Internship report
2nde year SICOM 2018-2019

Super resolution of audio signal using deep learning

Joachim Rosseel

Tutors:
Benjamin Ricaud: benjamin.ricaud@epfl.ch
Helena Peic Tukuljac: helena.peictukuljac@epfl.ch

Summay:

Glossary:

STFT: Short Time Fourier Transform
MSE/mse: mean square error
FSDD: free spoken digit dataset
MAE/mae: mean absolute error
SGD: stochastic gradient descent
EPFL: ecole polytechnique fédérale de Lausanne

Figure list:

Lab presentation:

The LTS2 lab is a team of researchers working within the EPFL. The lab welcome a large variety of people, such as PHD student, master thesis student, and also post doc scientists. They are specialized in data processing, especially audio, image and graph representation. Their main tool to achieve their goal is the deep learning. That is why I have chosen to do my internship in this lab.

Thanks:

I would like to thanks the lab team. I am using this opportunity to express my special thanks to my tutors who in spite of being very busy with theirs duties, took time out to hear, guide and keep me on the correct path and allow me to carry out my project.

Introduction:

The expression “super resolution” names a method which aims to reconstruct a deteriorate signal by improving its quality and removing major flaws. Therefore, by removing or correcting defaults, informations which has been lost during the deteriorating process can be retrieved. A classic example of super resolution problem is removing noise or fuzzy from an image.

In this subject, instead of dealing with deteriorated images, reconstruction methods will handle filtered spoken digits from different speakers. Further more, an imposed tool in these methods is the used of deep learning. Nowadays, with more and more powerful computer, deep learning is a widely studied area and it has already been tested that a deep learning neural network can learn to remove flaws on an image. So, it would be interesting to develop this with audio, in order to improve communication quality for example. In our case, the deteriorating will be made by removing a frequency band.

However, even if networks called autoencoder classifying spoken digits already exist, the idea to apply a super resolution method using an autoencoder on them as on image is quite new. Indeed, autoencoders that can learn how to reconstruct an audio signal and give a listenable output for a human ear with a simple and quite efficient method are really rare and new. Why? Because, if you work in the frequency domain, phase must be reconstructed and it appears that networks have trouble to learn it. However, phase is crucial to have a pleasant audible output. Also, the human ear perceives an audio signal on a db scale so it would make sense to have a higher resolution in lower frequencies (under 1kHz) than in the higher ones. It implies more preprocessing and thus complicates our problem.

So the subject proposed by the LTS2 lab is to develop a quite simple method with an autoencoder which can improve the quality of a deteriorated audio signal and tackle the previous explicated issues. In order to accomplish those purposes, the data fed into the autoencoder must be preprocessed in a way that it can easily learn. Also, network output has to be audible, either directly or by going through a post processing method. Thereby, an innovative super resolution method and autoencoder must be created and implemented into python. So this report will show our different approaches to fulfil our goals.

This report is composed of five part. The first one aims to recall you what is an autoencoder and to describe you the known means used. Then, the three following parts will expose three different approaches and autoencoders. In each part, results will be discussed with a detail explanation of them. In the last part, it will be pointed out how to combined those networks to have greater performances. At the end of the report, a conclusion will also be present.

The project is available at the git hub link of the reference (0). Thus, you can check the code as well as a file called test_notebook which allows some testing on your own.

I) What is already known?

The purpose of this part is to provide a list and a description of the already known tools implemented in this project. There are also some explanations for unused persons in deep learning in the first section.

1)Autoencoders

An autoencoder is a convolutional neural network which implements a compression and decompression algorithm. Usually, autoencoder are lossy during the compression and reconstruction parts which means that the output will be less precise than the input, like a mp3 compression algorithm. Also, as a neural network is involved, an autoencoder will learn without human intervention and also only works with a specific set of data. If an autoencoder is trained to reconstruct low pass filtered signals, it will perform badly to reconstruct high pass filtered signals.

The figure n°1 illustrates a classic autoencoder architecture.

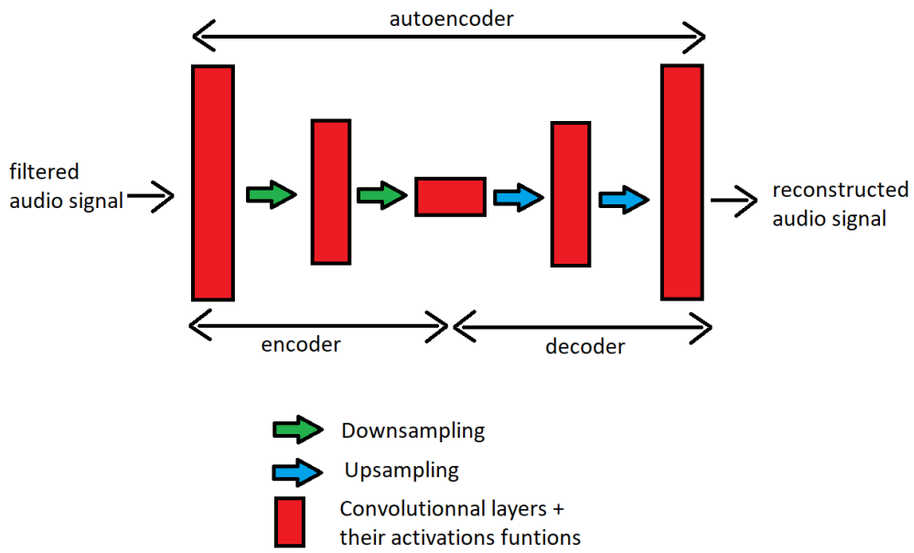


Figure n°1: a classic autoencoder architecture
(cf (1))

Basically, the convolutional layers are filters that learn to recognize some parts of an audio signal. The compression is made by the encoder which needs max pooling operation. It takes the highest value in a part of the a matrix representing the signal. The decoder re increases the dimension by doing upsampling operation which interpolates the added values.

Also the middle of the autoencoder, where dimensions are the most reduced is called the bottleneck.

As it can be seen, our inputs are bounded to the filtered signals and outputs to the unfiltered ones. Moreover, for the training, the autoencoder output is set to be the unfiltered signals.
It will be always like that in the project because during the training, an autoencoder will try to adjust its interns parameters so that its output matches the true wanted output, here the unfiltered signal.

The implementation of an autoencoder involves the keras library (cf(2)) in python which possesses all the operations explained in this section.

2)Spectrogram

It is generally acknowledged that the Fourier transform is a useful tool to study audio signals. However, in this project, as our spectrums evolve during time, the STFT is needed because the STFT represents the evolution of the frequency spectrum according to the time.

The spectrogram is just the squarred module of the STFT. The figure n°2 gives an example of a spectrogram.

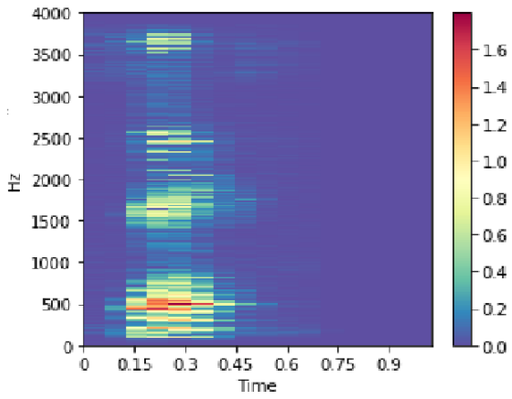


Figure n°2: spectrogram of a number six digit

In this project both of these tools are implemented. In deed, you can visualize the frequency spectrum of a signal and, as ear is sensitive to those frequency, it generally helps to understand what you listen. So it gives a new mean to interpret data. Further more, learning an audio signal amounts to learning its STFT or its spectrogram plus its phase. That's why, working in the frequency domain is an idea worth to be explored and studied.

Also, in order to come to grips with the lower frequencies resolution problems, the mel spectrogram, which calculates the frequency scale on a log one, quickly joins the project.

Since our project is coded in python, the librosa library (cf (3)) of python is widely exploited. This library allows to calculate STFT, mel spectrogram and to do many operations on audio signals or spectrograms.

3)Used data set

Currently, the dataset used for training and testing our autoencoders is composed of spoken digits from the FSDD dataset. There are 2000 spoken digits which 1800 are fed for training networks and 200 for testing and error measurement. The digits vary from 0 to 9.They are pronounced by 4 differents speakers in English. They are sampled at 41 kHz and the maximun frequency reachable for a human voice is 4 kHz. So Shannon is well respected.

This dataset was taken from a git hub repository called Free Spoken Digits Dataset (cf (4)).

Moreover, some functions of this repository inspire the means implemented to handle the dataset. Also, the reference (5) highlights some leads on this subject.

Also, some of the functions of this project uses the python library for filtering a folder containing an untouched data. Those filtered digits will be of course used as an input of our autoencoders. Especially, filtfilt is utilized to filter the digits. You can check the reference (6) to understand what this functions does in detail.

In more, the errors on the performances will be measured according to the cut off frequency of a low pass and a high pass filter. More points are taken in the low frequency and then after 1 kHz the spacing increases.

For the low pass the points are (in Hz):
{4000, 3000, 2000, 1500, 1250, 1000, 900, 800, 650, 500, 400, 250}
For the high pass, the points are (in Hz):
{100, 250, 400, 500, 650, 800, 900, 1000, 1500, 2000, 3000}

Further more, if spectrograms are going to be involved, it could be interesting to see the energy repartition of the data according to frequencies. That is why the figure n°3 is plotted. It could also underline the choices for the error measurement.

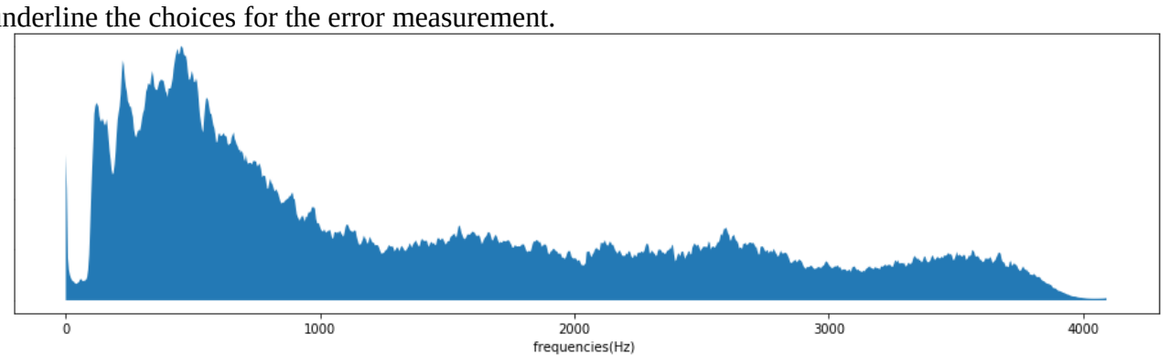


Figure n° 3: energy histogram of our dataset according to frequencies

By looking at the figure n°3, it could be affirmed that the majority of energy is located before 1kHz. That means that the harmonics with an important amplitudes should be located before 1 kHz. Thus, from this point of the report, the high frequencies/ low frequencies limit is defined as 1kHz

4)Error measurement

In order to evaluate the performances of our model, many error measurement methods can be employed. The mean square error is one among them. It was chosen because it has already been demonstrate to be accurate to estimate the error between images or audio signals, such as in adaptative filtering for example. In more, that proves to be easily computable in python.

The final MSE is an average one over the 200 test digits. Because all of the digits do not have the same amplitudes, each digits involved in the calculation is normalized by its L2 norm. It will prevent biased error results.

During this report, the MSE curves between audio signal and mel spectrograms will be given to you as a part of the results. The mel spectrogram used instead of the normal one because the mel really reflects how your ear listens.

However a good and easy way to evaluate the likeness between two audio signals is to directly listen to them. It permits to contrast with the MSE result. Actually, several error metrics such as MAE were tested but the MSE was finally retained because it was providing the results that makes the most sense with the listening.

Also, trough the report, many figures, such as spectrogram, will allow an analysis of listening and MSE results.

As the known basis has been introduced to you, the report will now delve into how those tools are set up by our team to try solving our problem trough different autoencoders. The first one is the real/imaginary autoencoder and will try to solve the phase retrieval problem.

II) Real/Imaginary autoencoder

1)Why this idea

The point here is to feed the network with the digit STFTs. Indeed, as explained on the Spectrogram sub section (page), learning the STFT will assure in theory a reconstruction of the correct harmonics. So by inverting the output STFT you can have a directly listenable signal.
The next figure n°4 presents you mel spectrograms of the input of the network and of the expected signal as a desired output.

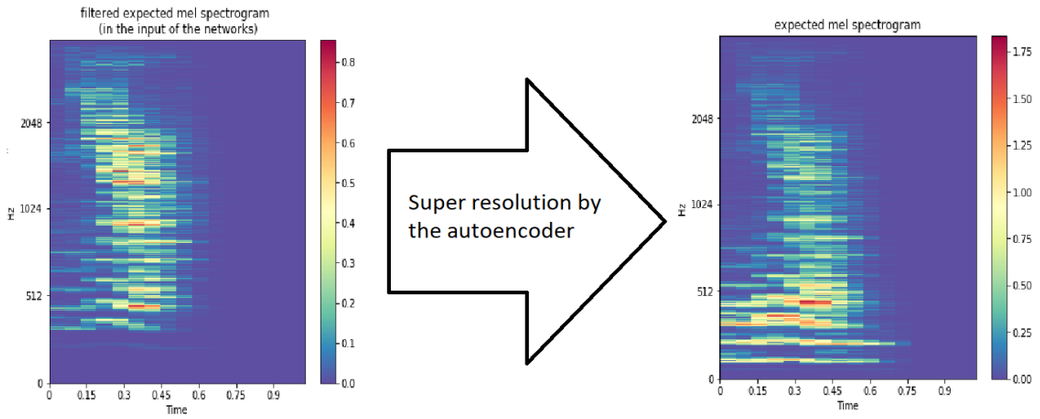


Figure n°4: expected super resolution process by our autoencoder on the magnitude

For this example, the input is a zero pronounced digit, filtered by a highpass butterworth filter at a cut off frequency of 800 Hz.

So, the main supposed advantage of this idea is that the phase doesn’t need to be learned as it is contained in the real and imaginary.

However, the STFT is a complex array. Thereby, a network can not learn it because an autoencoder learns to reconstruct by minimizing a loss real value function between a current output during the training and the final expected output. To tackle this, the trick is to look at the STFT as a 2D image with of two colours represented by the real and imaginary part. The axis of this 2D image will be (frequency, time). So finally, the audio super resolution problem amounts to an image super resolution problem, a more known subject.

That's also why this way of interpreting the problem seems interesting. So now let's explain how this idea was implemented during the step called the preprocessing.

2)Data preprocessing

Firstly, a point which may be useful to remember throughout all the report is that all of our preprocessing function take a path containing wav file and converts each wav file into an array. Then it applies whatever is needed on this array and return a global array where each wav file correspond to an index. The principal asset of doing that is that the wav files stay untouched and so no need to store a huge amount of them for each networks. Also, this kind of global array is required by the function fitting the autoencoder.

Moreover, every time a preprocessing method is mentioned, it should be understood that it is applied to each wav file.

In order to realise the previous cunning, the preprocessing function separates the real and imaginary part of the STFT and then stores it into an array of shape (1024,16,2) with the number 2 being the real and imaginary part "colours". The STFT is calculated with a hanning window and a FFT window size of 2048. The window length is equal to n_fft and the hop_lenght is n_fft/4. The complex number are coded on 64 bits and the STFT function has center equals to True and a reflect pad_mod. Those parameters for the STFT calculation will remained unchanged during all the report.

This function also resample the input audio data with a sample frequency of 8 kHz. Shannon is still respected and in fact few informations are lost during this process. It allows a lower calculation time than with 41kHz as a sample frequency. Since all the signal does not have the same duration, it also does zero padding in order that all signals last 1 second. So the dimension will be all the same for every audio signal and the autoencoder will not oppose any dimension issues.

Each spectrogram is normalized by their maximum. It should help the autoencoder to learn properly.

Now that the data is ready to enter into the network, the architecture of the autoencoder can be exposed in next sub section.

3)The network

The following schema (figure n°5) illustrates the developed and final tested network for this section: (remplacer spectrogram par stft)

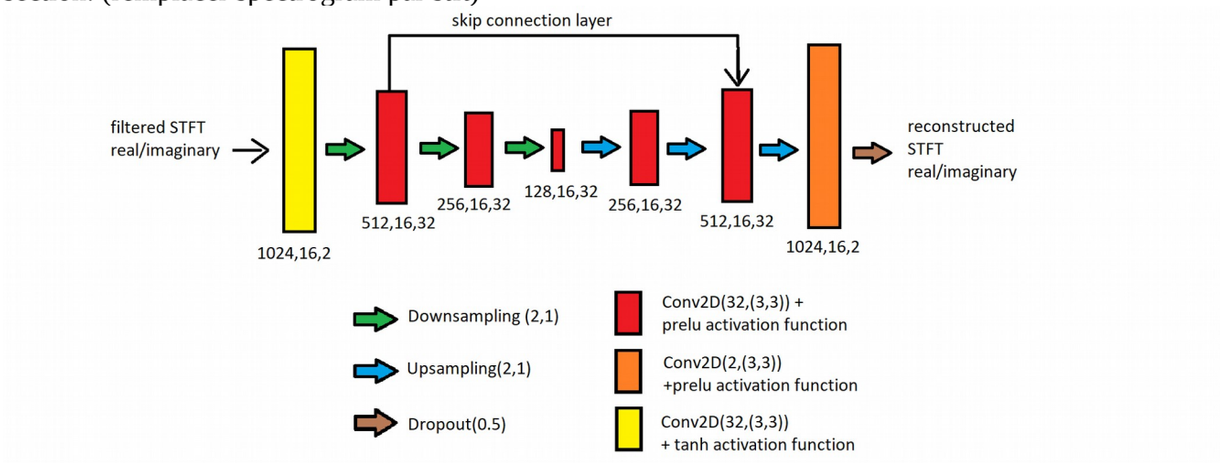


Figure n°5: real/imaginary autoencoder

As you can see, the network possesses a skip connection layer. It allows the layer to pick some informations from where the merge is. As you might guess, useful information is also lost when the

downsampling occurs. If too much information is lost during the process, it could be helpful to use the skip connection layer to retrieve some lost details in the untouched frequencies.

The final dropout layer only keeps half (due to the 0,5 parameter) of the nodes for the last layer. It should avoid overfitting and help the network to learn.

Also, Prelu is used as the activation in order to deal correctly with the negative numbers. Tanh basically do the same thing but the low values are more separated and the high one tend to be regrouped. It is used here because the majority of the values in the spectrogram are quite low. It should make the training easier.

The compilation is done with an Adadelta optimizer (learning rate of 10) and a MAE loss. The data is also multiplied by a gain of 10. This may seems to be high number for a learning rate but with lower number, the loss is almost constant. All of this shall be discussed with more details in the section right after.

The time dimension is not reduced because 16 is a quite low number comparing to 1024 and by reducing it, the network risks to not be able to reconstruct properly.

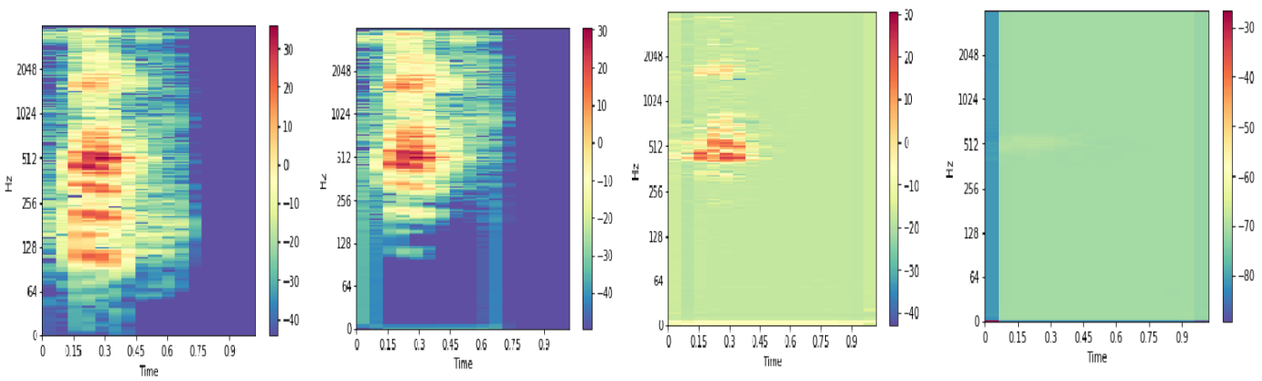
To train the network, batch sizes of 16 are created. Also the parameter shuffle is equal to True. Those parameters are supposed to help the training.

Now that the network has been set, its results can be discussed in the next sub section. The post processing will also be described.

4)Results and analysis

First of all, the output data is post processed to be listenable and displayable on a spectrogram. To do that, the complex spectrograms are reformed by merging all the real part and imaginary part into the complex they represent. Then, an invert STFT is applied to go back to the time domain.

The results here are quite insufficient. Indeed, the network seems to have trouble to learn effectively, even after several tests by changing parameters such as the loss for example or the network architecture. To illustrate what is going on, let’s have a look a the figure n° 6.



expected output | filtered expected output | output with a skip connection layer | output without a skip connection layer

Figure n°6: example of the output mel spectrogram of the real/imaginary network with a skip connection layer and without it, after 40 epochs of training.

The digit (a six) is filtered by a 4 order high pass butterworth, cutting at 500 Hz. The figure n°5 points out that the network is not able to learn anything without a skip connection layer. In deed, the picture does not fit at all with the expected output. With the skip connection layer, the network succeeds to pick the most important harmonics from the filtered signal, but it is degrading it without retrieving any lost informations. Those majors flaw greatly impacts the listening.

But why does not the network learn? A possible and very probable reason is that the values representing the spectrograms are too scattered. Several values are near 0 with ten power of -5 or -4 whereas the not so numerous important harmonics are near 1 due to the normalization. So minimizing the loss function become a tough task.

Despite many tests in order to solve this, the idea was finally abandoned. Successful classification dealing with audio signal have already been conducted with autoencoder using only spectrograms and so the next part intend to present the work performed on these.

III)Mel and Lin Spectrogram autoencoders

1)Why this idea

In order to have a dB scale learning on the frequency, and by doing that a higher resolution on the lower frequencies, it could be interesting to focus on mel spectrograms. That's why this part will delve into the study of an autoencoder dealing with them. In parallel of this, an autoencoder processing linear scale frequency spectrogram will also be implemented. The purpose of this one is to compare the performances of both of these networks. Thus, it would point out if the mel spectrogram is worth it.

The expected results are similar to the ones expected for the real/imaginary network. That means that it follows the same idea of giving an easy interpretable output using spectrograms. So it should be seen if a harmonic super resolution is possible.

As it was made during the part II, the next sub section will explained the preprocessing.

2) Data preprocessing

The functions which preprocesses the data are inspired from the preprocessing function of the real/imaginary part. Indeed, those functions applied a resampling at 8 kHz following by zero padding. After that, all audio signals have a shape of 8000 samples. Then the STFT is calculated with same parameters than in the sub section II)1). The only differences between those two functions is the computation and storage of the mel spectrogram for the mel one in an array. However, both store a spectrogram in an array of shape (1024,16,1).

It may also be precise that the mel spectrogram is calculated from the normal STFT. The librosa function computing this has the same parameters than the STFT function. Just some others parameters are setted such as the sample rate which is 8000, n_mels which values 1024. Also, to respect the definition, the power of the module is equal to 2. For this report, mel spectrograms are always calculated with those parameters, in particular for MSE measures.

Moreover, to avoid that the network may not learn, the following function is applied:

$$Final\_spec = \log_{10}(abs(S)+1)$$

with S a mel or lin spectrogram.

Final_Spec is the spectrogram that will be returned to the function and fed into the autoencoder if it's from a filtered signal. Thanks to the log, the amplitudes values will be much closer than with the real/imaginary part and so a minimum to the loss function could be find in an easier and quicker way than before.

Yet, the phase is lost during this process. So, the phase is also returned as a second output of those function. The shape of the complex array containing it is the same as the spectrogram. Thereby, by multiplying a spectrogram with a linear scale frequency and linear amplitude, an invert STFT can be applied and an audio signal should be listenable.

Now that the data is ready to enter into the network, the architecture of both autoencoders will be exposed in next sub section.

3)The networks

The figure n°7 shows the final mel spectrogram network enabling the super resolution and the figure n°8 the final lin spectrogram.

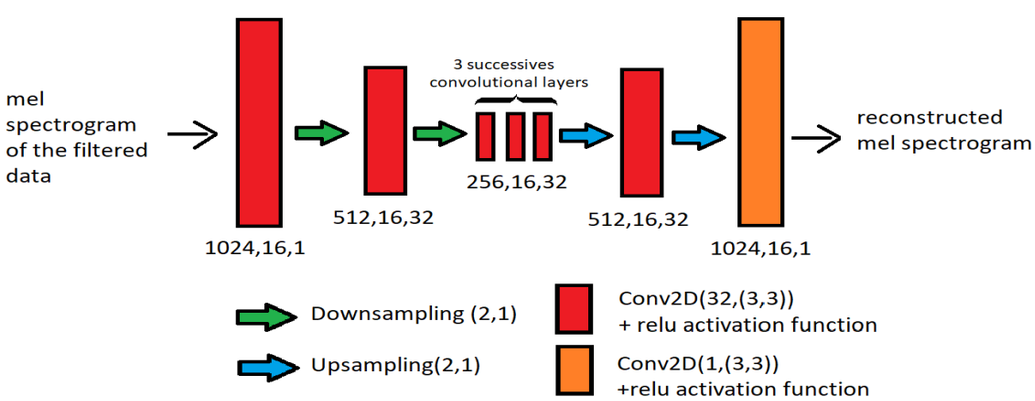


Figure n°7: mel autoencoder

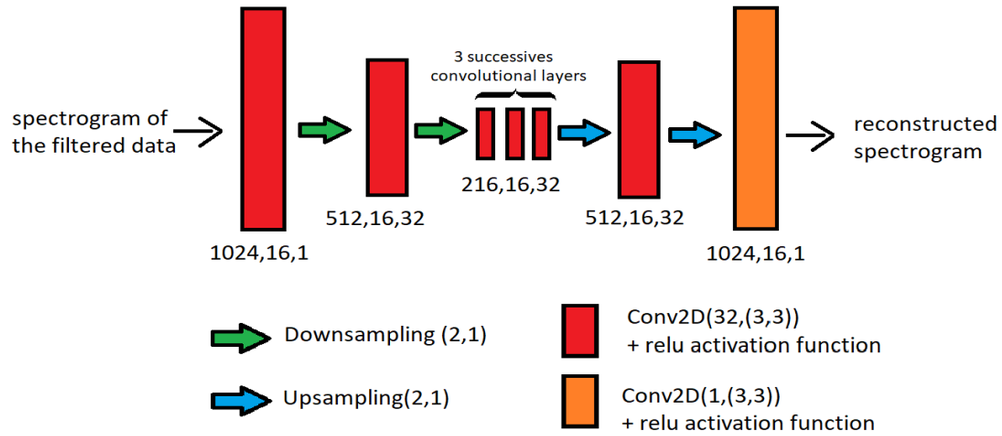


Figure n°8: lin autoencoder

Both of those network are compiled with a MAE loss in order to avoid forgetting small errors during the training. The Adam optimizer was also implemented for both of them. This optimizer is a better version of the SGD because it has an adaptative learning rate. It is initialized with a learning rate of 0,001. The training is accomplished with batch size of 20 and the parameter shuffle is set to True.

Tests with a time dimension reduction were made and the results of both autoencoders were less satisfying. Thus this dimension remains untouched. Also the kernel of the Conv2D, (3,3), was changed to (n,n) with n superior to 3 but the results remained the same. Increasing the kernel size adds more parameters to play with for autoencoders but it could lead to overfitting.

You might be surprised that both of the networks have the same architecture and only differ with the kind of spectrogram processed. That's because the purpose of the comparison is to evaluate which one of the mel or linear spectrogram is better. If a parameter of one of the autoencoder changes, it will have to be taken into account. So the comparison will be biased by this parameter and will lost its interest.

The bottleneck is composed of three successive convolutional layers because this number appears to facilitate the training and give better results.

Another point to note is that both networks are also simpler than the real/imaginary autoencoder. That means that it takes less time to train. That is another argument to abandon the part II idea.

Now that every parameters is noted, performances of each network in term or training and reconstruction shall be the theme of the next sub section.

4)Results, comparisons and analysis

Firstly, the next figure n°9 and n°10 demonstrates that both of the autoencoder are learning quite well. Indeed, the validation loss and loss values follow a classic good decreasing trend according to deep learning standards. That means that the networks are not under fitting or over fitting. So the outputs spectrograms should be closer to the expected one than the filtered ones. In other words, the super resolution of the harmonics may have worked.

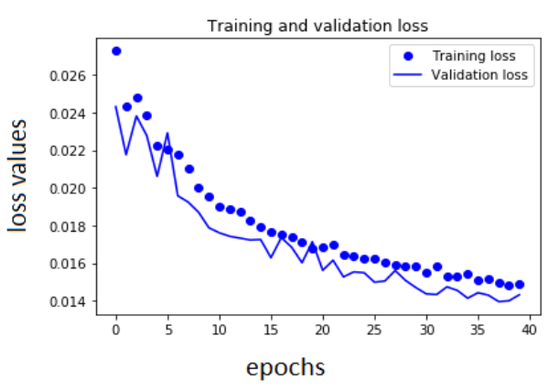


Figure n°9: mel autoencoder loss

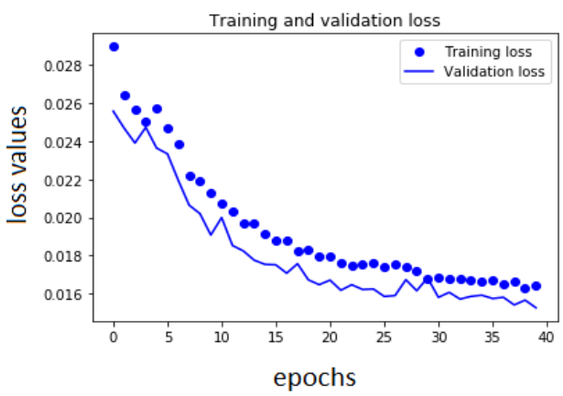


Figure n°10: lin autoencoder loss

For those figures, the input data is filtered by a 4 order butterworth highpass, cutting at 1,5 kHz. You can also observed that both network trains with a similar speed and accuracy. As both data belong to the frequency domain, that means that the training speed or accuracy will not be involved in the choice of the best one.

The post processing consists of getting back a linear amplitude, and combining spectrograms with their appropriate expected phases in order to have an audio signal. Also, in the case of the mel spectrogram, the post processing requires to come back to a linear scale frequency. Thus, it involves a matrix of the base change. Currently, the only way of doing that is to do a pseudo invert of the linear to mel base change matrix.

The two following figures n°11 and n° 12 show you two examples of a reconstructed spectrograms by each networks:

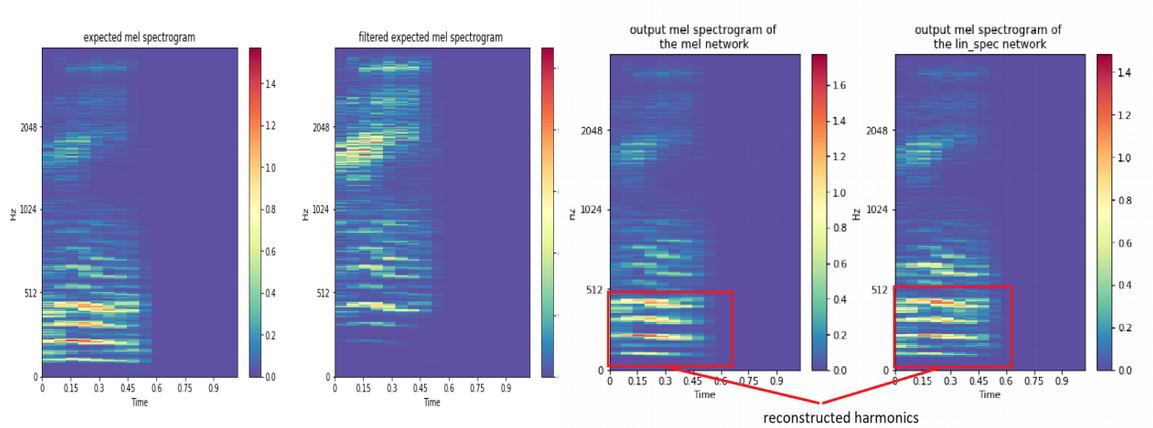


Figure n°11: reconstruction of a three digits filtered by a 4th order highpass butterworth, cutting at 800 Hz (the colorbar is the amplitude scale)

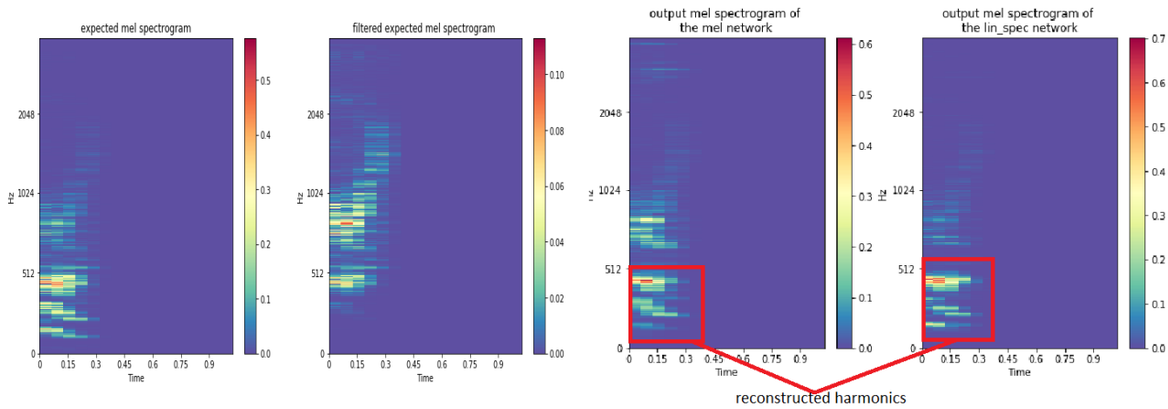


Figure n°12: reconstruction of a four digits filtered by a 4th order highpass butterworth, cutting at 800 Hz (the colorbar is the amplitude scale)

First of all, the two previous figures clearly illustrate that both autoencoders succeed quite well in finding back the filtered harmonics, although some of are completely missing. Moreover, it demonstrates that the networks can keep and give back the higher frequencies given at the input, without too many errors.

To generalize our comparison, MSE must be discussed with the help of figures n°13 and n°14.

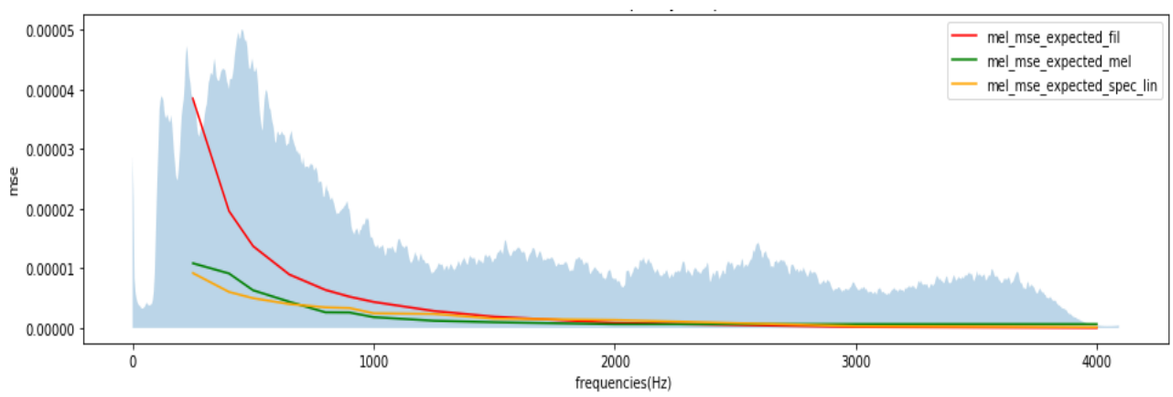


Figure n°13: evolution of the expected/filtered, expected/mel, expected/lin mel spectrogram mse according to the cut off frequency of a low pass filter

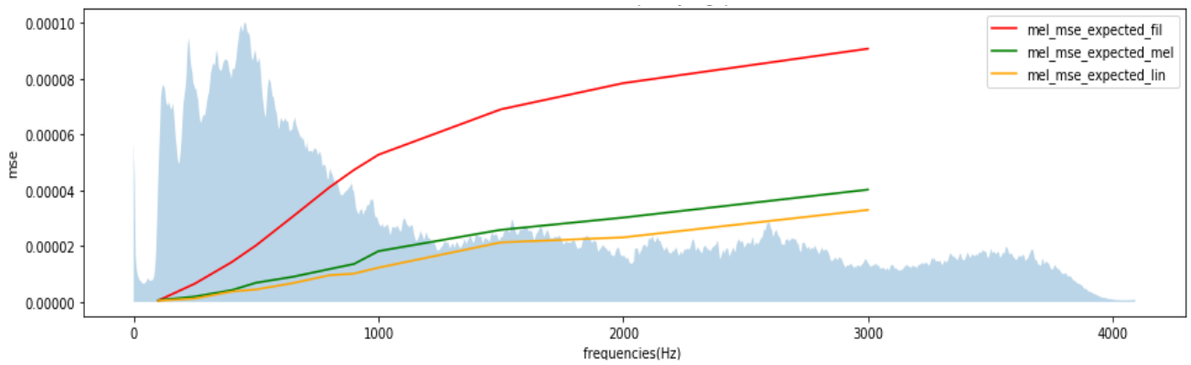


Figure n°14: evolution of the expected/filtered, expected/mel, expected/lin mel spectrograms mse according to the cut off frequency of a high pass filter

The filter which deteriorated the data is a 4 order butterworth in both case. The blue waveform in the background of both figures is the energy histogram of the spoken digits database seen in the page 5616. The amplitudes values of this one are adjusted to fit with the curves and the histogram itself is here just to give an understanding tool. Thus, those values should not be taken into account. However the harmonics amplitudes of the spectrograms are of course directly bounded to this histogram.

Let’s just add a little precision before moving to the analysis. To measure the MSE, the networks need to be trained for each mse values found. Indeed, autoencoders are data specifics and two same original signals but filtered at two differents cut off frequencies become in fact too much different. Thereby, both networks are trained 40 epochs at each survey frequency. Now, let’s explain the figures.

Firstly, they shows that both autoencoders improves the quality of the signal by reconstructing the harmonics. The filtered/expected error is either higher or at the best equal. It justifies what was claimed with the spectrograms results.

Secondly, the curves confirm a quite intuitive thought. Indeed, the more information is lost by filtering, the harder it becomes to retrieves all of it and precisely. It appears clearly thanks to the energy histogram. As the cut off frequency of the highpass increases, most of the energy disappears and our autoencoders get a tougher and tougher task of reconstruction. That is why all mse curves tend to grow in the figure n°14.

Regarding the figure n°13, one could seen that the mse are closed, with a low pass having a cut off superior to 1500 Hz. By looking at many spectrograms, it could be deduced that the committed errors by the autoencoders across spectrograms offset the error due to filtering, because of the lack of energy in high frequencies. After this point, the missing major hamonics (located before 1 kHz) increases the MSE values between the filtered and the expected so much that it become higher than the two others.

In both case, reducing the MSE if too much information is missing just requires more epochs to train the networks.

Thirdly, with both filters, the mel MSE is either higher or equals to the lin MSE. By observing many spectrograms, those trend can be understood. Actually, the lin autoencoder is much precise in high frequencies and manages to counter its imprecision in the lower frequencies. Although the mel autoencoder is designed to perform well in lower frequencies and not the lin autoencoder, this last one appears to be barely less precise than the mel in the low frequencies in several examples. Further more, the mel autoencoder is struggling with recovering high frequencies due to its frequency scale.

So, the lin autoencoder is more precise in high frequency and is a little less precise than the mel autoencoder in low frequencies. That is why in the figures n°13 and n°14, the lin curve grows slower than the mel curve. However, both networks have trouble with finding back the harmonics with a too small amplitudes in the higher frequencies. This issue must be tackle because even if those are minors harmonics, they will become important when it comes to listen to the digits since this default should be audible. The main reason why those harmonics are forgotten is because of the networks. It seems that to minimize their respective loss function, they tend to focus on the reconstruction of the harmonics with a significant amplitudes. Indeed, errors on them are the principal contribution to high loss values.

Finally, after postprocessing, many reconstructed digits were listened. The super resolution is clearly audible. In more, it just confirmed what has been explained during this sub section. The lin output is closer than the mel output to the expected digit when the curves are clearly separated. Also, it's equivalent where the curves are almost identical. For example, when the 3 curves are superimposed on the figure n°13, one could not distinguish clearly from listening the filtered, the mel reconstructed and the lin reconstructed one from another.

To resume this part, both networks provide quite good results and the lin autoencoder have generally better performances in harmonic reconstruction than the mel one. They tend to have some flaws with too tenuous high frequency harmonics. Further more, both networks lost the phase during the preprocessing. So without knowing the expected audio signal, it is impossible to listen to the output. So the next part will deal with a network having a directly listenable output.

IV) Audio autoencoder

1)Why this idea

Although the results were generally satisfying in the previous part, the output was not listenable. So the idea here is just to simply learn to reconstruct an audio signal by giving to the autoencoder an audio signal at the input. Thus, the autoencoder will furnish an audible output.

However, as the autoencoder does not directly treat with the harmonics, it could be expected a messier spectrogram than in the previous part for the reconstructed signal. That means finally an audible output but a bit distorted.

It should be explained the data preprocessing now as the main idea is introduced.

2)Data preprocessing

The preprocessing here is quite simple. First an audio signal is resampled to have 8000 samples. Then, it is normalized by its maximum in order to facilitate the learning. The maximun is stored into an array and returned by the function. Thus it can be multiplied back to the output. Finally the audio signal is stores into an array of shape (8000,1,1) and returned. As usual, the preprocessing function applied this to all the wav file present in a folder.

Let's now move to the network architecture.

3)The networks

The next figure n°15 puts in picture the audio network

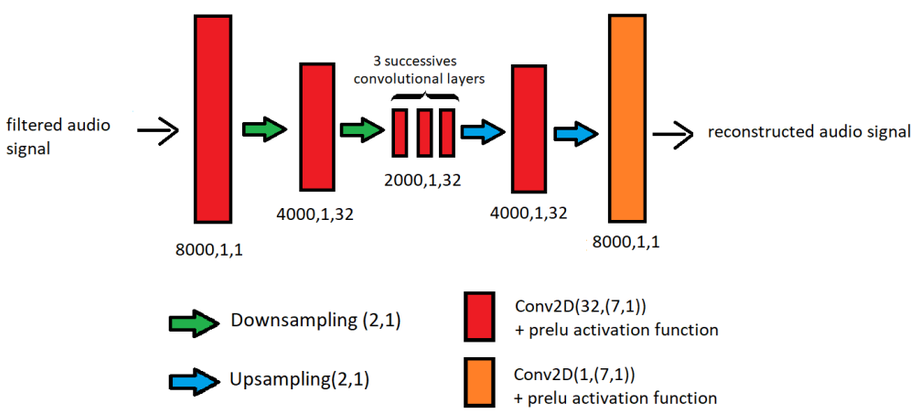


Figure n°15: the audio network

Since the architecture of the lin and mel networks has proved to be successful, the audio network is just an adapted version for the dimension of audio signal. It is also compiled with a MAE loss and Adam optimizer with a 0,001 learning rate. The training is implemented with batch sizes of 20 and the parameter shuffle is set to True.

The number 7 in the kernel is not taken by chance. In fact, results get better with 7 instead of lower integer. However, with a higher one, it just increased the training per epochs. But, this is counterbalanced by a reduction of the required number of epochs to have an acceptable output. So (7,1) is kept as kernel size.

Prelu is used because an audio signal can take negative values in time domain.

As usual, let's now analyse the results and compare them with the previous autoencoders of the part III.

4)Results, comparisons and analysis

First of all, the next figure n°16 serves as argument to assert that the audio autoencoder learns properly. Those curves are plotted with an input filtered data by a 4 order lowpass butterworth.

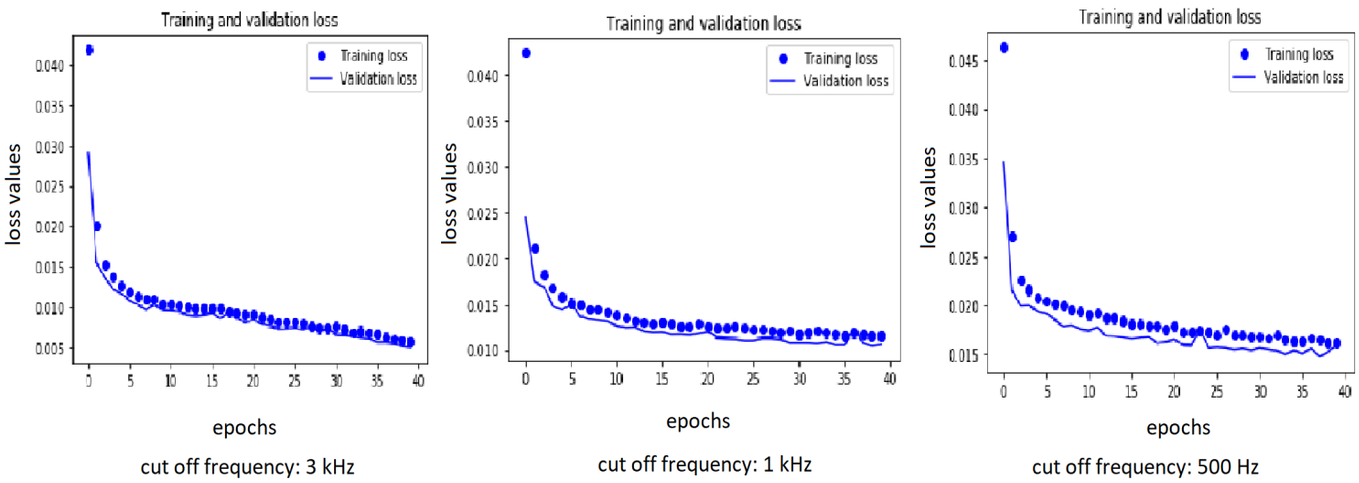


Figure n°16: training loss and validation loss for different cut off frequencies

As the spectrogram really allows a good comprehension of what one can hear, the post processing function calculates the mel spectrogram of the audio autoencoder output in the same way it was preprocessed with the mel network. In parallel, the output is re multiplied by the max of the corresponding entry. Then, the output is converted into a wav file and you can listen to it.

The following figure n°17 give you an example of a mel spectrogram produce by this post processing.

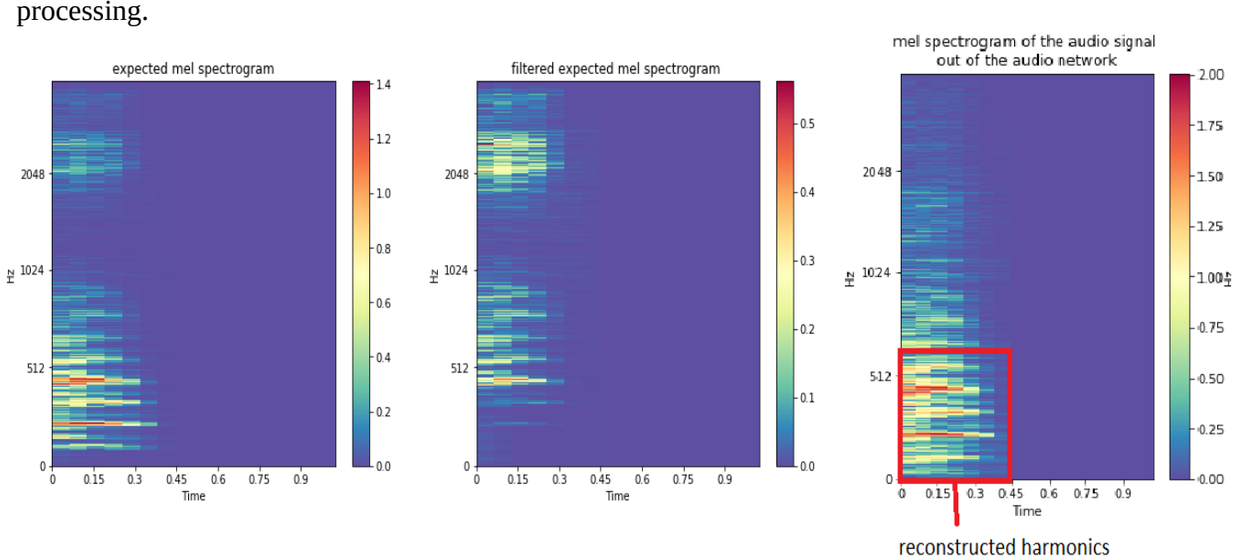


Figure n°17: audio network mel spectrogram reconstruction of an eight digit filtered by a 4th order high-pass butterworth, cutting at 800 Hz

This figure clearly exemplifies how the audio autoencoder seems to reconstruct the harmonics by means of an audio signal. The good news is that the low frequencies are quite well reconstructed with a high pass. It is of course less precise than the mel or lin autoencoders which directly learn the harmonics. However, an important flaw is that the audio autoencoder turns out to generally be unable

to reconstruct the high frequencies as you can see on the figure n°17. To illustrate these claims, the mel mse are plotted like in the part numero with the figures n°18 and n°19.

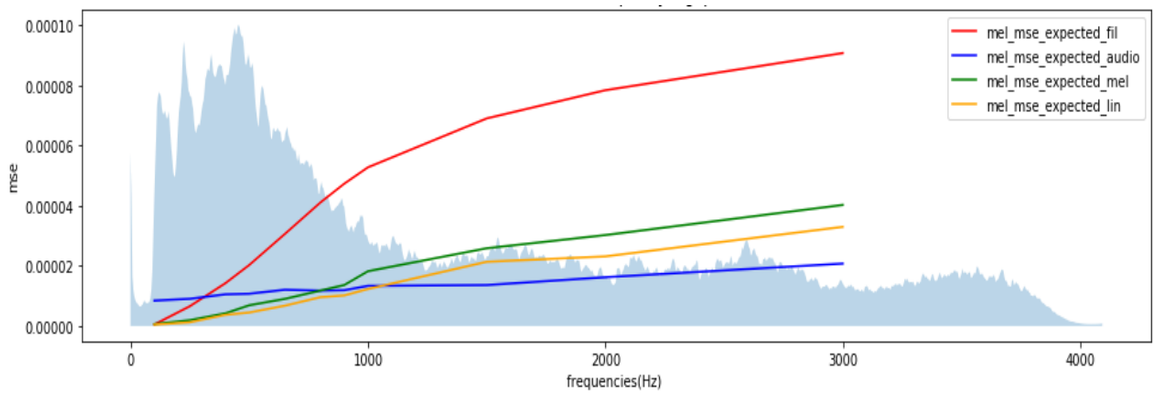


Figure n°18: evolution of the expected/filtered, expected/mel, expected/lin, expected/audio mel spectrogram mse according to the cut off frequency of a high pass filter

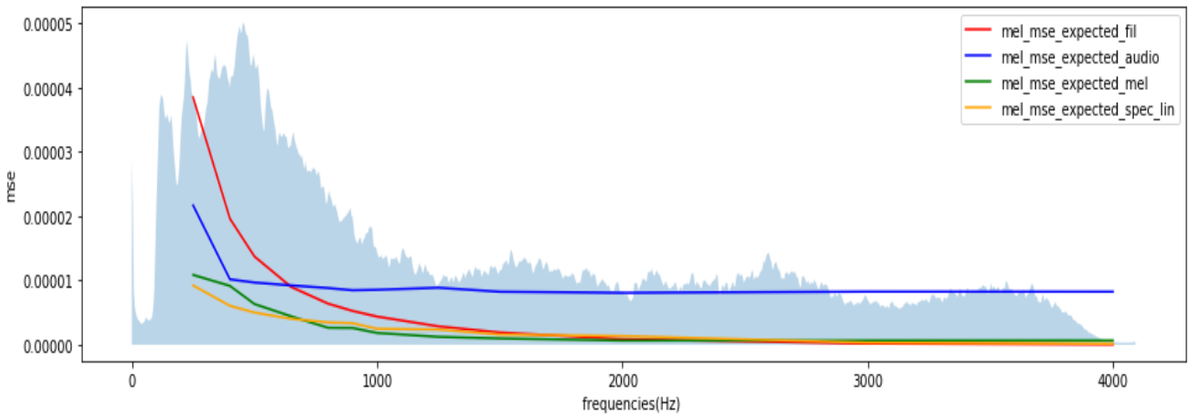


Figure n°19: evolution of the expected/filtered, expected/mel, expected/lin, expected/audio mel spectrogram mse according to the cut off frequency of a low pass filter

The filter is a 4 order butterworth. As you can observe, the audio mel error higher even with a signal almost not filtered in both figures. This is due to what has been said previously, the audio autoencoder can not provide a high frequency resolution. By seeing the energy histogram, it could be asserted that the lack of energy on those frequency is the main reason of this default. That is why it is almost constant with a low pass as the low frequencies remained untouched before a cut off of 1 kHz. In more, it appears that the audio network have trouble with reconstructing low frequencies if they are too much cutted by a low pass. The figure n°19 points out that fact.

Moreover, an interesting event happens with the high-pass on the figure n°18. The lin and mel curves cross the audio one. Actually, the mel and lin autoencoder are more sensitive to the lack of informations implied by the filtering. Of course, as its curve grows, the audio network will also perform more and more badly as the informations is more filtered. However, as it grows slower, the high frequencies errors will counterbalanced the ones made by the mel and lin autoencoders in the low frequencies as they struggle with a more and more distorted signal. Further more, the errors made in the low frequencies take a higher value as most of the energy is located here.

To support all of these explanations on the audio network, many digits were listened and it just confirms what was claimed. With a low pass, the sound of the audio network output become more muffled as the cut off frequency decreases. With a high pass and low cut off frequency, the lack of high harmonics is audible and then as it increases, the issues in low frequencies with the lin and mel network really contrasts with the audio network having a better low frequencies resolution.

To summarize this part, one could argue that the audio network effectively succeeds to do a super resolution with the great advantage of having a directly audible output. However, high frequencies are not well reconstructed. Thus, this impacts greatly the listening and clearly limits the performances. Moreover, the lin and mel autoencoders are genarally more accurate in harmonics recontructions. So the next part will present you how to merge both autoencoder advantages in order to improve the super resolution.

V) Audio/Mel and Audio Lin autoencoder

1)Why this idea

According to what has been said during the part III and IV, the lin autoencoder is more precise when reconstructing harmonics than the mel or audio one. Yet, this appears for the moment to be the most important point to focus to get closer to the original signal when listening. So it would be relevant to keep the reconstructed spectrograms from the lin one. As the audio network learns to somehow reconstruct the phase by processing directly audio signal, the phase of this output could be taken and combined with the lin reconstructed spectrogram. Thereby, it will permit to have a fully reconstructed STFT. Further more, by taking the advantages of both autoencoders, it may improve the listening of our direct audible audio output in theory. The same method is applied with the mel network to compare the listening with the lin and also the audio network of course.

As both previous part are just merged, the wav files will be preprocess the same way, respectively to each network.

Let’s now go straight to the next part. It will put into a picture what has just been developped.

2)Schema of the merging method

The next figure n°20 presents you the schema of the implemented idea during this part.

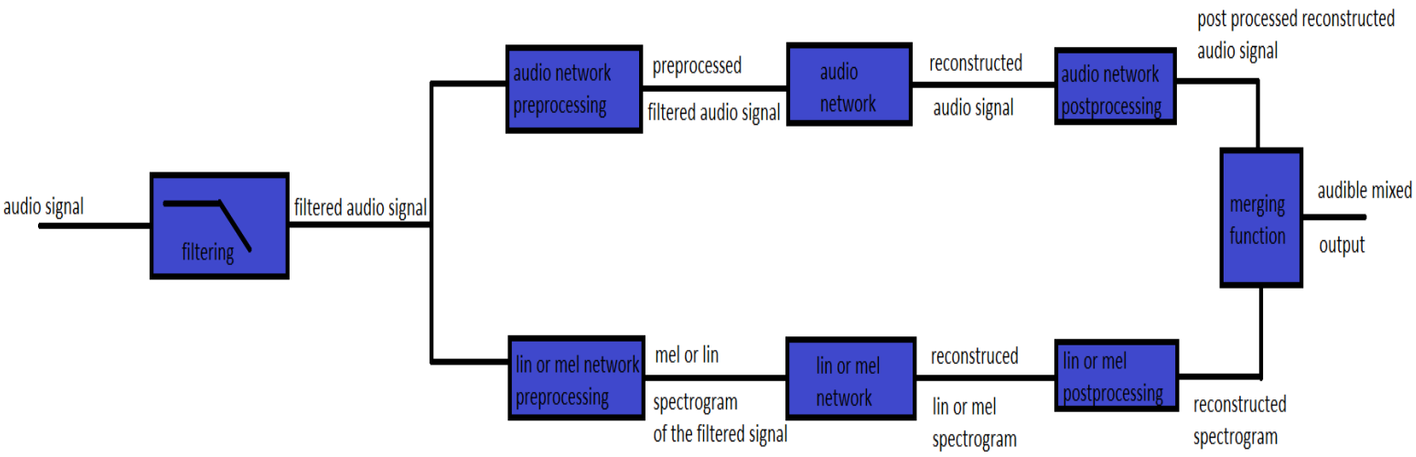


Figure n°20: block schema of the merging method

The merging function basically calculate for each reconstructed audio signal its STFT. Then it isolates the phase and multiplies it with the reconstructed spectrogram, which must have a linear frequency and amplitudes scales. Then it does the invert STFT with an imposed length of 8000 samples, corresponding to a duration of 1s. It returned the mixed audio signals.

With the lin autoencoder, the merging function also does the post processing for the spectrogram, which is to get back an amplitude linear scale.

Moreover, all parameters for the training explained in the two previous part are kept.

Let’s move to the results of the merging method.

3)Results

Since this method furnishes a fully reconstructed audio signal, MSE measures in the time domain were carried out. The purpose of those measures is to evaluate if this MSE bring a relevant error measure between two audio signals. It may be intuitive to think that if two audio signals are closed in the time domain, they will sound the same. That’s why the next figures will show audio signal in time domain. Moreover, as the merging method is compared to the audio network, the error between mel spectrogram doesn’t change as the mel or lin network is used for the merge.

The figure n°21 compares audio network alone and the merging method with audio signal for a reconstructed four. It was filtered by a 4 order highpass butterworth, cutting at 800Hz.

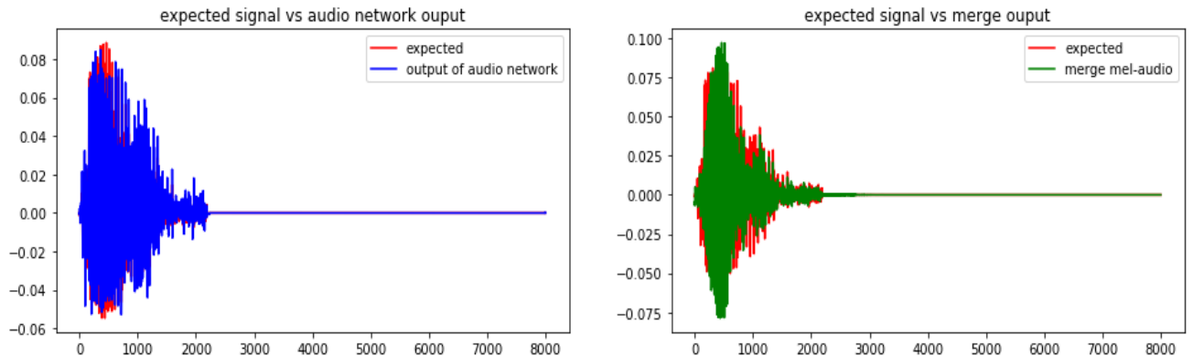


Figure n°21: audio signal comparison between audio network and merging method (the x axis scale is the sampling index)

As you can see, the reconstructed one seems quite similar to the original in both case. Like what has been done with specograms, the mse is plotted below with figures n°22 and n°23, in order to get a more global idea of errors made.

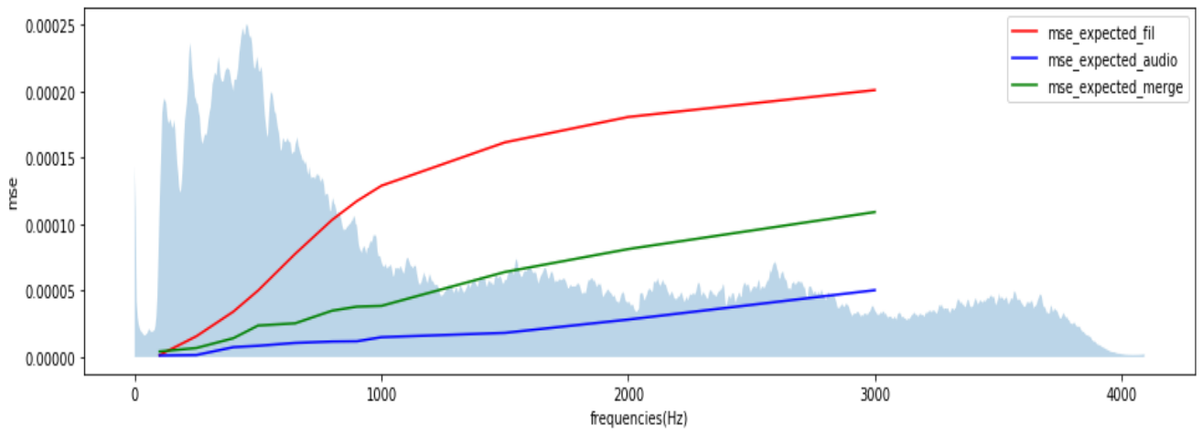


Figure n°22: evolution of differents network audio mse according to the cut off frequency (highpass)

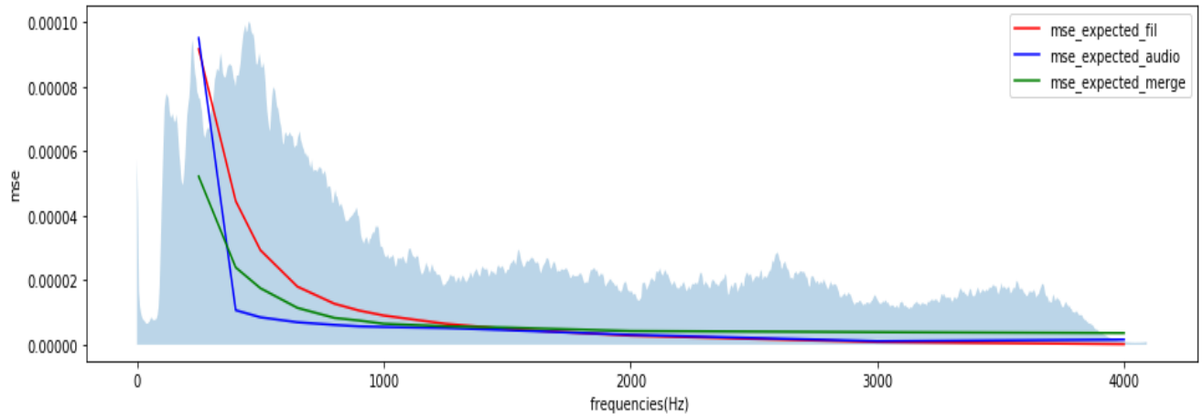


Figure n° 23: evolution of different networks audio mse according to the cut off frequency (lowpass)

Actually, as the audio autoencoder deals directly with audio signal, it is generally more precise. That's why its curve is generally lower than the merge one. To evaluate the relevance of these figures, listening is essential. What have been asserted during this report about mel MSE curves should also support the listening and thus our explanations.

Let's focus on the figure n°23. It was explained in the sub section IV)4) that the more the cut off frequency goes down, the more the audio network output has a muffled sound. The merge output clearly avoid this flaw and the listening is better. However the audio network mse is lower for the majority of the low frequency.

Let’s move to the figure n°22. It shows an audio mse curve always lower. However with a low cut off frequency, the merge audio is better and the listening quality coincides better with the mel mse. All those arguments confirms that this way of measuring the error is not accurate. It just measures an error between amplitudes which makes less sense for our ear. It is the mel mse error which really represents the quality of the listening.

Generally, the merging method, with lin or mel, produces better results than the audio alone. Also, as expected, the merging with lin is better than with the mel. In fact, it seems that by going from mel to linear scale during the post processing, it removes the precision in the lower frequencies as the listening is the same with a low pass having a cut off frequency after 1 kHz. Also, the lack of resolution in high frequencies is audible, especially with low pass cutting under 1kHz.

That’s why for the moment the merging method with the lin network get the better performances. However, the lose of low frequencies harmonics as the cut off frequency of a high pass goes up appears to be the major flaw and can be corrected. In fact, the frequencies after 2500 Hz tend to be poorly learned in several case, especially when the amplitudes is low. However, those harmonics are importants when one is listening. So, the next sub part will highlight some leads to come to grips with this issue.

4)How to improve it

This last sub section aims to give some indications to remove the explained flaws.

A first simple improvement could be just to take the low frequency spectrogram part from the audio network, after MSE mels curves cross in the figure n°18. By taking both networks advantages, it may prove useful to enhance the performances. However the main default with the high frequencies reconstruction will not be solved with this.

It could be gathered by all that has been said that the lack of energy in high frequencies is the main reason that they are missed by the networks. As the amplitude of the STFT is directly related to the energy, a gain could be applied to frequencies superior to 2000 Hz to give more importance to them. Then, during the post processing, this gain should be counterbalanced.

This method should be tested on the lin and mel network but also on the audio one. In fact, the audio one could be really promising. Indeed, the low frequencies are quite well reconstructed in the majority of case, whatever the filter or cut off frequency. This is because the energy is located here. Thereby, a compromise has to be find to give enough energy to the high frequencies but not too much in order to keep the low frequencies resolution and improve high one.

Yet, doing that with the lin network might be more secured. In other words, results with the lin network could be more predictable than with the audio network, since the lin network processes directly spectrograms. It could also offer the same results with the merging method in terms of listening.

Another method could be to add some preprocessing for the audio network. It aims at the same purpose as the previous solution. It consists of doing the STFTs and spectrograms of audio signal. Then a log scale one the amplitude is applied. After that, by inverting this new scaled spectrogram multiplied by the untouched phase, audio signal are again available and then fed to the audio network. The log scale goal is to distribute in a more equal way the energy among the harmonics. Thus, the network might take them more into account. Of course, inverting this log scale must be implemented during the post processing.

Also, I’d like to point out that replacing the FSDD dataset by music could be the final purpose. Actually, the energy spectrum for the music has the same appearance as the FSDD one but with a larger band of high energy area. So, after adapting the preprocessing and post processing to fit the new dataset, same results may appear.

In order to deal with the phase retrieval, another method exists. It is called the phase gradient heap integration. It uses the fact that the phase is bounded by a formula with the amplitude to calculate it. So this method is applied during the post processing. However, as it was quite complex and time starts to lack, it was not so much studied during the project. This method is used in the TIFGAN project (cf(7)) and this project could be a source of inspiration to realize this solution.

Finally, a totally new network work made in parallel by my tutors could get better performances. You can read about this idea with the reference (8). The idea is that the first layer of the network learns how to do the spectrogram from an audio signal. Then, the last layer should provide also an audio signal as output. For the moment, the first layer operates correctly and the network furnishes good results of audio classification. However, the last layer is still not developed to accomplish the return to the time domain.

Conclusion:

To conclude this project, it should be first remembered the purpose of this project. It was aiming to do super resolution on audio signals, using deep learning and especially autoencoders. Phase retrieval and the dB scale ear perception were the majors issues to deal with in order to go back to time domain.

Four different approaches were tested and their results were presented to you through this report. First, the real/imaginary network is directly trying to tackle the phase retrieval but it has trouble to learn and reconstruct harmonics because the data values are too scattered. Then, the lin and mel network succeed quite well and accurately to find back the missing harmonics. However, the phase is lost during this process and the high frequencies are sometimes forgotten. Finally the audio network furnishes directly a listenable output but it was demonstrate that, despite a quite good reconstruction in frequencies inferior to 2 kHz, it struggles with getting back those superior to 2 kHz.

The last approach, the merging method, regroups advantages from the lin or mel network and the audio network and finally gives a fully reconstructed audible output, better than the audio network alone. Moreover, the lin network is finally better than the mel one.

Also, one could underline the limits of the deep neural network. Actually, a large amount of time was spent on changing the networks to try making them work correctly, especially with the real/imaginary network. For example, the number of layers, the size of the convolution kernel, and compilation parameters were often modified and the network tested right after. That is not so miraculous as one could hope.

For the moment, the merging method provides quite satisfying results which fulfils the objectives and tackles enough the main initial problems. However, some flaws needs to be solve such as the reconstruction of harmonics with low energy. To remove those flaws should be the next step to carry on this project. Also, adapting our network with a music dataset may be another way of extending the project.

References:

(0)github repository of the project:
https://github.com/JoachimROSSEEL/AudioSuperResolution/tree/super-r%C3%A9solution

(1)tutorial on autoencoders: https://blog.keras.io/building-autoencoders-in-keras.html by François Chollet, 14 May 2016

(2)keras library: https://keras.io/

(3)librosa library: https://librosa.github.io/librosa/index.html

(4)Free Spoken Digits Dataset: https://github.com/Jakobovski/free-spoken-digit-dataset by Zohar Jackson, January 2019

(5) Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals: https://github.com/soerenab/AudioMNIST by Sören Becker, Marcel Ackermann, Sebastian Lapuschkin, Klaus-Robert Müller, Wojciech Samek, 9 July 2018

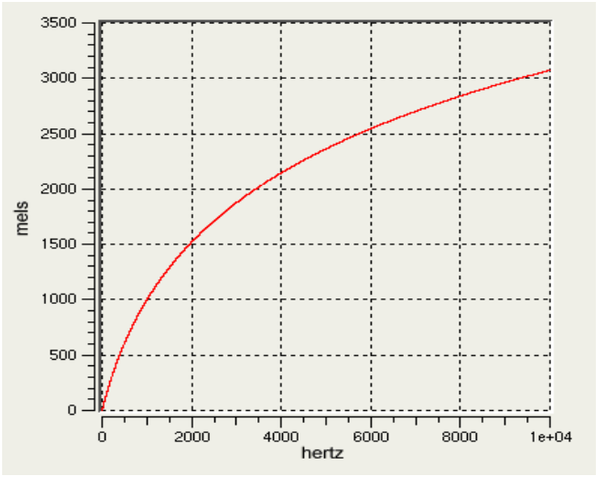
(6)Filtfilt function: https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.signal.filtfilt.html

(7)TIFGAN: https://tifgan.github.io/ by Andres Marafioti, Nicki Holighaus, Nathanael Perraudin, Piotr Majdak, 16 May 2019

(8) SpectroBanks: Learning filter banks for audio processing with convolutional neural networks by, Helena Peic Tukuljac, Benjamin Ricaud, Nicolas Aspert and Pierre Vandergheynst

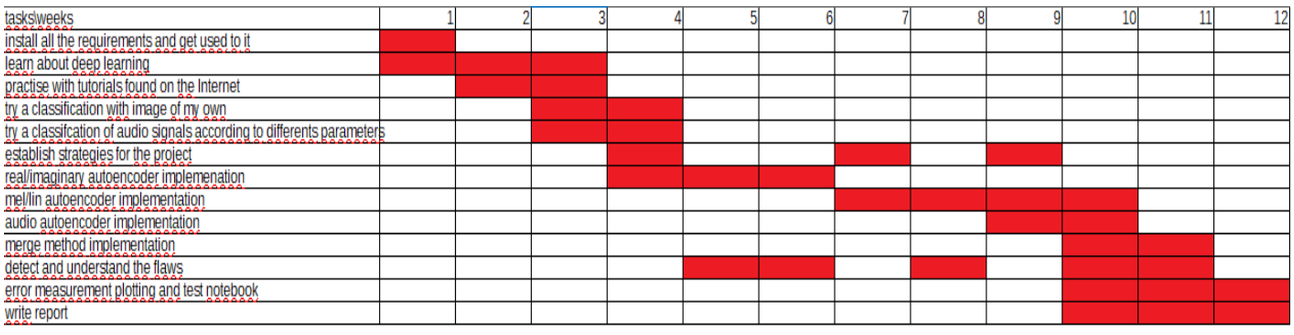
Appendix:

Hz to Mel scale:



Gantt diagram:

I'd like to precise that at the beginning of the project, I had a very limited knowledgeable about deep learning as the course are in third year. Thus a major part of this project was to learn how to use this tool.



Gantt diagram of the project

Abstract:

Super resolution consists to improve the quality of a signal by reconstructing a missing part of it. More precisely, the proposed study will process audio signals coming from the FSDD and which have been deteriorated by filtering. Thus, the missing part corresponds to a frequency band removed by this damaging. By using autoencoders, which are a set of neural networks, four solutions are tasked to restore the quality of those audio signals. They must also take into account the difficulties bounded to phase retrieval and the dB perception of human ear.

This report presents the implementation of these 4 different autoencoders, followed by the analysis of their results. The first one aims to learn the reconstruction of the real and imaginary parts of a STFT from an audio signal. However, the results here are unsatisfactory.

The two next ones support a bit different approach by learning to restore the harmonics on the spectrograms of an audio signal. One is closer to the human perception of audio by learning the mel spectrogram whereas the other one deals with normal spectrograms. All the harmonics here are quite well restored, if the high frequencies with too low energy are taken apart. Also, the more the deterioration is important, the more the networks will struggle with finding back the missing information. Thus, they need more training time.

Then, an autoencoder processing directly audio signals is introduced to you. It restores quite well the low frequency but not the high ones, which are often forgotten. Also, it allows to have an audible output.

Finally, the last part merges the advantages of the previous autoencoder and the networks using spectrograms. The results are now satisfying and a fully reconstructed audio output is available. This output is better than the one furnished by the audio network alone. This method fulfils quite well the initial objectives.

In order to improve those results, several leads are proposed, especially to obtain a better restoration of the high frequencies.

Résumé:

La super résolution consiste à améliorer la qualité d’un signal en reconstruisant une partie manquante de ce dernier. L’étude proposée va plus précisément traiter des signaux audios, venant de la FSDD, qui ont été dégrade par un filtrage. Ainsi, la partie manquante correspond à une bande de fréquence enlevée par cette détérioration. En utilisant des autoencoders, un ensemble de réseau de neurones, 4 solutions sont chargées de restituer la qualité des signaux audios. Elles doivent aussi prendre en compte les difficultés liées à la reconstruction de la phase d’un signal et de la perception en dB de l’oreille humaine.

Ce rapport vous présente donc la mise en oeuvre de ces 4 différents autoencoders, suivi de l’analyse de leurs résultats. Le premier a pour tache d’apprendre à reconstruire la partie réelle et imaginaire de la STFT d’un signal audio. Néanmoins, les résultats sont peu concluants.

Les 2 prochains réseaux changent un peu la tactique d’approche en apprenant à restituer les harmoniques via le spectrogramme d’un signal audio. L’un se veut plus proche de la perception humaine de l’audio en apprenant le mel spectrogramme tandis que l’autre traite des spectrogrammes normaux. Ici, les harmoniques sont généralement bien reconstruites mis à part les hautes fréquences où l’energie est trop faible. Aussi, plus la dégradation est importante, plus les réseaux ont du mal à retrouver l’information manquante. De ce fait, ils ont besoin de plus de temps d’apprentissage.

Ensuite, un autoencoder effectuant une super résolution directement sur les signaux audios est introduit. Il restitue assez bien les basses fréquences et permet d’avoir une sortie audible directement. Néanmoins, les hautes fréquences sont totalement oubliées.

Finalement, la dernière partie fusionne les avantages du précédent autoencoder et des réseaux utilisant les spectrogrammes. Les résultats sont satisfaisants et permettent d’avoir une sortie audible totalement reconstruite meilleur que celle du réseau audio seul. Cette méthode répond donc assez bien aux objectifs initiaux.

Afin d’améliorer ces résultats, plusieurs pistes sont proposées, notamment pour obtenir une meilleure reconstruction dans les hautes fréquences.