Grenoble INP

Deep Learning-based Speech Enhancement

Auteurs :
Paul Aimé
Antoine Bertrand

Encadrants :
M. Laurent Girin
M. Thomas Hueber

16 janvier 2020

Table des matières

1 Introduction

La présence d'un bruit de fond sur des enregistrements de parole peut sérieusement dégrader la qualité du signal d'intérêt, jusqu'à le rendre inintelligible au milieu du bruit. Alors que l'être humain arrive à extraire et interpréter le signal d'intérêt jusqu'à des conditions de bruit avancées, ce n'est pas encore le cas des systèmes automatiques de recognition de la parole pour qui la présence de bruit est une cause majeure d'échec.

Ainsi, l'objectif des techniques de *Speech Enhancement* est de réduire les composantes de bruit et donc d'améliorer le rapport signal/bruit (RSB) et la qualité perceptive.

Les capacités offertes par les outils du *Deep Learning* pour résoudre cette tâche de *Speech Enhancement* sont particulièrement étudiés ces dernières années [3, 14].

Ici sera juste implémenté un modèle précis choisi parmis la variété des possibles (celui présenté dans [9]), cela afin de se familiariser avec la manipulation du signal audio d'une part et les outils du *Deep Learning* d'autre part. Dans ce rapport, les manipulations réalisées seront donc décrites et interprétées avec attention, et les résultats obtenus seront commentés avec précaution au vu de l'amplitude relativement modeste des tests effectués, en accord avec l'ambition du projet.

2 Jeu de données

2.1 Jeu de parole

Le jeu de données de signaux de parole utilisé est le *TIMIT Speech Corpus* [1]. Il a été conçu par l'effort conjoint de Texas Instruments (TI) et du Massachusetts Institute of Technology (MIT). Il contient des enregistrements sonores de 630 locuteurs des huit dialectes majeurs de l'anglais américain, chacun lisant dix phrases choisies pour être phonétiquement riches. Pour chaque enregistrement, ce corpus comprend également une transcription phonétique, ortographique et mot à mot, alignée dans le temps avec le son. Ces dernières données ne seront pas utilisées ici, où seul le signal audio sera exploité.

Les données sont livrées séparées en une base d'entraînement et une base de test (*Test Set*). Cette dernière est maintenue à l'écart, et servira seulement pour évaluer les performances finales du modèle entraîné. La base d'entraînement est ensuite scindée en une base d'apprentissage et une base de validation, cette dernière ayant pour but de permettre un contrôle de l'évolution de l'apprentissage. Cette répartition est réalisée à un ratio de (87.8%, 12.2%), et la correspondance en nombre d'enregistrements par base est visible sur la table 1. La séparation a pu être réalisée en terme de nombre d'enregistrements, car la durée des enregistrements est courte (3s) et assez homogène – cette propriété du jeu de données sera réutilisée par la suite (cf. section 4.1).

	Train Set	Validation Set	Test Set
Nb d'enregistrements	4056	564	1680

TABLE 1 – Répartition des enregistrements de parole dans les différents jeux de données.

2.2 Jeu de bruit

Le bruit utilisé est un *babble noise*. Il simule le bruit ambiant dans une salle avec une dizaine de personnes parlant en même temps. L'idée d'utiliser un tel bruit, de même nature que le signal d'intérêt, est de se placer dans une situation compliquée pour le débruitage, puisque la distribution du signal d'intérêt peut alors être supposée en situation de recouvrement avec la distribution du signal de bruit.

Ce fichier de bruit dure 3min55. Or, il doit être utilisé pour bruiter les 3 différents jeux de données de manière indépendante. Pour ce faire le fichier est donc scindé en trois parties, selon le découpage décrit dans la table 2. Le ratio entre la taille de la base d'apprentissage et celle de la base validation est choisi identique à celui correspondant pour la séparation réalisée sur le jeu de parole.

Même si les différents jeux de données parole sont associés à des parties différentes du signal de bruit, on peut tout de même convenir que la distribution du bruit y est quasiment identique entre ces différentes

	babble_train.wav	babble_val.wav	babble_test.wav
Tronçon du fichier d'origine	0 :00 - 3 :00	3 :00 - 3 :25	3 :25 - 3 :55
Durée	180s	25s	30s

TABLE 2 – Spécificité découpage du fichier de bruit

parties. La tâche de débruitage apprise sur le jeu d'entraînement sera donc très proche de celle demandée lors du test, de bons résultats devraient alors pouvoir être obtenus sans qu'il puisse en être conclu que le modèle est robuste.

Toutefois, le choix du *babble noise*, de distribution proche de celle du signal, devrait constituer la version la plus compliquée de la tâche de débruitage d'un seul bruit spécifique.

Deux développements intéressants auraient pu être fait ici. Soit d'augmenter la diversité du babble noise de manière synthétique, comme dans [4]. Soit de prendre une plus grande variété de nature de bruit, voire à les mixer comme dans [6], et surtout d'étudier ensuite la réponse du modèle à un bruit encore jamais vu.

2.3 Mixage

Les données brutes ne sont pas bruitées de manière fixe. À chaque nouvel entraînement le bruit est ajouté artificiellement en ligne aux signaux de parole à un niveau de RSB donné. Cela permettrait par exemple un apprentissage sur une base à RSB variable dans le temps, mais cette étude n'a pas été réalisée ici.

Pour ce faire, un segment de bruit de la même durée que l'enregistrement est découpé de manière aléatoire dans le fichier de bruit de la base correspondante, puis mis à l'échelle de manière appropriée pour atteindre le niveau de RSB souhaité une fois l'ajout au signal de parole effectué.

2.4 Pré-traitement

Pour diminuer le nombre de données à traiter, et ainsi accélérer l'apprentissage, la fréquence d'échantillonnage des signaux audio ont été réduites à 8 kHz, comme réalisé dans [9].

L'apprentissage du réseau de neurone sera réalisé sur le module de la STFT (*Short-Time Fourier Transform*) du signal, et non sur les échantillons temporels de celui-ci. En effet, ce domaine de représentation est reconnu comme pertinent et grandement utilisé pour l'analyse de la parole [11] ; l'apprentissage devrait alors en être facilité.

Pour calculer la STFT d'un signal, on utilise la libraire librosa [8]. La fenêtre utilisée est une fenêtre de hanning de taille $nfft = 256$ (32 ms) générée avec la fonction correspondante de la librairie PyTorch [10]. Cette taille est adapté à l'évolution de la quasi-stationnarité d'un signal de parole moyen. Le recouvrement entre deux fenêtres succesives est de 50%. Pour garder la taille du signal on utilise un *padding* de type `'reflect'`. Pour un signal de taille L, la STFT retournée est une matrice de taille (H, N), où $H = (nfft/2) + 1 = 129$ et $N = \frac{L}{hop\_length} = \frac{L}{overlap \times nfft}$.

De manière classique les entrées d'un réseau de neurones sont normalisées pour qu'elles soient centrées avec une variance unitaire. Ici les valeurs des STFTs ont été normalisées entre -1 et +1, afin de pouvoir interpréter la valeur de la fonction de coût plus facilement. Dans la section 5.1 les bénéfices de cette normalisation seront remis en cause, au regard des effets produits sur la prédiction/reconstruction et l'évaluation du modèle.

Un exemple de STFT des signaux d'origines et bruités est visible dans la figure 3.

3 Modèle

3.1 Encodeur-Décodeur Convolutionnel

Le modèle utilisé est une implémentation du modèle R-CED (10 Conv) présenté dans [9], à l'exception des *skip connections*. C'est un réseau de neurone entièrement convolutionnel basé sur une architecture d'*autoencoder* dite redondante, puisque la taille des *feature maps* est constante d'une couche interne à l'autre (cf. figure 1).

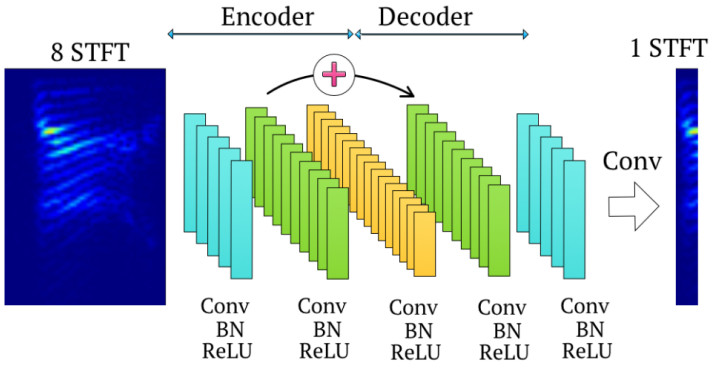


FIGURE 1 – Architecture du réseau R-CED (Redundant Convolutionnal Encoder-Decoder), proposé dans [9]. Figure issue de l'article correspondant.

Un modèle convolutionnel est pertinent pour l'approche prise pour la tâche de débruitage considérer, puisqu'il a été choisi de traviller avec les STFTs comme entrée. En effet les STFTs peuvent être considérées des images, et que les réseaux convolutionnels sont particulièrement adapté au traitement d'image parce qu'ils exploitent l'idée de champs récepteurs locaux [7]. De plus ils ont été démontrés efficaces dans la tâche particulière du débruitage d'image [4], qui est donc la tâche demandé au réseau étudié ici.

3.2 Convolution 1D en fréquence

Mis à part de ne pas changer la taille des *features maps*, la particularité du modèle R-CED est d'utiliser une convolution 1D, seulement le long de l'axe des fréquences ; choix qui a été fait empiriquement sur la base des performances obtenues.

Pour ce faire, chaque couche du réseau, toutes équivalentes, ont un noyau de convolution de largeur égale à celle de l'entrée qu'ils traitent, et aucun padding n'est réalisé dans cette dimension (la dimension temporelle). La sortie d'une telle couche est donc 1D, uniquement sur l'axe des fréquences. En particulier, la largeur du noyau de la première couche est la largeur de la section de STFT considérée en entrée du réseau, et la largeur de toutes les couches suivantes et de 1, car elles ont pour entrée la sortie des couches précédentes. Un padding est réalisé sur l'axe des fréquences afin de conserver la même taille de *feature maps* tout au long du réseau.

Chaque couche est également composée d'une opération de *Batch Normalization* (BN) ainsi que de la fonction d'activation ReLU, appliquée dans cet ordre après la convolution. L'ordre entre ces deux dernières opérations est ici toutefois inversé par rapport à l'implémentation de [9], qui est elle en accord avec l'ordre qui amène généralement la meilleure performance (cf. [2]).

3.3 Paramètres d'architecture

Le réseau étant basé sur une architecture de type *autoencoder* (cf. figure 1), il est symétrique, et son milieu, parfois appelé esapce latent, contient théoriquement la représentation des données la plus pertinente à l'égard de la tâche considérée. Les couches précédent l'espace latent forment un sous-réseau appelé encodeur, et les couches suivant cet espace forment un sous-réseau appelé décodeur.

Ici le nombre de couches internes est de 9 (4-1-4). Au long de l'encodage, le nombre de *feature maps* est croissant et la taille des noyaux est décroissante, et la situation est symétrique au décodage. C'est cette évolution qui permet à l'espace latent d'acquérir ses bonnes propriétés de représentation de la STFT, en amenant le réseau à prendre une approche d'analyse globale vers analyse particulière de la STFT, en extrayant ainsi les caractéristiques importante, éliminant ainsi *à priori* le bruit. Une dernière couche, de même nature, est ajoutée en dernier, elle possède un noyau de hauteur égale à celle de son entrée et peut donc être vue comme une couche de classification.

Les valeurs précises des paramètres des couches sont renseignées dans le tableau 3. Le nombre totale de paramètres d'apprentissage d'un tel modèle est de 32611 pour $(H, W) = (129, 7)$. C'est un petit nombre au regarde de la taille du réseau, qui est un autre avantage que présentent les réseaux à convolution au

	Encoder				Latent	Decoder				Out
Layer	1	2	3	4	5	6	7	8	9	10
(in, out)	(1, 12)	(12, 16)	(16, 20)	(20, 24)	(24, 32)	(32, 24)	(24, 20)	(20, 16)	(16, 12)	(16, 12)
kernel size	(13, W=7)	(11, 1)	(9, 1)	(7, 1)	(7, 1)	(7, 1)	(9, 1)	(11, 1)	(13, 1)	(H=129, 1)

TABLE 3 – Paramètres de *feature maps* et de taille du noyau de convolution pour chaque couche du modèle. W et H sont respectivement la largeur et la hauteur de l'entrée du réseau.

regard d'un réseau *fully-connected*, et qui repose sur leur propriété de partage des poids, qui correspond à l'application identique de chaque noyau sur l'ensemble de l'image.

La dimension de sortie d'un tel réseau de neurone est alors de $(H, 1)$, c'est à dire qu'à partir de W frames de la STFT en entrée, le signal en reconstruit une (une frame correspondant à la transformée de Fourier sur une fenêtre d'analyse de taille $nfft$). L'idée est de pouvoir ainsi permettre au réseau d'utiliser l'information de l'avolution temporelle du signal. Le traitement étant réalisé hors-ligne, il est donc possible de centrer ces W frames afin que la sortie désirée soit la version non bruitée de celle du milieu (à la différence de l'implémentation causale réalisée dans [9]).

Un bloc de $W = 7$ frames a été choisi en entrée. Une frame a une durée de $nfft/fs = 32ms$, et l'overlap est de 50%, soit 16 ms. La durée d'un bloc est donc de $32 + 6 \times 16 = 128ms$, soit légèrement supérieur à la durée de 88 ms utilisée dans [9], justifiée comme étant semblable à la durée moyenne d'une voyelle.

4 Apprentissage

4.1 Procédure

Puisque la distribution des durées des sons du jeux de données utilisés est assez homogène (cf. section 2.1), on utilise l'ensemble de la STFT d'un son pour constituer un batch d'apprentissage, qui seront alors a peu près de taille équivalente. Comme dit dans la section 2.4, les STFTs ont été normalisées à l'échelle du son, cette normalisation représente alors en quelque sorte une couche de *Batch Normalization*.

Une entrée unique étant de dimension $(H = 129, W = 7)$ (cf. section 3.3), en choisissant un saut entre les entrées de 1 frame et un padding de taille $(W - 1)/2$ en début et fin du signal, un batch est alors une matrice de dimension (N, H, W) où N est la taille du son (environ 187 pour un son d'une durée de 3s par exemple). Sa sortie est alors de dimension $(N, H, 1)$, ce qui, en réduisant la dernière dimension, est égale à la dimension de la STFT de tout l'ensemble du son.

Lors de l'apprentissage, les sons sont présentées dans un ordre aléatoire pour éviter toute tendance dûe à une séquence particulière dans l'exploration des données.

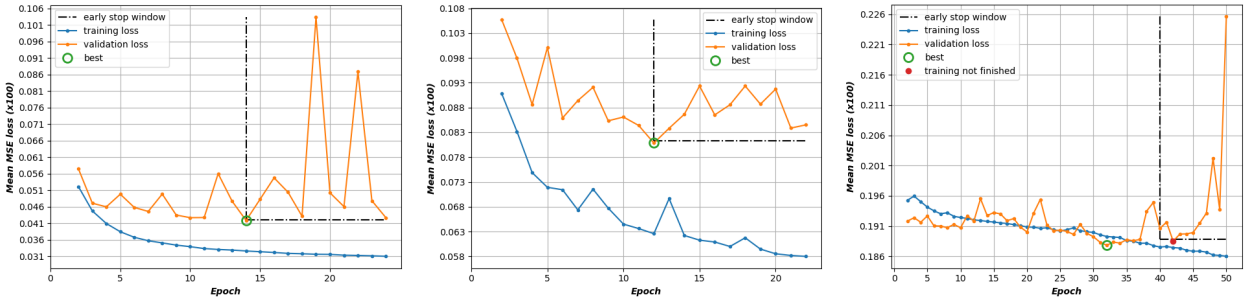
Le modèle a été entraîné avec l'optimiseur Adam [5] avec les paramètres par défaut de son implémentation PyTorch (cf. `torch.optim.Adam` sur la documentation PyTorch).

4.2 Évaluation de l'apprentissage

Différents modèles ont été entraînés sur le même jeu de données à différents niveau de bruit en RSB dB : -20, -5, 0.

La fonction de coût utilisée est l'erreur quadratique moyenne. Les STFTs étant normalisées entre -1 et 1, l'erreur maximale est de $-2^2 = 4$, ce qui permet d'interpréter les valeurs de la fonction de coût évaluée sur les jeux d'entraînement et de validation.

Un critère d'arrêt de l'entraînement de type *early-stopping* a été implémenté avec 0.5% du coût de validation pour paramètre de marge et 10 époques d'apprentissage pour paramètre de patience (cf. représentation par fenêtre en pointillés noirs sur la figure 2). La validation est réalisée à chaque époque. Un nombre maximum d'époques a été fixé à 50.



(a) RSB = 0 dB (b) RSB = -5 dB (c) RSB = -20 dB

FIGURE 2 – Évolution des erreurs d'entraînement (en bleu) et de validation (en orange), pour des modèles entraînés sur des jeux de données à différents niveau de RSB. En pointillés noir est représentée la fenêtre d'*early-stopping*. En cercle vert est représenté le point du modèle ayant la meilleur erreur de validation. Un point rouge signifie un entraînement non fini (existence d'un modèle ayant une erreur de validation en dehors de la fenêtre d'*early-stopping*). La première époque n'est pas représentée car ayant une erreur beaucoup plus grande que les autres époques elle écrase toute la dynamique du graphique.

La figure 2 présente l'évolution des erreurs d'entraînement et de validation. De manière attendue, on observe que la performance du modèle est d'autant moindre que sa base d'entraînement est bruitée, avec pour les meilleurs modèles respectifs une erreur 0.00042 pour un RSB de 0 dB, de 0.00081 pour un RSB de -5 dB, et de 0.00188 pour un RSB de -20 dB. Dans tous les cas ce sont des valeurs très faibles au regard de l'erreur maximal possible.

Le bruit d'apprentissage semble relativement élevé, cependant le jeu de données présentant peu de diversité (cf. section 2.2) ce ne doit pas en être la cause, tout du moins pas de la manière classique. La variation observée semble plutôt être de faible amplitude autour du point optimal, mais est perçue grande car la convergence est chaque fois quasiment atteinte dès l'époque 2, et que l'époque 1 n'est pas représentée sur la figure, justement à cause du fait que sa valeur est beaucoup plus élevée et écrase la dynamique d'affichage.

Il est alors plausible que le modèle soit "surdimensionné" au vu de la complexité de la tâche demandée. Cela semble soutenu par les valeurs de coût très faibles, et expliquerai par ailleurs le fait que la variation est d'autant plus grande lorsque les modèles sont entraînés sur des jeux de données moins bruités.

Le possible surdimensionnement avancé ne serait pas tant en terme de nombre de paramètres libres – qui reste faible aux égards d'autre modèles possibles (cf. section 3.3) du fait du choix d'un modèle convolutionnel –, mais plutôt dû au fait que l'architecture du modèle est particulièrement bien adapté au problème, ce qu'il est raisonnable de penser puisque elle a été proposé comme telle dans un travail de recherche récent [9].

On note toutefois que le seul modèle qui semble rentrer en situation de sur-apprentissage est le modèle entraîné sur la base la plus bruitée (-20 dB), ce qui est commun lors d'une tâche d'apprentissage automatique.

4.3 Détails

L'entraînement a été réalisé sur une machine virtuelle `n1-highmem-8` du *Compute Engine* de *Google Cloud Platform* possédant 8 vCPU et 1 GPU NVIDIA Tesla P4. La durée de traitement d'une époque est de 6 minutes 30 dont 15 secondes pour la validation (les tailles des jeux de données sont indiquées dans la table 2). Pour l'apprentissage d'un modèle arrivant jusqu'à la condition d'époque maximale, cela représente donc un apprentissage d'environ 6 heures.

L'implémentation a été réalisée avec la libraire PyTorch [10] et est disponible sur GitHub.

5 Prédiction

5.1 Reconstruction du signal audio

L'objectif applicatif est de débruiter le signal audio. Le débruitage du module de la STFT n'est que la méthode utilisée pour atteindre cet objectif. Il reste alors à reconstruire le signal audio à partir de la STFT débruitée. Pour cela il faut donc réaliser l'opération inverse de la STFT, ou ISTFT.

Cependant, la STFT est à valeur complexe, mais le réseau n'a été entraîné qu'à débruiter son module. Il faut donc définir quelle phase prendre pour le calcul de la ISTFT (réalisé ici avec la fonction correspondante de la libraire Librosa [8]). La seule phase qu'on peut supposer disponible pour le modèle est celle du signal bruité, c'est donc celle-ci qui sera utilisée dans l'implémentation réalisée.

On note toutefois qu'il existe des méthodes de reconstruction du signal à partir du seul module de sa STFT ([15, 12, 13]), et qu'il aurait pu être judicieux de les utiliser ici.

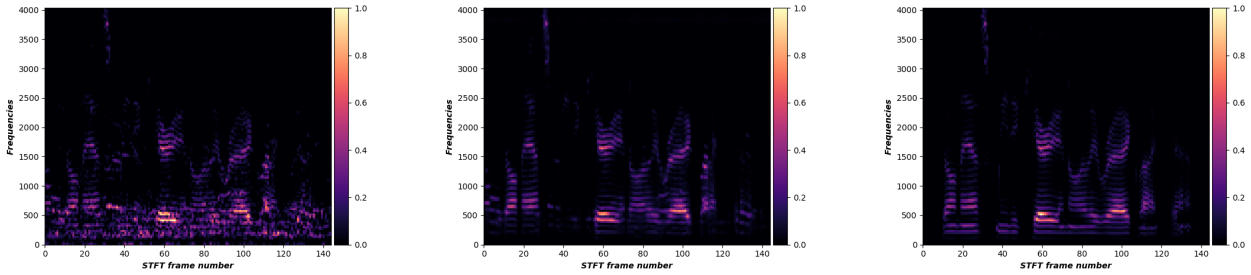
Notes sur la normalisation
La reconstruction du signal est rendue mauvaise par les normalisation succesives appliquées lors du traitement des signaux et leurs STFTs. En effet, appliquer la fonction `istft` de la librairie librosa [8] directement sur la sortie de la fonction `stft` appliquée à un signal x donne un RSB de environ 140 dB entre le x d'origine et celui reconstruit, théoriquement identique. Mais si la STFT est normalisée avant de reconstruire le signal, alos ce RSB tombe alors à 3 dB.

5.2 Résultats

Dans cette partie, nous évaluerons la performance de notre modèle sur la tâche de débruitage. Pour cela, nous étudierons les STFT prédites (*cf. figure 3*) ainsi que le signal sonore que l'on reconstruit à partir de ces derniers. Pour se faire, nous introduirons différentes métriques objectives mais également subjectives nous permettant de quantifier la qualité du débruitage.

5.2.1 Mesures objectives

A. Qualité du débruitage estimée sur les spectrogrammmes
La partie débruitage du traitement se fait sur la STFT des signaux. En utilisant des métriques classiques de comparaison d'images on peut donc estimer la qualité du débruitage.



(a) STFT du signal bruité (b) STFT débruitée prédite (c) STFT d'origine

FIGURE 3 – Visualisation des spectrogrammes d'un signal avant, pendant, et après débruitage.
(SNR d'entrée = 0dB)

Sur la figure 3, les spectrogrammes d'un signal brut, bruité et débruité sont visibles. On peut remarquer que le traitement semble fonctionner : les parties de bruit semblent en effet très fortement atténuées. On peut toutefois noter que certaines parties du signal de parole ont été retirées avec le bruit.

La visualisation de la figure 4, permet facilement reconnaitre les parties du spectrogramme du signal débruité qui correspondent au signal utile (en blanc), celles qui sont manquantes (en rose), et celles qui proviennent du bruit et n'ont pas été retirées (en vert).

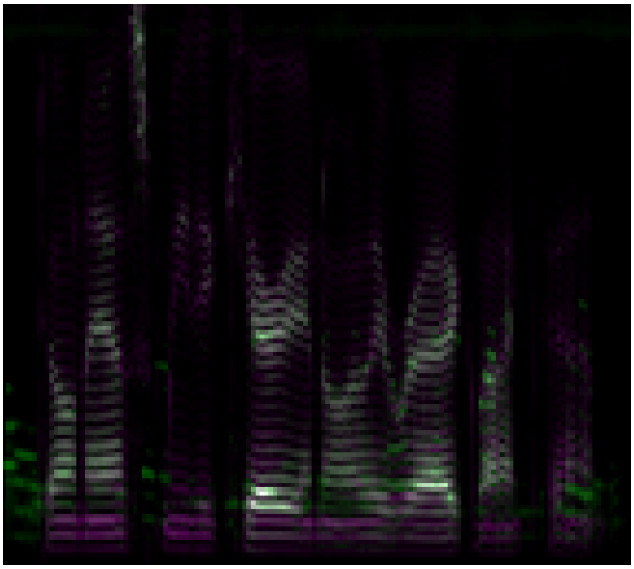


FIGURE 4 – Visualisation des différences entre les spectrogrammes du signal brut le spectrogramme débruité prédit.

B. Le Rapport Signal sur Bruit (SNR)

La métrique la plus simple pour estimer objectivement la qualité d'un signal est le rapport signal/bruit (SNR) qui compare les signaux de parole originaux et traités, échantillon par échantillon. Il est calculé comme suit pour des signaux centrés :

$$SNR_{dB}(x,y) = 10log_{10}\frac{\sum_{i=1}^{N} x(i))^2}{\sum_{i=1}^{N}(x(i)-y(i))^2}$$

où $x(i)$ et $y(i)$ sont les échantillons de parole originaux et traités indexés par i et N est le nombre total d'échantillons.

Nous avons calculé le SNR sur tous les sons du jeu de test et extrait différentes statistiques pour différents niveaux de bruit. (cf table 4).

Input SNR (clean vs noisy)	**-20dB**	**-10dB**	**-5dB**	**0dB**
Mean SNR (clean vs pred)	1.859163	1.859163	1.859163	1.859163
STD SNR (clean vs pred)	0.734523	0.734523	0.734523	0.734523

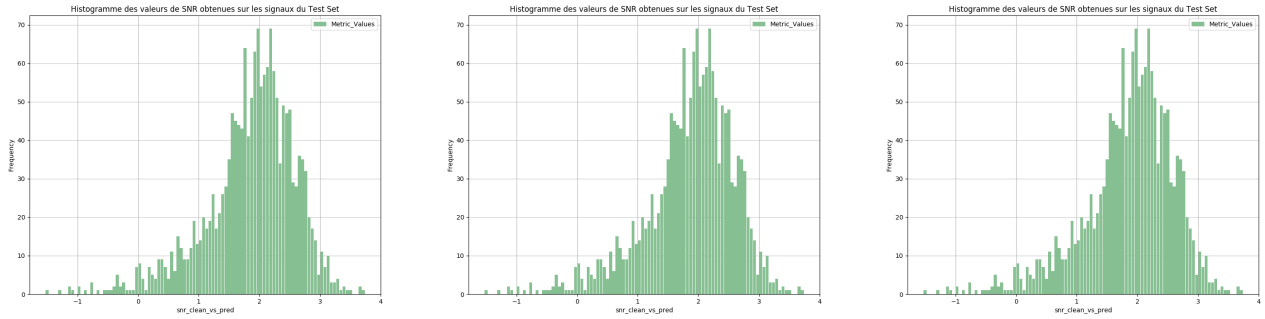
TABLE 4 – Statistiques calculées sur les valeurs de SNR des sons du jeu de test.

On remarque que quelque soit le SNR choisi lors de l'ajout du bruit aux données brutes, les données statistiques sont les mêmes. En s'y penchant de plus près, on peut également noter que le SNR prend la même valeur pour un son, quelque soit le niveau de bruit. C'est une observation auquel on ne s'attendait absolument pas. Une manière de l'interpréter serait de dire que le modèle parvient à complètement éliminer le bruit quelque soit le bruit ajouté en entrée, il enlève cependant en le faisant également de l'information utile. Le signal après traitement est donc le même quelque soit le niveau de bruit.

Cependant, une autre observation nous fait pencher plutôt sur l'hypothèse d'une erreur de notre part. En effet, il s'avère que les STFT des signaux débruités diffèrent bien suivant le niveau de bruit (cf. figure 6), alors que le signal reconstruit à partir de ces STFT est lui le même. C'est donc difficilement explicable, et provient donc plus certainement d'une erreur de code.

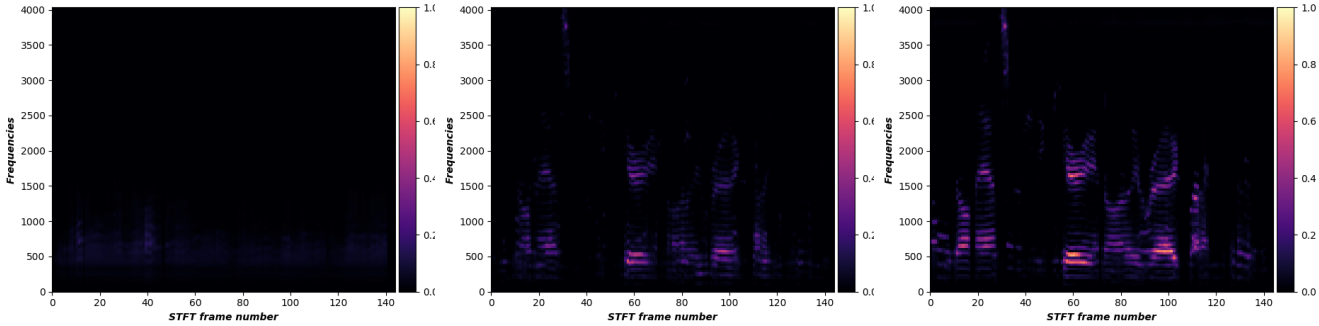
C. Le Rapport Signal sur Bruit Segmenté (segSNR)

Le rapport signal sur bruit segmenté (segSNR), ne travaille pas sur le signal entier, mais calcule la moyenne des valeurs de SNR sur des segments courts (15 à 20 ms). Il est calculé suivant l'expression suivante :



(a) Input SNR = -20dB (b) Input SNR = -5dB (c) Input SNR = 0dB

FIGURE 5 – Histogrammes des valeurs de SNR à différents niveaux de bruit d'entrée



(a) Input SNR = -20dB (b) Input SNR = -5dB (c) Input SNR = 0dB

FIGURE 6 – Spectrogrammes d'un même signal débruité, avec des niveaux de bruit différents

$$SNRseg = \frac{10}{M} \sum_{m=0}^{M-1} log_{10} \frac{\sum_{i=Nm}^{Nm+N-1} x(i))^2}{\sum_{i=Nm}^{Nm+N-1} (x(i)-y(i))^2}$$

où N et M sont respectivement la longueur du segment et le nombre de segments.

SNRseg doit donner des résultats plus fiables que le SNR car, en moyennant sur des segments de parole d'une longueur entre 15 et 20 ms, on réduit l'impact des disparités. Cette métrique est également moins sensible aux décalages temporels entre le signal d'origine et le signal traité, qui ont pu être introduit lors de la reconstruction du signal.

Metrics	**SNR**	**segSNR**
Mean SNR (clean vs pred)	1.859163	1.721648
STD SNR (clean vs pred	0.734523	0.636271

TABLE 5 – Statistiques sur métriques SNR et segSNR

5.2.2 Mesures subjectives

L'inconvénient des métriques vues dans la partie précédente est qu'elles ne sont pas toujours corrélées avec la qualité perçue à l'oreille. Une méthode également précise pour évaluer la qualité d'un signal de parole est alors de faire confiance à l'oreille humaine et de faire effectuer à un groupe de personnes un test d'écoute. Il existe différentes méthodologies, la plus simple est de décrire le niveau de bruit restant (BAK) puis de noter la qualité globale du débruitage effectué.

Dans notre cas, les sons reconstruit étant les mêmes quelque soit le niveau du bruit ajouté, on ne peut

5 - Not noticeable
4 - Somewhat noticeable
3 - Noticeable but not intrusive
2 - Fairly conspicuous, somewhat intrusive
1 - Very conspicuous, very intrusive

TABLE 6 – Echelle quantifiant le niveau de bruit restant (BAK)

5=excellent
4=good
3=fair
2=poor
1=bad

TABLE 7 – Echelle quantifiant la qualité globale du débruitage (OVRL)

faire une étude comparative de qualité. Cependant, en utilisant les deux échelles subjectives présentées précédemment nous obtenons les résultats présentés dans le tableau 8.

BAK	2
OVRL	3

TABLE 8 – Notes données à notre méthode de débruitage suivant les deux métriques subjectives

Les notes sont assez basses car le signal sonore débruité n'est pas du tout de bonne qualité. Il présente une fond sonore persistant semblable à du bruit blanc et la voix du locuteur sonne très robotique. POur autant, la tâche de débruitage du bruit babble semble très bien s'effectuer. Ainsi, même avec un niveau de bruit en entrée à -20 dB, où, à l'oreille il est impossible de comprendre la phrase prononcée, le son une fois débruité est tout à fait compréhensible mais a perdu en qualité. Il semblerait donc que la tâche de débruitage fonctionne, mais que cependant, en atténuant le babble noise, trop d'informations utiles sont également retirées, introduisant de fortes pertes en qualité.

6 Discussion

Les résultats obtenus ne sont pas pleinement satisfaisants. Le débruitage se fait, mais introduit de fortes distorsions sur les sons traités. En règle générale, pour différents niveaux de bruit, ce traitement va permettre de retirer le *babble noise* qui gêne fortement la compréhension de la phrase. Si l'objectif est donc la compréhension de la phrase ce traitement fonctionne, il va cependant fortement déformer la voix du locuteur.

Toutefois, si l'on se contente de juger le modèle sur sa tâche de débruitage de STFT, alors les performances sont plutôt remarquablement bonnes. Ainsi, pour pouvoir amélorer le résultat sur le rendu audio, il faut essayer de comprendre quel est la cause précise de la différence de performance entre les mesures sur les STFT et celles sur l'audio.

Il y a beaucoup d'autres idées qui auraient pu être intéressantes à tester. Notamment essayer de bruiter le dataset avec différents bruits, afin de voir si le modèle parviendra à se généraliser sur plusieurs distributions. On aurait aussi pu essayer de bruiter le dataset avec un RSB positif (5dB, 10dB), correspondant à des cas plus simples mais également plus réalistes. Ou encore réaliser un apprentissage à niveau de RSB dégressif au fur et à mesure de l'apprentissage.

Notes

— Le code est disponible sur GitHub à l'adresse suivante, et est laissé à la libre utilisation : `https://github.com/Paul-Aime/speech_enhancement`

— Il contient également la fonction de conversion `nist2wav.py`, pour convertir les .WAV en .wav

— Peu ou pas d'exploration des méta-paramètres a été faite, alors qu'elle est primoridale. Cependant elle demande du temps de calcul, et le coût qui va avec. Par exemple pour ce projet, l'entraînement des modèles aux différents RSB à généré un coup de +70€ sur le Compute Engine de Google Cloud Platform (utilisation de crédit offerts), l'apprentissage étant 20 x plus long sur ordinateur personnel. Il serait raisonnable et souhaitable de faire remonter des demandes aux personnes adéquates pour aller dans le sens de la mise à disposition de puissance de calcul pour les étudiants.

— Ce projet a été assez fastidieux à mener à bout dans les temps. Il comporte tellement d'éléments décisifs que l'on avait envie de bien comprendre que cela nous a pris bien plus que les 32h normalement prévues. On avait peut être trop tendance à faire les choses proprement que nous avons perdu du temps. Le résultat est tout de même satisfaisant, car nous avons maintenant une structure bien construite que l'on pourra réutiliser dans de futurs projets.

Références

[1] Ahmed Hussen Abdelaziz. Ntcd-timit : A new database and baseline for noise-robust audio-visual speech recognition. In *Proc. Interspeech 2017*, pages 3752–3756, 2017.

[2] ducha aiki. caffenet-benchmark.

[3] D. Hepsiba and Judith Justin. *Role of Deep Neural Network in Speech Enhancement : A Review*, pages 103–112. 07 2019.

[4] Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 769–776. Curran Associates, Inc., 2009.

[5] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[6] Anurag Kumar and Dinei Florencio. Speech enhancement in multiple-noise conditions using deep neural networks. 05 2016.

[7] Yann Lecun and Y. Bengio. Convolutional networks for images, speech, and time-series. 01 1995.

[8] Brian McFee, Colin Raffel, Dawen Liang, Daniel Patrick Whittlesey Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa : Audio and music signal analysis in python. 2015.

[9] Se Rim Park and Jin Won Lee. A fully convolutional neural network for speech enhancement. In *Proc. Interspeech 2017*, pages 1993–1997, 2017.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch : An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[11] M. Portnoff. Short-time fourier analysis of sampled speech. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(3) :364–373, June 1981.

[12] Z. Průša, P. Balazs, and P. L. Søndergaard. A noniterative method for reconstruction of phase from stft magnitude. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(5) :1154–1164, May 2017.

[13] Z. Průša and P. Rajmic. Toward high-quality real-time signal reconstruction from stft magnitude. *IEEE Signal Processing Letters*, 24(6) :892–896, June 2017.

[14] Rashmirekha Ram and Mihir Mohanty. The use of deep learning in speech enhancement. pages 107–111, 01 2018.

[15] Nicolas Sturmel and Laurent Daudet. Signal reconstruction from stft magnitude : A state of the art. 2011.