



ENSAE PARIS

Internship Report : Speech Enhancement with noise-specific Denoiser

Author:

Mohamed Fyras TELMINI

Internship Supervisors:

Francesco PACI

January 28, 2024

Contents

1	Introduction	4
2	Noise-specific Denoiser	5
2.1	Intuition and motivation	5
2.2	Model architecture	6
2.2.1	Classifier Architecture	6
2.2.2	Denoiser architecture	7
3	Datasets	9
3.1	Reference datasets	9
3.2	Single-noise dataset creation	10
3.3	Data augmentation	12
4	Training setup	15
4.1	Classifier	15
4.2	Denoiser	17
4.3	Test Metrics	17
5	Experimental setup	21
5.1	Final denoising model architecture	21
5.2	Test Datasets	22
5.3	First experiment: comparing single-noise models to all-noise model	24
5.3.1	Results for the Human test set:	24
5.3.2	Results for the Music test set:	25
5.3.3	Results for the Vehicle test set:	25
5.3.4	Conclusion:	25
5.4	Second experiment: comparing all different combinations of single-noise datasets	26
5.4.1	Results for the Human test set:	26
5.4.2	Results for the Music test set:	27
5.4.3	Results for the Vehicle test set:	27
5.4.4	Conclusion:	27
5.5	Third experiment: taking into consideration the classifier	28
6	Conclusion	30

List of Figures

1	Classifier + denoiser model architecture	6
2	The modified architecture of the proposed end-to-end 1D CNN for environmental sound classification based on [1dc]	7
3	The denoiser model architecture	8
4	Data augmentation pipeline	14
5	Triplet loss on two positive faces (Obama) and one negative face (Macron) . . .	16
6	Evolution of the training and validation loss during the training of the classifier model	16
7	Evolution of the training and validation loss during the training of the denoiser model	18
8	SNR histogram for Human training dataset	19
9	SNR histogram for Music training dataset	20
10	SNR histogram for Vehicle training dataset	20
11	SNR histograms for the different test sets before and after the removal of outliers	23

List of Tables

1	Repartition of the original noise labels of DNS dataset in the 3 new categories Human , Music and Vehicle	11
2	Comparison of length in hours of the single-noise datasets and the Valentini dataset	12
3	Target lengths in hours for each single-noise dataset	12
4	Observations on the effect of the data augmentation techniques on the short-time Fourier transform of the noise signals	13
5	Ranges of parameters for the transformations	14
6	Results of testing different architectures of the baseline denoiser model on the DNS test-set	22
7	Number of training epochs for each training dataset	24
8	Results of testing single-noise and All on the Human test set	25
9	Results of testing single-noise and All on the Music test set	25
10	Results of testing single-noise and All on the Vehicle test set	25
11	Number of training epochs for each training dataset including the new datasets for the second experiment	26
12	Results the second experiment for the testing on the Human test set	26
13	Results the second experiment for the testing on the Music test set	27
14	Results the second experiment for the testing on the Vehicle test set	27
15	Results of the third experiment	28

1 Introduction

In the field of speech enhancement and specifically in audio noise removal, state-of-the-art models like [empty citation] and [2] often have a large number of parameters and require a large amount of computing power. This makes them ill-fitted for problems that require fast computation time and low computer memory usage. An example of such problems is the implementation of real-time denoising algorithms for smart objects such as earbuds or phones. In order to address both these problems, this internship focused on testing if smaller deep-learning-based denoising models are better at performing this task if they are trained on specific kinds of noise rather than on multiple ones.

2 Noise-specific Denoiser

This part of the report focuses on the models that were developed during this internship, these models consist of a classifier network that labels input data according to the source of noise it thinks it contains. And a denoising network that tries to eliminate noise from noisy audio inputs. A fusion of these two models is also constructed, as it will be used in the final experiments of this internship.

2.1 Intuition and motivation

The Goal of this internship was to study the possibility of conceiving a small-scale denoiser model by exploiting the variance-bias[DBLP:journals/corr/abs-1810-08591] trade-off in small neural networks. The initial intuition was that by increasing the bias in the training data, smaller models would maybe be as efficient at doing the denoising than larger ones. In other words, by training multiple smaller models, each on a specialized dataset, the overall performance of these smaller models combined would be comparable to that of a larger model trained on all the available data. This hypothesis is quite difficult to check, but a first step would be to compare the performance of multiple small denoising models trained on specialized datasets with the performance of a similar model trained on all available data. This would show us if there is an interest in pursuing this study further, because if there is no tangible difference in performance between these two kinds of models then it wouldn't make sense to check if larger models perform as well or worse.

In the case of this internship, two main models were used. First, a classifier model that was trained on labeled data from different categories of noise. The architecture of this model and the data it was trained on will be explained further in this report. Second, a denoiser model that was trained on datasets corresponding to various categories of noise. Since the denoiser architecture is the same for all different training setups, each training setup would be used to characterize a unique denoiser model. In the case we have n noise categories, each labeled i , we would end up with n different denoiser models. To avoid confusion, each denoiser model would be named i corresponding to the noise category it was trained on.

Additionally, two kinds of denoising architectures would be used in the latter tests. The first one is the **extended model** also simply called **Classifier + denoiser**. This model is a succession of a classifier model and a denoiser model. The denoiser model is trained on different datasets, each corresponding to a classification label. The classifier model will return a label that will be used to select the set of weights that would be loaded into the denoiser model. This model would be compared to a the second denoising architecture that would be called **All**. **All** is constituted of one denoiser model that had been trained on all available data regardless of

label.

2.2 Model architecture

The extended model is simply a succession of a classifier and a denoiser. The architecture of both models is explained in the following sections. Note that each network is trained individually.

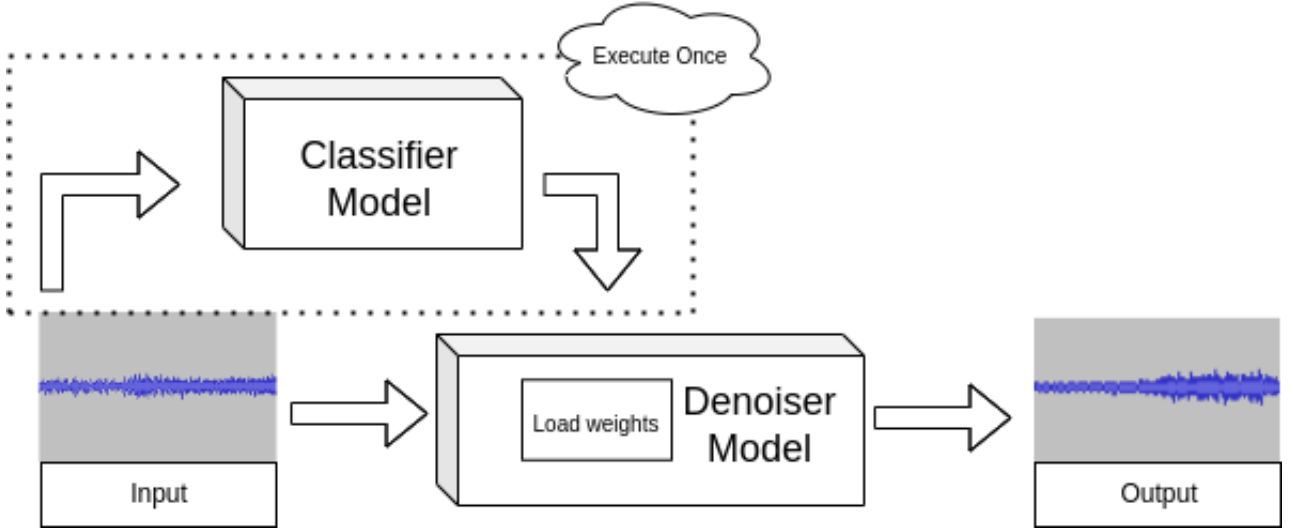


Figure 1: Classifier + denoiser model architecture

Figure 1 shows how the classifier + denoiser architecture works. An audio input is given to the model, one second of this input is first fed to a classifier network that returns a label for it. This label is used by the denoiser model to determine which set of weights to load. This makes it so that the denoiser network has multiple sets of weights, each set is the result of training on a particular label of noise. After loading the correct set of weights, the denoising is executed on the entire input.

In the rest of this report, since the denoiser model’s architecture is fixed, the different sets of weights would characterize different versions of the model and are considered to be individual models.

2.2.1 Classifier Architecture

The first part of the extended model is the classifier network. The chosen architecture was based on [1dc]. The choice of this architecture was due to its relatively low number of parameters ($<300\,000$) and its ease of implementation. The original architecture was slightly altered by adding another fully connected layer between the last two encoding layers to compensate for the increased complexity of the problem’s data compared to the original data in the paper. Since the original paper considered a problem of noise classification and in our situation, we’re classifying noisy speech and not just noise. The addition of the layer and its size were arbitrary.

More careful consideration of the layer’s size could be done empirically, but due to time and resources constraints, it was decided to not examine this matter any further.

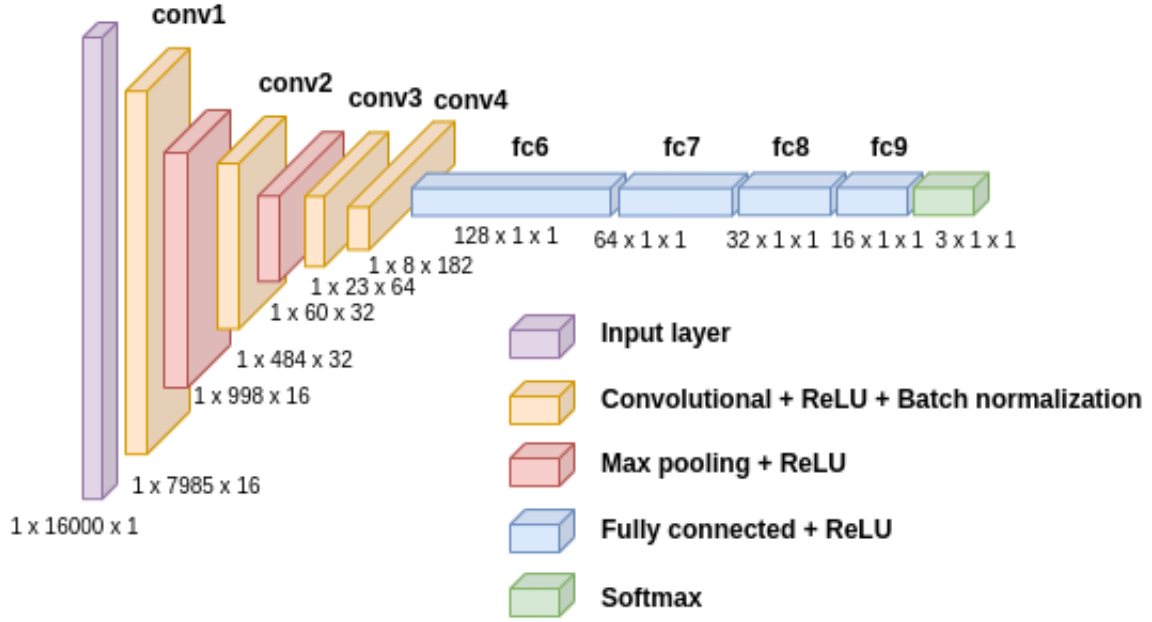


Figure 2: The modified architecture of the proposed end-to-end 1D CNN for environmental sound classification based on [1dc].

The model was implemented using Keras[chollet2015keras]. The figure 2 shows the modified architecture of the classifier model. The first layer of the model is a 1-dimension convolution layer initialized via gamma-tone filter-banks[1dc], which are a succession of filters primarily used to simulate the human ear’s perception of sound. This layer is not trainable. The following layers are a succession of convolution and max-pooling layers that create a feature map that is then fed to a succession of decreasingly small fully connected layers, until a final soft-max layer that determines a single output. The final output determines the label of the input noisy file.

2.2.2 Denoiser architecture

The denoiser is arguably the most important component of this architecture. This is because it determines if our initial hypothesis is true. The choice of the network was again motivated by the constraints of size and ease of implementation. The original architecture the denoiser was based on is DMUCS[defossez2021hybrid]. This architecture was then highly altered to reduce parameter size. Figure 3 shows this modified architecture. The implementation is made with Pytorch[NEURIPS2019_9015].

The model works directly on the short-time Fourier transform[stft] of the input audio (referred to as STFT). A sliding window of a given length extracts segments of the STFT of the input audio which are then fed one-by-one to the denoiser model. The model consists of an enhancement layer followed by two encoding layers that generate a binary mask. The input is then

multiplied by the generated binary mask in order to obtain the STFT of the denoised audio signal. The choice of the nature of the enhancement layer was based on internal testing done by the company's team. This will be explained later in the report.

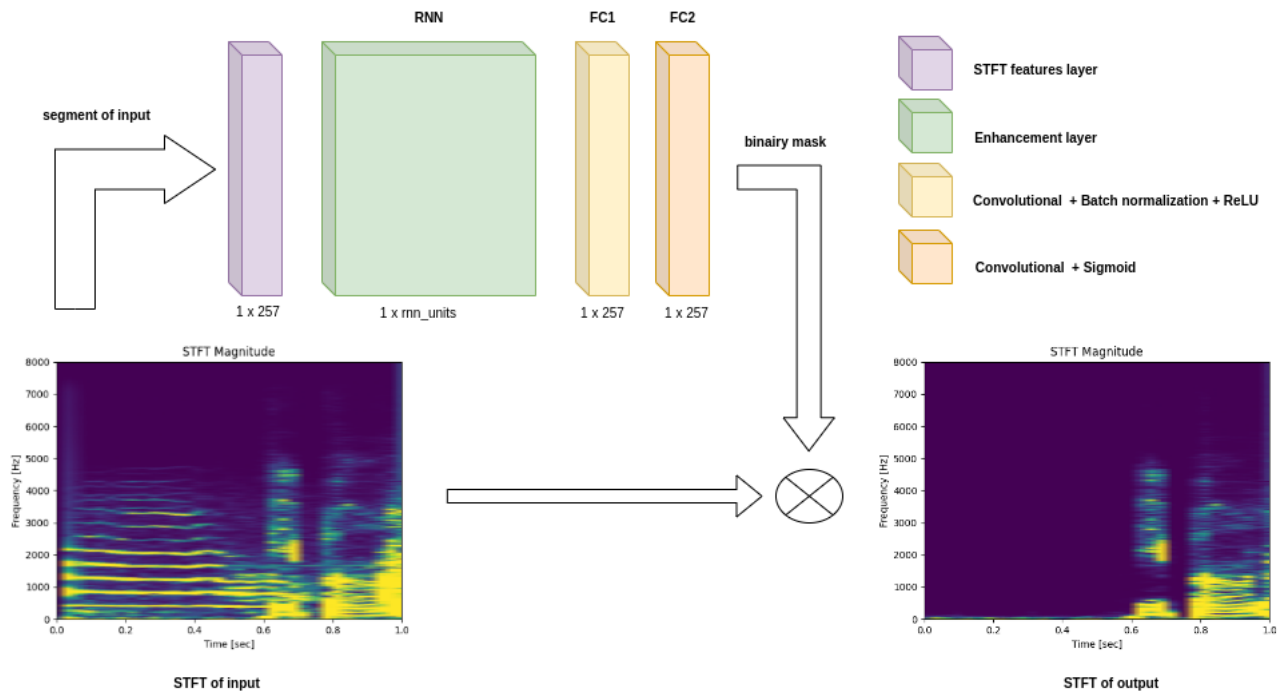


Figure 3: The denoiser model architecture

3 Datasets

This part focuses on the different datasets used in order to be able to compare the performance of the denoising networks. This part is crucial since later in the report, denoising models and their corresponding training datasets will be referred to by the same name. Thus it is important to explain in details the way these datasets were constructed, and what they contain. The first section talks about the two reference datasets that were used to construct the custom datasets of this internship. The second section explains the thought process behind the creation of these custom datasets. The third part explains the different data augmentation methods that were used to make the custom datasets larger in size.

3.1 Reference datasets

In the field of speech enhancement and specifically in deep-learning-based audio denoising, there is a commonly-used structure for the network training data. Since the task of the neural network is to identify noise and remove it from an audio file, the network would be trained on noisy files and their non-noisy counterparts, also referred to by "clean". This is so that the network can be able to quantify the difference between these two cases and optimize itself. As such, the structure of audio denoising datasets consists of pairs of audio files of a given length. Each pair has two versions of the same audio file, one with additive noise and one without.

Noise in this case would be a recording of any type of sound that we are trying to remove that is added on top of the original audio. The original audio is not distorted and the noise is strictly additive.

The training data is constructed such that: **Noisy** = **Clean** + **Noise**

Clean is generally an audio recording recovered from a voice recognition dataset. It contains a high quality recording of one speaker saying a given sentence. **Noise** can be obtained from multiple sources. Training datasets contain (**Noisy**, **Clean**) pairs of audio files.

There exists two commonly used datasets that have this structure. The first is called the **Valentini-Botinhao**[ValentiniBotinhao2017NoisySD] dataset, which we will later refer to as **Valentini**. It has two versions, the first consists of 11572 audio recordings of 28 different speakers. The second one consists of 23073 audio recordings of 56 different speakers. These recordings are then mixed with noises from eight categories (babble, cafeteria, car, kitchen, meeting, metro, restaurant, static). This dataset also contains 824 test files containing noise from five categories (bus, cafe, living, office, public square). The original source of the audio recordings is the CSTR-VCTK-Corpus[<https://doi.org/10.7488/ds/1994>] dataset. The noises are recovered from the Demand[thiemann_joachim_2013_1227121] database. This dataset is commonly used to train audio denoising networks as it is relatively small in storage

size.

The second dataset is called DNS[**dns**] dataset in reference to the annual Deep-Noise-Suppression challenge organized by Microsoft. This is the official dataset used for the challenge and changes every year, but the changes are mostly the addition of new files. This dataset contains a immense amount of clean speech files from multiple languages recovered from multiple clean speech datasets. This dataset has the particularity of being highly customizable, since it contains separates noise and clean speech files, it is never used as-is. Instead, scripts are used to generate (**Noisy,Clean**) pairs from the different available files. These pairs are then downloaded locally to the user’s machine. In this internship, the DNS dataset was not used directly, instead, it was used as a source for the **Noise** files that were later mixed with separate **Clean** files from the **Valentini** dataset in order to make the custom (**Noisy,Clean**) pairs. This was used because the DNS dataset contained a rich assortment of noise labels (520 were used), which were later grouped into larger labels.

3.2 Single-noise dataset creation

The first step during the construction of the dataset was to select the categories of noise that would be considered. Five categories were chosen. These categories were **Animal**, **Vehicle**, **Human**, **Music** and **Other**. **Animal** referred to animal sounds, **Vehicle** referred to vehicles and engine sounds, **Human** contained human-made noise ranging from background talk to caughing and baby sounds, **Music** contained various recordings of instruments and electronic music and **Other** contained the noises that didn’t fall into any of the preceding categories. The original labels of the **DNS** dataset **Noise** files were recovered and reassigned to one of the five categories mentioned above. For making the training and testing of the models quicker, only three categories were kept, those categories being **Vehicle**, **Human** and **Music**. The table 1 shows the different labels contained in each of those categories. For document layout reasons, not all labels are present on the table but a majority of them helps illustrate the semantic logic behind this segmentation.

	Label		
	Human	Music	Vehicle
Content	'Babbling', 'Baby cry, infant cry', 'Baby laughter', 'Bellow', 'Belly laugh', 'Bleat', 'Breathing', 'Burping, eructation', 'Chatter', 'Child speech, kid speaking', 'Children playing', 'Female singing', 'Female speech, woman speaking', 'Gargling', 'Gasp', 'Giggle', 'Gobble', 'Groan', 'Grunt', 'Gurgling', 'Hiccup', 'Male speech, man speaking', 'Mantra', 'Neigh, whinny', 'Pant', 'Screaming', 'Shout', 'Sigh', 'Sneeze', 'Sniff', 'Snoring', 'Snort', 'Speech', 'Stomach rumble', etc..	'A capella', 'Accordion', 'Acoustic guitar', 'Afrobeat', 'Ambient music', 'Angry music', 'Background music', 'Bagpipes', 'Banjo', 'Beatboxing', 'Brass instrument', 'Carnatic music', 'Cello', 'Chant', 'Choir', 'Chorus effect', 'Christian music', 'Steel guitar, slide guitar', 'String section', 'Strum', 'Swing music', 'Synthesizer', 'Synthetic singing', 'Tabla', 'Tambourine', 'Tapping (guitar technique)', 'Techno', 'Ukulele', 'Vibraphone', 'Video game music', 'Violin, fiddle', 'Vocal music', 'Wedding music', 'Wind instrument, woodwind instrument', 'Yodeling', etc..	'Accelerating, revving, vroom', 'Air horn, truck horn', 'Bus', 'Car', 'Car alarm', 'Car passing by', 'Emergency vehicle', 'Fire engine, fire truck (siren)', 'Fixed-wing aircraft, airplane', 'Foghorn', 'French horn', 'Gears', 'Heavy engine (low frequency)', 'Helicopter', 'Jet engine', 'Lawn mower', 'Light engine (high frequency)', 'Mechanisms', 'Medium engine (mid frequency)', 'Motor vehicle (road)', 'Motorboat, speedboat', 'Motorcycle', 'Police car (siren)', 'Propeller, airscrew', 'Race car, auto racing', 'Radio', 'Rail transport', 'Ship', 'Siren', 'Skidding', 'Subway, metro, underground', 'Vehicle horn, car horn, honking', etc..
Label count	53	147	55

Table 1: Repartition of the original noise labels of DNS dataset in the 3 new categories **Human**, **Music** and **Vehicle**

The table 2 below shows the number of hours of noise contained within each category with comparison to the **Valentini** dataset content.

Dataset	Human	Music	Vehicle	Valentini
Duration (hours)	4	6	9	21

Table 2: Comparison of length in hours of the single-noise datasets and the Valentini dataset

This shows how the noise data in each category is considerably small in comparison to the noisy files within the Valentini dataset. This is due to how only a fragment of the **DNS** dataset was considered in this segmentation (only 255 of the original 520 labels are taken into consideration). This issue is solved through data augmentation, which is done in two separate manners. The first step was to generate augmented noise files from transformations on the original noise files of each of the selected categories. The second step was mixing these files with different clean speech files. The second step allows for the same noise file to be mixed with distinct clean speech files.

3.3 Data augmentation

A fixed target length of 21 hours was chosen in order to be able to compare the results of each dataset to the Valentini dataset.

Dataset	Human	Music	Vehicle
Duration (hours)	4	6	9
Target duration (hours)	21	21	21

Table 3: Target lengths in hours for each single-noise dataset

The first step of data augmentation consisted in the generation of new noise files by using transformations on the original ones. Four transformations were selected:

1. **Polarity inversion:** This transformation is simply the multiplication of the values of the audio signal by -1. This doesn't change the way the signal is perceived by human ear.
2. **Time stretch:** This transformation stretches the audio signal duration without affecting its pitch. The parameter of this transformation is the stretch rate, which is the factor by which the original audio is multiplied duration-wise.
3. **Pitch scale:** This transformation changes the pitch of the audio signal without affecting its duration. The parameter of this transformation is the number of semitones by which the pitch of the audio is shifted.

4. **Gain:** This transformation changes the amplitude of the audio signal. The parameter of this transformation is the factor by which the audio amplitude is multiplied.

The effect of these transformations on the STFT spectrograms of the audio files can be seen in the table 4.

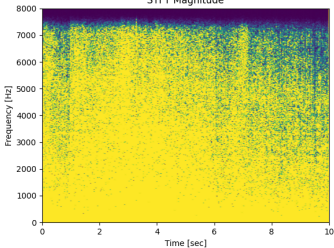
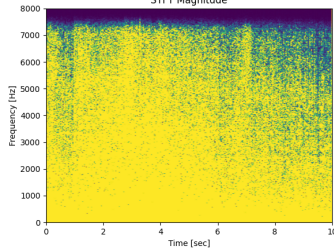
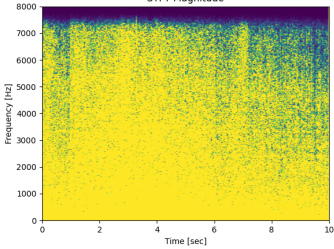
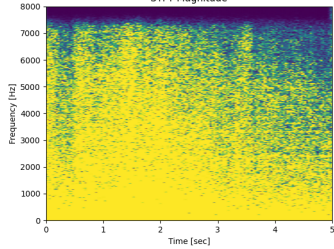
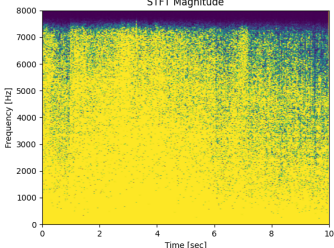
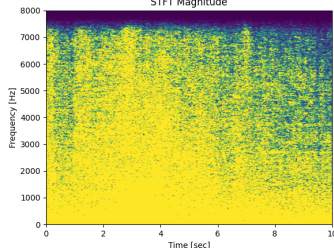
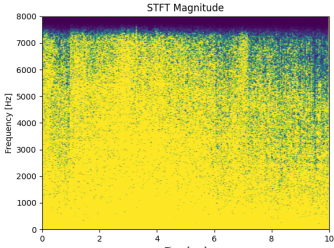
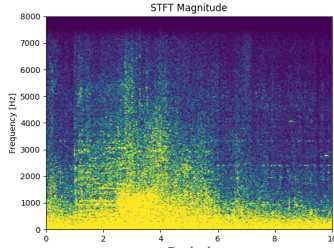
Transformation	Input	Output
Polarity inversion		
Time stretch time_stretch_rate = 2		
Pitch scale num_semitones = 2		
Random gain gain_rate = 0.12		

Table 4: Observations on the effect of the data augmentation techniques on the short-time Fourier transform of the noise signals

The polarity inversion is applied to all the dataset files. Then for each dataset file, from the three other transformations, two are randomly applied with random parameters. This is explained in figure 4.

Since the parameters of these transformations are random (except polarity inversion), a limit

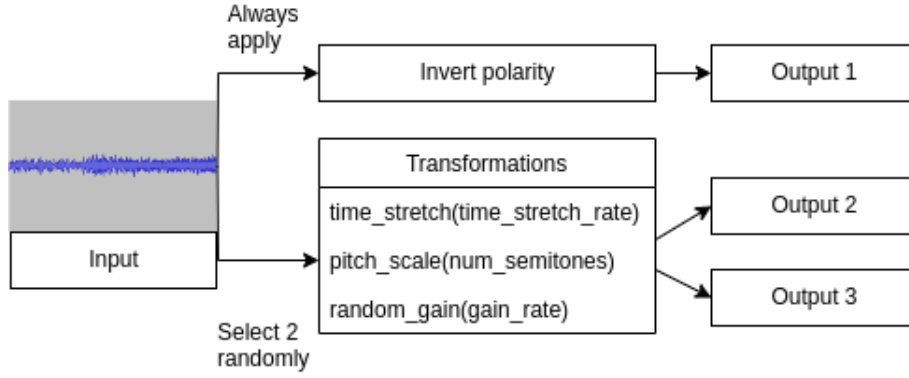


Figure 4: Data augmentation pipeline

had to be set in regards to their range. The value of each parameter is drawn from a uniform distribution within its parameter range every time the transformation is applied. Table 5 shows the ranges within which the parameters are drawn.

Transformation	Parameter	Parameter range
Polarity inversion	none	none
Time stretch	stretch rate	[1.5,2.7]
Pitch scale	number of semitones	[-3,3]
Random gain	gain rate	[0.1,1.12]

Table 5: Ranges of parameters for the transformations

To obtain the final noisy datasets, the original noise files were mixed with clean audio files from the **Valentini** dataset. A fraction of the augmented data was then mixed with the rest of the clean **Valentini** files until 21 hours of data were obtained for each category.

4 Training setup

This part of the report explains the training procedures for the classifier and the denoiser networks. Including the training hyper-parameters and the loss graphs of each network. The last section explains the metrics used in the context of this internship.

4.1 Classifier

The classifier model was trained on the non-augmented portion of the dataset which contained three different categories of noise (**Human**, **Music** and **Vehicle**) mixed with random clean speech files. Each file was associated to a label and the model trained on predicting the right label for each input. The loss function used was the triplet[JMLR:v11:chechik10a] loss function. This loss function is often used in facial recognition tasks and is more suited for situations where notions of similarity are needed. This loss function makes sure that the embeddings of files from the same category end up close, and those of files from different categories end up far. In our case, we want the classifier to be able to correctly assign labels to audio files that correspond to three very different categories. Which means that the embeddings of files corresponding to the same label need to be close while those corresponding to different labels need to be far. Thus triplet loss was chosen instead of more conventional distance-based loss functions. This function works as follows:

1. Select Anchor **A**: This is the embedding of the file that is to be compared
2. Select a positive **P**: This is the embedding of a file with a label similar to **A**
3. Select a negative **N**: This is the embedding of a file with a different label than **A**
4. Calculate $Loss = \max((distance(\mathbf{A}, \mathbf{P}) - distance(\mathbf{A}, \mathbf{N}) + \alpha), 0)$ with $distance()$ a measure of distance on the embedding space containing the triplet $(\mathbf{A}, \mathbf{P}, \mathbf{N})$ and given α a margin that separates positives and negatives pairs.

In our case, an $\alpha=0.05$ was chosen. And the euclidean norm $\| \cdot \|_2^2$ was used as the distance function.

Training of the classifier ran for 1000 epochs, the figure 6 shows the loss function evolution during training. It seemed quite clear that the model over-fitted fast. This could be due to the semantic nature of the labelling of the dataset, this will be explained in the last part of the report. Note that for both the classifier and the denoiser models, 10% of the training data was used as validation data.

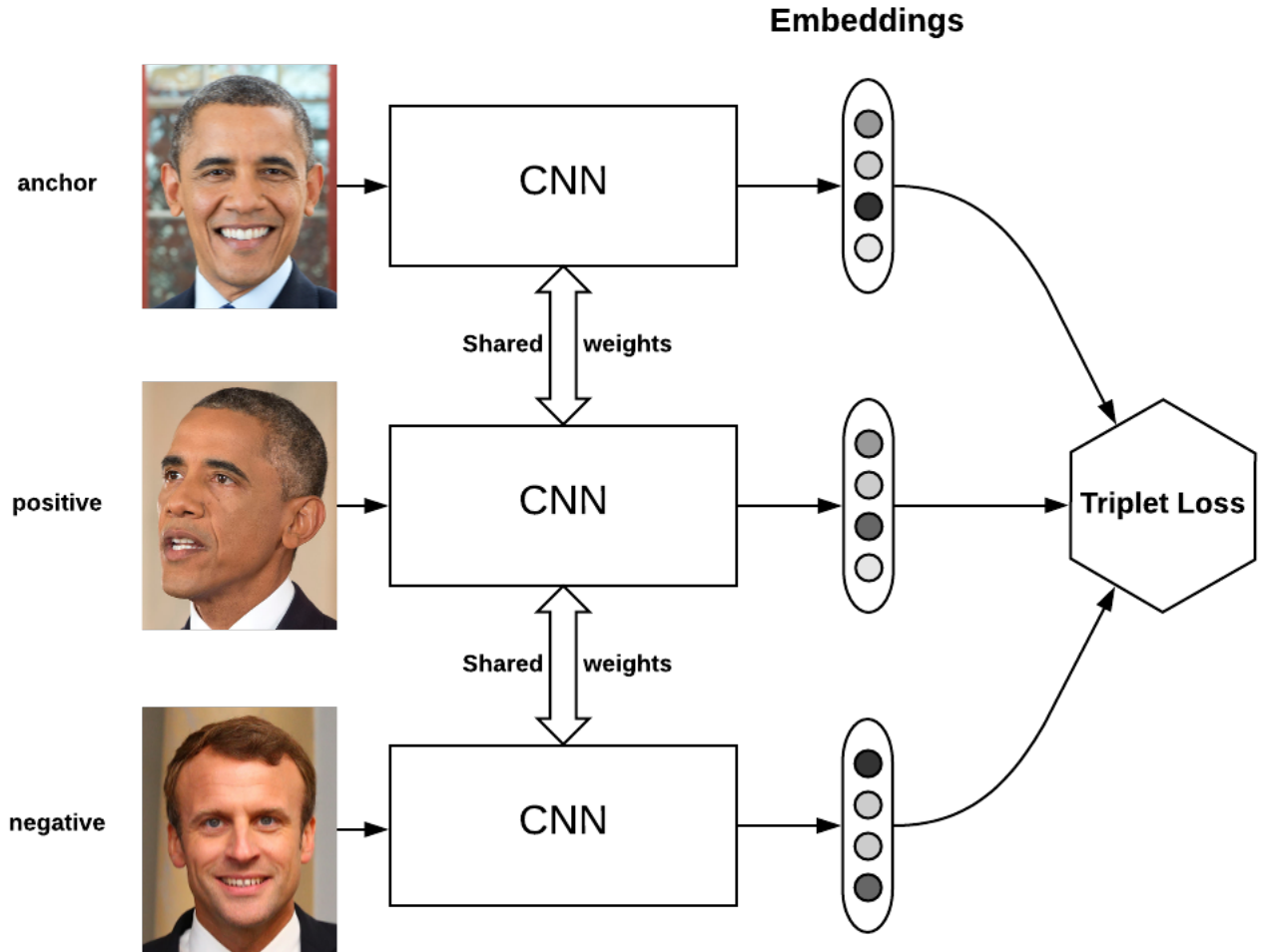


Figure 5: Triplet loss on two positive faces (Obama) and one negative face (Macron)

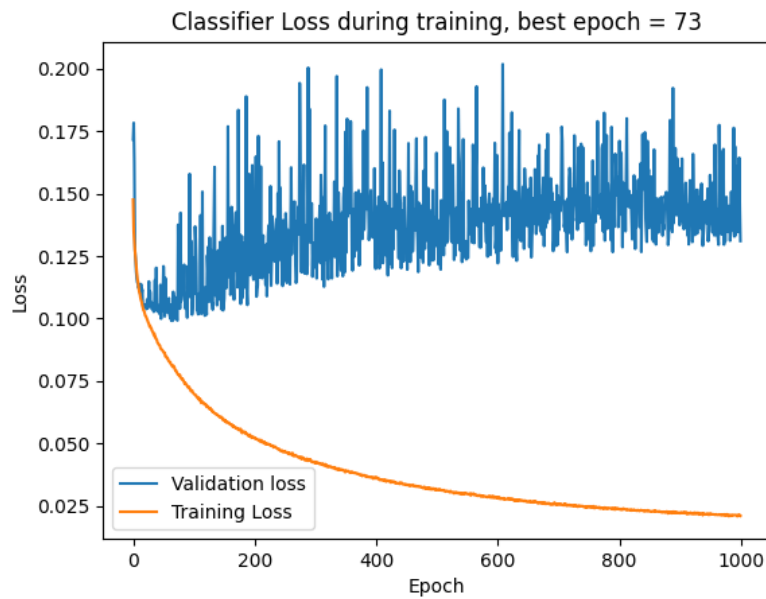


Figure 6: Evolution of the training and validation loss during the training of the classifier model

4.2 Denoiser

There are multiple versions of the denoiser model used, each version is trained on a different dataset with a different number of epochs. The explanation behind the choice of training on different number of epochs is in part 4 of this report, since it's fundamentally related to the experiments done on the models. All versions of the model are trained using l2-loss which has the formula: $L2_{loss}(Y, \hat{Y}) = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ for an element $Y = \{y_i\}_{i=1}^n$ and its estimation $\hat{Y} = \{\hat{y}_i\}_{i=1}^n$

The figure 7 shows the different loss curves for each model, the label of the model corresponds to the dataset with which it was trained. Models with multiple labels are trained on the sum of the datasets corresponding to their labels. This is explained in part 4 of this report.

4.3 Test Metrics

The most commonly used metric on speech enhancement problems is a mean opinion score (MOS) based on a very specific crowd-sourced testing protocol. This allows the quantification of the abstract notion of perception. Another alternative to these crowd-sourced protocols are computable metrics that simulate the human perception. These metrics were developed as a solution to the possible unavailability of crowd-sourcing during research.

The metrics used to test the performance of the models are PESQ[**pesq**] and STOI[**stoi**]:

1. **PESQ**: "Perceptual Evaluation of Speech Quality" is a metric created originally to evaluate the quality of phone conversations. Its calculated through an algorithm that, in our case, uses a clean reference speech file in order to determine its quality. This metric takes values ranging from -0.5 to 4.5, with the latter being an indicator of the highest quality of audio.
2. **STOI**: "Short-Time Objective Intelligibility" is another metric used to measure the quality of noisy speech with reference to a clean version of it. It is highly correlated with the intelligibility of the speech and takes values between 0 and 1, with the latter being an indicator of the highest quality of audio.

Both these metrics are used in the comparison of the models in order to make the conclusions more robust.

Another metric used in speech enhancement is SNR (Speech-Noise Ratio). This metric is usually used during dataset creation in order to control the amount of noise contained in the training and testing datasets. It is calculated using this formula:

$$SNR = 20 * \log_{10} \frac{(\sqrt{\widetilde{clean}^2})}{(\sqrt{\widetilde{noise}^2})} \text{ with :}$$

\widetilde{clean} : the empirical mean value of the clean speech signal

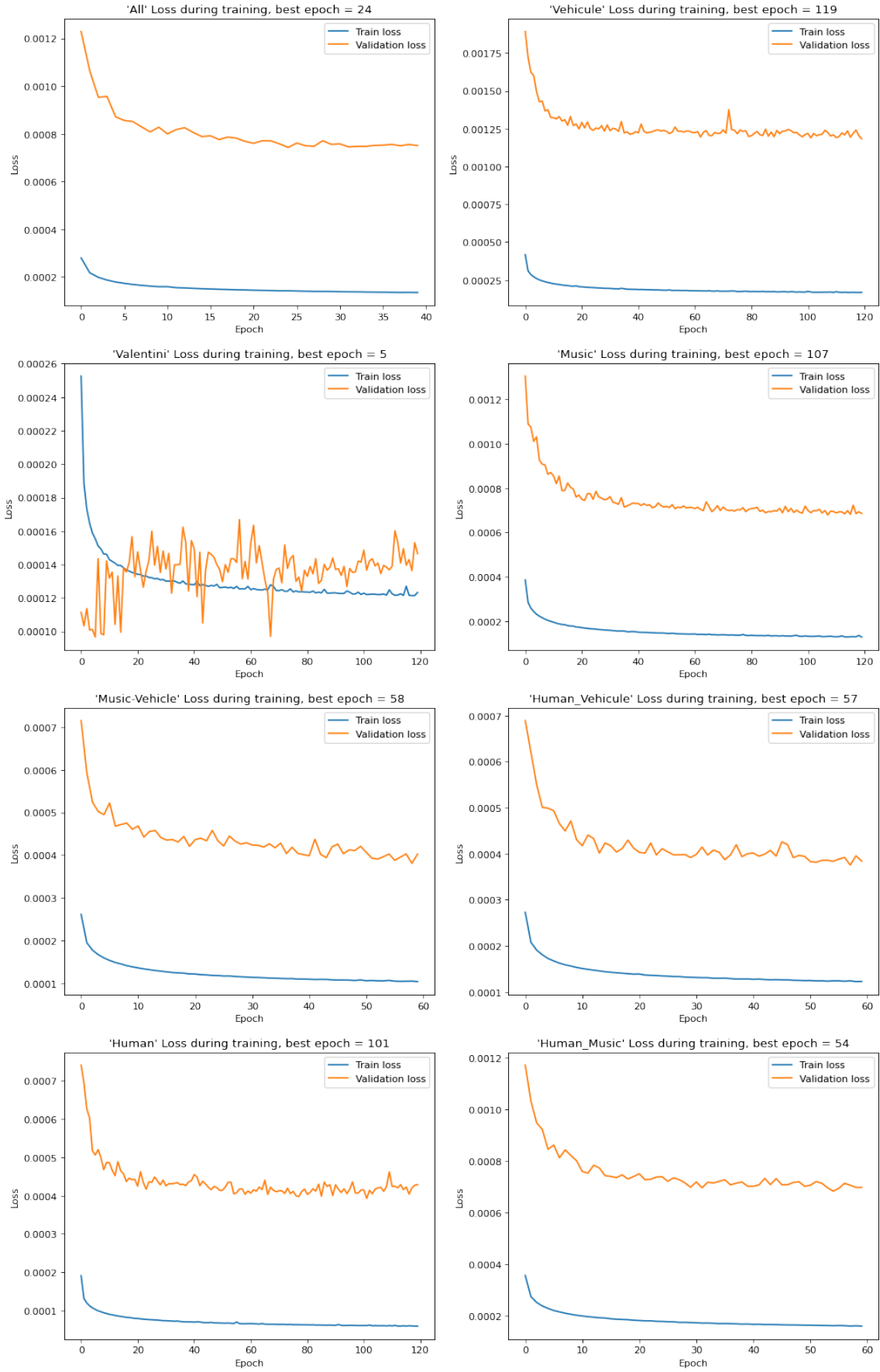


Figure 7: Evolution of the training and validation loss during the training of the denoiser model

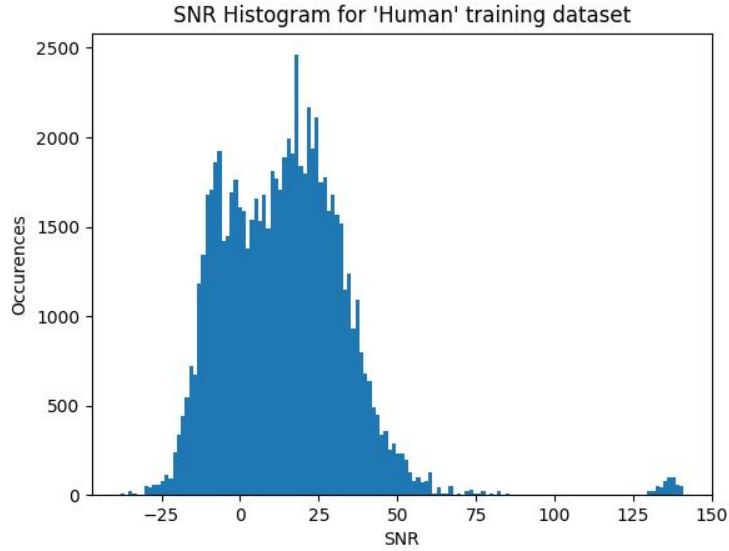


Figure 8: SNR histogram for **Human** training dataset

\widetilde{noise} : the empirical mean value of the noise signal

In the case of this internship, SNR was not calculated in advance. An observation of its values for the three single-noise training datasets (figures 8,9 and 10) shows it is mostly concentrated around the values between -25 and 25 for the **Music** and **Vehicle** datasets, and between -25 and 50 for the **Human** dataset. A few outlier exceptions with very high positive SNR exist in the range of 130 exist for all three datasets. These values correspond to files with very little noise compared to clean speech, and do not pose an issue in training since they are very rare. They were not removed before the training took place but do not pose an issue.

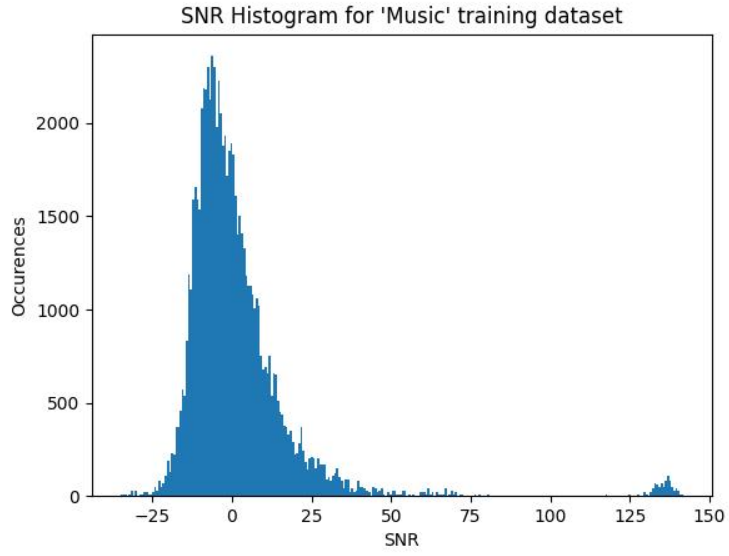


Figure 9: SNR histogram for **Music** training dataset

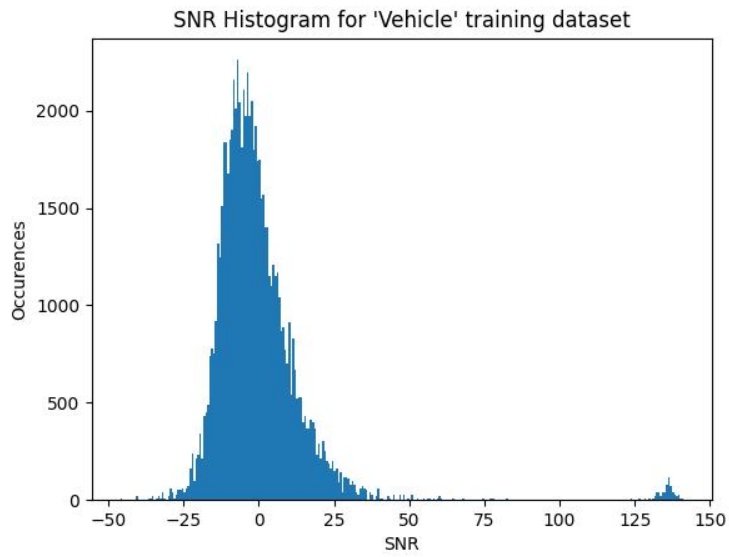


Figure 10: SNR histogram for **Vehicle** training dataset

5 Experimental setup

In essence, the objective of this internship was to compare two different types of audio denoising models:

1. Specialized or single-noise models: These models are trained on datasets containing one specific category of noise. And are tested on files corresponding to that category. They are:
 - (a) **Human**: Denoising model trained on the **Human** dataset.
 - (b) **Music**: Denoising model trained on the **Music** dataset.
 - (c) **Vehicle**: Denoising model trained on the **Vehicle** dataset.
2. Generalized or multiple-noises models: These models are trained on multiple datasets and are tested on data from their corresponding categories. They are:
 - (a) **Human-Music**: Denoising model trained on the fusion of the **Human** dataset and the **Music** dataset.
 - (b) **Human-Vehicle**: Denoising model trained on the fusion of the **Human** dataset and the **Vehicle** dataset.
 - (c) **Music-Vehicle**: Denoising model trained on the fusion of the **Music** dataset and the **Vehicle** dataset.
 - (d) **All**: Denoising model trained on all the single-noise datasets.

In this part, the experiments that took place in order to obtain a fair comparison between these two kinds of models will be explained. And the results of these experiments will be analyzed.

5.1 Final denoising model architecture

In part 2 of this report, the architecture of the denoising model was explained, without explicitly showing the content of the enhancement layer. This was because it was necessary to introduce the different speech enhancement metrics before justifying the choice of the content of this layer.

There were three possibilities for this choice:

1. Convolutional Layers[**conv**] : Used mainly in image processing, these layers execute a convolution operation on an input and allow for the extraction of high-level and low-level spacial features

2. Long-Short-Term-Memory[**LSTM**] Layers : Very used in Natural Language processing problems, these layers allow the capture of temporal dependencies in the data
3. Gated-Recurrent-Unit[**gru**] Layers : A type of layers that is conceptually very close to LSTM layers, except it is easier to train since it lacks some of the latter’s components.

Since this choice was important to make before the start of the experiment, different pre-trained versions of the model were tested on the official DNS test-set. These models were all trained on **Valentini** dataset for an equal amount of training iterations. The table 6 summarizes the performance of these different architectures.

Layer type	Units	PESQ	STOI
CNN	128	1.919	0.916
CNN	256	1.916	0.917
LSTM	128	2.284	0.936
LSTM	256	2.236	0.932
GRU	128	2.159	0.928
GRU	256	2.250	0.938

Table 6: Results of testing different architectures of the baseline denoiser model on the DNS test-set

In reference to these results, the final choice for the enhancement layer consisted of two LSTM layers with 128 units. More testing could be done to optimize the choice of the size of these layers, but this doesn’t affect the outcome of the comparison between the two types of models.

5.2 Test Datasets

The test datasets were created simultaneously with the training datasets, except that they did not contain any augmented data. They consisted of three sets of 100 files each, each file having a duration of 1 second. Each set corresponded to a single-noise category. These files were never seen by the models and were used in all the experiments that will be explained further in this report.

Since there were very few outliers on each test set (exactly 3 outliers per category), a choice was made to remove them. The figure 11 shows the SNR histograms of the different test sets before and after the removal of the outliers.

In short, the only difference between single-noise and multiple-noise models are the datasets they are trained on, everything else is similar. Earlier in this report it has been explained how the three single-noise datasets were created. Those three datasets were used to train three versions of the denoising model with similar architectures. A forth model was trained on the

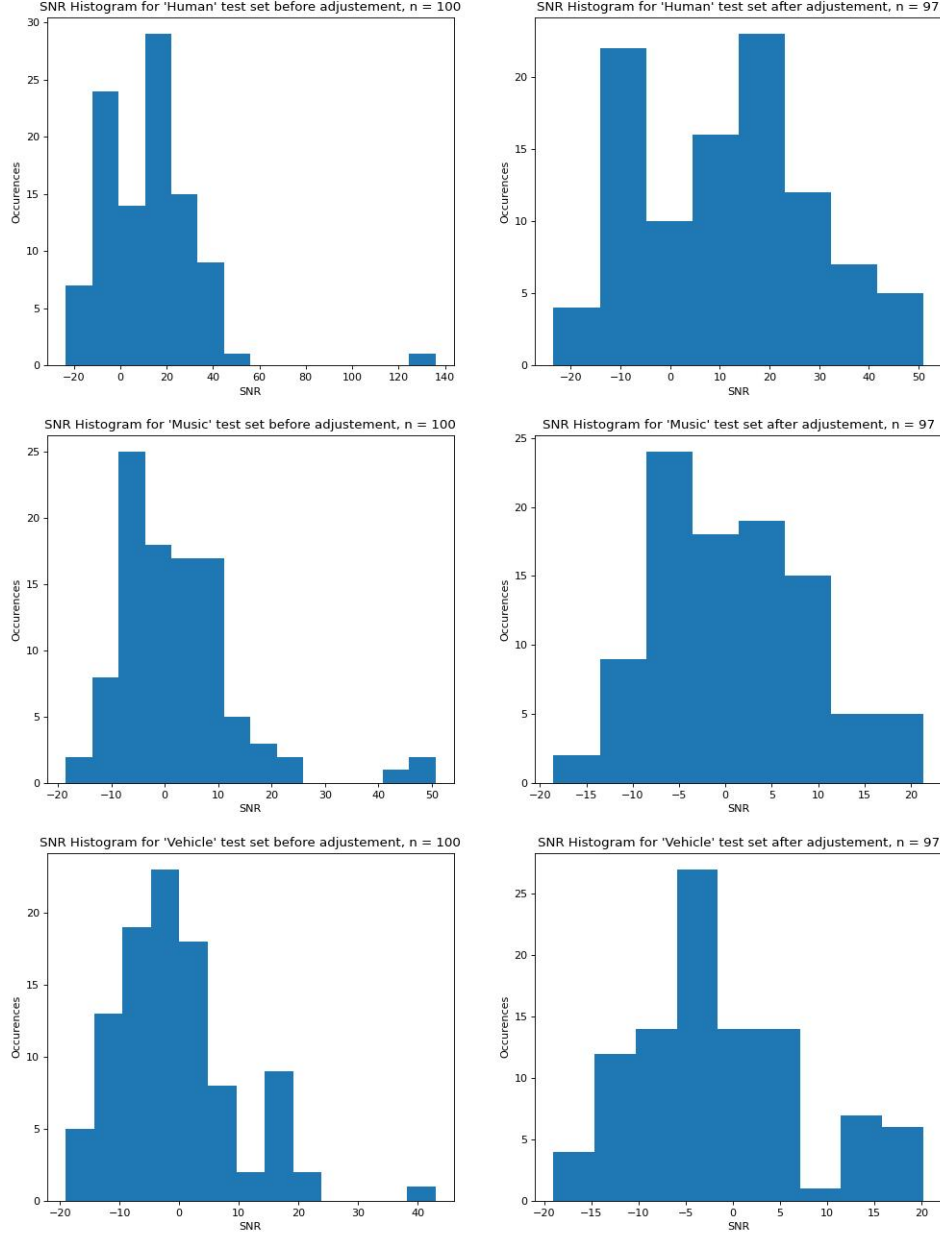


Figure 11: SNR histograms for the different test sets before and after the removal of outliers

combination of the three datasets.

In order to make the comparison fair, the four models had to be trained on equal numbers of iterations, an iteration being one file of the training dataset going through the model once. Since all audio files used in training were of equal length (1 second) this was easy to ensure. To ensure equal iterations its possible to refer to the number of training epochs: since all three

single-noise datasets were of equal size, and one epoch signified a number of iterations equal to the size of one dataset for the single-noise models and a third of the dataset of the multiple-noise models. All that had to be done was to train the single-noise models three times as much as the multiple-noise model.

During initial experiments, an important result became apparent: There were inconsistencies in regards to the comparisons, sometimes the single-noise datasets performed better, other times they did not. This interesting observation motivated the choice to dig deeper into these comparisons, by also including models that were trained on pairs of the single-noise datasets and reporting their performance for each noise category test-set.

5.3 First experiment: comparing single-noise models to all-noise model

The first experiment consisted in the comparison of the three single-noise datasets with the all-noise dataset. This was done by evaluating the performance of each single-noise dataset on the test-set associated to it, then by evaluating the performance of the all-noise model on all three test-sets separately.

Model name	Training Dataset	Number of epochs
Valentini	Valentini	120
Human	Human	120
Music	Music	120
Vehicle	Vehicle	120
All	Human + Music + Vehicule	40

Table 7: Number of training epochs for each training dataset

Two additional evaluations were done. The first was on the test files before any denoising, these are represented in the column **Noisy**. The second was on the performance of a model trained on the Valentini dataset, represented in the column **Valentini**. These two evaluations serve as a baseline in order to be able to observe the performance of the models that were trained on the custom datasets.

5.3.1 Results for the Human test set:

The analysis of table8 shows that **Human** outperforms **All** in both metrics. **Valentini** performs significantly worse than both models, this is due to the fact that it was optimized on noisy files of a different nature. Since **Valentini** only serves as a baseline in these experiments, this observation will be valid for both the tables 9 and 10.

	Model			
	Noisy	Valentini	Human	All
PESQ	1.907	1.788	<u>2.287</u>	2.235
STOI	0.797	0.729	<u>0.828</u>	0.82
Increase (PESQ)	0	-0.119	<u>0.38</u>	0.328
Increase (STOI)	0	-0.068	<u>0.031</u>	0.023

Table 8: Results of testing single-noise and **All** on the Human test set

5.3.2 Results for the Music test set:

	Model			
	Noisy	Valentini	Music	All
PESQ	1.331	1.336	1.822	<u>1.83</u>
STOI	0.72	0.678	0.784	<u>0.786</u>
Increase (PESQ)	0	0.005	0.491	<u>0.499</u>
Increase (STOI)	0	-0.042	0.064	<u>0.066</u>

Table 9: Results of testing single-noise and **All** on the Music test set

The Results on the music dataset contradict the results of table 8 since this time, **All** outperforms the single-noise model **Music** on both metrics.

5.3.3 Results for the Vehicle test set:

	Model			
	Noisy	Valentini	Vehicle	All
PESQ	1.251	1.269	1.612	<u>1.638</u>
STOI	0.673	0.606	0.722	<u>0.725</u>
Increase (PESQ)	0	0.018	0.361	<u>0.387</u>
Increase (STOI)	0	-0.067	0.048	<u>0.052</u>

Table 10: Results of testing single-noise and **All** on the Vehicle test set

Similarly to the results found in table 9, **All** outperforms **Vehicle** in both metrics.

5.3.4 Conclusion:

We cannot conclude in regards to the performance of single-noise models compared to **All**, since **Human** is the only one that outperforms it. This observation led to a series of other experiments that try to study how the mixing of various datasets affects the performance of the models in comparison to **All**.

5.4 Second experiment: comparing all different combinations of single-noise datasets

According to the results of the first experiment, additional investigations had to be made in regards to how fusing datasets from different categories affects the performance of the denoising models. A second experiment took place which was similar to the first, but with additional evaluations of the different models that were trained of pairs of the single-noise datasets. These three new pairs consisted in **Human-Music**, **Human-Vehicle** and **Vehicle-Music**.

To keep the comparisons consistent with the first experiment, the number of iterations during training was equalized in a similar logic as before. The table below sums up the different models.

Model name	Training Dataset	Number of epochs
Valentini	Valentini	120
Human	Human	120
Music	Music	120
Vehicle	Vehicle	120
Human-Music	Human + Music	60
Human-Vehicle	Human + Vehicle	60
Music-Vehicle	Music + Vehicle	60
All	Human + Music + Vehicle	40

Table 11: Number of training epochs for each training dataset including the new datasets for the second experiment

The study of the results of this experiment was similar to the first one. **Noisy** corresponded to the test files of each category before denoising, results for single-noise and all-noise dataset were also kept for comparison’s sake. The **Valentini** evaluation column was removed in order to have a consistent page layout in this report.

5.4.1 Results for the Human test set:

	Model				
	Noisy	Human	Human + Music	Human + Vehicle	All
PESQ	1.907	2.287	<u>2.306</u>	2.251	2.235
STOI	0.797	0.828	<u>0.827</u>	0.821	0.82
Increase (PESQ)	0	0.38	<u>0.399</u>	0.344	0.328
Increase (STOI)	0	0.031	<u>0.03</u>	0.024	0.023

Table 12: Results the second experiment for the testing on the Human test set

This time, table 12 shows that **Human+Music** outperforms all other models. This would indicate that these two noise categories could be well correlated with each-other. **Hu-**

man+Vehicle also outperforms **All** but not **Human**. This indicates the the addition of **Vehicle** dataset to **Human** actually lowers the performance of this model.

5.4.2 Results for the Music test set:

	Model				
	Noisy	Music	Human + Music	Music + Vehicle	All
PESQ	1.331	1.822	<u>1.853</u>	1.841	1.83
STOI	0.72	0.784	<u>0.789</u>	0.787	0.786
Increase (PESQ)	0	0.491	<u>0.522</u>	0.51	0.499
Increase (STOI)	0	0.064	<u>0.069</u>	0.067	0.066

Table 13: Results the second experiment for the testing on the Music test set

Similarly, table 13 shows that **Human+Music** still outperforms all other models when tested on the **music** test set. This would indicate that these two noise categories could be well correlated with each-other. It is also noticeable how **Music+Vehicle** outperforms both **Music** and **All**. This could indicate the **Music** performs better when mixed with other individual noise categories, but not both.

5.4.3 Results for the Vehicle test set:

	Model				
	Noisy	Vehicle	Human + Vehicle	Music + Vehicle	All
PESQ	1.251	1.612	1.608	1.625	1.638
STOI	0.673	0.722	0.721	0.721	<u>0.725</u>
Increase (PESQ)	0	0.361	0.357	0.374	<u>0.387</u>
Increase (STOI)	0	0.048	0.048	0.048	<u>0.052</u>

Table 14: Results the second experiment for the testing on the Vehicle test set

Table 14 shows that **All** outperforms all other models when tested on the **Vehicle** test set. The worst performance is that of **Human+Vehicle**, this confirms the intuition that these two noise categories should not be mixed together during training. Another observation is that the mixing of **Music** with **Vehicle** makes the model perform better. **Music** seems to enhance the performance of other single-noise models when it’s mixed with them, but also enhance it’s own.

5.4.4 Conclusion:

The results of this part confirm our intuition that denoising models perform better on specific cases when trained on certain datasets rather than others. The inconsistency of these results also proves that a semantics-based single-noise dataset creation, meaning that the noises within these datasets are grouped by source, isn’t the optimal way to construct datasets with noises

that are similar to each-other. A proposed method of dataset construction that was not applied in this internship was the usage of an encoding of noise files in order to separate them by clusters. These clusters would construct the new single-noise datasets and could contain files that seem contradictory semantically (maybe a baby’s scream is closer to a vehicle sound than a human’s cough) yet correlate well when we observe their embeddings. Nevertheless, we are not able to say that this proves anything in regards to our hypothesis (a group of smaller specialized models would out-perform one more general model) since the difference in performance of these models is not only inconsistent, but also relatively small. It would be cheaper in regards to time and resources to avoid doing all the work done in this study and to just train a model on all available data, since the improvement of separating the data does not seem as important as initially expected.

5.5 Third experiment: taking into consideration the classifier

The last experiment showed that in some cases, mixing datasets can ensure better performance overall. This third experiment serves to check if the classifier element helps to enhance the performance of single-noise models. The intuition behind this was that some files could be labelled under either of the three single-noise categories (due to semantic reasons during the initial creation of these datasets), yet would be more similar (in terms of embeddings created by the classifier) to those of another category. The classifier in this case would assign the input files to more similar categories and execute the most fitting denoising model on them.

The final three select models weren’t the single-noise models, but the three most performing models in each noise category. The model chosen for the category **Human** was **Human + Music**, the model chosen for the category **Music** was **Human + Music** and the model chosen for the category **Vehicle** was **All**. These three models are kept in the final "classifier + denoiser" model, and the performance of this new model was compared to the performance of the **All** model without a classifier. The performance of a "perfect" classifier was also evaluated to see if the difference between the theoretical best performance of the **classifier + denoiser** model and that of the **All** model.

		Model		
	Noisy	Classifier	No classifier	Perfect classifier
PESQ	1.452	1.745	1.901	<u>1.933</u>
Increase (PESQ)	0	0.293	0.449	0.481

Table 15: Results of the third experiment

The results of this experiment show that the addition of the classifier actually decreases the performance of the new model. The intuition behind the embeddings of the classifier was

wrong, also, it's very likely that the classifier's errors caused even worse performance for the new model. Since the perfect classifier still outperforms **All**, a case could be made for finding a better classifier architecture. Still, as previously argued in the second experiment, this approach is way too high effort for such a low increase in performance. **All** is still the better model overall.

6 Conclusion

This internship revolved mostly around two main problems in deep-learning based audio processing: noise classification and noise removal. An additional layer of complexity was added by coupling these two problems in order to study if deep-learning based noise removal works better if we know the noise label beforehand.

Certainly, the various experiments conducted in the later parts of this report were not able to confirm this initial guess. But the work done to arrive there allowed for a better understanding of how audio denoising models perform. The creation of the datasets also allowed for a better knowledge in regards to the particularities of this problem.

Overall, this internship was very technical, and involved very advanced applications of knowledge acquired within ENSAE.