exactly this shape. Remark that these architectures need a 1D signal as input. It is still possible to process images (or higher dimensional data) by reshaping them into a 1D vector. As the spatial correlations are not used, the position of each pixel in the resulting vector has no impact on the result, as long as it remains the same from training to inference.

Recurrent neural networks (RNN)

For overcoming these issues, other kinds of architectures have been proposed. Among them, Recurrent Neural Networks (RNN) have had a large success. For these architectures, each neuron will consider only the current time step and the output of the previous time step. This makes them able to learn temporal correlations. Theoretically, they could learn correlations for arbitrary long time ranges, but the training restricts this possibility. They are also much more robust to translation than FCNN. It is also possible to use bidirectional process, that looks for both the future and the past. In this case, the computation is made by having two parallel processes, one for the past, the other for the future, that are combined to obtain the final result.

Despite these advantages, RNN still have several drawbacks. First, the computation of the value of one neuron at a certain time requires having already computed the output of this neuron for the previous time. As a consequence, these architectures are hardly parallelizable and require long computation time. Another issue is linked to the gradient descent. RNN have the problem of a vanishing gradient (the gradient falls to 0 and the parameters of the network are not updated anymore) or exploding gradient (the gradient takes extremely high values and the training becomes unstable). For solving the vanishing gradient issue, some solutions have been proposed, such as Long Short-Term Memory (LSTM) networks [89] or Gated Recurrent Units [90], which replace the conventional recurrent unit with a small internal network. These solutions however increase the amount of parameters.

1.5.2 Convolutional neural networks (CNN)

Before detailing what a convolutional neural network is, we should first recall what a convolution is. A convolution is an operation that takes two signals $\mathbf{x}$ and $\mathbf{y}$ and returns a new signal $\mathbf{z}$ such that:

$$\forall n \in \mathbb{Z}, \mathbf{z}_n = \sum_{k=-\infty}^{\infty} \mathbf{x}_k \mathbf{y}_{n-k}. \tag{1.19}$$

The signal $\mathbf{z}$ can be written with the convolution operator $*$ as $\mathbf{z} = \mathbf{x} * \mathbf{y}$. The convolution is commutative, i.e., $\mathbf{x} * \mathbf{y} = \mathbf{y} * \mathbf{x}$. This process can also be extended to 2D:

$$(\mathbf{X} * \mathbf{Y})_{i,j} = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \mathbf{X}_{k,l} \mathbf{Y}_{i-k,j-l}. \tag{1.20}$$

The sums are theoretically infinite, but in practice, they are reduced to cover the temporal support of both signals (for example, if one of the two signals is defined from instant 0 to 9, then the sum cannot contain more than 10 terms). Here, we will make all comments by speaking about 1D signals, but all notions can be extended to 2D (or more) signals.

A convolutional neural network is a network where the weights of each neuron are replaced by convolutional kernels. A convolutional kernel is a signal w of finite length, generally centered and non-causal (its temporal support goes from $-K$ to K). Each layer contains a set of channels (signals) instead of having a set of scalar values. If the previous layer contains N channels $\mathbf{o}^{(p,0)}, \dots, \mathbf{o}^{(p,N-1)}$, then the output of the i-th neuron will be defined as:

$$f\left(\sum_{m=0}^{N-1} \mathbf{w}^{(p,i,m)} * \mathbf{o}^{(p,m)} + \mathbf{b}\right) \tag{1.21}$$

with $\mathbf{w}^{(p,i,0)}, \dots, \mathbf{w}^{(p,i,N-1)}$ the kernels of the neuron, $\mathbf{b}$ the bias signal, which has the same dimension as $\mathbf{w}^{(p,i,0)} * \mathbf{o}^{(p,0)}$, and f an activation function, defined as for other architectures but applied element-wise. All neurons have kernels of the same length, but the weights can differ. Therefore, each neuron will output a different signal. The obtained signals are the output "channels" of the layer.

Using convolutions allows obtaining a "receptive field", that is to say that each sample is being processed by observing a local context around this sample, composed by the surrounding input samples. Properly written, for one convolution of a kernel $\mathbf{w}$ of length $2K+1$ with an input signal $\mathbf{o}$, the output $\mathbf{u}$ of the convolution at instant n is defined as:

$$\mathbf{u}_n = (\mathbf{w} * \mathbf{o})_n = \sum_{k=-K}^{K} \mathbf{w}_k \mathbf{o}_{n-k}. \tag{1.22}$$

Therefore, the output of a convolutional layer at time n is a function of the surrounding samples in the previous layer. The samples of the previous layer that define the output at

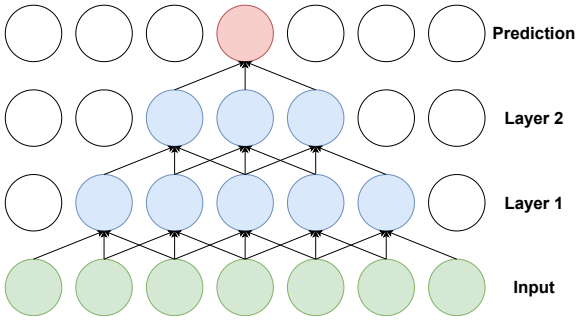


Figure 1.11: Illustration of the expansion of the receptive field by applying successive convolutions

instant n are called the local receptive field of the convolution layer. By applying a second convolutional layer, the area of samples considered for processing one sample of the signal will increase, as shown in Figure 1.11.

It is possible to use networks starting with convolutional layers and ending with fully-connected layers. This is for example the case for a lot of classification tasks, such as with the well-known ImageNet [91], ResNet [92], AlexNet [93] and GoogLeNet [94] for image classification. However, it is also possible to use networks containing only convolutional layers. These networks are called Fully Convolutional Networks (FCN). For such architectures, the output for an instant n only depends on the receptive field created by all the convolutions. The first advantage of this method is that the network will learn to observe the local context surrounding the sample rather than observing the full signal, and as a consequence, it will be translation-invariant. Additionally, the information of the input samples in the receptive field is the only information needed for computing the result at instant n, which means that the denoising of two samples at different instants can be performed independently, which allows large parallelization of the architecture.

We should also mention that convolutional layers also have the possibility to use padding. Padding generally refers to the fact of replacing unknown values at extremities of the signal. There are three kinds of padding:

Full padding In (1.22), the value is returned as long as at least one of the indexes $n-k$ is part of the temporal support of the input (the indexes $-2K, \dots, -1$ and $N, \dots, N+2K-1$ are obtained by adding zeros to the signal). The result is then given from indexes $-K$ to $N-1+K$ resulting in an output of length $N+2K$. Therefore, a convolution with kernel size $2K+1$ and a full padding results in an augmentation of the signal length.

Valid padding In (1.22), the value is returned only if all indexes $n-k$ are part of the temporal support of the input (the original signal is not padded). The result is then given from indexes K to $N-1-K$ resulting in an output of length $N-2K$. Therefore,

a convolution with kernel size $2K+1$ and a valid padding results in a reduction of the signal length.

Same padding In (1.22), the value is returned only if n is part of the temporal support of the input (the indexes $-K, \dots, -1$ and $N, \dots, N+K-1$ are obtained by adding zeros to the signal). The signal keeps the same length.

In our implementations of FCN, we will use the "Same" padding.

1.5.3 Specific operations

Parallel blocks

In neural networks, it is possible to use specific blocks for improving the variety of the features extracted by the network. A first possibility is to use parallel blocks, where the input is fed to two (or more) subnetworks which give different outputs. These different outputs are then combined and postprocessed (the combination can be done through a sum, a multiplication, a concatenation, ...). The inception networks [94, 95] are well-known architectures using this idea for image classification.

Skip connections

Another possibility is to use skip connections, where the n-th layer takes into account the output of the $n-1$-th layer but also of some earlier layers (for example the $n-2$-th layer). As for parallel blocks, the different outputs are combined before entering the new layer. This idea can be useful for combining several features instead of just building a new one that will erase the previous ones. According to [96], it might also results in easier to optimize loss functions.

Dilated convolutions

Concerning convolutional networks, it is possible to use other kinds of convolutions. Convolutional layers can have a dilation rate, resulting in a dilated convolution, which is defined

as a convolution that skips some of the elements of the input signal. For a dilation rate r, the Equation (1.22) becomes:

$$\mathbf{o}_n^{(p+1)} = \sum_{k=-K}^{K} \mathbf{w}_k \mathbf{o}_{n-kr}^{(p)}. \tag{1.23}$$

A dilated convolution of kernel size $2K+1$ and a dilation rate r could also be seen as a conventional convolution with a kernel of length $2Kr+1$, where all elements except the ones with indexes $-Kr, \ldots, -r, 0, r, \ldots, Kr$ are null.

Strided convolutions and downsampling

Strided convolutions can also be used. Using strided convolutions with stride s implies that only 1 out of s output values will be computed. This can also be seen as performing the full convolution, then removing the corresponding values. In other words, the result of Equation (1.22) is kept only for indexes $n + js$ with j an integer. It is possible to use both stride and dilation together.

By using a stride other than 1, the output channel will be reduced. Strided convolutions are a way for downsampling the signal in a trained manner, but other methods are possible. A classic one is pooling, which creates groups of values from the input and returns a single value for each group. For example, max-pooling will retain only the maximal value of each group, whereas average pooling will return the mean value of each group. Remark that pooling is not necessarily specific to convolutional networks.

Upsampling

Oppositely to downsampling, it is also possible to perform upsampling. The simplest method is to duplicate each value from the input. It is also possible to use an interpolation (e.g. linear interpolation). For convolutional networks, it is possible to use "transposed convolutions". For an input $\mathbf{o}^{(n)}$, a transposed convolution with a dilation rate r, a stride s and specific padding mode and kernel size aims to retrieve an output $\mathbf{o}^{(n+1)}$ that would have resulted in a signal of the same shape as $\mathbf{o}^{(n)}$ if a convolution with a dilation rate r, a stride s and the same padding mode and kernel size had been applied to it. This is done by duplicating each sample s times, padding appropriately, then applying a convolution without stride and with the desired kernel size and a dilation rate r.

1.5.4 Denoising and separation with FCNN

While methods using FCNN have lost interest in the last years due to their drawbacks, the first methods for performing denoising or source separation with Deep Learning were often based on these architectures.

Based on the success of using spectrogram representations in methods such as NMF, and for exploiting similar networks as for image applications (such as in [97]), the first implementations of signal denoising were mostly performed with spectrograms [98, 99, 100] or cepstrograms [101], which are another 2D representation of signals based on Fourier transforms. In [102], several approaches, including with FCNN, are used and one architecture is designed for each source.

1.5.5 Denoising and separation with RNN

As for FCNN, some methods are performed using 2D representations of the signal. For such implementations, one column of the spectrogram (or other representation), corresponding to one time step, is given to the network which processes the successive columns using recurrent layers [103, 104, 105, 106]. Similar approaches were performed with LSTM layers instead of classic recurrent layers [107, 108].

In [109], a CNN is first applied to obtain mixture weights for basis signals that are built with a decoder. LSTM layers are used to find the contributions of each source to the weights of the mixture.

A few methods also tried to use LSTM in the waveform domain. An example is given in [110] for ECG denoising. However, in this case, the method is compared to a convolutional network which generally gets better results.

Recurrent layers are sometimes combined with convolutional layers. This is for example the case of [111] with time-frequency features, or in [112] directly in time domain, both for audio source separation.

1.5.6 Denoising and separation with CNN

As for other methods, FCN were first used on 2D representations of the signal. In [113, 114], denoising is learned in an adversarial way with a fully convolutional generator. In [115], an

autoencoder is defined using two convolutional layers and one fully connected layer for the encoder. Instead of using a single decoder, they use parallel decoders, each of them having two convolutional layers and one fully connected layer for performing source separation. In [116], the separation is performed by dividing the full spectrogram into several sub-spectrograms, obtained by removing some of the frequencies, that are processed in parallel and combined afterwards. In [117], two fully convolutional autoencoders are trained simultaneously to create an autoencoder for encoding and decoding a mixture spectrogram, and another autoencoder for encoding and decoding the source. The separation is then performed by encoding the mixture with the mixture encoder and decoding it with the source decoder.

When using spectrograms, the receptive field is often not mentioned. It is probably due to the creation of the spectrograms that reduces the time resolution, and each created time frame already depends on a time window surrounding it. Therefore, even small receptive fields are already able to observe a sufficient amount of samples for analyzing relevant temporal correlations. However, for methods in time domain, the amount of samples that need to be observed in the receptive field might be much larger, especially with high sampling rates. For this reason, methods performing denoising or separation in time domain generally focus on a way to obtain a long enough receptive field in a limited amount of layers.

In [118, 119, 120], this problem is solved by applying convolutions with large kernels to obtain the adequate receptive field. This solves the problem of having a too short receptive field, but the way it is done is not necessarily efficient, as the same input samples are considered multiple times for each output sample. This is clearly illustrated in Figure 1.11, where despite using kernels of length 3 and only two hidden layers, each sample other than the ones at the extremities is used three times. Therefore, the model uses a huge amount of parameters for processing the same information several times.

Here, we will detail two models that have been designed for obtaining a long receptive field with a limited amount of parameters and avoiding repetition of the same information. Our Deep Learning experiments in this thesis are based on these two models. We will also mention some other models that have been defined by applying similar processes as these two models.

1.5.7 A WaveNet for speech denoising

The "wavenet for speech denoising" [9] is a fully convolutional architecture based on [121], a model used for generating audio in an end-to-end manner to avoid making any assumption on the appropriate features and to be able to use every information in the signal, including the phase.

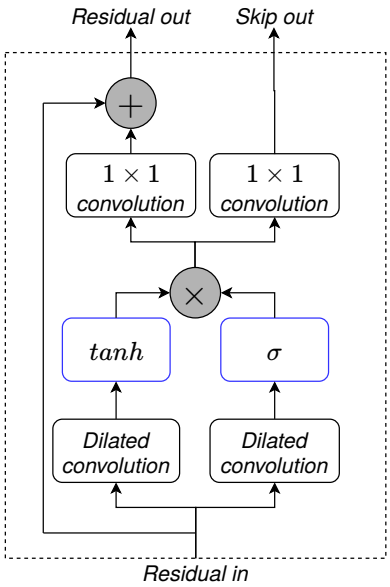


Figure 1.12: Details of a residual layer

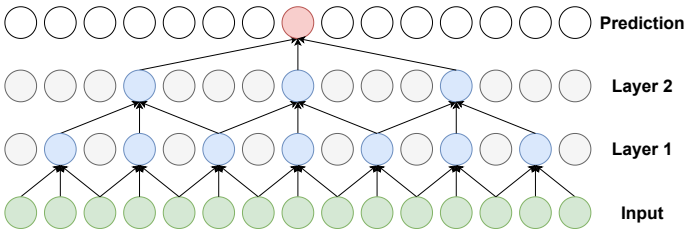


Figure 1.13: Illustration of the expansion of the receptive field by applying successive dilated convolutions

In both architectures, the problem of the size of the receptive field is solved by using stacks of residual layers. A residual layer is a gated unit which combines two parallel dilated convolutions, one of them activated by a tanh to extract the information from the signal, and the other one activated by a sigmoid to select the information that should be kept. 1×1 convolutions (convolutions having a kernel of length 1) are then used to define skip and residual connections. Let us notice that each output channel of a 1×1 convolutions is a linear combination of all input channels. Such convolutions are therefore useful for changing the amount of channels. The details of a residual layer can be seen in Figure 1.12.

Each stack is made of K residual layers with dilation factors $1, 2, \ldots, 2^{K-1}$, which efficiently increases the receptive field. As a comparison with the Figure 1.11, Figure 1.13 shows that using dilated convolutions with an increasing dilation rate allows obtaining a longer receptive field with the same amount of parameters and less repetition in terms of used input samples.

There are S successive stacks, allowing a deeper architecture without increasing the receptive field as much as with a single stack of $S \times L$ layers. 1×1 convolutions are used at

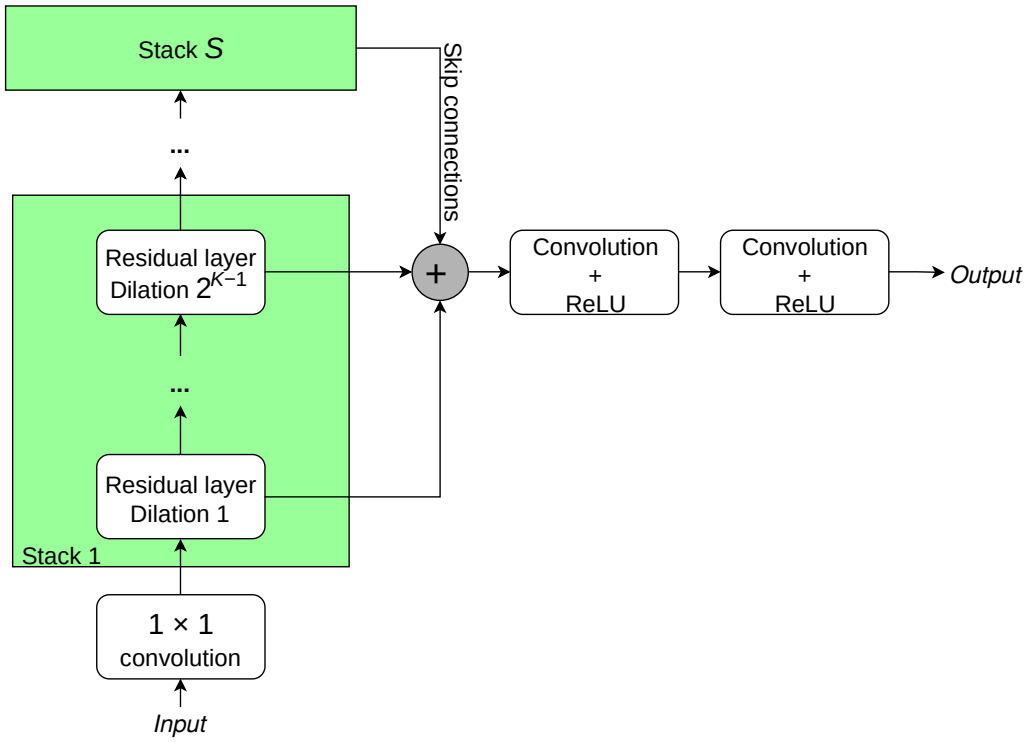


Figure 1.14: Overview of the WaveNet architecture

the beginning of the network for obtaining the adequate amount of channels, and the skip connections of all residual layers are summed and postprocessed by two last convolutional layers, which allows a multiscale analysis, as for wavelets. The channels of the last convolutional layer are then combined with a weighted sum to obtain the output. An overview of the architecture can be seen in Figure 1.14. When using kernels of size 1×3 for all convolutions (except 1×1 convolutions), the global receptive field length is:

$$S \times (2^{K+1} - 2) + 5. \tag{1.24}$$

As [9] is used for denoising, it also has some specificities compared to [121]: all convolutions are non-causal and time continuity is ensured by using kernels of length 3 instead of 1 in the two convolutions performed after summing skip connections, rather than using an autoregressive process as in [121] (which is non-parallelizable and therefore time-consuming).

The original architecture replaced biases by features obtained from information about the speaker being denoised. In our implementations, such information is not available. Therefore, biases are used in the pre- and post-processing layers (but not in residual layers).

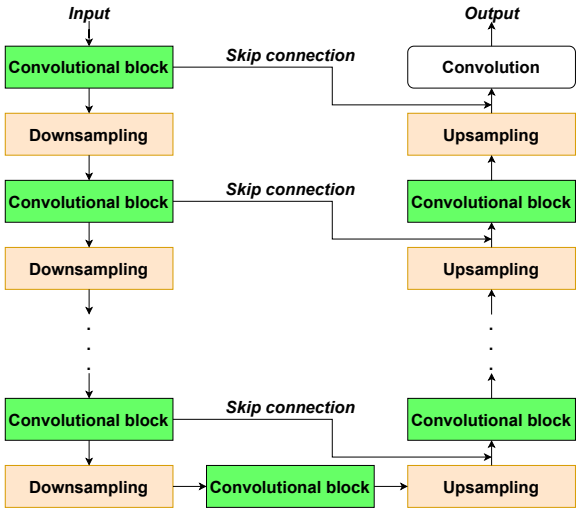


Figure 1.15: Overview of the Wave-U-Net architecture

As for Wavenet, several architectures [122, 123, 124] use dilated convolutions for obtaining an adequate receptive field.

1.5.8 Wave-U-Net

The dilated convolutions used in the Wavenet for speech denoising allow obtaining an adequate receptive field in an efficient way, but as the shape of the signal remains the same during the entire process, each channel of each layer will require as much memory as the original signal. Considering the entire network, the required memory limits the possibility of using too long signals.

To overcome this issue, Eeference [11] proposed the Wave-U-Net, an architecture based on the image processing U-Net, where the dilations are replaced by downsampling layers (in our case, we use max-poolings). This method allows to have the same evolution of the receptive field as in Figure 1.13 (if kernels of size 3 are used), but as the length of the signal is divided by 2 after each downsampling, the required memory is much smaller.

To retrieve the original shape of the signal, several layers of upsampling are also used (in our case, we use transposed convolutions with kernel size 2 and stride 2). After each upsampling, the output channels of the output are concatenated with the ones of the signal at the same resolution obtained during the downsampling path. An overview of the architecture can be seen in Figure 1.15. The convolutional blocks are a convolutional layer followed by a ReLU activation.

There might be three issues with the Wave-U-Net architecture. First, using max-pooling implies that the network is not completely time-invariant. This is illustrated in Figure 1.16,

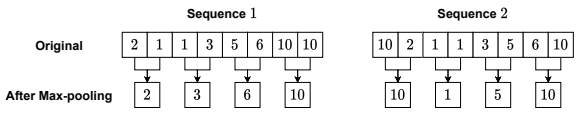


Figure 1.16: Illustration in the variance to translation in max-pooling. The same subsequence $2, 1, 1, 3, 5, 6$ depending on its relative position to the two 10.

where the same subsequence $2, 1, 1, 3, 5, 6$ produces two different results depending on the position of the two 10 in the full sequence. In our case, this should be a minor issue as the high sampling rates imply low variations from one sample to another, except for discontinuities.

The other issue is that each downsampling block will divide the length of the input by 2, rounded to an integer, while each upsampling block will multiply it by 2. Therefore, for ensuring the same length for the input and the output, the signal has to have a length that is a multiple of 2^D with D the amount of downsampling blocks. Again, this is not a major issue as it is still possible to use zero-padding to reach an adequate length.

Lastly, as there is only one downsampling path and only one upsampling path, this architecture does not offer the possibility to adapt both the depth and the receptive field together, as it is done by starting a new stack in WaveNet. It would be possible to use several successive Wave-U-Nets, but summing all features would not be possible as they have different lengths. This shows a different behavior compared to WaveNet, but it is not necessarily a major issue.

The idea of using successive downsamplings has been used in other studies, such as in [125] which proposes a Generative Adversarial Network where the generator has a similar structure as the Wave-U-Net, or in [126] which explicitly extends the Wave-U-Net to the attention framework.

1.6 Evaluating denoising performance

Several approaches can be used for assessing the quality of a denoising and compare several denoisers. We present some of the most common ones with their advantages and defaults.

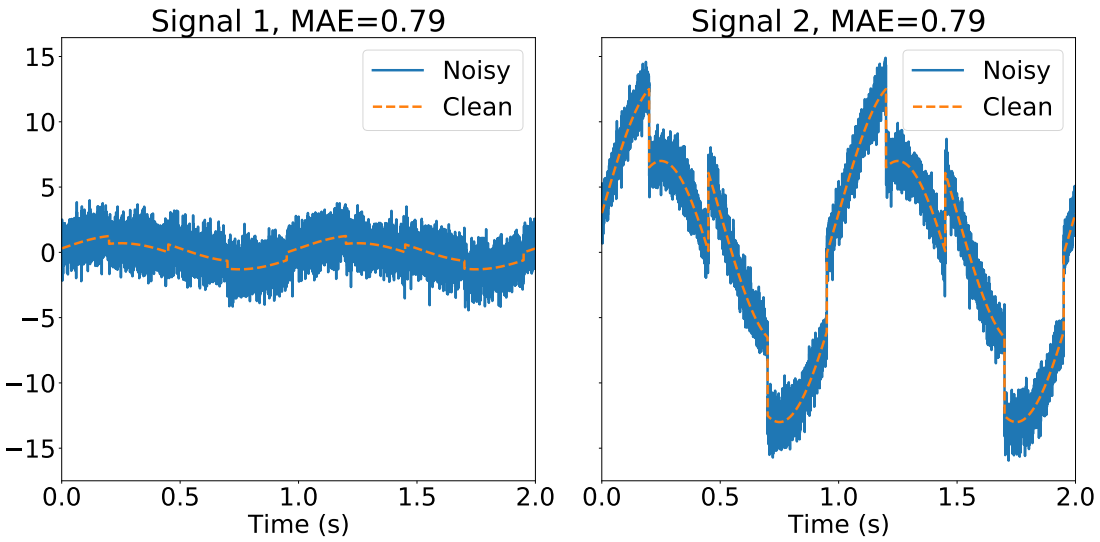


Figure 1.17: Illustration of the problem of MAE being independent of the signal

1.6.1 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) between an estimation $\hat{\mathbf{y}}$ and the target signal $\mathbf{y}$ of length N is defined as:

$$MAE\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \frac{1}{N} \sum_{n=0}^{N-1} \left|\mathbf{y}_n - \hat{\mathbf{y}}_n\right| = \frac{1}{N} \left\|\mathbf{y} - \hat{\mathbf{y}}\right\|_1 . \tag{1.25}$$

This metric is helpful for evaluating cases where it is important to improve the results in every part. For example an improvement from $\mathbf{y}_n - \hat{\mathbf{y}}_n = 0.8$ to $\mathbf{y}_n - \hat{\mathbf{y}}_n = 0.2$ will result in the same improvement in the MAE as an improvement from $\mathbf{y}_n - \hat{\mathbf{y}}_n = 10.8$ to $\mathbf{y}_n - \hat{\mathbf{y}}_n = 10.2$.

A possible drawback of MAE is that it depends only on the error of estimation, the signal by itself has no impact on the obtained value, i.e., it is not representative of the actual signal-to-noise ration, but an absolute measure of the error. Two signals containing exactly the same noise will have the same MAE, while one of them could be much easier to interpret. For example, for two signals having the same shape but different scales (each signal is a multiple of the other signal) and containing the same noise, the MAE will be the same. This is illustrated in Figure 1.17, where the first signal appears to be much harder to interpret than the second, on which discontinuities are clearly visible despite the noise.

1.6.2 Mean Square Error (MSE) and Root Mean Square Error (RMSE)

The Mean Square Error (MSE) has a similar definition as the MAE, except that it uses the square values instead of the absolute value. The MSE between an estimation $\hat{\mathbf{y}}$ and the target signal $\mathbf{y}$ of length N is defined as:

$$MSE\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \frac{1}{N} \sum_{n=0}^{N-1} \left(\mathbf{y}_n - \hat{\mathbf{y}}_n\right)^2 = \frac{1}{N} \left\|\mathbf{y} - \hat{\mathbf{y}}\right\|_2^2. \tag{1.26}$$

Oppositely to MAE, MSE will have a different behavior depending on the difference between a target sample and its estimation. While some small errors will have a minor impact on the global MSE, large errors will largely penalize the result, even if they are limited to only a few samples (or even to single samples). Therefore, MSE is used for ensuring good reconstruction for all samples, with the counterpart of sometimes preventing further optimization when all samples are already quite well approximated.

MSE can also be seen as an estimation of the second order moment of the noise, and therefore as an estimation of the variance of the remaining noise if the remaining noise is centered.

Another metric very close to the MSE is the Root Mean Square Error (RMSE), which is simply defined as:

$$RMSE\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \sqrt{MSE\left(\mathbf{y}, \hat{\mathbf{y}}\right)} = \frac{1}{\sqrt{N}} \left\|\mathbf{y} - \hat{\mathbf{y}}\right\|_2 = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} \left(\mathbf{y}_n - \hat{\mathbf{y}}_n\right)^2}. \tag{1.27}$$

For centered noise, RMSE is an estimation of the standard deviation of the remaining noise.

MSE and RMSE have the same drawbacks as the MAE concerning their independence to the signal.

1.6.3 Normalized measures of error: signal-to-Noise Ratio (SNR) and its derivatives

Signal-to-Noise Ratio (SNR) is used for comparing the level (magnitude) of the clean signal to that of the remaining noise. Rigorously, the SNR is computed as the ratio between the theoretical power of the signal and the theoretical power of the noise. It can then be expressed using a logarithmic scale to obtain a result in decibels. In our work, we will assimilate the SNR to its expression in decibels. We will also use estimates of the powers instead of their theoretical values (even when this theoretical value is available, for example as the variance of a centered noise). The SNR between an estimation $\hat{\mathbf{y}}$ and the target signal $\mathbf{y}$ of length N is then defined as:

$$SNR\left(\mathbf{y}, \hat{\mathbf{y}}\right) = 10 \log_{10} \left(\frac{\|\mathbf{y}\|_2^2}{\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2} \right). \tag{1.28}$$

It is possible to rewrite it as:

$$SNR = 10 \log_{10} \left(\frac{\|\mathbf{y}\|_2^2}{N \times MSE\left(\mathbf{y}, \hat{\mathbf{y}}\right)} \right). \tag{1.29}$$

Contrary to MAE, MSE and RMSE, SNR considers the clean signal and is therefore able to differentiate signals of various scales. However, two signals differing only by their average value will have different SNR. This is illustrated in Figure 1.18. While a solution might be to remove the average value to avoid this issue, a constant baseline often has some significance in the signal. Additionally, for the signals we are dealing with (except the FIB and SEM image sequences), the signals have a low average value, which will have a minor impact on the SNR.

Generally, SNR is used for evaluating a method but not to optimize it. Two observations might explain this:

- Using the properties of the logarithm function, SNR can be written as:

$$SNR\left(\mathbf{y}, \hat{\mathbf{y}}\right) = 10 \log_{10} \left(\|\mathbf{y}\|_2^2 \right) - 10 \log_{10}(N) - 10 \log_{10} \left(MSE\left(\mathbf{y}, \hat{\mathbf{y}}\right) \right). \tag{1.30}$$

  Therefore, as the models cannot change $\mathbf{y}$ or N, finding the optimum (maximum) of the SNR is equivalent to finding the optimum (minimum) of MSE, but requires computing more terms. As these two metrics have different derivatives due to the logarithm, some methods based on gradient descent would follow different optimization

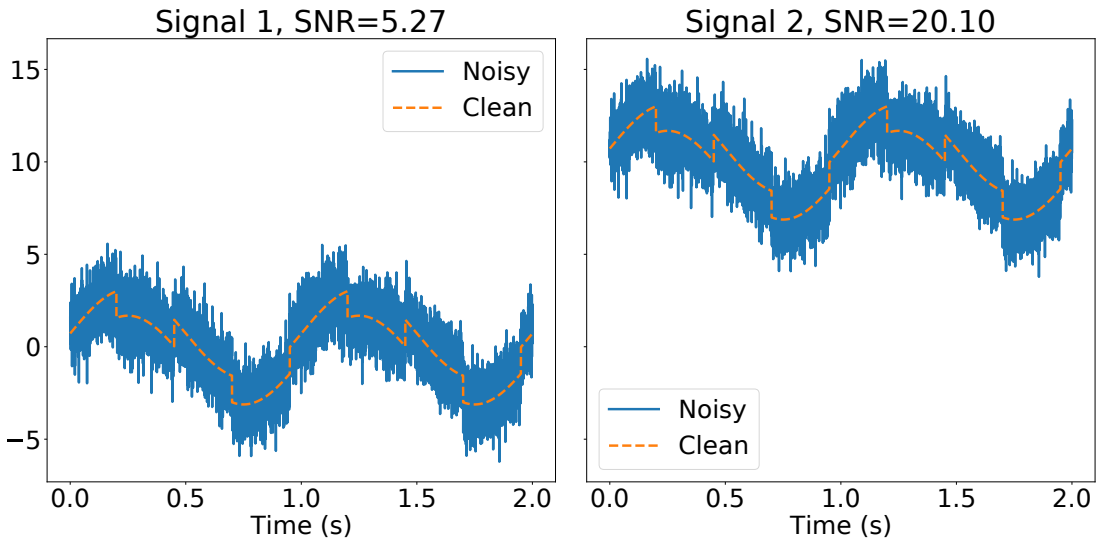


Figure 1.18: Illustration of the problem of SNR being affected by a shift

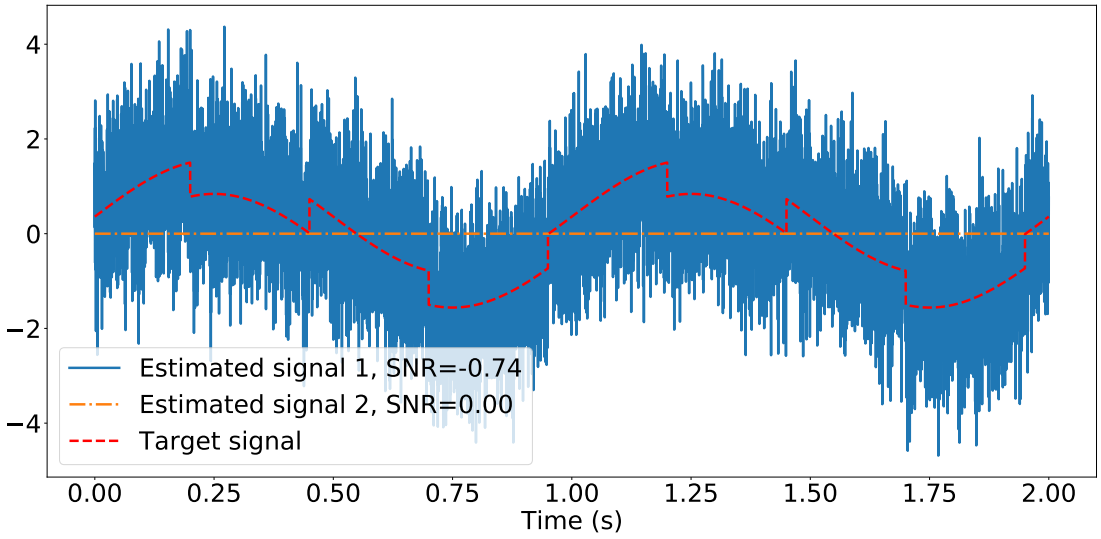


Figure 1.19: Illustration of the problem of SNR with complete removal of the signal

paths depending on the metric. Therefore, optimizing the MSE should give a different result as optimizing SNR, but the two results should still be close to each other.

- If the model entirely removes the signal (i.e. if it creates an output noise exactly opposite to the clean signal), the obtained SNR will be 0dB, despite the denoised signal having no utility. Therefore, there is a risk that a model trained with SNR learns to completely remove the signal if it has not been able to perform satisfactory predictions otherwise. An example is given in Figure 1.19, where the second estimation gives the best quantitative result despite being clearly useless for any analysis of the signal. Let us notice that while the SNR makes this problem obvious, this is also true for other metrics. In the provided example, it is possible to say, even without computing it, that the MSE of the second estimation would be better than the one of the first estimation, as the SNR decreases when the MSE increases. The main interest of this observation is to point out that a small variation of the SNR is not necessarily significant in terms of improvements of the results.

It is also possible to use the Segmental Signal-to-Noise-Ratio (SNR_{seg}) which computes the SNR on subsegments of the signal instead of directly computing the global SNR, then averages the different values. For a division in M segments, the result (in decibels) will be:

$$SNR_{\text{seg}}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \frac{10}{M} \sum_{m=0}^{M-1} \log_{10} \left[\frac{\sum_{n=m\frac{N}{M}}^{(m+1)\frac{N}{M}-1} \mathbf{y}_n^2}{\sum_{n=m\frac{N}{M}}^{(m+1)\frac{N}{M}-1} \left(\mathbf{y}_n - \hat{\mathbf{y}}_n\right)^2} \right]. \tag{1.31}$$

Using this term will limit the impact of some parts of the signals that might contain a noise that is much higher than for the majority of the signal. There is however a risk that some segment has an extremely high SNR resulting in a high SNR_{seg}, even if the general behavior is not good. In our case, we will mostly focus on the SNR as we expect the behavior to remain the same (or almost) for the whole signal.

1.6.4 Comments about the notations for metrics

For the rest of this thesis, we will assimilate the name of the metric to its value when this does not create any confusion. For example,we will use SNR for mentioning $SNR\left(\mathbf{y}, \hat{\mathbf{y}}\right)$.

1.7 Conclusion

In this chapter, we reviewed some of the methods that are usually used for performing denoising or source separation, another task that is closely related to denoising. After reviewing some conventional methods, we focused on Deep Learning based methods, especially methods using fully convolutional denoisers. In this thesis, we will first use a model fitting approach in Chapter 2. This method will let us perform a first denoising, but will mostly provide us some coherent parameters for friction models, that we will use throughout this thesis with deep learning based fully convolutional denoisers. Lastly, we made some comments on some of the metrics that are typically used for assessing the quality of denoising.

2

Analysis of the friction signals and model-based approach for denoising

The friction signals provided by Ireis seem to be cyclostationary. Examples of these signals are available on Figures 0.5 and 0.6. However, this hypothesis needs to be confirmed before we exploit it. In [127], some methods are provided for confirming such hypotheses up to fourth order. In this thesis, we are mostly interested in first order cyclostationarity, i.e. periodicity of the signal's expectation, and we aim to build methods that are able to adapt to random variations from one period to another that break cyclostationarity. We also want the designed methods to remain competitive even when the cyclostationarity cannot be exploited anymore, for example for transitions from one state to another. Therefore, we will perform simpler analysis, mostly based on visual observations, instead of relying to more complex analysis.

After this analysis, we propose a first method for denoising the signals. We base this method on curve fitting (see Section 1.4.6). Several parametric models have been proposed in the literature for viscous friction [128, 129, 130, 131]. Here, we rely on two simple models defined in [8] and [7], which link the friction coefficient to several parameters, including the velocity at the contact point. In our case, other parameters can be considered as constant. Therefore, the function depends only on the velocity. One section of this chapter shows that a sine wave is an appropriate model for the velocity. We estimate the parameters of this sine wave using a Levenberg-Marquardt algorithm [4, 5, 6].

Once the velocity is computed, it is possible to use it to fit the parameters of the viscous friction models. We use successive optimizations performed with the trust-region algorithm [132, 133]. Note that these results provide a first denoising, they are not meant to efficiently denoise all parts of the signals. The objective is merely to define a model for performing quantitative analysis in the next chapters.

For this chapter, all implementations have been made using Matlab [134].

2.1 Analysis of the signals

2.1.1 Preliminary description of experiments

The first step of our analysis consists in checking the cyclostationary nature of the friction signals and whether it constitutes a relevant assumption to exploit. Here, we already know the setpoint rotation speeds used for the acquisition of each signal, which means that we already know a frequency that should be close to the fundamental frequency or to a multiple of the fundamental frequency. After one full rotation, the mechanism is back to its initial state and is expected to have the same behavior again. As it is unlikely for the mechanism to have a rotation speed that is exactly the setpoint frequency, we apply a correction on the given frequency. This correction is based on Algorithm 1, which also uses Algorithm 2 and first detects the frequency based on the N computed points of the discrete Fourier transform, then uses a dichotomous approach to obtain a more precise value with a frequency that is possibly not in the set of frequencies $(0, \frac{f_e}{N}, \ldots, \frac{(N-1)f_e}{N})$ with N the amount of acquired samples. Afterwards, we use the detected frequency to define the expected amount of samples in one period of the signal. It is then possible to cut the signals in successive periods. If the cyclostationary hypothesis is verified, then the successive periods should verify the following properties, with $\mathbf{u}$ the studied signals:

- The full signal should visually be a repetition of the same pattern over time;
- Each period should be visually close to the average period. The average period is obtained as $\mathbf{u}_n^{(m)} = \frac{1}{P}\sum_{p=0}^{P-1}\mathbf{u}_{n+pT}$ with T the amount of samples in one period and P the amount of full periods. The last period is removed to avoid using zero-padding, or any other padding;
- The variance for each sample of the periods should be small to confirm the observations related to previous point. The variance is obtained as $\mathbf{v}_n = \frac{1}{P-1}\sum_{p=0}^{P-1}(\mathbf{u}_{n+pT} - \mathbf{u}_n^{(m)})^2$.

Here, the signal of interest $\mathbf{u}$ can be directly the friction signal $\mathbf{x}$ for checking first order cyclostationarity, but could also be another signal based on the friction signal. In our case, we will also consider the case $\mathbf{u} = \mathbf{r}$ where $\mathbf{r}_n = \mathbf{x}_n\mathbf{x}_{n+\tau}$ for checking second order cyclostationarity. We will test three values of τ, but theoretically, we should check all possible values to be sure that the signal is cyclostationary.

To check if the mentioned properties are verified, we will display for each order the entire signal, one period of the signal, the average of the periods and the variance of the periods.

Algorithm 1: Find fundamental frequency and phase of a signal

Input: $\mathbf{x}$ // Temporal signal
Input: f_{th} // Expected theoretical frequency (e.g. the setpoint frequency)
Input: f_e // Sampling frequency
Output: f_0 // Estimated frequency
Output: A // Estimated magnitude for f_0
Output: φ // Estimated phase for f_0

$N \leftarrow \text{length}(\mathbf{x})$
$\mathbf{t_{vect}} \leftarrow \frac{1}{f_e}(0, \ldots, N-1)^T$
$Ind \leftarrow 1$
/* First estimate of the frequency */
$\mathbf{TF_x} \leftarrow \mathcal{F}_d(\mathbf{x})$
$\mathbf{f_{max}} \leftarrow 10$ frequencies with the maximal magnitudes/* Here, we pick ten frequencies, but less could be used (e.g. if no theoretical frequency is available, then we would just pick one) */
$f_0 \leftarrow \arg\min_{f \in \mathbf{f_{max}}} |f - f_{th}|$/* The frequency is initialized with the peak that is the closest to the expected fundamental frequency */
$\varphi, A, po_0 \leftarrow FTf(\mathbf{x}, \mathbf{t_{vect}}, f_0)$
/* Optimization of the frequency. This function is described in Algorithm 2 */
while *Stop criteria have not been reached* **do**
    shift $\leftarrow \frac{f_e}{2^{Ind}N}$
    $f_g \leftarrow f_0 - \text{shift}$
    $\varphi_g, A_g, po_g \leftarrow FTf(\mathbf{x}, \mathbf{t_{vect}}, f_g)$
    $f_d \leftarrow f_0 + \text{shift}$
    $\varphi_d, A_d, po_d \leftarrow FTf(\mathbf{x}, \mathbf{t_{vect}}, f_d)$
    /* Improvement of the frequency with a dichotomous approach */
    if $\min(po_g, po_d, po_0) = po_g$ **then**
        $f_0 \leftarrow f_g$
        $\varphi \leftarrow \varphi_g$
        $A \leftarrow A_g$
        $po_0 \leftarrow po_g$
    else if $\min(po_g, po_d, po_0) = po_d$ **then**
        $f_0 \leftarrow f_d$
        $\varphi \leftarrow \varphi_d$
        $A \leftarrow A_d$
        $po_0 \leftarrow po_d$
    end

    $Ind \leftarrow Ind + 1$
end

Algorithm 2: FTf : Compute the Fourier Transform at a specific frequency

Input: $\mathbf{x}$ // Temporal signal
Input: $\mathbf{t}$ // Time vector
Input: f // Frequency
Output: φ // Phase of the component
Output: A // Magnitude of the component
Output: wo // Power of the signal without the component

$N \leftarrow \text{length}(\mathbf{t})$;
$\mathbf{e} \leftarrow \left(e^{-2j\pi f \mathbf{t}_0}, \ldots, e^{-2j\pi f \mathbf{t}_{N-1}}\right)^T$;
$TF \leftarrow \mathbf{x}^T \mathbf{e}$;
$\varphi \leftarrow \arctan\left(Im(TF), Re(TF)\right)$;
$\mathbf{c} \leftarrow (\cos(2\pi f \mathbf{t}_0 + \varphi), \ldots, \cos(2\pi f \mathbf{t}_{N-1} + \varphi))$;
$A \leftarrow \frac{\mathbf{x}^T \mathbf{c}}{\mathbf{c}^T \mathbf{c}}$;
$\mathbf{x_{del}} \leftarrow \mathbf{x} - A \times \mathbf{c} - \text{mean}(\mathbf{x} - A \times \mathbf{c})$;
$wo \leftarrow \mathbf{x_{del}}^T \mathbf{x_{del}}$;

2.1.2 Analysis of first order cyclostationarity

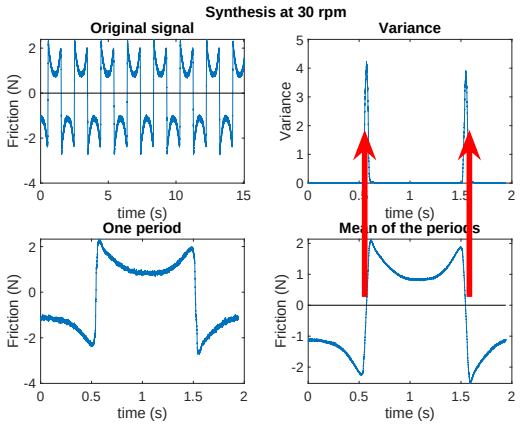


Figure 2.1: First order analysis for a signal at 30 rpm

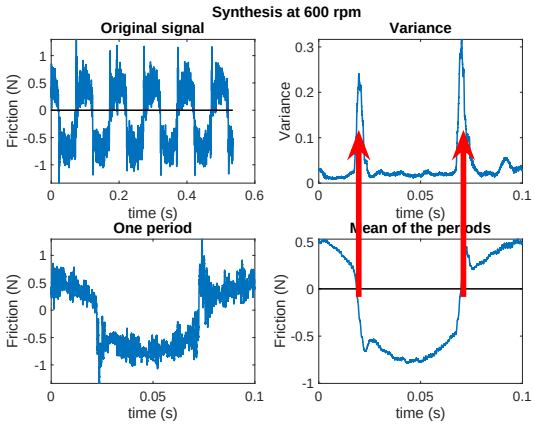


Figure 2.2: First order analysis for a signal at 600 rpm

In Figure 2.1, we show the first order analysis for a friction signal acquired with a setpoint rotation speed of 30 rpm. As expected, the signal exhibits cyclostationary properties at this order, with a low variance between the periods for most parts of the signals. However, a large variance appears around discontinuities, as illustrated with the red arrows. It is possibly due to the discontinuities corresponding to a direction change for the mechanism, and it is likely that the instability is larger in those conditions. It is also possible that there are little variations in the duration of the period, implying that the discontinuity is not always located exactly at the same instant of the period. Therefore, for a given instant, the discontinuity might have already occurred for some periods whereas other periods are still in the smooth area before the discontinuity, resulting in a large difference between these periods.

In Figure 2.2, we show the same analysis for a setpoint rotation speed of 600 rpm. The observations are mostly the same but the variance remains smaller, even around discontinuities. This might be explained by the lower magnitude of the discontinuities in this signal (the peak-to-peak difference is lower than $3N$, it was greater than $4N$ in Figure 2.1). It is also possible that the mechanism is more stable at this rotation speed. However, the fluctuation occurring near peaks seems to cover a larger portion of the period than in the low-frequency case.

Here, a simple test that could be done would be to apply a threshold on the variance and check which proportion of the samples are above this threshold. If the found proportion is low enough, then the first order cyclostationarity is confirmed. However, setting the right threshold and right proportion is not easy. Here, we can assume that the variance remains low for most of the period (outside of discontinuities), and the first order cyclostationarity hypothesis seems reasonable.

2.1.3 Analysis of second order cyclostationarity

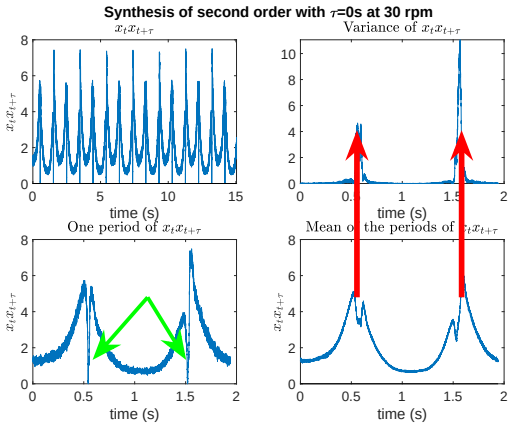


Figure 2.3: Second order analysis for a signal at 30 rpm with $\tau = 0s$

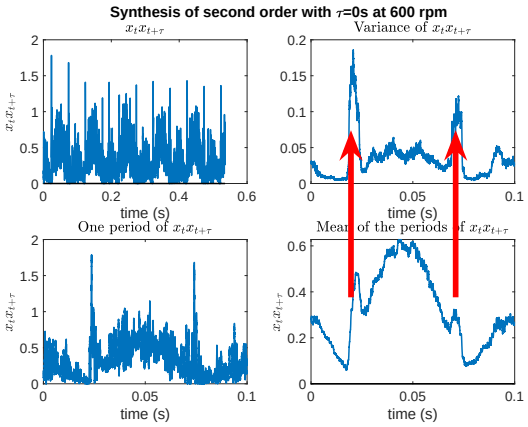


Figure 2.4: Second order analysis for a signal at 600 rpm with $\tau = 0s$

In Figure 2.3, we show the second order analysis for $\tau = 0s$, which corresponds to an estimate of the second order moment with a single sample, or the average of several of these estimates for the mean value. The obtained value falls to an extremely small value (as shown by the green arrows) on areas that correspond to discontinuities, which is due to the fact that for these signals, what we call discontinuities are actually fast transitions between two states of the mechanism, and the value 0 will be obtained during this transition. Obtaining a real discontinuity is hard in measurements. Even if the original signal actually contains a discontinuity, the sensor generally has a response time that makes the acquired signal transit between the two states in a smoother way. Therefore, zero-crossings occurring during discontinuities in the original signals imply going through values close to 0, explaining that

the second order moment is close to 0 in this area. Concerning the periodicity of the signal, the observations correspond to what would be expected from a second order cyclostationary signal. Again, the variance is mostly located around discontinuities, as shown by the red arrows. The same observations can be done for a signal at 600 rpm, even though the drop of the value around discontinuities is now less significant, even though small values are still obtained.

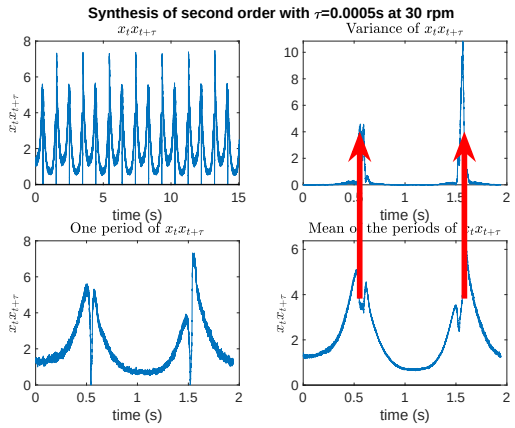


Figure 2.5: Second order analysis for a signal at 30 rpm with $\tau = 0.0005s$

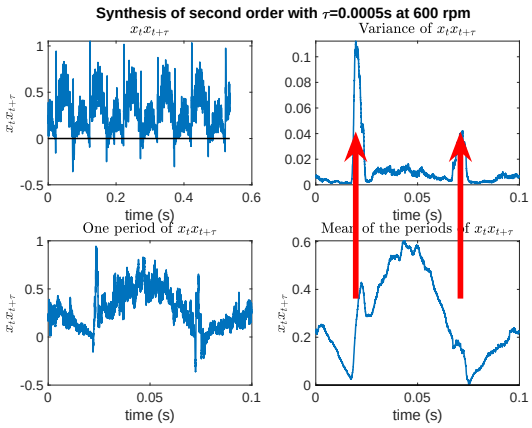


Figure 2.6: Second order analysis for a signal at 600 rpm with $\tau = 0.0005s$

Using $\tau = 0.0005s$ (50 samples as $f_e = 100$kHz) makes almost no difference for the signal acquired at 30 rpm (Figure 2.5). However, for this signal, $0.0005s$ corresponds to one 4000th of a period, and it is likely that this is almost similar as getting the second order moment. For the signal at 600 rpm (Figure 2.6), the pattern remains the same but now has some negative values, which correspond to the cases where one sample is before the discontinuity and the other one is after. Here, this is also representative of the anti-correlation obtained in this area, as the friction is close to being antisymmetric depending on velocity. However, due to the shifts between the periods, these negative values mostly disappear in the average period of the signal $\mathbf{x}_t\mathbf{x}_{t+\tau}$.

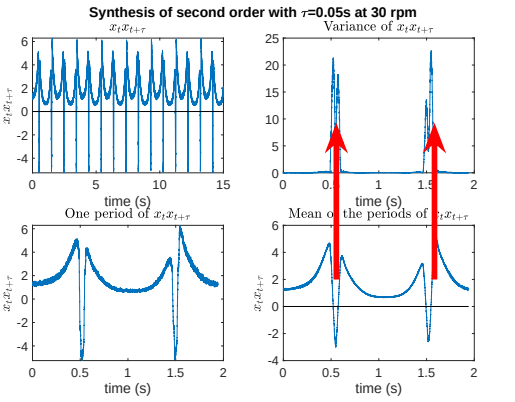


Figure 2.7: Second order analysis for a signal at 30 rpm with $\tau = 0.05s$

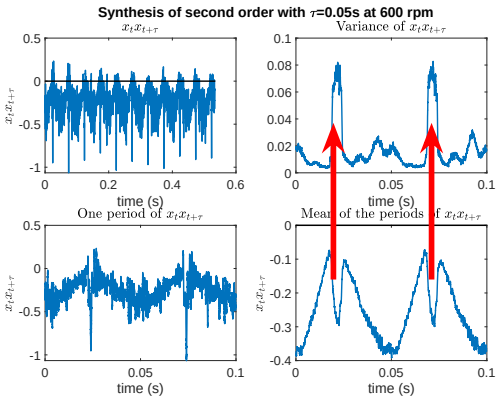


Figure 2.8: Second order analysis for a signal at 600 rpm with $\tau = 0.05s$

Finally, with $\tau = 0.05s$ (5000 samples), both signals (at 30 rpm in Figure 2.7, at 600 rpm in Figure 2.8) now clearly contain some negative values due to the transitions of the friction signals. Again, the negative values are largely attenuated in the average signal due to the shifts from one period to another.

As for the first order analysis, it would be possible to perform some statistical tests on the variance of the periods to confirm the cyclostationarity. Additionally to the problems of setting the thresholds and proportions, the second order cyclostationarity would also imply that the hypothesis is confirmed for any value of τ, and not only for the three values that we used here.

2.2 Computing the velocity

In this chapter, we are aiming to fit the parameters of some friction models linking the velocity at the contact point to the friction. In order to adjust such a model, we first need to have an estimate of the velocity. In our case, we already have some measurements of the position. However, these measurements are noisy and the sampling frequency f_e is high, and using a simple derivative filter, such as the one defined by $H(z) = f_e(1 - z^{-1})$, might result in high variations despite a low level of noise. Additionally, a filter would possibly require padding for computing the values at the extremities of the signal, which would damage the result.

In this section, we show that a sine wave is a good estimate of the measurements for the position. Based on the estimated sine wave for the position, the velocity can then be obtained by easily deriving the function.

The theoretical analysis for justifying the sine model is shown in Appendix C. In this section, we focus on the validation of the sine model for the measurements.

2.2.1 Methodology

2.2.1.1 Formulation of the problem

Based on the sine model demonstrated in Appendix C.2, we seek to adjust a sine wave to the signals of positions acquired by Ireis. We are looking for four parameters:

- A the amplitude of the sine wave;

- B the mean value;
- f_0 the frequency of the sine wave;
- φ the phase. This term is not present in Equation (C.8), but lets us define the sine wave without finding an origin time where $\varphi = 0$.

The position, as given by the model, is then:

$$\hat{l}(t) = A\sin(2\pi f_0 t + \varphi) + B \tag{2.1}$$

The velocity will then simply be:

$$\hat{v}(t) = 2\pi f_0 A\cos(2\pi f_0 t + \varphi) \tag{2.2}$$

For each signal, we have a set of observations $\{(t_n, l(t_n))\}_{n=0,\dots,N-1}$ with t_n a temporal measurement and $l(t_n)$ the associated position measurement. Our objective is to minimize the mean square error between the measurements and the parametric model of the velocity, i.e. we aim to resolve the following fitting problem:

$$\begin{pmatrix} \widehat{A} \\ \widehat{B} \\ \widehat{f_0} \\ \widehat{\varphi} \end{pmatrix} = \underset{\begin{pmatrix} A \\ B \\ f_0 \\ \varphi \end{pmatrix}}{\arg\min} \sum_{n=0}^{N-1} \left(l(t_n) - A\sin(2\pi f_0 t_n + \varphi) - B\right)^2 \tag{2.3}$$

2.2.1.2 Initialization of the parameters

For the parameters, the initial parameters are first estimated with the procedure of Algorithm 1 for f_0, A and φ. The expected frequency used by the algorithm is the setpoint frequency of the mechanism. For B, it would be possible to remove the sine wave component defined by the parameters f_0, A and φ, from the measured signal, and extract the average of the remaining signal. However, if the amount of periods is not an integer, then the last period would degrade the estimation. Here, we assume that the noise in the signal is white and that the signal contains at least one full period. Therefore, we initialize B as $\frac{\max_n(l(t_n))+\min_n(l(t_n))}{2}$. Let us notice that, it would have been possible to retain the values at the peak frequency, which is equivalent to stop Algorithm 1 before the dichotomous optimization. Considering the further optimizations of the parameters that we will perform

to minimize the error, the difference in the final results would probably have been the same or the differences would have been negligible.

2.2.1.3 Resolution of the problem

For solving Problem (2.3), we use the Levenberg-Marquardt algorithm [4, 5, 6], which is an iterative algorithm close to trust-region algorithms, with a linear approximation of the function based on the Jacobian matrix. In our case, a joint estimation of all parameters failed to estimate the frequency and phase correctly. Therefore, we found a pragmatic and efficient strategy close to alternating least squares by estimating the phase alone, while fixing the other parameters. The frequency is then estimated alone according to the same strategy, then all parameters are adjusted together. Generally, the first iteration of this procedure did not give significant improvements. Repeating the procedure a second time let us obtain a satisfactory model. Further iterations did not give significant improvements.

Considering this iterative optimization, it is possible that the initialization could have been done by simply using the original points of the Discrete Fourier Transform, without the dichotomous approach. However, the additional time consumption due to the dichotomous approach remained small and we therefore preferred having this complete procedure.

The computations are made with the *lsqnonlin* function of Matlab [134].

2.2.2 Visualization of the results

Three results of fitting the sine wave on the real position measurements are shown in Figure 2.9 (with a rotation speed of 15 rpm), Figure 2.10 (with a rotation speed of 120 rpm) and Figure 2.11 (with a rotation speed of 750 rpm). As shown on these figures, the acquired measurements have an extremely low noise level and it is therefore easy to adjust the model on the measurements. Even though the initial model is extremely close to the target signal and could be used directly without further refinement, the fitting procedure further improves the quality of the estimation.

For most parts of the signals, the observations confirm that the sine wave is an appropriate model for the position signal. It is still possible to observe some minor variations , such as pointed out by the zoom in Figure 2.9, but the impact on the velocity should remain negligible.

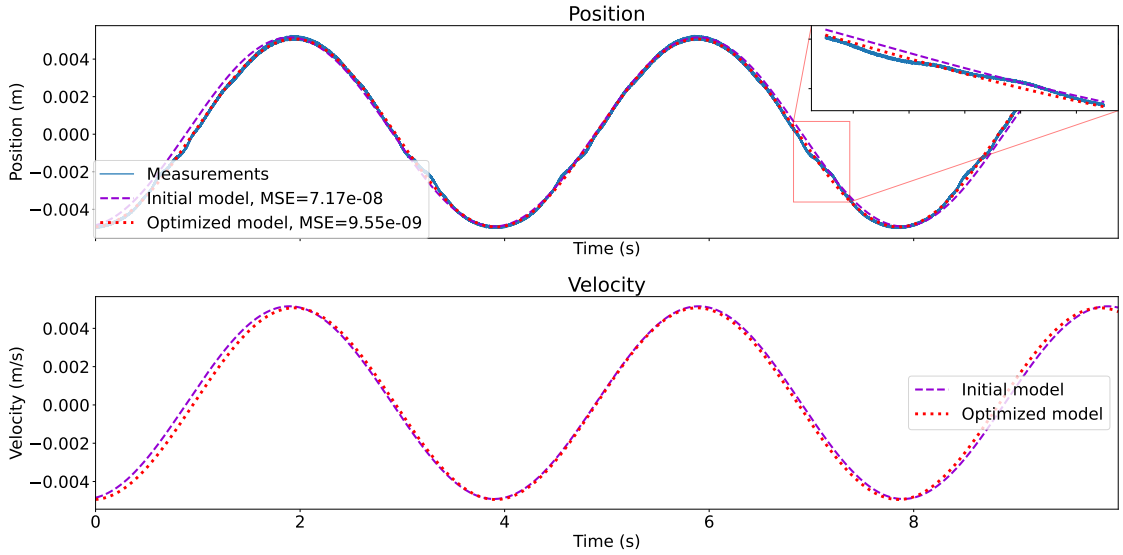


Figure 2.9: Curve fitting of the position at 15rpm.
Despite some local deviations, the approximation by a sine wave is satisfactory for most parts of the signal.

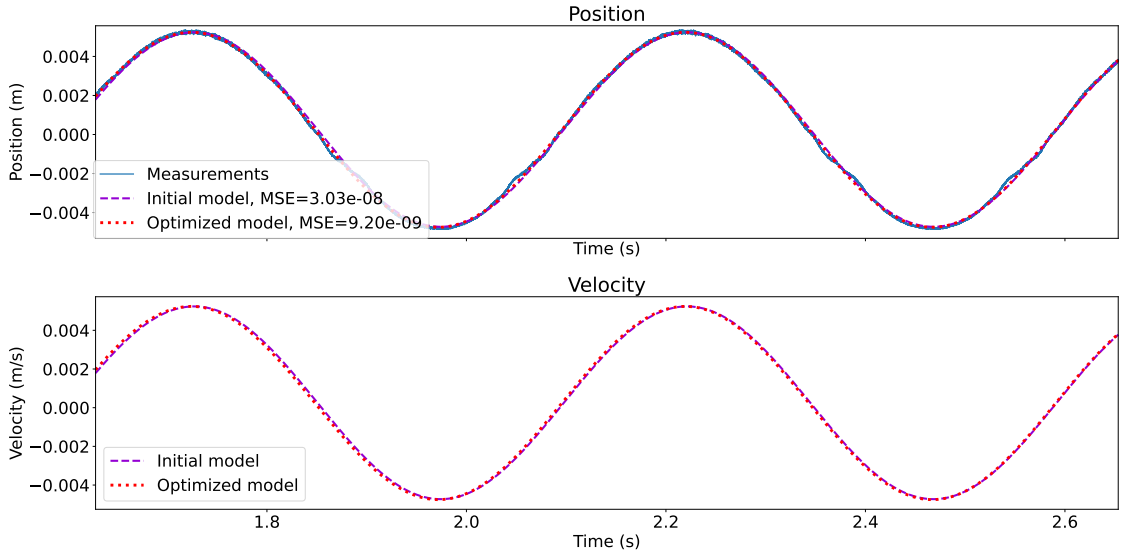


Figure 2.10: Curve fitting of the position at 120rpm

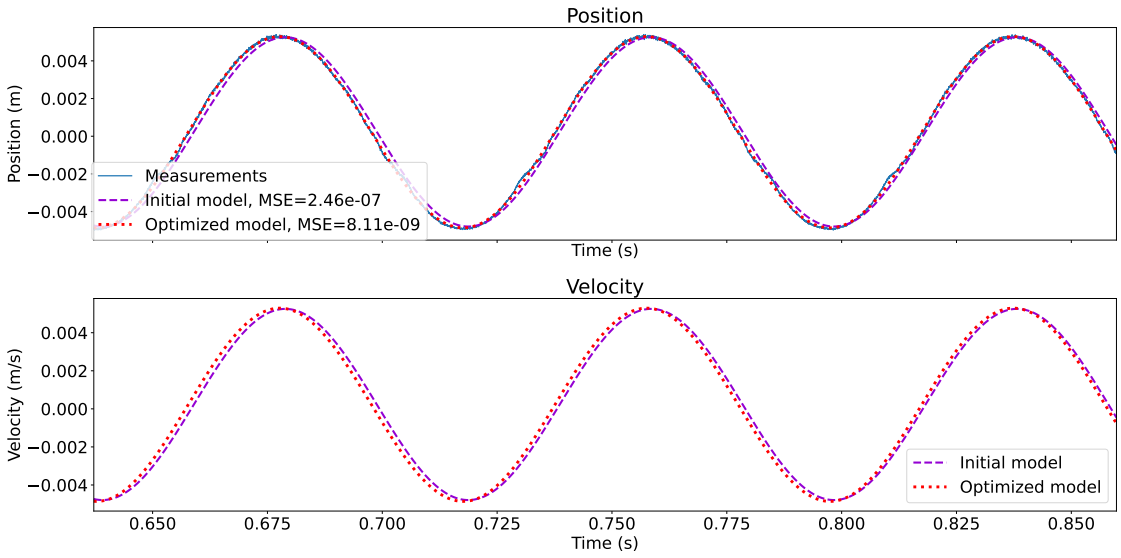


Figure 2.11: Curve fitting of the position at 750rpm

2.3 Model fitting for denoising

In Section 2.2, we obtained extremely close estimates of the position measurements based on the sine wave, and we could therefore easily obtain an estimate of the velocity signal. It is now possible to fit a viscous friction model linking the velocity to the recorded friction signal (or the friction coefficient). In this section, we apply this parametric model fitting for two different friction models, showing both the validity of such models for representing the friction coefficient, but also their limits concerning some phenomenons occurring in the signals and generalization to all kinds of lubrication.

2.3.1 Methodology

2.3.1.1 Parametric models for friction measurements

Andersson model [7]

In [7], several models linking the friction coefficient μ to the velocity v are proposed. Among them, we use the one defined as:

$$\mu(v) = \left(\mu_c + (\mu_s - \mu_c)\, e^{-\left(\frac{|v|}{v_s}\right)^i}\right) \times \text{sign}(v(t)) + k_s \times v. \tag{2.4}$$

In this model, the parameters μ_c, μ_s, k_s and v_s have to be estimated for a specific experimental setting, i.e. they will change when using different materials or lubricants. Here, μ_c is the Coulomb friction coefficient, obtained for the transition between mixed and hydrodynamic lubrications, μ_s is the static friction coefficient, corresponding to boundary lubrication (the different lubrication states can be observed in Figure 0.3). v_s is the sliding speed coefficient and k_s is the viscous friction coefficient.

Here, we will denote this model as the Andersson model even though it has also been defined similarly in other works, such as [131]. In [135], a similar model is used with $i = 1$. In [136], several values are proposed including $i = 1$, depending on the materials of the surfaces in contact. Here, we could include i among the unknown parameters when fitting the measurements. However, based on these references, $i = 1$ seems to be an adequate value for most cases, and we will therefore use this value, which should offer satisfactory results for the mixed and hydrodynamic lubrications. The model should however have more difficulties fitting the boundary lubrication as the exponential term will strongly vary even for small values of v.

Additionally, our signals also seem to contain an offset value. Therefore, we aim at fitting the following model to the measurements:

$$\mu(v;\mu_c,\mu_s,\mu_0,k_v,v_s) = \left(\mu_c + (\mu_s - \mu_c)\, e^{-\left(\frac{|v|}{v_s}\right)^i}\right) \times \text{sign}(v) + k_s \times v + \mu_0. \tag{2.5}$$

Hess model [8]

In [8], a model is defined for representing all three lubrications. Actually, [8] uses four lubrications as the mixed lubrication is decomposed into a mixed lubrication and an elasto-hydrodynamic lubrication). This model links the friction force μ to the velocity v, the load w and the viscosity η:

$$\mu(v,w,\eta) = \frac{\mu_b}{1 + c_1\left(\frac{\eta v}{\sqrt{wE}}\right)^2} + c_2\frac{\eta v L}{w} \tag{2.6}$$

with E the Young's modulus, L is the contact length, μ_b represents the boundary lubrication, and c_1 and c_2 are parameters for this model. In our experiments, we do not know E, L and μ_b, therefore, we include these parameters in the optimization. η and w are unknown as well, but they are assumed to remain the same for signals acquired with the same lubricant. Therefore, we can include these parameters in the estimation. The model we adjust is then:

$$\mu(v;\mu_b,\mu_0,\alpha,\beta,\gamma) = \frac{\mu_b}{1 + \alpha\,(\beta v)^2} + \gamma\beta V + \mu_0 \tag{2.7}$$

with $\alpha = \frac{c_1 w}{E}$, $\beta = \frac{\eta}{w}$ and $\gamma = c_2 L$. μ_0 is used for the offset, as for the Andersson model.

We will denote this model as the Hess model.

2.3.1.2 Fitting the parameters of the models

For fitting the parameters of the model, we constitute our measurement's dataset as a concatenation of several recorded signals, obtained with the same lubricant but various rotation speeds. The objective of doing so is to fit global parameters that are consistent with various rotation speeds instead of having a model that is specific to a single rotation speed, which would imply using only some of the terms of the model. For example, both models use a linear term for the hydrodynamic lubrication, thus making the estimation for signals where this lubrication does not occur would result in having a null coefficient. Fitting on a single signal would therefore result in more specific estimated values but the obtained parameters would have little (or not at all) physical meaning.

Here, we use the *lsqnonlin* function of Matlab [134] for implementing a trust-region algorithm. The criteria is approximated by a quadratic function based on its gradient and Hessian inside the trust region. The objective is to solve:

$$\widehat{\theta} = \arg\min_{\theta} \sum_{n=0}^{N-1} \left(\mu_\theta(\mathbf{v}_n) - \boldsymbol{\mu}_n\right)^2 \tag{2.8}$$

where θ is the set of searched parameters, μ_θ is the model defined by Equation (2.6) or Equation (2.7), $\boldsymbol{\mu}$ is the vector obtained by concatenating all acquired signals and $\mathbf{v}$ is the vector obtained by concatenating all estimated velocity signals based on the sine wave estimation of Section 2.2.

Here some parameters can have very different scales. For example, in [8], $c_1 = 1.43 \times 10^{17}$, $c_2 = 819$ and $\mu_b = 0.145$. Therefore, the trust-region approach applied to all parameters simultaneously could fail optimizing some of the parameters, as their variations inside the trust region would be negligible. One possibility would be to normalize all parameters so that they have similar scales. However, this would require having a good *a priori* knowledge of what those parameters should be. Here, we rather use a similar approach as for fitting the sine wave for position in Section 2.2. For one iteration, each parameter is first optimized with all other parameters fixed then all parameters are optimized together. For ease of implementation, we use the *lsqnonlin* function even for parameters that could be solved linearly. We perform ten iterations of this procedure. Even with this method, it appears that some of the parameters have almost no variations during optimization. For this reason, we have to perform the initialization of the parameters mostly manually, by trying different initializations until one that allows satisfactory optimization is obtained. Only the parameters μ_s and μ_c of Andersson model are initialized based on the measurements, with $\mu_s = \mu_c = \max(\boldsymbol{\mu})$, i.e. we assume that the friction coefficient reaches its maximal

Table 2.1: Parameters for the Andersson model

	Initial model	After optimization
μ_c	Max. of measurements	0.0077
μ_s	Max. of measurements	0.092
v_s	0.1	0.0040
k_v	1000	0.12
μ_0	0	-0.0076

Table 2.2: Parameters for the Hess model

	Initial model	After optimization
μ_b	0.1	0.079
α	1.2×10^7	6.3×10^6
β	0.1	0.094
γ	10	1.67
μ_0	0	-0.0076

value in the boundary lubrication and the only contribution at the transition from mixed to hydrodynamic lubrication is due to the linear term. For most parameters, the initialization has a limited influence on the final result. Therefore, our "first try" value could be used to obtain satisfactory result. However, for the α parameter of the Hess model, the convergence to an adequate value required several tries for the initialization. This might be because the large magnitude of this parameter implies that the size of the trust region is significant only after several iterations. It is then possible that the first iterations do not have any impact for this parameter. One possible solution might be to normalize this parameter when optimizing it individually.

We show in Table 2.1 (resp. Table 2.2) the initialization and obtained parameters for the Andersson (resp. Hess) model, corresponding to the results presented in Section 2.3.2. The variations between initialization and final model confirm that for most parameters, the initialization did not require a deep exploration. Let us notice the consistency between the two models concerning the μ_0 parameter used for the offset, as well as the consistent order of magnitude for μ_b and μ_s which define the values obtained in the boundary lubrications for both models.

2.3.2 Results

Here, we present some results obtained for measurements where the lubricant used is pure glycerol. Let us notice that a single set of parameters is obtained for all rotation speeds. For this lubricant, this set is the one presented in Table 2.1 for the Andersson model, and in Table 2.2 for the Hess model.

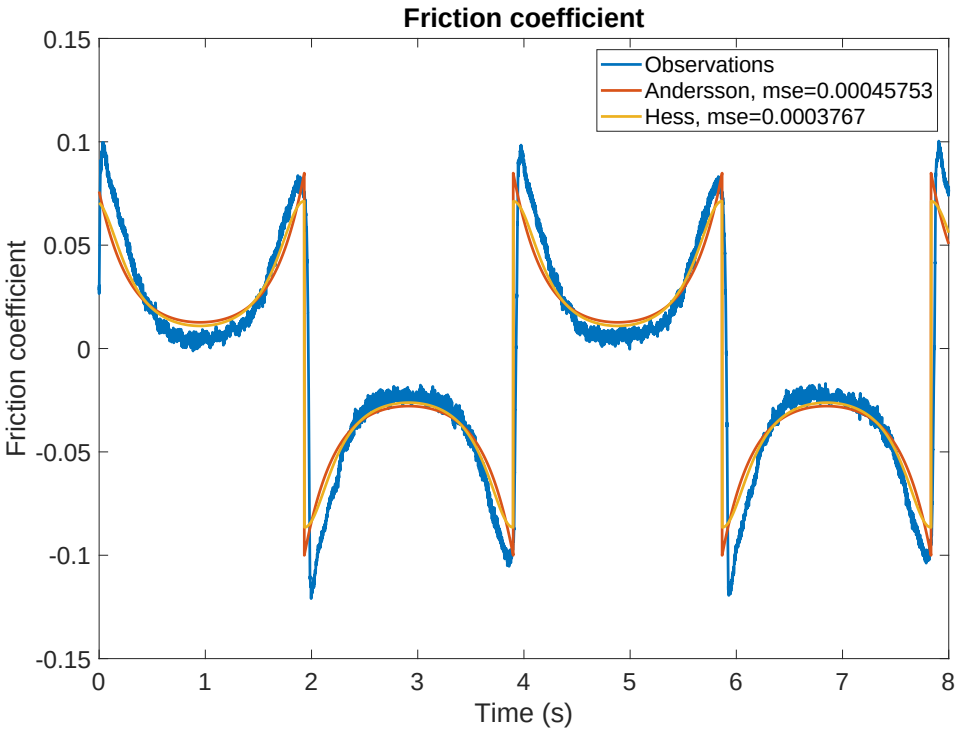


Figure 2.12: Model fitting of a friction signal at 15 rpm
The Hess and Andersson curves are obtained with the parameters found during the estimation

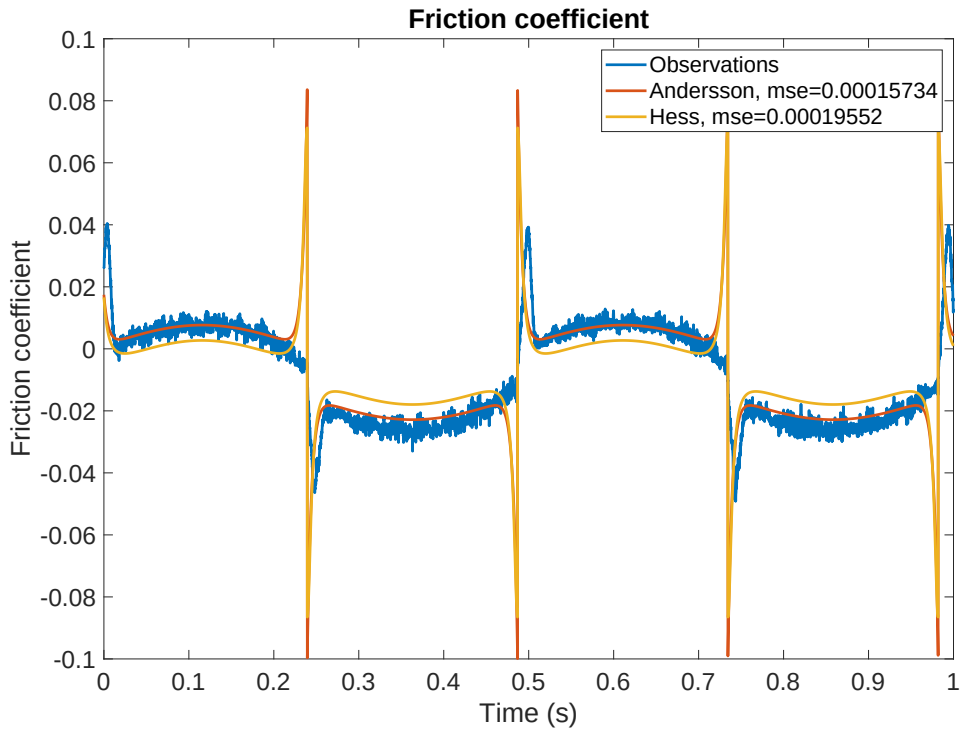


Figure 2.13: Model fitting of a friction signal at 120 rpm
The estimated parameters of the models are the same as in Figure 2.12

In Figure 2.12, we show the results of the two models for a signal at 15 rpm. Both models manage to obtain a shape similar to the expected one. The Andersson model seems to get better results than the Hess model for the magnitude of the peak, but the Hess model obtains better results in terms of MSE. This might be due to the smooth part, where the Hess model seems a bit better, but also to the fact that the peaks are not exactly discontinuities in the real signals. As the models have an instantaneous transition from negative to positive values, while the response time of the sensor implies a smoother transition in the real signal, large errors are obtained in these areas for the Andersson model.

Figure 2.13 shows a result at 120 rpm. This time, the Andersson model clearly outperforms the Hess model, both in terms of visual results and MSE. However, both models obtain a peak with a much larger magnitude than the real one. This can be easily explained. Even though the velocity is extremely low, the lubricant film that appeared between the two surfaces at higher velocities is still partially present. Therefore, even though the viscosity of the lubricant remains the same, the resulting viscosity at the contact surface has changed. This phenomenon illustrates a "memory" in real signals, that the models are not able to deal with. Similar observations can be made for signals at 300 rpm (Figure 2.14).

At higher rotation speeds (750 rpm for Figure 2.15), the Andersson model still outperforms the Hess model.

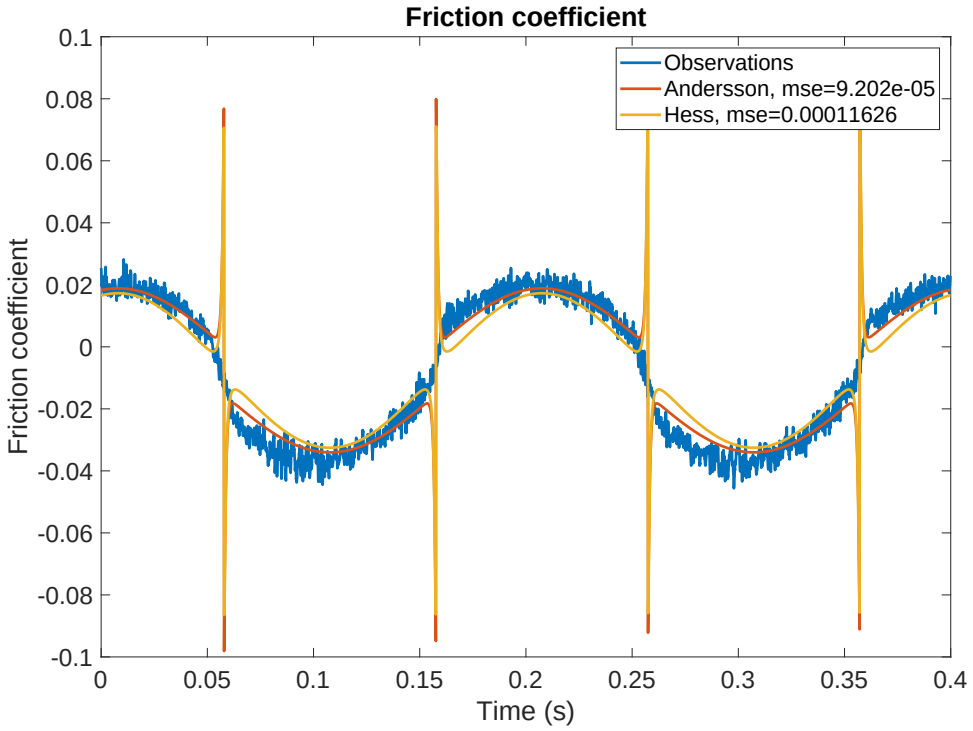


Figure 2.14: Model fitting of a friction signal at 300 rpm
The estimated parameters of the models are the same as in Figure 2.12

In all those examples, it is possible to observe an hysteresis in the real friction measurements. This was particularly significant in the Figure 2.13, where a peak appeared after each stop, but not before. Even though this is less significant in Figure 2.15, it appears that the friction coefficient gets lower values during the deceleration than during the acceleration. This is likely due to the lubricant film being impacted by the motion. In [8], the model is optimized to deal with this hysteresis by adding a constant delay in the velocity. However, in their case, the acceleration is piecewise constant. In our case, using a delay would not solve all problems encountered in this study (using a delay will just result in changing the position of the symmetry center, but the symmetry will still be there), and a more sophisticated term depending on the acceleration would be needed.

In Figure 2.16, we show the MSE of each model depending on the rotation speed used for the acquisition. The Andersson model outperforms the Hess model for almost all rotation speeds except low ones. However, by checking all samples at all rotation speeds, the Hess model gets a better global MSE, which might be due to the large errors of Andersson model due to the higher sensitivity of the Andersson model to the response time of the sensor.

Lastly, it is possible to observe the obtained models in a velocity-friction domain (Figure 2.17). This representation clearly illustrates that the slope of the Hess model for the linear term representing the hydrodynamic lubrication is too high, which should result in

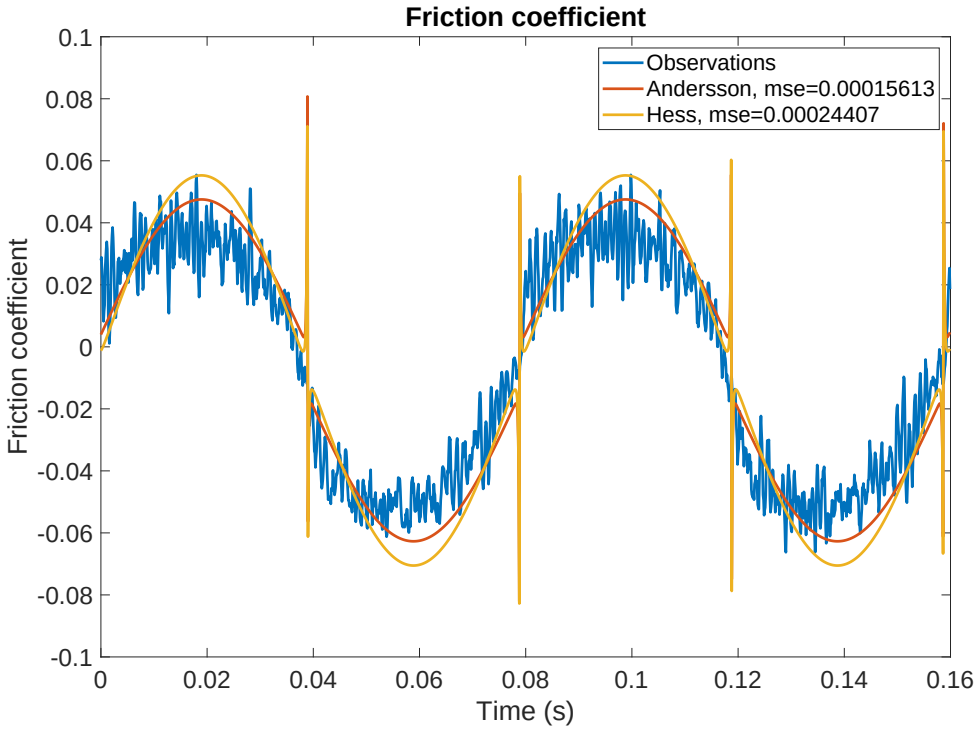


Figure 2.15: Model fitting of a friction signal at 750 rpm
The estimated parameters of the models are the same as in Figure 2.12

even much larger errors for higher rotation speeds, even though it would then be possible to adjust a new model including the new measurements. This figure also show that both models manage to retrieve the behavior of the Stribeck curve (Figure 0.3), at least concerning the mixed and hydrodynamic lubrications.

2.4 Conclusion

In this chapter, we have applied a model fitting approach to perform a first denoising of the friction signals. Even though the results share similar patterns with the original friction signals and are able to retrieve shapes close to the expected ones, the current state of the denoising is clearly not satisfactory, for several reasons. In terms of quality of the results, the models are not able to fit properly all parts of the data and tend to overestimate the discontinuities. Additionally, the method for obtaining these results is not easily generalizable. Here, we could apply it because we had measurements concerning the position of the mechanism, a sine wave model for estimating the position and the velocity, and models for linking the velocity to the friction coefficient. Without all those informations, fitting the models would not have been possible. Here, we are aiming for methods that could

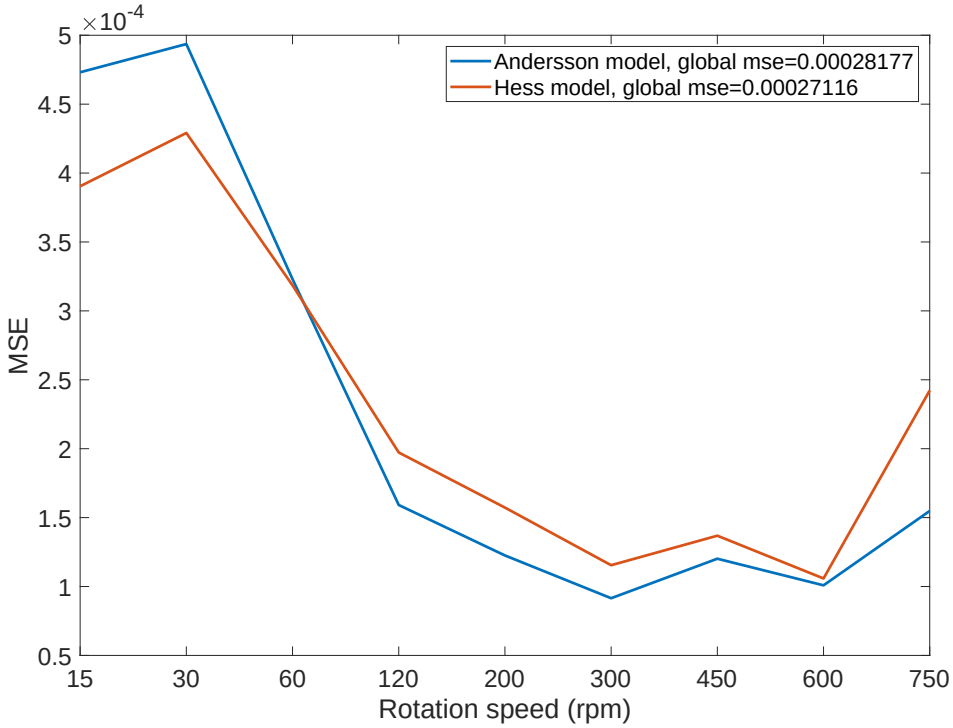


Figure 2.16: MSE for each rotation speed.
The same parameters are used for all rotation speed.

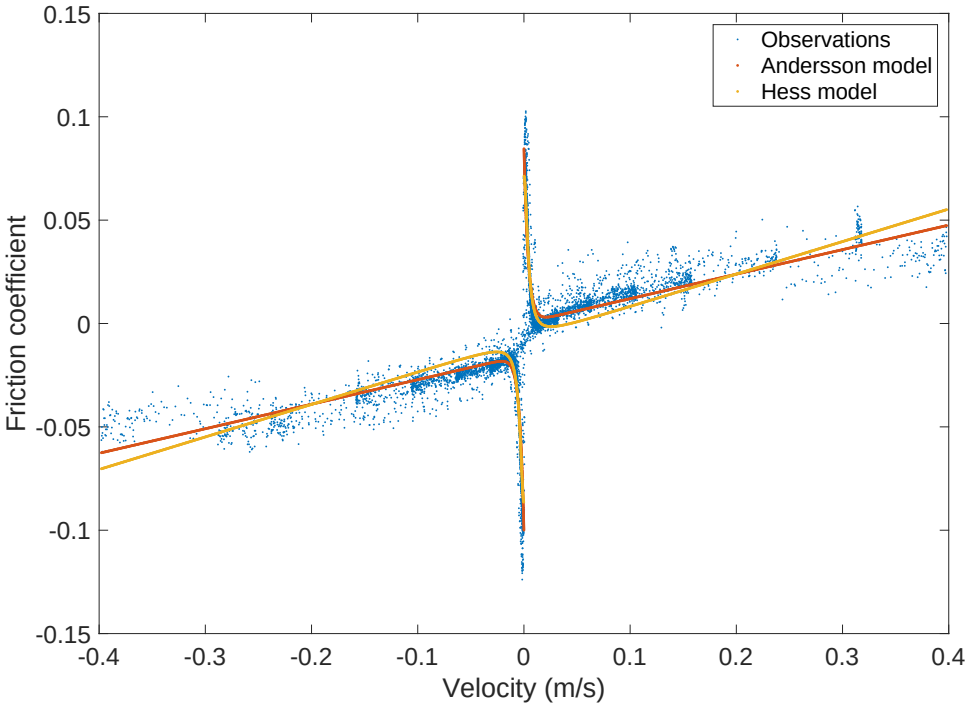


Figure 2.17: Representation of the friction coefficient as a function of velocity

generalize to various periodic signals, including some where only the noisy signals would be available. Therefore, methods that are able to adapt to the specificities of each signal are required. Such methods would also be interesting in the case of the friction signals, as some of the phenomena are not well represented in the models. This was for example the case of the hysteresis.

However, despite not being satisfactory for denoising the signals, the work proposed in this chapter has shown that the friction models of Hess [8] and Andersson [7] are able to generate signals that have similar shapes as the friction signals that we are working with. As we do not have any ground truth for the friction signals, such models can be helpful, for training supervised methods, but also for generating signals on which we can perform quantitative evaluations. In the following sections, the model for generating training signals will be defined in another way, but we will use the Andersson model to generate evaluation signals with a different shape as training data. We focus on the Andersson model for both its better results (visually for all signals, quantitatively for signals with high rotation speeds) and the large adaptability of this model. In [7], other models are proposed based on the one we used. We tried to implement one using a tanh instead of the sign function, as proposed by the authors, but we did not manage to fit it properly for our signals. We also tried weighting both exponential and linear terms with sigmoids to ease the separation of the different lubrication states. The obtained fittings were sometimes (rarely) visually better, but remained equivalent or even a bit worse in terms of quantitative results, and were hardly explainable in terms of physical meaning. Lastly, it would also be possible to optimize the parameter i, which should help improving the estimation of the boundary lubrication.

3 Total variation based regularization for WaveNet denoisers

In the previous chapter, we proposed a first approach for denoising friction signals based on a curve fitting approach. In this chapter, we aim to propose a first approach using Deep Learning for denoising our signals. For overcoming the lack of ground truth, some methods use unsupervised methods [137, 138, 139]. Among those unsupervised methods, denoising autoencoders have had a large success [97, 140]. Autoencoders are networks consisting of an encoder which reduces the dimension of the input and a decoder which aims to reconstruct the input based on the features extracted by the encoder. As the feature space is of lower dimension than the input, it should contain less information and the encoder should learn to extract the most relevant information. The assumption behind these methods is that the removed information will be mostly noise.

This assumption might be true for white noise, but colored noises, especially low-frequency ones, might result in patterns that are encoded by an autoencoder. Therefore, we rely on a supervised method for training networks. In our case, such methodologies are not immediate to apply as they imply using a dataset with both noisy and clean signals. Here, we have a limited amount of noisy signals and no clean signal.

To overcome this difficulty, we propose to create a dataset of synthetic signals. Synthetic data have already been used in other Deep Learning applications such as applications on images [141, 142] or videos [143, 144]. However, in these works, the generation of the data relies on a previously existing database of objects. In our case, we create the data on the fly, which allows reducing the physical storage.

We propose a new loss term, based on TV (see Section 1.4.3). This new loss term is meant to avoid over-fitting discontinuities contained in our synthetic signals, and should also help the networks generalize to multiple colors of noise, as we will detail in Section 3.2.2.

Let us notice that for this chapter, as well as for the next chapter, no seed for random number sampling was configured for generating the data (both for training and evaluation), which means that the results are not exactly reproducible, but also means that we cannot rely on a specific seed that gives good results and each new experiment will let us have a better idea of the global result. Especially, the results are expected to be close to the ones obtained in [24]. As some of the results have been performed again, this might result in some changes

compared to the results published in [24]. In most cases, the changes are minor and the observations remain similar, but there are a few cases were a large variation is observed, showing the instability of some models.

In Section 3.2, we present our methodology for defining the neural networks and training them despite the unavailability of real data with ground truth. In Section 3.3, we show some applications of our method on the friction signals. In Section 3.4, we show an application of our method on some specific signals containing both discontinuities and smooth parts, allowing us to perform a comparison with another method, based on Wavelets and total variation [10]. Lastly, in Section 3.5, we apply our 1D neural networks on FIB image sequences. In this section, we apply the networks on the different dimensions of the sequences, allowing us to check which dimension is the most relevant for our networks and how some hyperparameters deal with each dimension.

Communications linked to this chapter

International conferences papers

[24] J. Rio et al. “WaveNet based architectures for denoising periodic discontinuous signals and application to friction signals”. In: *2020 EUSIPCO*. IEEE. 2021, pp. 1580–1584

Presentations in international conferences

[27] J. Rio et al. “Wavenet-Based Architectures Adapted to the Denoising of FIB/SEM Image Sequences”. In: *2021 IEEE 17th International Conference on Automation Science and Engineering*. IEEE. 2021

3.1 Denoising signals with strong discontinuities

3.1.1 Problem statement

In this chapter, we want to solve the task of denoising in the case of additive noise, i.e. we want to extract an approximation $\hat{\mathbf{y}}$ of the clean signal $\mathbf{y}$ from a mixture $\mathbf{x} = \mathbf{y} + \mathbf{e}$, with $\mathbf{e}$ the noise signal.

We are aiming for denoising signals containing strong discontinuities. However, we assume that these discontinuities remain rare and other parts of the signals are assumed to be smooth and varying slowly, at least compared to the sampling frequency, i.e. there are thousands or even tens of thousands of samples between two discontinuities. While this property will not be used in this chapter, we also assume that the signals are periodic. In some of the applications of this chapter, this will not be the case, but it is still possible to consider a non-periodic signal as one period of a periodic signal. As a result, the clean signal $\mathbf{y}$ can be modeled as the discrete acquisition of the continuous signal $s(t)$ defined as:

$$s(t) = \gamma_1 c(f_1, t, \varphi_1) + \gamma_2 \sum_{k=0}^{+\infty} a_k \sin(2k\pi f_2 t + \varphi_{2,k}) \tag{3.1}$$

with $c(f_1, t, \varphi_1) = \sum_{k=0}^{+\infty} \sin\left(2(2k+1)\pi f_1 t + \varphi_1\right)$ a square wave, which contains periodic discontinuities. These definitions are obtained through the decomposition in Fourier series. The signal $\mathbf{y}_n$ can then be obtained as:

$$\mathbf{y}_n = s\left(\frac{n}{f_e}\right) = \gamma_1 c\left(f_0, \frac{n}{f_e}, \varphi_1\right) + \gamma_2 \sum_{k=0}^{+\infty} a_k \sin(2k\pi f_2 \frac{n}{f_e} + \varphi_{2,k}). \tag{3.2}$$

Here, we add some other assumptions to these signals. First of all, we assume $f_1 = f_2 = f_0$. Additionally, we assume $a_0 = 0$ and $a_1 = 1$. While an offset appeared in the friction signals, this can be tackled off by normalizing the signals first. The assumption $a_1 = 1$ can be included in a modification of the term γ_2. Lastly, as we assume slow variations outside of discontinuities, we can say that the coefficients a_k are negligible for high values of k and the clean signal can therefore be approximated as:

$$s(t) = \gamma_1 c(f_0, t, \varphi_1) + \gamma_2 \sin(2\pi f_0 t + \varphi_{2,1}) + \gamma_2 \sum_{k=2}^{k_0} a_k \sin(2k\pi f_0 t + \varphi_{2,k}) \tag{3.3}$$

where k_0 defines the last non-negligible component. $\mathbf{y}$ is then defined as:

$$\mathbf{y}_n = s\left(\frac{n}{f_e}\right) = \gamma_1 c\left(f_0, \frac{n}{f_e}, \varphi_1\right) + \gamma_2 \sin(2\pi f_0 \frac{n}{f_e} + \varphi_{2,1}) + \gamma_2 \sum_{k=2}^{k_0} a_k \sin(2k\pi f_0 \frac{n}{f_e} + \varphi_{2,k}). \tag{3.4}$$

Concerning the noise in the signals, we assume it to have mostly short-time correlations.

For the context of the project IMOTEP, the friction signals can have various frequencies f_0 and various initial SNR. While it would be possible to build a network for each setting, especially for the frequency, we are aiming for a method that is able to generalize well to various settings. Therefore, developed networks should be trained with various frequencies in the same training set.

Lastly, we need to tackle the absence of ground truth.

3.1.2 Global approach for solving this problematic

Our approach for solving the problem consists in three steps:

- We generate synthetic data having the expected properties of Model (3.4). It is possible to use a different model for evaluation and training. The details of generation of training data, which will be common to all applications, are given in Section 3.2.3. The data will be generated with various frequencies and SNR to achieve our objective of generalization. In our case, we will use a model for training and validation. For evaluations in the case of friction signals, we will perform some evaluations with the same model but different frequencies, and another evaluation with a friction model (the Andersson model presented in Section 2.3.1.1), with the same range of frequencies and SNR as in training set. The Model (3.4) does not correspond exactly to the data obtained with our implementation of the Andersson model, which might contain more discontinuities. However, it should remain consistent, at least locally.

- We use a 1D fully convolutional denoiser. Here, a long receptive field is needed for observing a large context around the denoised sample, but using a too long receptive field might be dangerous as the assumptions made about the training data might not be hold when looking a long context. Some details about the defined models are given in Section 3.2.1, then details for each applications will be given in the corresponding sections.

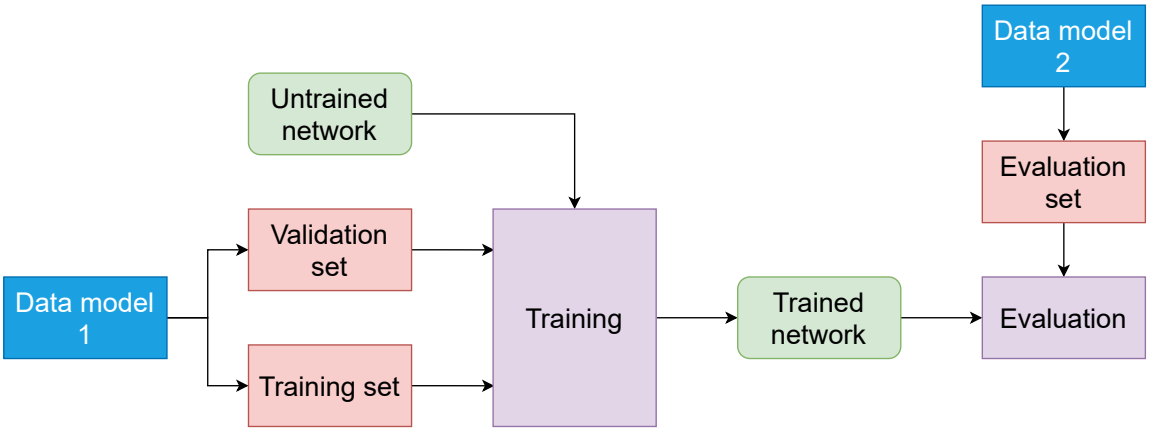


Figure 3.1: Overview of the procedure applied in this chapter

- During the training of the model, we assume that using colored noise would not be helpful as there would be no way for the model to know whether it is useful information from a signal component appearing locally or a colored noise. However, based on the assumption that we are dealing mostly with short-time correlations, we define a new loss term, as detailed in Section 3.2.2.

Figure 3.1 gives an overview of the general procedure used along this chapter.

3.2 Methodology

3.2.1 Architectures for denoising

In this chapter, we base all neural networks on the WaveNet for speech denoising [9], which is a fully convolutional neural network performing denoising directly in the wave domain. This model uses dilated convolutions for obtaining a sufficiently long receptive field with a limited amount of parameters. For more details about this architecture, refer to Section 1.5.6.

In this chapter, we use the original architectures, but we also build several other architectures with the same methodology. These modifications impact the receptive field, which can be shortened, the depth of the network, its amount of parameters and its memory use.

As we use different architectures depending on the application, we will give details about the architectures used for a specific application in the corresponding sections.

3.2.2 Loss term for improved generalization

Here, we want to perform a supervised training. It is therefore necessary to use a reconstruction term measuring a distance between the output of the network $\hat{\mathbf{y}}$ and the clean signal $\mathbf{y}$. For that, several terms are possible. In [9], it is observed that using the MAE offers better results than using the MSE. As our first experiments showed similar observations, we decided to follow the same process concerning the reconstruction term.

Here, we are dealing with signals having strong discontinuities, and smooth variations everywhere else. Using the reconstruction term only might result in over-fitting the discontinuities and preserving all strong variations when denoising. However, except for discontinuities, strong variations in real signals might result from colored noises rather than the source of interest. In this chapter, rather than including colored noises in the training set, we decide to use white noise and add a TV-based term for penalizing strong variations, which, outside of discontinuities, should originate from colored noises. The resulting loss term is:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{N} \sum_{n=0}^{N-1} |\hat{\mathbf{y}}_n - \mathbf{y}_n| + \frac{\gamma}{N-1} \sum_{n=1}^{N-1} |\hat{\mathbf{y}}_n - \hat{\mathbf{y}}_{n-1}| = \text{MAE}(\hat{\mathbf{y}}, \mathbf{y}) + \frac{\gamma}{N-1} \text{TV}(\hat{\mathbf{y}}) \quad (3.5)$$

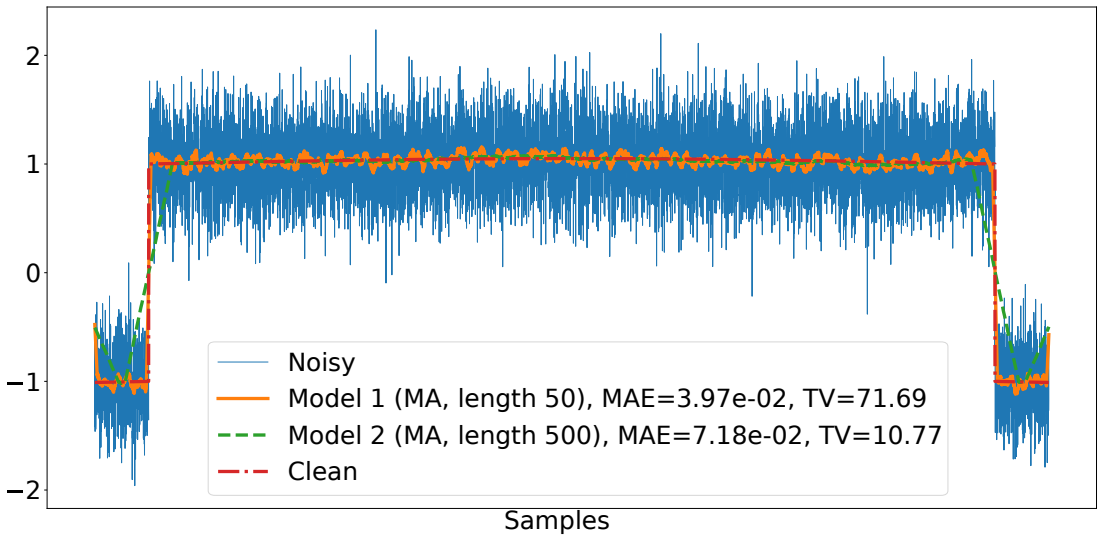


Figure 3.2: Illustration of the results of two models with white noise. Without consideration of TV, the shorter MA would be selected.

with γ a weight for balancing both contributions. The idea is to allow variations to appear in the denoised signal only if they result in a sufficiently high improvement of the MAE for compensating the TV penalty.

An illustration of the interest of such terms is shown with Figures 3.2 and 3.3, where the same models, two moving averages with different lengths, 50 samples and 500 samples respectively, are used for denoising signals with white noise (Figure 3.2) and relatively low frequency noise (Figure 3.3), similar to a noise that could appear in real signals. The model 1, which is a moving average of length 50 gets better results with white noise in term of MAE and training with only MAE on such data would therefore result in selecting this model rather than model 2, which is a moving average of length 500. However, model 2 gets a much lower total variation and a better generalization to the signal with colored noise, therefore, it might be interesting to choose this model rather than model 1, which is made possible with the TV penalty depending on the value of γ. Here, the displayed values of MAE and TV are the average of 1000 realizations of the noisy signal. Let us notice that in this case, there is a counterpart in using a longer moving average, which is the fact that the discontinuities are smoothened, which actually explains the bad result of this model with white noise. These observations prove that defining the weights between the two terms is equivalent to finding a compromise between the ability to preserve discontinuities and the smoothness outside of discontinuities. In a neural network, this total variation constraint in the loss should help the network learning to use its entire receptive field to identify smooth parts.

Similar regularization terms were proposed in [145] and [146] for image processing.

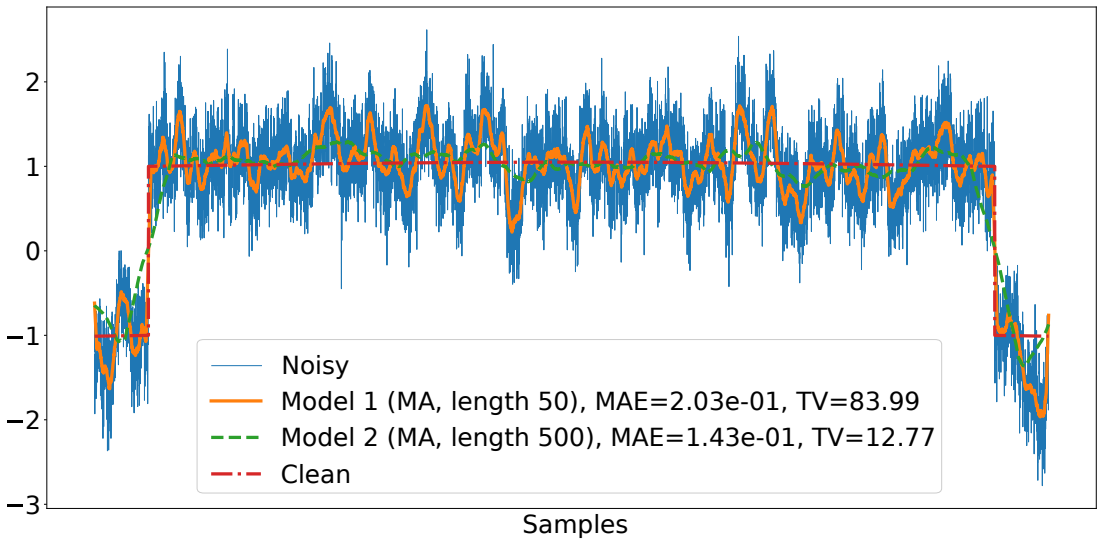


Figure 3.3: Illustration of the results of two models with colored noise. The best model is now the longer MA.

3.2.3 Synthetic dataset for training

As previously said, our denoisers are dedicated to smooth periodic signals presenting periodic discontinuities. To obtain signals having similar properties and easy to generate, we combine square waves for discontinuities with sine waves for smooth variations. The resulting clean signals have the shape:

$$\mathbf{y}_n = \lambda c\left(f_0, \frac{n}{f_e}, \varphi_1\right) + (1-\lambda)\sin\left(2\pi\frac{f_0}{f_e}n + \varphi_2\right) \tag{3.6}$$

with $(\varphi_1, \varphi_2) \in [0, 2\pi] \times [0, 2\pi]$ the phases of the two components, f_0 a frequency shared by the two components, λ a random weight in $[0, 1]$. This model is the same as Model (3.4) where $k_0 = 1$, which is also equivalent to enforcing $a_k = 0$ for $k \geq 2$, and the two parameters γ_1 and γ_2 are both defined with a single parameter λ to ensure $\gamma_1 + \gamma_2 = 1$. The frequency f_0 is randomly selected from a predefined set of frequencies for each generated signal.

The noisy signal $\mathbf{x}$ is obtained by adding a white Gaussian noise on the clean signal. The power of the noise is defined so that the SNR of the noisy signal reaches a value randomly selected from a set of predefined possible initial SNR.

All generated signals contain the same amount of samples. Some examples of generated signals can be seen in Figure 3.4. Despite the simplicity of the model, the generated data can have various shapes, including some similar to the ones observed in real friction signals. Here, the presence of discontinuities will have an impact on the models, but as we

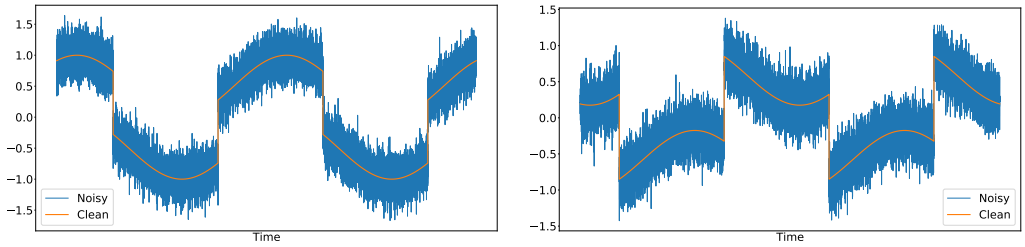


Figure 3.4: Examples of signals generated with Equation (3.6)

will explain later, the periodicity will not be exploited in this part, despite the data being generated with a periodic model.

Additionally to the signals generated following Equation (3.6), we generate signals where the clean signal is a constant. Using "noise-only" signals was shown efficient by [9]. Here, we use 50% of such data, which is a larger amount than proposed in [9]. We chose this amount because we find it useful for better generalization in our case.

Additionally to the training signals, we also generate validation signals in a similar way. We assume that the smooth parts of the signals with a low fundamental frequency will already be sufficient for evaluating the performance of the networks on almost constant signals during the validation. Therefore, we do not use noise-only data in the validation set, but only sums of square waves and sine waves.

3.3 Experiments for friction signals

3.3.1 Experimental setup

3.3.1.1 Details of the architectures

For this section, we use four different architectures based on the "Wavenet for speech denoising" [9]. Additionally to the original architecture, we propose three smaller architectures, as detailed in Table 3.1. The gain of these smaller architectures is not only in terms of trainable parameters, but also in GPU memory use, as reducing the amount of channels implies a reduction of signals stored during processing, as well as reducing the amount of layers. As a consequence, it is possible to process longer signals without having to cut them in several sub-signals. Considering the simplicity of the model for generating training data, reducing the amount of parameters might also reduce the risk of over-fitting. The counterpart is a reduction of the receptive field, which implies the observation of a smaller context, in which

Table 3.1: Details of the architectures

	Original [9]	**Light-Full**	**Medium**	**Small**
Residual layers per stack	10	10	8	6
Stacks	3	3	2	2
Residual layers channels	128	64	64	64
Channels in the last layers	2048,256	256,64	256,32	256,32
Parameters	6.3M	1.1M	600K	460K
Receptive field (samples, Equation (1.24))	6143	6143	1025	257

the assumptions made for generating the synthetic training data should be verified. The models with a long receptive field are expected to be better in terms of smoothness outside of discontinuities, while the ones with a small receptive field should be able to denoise the discontinuities properly.

3.3.1.2 Details of the training dataset

For training the reduced architectures, i.e. all architectures except the original one, we use a total of 10000 signals having 20000 samples each, with a sampling frequency $f_e = 100000$ Hz. The set of frequencies for training signals is $\{0.25, 0.5, 1, 2, 4, 8, 16, 24, 32\}$Hz and the possible initial SNR are $\{-10, -5, 0, 5, 10, 20\}$dB. These frequencies and SNR cover the ones seen in the available real friction signals, but also more extrem cases, which are targeted in the IMOTEP project. Validation is performed with 3000 signals with the same settings.

Due to the memory issues with the original architecture, it is not possible to use signals of 20000 samples. Therefore, we use signals of 8000 samples instead. To obtain the same total amount of samples as for other architectures, we use 25000 training signals instead of 10000. However, we do not change the amount of validation data.

3.3.1.3 Details of the training algorithm

Here, we have two different strategies for training the architectures. The original architecture is trained for 15 epochs, with an initial learning rate of 0.001, which is divided by 10 after 10 epochs. Longer trainings were tried without significant improvement.

The other architectures are trained for 30 epochs. The same initial learning rate is used, and is divided by 10 after 20 epochs.

For both training methods, we use Adam [87] optimizer. We use batches of 8 signals, and the best model is selected based on the best results on validation data.

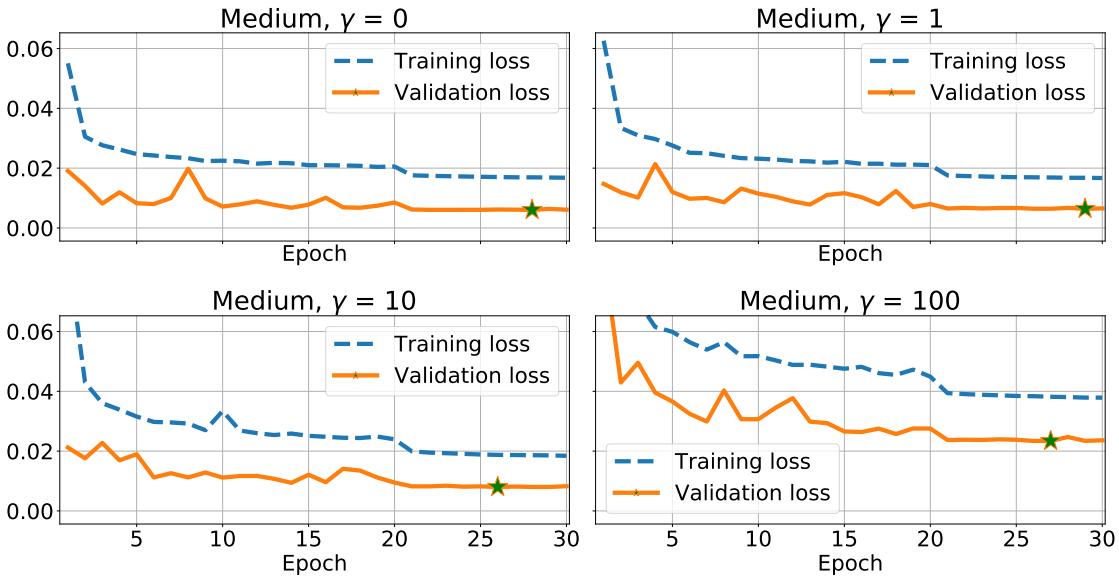


Figure 3.5: Examples of loss curves for the Medium architecture, with various values of γ. The best score for validation is shown by the green stars.

Additionally, concerning the loss term, we test four values for the parameter γ: $\gamma = 0$, $\gamma = 1$, $\gamma = 10$ and $\gamma = 100$.

Examples of training curves for the Medium architecture are given in Figure 3.5. It appears clearly that the training loss is impacted by the learning rate decay after 20 epochs. For the validation score, the decay is mostly beneficial in terms of stability. However, the best score, shown by the green star, is always obtained after this decay, showing that it is also beneficial in terms of results.

3.3.1.4 Description of the inference for evaluation

During training, the signals are a weighted sum of a square wave and a sine wave. As those two components have values in range $[-1, 1]$, their weighted sum will also have a value in this range and only the noisy signal can have values outside this range. As a consequence, the networks will learn that their prediction should always be in this range. However, for inference, it is unsure whether the ground truth is in this range or not. Therefore, using the signals in their original shape might result in a context that has not been seen during training. We center and normalize the signals to have similar magnitudes as in the training set. As the ground truth is not supposed to be known, the normalization and centering are based on the noisy signals instead of the ground truth.

Concerning the original architecture, we cut each signal in several segments to ensure not running out of GPU memory. Each segment has a length of 8000 samples at most. This might

result in a small degradation in the results as more zero-padding will be used, however the training was already done with similar data and the models should have learned how to deal with zero-padding. Therefore, the decrease should be minor. It would also be possible to use overlapping segments to avoid zero-padding, but the process would be much more time consuming.

As previously said, we will perform the evaluations for both synthetic signals following the same generation procedure as for the training data, and data generated with the Andersson model, in order to show the generalization to data closer to the target ones in terms of shapes. For the Andersson model (see Model (2.6)), the parameters are defined to have values close to the ones obtained after optimization in Chapter 2.

3.3.2 Evaluation on synthetic signals

3.3.2.1 With the training model

We test the models on signals generated the same way as in Equation (3.6). For this evaluation, the frequency of each signal is randomly selected from set $\{10, 20, 25, 35, 40\}$ Hz. White Gaussian noise is added to reach a SNR randomly selected from set $\{-15, -7, -3, 3\}$ dB, with an averaged initial MAE of 0.169 and an average initial SNR of -5.64 dB. The objective of taking random frequencies and SNR is to obtain a value that corresponds to the generalization objective. A total of 3000 signals are used with 20000 samples generated with a sampling rate of 100000 Hz. The results in terms of output SNR can be seen on Table 3.2. As it could be expected from observations made in Section 3.2.2, the TV based regularization is not useful for signals having the same properties as in training data. The only architecture that obtains an improvement with $\gamma = 1$ compared to $\gamma = 0$ is the "small" architecture, and the gain is too small to be significant. The "medium" architecture obtains the best results, confirming that the original architecture was not necessary for our application. The results also seem to indicate that too high values of the regularization parameter might be damaging. The only architecture that remains close to its maximal scores with $\gamma = 100$ is the "small" architecture, but even for this architecture, the decrease is clearly present. The poor results obtained by the Original architecture trained with $\gamma = 10$ are due to the fact that the model got stuck during training with parameters that result in a constant signal. The same problem occurred with the Original architecture trained with $\gamma = 100$, but only after a few epochs. This architecture is however clearly not suited for high values of regularization. As the best model is selected based on validation, a satisfactory model was kept. This problem however illustrates that a high regularization might be dangerous for an architecture with a too long receptive field.

Table 3.2: Results on synthetic set (output SNR in dB, average input SNR of noisy signals: -5.64 dB).
The best result for each value of γ is given in bold blue.

	Original	**Light-Full**	**Medium**	**Small**
$\gamma = 0$	17.22	15.64	**17.91**	16.17
$\gamma = 1$	16.59	15.42	**17.83**	16.26
$\gamma = 10$	-0.01	15.27	**17.19**	15.78
$\gamma = 100$	12.51	14.13	**15.78**	14.90

Table 3.3: Results on synthetic set (MAE, average input MAE of the noisy signals: 0.169)

	Original	**Light-Full**	**Medium**	**Small**
$\gamma = 0$	0.011	0.013	**0.010**	0.013
$\gamma = 1$	0.012	0.014	**0.010**	0.012
$\gamma = 10$	0.122	0.014	**0.011**	0.013
$\gamma = 100$	0.019	0.015	**0.013**	0.014

In Table 3.3, we show the results for the same experiment in terms of MAE. These results show that the MAE does not add significant information compared to the information provided by the SNR. Here, this is not really surprising, as all signals lay in the same range and are centered. For these reasons, we will focus only on the SNR for the next experiments.

In Figure 3.6, we show an example of denoising for some of the models. Here, all illustrated models are able to obtain a satisfactory denoising. It is also clear that a large regularization helps getting a smoother denoised signal, but also flattens the discontinuities.

3.3.2.2 With the Andersson model

We also perform an evaluation on signals generated with the Andersson model (see Section 2.3.1.1). Here, we first generate a velocity signal based on Equation (3.6), multiplied by 0.5 to avoid too high values where only the linear contribution would have any impact, then we apply the Andersson model with parameters randomly selected in an area close to the ones obtained during curve fitting. Let us notice that for these signals, it is then possible to have two kinds of discontinuities. The square wave will create a first kind of discontinuity by switching from a positive velocity to a negative velocity, or from a negative velocity to a positive velocity, and the smooth variations of the sine wave might sometimes result in crossing the value 0, which will then result in obtaining the discontinuity that is due to the Andersson model at null velocities. The discontinuity caused by the square waves is not really realistic as it implies discontinuities in the velocity, but it will add some complexity to the task. The frequencies and SNR used for this experiment are the same as for the training dataset. The results are shown in Table 3.4. The average initial SNR of this dataset is 2.97 dB. For this experiment, the "medium" architecture is outperformed by the original

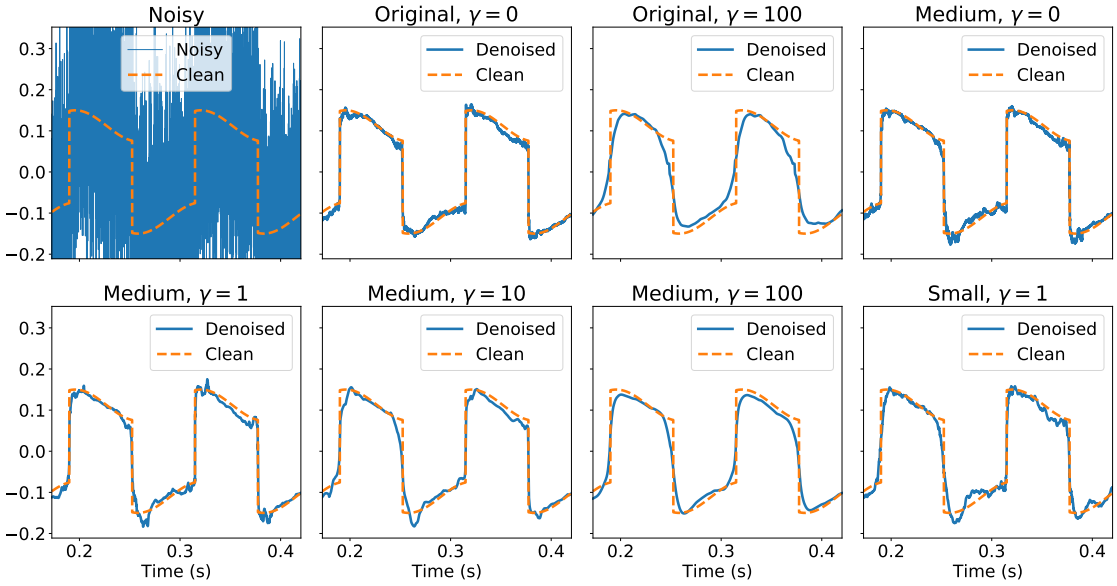


Figure 3.6: Results on a synthetic signal

Table 3.4: Results with friction model (output SNR in dB, average input SNR of noisy signals: 2.97 dB)

	Original	**Light-Full**	**Medium**	**Small**
$\gamma = 0$	**24.88**	23.56	24.22	22.26
$\gamma = 1$	**24.46**	24.26	24.43	22.36
$\gamma = 10$	5.52	23.47	**24.19**	22.00
$\gamma = 100$	20.44	22.42	**22.70**	20.88

one, but remains competitive. All reduced architectures also seem to have slightly better results with $\gamma = 1$, but the gain remains small and it is not possible to conclude definitely in terms of generalization to other shapes of the signals. The only possible conclusion is that $\gamma = 1$ does not damage the results for these two experiments with white noise. Even in Table 3.2, the results remained closed to the ones obtained without the TV penalty. From the two experiments, is is also possible to say that the "medium" architecture is the best one for this kind of signals with high sampling frequency and relatively low frequencies, as it gets results that are at least competitive with the ones obtained by the original architecture with much less parameters and memory use. For this reason, we will mostly focus on this architecture for the next steps of this thesis. The "small" architecture will also be used for some experiments as it is a good compromise between a low amount of parameters and memory use, and satisfactory results. Another interest of this architecture is its low receptive field, which might be more suitable for high frequencies or signals with lower sampling frequency. However, the "light-full" architecture is clearly not interesting. It still outperforms the "small" architecture for the experiment with the Andersson model, but never outperforms the "medium" architecture, despite using much more parameters.

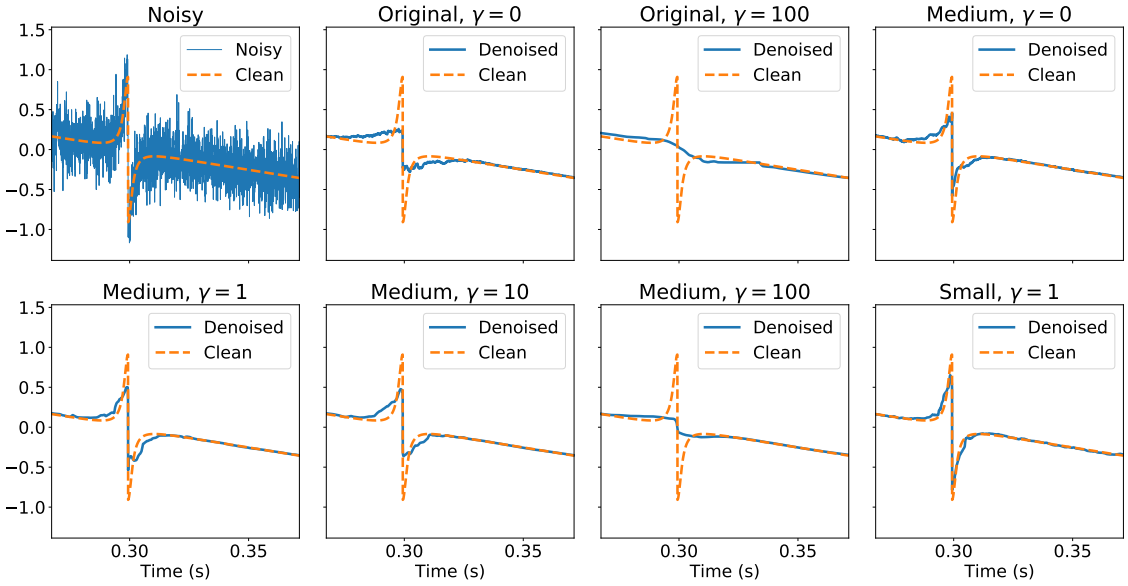


Figure 3.7: Results on a signal generated with a friction model

In Figure 3.7, we show an example of result for these signals. We focus on an area where the velocity crosses the 0 value, resulting in an exponential evolution followed by a discontinuity, then an exponential evolution again. Here, it appears that a too long receptive field is not appropriate for denoising such areas, as the Original architecture ignores them. A large regularization also damages these areas. The "small" architecture and the "medium" architecture, with no or small regularization, are able to conserve at least partially this phenomenon.

Based on this example, it might seem surprising that the Original architecture obtains the best quantitative results for these signals. However, we are using a mixture of frequencies, including low ones. In one period, the worst case would be if the square wave creates two discontinuities and the sine wave crosses creates four zero-crossings, resulting in a total of six discontinuities. As a consequence, signals at low frequencies might contain only a few discontinuities. Sometimes, there is not even a single discontinuity. The good performance of the Original architecture on other parts of the signal might explain these good results.

3.3.2.3 Data with colored noise

General behavior for other kinds of noise

Now that we have chosen one of the architectures (the "medium" one), for performing further experiments, we also want to confirm that the penalty term helps generalizing to colored noises, particularly low frequency ones. Here, we use the signals generated with

Equation (3.6), with frequencies in set $\{4, 8, 12\}$ Hz. To create the noise, we first create a white noise with a power that would result in a SNR in the set $\{-2.5, 0, 2.5, 5\}$ dB if it was added directly on the clean signal. Here, we modify the noise by multiplying its Fourier Transform by the theoretical target Fourier Transform magnitude, then we obtain the new noise by taking the real part of the inverse transform of the resulting signal. This method is not a good filtering method as it will not consider the effects of the finite time acquisition, but it is easy to implement and should not create too much damage as it is applied on a white noise and not on a meaningful signal (changing a bit more or a bit less information as expected is not a problem). We do not rescale the obtained noise afterwards, and this method might therefore also change the power of the noise and the obtained SNR will therefore not necessarily be in the set of possible SNR.

Here, we apply six kinds of filters by defining the transfer function H, with the normalized frequency ν ($0 \leq \nu \leq 1$):

Low-pass $H_{LP}(\nu) = 4(\nu - 0.5)^2$;

Amplified low-pass $H_{ALP}(\nu) = 4H_{LP}(\nu)$;

Band-pass $H_{BP}(\nu) = 16\nu^2$ for $\nu \in [0, 0.25]$. The rest of the transfer function is defined by symmetry around 0.25 and 0.5;

Amplified band-pass $H_{ABP}(\nu) = 4H_{BP}(\nu)$;

High-pass $H_{HP}(\nu) = 4\nu^2$ for $\nu \in [0, 0.5]$.

Amplified high-pass $H_{AHP}(\nu) = 4H_{HP}(\nu)$;

We represent H_{LP}, H_{BP} and H_{HP} in Figure 3.8. Here, due to the sampling frequency, all noises contain relatively high frequencies. For example, the cutoff normalized frequency of the low-pass filters is $\nu = 0.5 - \frac{1}{2\sqrt{\sqrt{2}}} \approx 0.08$ sample^{-1}, which results in a cutoff frequency of 8000 Hz. The clean signals for this experiment are generated with the same model as for training signals.

Based on the observations for the previous evaluations, the evaluations are performed only with the "medium" architecture, with no further training compared to the previous experiments. As for previous evaluations, a total of 3000 signals of 20000 samples are used for each evaluation.

The results are shown on Table 3.5. Here, it seems that increasing γ is not useful for high-frequency noises. However, concerning low frequency noises, there is a clear improvement,

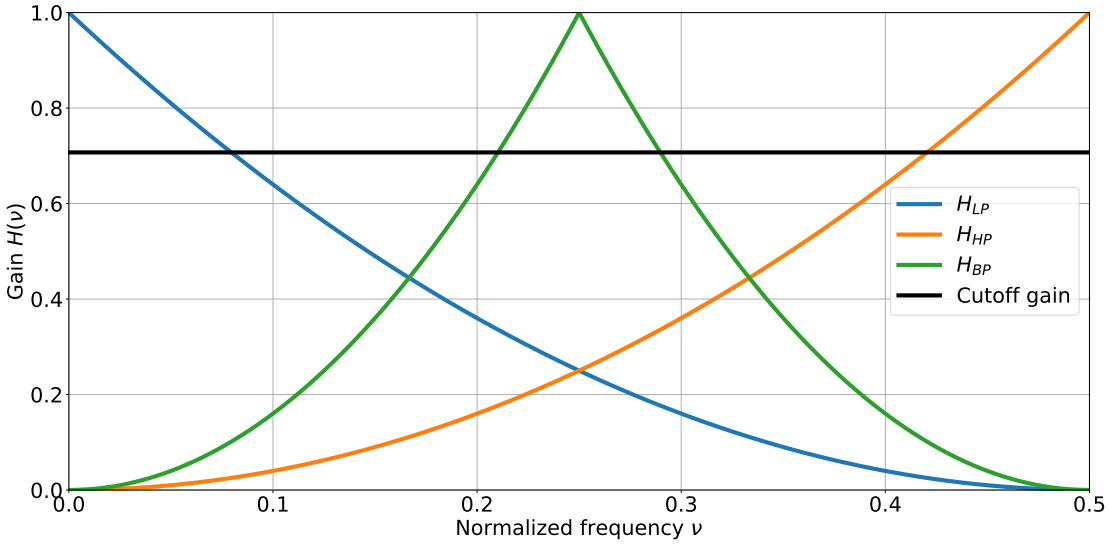


Figure 3.8: Representation of the gain for the not amplified filters.
The amplified filters are obtained by simply multiplying the corresponding not amplified filter by 4.
The values for $0.5 < \nu \leq 1$ can be obtained by exploiting the symmetry.

Table 3.5: Output SNR (dB) obtained with colored noises, using the Medium architecture.
The filters are applied to a white noise and the resulting noise is added to the clean signal

	Noisy	$\gamma = 0$	$\gamma = 1$	$\gamma = 10$	$\gamma = 100$
High-pass	8.16	**36.15**	35.22	33.73	28.21
Amplified high-pass	−3.80	31.41	29.83	**32.22**	26.98
Low-pass	8.18	27.19	**27.58**	26.95	25.28
Amplified low-pass	−3.80	13.81	13.59	14.41	**16.40**
Band-pass	8.20	**36.33**	35.49	33.92	28.36
Amplified band-pass	−3.90	30.90	30.90	**31.02**	26.74

especially when the amplified filter is used. The results for $\gamma = 1$ might indicate that this value is not performing well with too low initial SNR. However, the only case with a significant drop compared to no regularization is for the amplified high-pass, where good results are obtained by all models. It is therefore hard to ensure that the model trained with $\gamma = 1$ is not performing well based on these results. In general, it seems that using no or little regularization is better, but for contexts with a noise that has a high power and contains mostly low frequencies, using $\gamma = 10$ or even $\gamma = 100$ might be useful. For the case of the friction signal, the SNR seems to be positive in most cases. It might be locally negative, but it remains rare. The value $\gamma = 100$ is therefore probably too high to obtain satisfactory results.

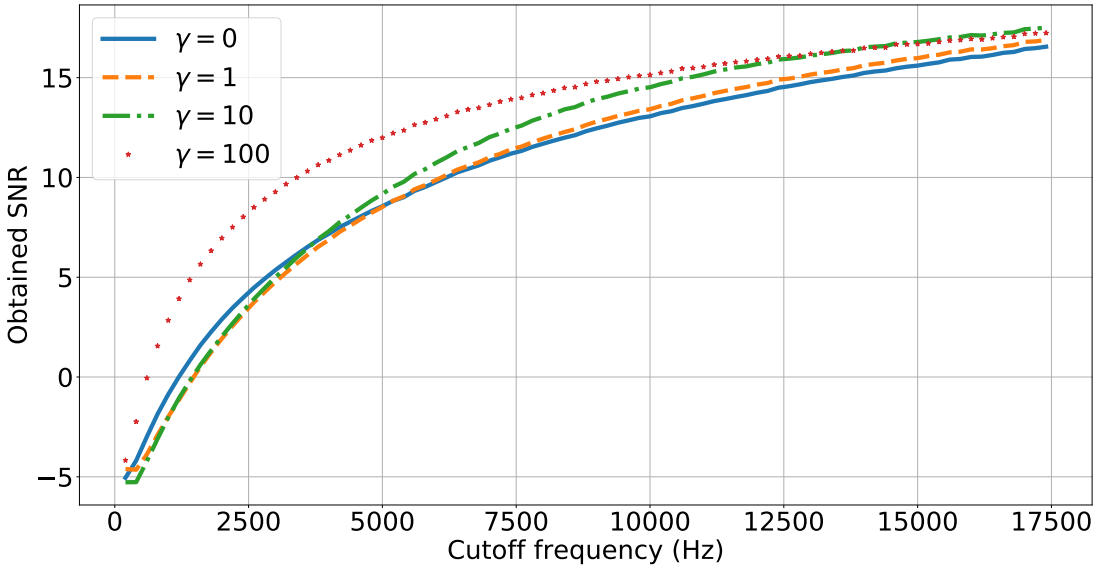


Figure 3.9: Evolution of the obtained SNR for low frequency noises, with an initial SNR of -5dB. The four curves are obtained by training the Medium architecture with different values of γ.

Focus on low-frequency noises

To complete the previous experiment, we perform new evaluations for noise with low frequencies. As for the previous experiment, we build masks that are applied to the Fourier transform. This time, for a specific cutoff frequency, the mask is defined through its ideal definition, i.e. that frequencies lower than the cutoff frequency are not affected, frequencies higher than the cutoff frequency are removed. Again, this filtering technique is not good in practice, but the consequences should be limited as we apply it only on the noise.

For this experiment, we select all frequencies from 200 to 17400 Hz, with a step of 200 Hz, as cutoff frequency. For each cutoff frequency, an evaluation is performed with 3000 signals of 20000 samples, with a fundamental frequency in set $\{4, 8, 12\}$ Hz. To avoid the evolution from one frequency to another to be due to the larger initial SNR when taking a lower cutoff frequencies, we rescale the noise so that each signal has an initial SNR of exactly -5dB.

As for the previous experiment, we use only the Medium architecture and we compare the results with the different values of γ.

The evolution of the obtained SNR for the different values of the cutoff frequency are shown in Figure 3.9. The evolution clearly shows that all models fail denoising for extremely low frequencies, and obtain better results when reaching higher cutoff frequencies. Concerning the impact of the γ value, it is clear that the largest regularization helps dealing with the low frequency noises.

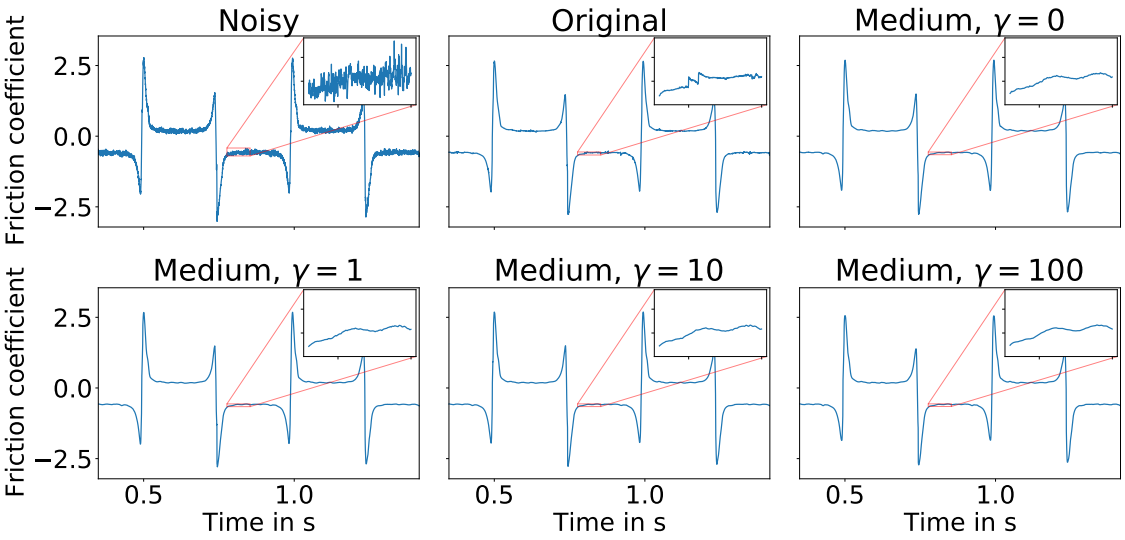


Figure 3.10: Results of denoising on a real signal at 120 rpm (using motor oil)

Here, the model trained with $\gamma = 1$ seems to be disadvantageous compared to the models trained with $\gamma = 10$ and $\gamma = 100$, even though the difference with the model trained with $\gamma = 10$ remains small. However, this does not necessarily imply that this value should not be selected. The evaluations detailed in other sections show that with this value, it is always possible to obtain satisfactory results. Here, we focused on low frequency noises and used a specific initial SNR, but in real cases, the models have to deal with various frequencies, including high ones. The results for these low frequency noises prove the impact of the regularization and its benefits in such scenarios, but the selection of the regularization would have to be done considering not only the improvement on the low frequency noises, but also the impact on high frequency. This observation about high frequencies is true for the noise, but also for the fundamental frequency of the clean signal, which remained low in this case. It is expected that with very high frequencies, too much regularization would be damaging.

3.3.2.4 Results on real friction signals

In this section, we check the generalization of the models on the real friction signals acquired by Ireis with a linear reciprocating tribometer. Examples of these signals are provided on Figures 0.5 and 0.6. As for the evaluation signals, each friction signal is centered and normalized before being denoised, then it is brought back to its original space. Here, we do not have any ground truth and it is therefore not possible to perform any quantitative analysis. Comparing the results of the different methods can only be based on visual observations. Here, we will compare the original architecture without regularization and the "medium" architectures with all four values of regularization.

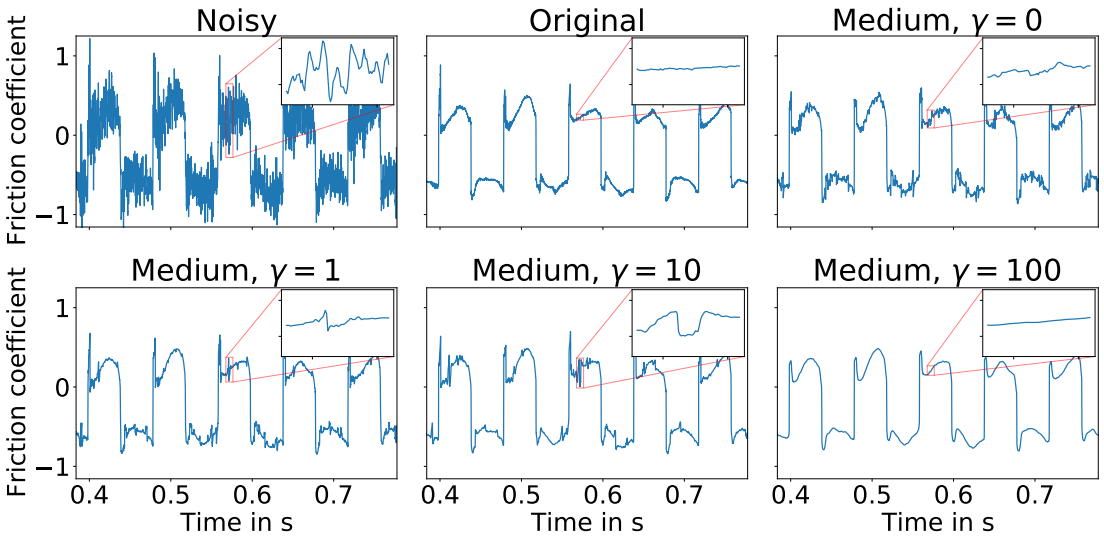


Figure 3.11: Results of denoising on a real signal at 750 rpm (using motor oil)

In Figure 3.10, we show a signal acquired at 120 rpm where the lubricant is motor oil. Here, the original architecture creates some artifacts by adding some discontinuities at places where none is expected. This might be partially due to the way this architecture is used, by cutting each signal in small sub-signals, but it seems to happen too often to be the only explanation, and it might result from some local phenomena. Concerning "medium" architectures, it seems that the regularization has no significant impact for such a low level of noise (the initial SNR is estimated to 25.20 dB based on the denoising with $\gamma = 10$), even though the peaks are slightly reduced with $\gamma = 100$.

In Figure 3.11, the lubricant remains the same, but the rotation speed is now of 750 rpm. Using the "medium" architecture with no regularization seems to be ineffective for removing all artifacts. With $\gamma = 100$, the signal is extremely smooth, but the peaks are much smaller than in the noisy signal. With $\gamma = 1$ or $\gamma = 10$, some artifacts remain but their number is already largely reduced compared to the architecture without regularization. It also seems that some of these artifacts appear periodically, and they might therefore contain some useful information, depending on the analysis performed with the denoised signals. It is hard to be affirmative on whether these artifacts are part of a periodical pattern or are due to a colored noise. The original architecture removes many artifacts but is not as smooth as the model with $\gamma = 100$. It is however better in preserving the peaks. For this signal, the initial SNR was estimated to 9.41 dB.

Lastly, in Figure 3.12, another signal acquired with a rotation speed of 750 rpm is shown. This time, a mix of glycerol and water is used instead of motor oil, which results in different behaviors. Here, it seems that the regularization is essential to obtain satisfactory results with the "medium" architecture. As for the previous signal, the model trained with $\gamma = 100$ gets the smoother results, but it is possible that it also removes some relevant information,

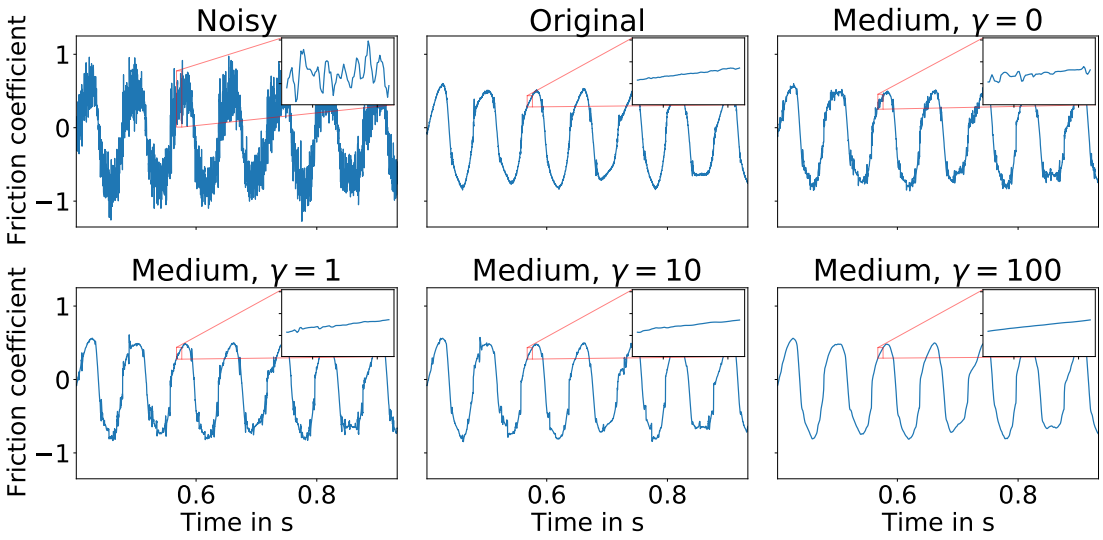


Figure 3.12: Results of denoising on a real signal at 750 rpm (using a mix of 95% glycerol and 5% water)

especially concerning some small peaks appearing around discontinuities. The original architecture now seems to be the best compromise between getting a smooth signal and preserving the peaks. With $\gamma = 1$, the "medium" architecture also preserves the peaks (at least partially), but a large amount of artifacts still remains. With $\gamma = 10$, the amount of artifacts is quite low, but the peaks are almost always removed. The estimated SNR is close to the one of the previous signal, estimated to 9.83 dB.

From all these observations, it appears that deciding which architecture is the best one is not easy and might largely depend on the objective of further analysis. By considering its amount of parameters and memory use, the original architecture is probably not the best choice for our applications. Using $\gamma = 100$ might results in removing too much information, while using no regularization fails to remove most artifacts. The values $\gamma = 1$ and $\gamma = 10$ seem to be the best performing ones in terms of compromise, but choosing between both is not easy. Based on the quantitative results, we will mostly focus on using the "medium" architecture with $\gamma = 1$ in the following parts of this thesis, when performing denoising with methods coming from this chapter.

3.4 Comparison with a method using wavelets and total variation

In [10], a method was proposed for denoising discontinuous signals with wavelets combined with a TV penalty. Here, we compare our deep learning based method with their method.

For implementing their method, we use the "WATV software" that they provide[1], with the same parameters as the ones they propose in their paper. We apply this method on five of the signals they propose, which consist of 1024 samples. The chosen signals are *Piece-Regular* (**PR**), *Piece-Polynomial* (**PP**), *Ramp* (**R**), *HeaviSine* (**HS**) and Blocks (**B**). These signals are not periodic, but the main aspect of our work in this chapter is about the presence of discontinuities.

Considering the amount of samples and the possibility to have a large amount of discontinuities in a short time range, the only architecture that remains possible from the ones defined previously is the "small" architecture. Even with this architecture, it is likely that for some signals, the receptive field would already be too long. Here, we retrained the architecture with signals of 20000 samples and sampling rate of 1024 Hz and we remove frequencies above 4 Hz from the training set. The architecture trained with $\gamma = 1$ generally gave the best results, and we will therefore focus on this architecture. As for the previous methods, the evaluations are performed by centering and normalizing the signals so that they fit in range $[-1, 1]$, in accordance with training signals.

We also trained a new architecture with two stacks of three residual layers. Other parameters and training algorithm are unchanged compared to the "small" architecture. We denote this architecture as "XS".

Here, we obtain the results of each signal by performing 100 realizations of the experiments, i.e. that the same ground truth is generated 100 times with different noises each time, which lets us obtain a more robust value. Four different values of initial SNR are used. Notice that the method of [10] requires knowing the standard deviation of the noise for performing the denoising, whereas our method can be used without any prior knowledge as long as the SNR values remain close enough from the ones in training set.

The results of the method of [10] are shown in Table 3.6, and the results of our method are shown in Table 3.7 for the "small" architecture and in Table 3.8 for the "XS" architecture. With very low initial SNR, our method always outperforms the one of [10]. In less noisy contexts, signals with a lot of discontinuities are better denoised by the method of [10], even though our method remains satisfactory at least up to 5dB. For the ramp and HeaviSine signals which contain only one or no discontinuity, our method gets better or at least similar results. The "XS" architecture is never able to outperform both the WATV software and the "small" architecture, but keeps improving the signals with a lot of discontinuities even with large initial SNR.

Our method also has a large advantage over the one of [10] in terms of time complexity. The Wavenet for speech denoising has a linear complexity, while the WATV method has

[1] `http://eeweb.poly.edu/iselesni/software/index.html`

Table 3.6: Results for the WATV software (obtained SNR)

Original SNR	PR	PP	R	HS	B
-15dB	1.10	0.74	1.32	1.55	0.70
-5dB	**8.89**	**8.88**	10.61	11.30	**8.49**
5dB	**15.94**	**16.41**	20.29	**20.06**	**15.78**
15dB	**25.11**	**26.46**	30.33	**28.50**	**25.87**

Table 3.7: Results for the "small" architecture, trained with $\gamma = 1$

Original SNR	PR	PP	R	HS	B
-15dB	**2.43**	**2.55**	**5.12**	**6.28**	**3.00**
-5dB	7.00	8.46	**14.09**	12.21	7.55
5dB	13.85	15.92	**24.60**	19.83	14.60
15dB	16.27	21.81	**32.92**	28.10	16.48

Table 3.8: Results for the "XS" architecture, trained with $\gamma = 1$

Original SNR	PR	PP	R	HS	B
-15dB	-0.91	-1.21	-0.98	-0.74	-1.59
-5dB	8.67	8.48	9.66	10.38	8.04
5dB	13.62	15.47	20.38	19.67	15.17
15dB	19.04	25.54	30.28	26.63	23.07

a quadratic complexity. For signal of 1024 samples, the consequences remain restricted. Here, denoising with WATV software was about 50 times longer than with our method on an Intel©Core^{TM} i7-7700 CPU@3.60GHz x 8. A test with a signal of 16384 samples resulted in a 2000 times longer process, showing that this would become a major issue for applying the WATV software to signals as long as the friction signals. The training of the "small" architecture for this experiment lasted around two hours on a GTX Titan X. The training of the "XS" architecture lasted around an hour on a GTX 1080 Ti.

As an illustration for the results obtained, we show an example of the denoising of the ramp signal in Figure 3.13, and an illustration of the denoising of the Piece-Regular signal in Figure 3.14. These two examples show that the "small" architecture is able to efficiently denoise smooth parts of the signals even with an initially low SNR, but struggles denoising areas with a lot of variations.

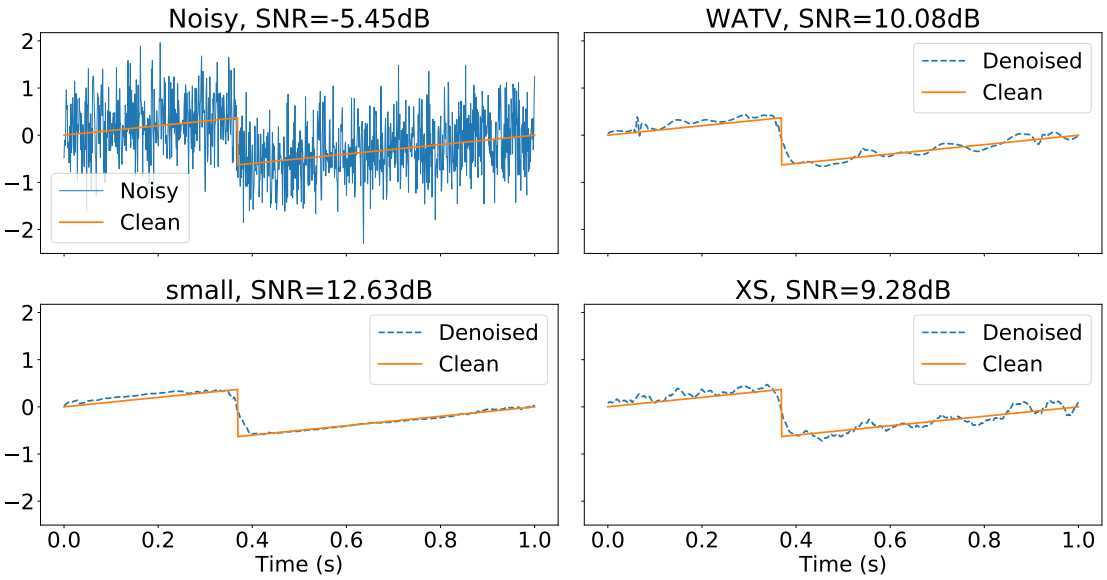


Figure 3.13: Example of denoising the ramp signal.
For our methods, the parameter γ for regularization was set to 1.

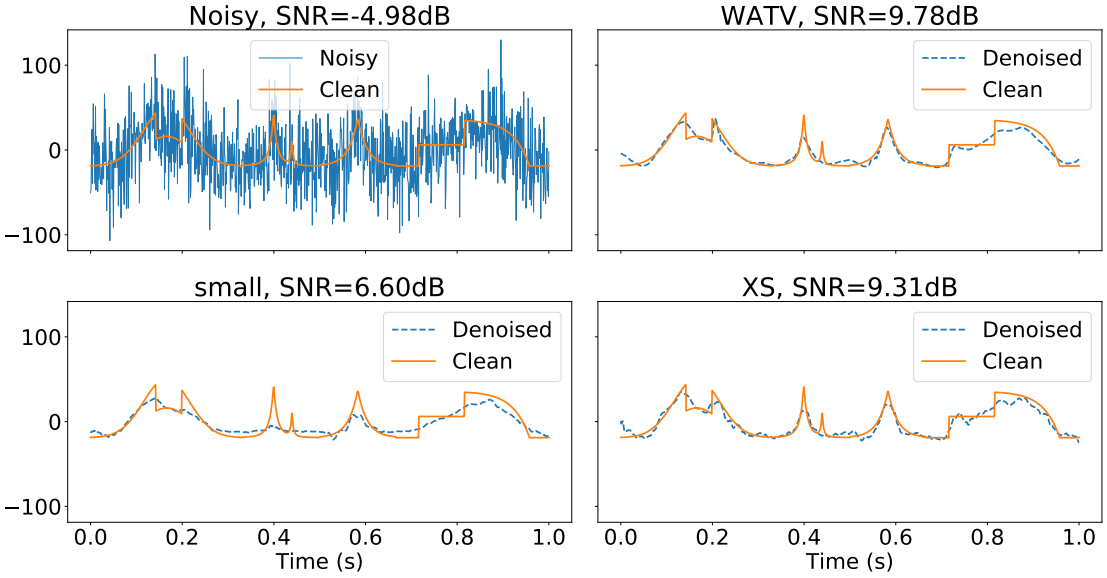


Figure 3.14: Example of denoising the Piece-Regular signal.
For our methods, the parameter γ for regularization was set to 1.

3.5 Experiments for FIB image sequences

3.5.1 Introduction and methodology

To observe the behavior of our method on other kinds of real data, we also denoise FIB image sequences. Several studies have already focused on denoising FIB or SEM images. In [147], the noise is assumed to have a Gaussian distribution. Some other methods have already focused on using Deep Learning methodologies for this task. Among recent methods, [148] trains a CNN with synthetic SEM images, sharing similar properties with FIB images, containing white Gaussian or Poisson noise. For their application, they denoise only one image and use a 2D convolutional denoiser. Here, we have image sequences, implying three dimensions, two for each frame and the temporal dimension. To apply our method directly without defining a new method that considers all three dimensions, such as by using 3D convolutions or combining 2D convolutions with recurrent layers for the temporal dimension, we focus on denoising the sequences by applying the networks along one dimension only, resulting in three approaches:

1. **Line-wise** : denoising the signals defined by each line of each frame of the original sequence;

2. **Column-wise** : denoising the signals defined by each column of each frame of the original sequence;

3. **Pixel-wise** : denoising the temporal evolution of each pixel. For this method, we also perform a second denoising to limit the border effects at the beginning and the end of the signal.

Here, the amount of frames is quite limited and the size of the area of interest (the area being processed) in each frame contains only a few hundreds of pixels. As for the comparisons with the WATV software, we have to focus on methods having a small receptive field. Therefore, we pick the "small" architecture. We also define a new architecture, which contains two stacks of three layers, with 16 channels for each residual layer and 32 and 8 channels for the two post-processing layers. The resulting architecture contains only 14000 parameters and has a receptive field of length 65. We denote this architecture as "XS_2". The data for training this model are similar to the ones used for the architecture in the comparison with the WATV software, but signals of 2000 samples with a sampling rate of 100Hz are used instead of signals of 20000 samples. Only 25 epochs are performed, with a decay after 20 epochs, and each batch now contains 32 signals.

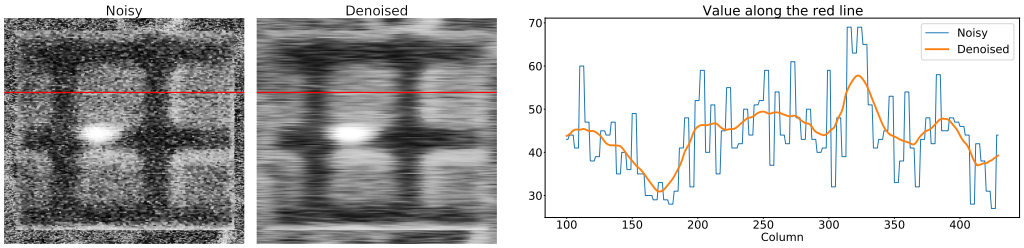


Figure 3.15: Result with line-wise denoising, using the "XS_2" architecture with $\gamma = 10$

3.5.2 Results

For all frames displayed in this section, the contrast has been amplified. We will focus mostly on the pixel-wise denoising, which seems to give the best results.

3.5.2.1 Line-wise denoising

For this method, the "small" architecture was unable to get any satisfactory result. Therefore, we focus on the "XS_2" architectures.

In Figure 3.15, we show an example of result with this method for one frame. Here, the result shows that the method manages to obtain a more regular value in the middle of the squares appearing in the frame, but the borders are extremely blurry. However, the value along the red line seems quite consistent with the noisy values.

Additionally, if we observe how the value of a pixel evolves from one frame to another after denoising (Figure 3.16), it is clear that this method is not consistent in terms of temporal continuity.

3.5.2.2 Column-wise denoising

In Figure 3.17, we show the result of column-wise denoising with the "XS_2" architecture trained with $\gamma = 1$. The result seems a bit better than with line-wise denoising. This might be due to the lower regularization value. However, the videos seem to contain some streak noise that applies horizontally, and it is possible that the line-wise denoising amplifies this streak noise, while column-wise denoising reduces it. The value along the red line seems less smooth than with line-wise denoising, which is not surprising. With line-wise denoising, the red line corresponded to the direction on which denoising implied spatial continuity, which is not the case with column-wise denoising.

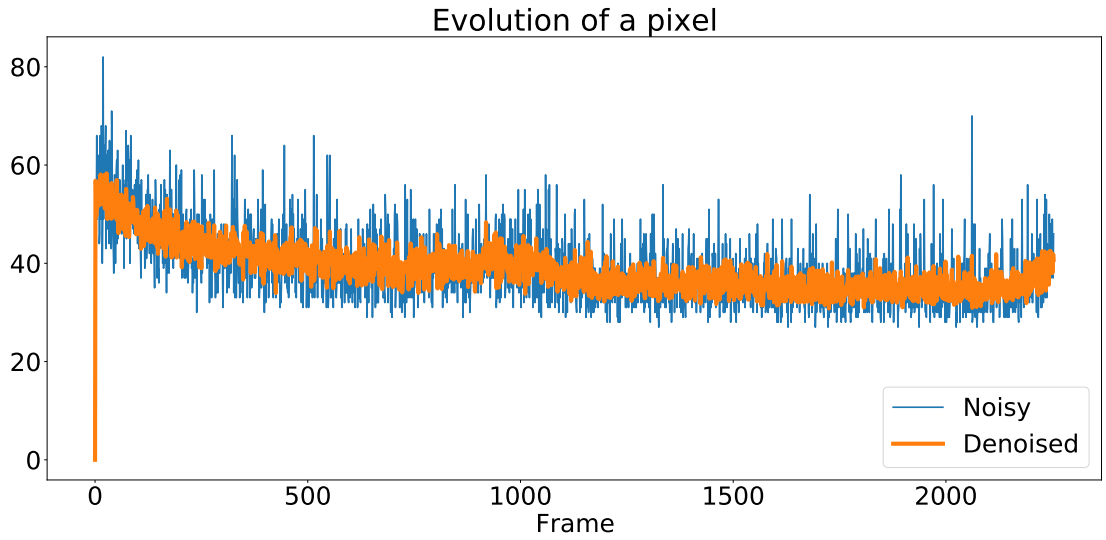


Figure 3.16: Evolution of a pixel with line-wise denoising, using the "XS_2" architecture trained with $\gamma = 10$.

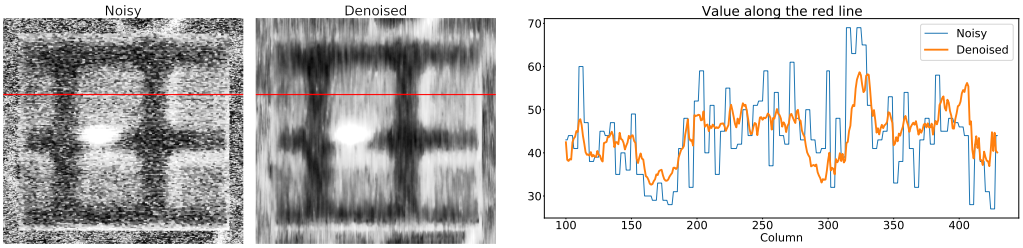
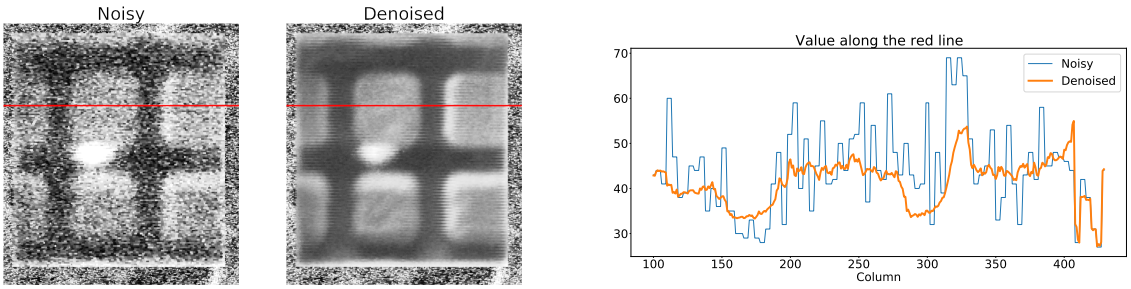
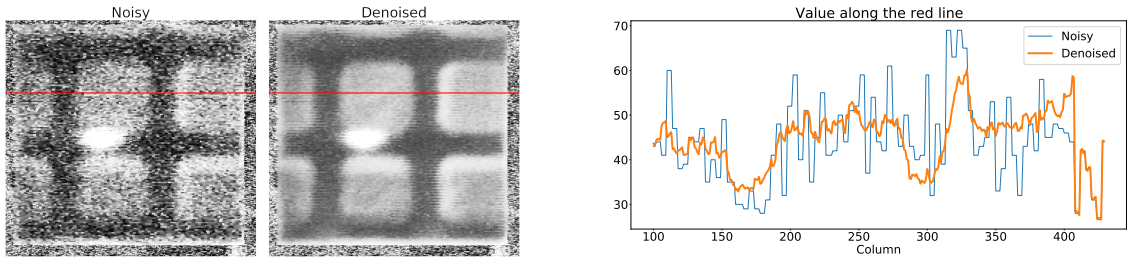


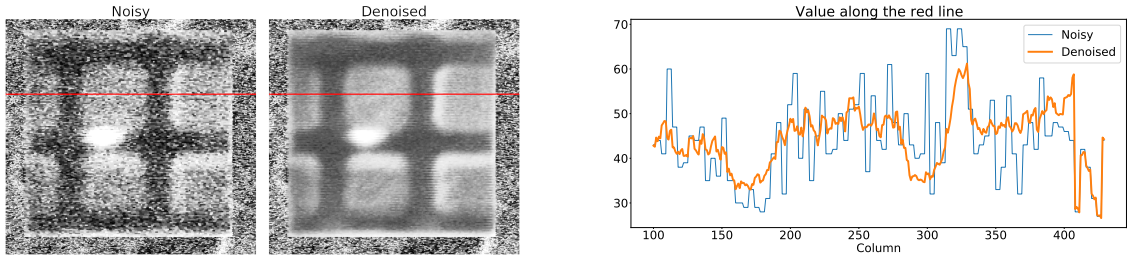
Figure 3.17: Result with column-wise denoising, using the "XS_2" architecture with $\gamma = 1$



(a) using the "small" architecture with $\gamma = 1$



(b) Using the "XS_2" architecture with $\gamma = 0$



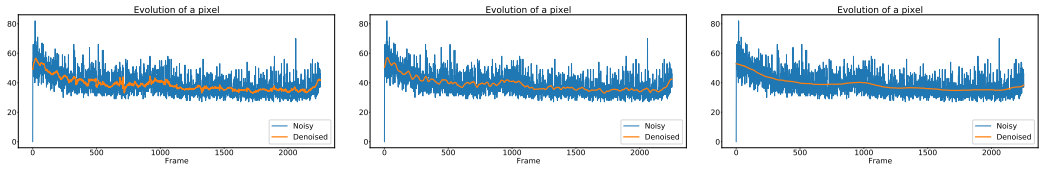
(c) Using the "XS_2" architecture with $\gamma = 10$

Figure 3.18: Results with pixel-wise denoising

Here, we do not show the evolution of a pixel, which is similar to the one observed with line-wise denoising.

3.5.2.3 Pixel-wise denoising

We show the results of method 3) with the "small" architecture trained with $\gamma = 1$ in Figure 3.18 (a), with the "XS_2" architecture trained with $\gamma = 0$ in Figure 3.18 (b) and with the "XS_2" architecture trained with $\gamma = 10$ in Figure 3.18 (c). Here, all three figures seem correctly denoised. The result obtained by the "small" architecture seems a bit smoother than the ones obtained with the "XS_2" architecture, but the streak noise is clearly visible on all three results. The value of γ does not seem to make a large difference when focusing on a frame.



(a) "XS_2" architecture, $\gamma = 0$ **(b)** "XS_2" architecture, $\gamma = 10$ **(c)** "small" architecture, $\gamma = 1$

Figure 3.19: Temporal evolution of a pixel with pixel-wise denoising

In Figure 3.19, we show the temporal evolution of a pixel for each architecture. It is now possible to observe the impact of the parameter γ, which allows obtaining more temporal smoothness. However, here using the "small" architecture, which has a longer receptive field, seems to be a better way for obtaining temporal continuity when denoising along the temporal dimension. However, considering the amount of parameters, the "XS_2" architectures are already satisfactory, especially when considering that they also obtain exploitable results with line-wise and column-wise denoisings, which is not the case for the "small" architecture, and it is likely that it would be possible to build an intermediate architecture, with only a few more parameters, that would be able to get similar results as the "small" architecture. Additionally, this shows that denoising the sequences, including removal of the streak noise, might be done with a single 3D architecture with a larger receptive field along the temporal dimension than spatial dimensions, or by first performing the pixel-wise denoising then applying another architecture to postprocess the images with a 2D architecture.

3.6 Conclusion

In this chapter, we have proposed a first method for exploiting deep learning with our signals. Despite the lack of ground truth, we were able to obtain satisfactory results by generating synthetic data. We have shown that these data can be used for training neural networks in various applications, despite the simplicity of the model used to generate these data. We have also proposed a new loss term for enforcing smoothness outside of discontinuities. This loss term clearly had a large impact on real friction signals, and allowed quantitative improvements on synthetic colored noises, especially low frequency noises, but it is however possible that some of the removed phenomena are part of the periodic component, and not only a colored noise. This lack of exploitation of the periodicity is the main limitation of the work presented in this chapter. Even though the generated data are periodic, the sampling frequency implies that the networks never see several periods inside their receptive fields and are therefore unable to exploit the periodic information. The networks are only able to exploit the presence of discontinuities in the training data.

Concerning the variations in the results compared to the work published in [24], the ideal approach for knowing how much each model is affected would be to perform an estimation of the variance for each model, by performing multiple trainings, such as in cross-validation or bootstrapping. However, obtaining a robust estimate would require tens of trainings, which is not reasonable for so many networks and configurations. For small values of γ, especially without regularization, new experiments will be performed in the next chapters with similar trainings, confirming the good results obtained in this chapter. Therefore, we assume that the risks of obtaining much worse results are limited to the cases with large receptive fields and high values of γ.

The next chapters will focus on proposing methods that are able to access the periodic information, which might actually also result in a better regularization than the one proposed in this chapter, especially when trying to distinguish colored noises from periodic phenomena.

4

P1D architectures : Extending the Deep Learning methods to exploit periodicity

In the previous chapter, we proposed a first implementation of a deep learning methodology for denoising. This method was shown more efficient than curve fitting for adaptation to the different parts of the signals, and was much easier to adapt to other applications. The proposed regularization was also shown successful for obtaining smoother results. However, despite the training signals being periodic, periodicity was not exploited by the models. It is therefore not possible to say whether the variations removed with larger regularization were part of a periodic component or of a colored noise.

In this chapter, we propose two methods for exploiting periodicity. One method is based on a preprocessing of the signal while the other is based on an adaptation of the neural networks to exploit periodicity.

Communications linked to this chapter

International conferences papers

[25] J. Rio et al. “Leveraging end-to-end denoisers for denoising periodic signals”. In: *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE. 2021

Journal papers (to be submitted)

[26] J. Rio et al. “Towards a unified framework for denoising periodic signals”. In: *To be defined* (To be submitted in 2022)

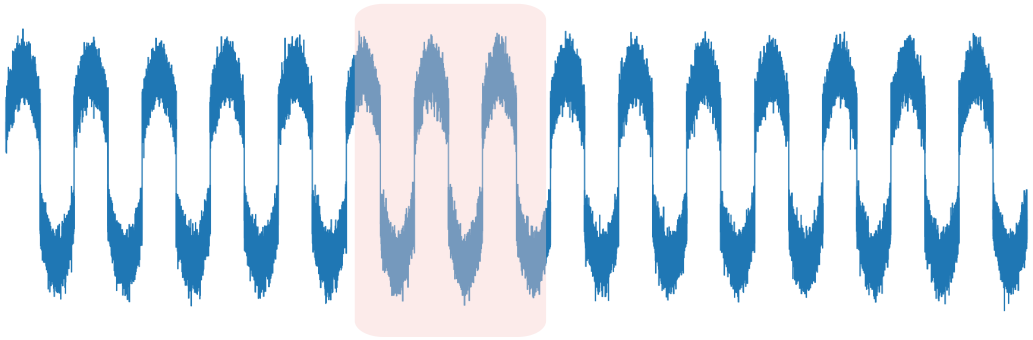


Figure 4.1: Illustration of a case where the receptive field is sufficiently long to observe several periods

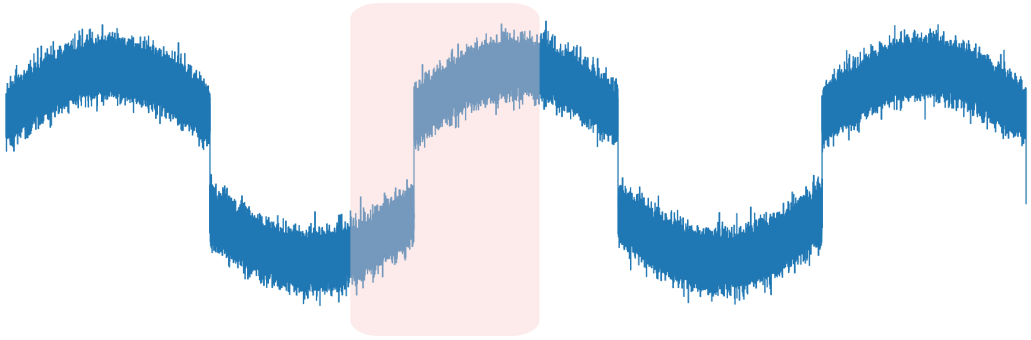


Figure 4.2: Illustration of a case where the receptive field is too short to observe several periods

4.1 Problem with the receptive field

As explained in Section 1.5, CNN have a fixed receptive field. We already presented some of the advantages of this property, such as translation invariance, but this fixed receptive field also has some drawbacks, especially when dealing with periodic signals.

To exploit periodicity, the receptive field should be long enough to analyze several periods of the signal, i.e. the receptive field should have a length L that is at least $2T$ with T the amount of samples in one period. This happens when the normalized frequency of the signal is quite high, as shown in Figure 4.1, where the receptive field, shown in red, can observe a bit more than two periods (which already gives only a limited access to the periodic information).

However, with longer periods (low normalized frequencies), the receptive field might become too small to observe several periods. This is the case in Figure 4.2.

Generally speaking, considering that the receptive field is centered on the current sample, the receptive field length L should be higher than two periods, i.e., $L \geq \frac{f_e}{f_0}$, with f_0 the fundamental frequency of the signal.

In the case of the friction signals, the sampling frequency is extremely high (100000 Hz) and the frequencies are relatively low. Even for the maximal frequency of 100 Hz (no signal is available at such frequency for now), one period has 1000 samples. For the current signals, the frequencies are under 20 Hz and each period contains at least 5000 samples. Additionally, even if some of the signals at 100 Hz were accessible, we are aiming to build a model that is able to generalize to various frequencies, including smaller ones (for example at 2 Hz, where the receptive field should observe at least 100000 samples to observe several periods). Of course, it is possible to subsample the signals, but reducing the resolution also implies losing some information. As we aim to retrieve all interpretable phenomena, including short-time phenomena, this solution is not ideal.

One possibility would be to build a network with an extremely long receptive field. For example, we could use a version of the WaveNet for speech denoising [9] with a single stack containing a large amount of residual layers. However, the network would use a huge amount of parameters for observing some samples in-between the samples separated by a period, and it is possible that these areas do not contain any useful information for the task.

Another method for avoiding observing the entire signal would be to use dilated convolutions to obtain a receptive field that does not observe all samples. For example, a post-processing could be applied with convolution layers with a dilation rate of T, which would result in observing a receptive field around the sample n and a receptive field of the same length around samples $n + T$ and $n - T$. However, this method would work only for one specific frequency.

Another problem with the two proposed solutions is the length of the signals needed for training the network. If several hundreds of thousands of samples are needed for each signal, then both the batch size and the network have to be reduced.

4.2 Proposed solutions

Based on the preceding information, it is clear that using 1D convolutional denoisers for performing the entire process of denoising is not the more adequate solution in the case of periodic signals. Here, we propose two methods for leveraging 1D convolutional denoisers in order to obtain a denoiser which is able to exploit the periodicity of the signals. The proposed methods can generalize to a large set of frequencies, including unseen ones.

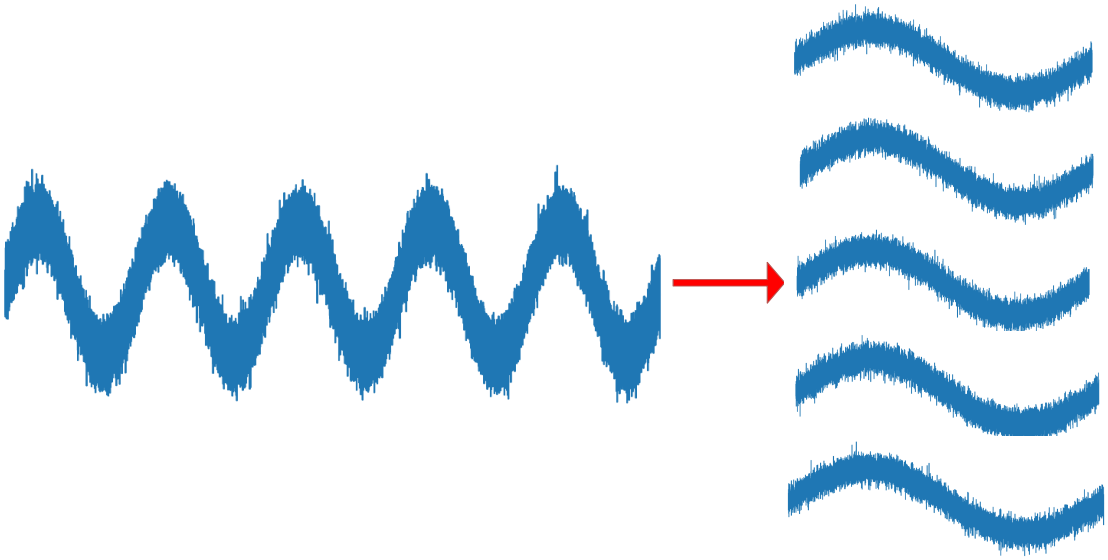


Figure 4.3: Illustration of the transformation of the signal into a matrix

4.2.1 Reshaping the data

For both proposed methodologies, the first step is the same and consists in reshaping the signal into a matrix.

For a noisy signal $\mathbf{x}$, the first step is to detect the fundamental frequency. For that, it is possible to use the Algorithm 1, a prior knowledge or a curve fitting on a signal related to $\mathbf{x}$. As an example for this third possibility, in the case of friction signals, there is a signal of position for each noisy friction signal, it is then possible to fit a sine wave to obtain the fundamental frequency. For our experiments, we focus on the method for exploiting the periodicity rather than the method for detecting the periodicity. Therefore, as it is previously known in the case of the friction signals, we simply use the setpoint frequency as the fundamental frequency of the signal.

Once the frequency has been found, it is possible to compute the amount of samples T in one period. The signal $\mathbf{x}$, of length N is cut into several sub-signals of length T (here, we assume $\frac{N}{T} = P \in \mathbb{N}$, if it is not the case, it is possible to complete the signal, e.g., with zero-padding), resulting in a matrix $\mathbf{X}$ of size $P \times T$ such that:

$$\forall 0 \leq i \leq P-1, \forall 0 \leq j \leq T-1, \mathbf{X}_{i,j} = \mathbf{x}_{iT+j}. \tag{4.1}$$

An illustration of this process is shown in Figure 4.3.

This new shape allows an easier observation of the data. The first (temporal, along each row) dimension allows observation of the local evolution, while the second (periodic, along each column) dimension results in moving from one period to another.

4.2.2 Method based on a preprocessing

The first method for exploiting the periodicity with the new shape of the data is to preprocess the data. Here, the idea is to perform an average of the periods along the second dimension. This method can be seen as a naive solution where the moving average is a preprocessing not integrated in the denoiser.

It is of course possible to average all periods and assume that the signal is exactly a repetition of the same pattern. However, it is likely that realistic signals have actually little evolutions over time. Therefore, we perform a moving average rather than a global average, i.e., we transform the matrix $\mathbf{X}$ into a new matrix $\mathbf{X}^{(m)}$ such that:

$$\forall i, \forall j, \mathbf{X}^{(m)}_{i,j} = \frac{1}{K_i} \sum_{k=-K}^{K} \mathbf{X}_{i+k,j} \tag{4.2}$$

with a moving average of length $2K+1$ and a matrix completed with zero-padding. Here, we use a constant weight moving average, but weights could be used to increase focus on the current period. For the value K_i, the easiest method would be to use $2K+1$, but it would results in a too high division for the K first and K last periods, which would be affected by zero-padding. Therefore, if the matrix $\mathbf{X}$ contains P rows, then we use

$$K_i = \begin{cases} K+1+i & \text{if } i \leq K \\ K+P-i & \text{if } P-1-i \leq K \\ 2K+1 & \text{otherwise.} \end{cases} \tag{4.3}$$

Here, as they are distant from one period, the samples used in each average should be independent (or almost independent), or at least uncorrelated as the correlations are generally decreasing over time. As a consequence, averaging K_i values should result in dividing the variance of the noise by K_i. Therefore, recomposing the signal $\mathbf{x}^{(m)} = [\mathbf{X}^{(m)}_{0,.} \dots \mathbf{X}^{(m)}_{P-1,.}]^T$ will result in a signal having the same ground truth as the original signal $\mathbf{x}$ but a larger SNR, and will therefore be easier to denoise. It is then possible to end the denoising with a 1D denoiser (which does not need any specific training compared to chapter 3).

The global framework of this method can be seen in Figure 4.4.

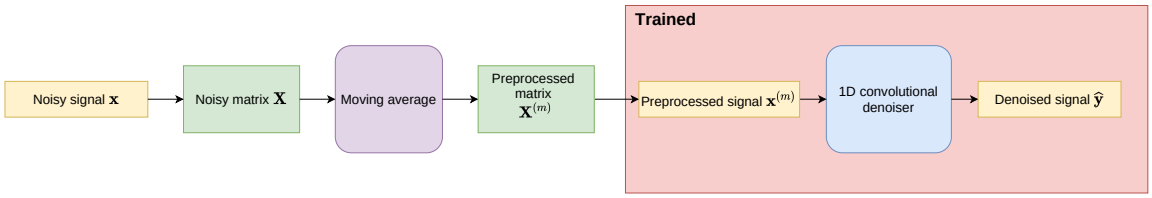


Figure 4.4: Framework for the method with preprocessing

In the following parts of this thesis, methods using this process will be denoted by using "MA" as a prefix to their name.

A similar method could be proposed by applying a postprocessing. However, the advantage of averaging the periods is lost when the hypothesis of independence of averaged noise samples is not possible. Before denoising with neural networks, this hypothesis is possible, even though it is not strictly correct for real signals. After denoising, the remaining noise is clearly impacted by the nature of the signal in each area (for example a peak). Therefore, it is likely that a large part of the remaining noise is actually due to a bias that will remain the same for all periods, and doing an average will not allow removing this bias.

Let us notice that it might be more relevant for real cases to use a weighted average to optimally combine the different periods. However, we stick to an unweighted average in our case both for ease of implementation and because the weights that would be ideal in the case of synthetic signals are not necessarily the same as the ones that would be obtained with real signals. This second remark is amplified in our case by the fact that the synthetic signals are exactly periodic, implying that the optimal case would probably be to perform an unweighted average of all periods instead of a moving average. The best case would however be to propose a method that defines the ideal weights based on the signal itself, and not the same set of weights for all signals.

Here, we are using this method as a preprocessing to the neural network, but it could easily be extended as a preprocessing to any kind of denoiser.

4.2.3 Method based on leveraging the neural networks

4.2.3.1 Description of the method

Another solution is to use the matrix as an input to the neural network. In this case, the network has to be able to exploit both dimensions. There are several possibilities for doing this. The first possibility is to use a network designed for image denoising. This method gives access to a multitude of solutions, but image denoisers generally do not differentiate

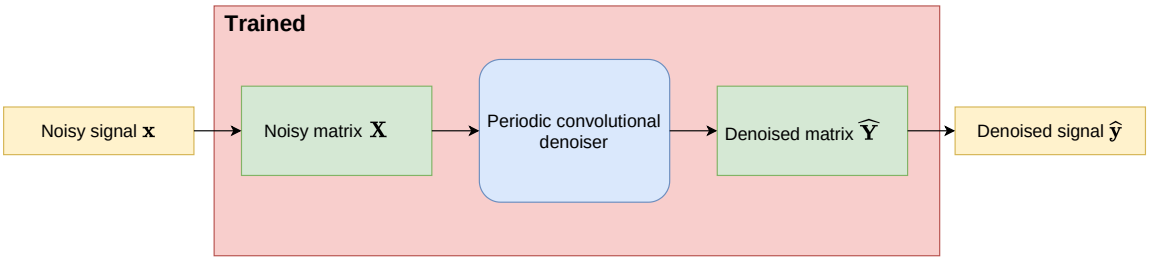


Figure 4.5: Framework for the method with periodic architecture

both dimensions of the image. In the case of the periodic matrices, it is expected that a longer receptive field is needed for the temporal dimension than for the periodic dimension. As a consequence, we rather leverage a 1D denoiser by modifying some of its 1D kernels by 2D kernels. The network is then able to learn itself how to exploit the periodicity. The advantage of this method compared to the moving average preprocessing is its ability to use both the local content and the periodicity simultaneously, instead of first using the periodicity, then the local content. This means that depending on how it has been trained, the network might be able to decide itself whether the periodicity is exploitable and should be used, or the local content should be used alone. As a consequence, this method should gain robustness for cases where the parallel rows of the matrix do not contain the same phase information, which might be the case because the periodicity has locally been lost or more likely because the fundamental frequency detected is not correct, which would result in small shifts between the rows. In this second case, a network properly trained might be able to adapt to access the periodicity despite the shifts. The counterpart is that the amount of parameters will be a bit larger and the zero-padding becomes necessary also on the second dimension and its impact cannot be reduced as easily as by just correcting the factor.

The global framework of this method can be seen in Figure 4.5.

In the next parts of this thesis, we will denote the architectures using this process as periodic 1D architectures, shortened in P1D.

4.2.3.2 Incomplete or over-complete rows for training

Training the periodic architecture requires matrices of periodic data. Even though the process of creating the matrix is independent of the frequency, the amount of variations within the receptive field will depend on how high the frequency is. Therefore, for training the architecture, matrices have to be created with a large set of frequencies, despite the theoretical better generalization to unseen frequencies. This might create two issues:

1. For two different frequencies, the amount of samples in one period will be different, implying that the matrices will have different shapes, even if they have the same amount of periods. This problem is mostly computational as most frameworks have predefined functions for trainings, but require the same shape for all training data.

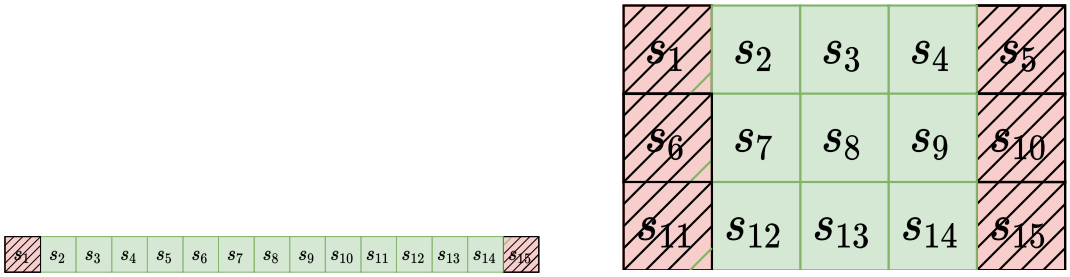
2. For small frequencies, the amount of samples in one period might be very high, and the matrices will require a huge amount of memory. Performing the training with full matrices might require reducing the batch size or the architecture.

Our solution for tackling these problems is to generate incomplete (for low frequencies) or over-complete (for high frequencies) rows. For that, we take a clean signal $\mathbf{y}$ of length T, with T the desired length of each row. We generate P realizations of a noisy signal $\mathbf{x}$ which all have $\mathbf{y}$ as the clean contribution. As the concatenation of the signals results in a signal having a periodic ground truth, the P created rows have the same properties as a matrix that would have been generated by first creating a long periodic signal of periods of T samples, then cutting it into P rows. The only difference will be that there will not be anymore correlations between the end of a row and the beginning of the next row. However, as the receptive field will look only a part of each period, the beginning and the end of the two rows will not be used together and the absence of correlations will have no impact.

This means that we can generate training signals with a signal $\mathbf{y}$ that has been first generated with a periodic model with low or high frequencies (actually, it would even be possible to use a non periodic model) and still obtain the same shape for each matrix.

A similar process could be applied with a real signal $\mathbf{x}$ of period with T_r samples. We could generate the first row as $[\mathbf{x}_0, \dots, \mathbf{x}_{T-1}]$ then the second row as $[\mathbf{x}_{T_r}, \dots, \mathbf{x}_{T_r+T-1}]$, ... If $T > T_r$, then some information would be repeated (the end of a row would be the same as the beginning of a next row), and if $T < T_r$, some information would be removed (the process could be seen as extracting a sub-matrix from the full matrix with rows of T_r samples), but in both cases, the resulting matrix would be adequate for training.

Let us notice that using incomplete or over-complete rows could be done outside of training, e.g., if we want to denoise a specific part of the matrix instead of the entire signal. However, in our applications, we assume that all parts of the signals have to be denoised and use incomplete rows only during training. Over-complete rows are also used during inference, as discussed in Section 4.2.3.3.



(a) 1D architectures use zero-padding for the extremities

(b) Naive method
Higher impact of zero-padding

Figure 4.6: Illustration of the problem appearing when each row is created with only the current period

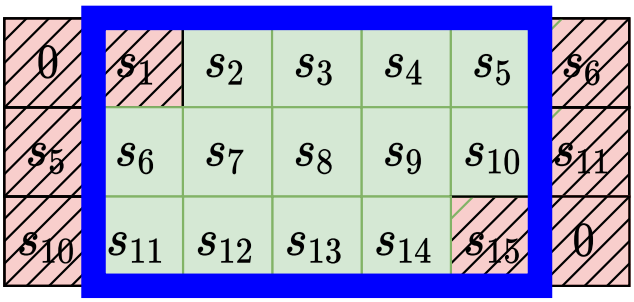
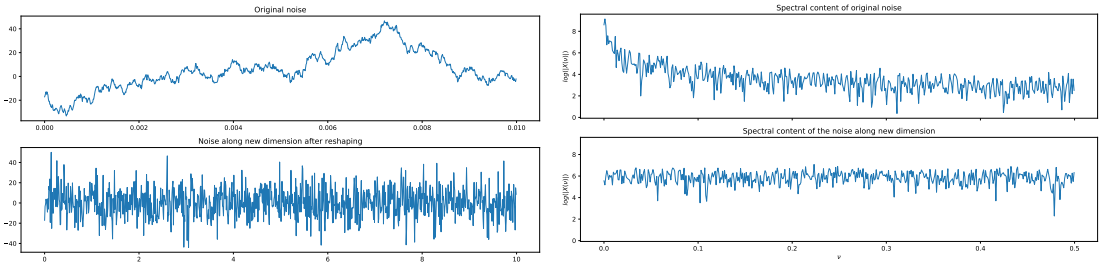


Figure 4.7: Illustration of the solution for avoiding zero-padding along the temporal dimension. In this representation, the first row is at the bottom of the created image.

4.2.3.3 Avoiding too much zero-padding on the temporal dimension

Another problem with the creation of the matrix is the amount of zero-padding along the temporal dimension. If only the second dimension of some kernels is changed, then the receptive field of the temporal dimension remains the same. Therefore the need of zero-padding for completing the extremities of the signals will now appear for each row, resulting in an increased amount of samples impacted by zero-padding, as illustrated in Figure 4.6. Here, we will propose a method for solving this problem during inference. During training, this problem will still occur.

Even though it is possible to consider that this problem is minor as the training data will also face it which should imply that the models are able to adapt to zero-padding, here we propose to use similar properties as the ones used for designing training signals. The easiest would be to put the full signal completed by zeros at each row (then only the central part of the matrix is preserved for building the denoised matrix), however, this solution implies creating a matrix that has much more samples than the original signal. Here, we extend each row by only what is required to avoid the zero-padding based on the length of the receptive field. The process is shown in Figure 4.7. Here, only the part of the signal inside the blue box is preserved after denoising, resulting in the same amount of denoising along the temporal dimension as in the 1D denoising.



(a) In the time domain

(b) In the frequency domain

Figure 4.8: Whitening of the noise by reshaping the data.
For both subfigures, the top subplot corresponds to the original signal, the bottom subplot is the signal obtained by extracting the samples along the periodic dimension of the periodic matrix.

Here, the figures do not show it, but the problem of zero-padding will also appear along the periodic dimension. Again, this is also true during training and the models should therefore be able to adapt partially to this problem. However, depending on the shape of the training data, this phenomenon might appear more often during inference. Especially, for data with only a few periods (or only one period), zero-padding will be used both before and after the current period, which is a case that will not be encountered in the training data, and it is unsure how the networks will generalize to this specific case. For these signals with a very limited amount of periods, the method based on a preprocessing is expected to perform better as we already handle the problem by reducing the length of the moving average near borders (and it is actually possible to use only one period and retrieve the exact same results as by performing the 1D models).

4.2.4 Theoretical advantages of the proposed methods

We already mentioned the first advantage of our method, which is the decorrelation of the noise signal along the periodic dimension. The noise along the periodic dimension should therefore be close to a white noise, as illustrated in Figure 4.8, where the second signal is obtained by taking one sample out of T samples from the first signal, which is a colored noise obtained by performing a moving average on a white noise.

Due to this property, the models should be able to denoise signals containing colored noise in a much more efficient way.

This property also means that it becomes easier to differentiate colored noises from periodic components. A colored noise should be whitened by the new shape of the data and periodic models should be able to remove it. Periodic components will however remain even after

whitening of the signal, and the models will be able to know that these components should not be removed.

Another consequence that will make the denoising easier, is that as the ground truth is assumed to be constant along the periodic dimension. With a stationary noise along this dimension, which would be obtained if we assume that the mentioned decorrelations are sufficient to obtain a white noise, the process will be closer to a stationary signal along the periodic dimension. This is also a reason why even the method based on a preprocessing should already improve the results.

Additionally, using a new dimension also means that the models can observe a larger amount of samples without needing to use a large receptive field in the temporal dimension. With a small receptive field, this implies that it is possible to assume that the signal is locally stationary, except if the receptive field contains a discontinuity. Therefore, the models should be able to perform a regularization based on the new dimension, even if the loss term contains only the reconstruction term.

Lastly, if the evolution of the second order statistics of the noise is slow, then it is possible to assume that it is constant within the length of the temporal receptive field. Therefore, training with stationary noise could result in networks able to consider the second order cyclostationarity, at least partially. In our case, we will use white noise, implying that this assumption will be true only for slow variations of the variance of the noise, but other stationary noises would be possible to consider other kinds of correlations.

4.3 Experiments on friction signals

4.3.1 1D models

Here, we use three 1D models for comparison with the new architectures (and implementations of MA Nets, as we will detail in Section 4.3.2). The first two architectures are the medium and original architectures from previous chapters, as presented in Table 3.1. For the original architecture, we take the model trained without regularization. For the medium architectures, we take the model trained with $\gamma = 1$ based on the observations made in the previous chapter.

Additionally to these two architectures, we also train a new architecture based on the Wave-U-Net [11] model presented in Section 1.5.6. Here, we use an implementation with seven downsampling blocks, where a downsampling block is a convolutional block followed

Table 4.1: Details of the UNet architecture

Downsampling blocks	7
Kernel lengths (downsampling)	$15, 13, 11, 9, 7, 5, 3$
Channels (downsampling)	$8, 16, 32, 64, 128, 128, 128$
Channels (upsampling)	$128, 128, 64, 32, 16, 16, 8$
Receptive field (temporal dimension)	$\approx$ 1600 samples

by a downsampling operation. Similarly, we will denote a convolutional block followed by an upsampling as an upsampling block. The downsamplings are performed with max poolings and the upsamplings are performed with transposed convolutions. The convolution layer in the i-th (starting at $i = 0$) downsampling block has a kernel of length $15 - 2 \times i$, while the convolution layers in the upsampling blocks all have a kernel of length 5. The last convolution is a 1×5 convolution. The amount of channels in the downsampling blocks are 8 for the first convolutional block, 16 for the second, 32, 64, 128, 128 and 128. For the upsampling, they are 128, 128, 64, 32, 16, 16 and 8. This means that we create more features of low resolution signals than high resolution signals. This architecture is trained with the same algorithm as the WaveNet architectures from the previous section, with no regularization, and signals of 20480 samples instead of 20000, as the amount of samples has to be a multiple of 2^7). Here, it would be possible to use larger batches, but we preferred keeping the same batches to make an easier comparison. The epochs were around 10 times shorter with this model than for the $medium$ architectures, despite a similar amount of parameters (660000 parameters for the Wave-U-Net, 600000 for the medium architecture). We denote this architecture as UNet. A summary of the implementation is given in Table 4.1.

4.3.2 Experimental setup for MA Networks

For these experiments, we simply apply the preprocessing with a moving average of size 9 along the periodic dimension, i.e., we use the moving average presented in Equation (4.2) with $K = 4$. Then, we apply the 1D models, resulting in three models:

- MAOriginal by applying the original model after preprocessing;
- MAMedium by applying the medium architecture after preprocessing;
- MAUNet by applying the UNet architecture after preprocessing.

No additional training is performed for these architectures.

Based on estimation theory, it is possible to estimate the gain in terms of SNR after preprocessing and before denoising by the deep learning architectures. If we assume that all samples within the average contain an independent noise, or at least that the dependencies should be small, the variance should be divided by 9. As the variance in this case can be estimated by the power of the noise, then it is possible to assume that the power of the noise is divided by 9, and therefore the SNR_{old} becomes:

$$SNR_{new} = 10\log_{10}(9) + SNR_{old} \tag{4.4}$$

i.e. that the SNR is improved of 9.54 dB even before denoising.

4.3.3 Experimental setup for P1D models

4.3.3.1 Exactly periodic architecture

For testing our method, we build three models:

- The first model is based on the "small" architecture presented in the previous chapter. Its architecture remains the same, except for the third and sixth residual layers of each stack, where the 1×3 convolutions are replaced by 3×3 convolutions. Dilation is applied only along the temporal dimension. We denote this architecture as "PSmall". As for the 1D model, the temporal receptive field is of 257 samples.

- The second model is based on the "medium" architecture. As for the "PSmall" architecture, it is almost unchanged compared to the 1D architecture, except for the fourth and eighth layers of each stack, where the 1×3 convolutions are replaced by 3×3 convolutions. Dilation is applied only along the temporal dimension. We denote this architecture as "PMedium". The temporal receptive field is of 1025 samples.

- The third model is based on the "UNet" model. Here, we change the 1×9 (resp. 1×3) convolution of the fourth (resp. seventh) downsampling block by a 3×9 (resp. 3×3) convolution. Similarly, the 1×5 convolutions of the second and fifth upsampling block are replaced by 3×5 convolutions. The max-poolings and transposed convolutions do not affect the periodic dimension. We denote this architecture as "PUNet". The temporal receptive field is of approximately 1600 samples.

Here, as all architectures use four convolutions with a local receptive of length 3 along the periodic dimension, they all have a global receptive field of length 9 along this dimension, which results in using the same amount of periods as for the preprocessing-based models.

The algorithm for training these models is similar to the one used for training the 1D models. The MAE is minimized during the training (without regularization), with a learning rate starting at 0.001 and divided by 10 after 20 epochs. The best model is chosen based on results on a validation set. The only difference with the $1D$ models is for the training data. Instead of generating signals of 20000 samples, we generate matrices of 10 rows, with 2048 samples in each row. Each row of the grid has the same ground truth, generated as the weighted sum of a square wave and a sine wave, exactly as what we did for the 1D models. The rows of the grid differ only by the noise that is added. Here, we use only white noise. The frequencies and initial SNR remain the same as the ones used for training the 1D models.

4.3.3.2 Training with shifts

In addition to all the exactly periodic models, we also train two additional architectures:

- PMedium*, which has the same architecture as PMedium;
- PUNet*, which has the same architecture as PUNet.

The difference with exactly periodic architectures lays in the training data. For half of the training grids, the phase of each row contains an additional component that is defined by a uniform law $\mathcal{U}([-0.05, 0.05])$. This process implies that the models should also be able to adapt to small shifts between the rows, and be able to try to look within the receptive field for finding the adequate samples in each period, instead of just considering that they are always at the same position. In cases where the shifts are too large, the networks might not be able to observe the same phenomena at different periods simultaneously. In such cases, they should learn to rather focus only on the current period, which might also increase robustness to cases where the amount of periods is small, where the phenomena are only available in a limited amount of periods.

4.3.4 Summary of all architectures

In Table 4.2, we show a brief summary of all models and recall if they contain 2D convolutions for exploiting periodicity, exploit a MA preprocessing, are trained with shifts and which architecture is used as a base.

Table 4.2: Summary of the models depending on how they are built, trained and used.

Base architecture	Shifts (training)	MA preprocessing	2D convolutions	Name
Original	X	X	X	Original
Medium	X	X	X	Medium
UNet	X	X	X	UNet
Original	X	✓	X	MAOriginal
Medium	X	✓	X	MAMedium
UNet	X	✓	X	MAUNet
Small	X	X	✓	PSmall
Medium	X	X	✓	PMedium
UNet	X	X	✓	PUNet
Medium	✓	✓	✓	PMedium*
UNet	✓	✓	✓	PUNet*

4.3.5 Further details of the training algorithm

As in Section 3.3, we train with 10000 generated synthetic signals. We use batches of 8 signals and train for 30 epochs, with an initial learning rate of 0.001, divided by 10 after 20 epochs. The Adam optimizer is used to minimize the MAE and the best model is selected based on the validation score obtained on a set of 3000 signals. Let us notice that in this chapter each model uses a different realization of the training dataset to be trained.

We use the same procedure as in Section 3.3, i.e. we generate each clean signal as the randomly weighted sum of a square wave and a sine wave, generated with a randomly selected frequency, and we add a white noise to obtain a randomly defined initial SNR. Both components have a random phase shift. The possible values for the frequency are unchanged, i.e., the possible values are $\{0.25, 0.5, 1, 2, 4, 8, 16, 24, 32\}$ Hz. Similarly, the possible initial SNR are $\{-10, -5, 0, 5, 10, 20\}$ dB.

4.3.6 Results on synthetic signals

4.3.6.1 Reference method for comparison

As a reference method not based on Deep Learning, we use the non-local-means method of [60]. For our applications, we use a bandwidth of $\lambda = 0.6\sigma$ with σ the standard deviation of the noise, which has to be known before denoising for applying this method, a patch half-width of 500 samples and a neighborhood half-width of 10000 samples. With this neighborhood, exploitation of the periodicity will not be possible for low frequencies but will become possible for high frequencies. A higher value would have allowed exploitation of the periodicity even for low frequencies, but would also have increased the computation

time. This method will be denoted as "NLM". Due to its high time complexity, we will perform the results of this method only for a limited amount of experiments.

Even though there might be some methods which would give better results (especially based on Deep Learning), this method is well suited for processing periodic signals. Additionally, the 1D models (without preprocessing) can already be seen as reference deep learning methods, especially considering that P1D models are meant to improve some specific architectures, i.e., we are not looking for a model that would beat state-of-the-art for sure, but rather for a method that improves a specific model).

4.3.6.2 Method for the evaluation

For the results presented in this section, all evaluations are performed on a set of 60 signals containing 100000 samples. For each evaluation, all signals have the same initial SNR and the same frequency, which will allow a more specific analysis of the results compared to the previous chapter. The frequency is assumed to be known previously (we do not perform any detection) and the grids are generated based on this previously known frequency, using the method illustrated in Figure 4.7 for limiting the impact of zero-padding.

The set of frequencies tested for the evaluations is $\{1, 5, 10, 15, 20, 25, 35, 45, 60\}$ Hz. Signals at 1Hz will contain only one period and can be seen as cases where the periodicity is not exploitable, and signals at 5 Hz can be seen as cases where it is exploitable but only gives a limited amount of information. For the initial SNR, we use the set $\{-30, -25, -20, -15, -10, -5, 0, 5, 10\}$ dB.

We also test two different kinds of noise (white and colored, we will detail the implementation of colored noise later) and two data models, the Andersson model and the model used for training. Due to its large computation time, we will implement the "NLM" method only for Andersson data with white noise, which will already be sufficient to show that this method is outperformed for the signals in this section.

During the training, each model was trained with a different realization of the dataset. For these evaluations, all models use the same realizations of the datasets, i.e., the dataset for a specific frequency and initial SNR is created only once and shared for all models.

Another difference with the training is that the shapes of the grids might change depending on the frequency of the signal. For example, with a frequency of 10 Hz, the obtained signals would have 10 rows of 10000 samples. With a frequency of 25 Hz, the grid would contain 25 rows of 4000 samples. Let us notice that each row would then be extended by using the trick presented in Section 4.2.3.3 to limit the impact of zero-padding.

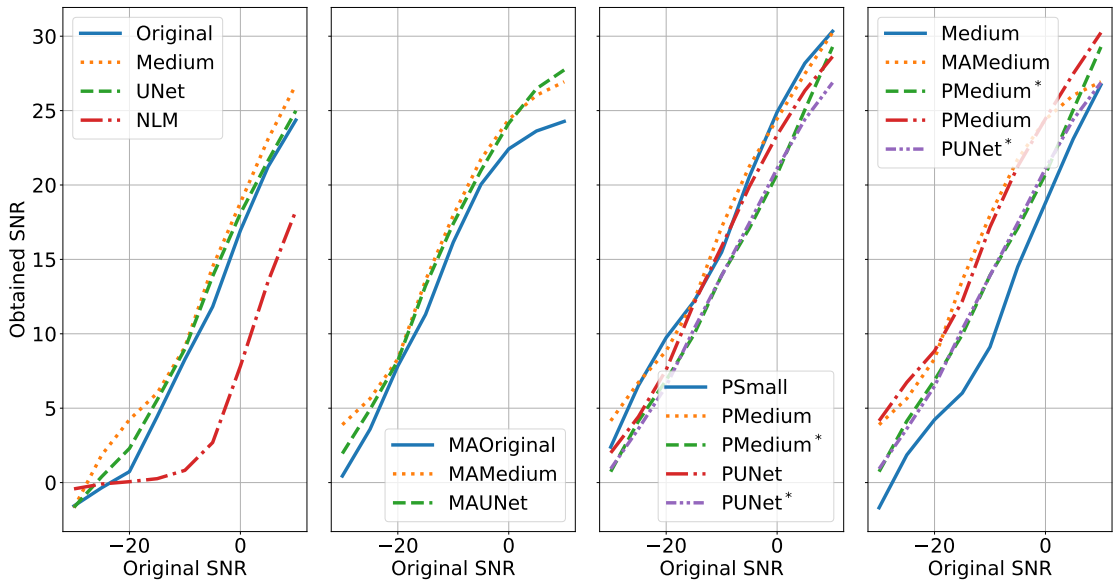


Figure 4.9: Evolution of the obtained SNR for signals at 60 Hz generated with the Andersson model and white noise
The fourth graph gives redundant information, but makes the comparison between the different methods easier

4.3.6.3 Quantitative results with white noise

Signals generated with the Andersson model

We first evaluate with the Andersson model with white noise. in Figure 4.9 we show the results for signals at 60 Hz, with various initial SNR. The results clearly show that the "NLM" method is largely outperformed by all Deep Learning based methods, including 1D models without preprocessing. There is also a clear gain of methods using the periodicity over 1D models, even though MA-models seem to reach a limit for high initial SNR. The methods trained with shifts are clearly outperformed by the exactly periodic models, including PSmall, but are still able to outperform the 1D models, which proves that they manage to access at least partially the periodicity. Generally, the WaveNet based models seem to obtain better results than the Wave-U-Net based models, even though the difference remain small in most cases.

Similarly, Figure 4.10 shows the results obtained with a fixed initial SNR of -5dB, with various fundamental frequencies. Again, the "NLM" method is largely outperformed by Deep Learning based methods in most cases. However, the exactly periodic models give poor results for signals at 1 Hz, even compared to "NLM". They start outperforming other models only after reaching higher frequencies, showing that they are highly dependent on easily accessible information about the periodicity. Models trained with shifts also have a drop of performance for signals at 1 Hz, but still outperform the "NLM" method in this

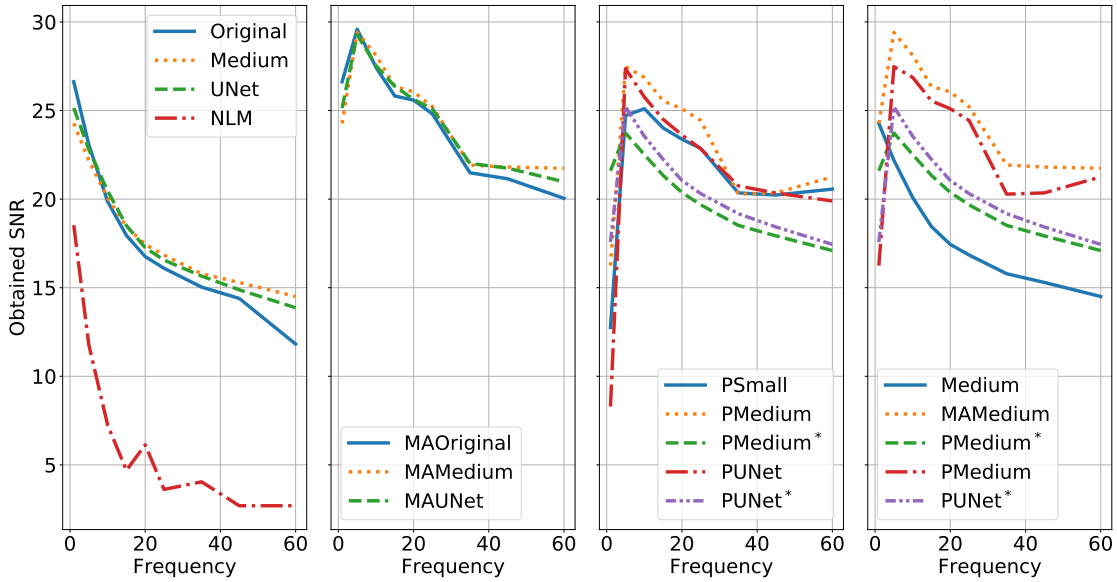


Figure 4.10: Evolution of the obtained SNR for signals generated with the Andersson model and white noise and an initial SNR of -5 dB

case and get results close to the ones of 1D models. Even though this shows that these models are not adapted to the case of non-periodic signals (or more generally, signals where the periodicity is not accessible), this behavior exhibits stronger robustness than exactly periodic models. Due to the preprocessing that allows them to perform the same denoising as 1D models when only one period is available, MA-based models obtain satisfactory results for all frequencies. Here, it is also possible to see that for some frequencies, there is a drop of performance for periodic models trained without shifts (for example at 35 Hz and 45 Hz). It is possibly due to the fact that these frequencies imply an incomplete last row, which is an unseen case in the training data. However, there might be other reasons as the signals at 60 Hz also have an incomplete last row, without observing a similar phenomenon. Additionally, despite this drop of performance, the periodic models remain efficient and obtain much better results than the 1D models. We can also observe that the WaveNet based models still seem to outperform the Wave-U-Net based models when trained without shifts, but the model PUNet* now outperforms the model PMedium*. Again, the differences remain small.

From the two figures, it seems that despite their saturation for high initial SNR, MA-methods outperform, or at least get similar results as periodic models in most cases.

In Figure 4.11, we show an example of signal generated with the Andersson model, and the results for some of the models. This result shows that despite the large noise, periodic models (even the ones trained with shifts), are able to preserve the discontinuities, and even the peak is partially preserved.

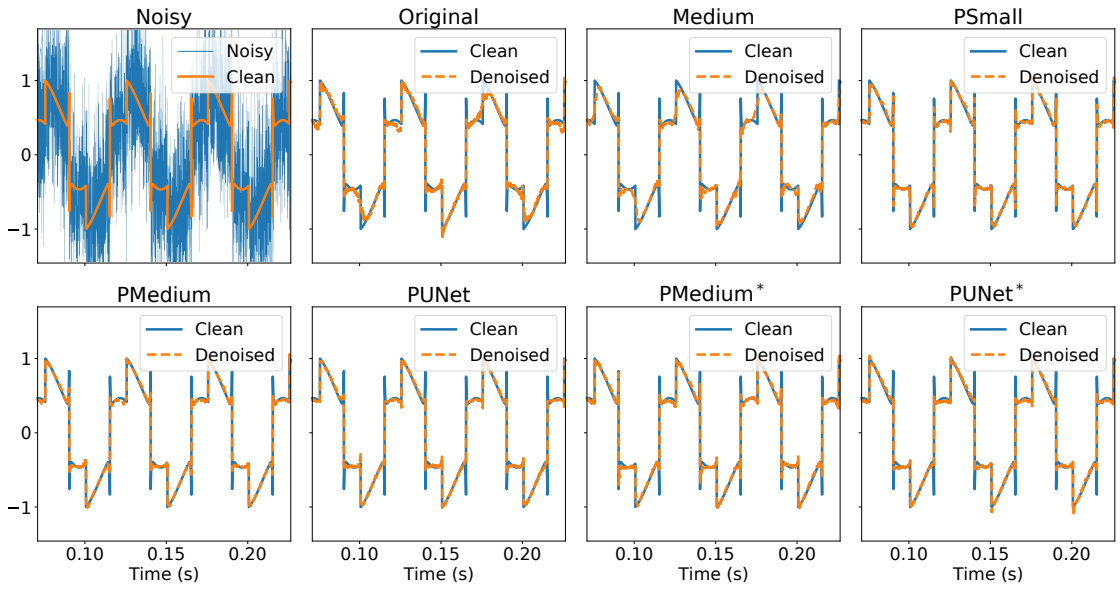


Figure 4.11: Example of results for a signal generated with the Andersson model

Signals generated with the model of training data

As a complement to the study on signals generated with the friction model, we show some results obtained by generating the data with the same model as the one used for generating training data.

In Figure 4.12, we show the evolution of the results for signals at 60 Hz, with various initial SNR, and in Figure 4.13, we show results for an initial SNR of -5 dB, with various frequencies. It appears that there is no significant difference with the observations made with evaluations on Andersson data.

We show an example of signal and some denoisings in Figure 4.14. Again, the discontinuities and peaks are well preserved by periodic models. By focusing on a discontinuity (Figure 4.15), it is possible to see that the 1D models tend to have difficulties preserving some discontinuities, despite being trained for this kind of data.

Signals generated with the Andersson model and an error in the frequency

In the previous experiments, we assumed knowing the exact frequency of the signals. While this is convenient for a theoretical study, obtaining this setting with real signals will be rare for two reasons:

- The detected frequency will rarely be the exact fundamental frequency;

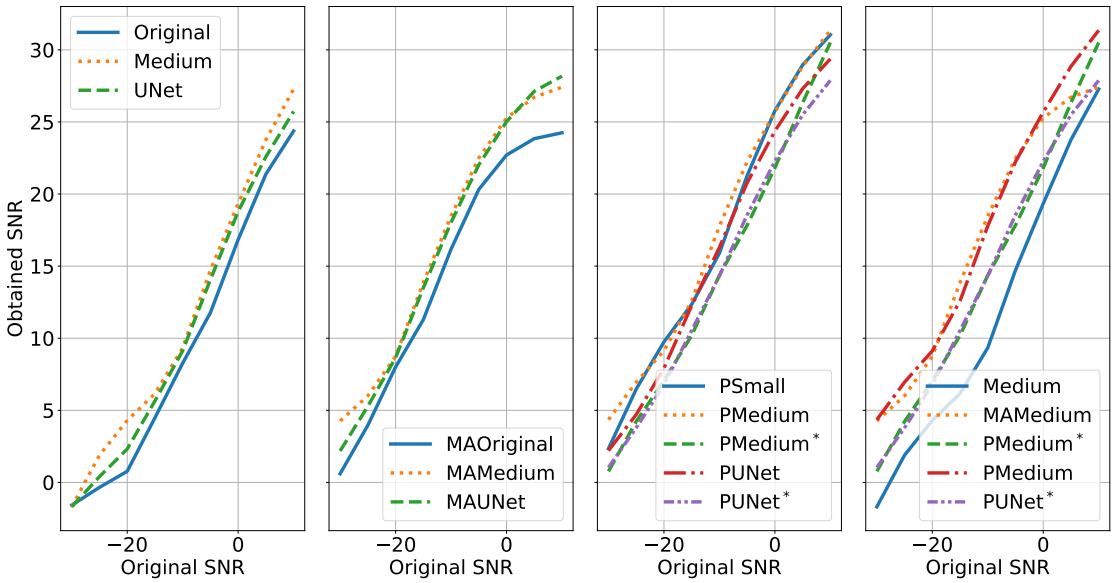


Figure 4.12: Evolution of the obtained SNR for signals at 60 Hz generated with the training model and white noise

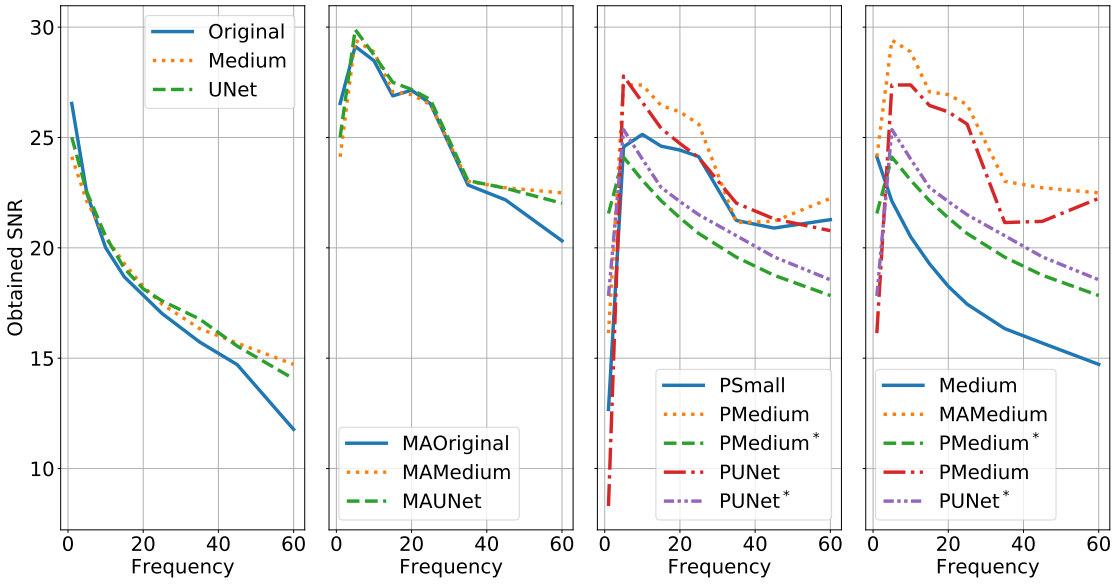


Figure 4.13: Evolution of the obtained SNR for signals generated with the training model and white noise and an initial SNR of -5 dB

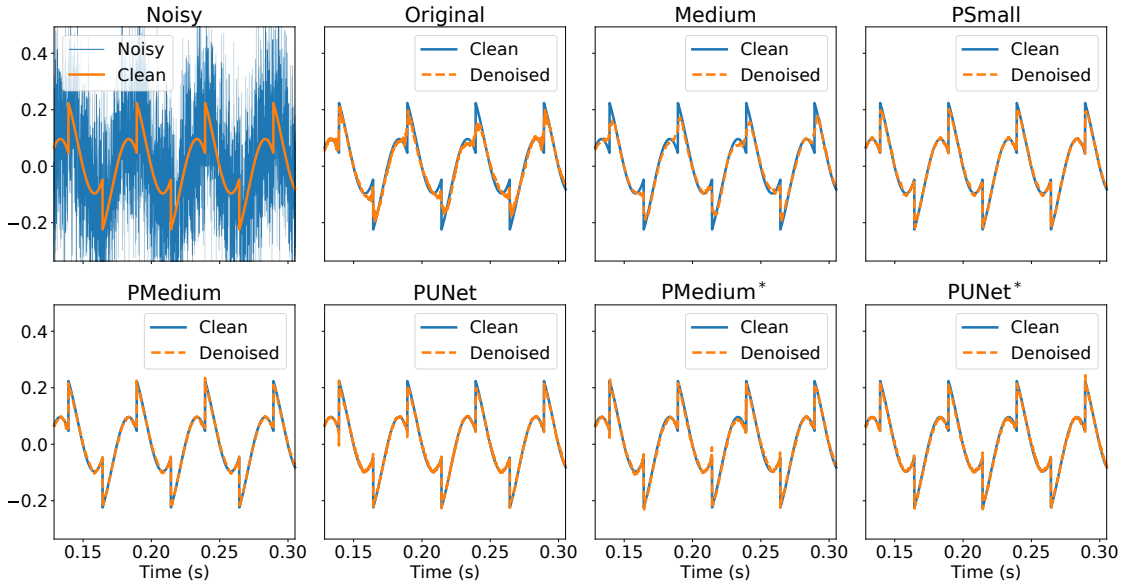


Figure 4.14: Example of results for a signal generated with the training model

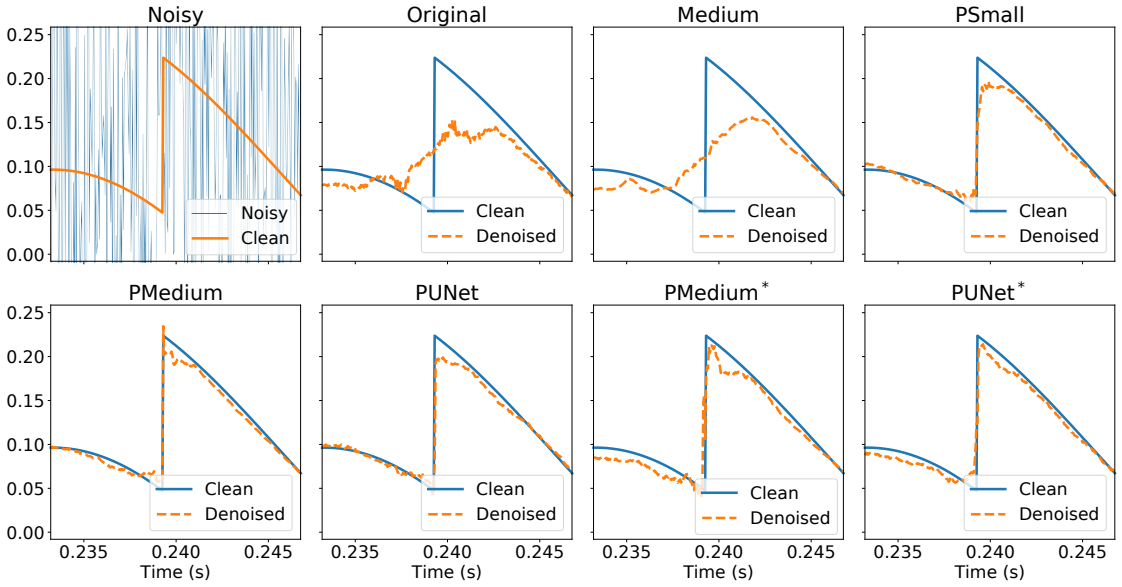


Figure 4.15: Zoom on one discontinuity of Figure 4.14

Table 4.3: Results (in dB) with a random given frequency, resulting in a shift between the lines of the created matrix

Frequency	20 Hz		45 Hz		60 Hz	
Initial SNR	**-10**	**10**	**-10**	**10**	**-10**	**10**
Original	11.17	30.46	9.23	27.49	8.28	24.36
Medium	12.38	31.56	10.00	28.96	9.12	26.74
UNet	11.38	30.01	9.28	27.34	8.98	24.99
MAOriginal	14.46	16.67	14.67	18.07	13.39	17.08
MAMedium	14.96	16.77	**15.33**	19.07	**15.16**	19.21
MAUNet	14.63	16.73	15.17	19.08	14.55	19.38
PSmall	14.26	16.29	14.11	18.24	13.66	19.01
PMedium	14.45	17.21	14.43	20.15	14.37	20.93
PMedium*	14.88	**32.35**	14.23	**30.82**	13.69	**29.11**
PUNet	14.61	18.74	15.02	21.45	14.91	21.92
PUNet*	**15.34**	29.99	14.32	27.92	13.66	26.55

- Even with a perfect detection, it is rare that the frequency remains exactly the same during the full recording.

As a consequence, processing real signals will result in some shifts between the rows of the created grid. In terms of processing, it does not make any difference whether these shifts are due to variations of the fundamental frequency or error in the detection. Therefore, we perform again some of the previous evaluations with Andersson data, but we assume a random frequency that is uniformly distributed in $[0.99f_0, 1.01f_0]$ with f_0 the correct fundamental frequency, rather than using f_0. Here, we show in Table 4.3 the new evaluations for signals at -10 or 10 dB, with fundamental frequency of 20, 45 or 60 Hz. These results clearly show that despite their good results in the previous evaluations, exactly periodic models and MA-based models are not able to adapt well to these shifts. Even though their results remain satisfactory for an original SNR of -10 dB, they do not have an advantage (or at least not a significant one) over models trained with shifts anymore. With an original SNR of 10 dB, these models are now clearly outperformed by all models, including 1D models. MA-models and PModels get extremely close results, which shows that it is likely that the process for combining the periods learned by PModels is close to an average of the periods within the receptive field. Models trained with shifts show much better behavior. They have almost no drop of performance compared to the non-shifted case and are able to remain competitive both with low and high initial SNR. At -10 dB, they clearly outperform 1D models, and at 10 dB, they obtain similar (or better) results, showing that they are still able to exploit periodicity even when the rows do not share exactly the same phase.

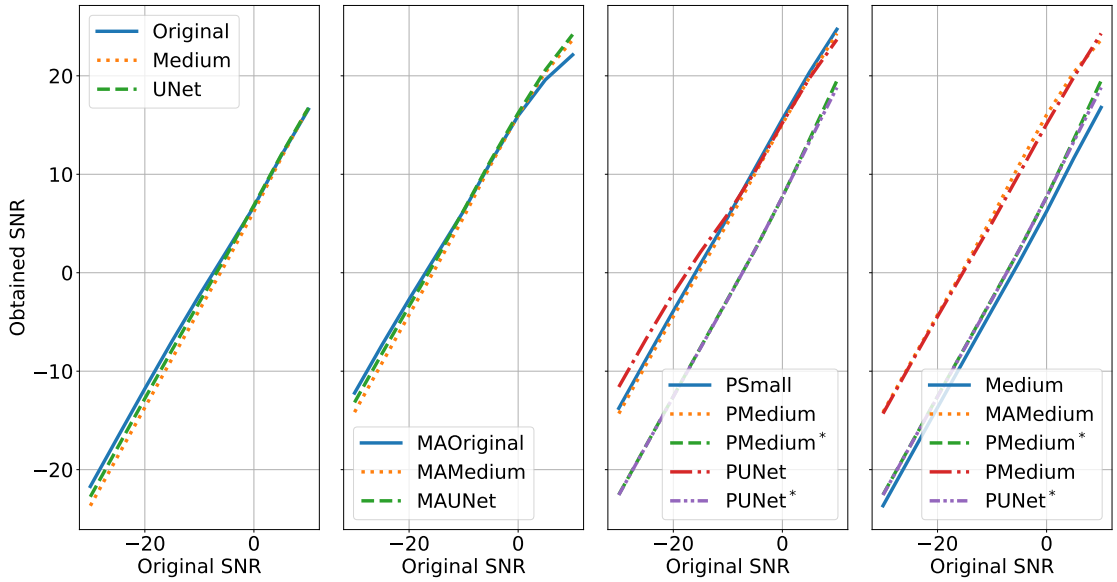


Figure 4.16: Evolution of the obtained SNR for signals at 60 Hz generated with the Andersson model and colored noise

4.3.6.4 Quantitative results with colored noise

Based on the previous results, we detail only the results obtained with the Andersson data, as the evaluations made on signals generated similarly as training signals did not bring any significant information. Here, colored noise is generated as:

$$\xi \mathbf{e}_c + (1 - \xi)\mathbf{e}_w \tag{4.5}$$

where $\mathbf{e}_w$ is a white noise and $\mathbf{e}_c$ is obtained by applying a moving average of 200 samples on white noise (which results in a low-pass filter with cutoff frequency of 221.5 Hz). For these experiments, we use $\xi = 0.9$.

Let us notice that these colored noises are generated only for evaluation signals. No further training is performed compared to the previous sections, i.e., all models are still the ones trained with white noise. This section can therefore be seen as a demonstration of the improved ability to generalize to various kinds of noises when exploiting the periodicity. Additionally, the fundamental frequency for the signals in this section is assumed to be exactly known.

In Figure 4.16 we show the evolution of the obtained SNR for signals with a fundamental frequency of 60 Hz. The exactly periodic models, as well as the MA-based models, obtain good results, confirming the previous assumptions of robustness to colored noises. However, the models trained with shifts clearly lose interest for low initial SNR and outperform 1D models only when reaching higher initial SNR.

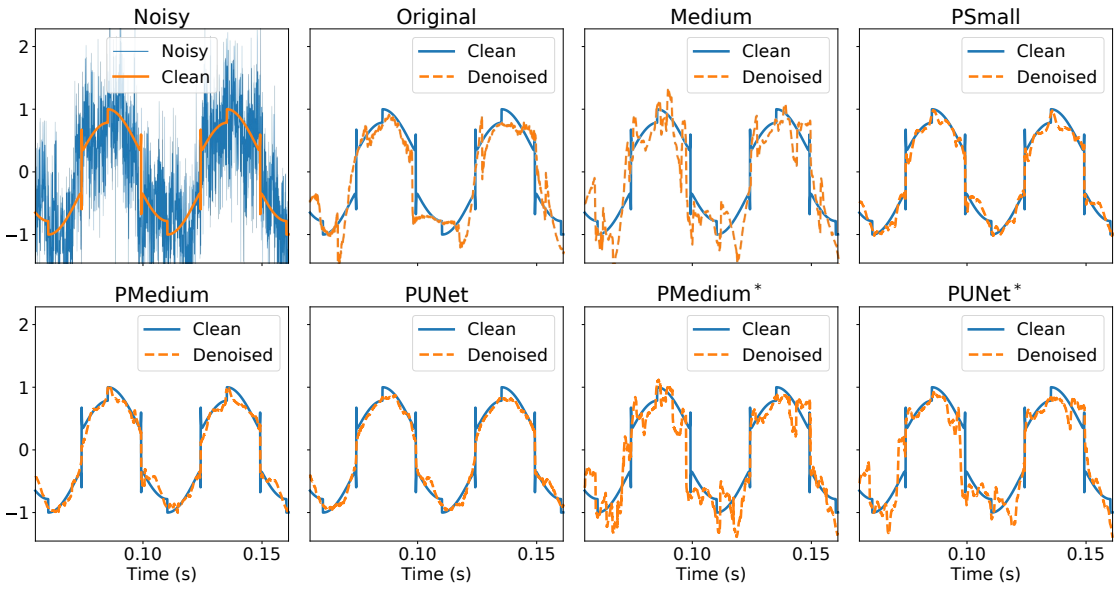


Figure 4.17: Example of results for a signal generated with the Andersson model and colored noise

In Figure 4.17, we also show an example of denoised signal. This result clearly illustrates that the 1D models are not able to differentiate colored noises from local phenomena. It is visually hard to decide based on the noisy signal only, whether the signals are always at the same phase or not. As they have been trained only with signals where this is the case, P1D models, as well as MA models through the preprocessing, assume that they can still use the periodicity by just considering that this is still the case (which is true for these evaluations). However, models trained with shifts might decide to rather focus on the current period as they are not able anymore to recognize the same patterns in other periods, explaining why their results are so close to the ones of 1D models, even though they seem a bit better in terms of detection of the largest discontinuities.

4.3.7 Results on real data

As a last experiment on friction signals, we also perform some evaluations on real signals. We use the setpoint frequency to create the grids. We show some results at low frequency (2 Hz) in Figure 4.18. Despite a low level of noise, exactly periodic and MA-based models clearly give poor results around the peaks. The similarities of the results obtained by these two kinds of models tend to indicate that training without shifts results in a similar behavior as averaging the periods. Models trained with shifts are able to retrieve similar results as 1D models.

In Figure 4.19, we show some results with a high frequency (12.5 Hz) signal. Here, the mechanism gains in stability and the shifts due to the error in the setpoint frequency are

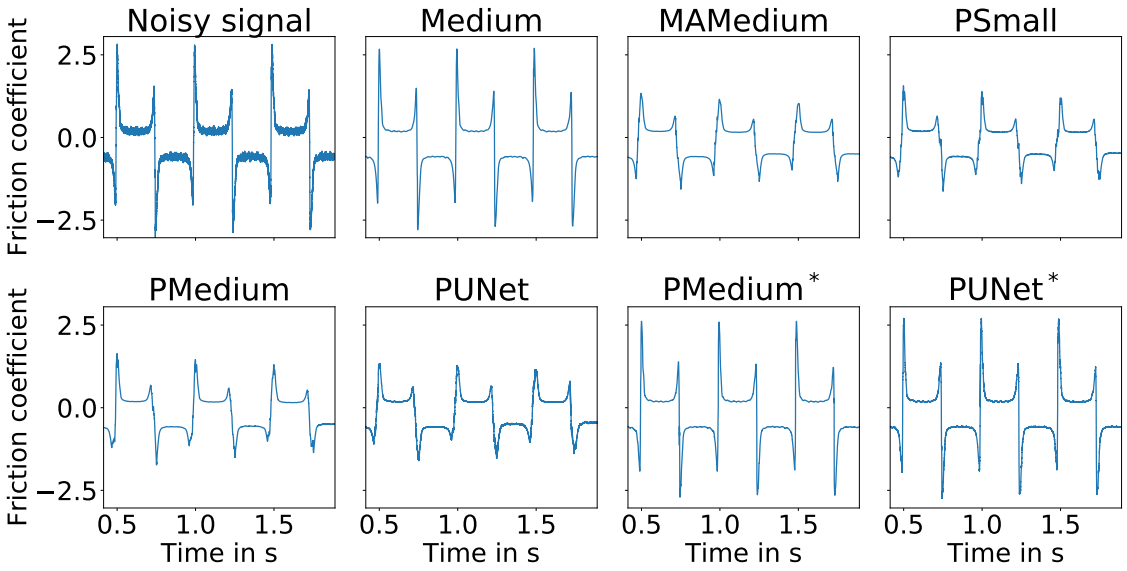


Figure 4.18: Example of denoising for a real friction signal at low frequency (2 Hz)

reduced, resulting in satisfactory results for exactly periodic models. However, it is possible to observe that all models seem to be affected by some phenomena. It seems (especially from results of PMedium) that at least some of these phenomena are periodic, but some might also be part of a colored noise, and it is hard to distinguish them with these results.

By superposing three consecutive periods (Figure 4.20), it seems that the phenomenon detected by the PMedium and shown in the zoom is indeed part of a periodic pattern which could possibly be due to the sensor or to vibrations of the mechanism, or to local modifications of the viscous properties. All models (even 1D models) seem to detect this phenomenon (even though most of them attenuate it) and the noisy signal seems to indicate that most oscillations in this area are indeed occurring at the same time in all periods (whereas a colored noise would more likely result in similar oscillations but different phases).

4.4 Experiments on ECG signals

4.4.1 Motivation of the experiment

The previous experiments clearly showed significant improvements compared to the 1D method. However, the models trained without shifts are clearly not able to adapt well to the variations of the periodicity appearing in real signals. Concerning the models trained

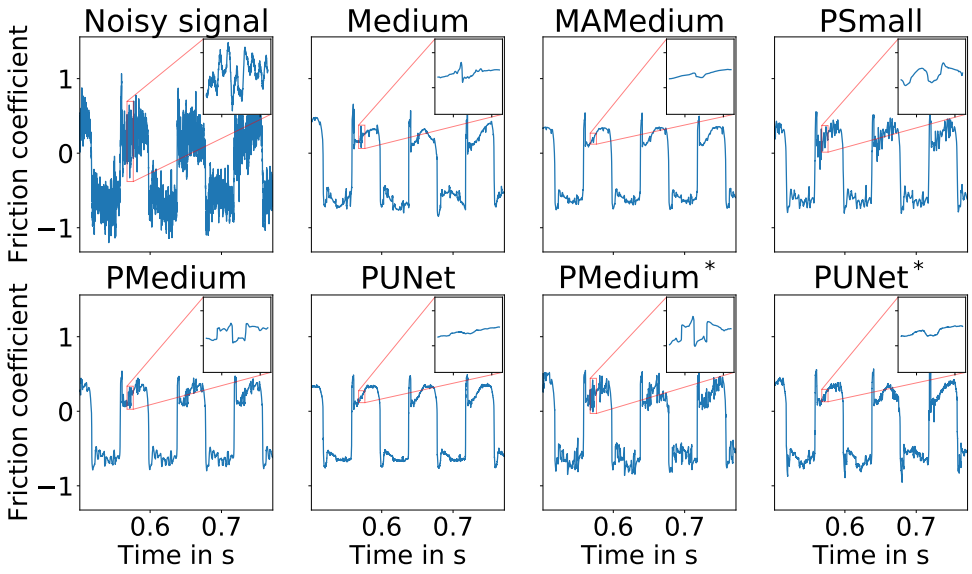


Figure 4.19: Example of denoising for a real friction signal at high frequency (12.5 Hz)

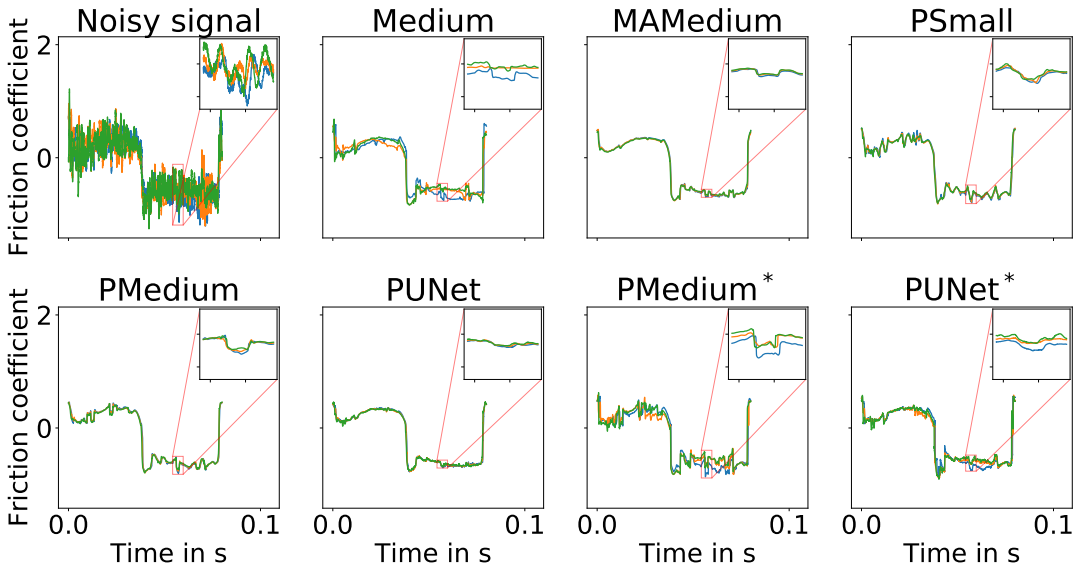


Figure 4.20: Superposition of three consecutive periods of Figure 4.19

Table 4.4: Details of the 1D architectures

	$\mathbf{UNet}_3$	$\mathbf{UNet}_4$	$\mathbf{UNet}_7$
Downsampling blocks	3	4	7
Kernel lengths	$7, 5, 3$	$9, 7, 5, 3$	$15, 13, 11, 9, 7, 5, 3$
Channels (downsampling)	$32, 64, 128$	$16, 32, 64, 128$	$8, 16, 32, 64, 128, 128, 128$
Channels (upsampling)	$128, 64, 32$	$128, 64, 32, 16$	$128, 128, 64, 32, 16, 16, 8$
Receptive field	≈ 80 samples	≈ 180 samples	≈ 1680 samples
Parameters	260K	272K	664K

with shifts, they are much more robust, but it is hard to ensure that they truly outperform the 1D models for real data as we could only make qualitative observations here. To be able to check how our method deals with realistic cases, we need quantitative results for these real cases. By using the available ECG databases, it is possible to perform quantitative experiments on real data.

Additionally, ECG signals of people at rest are expected to have periodic properties, with a pattern that is repeated with little variations of shapes but possibly large fluctuations of duration. This kind of properties corresponds well to the properties we are wanting to deal with.

4.4.2 Models used for this experiment

For this experiment, we will only build model based on Wave-U-Net. We build a total of six models. Three models are 1D models. The first one is the same as the UNet model we implemented for the previous experiments, except for the last convolution, which is a 1×1 convolution. We denote it as UNet_7. The other two are the same by reducing the amount of downsampling blocks (and therefore also upsampling blocks) to four (resp. three) and we will denote it UNet_4 (resp. UNet_3). A summary of these architectures is given in Table 4.4.

The other three models are periodic architectures. We build three architectures, which are the periodic equivalent architectures of the 1D architectures. The only difference between a 1D architecture and its periodic equivalent is that the periodic architecture uses a receptive field of length 3 in four convolutional layers, instead of the receptive field of length 1 in the 1D architecture, resulting in a global receptive field of length 9 for the periodic dimension. For k in $\{3, 4, 7\}$, the periodic equivalent of UNet_k will be denoted as PUNet_k. A summary of these periodic architectures is given in Table 4.5. Let us notice that due to their long temporal receptive field, the UNet_7 and PUNet_7 might be able to access the periodic information by using only the temporal dimension.

Table 4.5: Details of the P1D architectures

	PUNet$_3$	**PUNet$_4$**	**PUNet$_7$**
Downsampling blocks	3	4	7
Kernel lengths (downsampling, temporal)	$7, 5, 3$	$9, 7, 5, 3$	$15, 13, 11, 9, 7, 5, 3$
Kernel lengths (downsampling, periodic)	$1, 3, 3$	$1, 3, 1, 3$	$1, 1, 1, 3, 1, 1, 3$
Kernel lengths (upsampling, periodic)	$1, 3, 3$	$1, 3, 1, 3$	$1, 1, 3, 1, 1, 3, 1$
Channels (downsampling)	$32, 64, 128$	$16, 32, 64, 128$	$8, 16, 32, 64, 128, 128, 128$
Channels (upsampling)	$128, 64, 32$	$128, 64, 32, 16$	$128, 128, 64, 32, 16, 16, 8$
Receptive field (temporal)	≈ 80 samples	≈ 180 samples	≈ 1680 samples
Receptive field (periodic)	9	9	9
Parameters	505K	503K	970K

4.4.3 Training the models

4.4.3.1 Training data

For these experiments, the clean signals are extracted from the MIT-BIH Arrhythmia database [12]. For training all patients, except 102 and 104 (which use another acquisition method) and $107, 112, 117, 119, 203, 207, 210, 212, 213$ and 217 (which will be used for the evaluations) are selected (only the MLII channel is preserved). For each patient, we first take the half-hour recordings, then we extract 31 signals of 20480 samples from this signal. The signals obtained with this method are used as such for the 1D models. For training the periodic models, we randomly select a point close to the beginning of each signal, and generate 10 rows of 2048 samples by shifting the starting point of one period (for example if the selected starting point is the 200th sample and each period has 300 samples, then the first row will be from sample 200 to 2247, the second row will be from sample 500 to 2547, the third row will be from sample 800 to 2847, ...). Here, the detection of the frequency is based on Algorithm 1. For applying the algorithm, we first compute the derivative of the signal by taking finite differences, then we remove the values below the third of the max value. We then apply the algorithm, with an expected frequency of 1Hz. This procedure generally improves the prediction of the frequency compared to applying the algorithm directly on the signal, which might be due to the baseline component that is still present in some signals despite these signals being theoretically clean. We apply our methodology on the clean signal, so that we can evaluate the interest of the periodic method in cases where a robust detection is available. However, a single frequency is used for an entire recording, and there might therefore be some areas where this frequency is not close to the actual frequency.

The training dataset results in a total of 36 patients with 31 signals (or matrices) each. For each patient and each clean signal, four noisy signals containing white noise are generated, with four different levels of SNR (0, 5, 10 and 15dB).

All signals are centered and normalized based on their maximal absolute value before extracting the subsignals.

4.4.3.2 Cross-validation

For this experiment, we perform cross-validation. A total of four folds are used. Each fold is defined by extracting one quarter of the patients for validation, while the rest of the dataset is kept for training. Consequently, each model will be trained four times to obtain more robust estimations of the results.

4.4.3.3 Algorithm

Each model is trained with the corresponding dataset (periodic or 1D), using the Adam optimizer to minimize the MAE. The learning rate is initialized at 0.001 and divided by 10 after 20 epochs, with a total of 30 epochs. The validation data are used to decide at which epoch the best model was obtained.

4.4.4 Evaluations

4.4.4.1 Evaluation data

The signals of the patients that were excluded from training data (patients 107, 112, 117, 119, 203, 207, 210, 212, 213 and 217) are used for the evaluations. For each patient, the entire signal (the 650000 extracted samples) is used. The matrices for periodic models are created with the same method as for training signals.

Each signal is tested with four different initial SNR which are the same as the ones used for training. Each result is obtained by averaging the results of the four folds.

The detection of the frequency is done as for the training.

4.4.4.2 Reference methods

The performed experiment is very close to the white noise experiment of [149]. It is then possible to use the method presented in this paper, which is a GAN implementation,

as a reference to compare with. Rather than performing their experiments again, which would be time consuming in terms of both implementation and computation, we directly compare to the results they claim. They also show some results obtained by two NLM based methods [60, 150], so we also compare to this method. It is largely outperformed by the GAN, but there might be some errors in the GAN results, as the SNR sometimes increases while the MSE increases too. Additionally, their model is much more complex than ours, with an adversarial training and around 50 million parameters for the generator only. Lastly, their results with realistic noise largely outperforms the quantitative results claimed in another recent paper [151]. Even though the experiments in these two papers are not exactly the same and the errors mentioned mostly appear to be an inversion of columns when reporting MSE, we do not expect our method to compete with theirs, but still show their results as they are the best claimed results so far in our knowledge.

Remark that in their work, the results are shown in terms of improvements of the SNR and not in terms of SNR. Here, we will use the obtained SNR as the evaluation metric to remain consistent with our other experiments (however, going from obtained SNR to SNR improvement is easy as we know the original SNR). Lastly, in their case, the results are only available for signals at 5 and 10dB. Therefore, comparison will be possible only for these two values.

Additionally to their methods, we also show the results of [152], which is a combination of variational mode decomposition, NLM and wavelet transforms and appears to be the best non-DL-based method according to [153].

4.4.4.3 Results

In Table 4.6, we show the results of all methods, including reference ones, for signals at 5 and 10dB. Here, we only show the average of all patients. These results show that our models outperform non-DL-based methods, even though they are largely outperformed by the method of [149]. Additionally, it appears that periodic models always outperform their 1D equivalent. Interestingly, it seems that the largest models are not the ones giving the best results, which tends to confirm the fact that using the content of the period at a largely different phase is not necessarily helpful.

Even though the differences remain small, the detailed results given in Appendix D show that periodic models almost always outperform their 1D equivalent, with a large difference obtained for some signals, such as patient 112, whereas 1D models only have a small advantage even when they outperform their periodic equivalent.

Table 4.6: Obtained output SNR (in dB) for the signals at 5 and 10dB

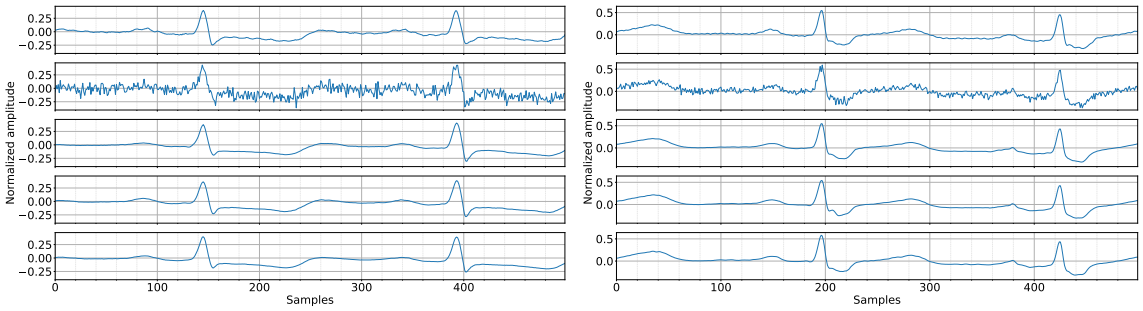
Method	5dB	10dB
NLM [60]	12.85	16.27
VMD-NLM [150]	14.77	18.27
ECG-GAN [149]	**19.52**	**24.64**
VMD-NLM-WT [152]	15.67	19.20
$UNet_7$	16.12	19.25
$PUNet_7$	16.22	19.36
$UNet_4$	16.05	19.28
$PUNet_4$	16.28	19.41
$UNet_3$	16.21	19.51
$PUNet_3$	**16.34**	**19.65**

Table 4.7: Obtained output SNR (in dB) for the signals at 0 and 15dB

Method	0dB	15dB
VMD-NLM-WT [152]	11.96	21.99
$UNet_7$	12.60	21.57
$PUNet_7$	12.74	21.76
$UNet_4$	12.51	22.08
$PUNet_4$	**12.92**	22.15
$UNet_3$	12.58	22.36
$PUNet_3$	12.80	**22.79**

In Table 4.7, we show the results for the two other values of initial SNR (0dB and 15dB). The observations remain similar as the ones already done with the previous table. From both tables, it also seems that the advantage of models with a small receptive field becomes higher when the original SNR increases. Even though we have no proved explanation for this phenomenon, we believe it is due to the baseline noise that is still considered as being part of the ground truth in these signals. As stated previously, using a small receptive field is helpful for ensuring focus on the relevant section of the period. However, with a high level of noise, a small receptive field hardly distinguishes baseline from noise, and possibly partially removes the baseline as well as white noise. When the level of noise is lower, it becomes easier to distinguish the baseline from white noise and the models with a small receptive field can get closer to their optimal behavior, whereas models with a large receptive field still struggle focusing on the relevant part of the signal.

To illustrate the results of this experiment, we also show two examples of denoising in Figure 4.21. For both results, it appears that it is visually hard to distinguish the quality of denoising by 1D and periodic models. Another interesting observation is that the denoised signals actually appear to be less noisy than the clean signal. This observation shows that even when real data with ground truth are available, it is actually not possible to obtain a ground truth that exactly matches the signal. The imperfect annotations might then result in bad interpretations of the evaluation. It is also possible to notice that the remaining noise in the clean signal does not look as white noise, and it is possible that the adversarial



(a) ECG record 112 at 5dB **(b)** ECG record 212 at 10dB

Figure 4.21: Examples of denoising at two different initial noise levels
From top to bottom : Clean signal, noisy signal, UNet_3, PUNet_3, PUNet_4

framework of ECG-GAN results in learning to distinguish this specific noise from white noise, which might explain the extremely good quantitative results of this method.

In our case, the figures still seem to indicate that PUNet_3 is slightly better at retrieving the magnitude of the peaks, and guarantee that training with proper data allows obtaining periodic models that are able to adapt to fluctuations of periodicity.

4.5 Conclusion

In this chapter, we proposed to exploit the periodicity of the signal by reshaping them into a periodic grid, where each line contains one period of the original signal. Is is then possible to exploit this new shape of the data, either by preprocessing the grid, or by replacing the 1D operations of the neural networks by 2D operations. Exploiting the periodicity with a preprocessing might seem advantageous in theoretical contexts as it allows similar performance as models with 2D operations but uses less parameters. However, when considering realistic data, where the signals are never exactly periodic and where it is not possible to detect the exact fundamental frequency, neural networks are able to adapt when trained properly. This ability to adapt to realistic cases was shown through the experiment with ECG, where real data could be used for the training, even though the gain in performance is then much smaller than in the exactly periodic case. Considering these shifts would not be possible (at least not easily) with the methods based on a preprocessing.

However, despite allowing improvement of the results on real data compared to the methods trained without shifts, methods trained with shifts also lose their interest in some cases, for example with large colored noises. Methods trained without shifts also seem able to retrieve additional information when the shifts remain small. This shows that all models can have their own advantage. When it is possible to know beforehand which case will be faced, then the more adequate model can be picked (for example, with ECG, we knew that the models

would be appropriate as it was trained on data having similar properties as the test data). But in some applications, it is not possible to know by advance which model will be the best, and it is then necessary to be able to build a model that will have satisfactory results in any case.

5 Ensemble methods for improving the robustness of the models

In the previous chapter, we proposed some methods to account for periodicity with Deep Learning models. The results for synthetic signals were good compared to conventional methods or Deep Learning methods without exploitation of the periodicity. However, generalization to real signals having some period shifts was a problem when no real data were available for training. Some models trained to be able to adapt to shifts were able to obtain similar results as the 1D models for real signals, and outperformed them on theoretical evaluations with white noise, but they were almost unable to show similar behavior in theoretical experiments with colored noise, even without any shifts between the lines of the generated grids. Additionally, the results obtained with the lowest frequency (1 Hz), which corresponded to cases where only one period was available, tend to show that all periodic models are outperformed by 1D models when the periodicity cannot be exploited anymore. Therefore, it seems that all methodologies have their own advantages depending on which kind of data they are applied on.

In this chapter, we propose to enhance the robustness of our method by trying to include some colored noises in the training datasets, but also by trying to combine the different models with an ensemble methodology. Here, the objective of the ensemble methodology is not necessarily to obtain the best results in most cases, but to be able to obtain satisfactory results (close to the best ones) in all or almost all cases, i.e. we focus on obtaining models that are able to adapt to different contexts (exactly periodic, almost periodic, not periodic or with no possibility of exploiting the periodicity) rather that one model that is specific to one context. That second option would be adequate for cases where we have *a priori* information about the data, which let us know which model to apply. The objective of the ensemble methods will be to remain efficient when such information for choosing the adequate model is not available.

Communications linked to this chapter

Journal papers (to be submitted)

[26] J. Rio et al. “Towards a unified framework for denoising periodic signals”. In: *To be defined* (To be submitted in 2022)

5.1 Background : Ensemble methods for denoising

Several works already focused on using ensemble methods for denoising. These methods are a combination of several different models in order to cumulate the advantages of each model.

The way of combining models can vary according to the methods used. The simplest method is to perform an average of all models. The average can use the same weights for all models, such as one of the methods proposed in [154], but it is generally more efficient to use a weighted average, such as in [102] for averaging two different models trained with different data, or in [155], where the same model is trained several times with different data grouped through clustering. Finding the adequate set of weights however requires trying the different values before concluding, which can be an extremely time-consuming approach if the amount of models is large. In a similar idea of averaging results of all models, it is also possible to average results obtained with the same clean signal polluted by several different realizations of the noise, denoised by the same model, such as in [156, 157] using an empirical mode decomposition (EMD) several times. The advantage of this method is that it does not require to design several models and it is therefore possible to use a large amount of denoising. However, it would be hard to have several realizations of the same noisy signal with independence of the generated noises (in [156], each realization is obtained by adding a white noise to the already noisy signal, so one part of the noise remains the same for all realizations), which means that the convergence will not be as fast as with independent realizations and might be affected by a bias. The same idea of combining several denoising results obtained by EMD applied on different realizations of the same signal has been proposed in [158, 159] with an iterative algorithm instead of a simple average.

Another approach is to use a succession of denoisers instead of using a single one. In such methods, the output of the first model is given as an input for the second model, the output of the second model is given to the third one, ... This approach is for example used in [160]. With this approach, it would actually be possible to use the same models several times to obtain a costless implementation. However, the amount of iterations is again not easy to define. Furthermore, in our case, we have already seen that some parts of the signals might be degraded by some of the models (for example, peaks are attenuated when using 1D models) and these parts of the signals might be degraded by each iteration. Furthermore, the receptive field is also artificially increased by this method, and the models then have to deal with much more zero-padding.

Other methods rely on a procedure called "multicontext stacking". For these methods, several different architectures are first used in parallel, then combined by a new trained

model instead of simply performing an average. The model used for combining the results is not necessarily a Deep Learning model. In [161], a Hidden Markov Model is preferred for postprocessing the outputs of the denoisers. In [154], one of the proposed methods relies on this methodology, with a postprocessing model that is itself an ensemble method.

In our work, we propose a method close to multicontext stacking, except that instead of using different architectures for all models, we propose to use different data to train each one of them. A similar approach is used in [162]. However, in their work, the combinination consists in selecting the most likely output from the different denoised signals. In our case, the denoised signals are fed to a new denoiser, which can learn, for each sample of the signal, how the combining should be done. As such, it can theoretically pick one of the model, perform a weighted average of the different denoised signals, or even perform an additional denoising.

5.2 Proposed approach

5.2.1 Improving the variety of the training data

In the previous chapter, we only trained with white noise. For 1D models, we assumed that using colored noise would not have a significant impact on the results as even very short-time correlations would be hard to distinguish from local phenomena. The impact of using colored noise would probably be limited to assuming smoothness of the clean signal in most parts. It could therefore act as a regularization, but might also degrade some parts of the signal. For the periodic models however, it is possible, by comparing the different periods, to know whether a local phenomenon is happening periodically with exactly the same evolution or not. In the case of a colored noise, the local spectral content should remain the same from one period to another in terms of magnitude, but might have differences in terms of phase. Therefore, colored noise should not result in a repetition of the same pattern, contrary to periodic phenomena.

To check how the noise can impact the trained models, we propose to perform trainings of the periodic architecture with four different datasets:

- A dataset where the matrices have no shifts and contain only white noise. By shifts, we mean the same kind of transformations as in Section 4.3.3.2, i.e., addition of a random constant to the phase of each line of the matrix. In this first dataset, all lines contain exactly the same phase.

- A dataset where the matrices have no shifts and contain a variety of different kinds of noise;
- A dataset where the matrices have shifts and contain only white noise;
- A dataset where the matrices have shifts and contain a variety of different kinds of noise.

Additionally, even though we do not expect it to have a significant impact in most cases, we also train the 1D models with two different datasets, one containing only white noise, and the other one containing both colored and white noises.

5.2.2 Ensemble method

To further enhance the results of the method, we propose an ensemble method to combine the strengths of several models and obtain a model that gives satisfactory results in almost all settings. The objective is not necessarily to beat each single model, but to improve the generalization to various cases.

Our approach is to first train several models with different kinds of data, to obtain models that are able to obtain good results in different settings. The output of each pretrained model will then be used as a channel for the input of the combining model.

In our experiments, all combined models will be based on the same original architecture, with its 1D and periodic versions. In our case, we will perform the ensemble method with combinations of Wave-U-Net based models only, due to the lower usage of memory with this model compared to WaveNet. The method could also be applied by designing a specific architecture for each kind of data.

Additionally to the outputs of the models, it is also possible to use the noisy signal as an additional channel. We expect it to help the combining model decide which of the denoised channels gives the more consistent output based on what was observed in the noisy signal.

As for the periodic models, the combining model is designed to be able to exploit the periodicity in his decision. In particular, we want the model to be able to check whether the different periods were similar enough to perform a periodic model trained without shifts, if they were similar but shifted and could be denoised by a model trained with shifts, or if it was not possible to exploit the periodicity and therefore more suited for a denoising with a 1D model. Therefore, the signals are shaped into a periodic matrix based on their

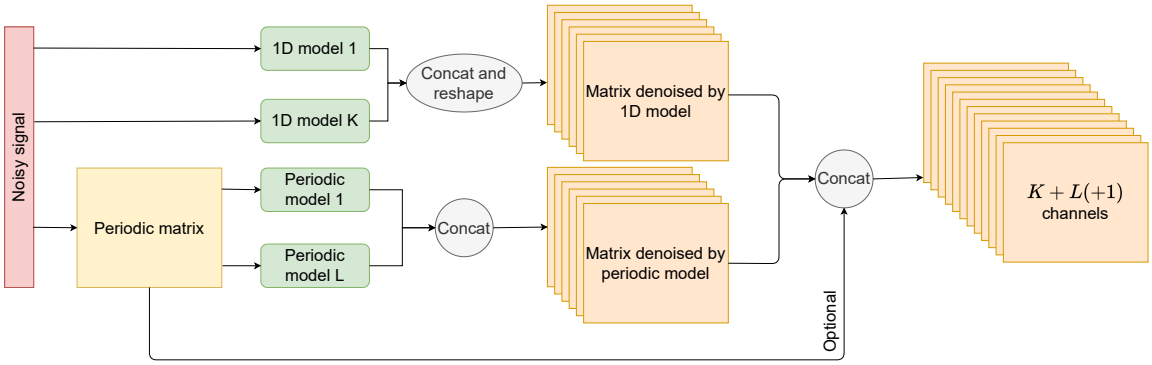


Figure 5.1: Preparation of the data for the ensemble method.

fundamental frequency. The detection of this fundamental frequency is assumed to have been done previously.

The framework for preparing the data is shown in Figure 5.1.

Concerning the architecture of the combining model, we use the same idea as in the WaveNet for speech denoising [9] to obtain a large receptive field in a few layers. However, we simply use convolutional layers, followed by a ReLU activation, instead of residual layers. We also use larger kernels to limit the amount of layers.

The architecture of the resulting model can be seen in Figure 5.2. Let us notice that this model indirectly increases the receptive field of the overall framework in both temporal and periodic dimensions. However, as the single models and the combining model are not jointly trained, the combining model should not be able to exploit this property and should rather focus on its own local receptive field.

The global framework of our ensemble method is shown in Figure 5.3.

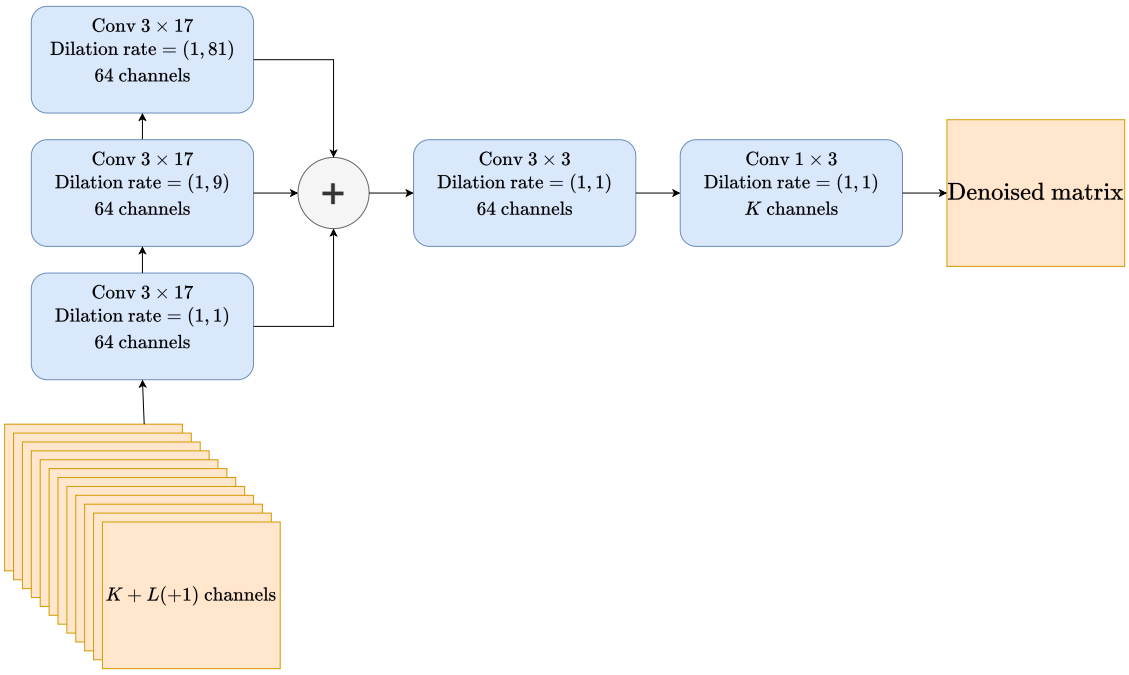


Figure 5.2: Architecture of the combining model.
The input channels are the ones prepared in Figure 5.1.

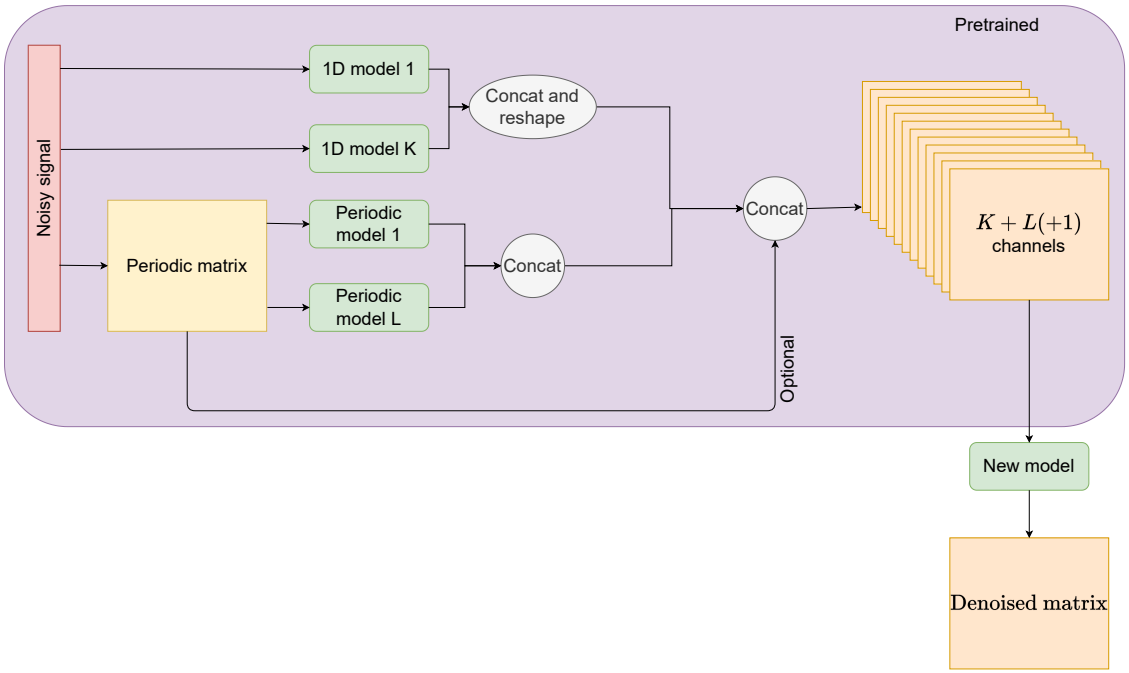


Figure 5.3: Illustration of the global framework for the ensemble method.
The data are prepared by pretrained models as show in Figure 5.1, then a new model as defined in Figure 5.2 is trained to exploit all denoised signals simultaneously.

5.3 Experiments

5.3.1 Settings for the single models

5.3.1.1 Architectures

In this chapter, we use WaveNet and Wave-U-Net based architectures.

We use the Medium architecture for the 1D versions of the WaveNet based architecture, and the PWNet architecture for the periodic models.

Concerning the Wave-U-Net models, we do not change the architecture compared to the previous chapter. The 1D models have the same architecture as the UNet model defined in Section 4.3.1, and the periodic models have the same architecture as the PUNet model defined in Section 4.3.3.

5.3.1.2 Training data

As mentioned in Section 5.2, we use six training datasets. First, let us mention the parameters that are shared for all datasets. For each dataset, the ground truth signal is generated as the weighted sum of a square wave and a sine wave, as in the previous chapters (Model (3.6)). The two components share the same frequency. For each signal, this frequency is randomly picked in set $[0.25, 0.5, 1, 2, 4, 8, 16, 24, 32]$ Hz, with a sampling frequency of 100000 Hz. For each signal, the noise is added to obtain a predefined initial SNR, randomly picked in set $\{-10, -5, 0, 5, 10, 20\}$ dB.

Now, concerning the specificities, the first dataset consists of 12000 1D signals of 20480 samples. The noisy signal are created by adding white noise to the clean signals. The 1D Wave-U-Net (resp. WaveNet) architecture trained with this dataset will be denoted as WUNet (resp. WNet).

The second dataset consists of 12000 1D signals of 20480 samples. The noise is created by applying a moving average on a white noise. The length of the moving average is randomly picked in set $\{1, 2, 3, 4, 5, 10, 20, 50, 100, 200\}$, resulting in various low frequency noises. The obtained colored noise $\mathbf{e}_c$ is added to a white noise $\mathbf{e}_w$ to create $\mathbf{e}$ the noise that is added to the signal, defined as:

$$\mathbf{e} = (1 - \xi)\mathbf{e}_c + \xi\mathbf{e}_w \tag{5.1}$$

Table 5.1: Summary of names given to the models depending on the method to train them.

Dataset	Shifts	Colors	Periodic	WaveNet model	Wave-U-Net model
1	X	X	X	WNet	WUNet
2	X	✓	X	WNet_c	WUNet_c
3	X	X	✓	PWNet	PWUNet
4	✓	X	✓	PWNet^*	PWUNet^*
5	X	✓	✓	PWNet_c	PWUNet_c
6	✓	✓	✓	PWNet_c^*	PWUNet_c^*

with ξ a random weight in set $\{0.1, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. Let us notice that this implementation still allows signals containing only white noise, when the length of the moving average is 1 or when the weight of the white noise is 1. The 1D Wave-U-Net (resp. WaveNet) architecture trained with this dataset will be denoted as WUNet_c (resp. WNet_c).

The third dataset consists of 12000 periodic matrices with 10 rows of 2048 samples. For this dataset, all lines share exactly the same phase. White noise is added to each signal. The periodic Wave-U-Net (resp. WaveNet) architecture trained with these datasets will be denoted as PWUNet (resp. PWNet).

The fourth dataset consists of 12000 periodic matrices with 10 lines of 2048 samples. For this dataset, the phase of each line contains the expected phase, shifted by a constant defined by a uniform law $\mathcal{U}([-0.05, 0.05])$. White noise is added to each signal. The periodic Wave-U-Net (resp. WaveNet) architecture trained with these datasets will be denoted as PWUNet^* (resp. PWNet^*).

The fifth dataset consists of 12000 periodic matrices with 10 lines of 2048 samples. For this dataset, all lines share exactly the same phase. The noise is created similarly as the one for the second dataset, i.e. by summing a white noise and a colored noise obtained by performing a moving average on a white noise, as presented in Model (5.1). The periodic Wave-U-Net (resp. WaveNet) architecture trained with these datasets will be denoted as PWUNet_c (resp. PWNet_c).

Lastly, the sixth dataset consists of 12000 periodic matrices with 10 lines of 2048 samples. The matrices contain shifts as for the fourth dataset, and the noise is created similarly as in the second and fifth datasets, according to Model (5.1). The periodic Wave-U-Net (resp. WaveNet) architecture trained with these datasets will be denoted as PWUNet_c^* (resp. PWNet_c^*).

A summary of the contents of the six datasets and the models trained with each of them is given in Table 5.1.

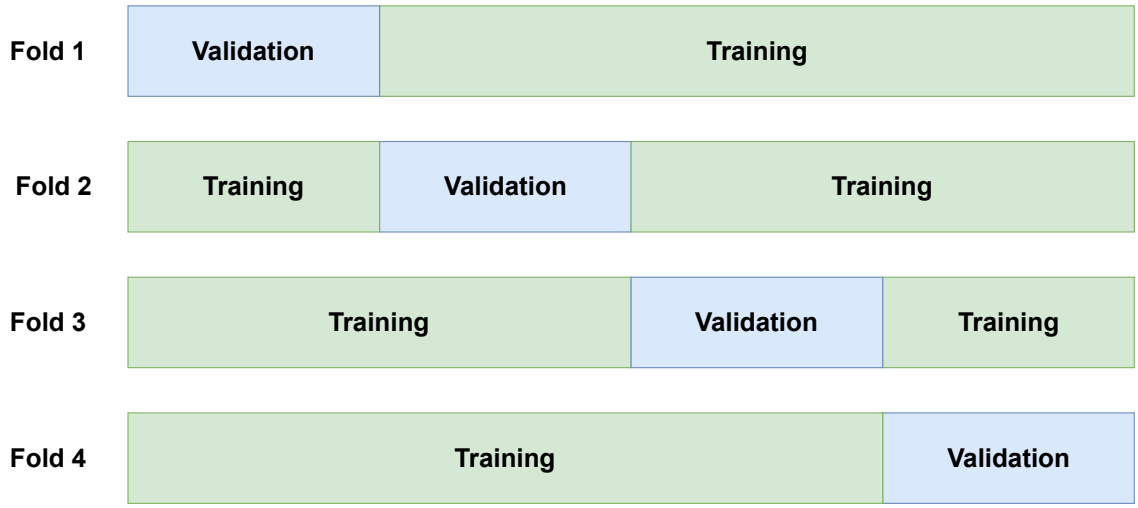


Figure 5.4: Illustration of the four different folds for one dataset

For all models, the training will be performed four times for four fold cross-validation. The dataset is subdivided into four datasets of 3000 signals or matrices. Each training will take one of the four sub-datasets of 3000 signals as the validation set for selecting the best model, and the other 9000 signals for performing the optimization. This procedure is illustrated in Figure 5.4.

5.3.1.3 Training algorithm

We use the MAE for training all architectures. As in Chapter 4, we assume that periodicity already provides an adaptable regularization. Therefore, we do not use any additional regularization in the loss term. It would be possible to use regularization for the 1D models, but it would then be hard to compare 1D and periodic architectures.

We train all architectures with Adam optimizer. We use batches of 8 matrices and perform validation at each epoch to chose the best model. The training uses a learning rate of 0.001, which is divided by 10 after 20 epochs.

5.3.2 Settings for the ensemble models

5.3.2.1 Chosen models for the input channels

Here, we define a total of four different models, which differ only by the models that provide the input data.

The first model, denoted as U-Ens, takes the matrices of all Wave-U-Net based models, as well as a noisy channel. Concerning the matrices provided by the 1D denoisers, they are reshaped into a matrix after processing by the 1D model, as shown in Figure 5.1.

The second model, denoted as U-Ens* also takes the matrices of all Wave-U-Net based models, but does not use the information of the noisy signal.

The third model, denoted as PU-Ens, considers all Wave-U-Net based periodic models, as well as a noisy channel.

The fourth model, denoted as PU-Ens*, considers all Wave-U-Net based periodic models, without the noisy channel.

5.3.2.2 Training data

Once the single models have been trained based on the procedure described in Section 5.3.1, it is necessary to train the ensemble model to exploit their outputs. Let us notice that only the weights of the combining model are trained. The weights of the single models are kept at the values obtained after their respective trainings.

As for the single models, we perform cross-validation. The datasets are obtained by extracting one sixth of each dataset used for training the single models. We pick the data used for optimization from the same data as the one used for optimizing the single models, i.e., we ensure no mixture of different folds, with an example for one fold given in Figure 5.5. We apply the same procedure for the validation data, resulting in a total of 9000 signals used for optimization and 3000 used for validation. This procedure is used even for the combining models that do not use all pretrained Wave-U-Net based models. For the datasets that were used for training the 1D models, we simply reshape them as a matrix without considering the frequency, i.e., we put the first 2048 samples in the first lines, the samples from 2049 to 4096 in the second line, ... The objectives is to obtain data that would result in an unusable periodicity, for which the 1D models should not be affected, whereas the periodic models will be largely affected.

A preprocessing is then performed on the obtained datasets, by processing each signal with all pretrained models that should be combined with the selected ensemble model, resulting in the multichannel input. This process is the same as the one detailed in Figure 5.1, applied to each signal of the datasets. Let us notice that we use the 1D models similarly as the periodic models, by defining $1 \times K$ convolutions. This process allows us to apply all models without having to differentiate the case of 1D models and the case of periodic models.

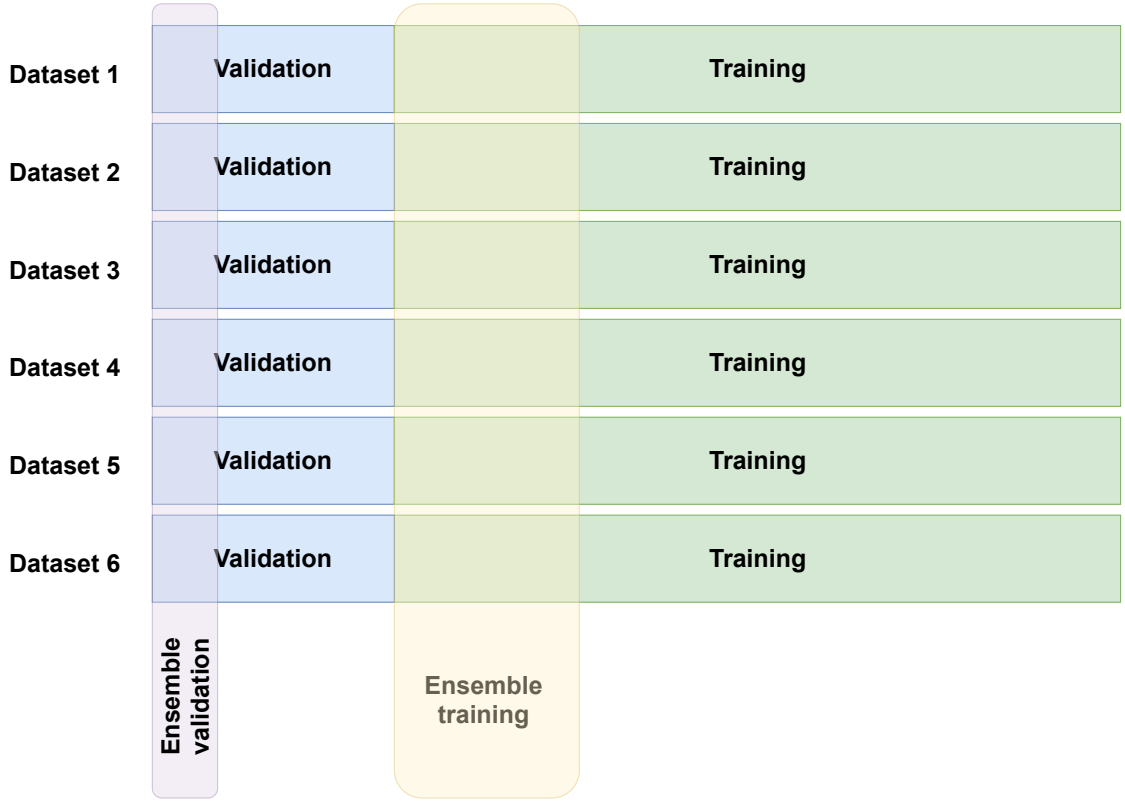


Figure 5.5: Illustration of the creation of one fold for the ensemble models

5.3.2.3 Training algorithm

As for the single models, we use MAE and Adam optimizer. We perform 15 epochs with an initial learning rate of 0.01 divided by 10 after 3 epochs and after 7 epochs. The best model is chosen based on validation scores.

5.3.3 Evaluations

5.3.3.1 Testing data

In this section, we will perform a large amount of evaluations, for testing the robustness of the different methods to various contexts. For each test, the ground truth is generated with the Andersson model, and a total of 60 signals of 100000 samples with a sampling frequency of 100000 Hz are used. As each model has been trained four times in the cross-validation process, each result will be the average of $60 \times 4 = 240$ values of the metrics.

Here, for each test dataset, the 60 signals have the same fundamental frequency f_0, and the same initial SNR. The method for generating the noise remains the same for all signals

within the dataset. Lastly, the frequency that is used for cutting the signals into several lines and creating the matrix is randomly selected in $[(1-\epsilon)f_0, (1+\epsilon)f_0]$ according to a uniform distribution. The value of ϵ remains the same for all signals in the dataset. In the different datasets, the value of ϵ used to define the uniform law for detecting the frequency of the signals can take 11 different values: 0, 0.01, 0.02, $\ldots$, 0.1. The case of $\epsilon = 0$ corresponds to the case where the exact fundamental frequency of the signal is exactly known. Let us notice that with high frequencies, even the largest error would result in small shifts, where the observed phenomena at a certain period is still within the receptive field in the next and previous periods. As an example, for signals at 60 Hz, even if the given frequency is 54 Hz, each row will contain 1851 samples instead of 1666 with the correct value. Therefore, when observing the j-th sample of the i-th row, the corresponding phenomenon of the next period will be in the $j-185$-th sample of the $i+1$-th row. As the receptive field covers samples from $j-512$ to $j+512$ for the WaveNet based models, and even a bit more for the Wave-U-Net based models, it will still be possible for the networks to retrieve these phenomena, even though it is possible that the amount of useful periods is reduced.

We use a total of 9 different fundamental frequencies: 1 Hz, 5 Hz, 10 Hz, 15 Hz, 20 Hz, 25 Hz, 35 Hz, 45 Hz and 60 Hz. As we are generating signals of 100000 samples at sampling frequency 100000 Hz, each signal corresponds to 1s of recording. The signals at 1 Hz will have only one period, and can therefore be seen as signals where the periodicity is not exploitable at all, which could be encountered even with assumptions of periodicity, for example if the mechanism is going from one state to another.

We also use a total of 9 different values for the initial SNR of the signals: -30 dB, -25 dB, -20 dB, -15 dB, -10 dB, -5 dB, 0 dB, 5 dB and 10 dB.

Lastly, we define three kinds of noise, i.e., three methods for generating the noise. The first kind of noise is simply white noise. The second and third kinds of noise are both generated with the same methodology as for generating colored noises in the training data, i.e., by summing a white noise and a low frequency noise obtained by applying a moving average on a white noise, with a weight to balance the two terms, as in Equation (5.1). For both kinds of noise, the noise of each signal uses a moving average of length randomly picked in set $\{1, 4, 10, 20, 50, 100, 200, 250\}$. Let us notice that longer lengths, implying lower frequencies, are more common than in the training dataset, even though it is still possible to obtain a white noise if the value 1 is chosen. For the second kind of noise, the weight ξ is randomly selected in set $\{0.05, 0.25, 0.4, 0.5, 0.65, 0.8, 0.9, 1\}$. We will mention mixture of noise when mentioning this kind of noise. For the third kind of noise, the weight ξ is always equal to 0.1, implying that the colored part of the noise is always preponderant, and the length of the moving average is the only way to obtain a white noise. We will mention colored noise for denoting this kind of noise.

Table 5.2: Summary of all tested parameters

Parameter	Possible values	Amount of values
Frequency (Hz)	$1, 5, 10, 15, 20, 25, 35, 45, 60$	9
Initial SNR (dB)	$-30, -25, -20, -15, -10, -5, 0, 5, 10$	9
Shift (value of 100ϵ)	$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$	11
Noise	White, Mixture, Colored	3
Total		$9 \times 9 \times 11 \times 3 = 2673$

As we are performing an evaluation for each combination of fundamental frequency, initial SNR, ϵ value and kind of noise, there is a total of 2673 evaluations. As it is not reasonable to deeply investigate all of these evaluations, we will start with the cases that we assume to be the most common ones to first eliminate some models, then we will focus on the selected models for further investigation. We will focus on SNR improvement in this case, as it is easier to compare cases with different initial SNR, even though the improvement is not expected to be linear.

5.3.3.2 Quantitative results

Joint observation of all evaluations

Here, we make some global observations about all evaluations performed. The objective is to get an idea of the performance of all models and to pick some models that provide a good general behavior. The selected models will be further studied in the following sections. Other models will be eliminated as they do not correspond to our objective of obtaining a model that is robust to various cases.

In Figure 5.6, we show the distribution of SNR improvements obtained by all models in all evaluations. The first observation that can be done is that there is no major difference among the models regarding the third quartile. Oppositely, the first quartile clearly shows that some models are not able to generalize well to various cases. For WaveNet based architectures, only WNet_c and PWNet_c^* obtain a good generalization. Despite obtaining quite good results in terms of median and 5% percentile, other architectures clearly have a high amount of cases where the improvement is not sufficient. For Wave-U-Net based architectures, similar observations can be done. The WUNet_c and PWUNet_c^* architectures obtain satisfactory results, and the PWUNet^* model, despite degrading the first quartile, remains quite satisfactory compared to its WaveNet equivalent. This better behavior of Wave-U-Net based architectures compared to WaveNet based architectures is not a general observation that can be done for WaveNet and Wave-U-Net as it depends also on the task.

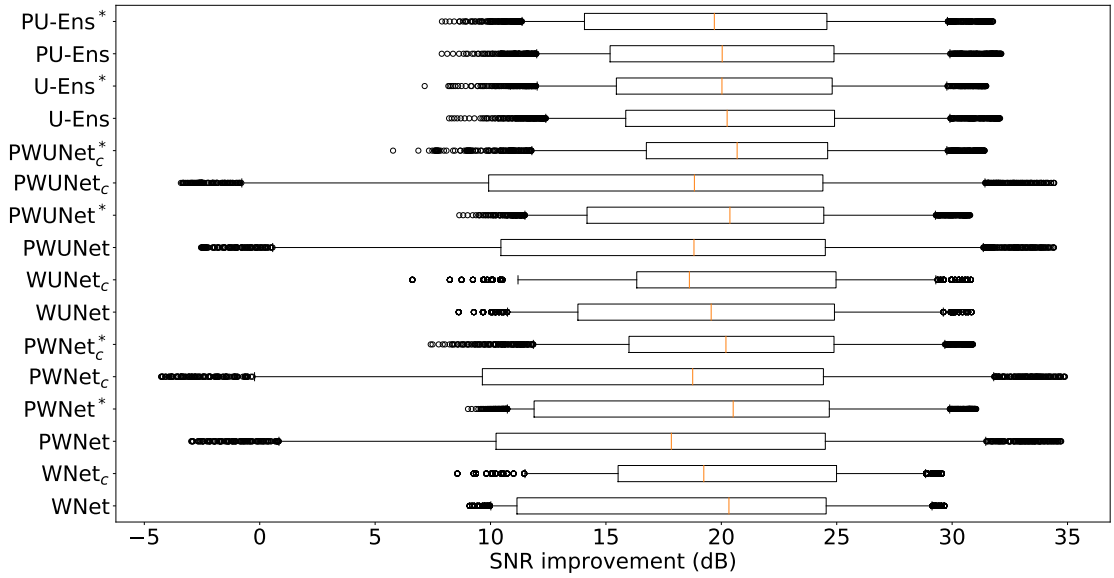


Figure 5.6: Joint observation of all evaluations
The orange segment represents the median, the first extremity of the box is the first quartile and the last extremity is the third quartile, the extremity of the segments are the 5% and the 95% percentiles.

Concerning the impact of colored noise during training, it does not seem to be relevant when applied to exactly periodic architectures. Oppositely, it seems to have a significant impact on the first quartile when used for periodic architectures trained with shifts, showing that improving the variety of noise helps improving the robustness of the models. Interestingly, the same observation concerning the first quartile and 5% percentile can be done for 1D models. However, in this case, it is hard to be sure that it is due to a better generalization to colored noises or to an effect similar to the one obtained through regularization in Chapter 3. The second case would imply that the obtained models would also result in a degradation of short-time periodic phenomenons.

Concerning ensemble models, their results seem quite satisfactory, but it is hard to say whether they outperform other models or not. Comparing them with PWUNet$_c^*$ especially, they seem to be a bit better in terms of 5% percentile and third quartile, but the differences remain too small to be relevant, and other indicators, such as the median, are in favor of PWUNet$_c^*$. The ensemble models that do not use a noisy channel seem to have lower results than their equivalent with noisy channels. Methods using only periodic models generally outperform methods using all models in terms of 95% percentile, but also seem to have a larger variance, resulting in a higher amount of low results. As a consequence, the model PU-Ens*, which combines the absence of 1D models and the absence of noisy channel, seems to have quite low results and clearly does not seem to be interesting in terms of robustness.

	First	Top 3	Top 5	Last 5	Last 3
PU-Ens*	0.00	4.60	11.63	25.22	1.91
PU-Ens	6.92	16.31	31.65	1.80	0.00
U-Ens*	0.97	8.90	17.28	27.80	10.66
U-Ens	7.63	19.19	31.58	5.69	0.86
PWUNet$^*_c$	5.69	18.33	45.45	7.48	0.04
PWUNet$_c$	15.56	26.97	41.11	48.11	44.56
PWUNet*	1.27	8.31	22.78	22.82	1.50
PWUNet	4.64	35.28	46.95	43.88	8.12
WUNet$_c$	7.22	13.73	21.51	33.48	18.86
WUNet	5.39	20.20	27.65	51.10	41.53
PWNet$^*_c$	3.22	14.18	43.32	16.54	0.41
PWNet$_c$	14.85	38.76	43.66	49.35	43.47
PWNet*	6.55	18.48	33.03	32.44	25.10
PWNet	6.51	18.93	27.72	47.62	40.10
WNet$_c$	1.23	15.38	24.65	33.41	21.25
WNet	12.35	22.45	30.00	53.27	41.64

Table 5.3: Percentage of presence at certain ranks (in %)

Based on these observations, we will continue the study with WNet$_c$, PWNetc^*, WUNet$_c$, PWUNet*, PWUNet$^*_c$, U-Ens, U-Ens* and PU-Ens.

As a complement to the previous comments, we also show in Table 5.3 the ranks obtained by the networks. Here, the ranks are computed for each evaluation, and the displayed values are the proportion of evaluations in which the model reaches a given position. Let us notice that the results might be confusing. To clarify this, let us consider a theoretical context with a model A, a model B and fourteen other models. In 50% of the evaluations, model A obtains a 10.2 dB improvement, model B obtains a 10 dB improvement and all other models obtain a 10.1 dB improvement. In the remaining evaluations, model B obtains a 10 dB improvement, model A obtains a 0.1 dB improvement, and all other models obtain no improvement. In this theoretical context, both model A and B would obtain a 50% proportion of first position, but model A would be second in other case and model B would be last. By not considering this possibility, the conclusion would be that model A is much more robust than model B, which is proved false by looking at the actual improvements. This is the reason why we focused in Figure 5.6 to select the networks we will further study rather than this table. It is however possible to expect the models PWUNet$^*_c$, U-Ens and PU-Ens to have a better general behavior than other selected models in the next observations.

Robustness to various shifts

Now that we have picked some models that seem to give satisfactory results in a sufficiently high amount of cases, we can further investigate these models. As detailed previously, one

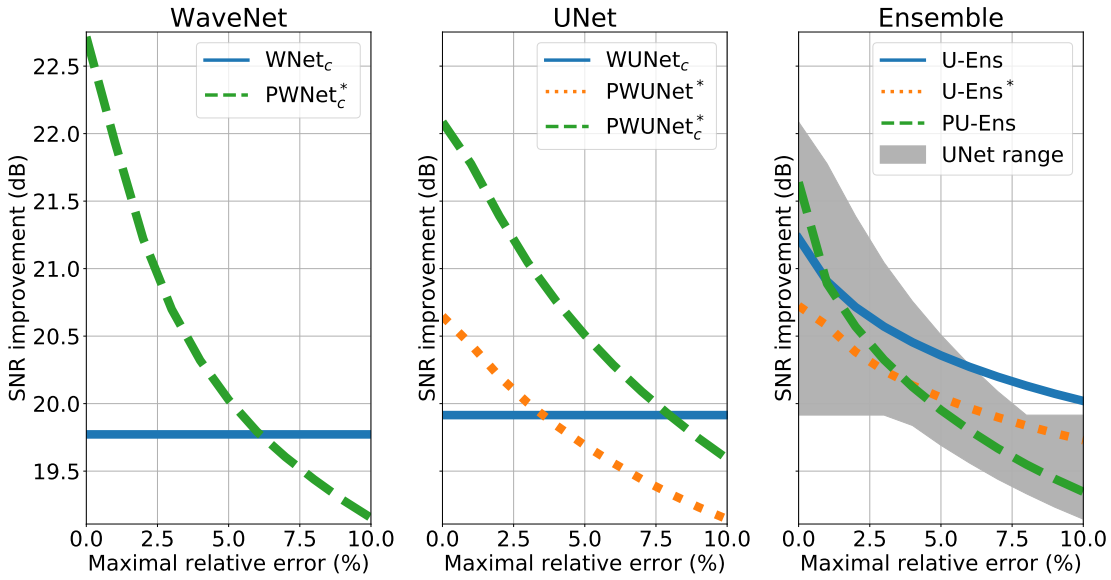


Figure 5.7: Evolution of the results regarding the shifts
Each value is the average of results obtained with all evaluations where the shift was set to the specified value, without distinction of the frequency, initial SNR or kind of noise. The UNet range is obtained with the three selected Wave-U-Net based models, without the ones that were removed after general observations.

of the main constraints for applying our periodic models to real data is the variability of the fundamental frequency and the impossibility to exactly detect this frequency, which results in the presence of shifts in the created matrix. By giving a false value of the fundamental frequency, the matrices will contain similar shifts as the ones that would be obtained with real exactly periodic data, resulting in constant shifts. For example, if each period contains 5000 samples but the matrix is created by assuming that each period contains 4900 samples, then each row will be shifted of 100 samples compared to the previous row. In real data, the fluctuations of the periodicity would result in different values for a single signal. For example, the first row could be shifted of 112 samples compared to the second row, and the second row could be shifted of 89 samples compared to the third row. However, the models have been trained either without any shifts, or with shifts generated randomly for each row, and should therefore not be able to exploit the stability of the shifts in our test data.

In Figure 5.7, we show the evolution of the results depending on the shifts. For this figure, there is no distinction of the frequencies used. As for the experiments in Chapter 4, 1D models do not have to split the signals into a matrix and are therefore not affected by shifts. This makes them suitable as a baseline for comparison with models considering the periodicity. It appears that with low shifts, WaveNet and Wave-U-Net based models have good results. However, the model PWUNet* gets much lower results than models trained with colors. Based on the observations made in Chapter 4, it is likely that this model gets good results with white noise, but fails to outperform 1D models with colored noise, explaining how models trained with colored noise can outperform this model. When comparing PWNet$_c^*$

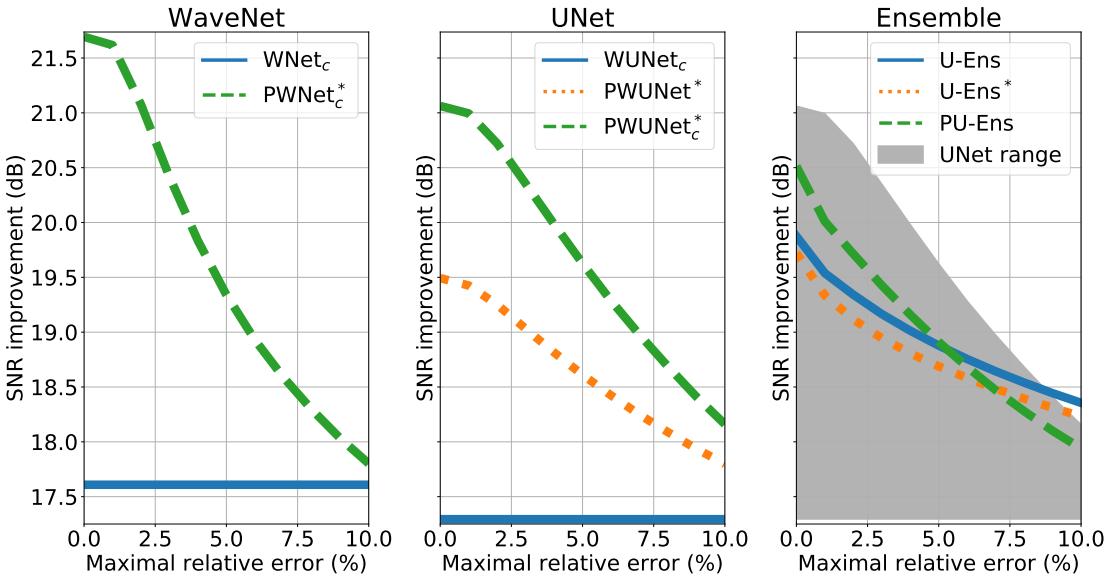


Figure 5.8: Evolution of the results regarding shifts, for signals at 35, 45 and 60 Hz

and PWUNet$_c^*$, it appears that WaveNet based models get high improvements without shifts, but are largely affected by the presence of shifts, even small ones. Ensemble methods get lower results but seem to be less affected by shifts, even though PU-Ens, which gets the best results without shifts, is still largely affected. The model U-Ens, despite being initially much lower than the model PWUNet$_c^*$, ends with better results for larger shifts, and it seems that even larger shifts would accentuate this observation. For the observed values, it even remains better than 1D models for all shifts.

When focusing on higher frequencies (35, 45 and 60 Hz, Figure 5.8), it appears that the robustness of the U-Ens model is less important, even though the slope remains much lower than for other models. Here, the advantage of ensemble models compared to PWUNet* is only slightly observable for the larger shifts. However, they clearly outperform the 1D models. This observation can be linked to the observation of Section 5.3.3.1, about the receptive field being long enough to still contain the phenomenon in several period.

When observing other frequencies (5, 10, 15, 20 and 25 Hz, Figure 5.9), the robustness is now improved compared to single models, and the model U-Ens and U-Ens* now outperform all other models with the largest shifts, even 1D models. This seems to confirm the improved ability of these models to remain efficient when the temporal receptive field is too small to retrieve the observed part of the signal in other periods.

To confirm the ability of ensemble models to better generalize to data for which the periodicity cannot be exploited anymore, we show the distribution of the results obtained on signals at 1 Hz in Figure 5.10. Let us notice that a frequency is still given for creating the matrix in the case of periodic models. When this frequency contains an error, it should have

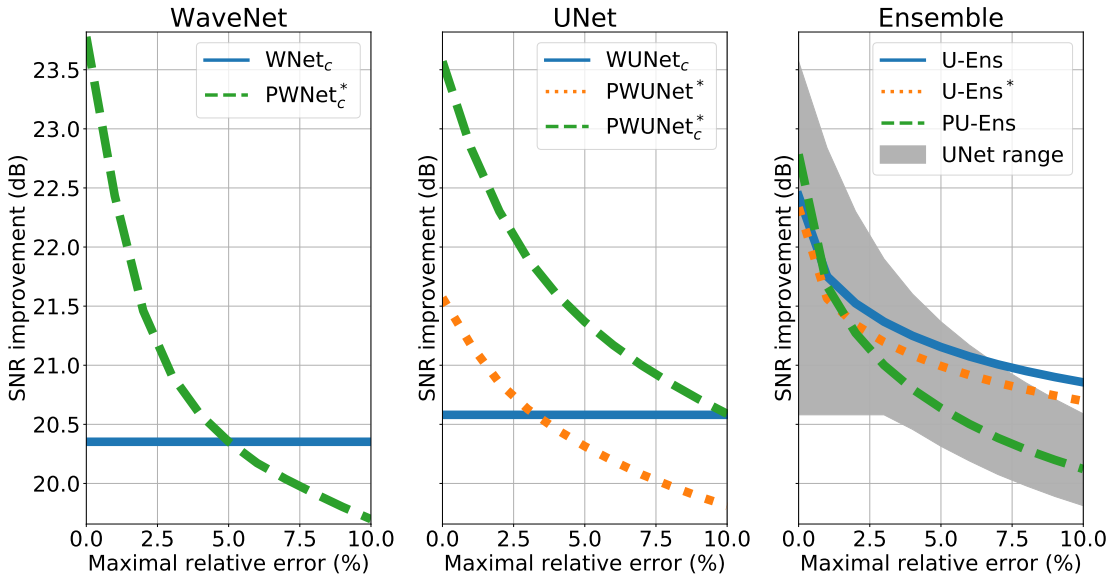


Figure 5.9: Evolution of the results regarding shifts, for signals at 5, 10, 15, 20 and 25 Hz

no impact when the given frequency is lower than the correct frequency as the entire signal will still be contained in the first period, but with a too high given frequency, the creation of the matrix would be done by thinking that each period contains less than 1 s of signal. As a consequence, a second row will be created and will contain the end of the unfinished period, which might confuse the networks.

The results on these signals clearly emphasize the advantage of ensemble models compared to single periodic models. Despite getting lower results than 1D models, ensemble models get a relatively high median and 5% percentile, at least compared to PWUNet$_c^*$. In particular, the U-Ens model is now outperformed only by the 1D models for all displayed quantiles. Surprisingly, WNet$_c^*$ now seems to outperform WUNet$_c^*$.

When considering all observations made in this section, it appears that ensemble models are able to get an improved robustness to cases with unusable periodicity, which might be useful for signals with periodic properties but large fluctuations of the periodicity. However, the results obtained by single models, especially PWUNet$_c^*$, tend to show that for signals where the periodicity is stable, a P1D model should be preferred even if it has been trained with less diverse data. The advantages of ensemble models for such signals are indeed not significant enough for justifying the addition of complexity, memory usage and time computation.

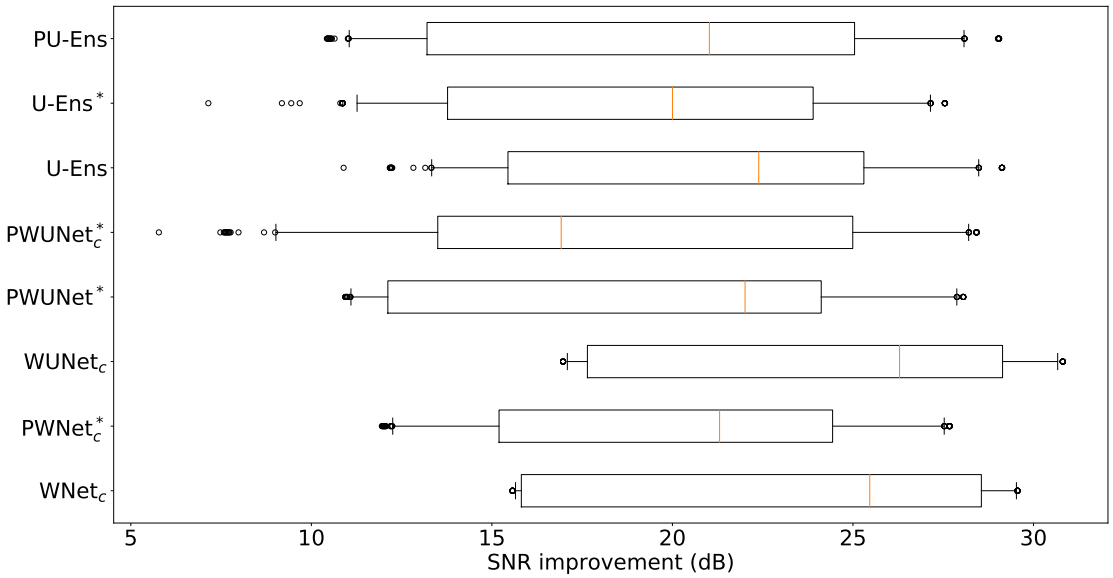


Figure 5.10: Repartition of the results for signals at 1 Hz

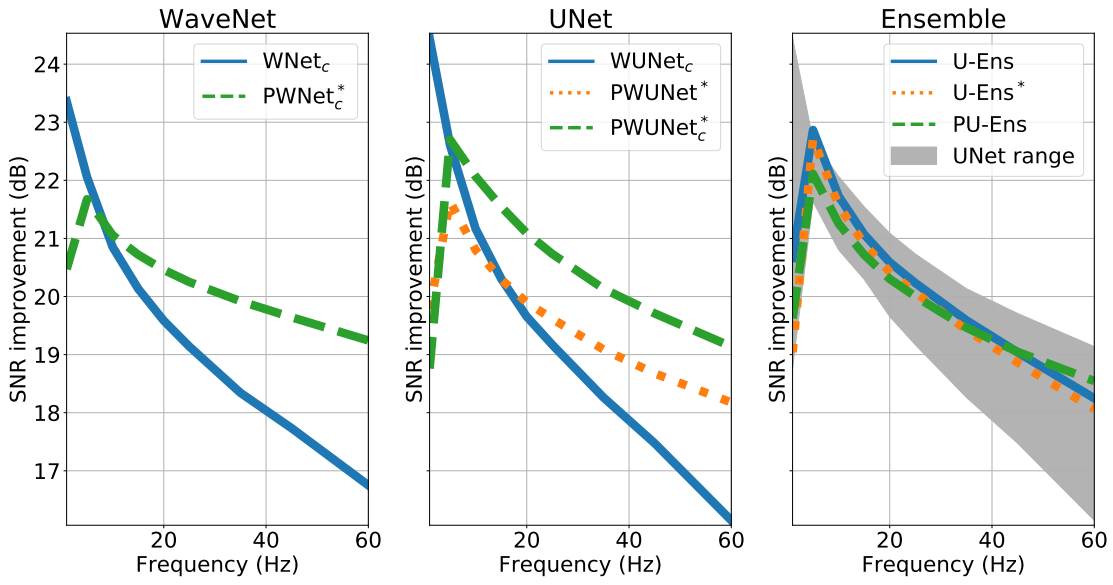


Figure 5.11: Evolution of the results regarding frequency

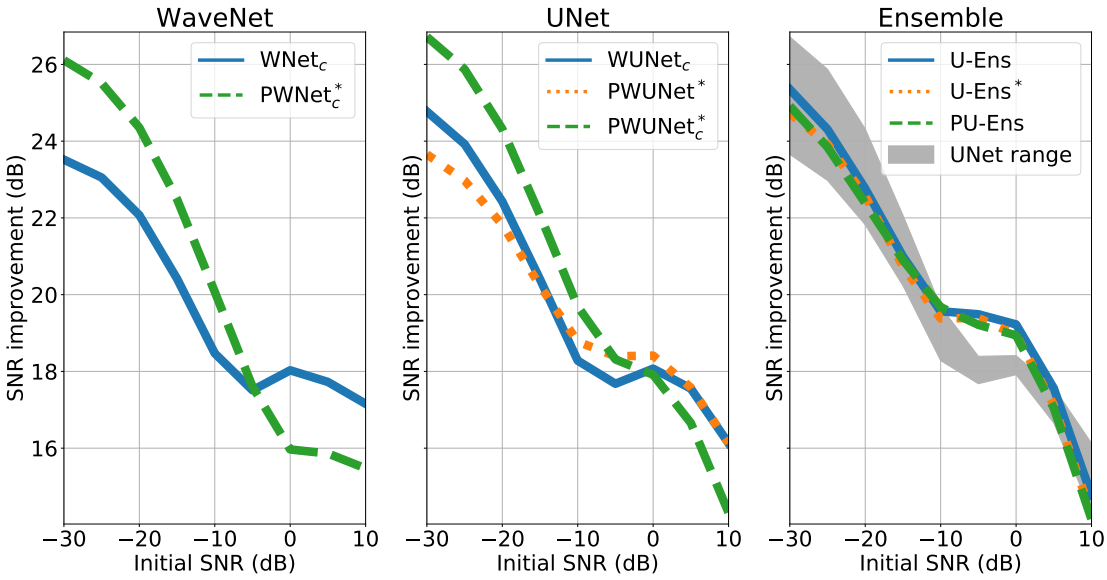


Figure 5.12: Evolution of the results regarding initial SNR

Generalization to various frequencies

We now study the impact of the frequency on the results of the different models. We show the evolution of the results depending on the frequency in Figure 5.11. Here, as it was the case in Chapter 4, the 1D models get the best results for signals at 1HZ, but as it was already shown in Figure 5.10, ensemble models manage to reduce the weakness of Wave-U-Net based models for this kind of data with unusable periodicity. These results also clearly show that ensemble models are not able to compare with periodic models when reaching higher frequencies, even though they remain much better than 1D models. This observation confirms that our implementation of the ensemble models is not suitable for data for which it is possible to assume stability of the periodicity.

Generalization to various levels of noise

Concerning the impact of the noise level on the results, we show the evolution of the mean score obtained depending on the initial SNR in Figure 5.12. Here, several interesting remarks can be done. First of all, we can observe that the 1D models tend to obtain similar, or even better results than periodic models when reaching high initial SNR. We know due to experiments in Chapter 4 that this is not the case for exactly periodic signals with a high frequency and white noise, and that models trained with shifts are still able to outperform 1D models with small shifts. However, the previous experiments have shown that large shifts might result in better results for the 1D models. Another interesting observation is that ensemble models seem to outperform all other models for an initial SNR between -10 and

	White	Mixture	Colored
WNet$_c$	23.97	21.48	13.87
PWNet$_c^*$	23.28	21.58	16.28
WUNet$_c$	23.63	21.47	14.64
PWUNet*	**24.44**	21.91	12.97
PWUNet$_c^*$	23.42	21.82	**16.71**
U-Ens	24.31	**22.21**	14.82
U-Ens*	24.09	21.95	14.32
PU-Ens	24.21	22.00	14.15

Table 5.4: SNR improvement (in dB) of the models depending on the method used for generating the noise

	White	Mixture	Colored
WNet$_c$	4.32	3.67	2.03
PWNet$_c^*$	5.28	4.76	3.83
WUNet$_c$	4.87	4.33	2.89
PWUNet*	3.91	3.28	1.35
PWUNet$_c^*$	5.12	4.75	3.84
U-Ens	4.67	4.01	1.96
U-Ens*	4.72	4.07	1.95
PU-Ens	4.86	4.09	1.76

Table 5.5: Standard deviations of SNR (in dB) of the evaluations depending on the method used for generating the noise

0 dB. This observation is promising for realistic data as such SNR is not rare, at least locally in real signals. However, they get a large drop for higher SNR, and are again outperformed by the single models. However, the difference remains quite small in this case and the single models also obtain a similar drop.

Generalization to various colors of noise

In Table 5.4, we show how the type of noise changes the results of the models. These results show that all models obtain very similar results for white noise or for the mixture of noise. For the colored noises only, PWUNet$_c^*$ and PWNet$_c^*$ clearly outperform other models. This tends to confirm that improving the variety of the training data was more relevant than building ensemble models in most cases. However, the ensemble models remain quite good in general, which is promising considering that there are still multiple possibilities for improvement with this method. Additionally, Table 5.5 shows that the models trained with colored noises and shifts also have a larger variance, whereas it remains low for ensemble models. This observation is not surprising, as the general observations in Figure 5.6 already showed that the results were contained in a lowest range, at least when observing the 5% and 95% percentiles.

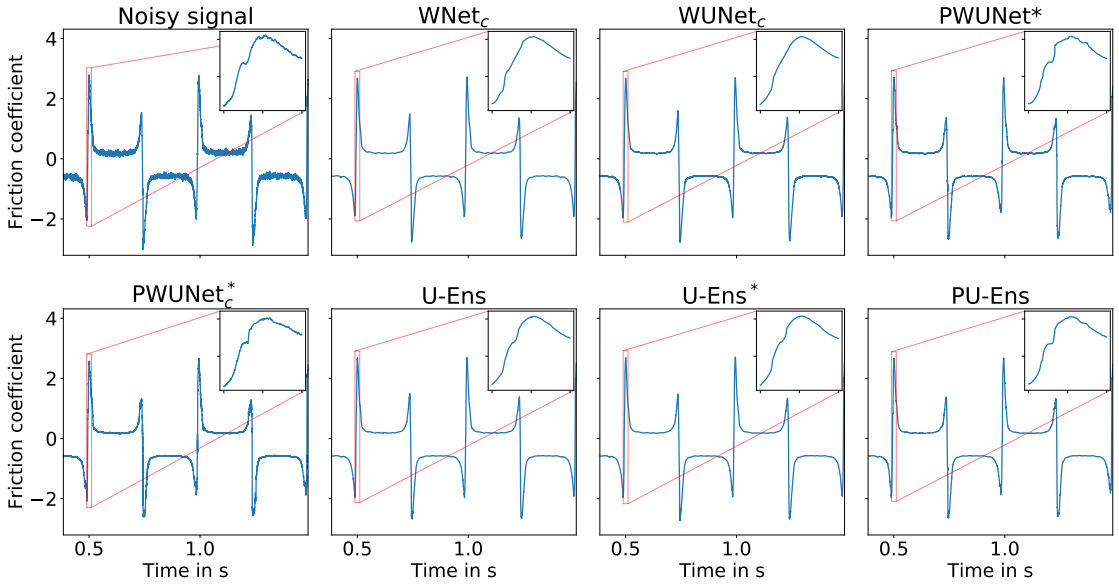


Figure 5.13: Example of denoising for a real friction signal at 200 rpm

5.3.3.3 Results on real signals

For the selected models, we also want to know how they behave with real signals.

For that, we show in Figure 5.13 an example of denoising for a signal with a rotation speed of 200 rpm, containing a low level of noise. Here, the results of the Ensemble models are clearly close to the ones of other architectures.

When focusing on the flat area of this signal (Figure 5.14), it even seems that the ensemble models clearly outperform the PWUNet$_c^*$, which seems to be unable to efficiently remove the white noise compared to other models.

In Figure 5.15, we show another signal, obtained with a rotation speed of 750 rpm, resulting in a higher level of noise. The ensemble method, as well as PWUNet$_c^*$, seem to be a good compromise between preservation of the peaks and smoothness outside of these peaks. Regarding the phenomena that appeared in the smooth part, it is still hard to say whether they are part of a periodic component or not. PWUNet$_c^*$ seems to remove most of them, but still preserves some of them. The ensemble models reduce their magnitude compared to PWUNet*, but preserve most of them. This is the case for example with the phenomenon emphasized in the zoom, which seems to appear periodically. Models trained with colored noises remove this phenomenon, showing that training with colored noises might give similar behavior as using a large regularization.

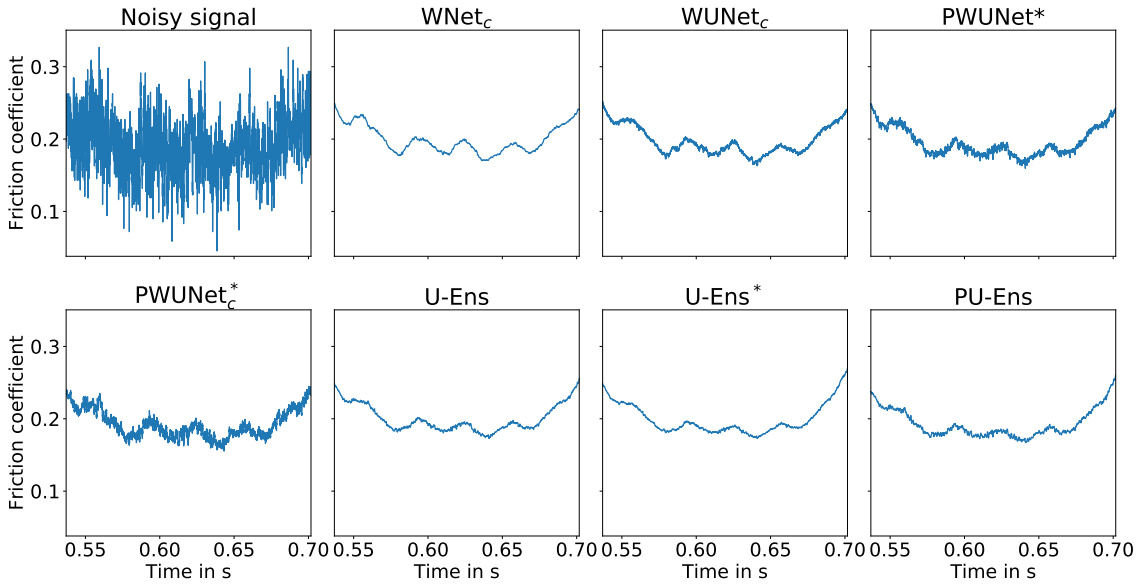


Figure 5.14: Flat area of a friction signal at 200rpm

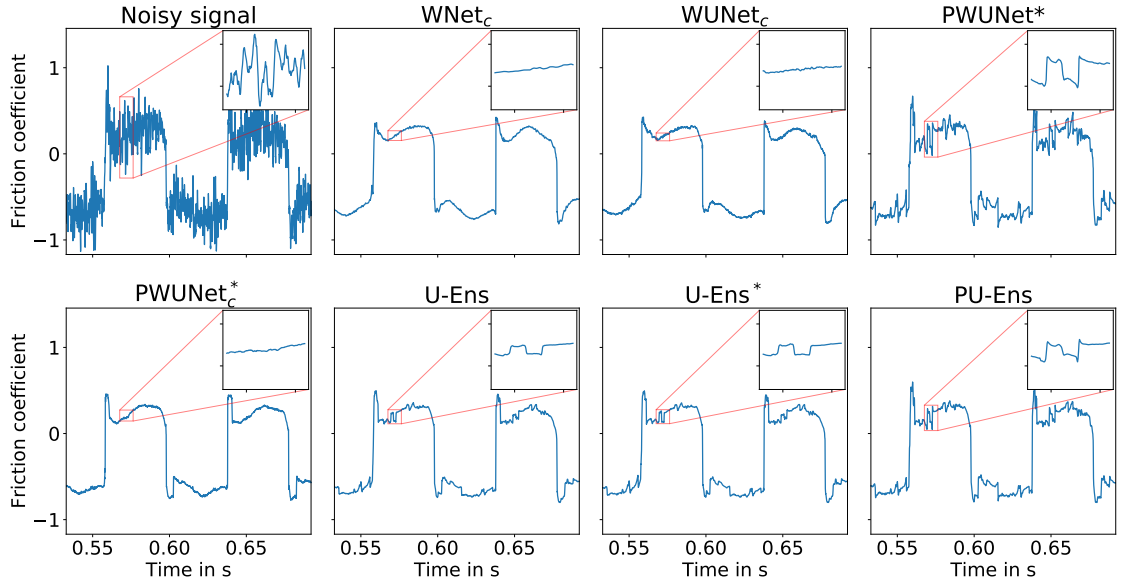


Figure 5.15: Example of denoising for a real friction signal at 750 rpm

5.4 Conclusion

In this chapter, we proposed an extension of Chapter 4, by both performing a deeper analysis of the strengths and weaknesses of the methods proposed, and proposing extensions of these methods to improve their robustness in cases where no labeled data are available. The improved robustness was obtained either through a larger variety of the training data or through a combination of different models in an ensemble framework.

Adding colored noises in the training set resulted in large improvements of the quantitative results in most cases. However, results on real signals tend to show that despite being better with colored noises and remaining quite good with white noise, the models trained with colored noises were not necessarily better than the ones obtained by other models.

Concerning ensemble models, the results were contrasted. They generally did not manage to outperform models trained with colored noises and shifts, but obtained smaller decreases when the periodicity became unusable. They also obtained satisfactory qualitative results on real data, with a good compromise between smoothing the signal and preserving local phenomena. The obtained improvements were not sufficient to consider that the current ensemble methods should be preferred to other models. However, considering that this ensemble method is still a first step, without a large study of the models to combine, of the architecture of the combining model and of the creation of the training data, these first results are already promising. In particular, they already provided great results for some realistic initial SNR values, and were rarely far from the best results, even when focusing on other parameters than the initial SNR. Therefore, despite not being suboptimal in their current exploitation, ensemble methods should be studied in further details to optimize their combination.

6 Conclusion and future work

6.1 General conclusion

In this thesis, we studied the denoising of signals having periodic properties and containing discontinuities or fast variations. After presenting the necessary background in Chapter 1, our first method, presented in Chapter 2, consisted in fitting a parametric model on the measurements. This method obtained satisfactory results in some parts of the signals, but also failed to adapt well to some phenomena. In particular, they were unable to differentiate acceleration and deceleration phases of the friction signals, and assumed no changes of the periodicity. Therefore, building models that are able to better adapt to the local properties of the signals was necessary. In Chapter 3, we proposed a first method based on a fully convolutional deep neural network. We compensate for the lack of annotated data by the creation of synthetic datasets for training and evaluating the models. We also proposed a new loss term, incorporating TV-based regularization, that proved useful for improving the smoothness of the denoised signals. However, this also implied flattening the discontinuities and possibly losing some local but relevant phenomena. Additionally, this method did not exploit the periodicity of the signals. In Chapter 4, we proposed to reshape the data so that data from different periods but sharing the same phase are put in parallel. It was then possible to reshape the networks as well, or to preprocess the data by performing a moving average. While both methods provided satisfactory results in theoretical cases, reshaping the networks was the only method that also provided good generalization to real data. The results of this chapter were also indicating that at least some of the phenomena removed by using regularization were actually part of a periodic pattern. Experiments on ECG also proved that when trained with the proper data, i.e., with similar variations of the periodicity as in the test signals, our periodic models were able to generalize to real data and already outperformed 1D models for some signals. To improve the robustness of this method in cases where annotated data are not available, we proposed further investigation of the performances of the models in Chapter 5. We also studied how incorporating colored noises in the training set and combining several models by feeding their outputs to a postprocessing model could help improve the robustness of the methods. Incorporating colored noises allowed obtaining much better results in general. The ensemble methods also obtained satisfactory results, especially in terms of robustness to cases with hardly exploitable periodicity, and showed satisfactory generalization to real data. Even though we cannot claim that this ensemble method is already clearly outperforming

other methods, it already manages to get satisfactory results in most cases, which was the main purpose of these methods. The results are therefore promising enough to encourage further investigation of ensemble methods.

6.2 Ideas for the future work

In this work, we proposed to compensate for the lack of annotated data by generating synthetic data, obtained with a simple model. An interesting study would be to check how creating more realistic data, for example by using generative adversarial networks (GAN), could help improving the results. Using unsupervised methods could also be useful for training with real data. Several unsupervised methods have already been proposed for training deep signal or image denoisers [138, 139, 163]. It might be that we still do not have enough real signals to train the networks completely. However, it would still be possible to train with synthetic data first, before fine-tuning the model with real data.

The regularization we proposed in Chapter 3 helped us getting smoother results but also resulted in degradation of the peaks. An idea to overcome this problem could be to propose an adaptable balance between the reconstruction terms and the regularization. In other words, we would give an extremely small or null weight to the TV-based term in discontinuities, and a high one in flat areas. However, our experiments showed that fixing the weight as a constant value is already not trivial. Finding the correct adaptable weight would be even harder. To obtain some kind of adaptable regularization, we made some experiments with a conditional generative adversarial network, such as the one presented in [125]. We hoped that the discriminator would end up giving a solution close to having an adaptable regularization. The first results obtained with this method were not good, and we decided to focus on other ideas instead, but this methodology could still be further studied.

In our work, we assumed that all phenomenons, including the ones occurring during a very short time range, could have an interest for the further analysis. However, if only the global shape is wanted, for example to estimate the Stribeck curve of the materials, then these local phenomenons might not be required anymore. In such cases, undersampling the signals would be possible to reduce the initial level of noise by averaging several successive samples. Additionally, when applying our periodic method, the shifts between successive periods would be reduced in terms of samples, which might make it easier to get the phenomenons within the receptive field without expanding it. However, it is possible that such approach would also require to reduce the temporal receptive field in order to preserve the approximative stationarity within this receptive field.

Concerning the new shape of the data for exploiting the periodicity, it is to be noticed that the experiments based on the moving average preprocessing could have been performed similarly with non deep learning based methods. Here, we focused on deep learning based methods, both because they generally obtain better results with a linear time complexity once trained, and because they allowed better adaptation to shifts by reshaping the network instead of preprocessing the data. However, our deep learning based methods require to be trained and some applications might require untrained models. For the theoretical cases with exactly periodic data, and known frequency, any signal denoising method could have been applied directly on the preprocessed signal. For more realistic cases, an additional step might be required. This step could consist in a recalibration of the lines so that the shifts are removed or at least reduced. We could also use some kind of guided NLM, i.e. looking for one adequate patch in each line of the grid. By limiting the search for patches to an area corresponding to the receptive field of the network, this search could remain time-efficient.

Another problem of the proposed methods, and of most of signal denoising methods, is that they do not provide any information about how sure they are of their prediction. It would be interesting to check how uncertain the results are, both for knowing how different the results would be by retraining the networks (epistemic uncertainties) or to detect what part of the noise cannot be reduced (aleatoric uncertainty). Cross-validation is somehow linked to this problem, but obtaining robust estimations would require hundreds of trainings, which is not reasonable. Several works have already focused on estimation of the uncertainties in Deep Learning [164], using methods such as Monte-Carlo Dropout [165, 166], returning the variance as an output, often estimated by maximizing the likelihood [167], Bayesian Neural Networks [168, 169], ... Bayesian neural networks and Monte-Carlo dropout are mostly used for modeling epistemic uncertainties, while outputting the variance is used for quantifying aleatoric uncertainties. In our case, the variance could also be estimated as the mean square error in an area around the current sample. The network should then return, for each sample, its prediction and an estimation of its own error. We could also try to return a confidence interval directly, even though it is quite unclear how we could define the loss term.

As shown in Chapter 5, using ensemble methods is promising but our current implementation does not seem to fully exploit the capacity of all networks. Further work in building the datasets, picking the best models to combine and building the combining model still has to be done. Of course, it would also be possible to train or fine-tune the combined models together with the combining model, but it might be restrictive in terms of memory usage. Another possibility would be to chose a method where the combining model makes a selection of the best model for each sample. We already tried to train a combining model to learn a set of weights for each sample, and to use the obtained weights to combine the models optimally, but the training failed. Another possibility would be to simply give a weight of 1 to the best model and 0 for other samples. The combining model should then

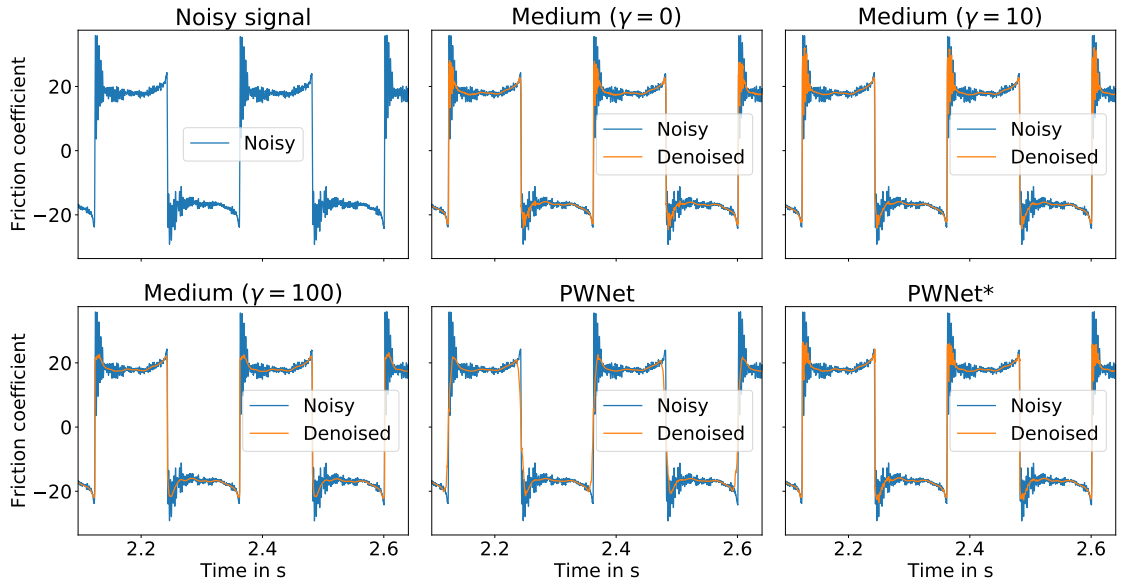


Figure 6.1: Signal acquired on the PSS at 250 rpm

learn a segmentation task. However, with such method, there is a risk that a lot of artifacts are created when switching from one model to another.

Lastly, some work might be done to detect the effects of the sensor and vibrations of the mechanism, i.e., we might try to remove convolutional noise as well as additive noise. This is especially relevant for the signals of the PSS, for which the oscillations due to vibrations are extremely high. An example of signal acquired on this mechanism at a rotation speed of 250 rpm, as well as some denoising results, is shown In Figure 6.1. Here, a high level of noise seems to be present in some parts of the signals. By zooming on this result (Figure 6.2), it appears that this noise is deterministic. Among methods studied in this thesis, using a large regularization seems to be the best one for eliminating this noise.

With a signal at 1250 rpm (Figure 6.3), similar observations can be done. For this signal, the deceleration part, occurring before the discontinuities, seems to be better preserved by the periodic architectures.

These results show that the consideration of the convolutional effects will be necessary for the next steps. It would be possible to consider them as coming from a linear time-invariant system, starting with a second-order system which can give such oscillations, and optimizing it afterwards. The presence of such convolutional and deterministic noise might also indicate that the phenomena that appeared in some denoised signals could already be due to this convolutional effect. Here, it is also possible to extend the proposed work to source separation, for isolating up to four sources:

- The signal of interest;

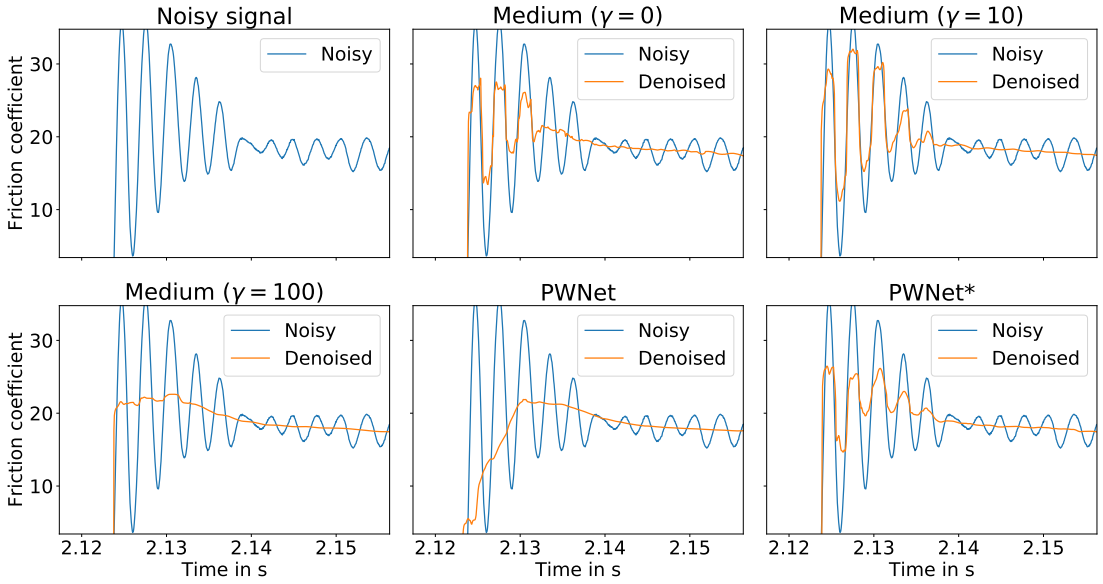


Figure 6.2: Zoom on Figure 6.1

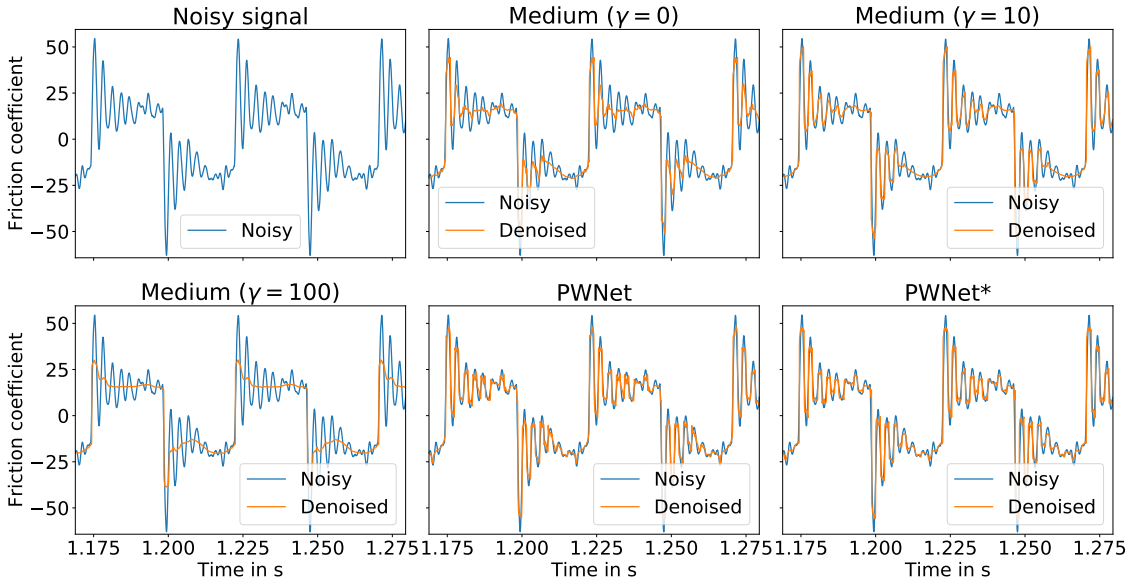


Figure 6.3: Signal acquired on the PSS at 1250 rpm

- The deterministic convolutional noises;
- The random colored noises;
- The random white noises.

From the observations made in the different parts of this thesis, a first implementation could use the 1D architectures trained with large regularization for isolating the signal of interest, and the periodic models for isolating the three other sources. Further studies might require generating signals including such convolutional noises and training the networks with these signals.

Bibliography

[1] S. Solaymani. “CO2 emissions patterns in 7 top carbon emitter economies: the case of transport sector”. In: *Energy* 168 (2019), pp. 989–1001 (cit. on pp. vii, 1).

[2] K. Holmberg, P. Andersson, and A. Erdemir. “Global energy consumption due to friction in passenger cars”. en. In: *Tribology International* 47 (Mar. 2012), pp. 221–234 (cit. on pp. vii, 1).

[3] K. Holmberg and A. Erdemir. “The impact of tribology on energy use and CO2 emission globally and in combustion engine and electric cars”. en. In: *Tribology International* 135 (July 2019), pp. 389–396 (cit. on pp. vii, 1).

[4] K. Levenberg. “A method for the solution of certain non-linear problems in least squares”. In: *Quarterly of applied mathematics* 2.2 (1944), pp. 164–168 (cit. on pp. x, 49, 57).

[5] D. W. Marquardt. “An algorithm for least-squares estimation of nonlinear parameters”. In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441 (cit. on pp. x, 49, 57).

[6] J. J. Moré. “The Levenberg-Marquardt algorithm: implementation and theory”. In: *Numerical analysis*. Springer, 1978, pp. 105–116 (cit. on pp. x, 49, 57).

[7] S. Andersson, A. Söderberg, and S. Björklund. “Friction models for sliding dry, boundary and mixed lubricated contacts”. In: *Tribology international* 40.4 (2007), pp. 580–587 (cit. on pp. x, 49, 59, 68).

[8] D. Hess and A. Soom. “Friction at a lubricated line contact operating at oscillating sliding velocities”. In: *Journal of tribology* 112.1 (1990), pp. 147–152 (cit. on pp. x, 49, 60, 61, 65, 68).

[9] D. Rethage, J. Pons, and X. Serra. “A wavenet for speech denoising”. In: *2018 ICASSP*. IEEE. 2018, pp. 5069–5073 (cit. on pp. x, xi, 37, 39, 74, 77, 78, 101, 138).

[10] Y. Ding and I. W. Selesnick. “Artifact-free wavelet denoising: Non-convex sparse regularization, convex optimization”. In: *IEEE SPL* 22.9 (2015), pp. 1364–1368 (cit. on pp. xi, 70, 89, 90).

[11] D. Stoller, S. Ewert, and S. Dixon. “Wave-u-net: A multi-scale neural network for end-to-end audio source separation”. In: *arXiv preprint arXiv:1806.03185* (2018) (cit. on pp. xii, 40, 109).

[12] G. B. Moody and R. G. Mark. “The impact of the MIT-BIH arrhythmia database”. In: *IEEE Engineering in Medicine and Biology Magazine* 20.3 (2001), pp. 45–50 (cit. on pp. xii, 7, 8, 126).

[13] M. Woydt and R. Wäsche. “The history of the Stribeck curve and ball bearing steels: The role of Adolf Martens”. In: *Wear* 268.11-12 (2010), pp. 1542–1546 (cit. on p. 4).

[14] B. Jacobson. “The Stribeck memorial lecture”. In: *Tribology International* 36.11 (2003), pp. 781–789 (cit. on p. 4).

[15] R. Stribeck. *Kugellager für beliebige Belastungen*. Buchdruckerei AW Schade, Berlin N., 1901 (cit. on p. 4).

[16] R. Stribeck. “Die wesentlichen eigenschaften der gleit-und rollenlager”. In: *Zeitschrift des Vereines Deutscher Ingenieure* 46 (1902), pp. 1341–1348 (cit. on p. 4).

[17] G. A. Kumar and S. Vegi. “Analyzing of an ECG signal mathematically by generating synthetic-ECG data”. In: *IJES* 4.4 (2015), pp. 39–44 (cit. on p. 6).

[18] R. Mark, P. Schluter, G Moody, P Devlin, and D Chernoff. “An annotated ECG database for evaluating arrhythmia detectors”. In: *IEEE Transactions on Biomedical Engineering*. Vol. 29. 8. IEEE-INST ELECTRICAL ELECTRONICS ENGINEERS INC 345 E 47TH ST, NEW YORK, NY . . . 1982, pp. 600–600 (cit. on p. 7).

[19] G. B. Moody, W Muldrow, and R. G. Mark. “A noise stress test for arrhythmia detectors”. In: *Computers in cardiology* 11.3 (1984), pp. 381–384 (cit. on pp. 7, 9).

[20] C. A. Volkert and A. M. Minor. “Focused ion beam microscopy and micromachining”. In: *MRS bulletin* 32.5 (2007), pp. 389–399 (cit. on p. 9).

[21] S. Reyntjens and R. Puers. “A review of focused ion beam applications in microsystem technology”. In: *Journal of micromechanics and microengineering* 11.4 (2001), p. 287 (cit. on p. 9).

[22] D. Stokes. *Principles and practice of variable pressure/environmental scanning electron microscopy (VP-ESEM)*. John Wiley & Sons, 2008 (cit. on p. 9).

[23] W. A. Gardner, A. Napolitano, and L. Paura. “Cyclostationarity: Half a century of research”. en. In: *Signal Processing* 86.4 (Apr. 2006), pp. 639–697 (cit. on p. 11).

[24] J. Rio, F. Momey, C. Ducottet, and O. Alata. “WaveNet based architectures for denoising periodic discontinuous signals and application to friction signals”. In: *2020 EUSIPCO*. IEEE. 2021, pp. 1580–1584 (cit. on pp. 13, 69, 70, 98).

[25] J. Rio, O. Alata, F. Momey, and C. Ducottet. “Leveraging end-to-end denoisers for denoising periodic signals”. In: *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE. 2021 (cit. on pp. 13, 99).

[26] J. Rio, K. Al-Saih, C. Ducottet, O. Alata, and F. Momey. “Towards a unified framework for denoising periodic signals”. In: *To be defined* (To be submitted in 2022) (cit. on pp. 13, 99, 134).

[27] J. Rio, O. Alata, F. Momey, C. Ducottet, and P. Gounet. “Wavenet-Based Architectures Adapted to the Denoising of FIB/SEM Image Sequences”. In: *2021 IEEE 17th International Conference on Automation Science and Engineering*. IEEE. 2021 (cit. on pp. 13, 70).

[28] S. Butterworth et al. “On the theory of filter amplifiers”. In: *Wireless Engineer* 7.6 (1930), pp. 536–541 (cit. on p. 17).

[29] S. Makino. *Audio source separation*. Springer, 2018 (cit. on p. 19).

[30] V. Pandey and V. Giri. “High frequency noise removal from ECG using moving average filters”. In: *2016 International Conference on Emerging Trends in Electrical Electronics & Sustainable Energy Systems (ICETEESES)*. IEEE. 2016, pp. 191–195 (cit. on p. 21).

[31] R. E. Kalman. “A new approach to linear filtering and prediction problems”. In: (1960) (cit. on p. 21).

[32] H. D. Hesar and M. Mohebbi. “ECG denoising using marginalized particle extended kalman filter with an automatic particle weighting strategy”. In: *IEEE journal of biomedical and health informatics* 21.3 (2016), pp. 635–644 (cit. on p. 21).

[33] M. B. Shamsollahi. “ECG denoising and compression using a modified extended Kalman filter structure”. In: *IEEE Transactions on Biomedical Engineering* 55.9 (2008), pp. 2240–2248 (cit. on p. 21).

[34] R. Peesapati, S. L. Sabat, K. Karthik, J Nayak, and N Giribabu. “Efficient hybrid Kalman filter for denoising fiber optic gyroscope signal”. In: *Optik* 124.20 (2013), pp. 4549–4556 (cit. on p. 21).

[35] K Paliwal and A. Basu. “A speech enhancement method based on Kalman filtering”. In: *ICASSP’87. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 12. IEEE. 1987, pp. 177–180 (cit. on p. 21).

[36] I. Daubechies. *Ten lectures on wavelets*. SIAM, 1992 (cit. on p. 22).

[37] C. Valens. “A really friendly guide to wavelets”. In: *ed. Clemens Valens* (1999) (cit. on p. 22).

[38] S. Sardy, P. Tseng, and A. Bruce. “Robust wavelet denoising”. In: *IEEE Transactions on Signal Processing* 49.6 (2001), pp. 1146–1152 (cit. on p. 23).

[39] S. Ghael, A. M. Sayeed, and R. G. Baraniuk. “Improved wavelet denoising via empirical Wiener filtering”. In: *SPIE Technical Conference on Wavelet Applications in Signal Processing*. 1997 (cit. on p. 23).

[40] R.-M. Zhao and H.-m. Cui. “Improved threshold denoising method based on wavelet transform”. In: *2015 7th International Conference on Modelling, Identification and Control (ICMIC)*. IEEE. 2015, pp. 1–4 (cit. on p. 23).

[41] Q. Pan, L. Zhang, G. Dai, and H. Zhang. “Two denoising methods by wavelet transform”. In: *IEEE transactions on signal processing* 47.12 (1999), pp. 3401–3406 (cit. on p. 23).

[42] D. L. Donoho and J. M. Johnstone. “Ideal spatial adaptation by wavelet shrinkage”. In: *biometrika* 81.3 (1994), pp. 425–455 (cit. on p. 23).

[43] M. Alfaouri and K. Daqrouq. “ECG signal denoising by wavelet transform thresholding”. In: *American Journal of applied sciences* 5.3 (2008), pp. 276–281 (cit. on p. 23).

[44] S. G. Chang, B. Yu, and M. Vetterli. “Adaptive wavelet thresholding for image denoising and compression”. In: *IEEE transactions on image processing* 9.9 (2000), pp. 1532–1546 (cit. on p. 23).

[45] G. Chen, T. D. Bui, and A. Krzyzak. “Image denoising using neighbouring wavelet coefficients”. In: *Integrated Computer-Aided Engineering* 12.1 (2005), pp. 99–107 (cit. on p. 23).

[46] H. Demirel and G. Anbarjafari. “Image resolution enhancement by using discrete and stationary wavelet decomposition”. In: *IEEE transactions on image processing* 20.5 (2010), pp. 1458–1460 (cit. on p. 23).

[47] L. Condat. “A Direct Algorithm for 1-D Total Variation Denoising”. en. In: *IEEE Signal Processing Letters* 20.11 (Nov. 2013), pp. 1054–1057 (cit. on pp. 23, 27).

[48] I. W. Selesnick, A. Parekh, and I. Bayram. “Convex 1-D Total Variation Denoising with Non-convex Regularization”. en. In: *IEEE Signal Processing Letters* 22.2 (Feb. 2015), pp. 141–144 (cit. on pp. 23, 27).

[49] I. Selesnick. “Total Variation Denoising Via the Moreau Envelope”. en. In: *IEEE Signal Processing Letters* 24.2 (Feb. 2017), pp. 216–220 (cit. on pp. 23, 27).

[50] C. R. Vogel and M. E. Oman. “Iterative methods for total variation denoising”. In: *SIAM Journal on Scientific Computing* 17.1 (1996), pp. 227–238 (cit. on pp. 23, 27).

[51] Y. Wang, J. Yang, W. Yin, and Y. Zhang. “A new alternating minimization algorithm for total variation image reconstruction”. In: *SIAM Journal on Imaging Sciences* 1.3 (2008), pp. 248–272 (cit. on p. 23).

[52] A. Chambolle. “An algorithm for total variation minimization and applications”. In: *Journal of Mathematical imaging and vision* 20.1 (2004), pp. 89–97 (cit. on p. 23).

[53] A. Beck and M. Teboulle. “Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems”. In: *IEEE transactions on image processing* 18.11 (2009), pp. 2419–2434 (cit. on p. 23).

[54] F. Malgouyres. “Minimizing the total variation under a general convex constraint for image restoration”. In: *IEEE transactions on image processing* 11.12 (2002), pp. 1450–1456 (cit. on p. 23).

[55] Y. Ding and I. W. Selesnick. “Artifact-Free Wavelet Denoising: Non-convex Sparse Regularization, Convex Optimization”. en. In: *IEEE Signal Processing Letters* 22.9 (Sept. 2015), pp. 1364–1368 (cit. on p. 23).

[56] E. J. Candes and F. Guo. “New multiscale transforms, minimum total variation synthesis: Applications to edge-preserving image reconstruction”. In: *Signal Processing* 82.11 (2002), pp. 1519–1543 (cit. on p. 23).

[57] G. R. Easley, D. Labate, and F. Colonna. “Shearlet-based total variation diffusion for denoising”. In: *IEEE Transactions on Image processing* 18.2 (2008), pp. 260–268 (cit. on p. 23).

[58] P. Chatterjee and P. Milanfar. “Patch-based near-optimal image denoising”. In: *IEEE Transactions on Image Processing* 21.4 (2011), pp. 1635–1649 (cit. on p. 24).

[59] A. Buades, B. Coll, and J.-M. Morel. “A non-local algorithm for image denoising”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 2. IEEE. 2005, pp. 60–65 (cit. on p. 24).

[60] B. H. Tracey and E. L. Miller. “Nonlocal Means Denoising of ECG Signals”. In: *IEEE Transactions on Biomedical Engineering* 59.9 (Sept. 2012). Conference Name: IEEE Transactions on Biomedical Engineering, pp. 2383–2386 (cit. on pp. 24, 113, 128, 129).

[61] J. V. Manjón, J. Carbonell-Caballero, J. J. Lull, et al. “MRI denoising using non-local means”. In: *Medical image analysis* 12.4 (2008), pp. 514–523 (cit. on p. 24).

[62] C.-A. Deledalle, V. Duval, and J. Salmon. “Non-local methods with shape-adaptive patches (NLM-SAP)”. In: *Journal of Mathematical Imaging and Vision* 43.2 (2012), pp. 103–120 (cit. on p. 24).

[63] C. Kervrann and J. Boulanger. “Optimal spatial adaptation for patch-based image denoising”. In: *IEEE Transactions on Image Processing* 15.10 (2006), pp. 2866–2878 (cit. on p. 24).

[64] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. “Image denoising by sparse 3-D transform-domain collaborative filtering”. In: *IEEE Transactions on image processing* 16.8 (2007), pp. 2080–2095 (cit. on p. 24).

[65] M. N. Schmidt and R. K. Olsson. “Single-channel speech separation using sparse non-negative matrix factorization”. In: *Ninth International Conference on Spoken Language Processing*. 2006 (cit. on p. 26).

[66] F. Weninger, J. L. Roux, J. R. Hershey, and S. Watanabe. “Discriminative NMF and its application to single-channel source separation”. In: *Fifteenth Annual Conference of the International Speech Communication Association*. 2014 (cit. on p. 26).

[67] T. Virtanen. “Monaural Sound Source Separation by Nonnegative Matrix Factorization With Temporal Continuity and Sparseness Criteria”. en. In: *IEEE Transactions on Audio, Speech and Language Processing* 15.3 (Mar. 2007), pp. 1066–1074 (cit. on p. 26).

[68] C. Févotte, E. Vincent, and A. Ozerov. “Single-channel audio source separation with NMF: divergences, constraints and algorithms”. In: *Audio Source Separation* (2018), pp. 1–24 (cit. on p. 26).

[69] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2021 (cit. on p. 26).

[70] Y. Wang. *Smoothing splines: methods and applications*. Chapman and Hall/CRC, 2019 (cit. on p. 27).

[71] J. H. Friedman. “Multivariate adaptive regression splines”. In: *The annals of statistics* (1991), pp. 1–67 (cit. on p. 27).

[72] M. P. Wand and M. C. Jones. *Kernel smoothing*. CRC press, 1994 (cit. on p. 27).

[73] P. Comon. “Independent component analysis, a new concept?” In: *Signal processing* 36.3 (1994), pp. 287–314 (cit. on p. 27).

[74] C. Jutten and J. Herault. “Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture”. In: *Signal processing* 24.1 (1991), pp. 1–10 (cit. on p. 27).

[75] M. Davies and N. Mitianoudis. “Simple mixture model for sparse overcomplete ICA”. In: *IEE Proceedings-Vision, Image and Signal Processing* 151.1 (2004), pp. 35–43 (cit. on p. 27).

[76] M. A. Casey and A. Westner. “Separation of mixed audio sources by independent subspace analysis”. In: *ICMC*. 2000, pp. 154–161 (cit. on p. 27).

[77] M. E. Davies and C. J. James. “Source separation using single channel ICA”. In: *Signal Processing* 87.8 (2007), pp. 1819–1832 (cit. on p. 27).

[78] N. E. Huang, Z. Shen, S. R. Long, et al. “The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis”. In: *Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences* 454.1971 (1998), pp. 903–995 (cit. on p. 28).

[79] M. Blanco-Velasco, B. Weng, and K. E. Barner. “ECG signal denoising and baseline wander correction based on the empirical mode decomposition”. In: *Computers in biology and medicine* 38.1 (2008), pp. 1–13 (cit. on p. 28).

[80] B. Mijović, M. De Vos, I. Gligorijević, J. Taelman, and S. Van Huffel. “Source separation from single-channel recordings by combining empirical-mode decomposition and independent component analysis”. In: *IEEE transactions on biomedical engineering* 57.9 (2010), pp. 2188–2196 (cit. on p. 28).

[81] A. Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE transactions on Information Theory* 13.2 (1967), pp. 260–269 (cit. on p. 28).

[82] S. T. Roweis. “One Microphone Source Separation”. en. In: (2000), p. 7 (cit. on p. 28).

[83] A. Varga and R. K. Moore. “Hidden Markov model decomposition of speech and noise”. In: *International Conference on Acoustics, Speech, and Signal Processing*. IEEE. 1990, pp. 845–848 (cit. on p. 28).

[84] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016 (cit. on p. 28).

[85] A. Shrestha and A. Mahmood. “Review of deep learning algorithms and architectures”. In: *IEEE Access* 7 (2019), pp. 53040–53065 (cit. on p. 28).

[86] D. Wang and J. Chen. “Supervised speech separation based on deep learning: An overview”. In: *IEEE/ACM TASLP* 26.10 (2018), pp. 1702–1726 (cit. on p. 28).

[87] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on pp. 29, 78, 188).

[88] G. Hinton, N. Srivastava, and K. Swersky. “Neural networks for machine learning lecture 6a overview of mini-batch gradient descent”. In: *Cited on* 14.8 (2012), p. 2 (cit. on pp. 29, 187).

[89] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 31).

[90] K. Cho, B. Van Merriënboer, C. Gulcehre, et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014) (cit. on p. 31).

[91] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105 (cit. on p. 33).

[92] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on p. 33).

[93] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014) (cit. on p. 33).

[94] C. Szegedy, W. Liu, Y. Jia, et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9 (cit. on pp. 33, 34).

[95] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Thirty-first AAAI conference on artificial intelligence*. 2017 (cit. on p. 34).

[96] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. “Visualizing the loss landscape of neural nets”. In: *arXiv preprint arXiv:1712.09913* (2017) (cit. on p. 34).

[97] P. Vincent, H. Larochelle, I. Lajoie, et al. “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.” In: *Journal of machine learning research* 11.12 (2010) (cit. on pp. 36, 69).

[98] X. Lu, Y. Tsao, S. Matsuda, and C. Hori. “Speech enhancement based on deep denoising autoencoder.” In: *Interspeech*. Vol. 2013. 2013, pp. 436–440 (cit. on p. 36).

[99] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee. “A regression approach to speech enhancement based on deep neural networks”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.1 (2014), pp. 7–19 (cit. on p. 36).

[100] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee. “An Experimental Study on Speech Enhancement Based on Deep Neural Networks”. en. In: *IEEE Signal Processing Letters* 21.1 (Jan. 2014), pp. 65–68 (cit. on p. 36).

[101] X. Feng, Y. Zhang, and J. Glass. “Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition”. en. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy: IEEE, May 2014, pp. 1759–1763 (cit. on p. 36).

[102] S. Uhlich, M. Porcu, F. Giron, et al. “Improving music source separation based on deep neural networks through data augmentation and network blending”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 261–265 (cit. on pp. 36, 135).

[103] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis. “Deep learning for monaural speech separation”. en. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy: IEEE, May 2014, pp. 1562–1566 (cit. on p. 36).

[104] A. Maas, Q. V. Le, T. M. O’neil, et al. “Recurrent neural networks for noise reduction in robust ASR”. In: (2012) (cit. on p. 36).

[105] F. Weninger, J. R. Hershey, J. Le Roux, and B. Schuller. “Discriminatively trained recurrent neural networks for single-channel speech separation”. In: *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE. 2014, pp. 577–581 (cit. on p. 36).

[106] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis. “Joint optimization of masks and deep recurrent neural networks for monaural source separation”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.12 (2015), pp. 2136–2147 (cit. on p. 36).

[107] F. Weninger, H. Erdogan, S. Watanabe, et al. “Speech enhancement with LSTM recurrent neural networks and its application to noise-robust ASR”. In: *LVA/ICA*. Springer. 2015, pp. 91–99 (cit. on p. 36).

[108] J. Chen and D. Wang. “Long short-term memory for speaker generalization in supervised speech separation”. In: *The Journal of the Acoustical Society of America* 141.6 (2017), pp. 4705–4714 (cit. on p. 36).

[109] Y. Luo and N. Mesgarani. “TasNet: time-domain audio separation network for real-time, single-channel speech separation”. en. In: *arXiv:1711.00541 [cs, eess]* (Apr. 2018). arXiv: 1711.00541 (cit. on p. 36).

[110] C. T. Arsene, R. Hankins, and H. Yin. “Deep learning models for denoising ECG signals”. In: *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE. 2019, pp. 1–5 (cit. on p. 36).

[111] N. Takahashi, N. Goswami, and Y. Mitsufuji. “Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation”. In: *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE. 2018, pp. 106–110 (cit. on p. 36).

[112] Z. Shi, H. Lin, L. Liu, et al. “FurcaNet: An end-to-end deep gated convolutional, long short-term memory, deep neural networks for single channel speech separation”. In: *arXiv preprint arXiv:1902.00651* (2019) (cit. on p. 36).

[113] S. Abdulatif, K. Armanious, K. Guirguis, J. T Sajeev, and B. Yang. “Aegan: Time-frequency speech denoising via generative adversarial networks”. In: *2020 EUSIPCO*. IEEE. 2021, pp. 451–455 (cit. on p. 36).

[114] C. Donahue, B. Li, and R. Prabhavalkar. “Exploring Speech Enhancement with Generative Adversarial Networks for Robust Speech Recognition”. en. In: *arXiv:1711.05747 [cs, eess]* (Oct. 2018). arXiv: 1711.05747 (cit. on p. 36).

[115] P. Chandna, M. Miron, J. Janer, and E. Gómez. “Monoaural Audio Source Separation Using Deep Convolutional Neural Networks”. en. In: *Latent Variable Analysis and Signal Separation*. Ed. by P. Tichavský, M. Babaie-Zadeh, O. J. Michel, and N. Thirion-Moreau. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 258–266 (cit. on p. 36).

[116] N. Takahashi and Y. Mitsufuji. “Multi-scale multi-band densenets for audio source separation”. In: *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE. 2017, pp. 21–25 (cit. on p. 37).

[117] S. Venkataramani, E. Tzinis, and P. Smaragdis. “A Style Transfer Approach to Source Separation”. en. In: *arXiv:1905.00151 [cs, eess]* (May 2019). arXiv: 1905.00151 (cit. on p. 37).

[118] S. Fu, Y. Tsao, X. Lu, and H. Kawai. “Raw waveform-based speech enhancement by fully convolutional networks”. In: *2017 APSIPA ASC*. IEEE. 2017, pp. 006–012 (cit. on p. 37).

[119] S. Fu, T. Wang, Y. Tsao, X. Lu, and H. Kawai. “End-to-end waveform utterance enhancement for direct evaluation metrics optimization by fully convolutional neural networks”. In: *IEEE/ACM TASLP* 26.9 (2018), pp. 1570–1584 (cit. on p. 37).

[120] N. Alamdari, A. Azarang, and N. Kehtarnavaz. “Improving deep speech denoising by Noisy2Noisy signal mapping”. In: *Applied Acoustics* 172 (2021), p. 107631 (cit. on p. 37).

[121] A. v. Oord, S. Dieleman, H. Zen, et al. “Wavenet: A generative model for raw audio”. In: *arXiv preprint arXiv:1609.03499* (2016) (cit. on pp. 37, 39).

[122] L. Casas, A. Klimmek, N. Navab, and V. Belagiannis. “Adversarial Signal Denoising with Encoder-Decoder Networks”. In: *arXiv preprint arXiv:1812.08555* (2018) (cit. on p. 40).

[123] F. G. Germain, Q. Chen, and V. Koltun. “Speech denoising with deep feature losses”. In: *arXiv preprint arXiv:1806.10522* (2018) (cit. on p. 40).

[124] Y. Luo and N. Mesgarani. “Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation”. In: *IEEE/ACM TASLP* 27.8 (2019), pp. 1256–1266 (cit. on p. 40).

[125] S. Pascual, A. Bonafonte, and J. Serra. “SEGAN: Speech enhancement generative adversarial network”. In: *arXiv preprint arXiv:1703.09452* (2017) (cit. on pp. 41, 160).

[126] R. Giri, U. Isik, and A. Krishnaswamy. “Attention wave-u-net for speech enhancement”. In: *2019 WASPAA*. IEEE. 2019, pp. 249–253 (cit. on p. 41).

[127] A. V. Dandawate and G. B. Giannakis. “Statistical tests for presence of cyclostationarity”. In: *IEEE Transactions on signal processing* 42.9 (1994), pp. 2355–2369 (cit. on p. 49).

[128] B. Armstrong-Hélouvry, P. Dupont, and C. C. De Wit. “A survey of models, analysis tools and compensation methods for the control of machines with friction”. en. In: *Automatica* 30.7 (July 1994), pp. 1083–1138 (cit. on p. 49).

[129] P. Dupont, V. Hayward, B. Armstrong, and F. Altpeter. “Single state elastoplastic friction models”. en. In: *IEEE Transactions on Automatic Control* 47.5 (May 2002), pp. 787–792 (cit. on p. 49).

[130] C. C. De Wit, H. Olsson, K. J. Astrom, and P. Lischinsky. “A new model for control of systems with friction”. In: *IEEE Transactions on automatic control* 40.3 (1995), pp. 419–425 (cit. on p. 49).

[131] K. Åström and C. De Wit. “Revisiting the LuGre friction model”. In: *IEEE Control systems magazine* 28.6 (2008), pp. 101–114 (cit. on pp. 49, 60).

[132] A. R. Conn, N. I. Gould, and P. L. Toint. *Trust region methods*. SIAM, 2000 (cit. on pp. 49, 188).

[133] Y.-x. Yuan. “A review of trust region algorithms for optimization”. In: *Iciam*. Vol. 99. 1. 2000, pp. 271–282 (cit. on pp. 49, 188).

[134] *MATLAB version 9.3.0.713579 (R2020b)*. The Mathworks, Inc. Natick, Massachusetts, 2020 (cit. on pp. 49, 57, 61).

[135] A. Tustin. “The effects of backlash and of speed-dependent friction on the stability of closed-cycle control systems”. In: *Journal of the Institution of Electrical Engineers-Part IIA: Automatic Regulators and Servo Mechanisms* 94.1 (1947), pp. 143–151 (cit. on p. 60).

[136] L. C. Bo and D Pavelescu. “The friction-speed relation and its influence on the critical velocity of stick-slip motion”. In: *Wear* 82.3 (1982), pp. 277–289 (cit. on p. 60).

[137] M. Zhussip, S. Soltanayev, and S. Y. Chun. “Training Deep Learning Based Image Denoisers From Undersampled Measurements Without Ground Truth and Without Image Prior”. en. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 10247–10256 (cit. on p. 69).

[138] J. Xu, Y. Huang, L. Liu, et al. “Noisy-As-Clean: Learning Unsupervised Denoising from the Corrupted Image”. In: *arXiv preprint arXiv:1906.06878* (2019) (cit. on pp. 69, 160).

[139] C. A. Metzler, A. Mousavi, R. Heckel, and R. G. Baraniuk. “Unsupervised Learning with Stein’s Unbiased Risk Estimator”. en. In: *arXiv:1805.10531 [cs, stat]* (July 2020). arXiv: 1805.10531 (cit. on pp. 69, 160).

[140] X. Lu, S. Matsuda, C. Hori, and H. Kashioka. “Speech restoration based on deep learning autoencoder with layer-wised pretraining”. In: *Thirteenth Annual Conference of the International Speech Communication Association*. 2012 (cit. on p. 69).

[141] J. Tremblay, A. Prakash, D. Acuna, et al. “Training deep networks with synthetic data: Bridging the reality gap by domain randomization”. In: *Proceedings of the IEEE CVPR Workshops*. 2018, pp. 969–977 (cit. on p. 69).

[142] A. Prakash, S. Boochoon, M. Brophy, et al. “Structured domain randomization: Bridging the reality gap by context-aware synthetic data”. In: *2019 ICRA*. IEEE. 2019, pp. 7249–7255 (cit. on p. 69).

[143] N. Mayer, E. Ilg, P. Hausser, et al. “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation”. In: *Proceedings of the IEEE CVPR*. 2016, pp. 4040–4048 (cit. on p. 69).

[144] A. Dosovitskiy, P. Fischer, E. Ilg, et al. “Flownet: Learning optical flow with convolutional networks”. In: *Proceedings of the IEEE ICCV*. 2015, pp. 2758–2766 (cit. on p. 69).

[145] A. Ignatov, N. Kobyshev, R. Timofte, K. Vanhoey, and L. v. Gool. “DSLR-quality photos on mobile devices with deep convolutional networks”. In: *Proceedings of the IEEE ICCV*. 2017, pp. 3277–3285 (cit. on p. 75).

[146] P. Wang, H. Zhang, and V. Patel. “SAR image despeckling using a convolutional neural network”. In: *IEEE SPL* 24.12 (2017), pp. 1763–1767 (cit. on p. 75).

[147] M. Peng, J. Murray-Bruce, K. K. Berggren, and V. K. Goyal. “Source shot noise mitigation in focused ion beam microscopy by time-resolved measurement”. In: *Ultramicroscopy* 211 (2020), p. 112948 (cit. on p. 93).

[148] E Giannatou, G Papavieros, V Constantoudis, H Papageorgiou, and E Gogolides. “Deep learning denoising of SEM images towards noise-reduced LER measurements”. In: *Microelectronic Engineering* 216 (2019), p. 111051 (cit. on p. 93).

[149] P. Singh and G. Pradhan. “A new ECG denoising framework using generative adversarial network”. In: *IEEE/ACM transactions on computational biology and bioinformatics* 18.2 (2020), pp. 759–764 (cit. on pp. 127–129).

[150] P. Singh and G. Pradhan. “Exploring the non-local similarity present in variational mode functions for effective ECG denoising”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 861–865 (cit. on pp. 128, 129).

[151] H.-T. Chiang, Y.-Y. Hsieh, S.-W. Fu, et al. “Noise reduction in ECG signals using fully convolutional denoising autoencoders”. In: *IEEE Access* 7 (2019), pp. 60806–60813 (cit. on p. 128).

[152] P. Singh and G. Pradhan. “Variational mode decomposition based ECG denoising using non-local means and wavelet domain filtering”. In: *Australasian physical & engineering sciences in medicine* 41.4 (2018), pp. 891–904 (cit. on pp. 128, 129).

[153] S. Chatterjee, R. S. Thakur, R. N. Yadav, L. Gupta, and D. K. Raghuvanshi. “Review of noise removal techniques in ECG signals”. In: *IET Signal Processing* 14.9 (2020), pp. 569–590 (cit. on p. 128).

[154] X.-L. Zhang and D. Wang. “A deep ensemble learning method for monaural speech separation”. In: *IEEE/ACM transactions on audio, speech, and language processing* 24.5 (2016), pp. 967–977 (cit. on pp. 135, 136).

[155] X. Lu, Y. Tsao, S. Matsuda, and C. Hori. “Ensemble modeling of denoising autoencoder for speech spectrum restoration”. In: *Fifteenth Annual Conference of the International Speech Communication Association*. 2014 (cit. on p. 135).

[156] J. Han and M. van der Baan. “Microseismic and seismic denoising via ensemble empirical mode decomposition and adaptive thresholding”. In: *Geophysics* 80.6 (2015), KS69–KS80 (cit. on p. 135).

[157] Z. Wu and N. E. Huang. “Ensemble empirical mode decomposition: a noise-assisted data analysis method”. In: *Advances in adaptive data analysis* 1.01 (2009), pp. 1–41 (cit. on p. 135).

[158] M. E. Torres, M. A. Colominas, G. Schlotthauer, and P. Flandrin. “A complete ensemble empirical mode decomposition with adaptive noise”. In: *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2011, pp. 4144–4147 (cit. on p. 135).

[159] P. Nguyen and J.-M. Kim. “Adaptive ECG denoising using genetic algorithm-based thresholding and ensemble empirical mode decomposition”. In: *Information Sciences* 373 (2016), pp. 499–511 (cit. on p. 135).

[160] X. Yang, Y. Xu, Y. Quan, and H. Ji. “Image denoising via sequential ensemble learning”. In: *IEEE Transactions on Image Processing* 29 (2020), pp. 5038–5049 (cit. on p. 135).

[161] L. Deng and J. C. Platt. “Ensemble deep learning for speech recognition”. In: *Fifteenth annual conference of the international speech communication association*. 2014 (cit. on p. 136).

[162] A. Benou, R. Veksler, A. Friedman, and T. R. Raviv. “Ensemble of expert deep neural networks for spatio-temporal denoising of contrast-enhanced MRI sequences”. In: *Medical image analysis* 42 (2017), pp. 145–159 (cit. on p. 136).

[163] S. Soltanayev and S. Y. Chun. “Training and Refining Deep Learning Based Denoisers without Ground Truth Data”. en. In: *arXiv:1803.01314 [cs, stat]* (Mar. 2019). arXiv: 1803.01314 (cit. on p. 160).

[164] M. Abdar, F. Pourpanah, S. Hussain, et al. “A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges”. en. In: *arXiv:2011.06225 [cs]* (Jan. 2021). arXiv: 2011.06225 (cit. on p. 161).

[165] Y. Gal and Z. Ghahramani. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. en. In: (), p. 10 (cit. on p. 161).

[166] Y. Gal, J. Hron, and A. Kendall. “Concrete Dropout”. en. In: *arXiv:1705.07832 [stat]* (May 2017). arXiv: 1705.07832 (cit. on p. 161).

[167] A. Kendall and Y. Gal. “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” en. In: *arXiv:1703.04977 [cs]* (Oct. 2017). arXiv: 1703.04977 (cit. on p. 161).

[168] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. “Weight uncertainty in neural network”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1613–1622 (cit. on p. 161).

[169] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun. “Hands-on Bayesian Neural Networks – a Tutorial for Deep Learning Users”. en. In: *arXiv:2007.06823 [cs, stat]* (July 2020). arXiv: 2007.06823 (cit. on p. 161).

[170] E. P. Wigner. “On the quantum correction for thermodynamic equilibrium”. In: *Part I: Physical Chemistry. Part II: Solid State Physics*. Springer, 1997, pp. 110–120 (cit. on p. 183).

[171] M. Long, H. Zhu, J. Wang, and M. I. Jordan. “Unsupervised domain adaptation with residual transfer networks”. In: *arXiv preprint arXiv:1602.04433* (2016) (cit. on p. 185).

[172] N. Qian. “On the momentum term in gradient descent learning algorithms”. In: *Neural networks* 12.1 (1999), pp. 145–151 (cit. on p. 187).

[173] M. D. Zeiler. “Adadelta: an adaptive learning rate method”. In: *arXiv preprint arXiv:1212.5701* (2012) (cit. on p. 188).

[174] J. Duchi, E. Hazan, and Y. Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of machine learning research* 12.7 (2011) (cit. on p. 188).

[175] T. Dozat. “Incorporating nesterov momentum into adam”. In: (2016) (cit. on p. 188).

[176] S. J. Reddi, S. Kale, and S. Kumar. “On the convergence of adam and beyond”. In: *arXiv preprint arXiv:1904.09237* (2019) (cit. on p. 188).

[177] S. Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016) (cit. on p. 188).

[178] Y.-x. Yuan. “Recent advances in trust region algorithms”. In: *Mathematical Programming* 151.1 (2015), pp. 249–281 (cit. on p. 188).

A Basic definitions of signal processing

Here, we recall some signal processing notions and notations that will be used in this work. We recall the definitions for discrete random signals.

A.1 Random signal

A random signal is a family of time-indexed random variables. The family can theoretically contain an infinity of random variables. Each random variable has its own distribution, and the random variables can be dependent on each other. As already said, we are interested in cyclostationary signals, where dependencies between variables create a periodic pattern. We will now define more rigorously the underlying concepts of signal processing.

A.2 Autocorrelation, autocovariance

For a random signal X of real values, the autocorrelation is defined as:

$$R_{XX}(n_1, n_2) = \mathbb{E}[X_{n_1} X_{n_2}] \tag{A.1}$$

The autocovariance is defined as the autocorrelation of the centered signal :

$$\begin{aligned} C_{XX}(n_1, n_2) &= \mathbb{E}[X^c_{n_1} X^x_{n_2}] = \mathbb{E}[X_{n_1} X_{n_2}] - \mathbb{E}[X_{n_1}]\mathbb{E}[X_{n_2}] \\ &= R_{XX}(n_1, n_2) - \mathbb{E}[X_{n_1}]\mathbb{E}[X_{n_2}] \end{aligned} \tag{A.2}$$

A.3 Stationary signals

In our studies, we will assimilate stationarity to second-order wide-sense stationarity, which is defined as :

$$\begin{cases} \exists \mu_X, |\mu_X| < \infty, \forall n, \mathbb{E}[X_n] = \mu_x \\ \exists \sigma_X < \infty, \forall n, \mathbb{E}[X_n^2] = \sigma_X^2 + \mu_x^2 \\ \forall (n_1, n_2, \tau), R_{XX}(n_1, n_1 + \tau) = R_{XX}(n_2, n_2 + \tau) \end{cases} \tag{A.3}$$

With the third property, it is conventional to define a one variable autocorrelation function with the same notation as the autocorrelation :

$$\forall (n, \tau), R_{XX}(\tau) = R_{XX}(n, n + \tau) \tag{A.4}$$

In this case, as the autocorrelation depends only on the shift between the two timestamps and the two expectations are constant, the same properties will appear with the autocovariance function.

A.4 Spectral density

For computing the spectral (frequency domain) content of a random stationary signals, it is possible to apply the Fourier transform on its autocorrelation function. The result is called Spectral density and is noted $S_X(f)$.

A.5 Cyclostationary signal

While we have already defined a cyclostationary signal, we can now use the previous definitions. A random signal X is said to be (second order wide-sense) cyclostationary if there is a period T such that:

$$\begin{cases} \forall t, \mathbb{E}\left[X_{t+T}\right] = \mathbb{E}\left[X_t\right] \\ \forall t, \forall \tau, R_{XX}(t, t + \tau) = R_{XX}(t + T, t + T + \tau) \end{cases} \tag{A.5}$$

Note that for such a signal, the sub-signal $\mathbf{y}_n = X_{nT+n_0}$ (with n_0 any integer) will be a stationary signal.

A.6 Fourier transform

The Fourier transform is a mathematical operator that projects a signal on a set of exponential functions. It is used for obtaining a representation of the spectral content of a signal.

For a (deterministic) signal $\mathbf{x}$ of infinite length, the Discrete Time Fourier Transform (DTFT) for the frequency f is defined as:

$$\mathcal{F}_d(\mathbf{x})(f) = \sum_{n=-\infty}^{\infty} \mathbf{x}_n e^{-j2\pi f n T_e} \tag{A.6}$$

As we will be working with finite acquisitions, we will be mostly interested in the Discrete Fourier Transform (DFT):

$$\forall k \in \{0, 1, \ldots, N-1\},\ \mathcal{F}(\mathbf{x})(f_k) = \sum_{n=0}^{N-1} x_n e^{-j2\pi f_k n T_e} = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi k n}{N}} \tag{A.7}$$

with N the amount of samples and $f_k = k\frac{f_e}{N}$. Note that apart from the quantization obtained when sampling the frequency axis, this definition can be obtained from the DTFT by assuming that the signal is null before and after the acquisition. Remark that the values for other frequencies can also be computed with the same formula. However, the family of functions $\left\{t :\to e^{-j2\pi f_k n T_e}\right\}_{k\in 0,1,\ldots,N-1}$ constitute a basis of the functions in this N-dimensional space. It is therefore not necessary to compute the other values as they will linearly depend on the already computed ones.

From the Discrete Fourier transform, it is possible to retrieve the signal by applying the inverse Fourier Transform:

$$\mathbf{x}_n = \frac{1}{N}\sum_{k=0}^{N-1} \mathcal{F}(\mathbf{x})(k f_e) e^{j2\pi k f_e n T_e} = \frac{1}{N}\sum_{k=0}^{N-1} \mathcal{F}(\mathbf{x})(k f_e) e^{j\frac{2\pi k n}{N}} \tag{A.8}$$

Among important properties of the Fourier transform, we can cite the linearity. For two signals $\mathbf{x}$ and $\mathbf{y}$ and a scalar λ, the Fourier transform verifies:

$$\mathcal{F}(\lambda\mathbf{x} + \mathbf{y}) = \lambda\mathcal{F}(\mathbf{x}) + \mathcal{F}(\mathbf{y}) \tag{A.9}$$

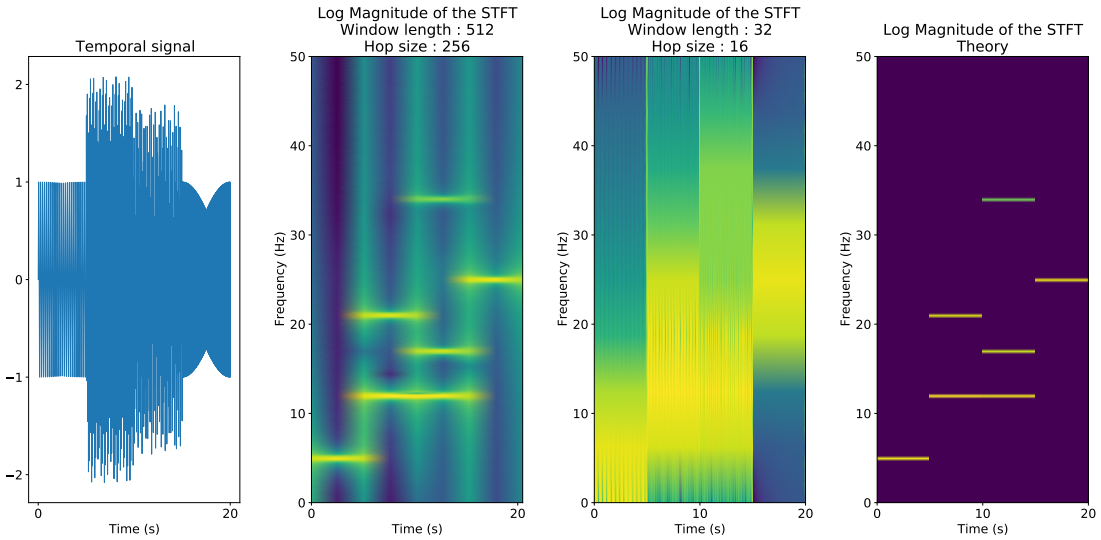


Figure A.1: Illustration of the impact of the window's length on resolution of STFT

A.7 Time-frequency representations

Time-frequency representations aim to give the local spectral content for a certain time slot instead of the global spectral content as the Fourier transform does. One of the most common methods is the Short-Time Fourier Transform (STFT). For the signal $\mathbf{x}$, the value $\mathbf{X}_{n,\cdot}$ at time slot n and frequency f is obtained as:

$$\mathbf{X}_{n,f} = \sum_{k=0}^{N-1} \mathbf{x}_{n+k} \mathbf{w}_k e^{-j2\pi f k T_e} \tag{A.10}$$

with $\mathbf{w}$ a window function, such as the rectangular window, the triangular window, the Hamming window, ... The window function has a shorter temporal support than the signal $\mathbf{x}$, so the STFT applies the Fourier Transform on a segment of the signal. This operation can then be repeated for several timestamps.

While it would theoretically be possible to compute $\mathbf{X}_{n,\cdot}$ for all timestamps n, that would also require a large computation time. Therefore, it is conventional to compute the STFT on a smaller amount of samples when using a larger window. Generally, for a window of length L, one sample will be computed every $\frac{L}{2}$ samples, which is denoted as a hop size of $\frac{L}{2}$. Using large windows will then result in a large hop size and a large amount of data for each computation, resulting in a low temporal resolution and a high frequency resolution. An illustration of these phenomenons is shown in Figure A.1.

The STFT is sometimes used with other scales for the representation, with a higher resolution for low frequency than high frequencies. This is the case for Mel Spectrograms that are often used for audio signals.

Some other methods, for example based on the Wigner-Ville transform [170], which is a nonlinear transform of the energy of the signal into a time-frequency domain, can also be used.

Lastly, even though it is not really a time-frequency representation but rather a time-scale representation, it is possible to obtain some representations based on the Wavelet transform. The obtained representations are named "Scalograms". Contrary to the STFT, which requires an additional step of translation, the translation is already a parameter of the Wavelet transform itself. The scalograms are simply a representation of the values obtained at different scales, which can be linked to frequencies, and translations. Further details about the Wavelet transform will be given in Section 1.4.2.

As the time-frequency representations convert the 1D signals into matrices, it becomes possible to interpret them as images, on which it is then possible to apply methods first developed for image denoising, or more generally image processing.

A.8 White and colored noises

A white noise is defined as a centered (its expectation is zero) stationary signal, with uncorrelated variables:

$$\begin{cases} \forall n, \mathbb{E}[X_n] = 0 \\ R_{XX}(\tau) = \sigma_X^2 \delta_n = \begin{cases} \sigma_X^2 \text{ if } n = 0 \\ 0 \text{ otherwise} \end{cases} \end{cases} \tag{A.11}$$

For a white noise, the spectral density will be a constant $\forall f, S_X(f) = \sigma_X^2$. The name of "White noise" comes from this property, by association with the spectrum of white light.

In our work, we will consider that the property of having a centered noise is always verified except for deterministic noises. We are making this assumption because it is generally not possible to distinguish whether a non-zero mean would be part of the signal or of a noise, except by having an *a priori* knowledge.

Therefore, only the second property will let us decide whether a noise is white or not. If the second property is not verified, we will say that the noise is a colored noise.

B

Concepts of Machine Learning

Machine Learning designs an ensemble of methods that use an ensemble of data to adapt their parameters and improve their results on a specific task. Most methods that we will use throughout this thesis are based on this approach.

Several methods allow one to find the best set of parameters:

Supervised learning The method uses a dataset of annotated data (both the input and expected output are known). Once trained, the model can be applied to other data.

Unsupervised learning The method is trained based on a dataset where the input signals are not linked to their corresponding expecting output, either because no ground truth is available, or because some expected outputs are available without the possibility to associate them with a specific input.

Semi-supervised learning The method combines supervised learning for some data and unsupervised learning for some other data.

Untrained methods Some methods can be applied directly on a signal without being trained previously. They will adapt their parameters for each signal. This could be assimilated to an unsupervised learning with a dataset of only one signal.

These methods are not always exclusive. It is for example possible to start a training with a supervised method and finish with an unsupervised one. This is the case for example in domain adaptation, such as in [171], where neural networks classifiers previously trained on labeled data are fine-tuned through an unsupervised method to adapt to the new distribution of the data.

B.1 Optimization of a method: gradient descent and trust-region algorithms

Adaptation of methods to a set of data is made by minimizing a "loss function" depending on the parameters of the method. As it is generally not possible to find the exact global minimum, it is necessary to design an algorithm that will approximate this minimum.

One of the most common methods is the gradient descent algorithm. This method uses the derivatives of the loss function according to the parameters of the model to decide how to adapt these parameters.

Let's consider a method designed with P parameters $\mathbf{p} = (\mathbf{p}_0, \dots, \mathbf{p}_{P-1})^T$. Each data $\mathcal{X}_i$ from the dataset $\mathcal{X} = \{\mathcal{X}_i\}_{i=0,\dots,D-1}$, which might be a set of couples linking an input to its expected output, a set of inputs only, or only one signal for untrained methods, will be associated with its own partial loss function l_i which depends on the parameters of the model. The global loss function is then defined as the average of all partial loss functions:

$$\mathcal{L}(\mathbf{p}) = \frac{1}{D} \sum_{i=0}^{D-1} l_i(\mathbf{p}) \tag{B.1}$$

and the optimization problem can be expressed as finding:

$$\hat{\mathbf{p}} = \arg\min_{\mathbf{p}} \mathcal{L}(\mathbf{p}) \tag{B.2}$$

If the loss function is differentiable, then it is possible to compute its gradient:

$$\nabla\mathcal{L}(\mathbf{p}) = \begin{pmatrix} \frac{\delta\mathcal{L}}{\delta\mathbf{p}_0}(\mathbf{p}) \\ \vdots \\ \frac{\delta\mathcal{L}}{\delta\mathbf{p}_{P-1}}(\mathbf{p}) \end{pmatrix} \tag{B.3}$$

From an initial point, gradient descent consists to use the opposite of the gradient as the update direction d_n at step n, i.e. that for a set of parameters $\hat{\mathbf{p}}^{(n)}$ obtained at iteration n, the set of parameters after iteration $n+1$ will be obtained by following a descent direction d_n defined by the gradient:

$$\hat{\mathbf{p}}^{(n+1)} = \hat{\mathbf{p}}^{(n)} - \gamma d_n = \hat{\mathbf{p}}^{(n)} - \gamma\nabla\mathcal{L}(\mathbf{p}^{(n)}) \tag{B.4}$$

with γ the iteration step, or learning rate, which can be fixed for all iterations, or be adapted to obtain the maximal improvement after each iteration. The process is then repeated until

a stop criterion has been reached: maximal amount of iterations, no significant change in the values of $\mathbf{p}$, no significant change in the value of $\mathcal{L}$, ...

Generally, computing all the partial loss functions l_i to obtain the exact global loss function $\mathcal{L}$ is time and memory consuming. Therefore, the global loss function is often estimated with only one value in stochastic gradient descent (SGD) or the average on a subset of $\mathcal{X}$ in mini-batch gradient descent. The chosen data are then changed from one iteration to another to ensure seeing the full dataset during the optimization process. In deep learning (DL) training, the observation of the full datasets after iterating on all batches is called an "epoch". The disadvantage of this method is that the value tends to fluctuate much more than when using the full dataset (based on estimation theory, using N independent examples results in dividing the variance of the estimator by N, therefore, the larger the batches are, the smaller the variance of the estimator will be). However, less stability might also be helpful to avoid getting stuck in a local minimum. There is then a consensus to find between large batches that imply slow iterations and a risk of staying in a local minimum, and small batches, that reduce stability.

Some more advanced methods propose extension of gradient descent:

Momentum [172] The descent direction d_n is computed based on the gradient but also on the previous direction:

$$d_n = \nabla\mathcal{L}(\mathbf{p}^{(n)}) + \eta d_{n-1} \tag{B.5}$$

The idea is to avoid changing direction too often.

RMSprop [88] This algorithm proposes a way to estimate the optimal step γ based on an estimate of the expectation of the square of the gradient $\widehat{m_{\nabla^2}}^{(n)}$. The estimate is computed as:

$$\widehat{\mathbf{m}_{\nabla^2}}^{(n)} = 0.9\widehat{\mathbf{m}_{\nabla^2}}^{(n-1)} + 0.1\nabla\mathcal{L}(\mathbf{p}^{(n)}) \odot \nabla\mathcal{L}(\mathbf{p}^{(n)}) \tag{B.6}$$

The value of γ is then obtained as:

$$\frac{\eta}{\sqrt{\widehat{\mathbf{m}_{\nabla^2}}^{(n)}}} \tag{B.7}$$

with η a fixed learning rate. To avoid issues with small values, the computation of γ is often done as:

$$\frac{\eta}{\sqrt{\widehat{m_{\nabla^2}}^{(n)} + \epsilon}} \tag{B.8}$$

with ϵ a small fixed value. The descent direction is then obtained with an element-wise multiplication of the learning rate with the gradient.

Adam [87] (ADAptive Moment estimation) Without entering into details of the algorithm that can be found in the original paper, the idea is to propose an adaptive step based on an estimate of the element-wise variance of the gradient and a memory pattern in the direction like for the Momentum.

Other algorithms Adadelta [173], Adagrad [174], Adamax [87], Nadam [175], AMSGrad [176], ... More of them (as well as more details on the ones already mentioned) can be found in [177].

Another class of methods, called "Trust-region algorithms" can be used for performing optimization. These methods are generally harder to implement than gradient descent based methods, but can result in more robust results. At the $n+1$-th iteration of the process, a region is defined around the current estimate of the parameters $\hat{\mathbf{p}}^{(n)}$. A quadratic approximation of the function (based on its gradient and an approximate of its Hessian) is built within this region and the new estimate $\hat{\mathbf{p}}^{(n+1)}$ is defined as the one minimizing the approximation of the function in the region. The region is generally built as a ball around $\hat{\mathbf{p}}^{(n)}$, with a specific radius that is updated at each iteration depending on how well the function is approximated. As for gradient descent, trust region algorithms have originated many methods and applications. A good review can be found in [132, 133]. In [178], a review including some more recent methods is also proposed. Here, we only mentioned the quadratic approximation, but it would actually be possible to use other models for approximating the function.

B.2 Over-fitting

During the training of the model, it is sometimes possible that the model over-fits the training data, i.e. that it reduces the loss function by sticking to close to the training data without actually learning to solve the problem. Generally, this happens when the model has too many parameters compared to the difficulty of the problem or quantity or quality of training data. For example, if we try to approximate a function $f : x \to f(x)$, with a training set containing N observations $(x_0, f(x_0)), \ldots, (x_{N-1}, f(x_{N-1}))$, and by using a polynomial model of order N, then it is clearly possible to obtain extremely good results on the training data. For example, the Lagrange polynomials can provide an infinity of solution for which the approximation will match exactly all training points). However, it is unlikely that the obtained model is actually able to approximate the model outside of the training points. We give an example in Figure B.1. Despite making no error on training points, the Lagrange solution is clearly far from the target function outside of these points, while the simpler model $x :\to x^2$ manages to remain consistent everywhere.

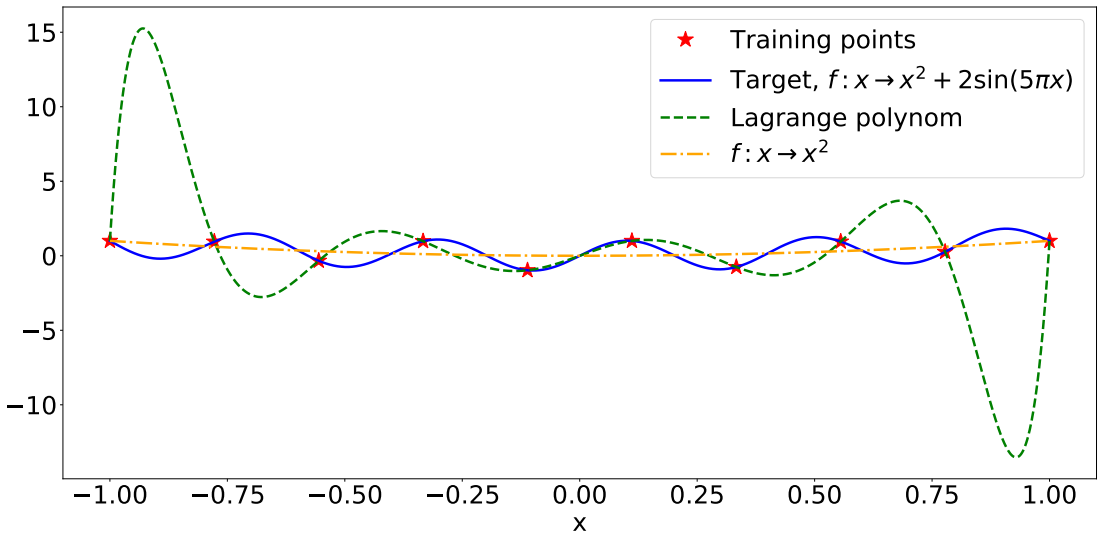


Figure B.1: Illustration of over-fitting

For some methods, it is possible to use criteria such as the Bayesian Information Criterion (BIC) or the Akaike Information Criterion (AIC) to decide which parameters are relevant for the problem (each parameters is considered as useful if the improvements obtained justify the increased complexity of the model). In methods such as Deep Learning, it is not possible to use such criteria. However, it is rare that such methods over-fit as soon as the first iteration (or "epoch") is performed. Instead, the models actually learn to solve the problem and start over-fitting when it is not able to improve significantly by actually solving the problem. One method to detect the moment when the model starts over-fitting is to use a second dataset during training, called the "validation set". This dataset is not used to perform gradient descent (or any method for updating the parameters) but an evaluation of the loss function is performed on this dataset at the end of each epoch. If the model is not over-fitting, then the value should be close to the one on the training set, and should at least be decreasing when the value on the training set is decreasing. When the model starts over-fitting, the loss function should keep decreasing on the training set, but should increase on the validation data. Therefore, the validation set allows to detect the moment when the model is the best for solving the problem. Several ways of using the validation data are possible, with two common ones:

Early-stopping The training is stopped if the validation score has not improved in the last K epochs, with K a predefined integer;

Model selection The training is performed till the end, then the model that has given the best validation score is selected.

Validation data might also be used to decide when it is necessary to adjust some parameters of the gradient descent algorithm. Generally, a large step size allows a fast convergence

to a point close to the optimum but tends to give large oscillations around this optimum afterwards, a small step would require more time to approach the optimum but is then able to avoid large oscillations. Therefore, it is conventional to start the training with a large step and reduce it progressively, and evolution of the score on validation data might be helpful to know when this change has to be done. When the over-fitting is too large, it is also generally possible to design a smaller model with similar results.

In our case, most trainings can be done in less than one day. Therefore, we will mostly use the validation for detecting the epoch when the best model was obtained.

B.3 Cross-validation

Machine Learning methods highly depend on several random phenomena. The initialization of the model is made with random selection of all parameters, the training data contain some randomness, the order of mini-batches in mini-batch gradient descent might change the results, ... Therefore, the parameters obtained for one model might change from one training to another. As a consequence, training the same model twice might result in very different results. By training only once, it is therefore not possible to say if the results obtained are representative of the general behavior of the model.

To overcome this problem, one solution is cross-validation. In cross-validation, the training set is divided into D equal sets. For each set, one model is trained by using all data outside the obtained set for training and the set is used for validation, i.e., detecting the best model or over-fitting. This results in the production of D different models trained with the same procedure but different data (and depending on the implementation, different initializations, ...). All models are then evaluated on the same evaluation set. Averaging the results of all models will then allow obtaining a more robust estimate of the general behavior of the model. It would also be possible to check the variance of the results.

C Model for the position and velocity

C.1 Mechanical model

For computing the speed, we first need to recall the mechanical system creating the signals. Figure C.1 shows the system with annotations. On this illustration, we represent R the radial of the eccentric, l the position of the plan, L the length of the connecting rod, v the velocity of the plan, v_{rot} the rotation speed of the eccentric, x and y are dimensionless values defining the position of the extremity of the connecting rod on the eccentric, with t the time.

We can observe that the velocity at the contact point is also the velocity of the plan v. Additionally, the velocity of the plan is directly obtained as $v = \frac{\mathrm{d}l}{\mathrm{d}t}$. As a consequence, the first step is to determine the value of l to within a constant.

We make three assumptions for performing this calculation:

1. The rotation speed of the eccentric v_{rot} does not depend on time. This is not exactly true in recordings. However all recordings have been performed with a constant setpoint rotation speed, so that the variations remain very small.

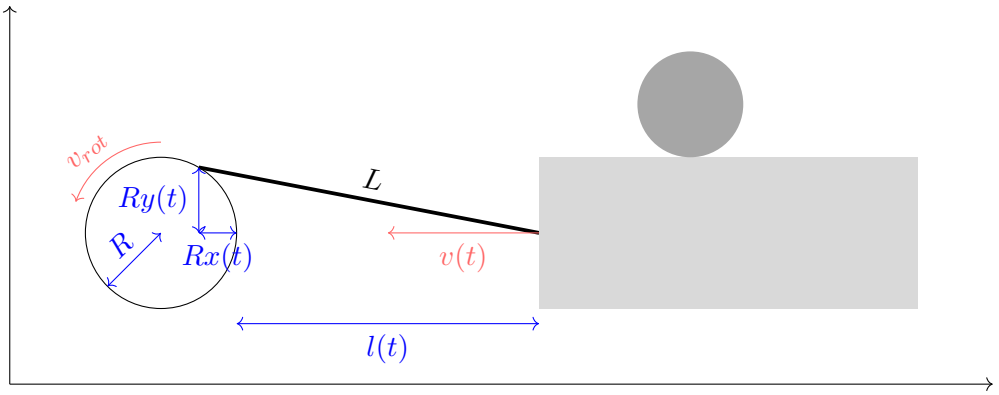


Figure C.1: Representation of the mechanical system, with annotations. Here, the motion is made only according to the abscissa. For this reason, we use the value $v(t)$ instead of the vector $v(t)(1,0)^T$.

2. At $t = 0$, the connecting rod is horizontal. This assumptions allows obtaining the analytical expressions of x and y:

$$\begin{cases} x = 1 - \cos(2\pi t v_{rot}) \\ y = \sin(2\pi t v_{rot}) \end{cases}. \tag{C.1}$$

   In this experiment, the time by itself has no influence, only the elapsed time is significant. Therefore, it is possible to define the time as being relative to an instant where the assumption was verified.

3. $L > 2R$, otherwise, there would be a contact between the plan and the eccentric when $x = 2$ and $y = 0$.

Using the Pythagorean theorem, we obtain:

$$L^2 = R^2 y(t)^2 + (Rx(t) + l(t))^2. \tag{C.2}$$

We put the analytical expressions of x and y into Equation (C.2) and develop the terms:

$$\begin{aligned} l(t)(l(t) + 2Rx(t)) &= L^2 - R^2(y(t)^2 + \cos(2\pi t v_{rot})^2 + 1 - 2\cos(2\pi t v_{rot})) \\ &= L^2 - 2R^2(1 - \cos(2\pi t v_{rot})) \end{aligned}. \tag{C.3}$$

It appears that $l(t)$ is the solution of:

$$u^2 + bu + c = 0 \tag{C.4}$$

with $b = 2R\,(1 - \cos(2\pi t v_{rot}))$ and $c = 2R^2(1 - \cos(2\pi t v_{rot})) - L^2$. The discriminant of this equation is:

$$\begin{aligned} \Delta &= 4\left[L^2 + R^2\left(\cos(2\pi t v_{rot})^2 - 1\right)\right] \\ &> 4(L^2 R^2) > 0 \end{aligned}. \tag{C.5}$$

As $l(t) > 0$, the only possible solution is:

$$l(t) = L\sqrt{1 + \frac{R^2}{L^2}\left(\cos(2\pi t v_{rot})^2 - 1\right)} - R\,(1 - \cos(2\pi t v_{rot})) \tag{C.6}$$

It follows:

$$v(t) = \frac{\mathrm{d}l(t)}{\mathrm{d}t} = -2\pi v_{rot}\frac{R^2}{L}\frac{\sin\left(2\pi t v_{rot}\right)\cos(2\pi t v_{rot})}{\sqrt{1+\frac{R^2}{L^2}\left(\cos(2\pi t v_{rot})^2-1\right)}} - 2\pi R v_{rot}\sin(2\pi t v_{rot}). \tag{C.7}$$

C.2 Simplification of the model

In the mechanism used by Ireis, the connecting rod has a length $L \approx 20cm$ and the eccentric has a radial $R \approx 5mm$. It is then possible to compute Taylor expansion of the position and velocity.

For the position, Equation (C.6) becomes:

$$l(t) = L\left(1 + o\left(\frac{R}{L}\right)\right) - R\left(1-\cos(2\pi t v_{rot})\right) \tag{C.8}$$

For the velocity, Equation (C.7) becomes:

$$\begin{aligned} v(t) &= -2\pi v_{rot}\frac{R^2}{L}\sin(2\pi t v_{rot})\cos(2\pi t v_{rot})\left(1+o\left(\frac{R}{L}\right)\right) - 2\pi R v_{rot}\sin(2\pi t v_{rot}) \\ &= -2\pi R v_{rot}\sin(2\pi t v_{rot}) + o\left(\frac{R}{L}\right) \end{aligned} \tag{C.9}$$

As a consequence, the position can be approximated as $l(t) \approx L - R + R\cos(2\pi t v_{rot})$ and the velocity can be approximated by simply deriving the obtained sine wave for the position, i.e.:

$$v(t) \approx -2\pi R v_{rot}\sin(2\pi t v_{rot}) \tag{C.10}$$

D

Detailed results for the experiments on ECG

This appendix gives the details of the experiment presented in Section 4.4.4.3. The results at 0 dB can be found in Table D.1. The results at 5 dB can be found in Table D.2. The results at 10 dB can be found in Table D.3. The results at 15 dB can be found in Table D.4. We also give again the average obtain by each model for a specific initial SNR, and we additionally provide the standard deviation.

As shown in these tables, the results are generally slightly in favor of periodic models. The advantage becomes easier to observe as the initial SNR increases, which might be because it becomes easier to retrieve the corresponding areas in several consecutive periods, whereas a high level of noise leads the models to focus mostly on the current period.

Additionally, it is possible to observe that PUNet_3 generally obtains the best results in terms of obtained SNR, but also has a lower standard deviation than other models (except UNet_3 for an initial SNR of 5 dB), proving its higher stability to different patients.

Table D.1: Obtained output SNR (in dB) for the signals at 0 dB

Patient	UNet_7	PUNet_7	UNet_4	PUNet_4	UNet_3	PUNet_3
107	11.82	11.92	11.82	11.79	12.11	**12.21**
112	11.55	12.00	11.94	13.15	12.28	**13.36**
117	14.20	14.34	13.85	**14.62**	13.62	13.81
119	14.30	**14.53**	14.06	14.51	13.78	14.09
203	12.06	**12.08**	11.94	12.01	12.02	11.88
207	**13.29**	12.92	13.21	13.06	13.12	12.75
210	13.06	**13.21**	13.00	13.19	13.07	13.06
212	11.33	11.73	11.36	11.67	11.35	**11.83**
213	11.58	11.77	11.37	12.42	11.90	**12.19**
219	12.82	**12.89**	12.59	12.74	12.56	12.88
Average	12.6	12.74	12.51	**12.92**	12.58	12.80
Standard deviation	1.04	0.98	0.93	0.98	0.75	0.75

Table D.2: Obtained output SNR (in dB) for the signals at 5 dB

Patient	$UNet_7$	$PUNet_7$	$UNet_4$	$PUNet_4$	$UNet_3$	$PUNet_3$
107	**15.88**	15.73	15.50	15.10	15.65	15.39
112	15.24	15.52	15.41	16.24	15.88	**16.51**
117	17.38	17.58	17.47	**17.93**	17.32	17.43
119	17.83	18.07	17.89	**18.29**	17.84	18.04
203	15.38	15.50	15.35	15.50	**15.59**	**15.59**
207	16.70	16.57	**16.84**	16.80	16.79	16.73
210	16.08	16.27	16.17	16.39	16.49	**16.53**
212	14.61	14.90	14.48	14.60	14.77	**14.99**
213	15.38	15.38	14.92	**15.62**	15.34	15.59
219	**16.68**	**16.68**	16.45	16.35	16.47	16.56
Average	16.12	16.22	16.05	16.28	16.21	**16.34**
Standard deviation	0.97	0.96	1.06	1.11	0.89	0.91

Table D.3: Obtained output SNR (in dB) for the signals at 10 dB

Patient	$UNet_7$	$PUNet_7$	$UNet_4$	$PUNet_4$	$UNet_3$	$PUNet_3$
107	**19.43**	19.21	19.01	18.55	19.00	18.95
112	18.72	18.91	18.79	19.27	19.21	**19.59**
117	20.05	20.38	20.24	**20.59**	20.29	20.45
119	21.13	21.35	21.11	**21.58**	21.55	21.53
203	18.25	18.43	18.48	18.66	18.79	**18.93**
207	19.51	19.52	19.98	19.96	20.01	**20.11**
210	18.83	18.98	19.21	19.27	19.50	**19.66**
212	17.43	17.68	17.39	17.58	17.78	**17.95**
213	**19.28**	19.27	18.61	18.93	19.01	19.27
219	19.85	19.83	19.92	19.68	19.93	**20.07**
Average	19.25	19.36	19.27	19.41	19.51	**19.65**
Standard deviation	0.97	0.96	1.01	1.07	0.96	0.93

Table D.4: Obtained output SNR (in dB) for the signals at 15 dB

Patient	$UNet_7$	$PUNet_7$	$UNet_4$	$PUNet_4$	$UNet_3$	$PUNet_3$
107	22.07	21.95	22.05	21.62	22.06	**22.57**
112	21.31	21.54	21.82	22.10	22.18	**22.55**
117	22.12	22.59	22.73	22.92	22.80	**23.24**
119	23.96	24.08	24.12	24.41	**24.84**	24.64
203	20.21	20.53	21.17	21.31	21.47	**22.07**
207	21.51	21.69	22.57	22.60	22.73	**23.17**
210	20.71	21.05	21.77	21.70	21.84	**22.40**
212	19.24	19.60	19.88	20.14	20.27	**20.99**
213	22.46	22.45	21.91	22.29	22.57	**22.91**
219	22.15	22.14	22.81	22.44	22.87	**23.35**
Average	21.57	21.76	22.08	22.15	22.36	**22.79**
Standard deviation	1.24	1.16	1.06	1.06	1.11	0.90

List of Figures

List of Tables

Abstract (english)

The task of denoising has been widely studied in the last decades. In the last years, conventional methods have been replaced in many situations by deep learning based methods. Deep neural networks have obtained large improvements in various situations. However, whereas some conventional methods, such as non-local means, exploited the periodicity to improve performance, no difference is made between periodic and non-periodic signals in deep learning based methods, and it is likely that exploiting the periodicity might help getting new improvements with these methods. In this thesis, after proposing first approaches for performing the denoising the signals with a model fitting approach, we propose to train a 1D fully convolutional denoiser by generating synthetic data and using a regularization in the loss term. Afterwards, we extend this work to exploit the periodicity. Our method for exploiting the periodicity is based on a reshaping of the 1D signals into a periodic grid, where each line contains one period of the original signal. Then, we propose one method for exploiting this new shape by performing a preprocessing before reshaping the grid back into a 1D signal and feeding it to a 1D fully convolutional denoiser. This method could easily be applied to almost any denoiser, and not only to fully convolutional neural networks. We also propose to replace some of the 1D kernels of the fully convolutional denoiser into 2D kernels and feed the grid directly to the neural network, which results in an improved robustness to fluctuations of the periodicity compared to the first method. This second method is however harder to generalize to other kinds of denoisers. We show that these two methods are getting significant improvements compared to the 1D models in theoretical cases with exactly periodic signals and known fundamental frequencies. In realistic cases when real data are available for training the proposed method is also able to outperform 1D models, even though the difference remains small for cases with large variations of periodicity. However, generalization to real data when no real data is available for training remains an issue. We propose to combine the outputs of some models exploiting the periodicity with the ones of some models which do not exploit the periodicity by feeding all of these outputs to a postprocessing neural network. We show the improvements obtained by this method in terms of robustness to various parameters, as well as its current limitations.

Abstract (french)

La tâche de débruitage a été largement étudiée au cours des dernières décennies. Ces dernières années, les méthodes conventionnelles ont été remplacées dans de nombreuses situations par des méthodes basées sur l'apprentissage profond. Les réseaux de neurones profonds ont obtenu des améliorations significatives dans diverses situations. Cependant, alors que certaines méthodes conventionnelles, comme les moyennes non-locales, exploitent la périodicité pour améliorer les performances, aucune différence n'est faite entre les signaux périodiques et non périodiques dans les méthodes basées sur l'apprentissage profond, et il est probable que l'exploitation de la périodicité pourrait aider à l'obtention de nouvelles améliorations avec ces méthodes. Dans cette thèse, après avoir proposé des premières approches pour effectuer le débruitage des signaux avec une approche d'ajustement de modèle, nous proposons d'entraîner un réseau 1D entièrement convolutif en générant des données synthétiques et en utilisant une régularisation dans la fonction de coût. Nous proposons ensuite une extension de ce travail pour exploiter la périodicité. L'exploitation de la périodicité repose dans la méthode proposée sur un remodelage des signaux 1D en une grille périodique, où chaque ligne contient une période du signal original. Nous proposons ensuite une méthode pour exploiter cette nouvelle grille en effectuant un prétraitement avant de retransformer la grille en un signal 1D et de la donner en entrée d'un réseau convolutif 1D. Cette première méthode pourrait facilement être généralisée à presque toutes les méthodes de débruitage existantes, et pas seulement aux réseaux convolutifs. Nous proposons également de remplacer certains des noyaux 1D du réseau par des noyaux 2D et la grille est directement donnée en entrée du réseau, ce qui permet d'améliorer la robustesse aux variations de la périodicité par rapport à la première méthode. Cette deuxième méthode est cependant plus difficile à généraliser à d'autres méthodes de débruitage. Nous montrons que ces deux méthodes obtiennent des améliorations significatives par rapport aux modèles 1D dans des cas théoriques avec des signaux exactement périodiques et des fréquences fondamentales connues. Dans des cas réalistes où des données réelles sont disponibles pour l'entraînement, cette méthode permet aussi une amélioration, bien que celle-ci soit plus légère, en particulier avec de fortes variations de périodicité. Cependant, la généralisation aux données réelles lorsqu'aucune donnée réelle n'est disponible pour l'entraînement reste un problème. Nous proposons de combiner les sorties de certains modèles exploitant la périodicité avec celles de certains modèles qui n'exploitent pas la périodicité en fournissant toutes ces sorties à un réseau de post-traitement. Nous montrons les améliorations obtenues par cette méthode en termes de robustesse à divers paramètres, ainsi que ses limites actuelles.