Information and Communication Engineering
Nunzio Alexandro Letizia
(700.604)

Baptiste CALLARD

June-July 2022

Table des matières

I Foreword

The purpose of this final report will be to discuss the results and the choices made in the assignments. I have chosen to do my algorithms on **Jupyter Notebook**. The codes will be attached separately to this report. I have also provided html files of my notebooks after execution with all the outputs if you do not want to run them. I have run all the code and it still goes to the end so if you have a problem do not hesitate to come back to me. In the report, I have used several images from the course slides to illustrate my points.

II Introduction

The aim of this course is to introduce the main methods of machine learning and deep learning. The perspective of this course is also to apply these methods to the field of Information and Communication Engineering. The fields of computer vision, speech recognition, natural language processing are historically the ones that have been studied first. The results obtained today with these methods surpass all other approaches in many aspects and constitute a real revolution. On the contrary, these methods arrived later in the field of Information and Communication. Indeed, the results without using artificial intelligence were very good. However, as in many other fields, machine/deep leaning algorithms now offer very good prospects. It is in this context that studying the application of intelligence in this field makes sense.

III Assignment 1 - Neural Decoder

In this first assignment we will see a first application of deep learning for communication. We will create a neural network to model the receiver. The goal of the model is to learn to estimate the signal that has been sent from the received signal. We have the labels of the sent signals. So we can supervise the learning.

III.1 Presentation of the problem to be solved

When we send a signal, it does not reach the receiver as it was. It is modified as it travels through the medium. Thus, the received signal is a combination of the sent signal and a noise which is random. Our aim is to estimate the signal sent from the received signal.

It is thus possible to create a receiver block as a one supervised learning problem.

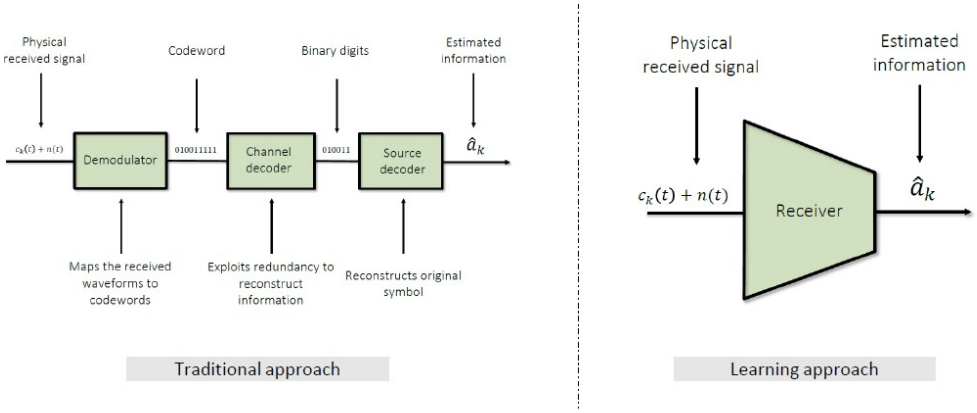


FIGURE 1 – comparison of traditional and deep learning approaches

We will try to replace the traditional approach : Demodulator, channel decoder and source decoder by a single receiver block.

III.2 Presentation of the model

For our study, we will have only 16 possible input signals, we code the messages on 4 bits ($2^4 = 16$). Our goal will be to classify the outputs among these 16 classes. Thus, we have to deal with a classification problem. First, a message s_k is sent according to a uniform distribution among the 16 possible classes. This message passes through a transmitter which is composed of several steps :

— Source encoder : converts the input into a binary sequence of bits
— Channel encoder : adds redundancy to the signal to aid signal reconstruction
— Modulator : process of imposing an input signal onto a carrier wave

The transmitter block is implemented with Hamming (7,4) coding scheme. This means that the 4-bit signal is transformed into a 7-bit signal by adding redundancy.

For this we use the following matrix :

$$G_{7,4} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

We get the messgae after channel encoding c_k like this :

$$c_k = b_k.G_{7,4} \ mod \ 2$$

The signal obtained from c_k passing through the BSPK Modulator is denoted $x_k(t)$. This whole pre-processing part is defined and is not trainable. This signal is then sent through the medium. It is then modified by random noise. The resulting signal is denoted y_k. To simulate the noise, we will add a random variable that follows a normal distribution. We will study the influence of the power of the noise later on.

$$y_k = x_k + n_k \ with \ n_k \sim \mathrm{N}(0, \sigma)$$

It is at this step that the **receiver** intervenes, which we will try to estimate in order to find the estimated message $\hat{s}$. All this is summarised in the following figure :

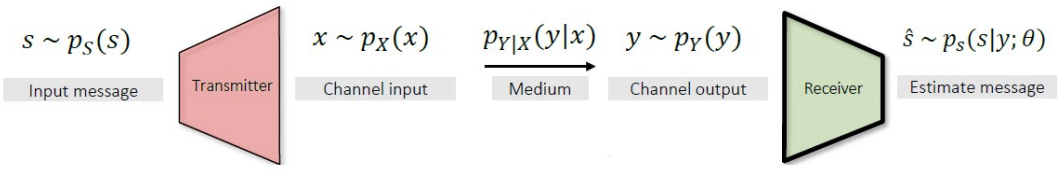


FIGURE 2 – Scheme of the overall process

A first essential method in any learning study is to choose the loss function. In the vast majority of classification problems, models are trained with cross-entropy :

$$\hat{\Theta} = argmin_{\Theta} - \mathbf{E}_{s \sim p(s|y)}[log(s|y;\Theta)]$$

where $\hat{\Theta}$ is the parameter of the model to be estimated which corresponds to the weight of the neural network

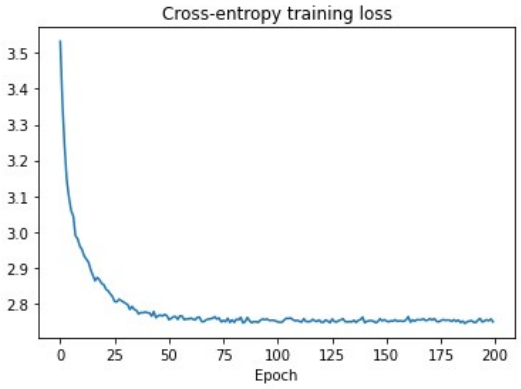
III.3 Study of the neural network for the receiver

The structure of the neural network is fully connected. That means that evry node fo two consecutives are linked with each other.

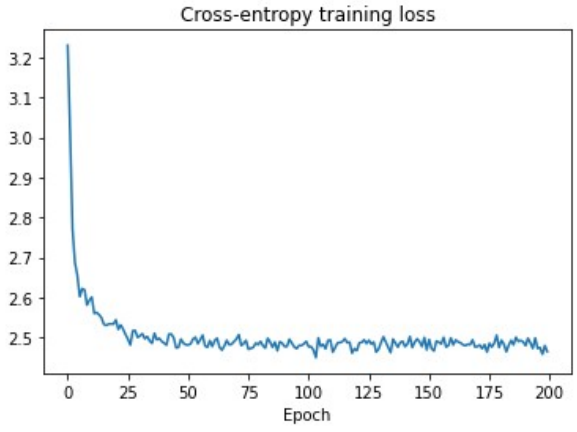
Layer	Shape	amount param
dense	128	1024
activation (relu)	128	0
dropout (0.3)	128	0
dense	64	8256
activation (relu)	64	0
dropout (0.3)	64	0
dense (softmax)	64	1040
		total params
		10,320

I chose to put in the last layer softmax to have a probability. This function is classically used in classification. Otherwise I used relu because there is no need to use anything else or to limit our values. I also tried setting tanh to the activation function before the softmax but the performance is equivalent. Finally, for the number of neurons I was inspired by assignment 2 and that's why my number of neurons decreases. I also took multiple of 2 for the shape. I have also found models with fewer parameters that give good results, but I think this is because the data is simple. I preferred to go for something more sophisticated. So it must be possible to have good performance if you increase the number of classes.

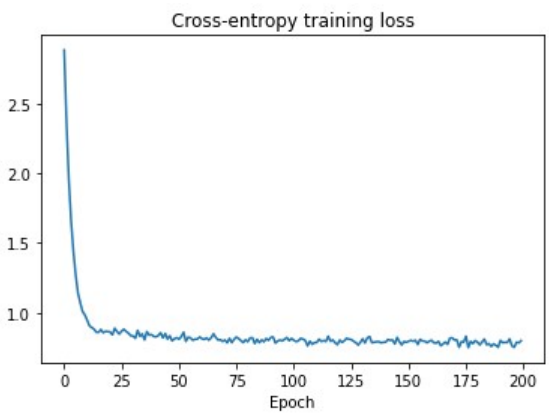
Here are the learning curves where the loss is represented during the epochs.



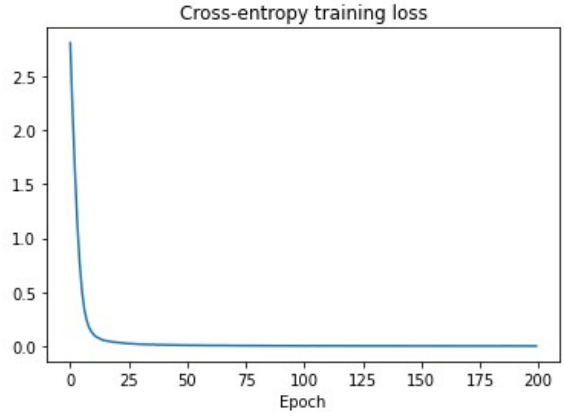
(a) cross entropy loss SNR = -20dB



(b) cross entropy loss SNR = -10dB



(a) cross entropy loss SNR = 0dB



(b) cross entropy loss SNR = 10dB

We can see that the loss decreases drastically when the SNR increases (the power of the noise decreases). To have an idea of the impact of the SNR on the learning process I have plotted for each SNR value the loss value at the last epoch fo training. Here is the figure :

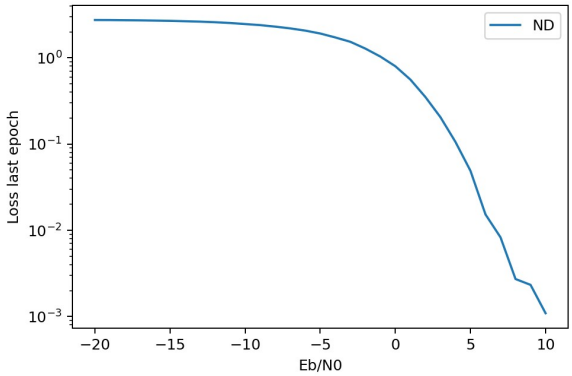


FIGURE 5 – bler in function SNR

We can see that we start learning when the SNR is higher than about 0. Here is also the evolution of BER and MaxL as a function of SNR.

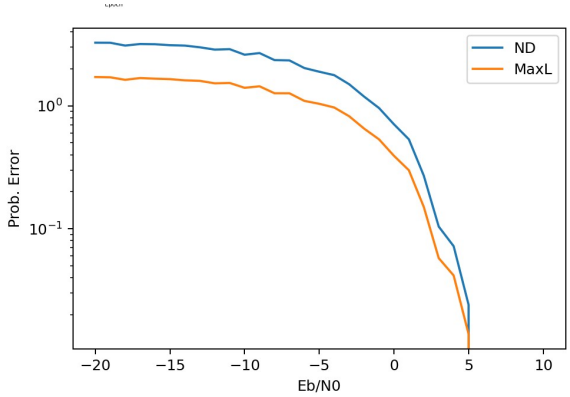


FIGURE 6 – bler in function SNR

Thus, we see that it is impossible to learn from noise and that noise makes learning impossible.

The more the noise is reduced, the better the results.Thus, we see one of the first advantages of neural networks is that it does not require advanced domain knowledge to obtain good results.

IV Assignment 2 - Autoencoder

IV.1 Presentation of the problem to be solved

In the last assignment we tried to optimise the receiver with respect to the given transmitter. As the transmitter was fixed, we can only optimise the receiver with respect to the transmitter. Thus, if we consider the complete communication chain, it was only possible to find a sub-optimum because the transmitter was not optimised in principle. With this type of method, one can only optimise the transmitter and the receiver separately. This approach does not seem to be the right one to optimise the whole communication chain. For this we can use a machine learning model that allows us to optimise both at the same time. This model is known as an autoencoder.

IV.2 Presentation of the model

To optimise the full communication chain as an unsupervised learning problem, a autoencoder model can be used. This model considers the receiver and the transmitter as two connected neural networks. The purpose of this model is to quickly learn an encoding, i.e. a representation of our data in a new space.

To do this, the model will try to estimate the inputs. This model takes our data as input and transforms it into an intermediate space. In a second step, it will try to reconstruct the initial data as best as possible from the internal representation. Thus, the model will try to learn how to find the internal representation that keeps the most information in order to be able to reconstruct our input as well as possible.

Thus, we can see that the idea of autoencoding is very close to communication. Indeed, our goal with the transmitter is to find a representation for our message that allows the receiver to best estimate the initial message. This type of approach is very felt in communication as the first works date from 2017. Here is what the autoencoder looks like in communication.

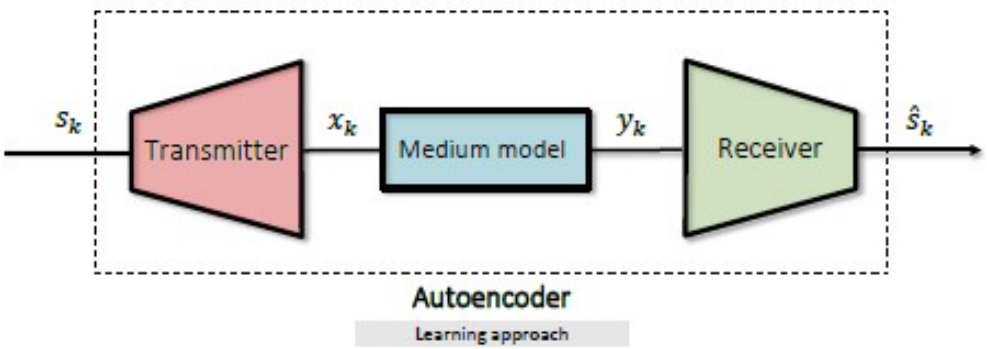


FIGURE 7 – Scheme of the auto encoder in communication

Thus, there are two neural networks for the receiver and the transmitter. These two models are linked together by the medium model.

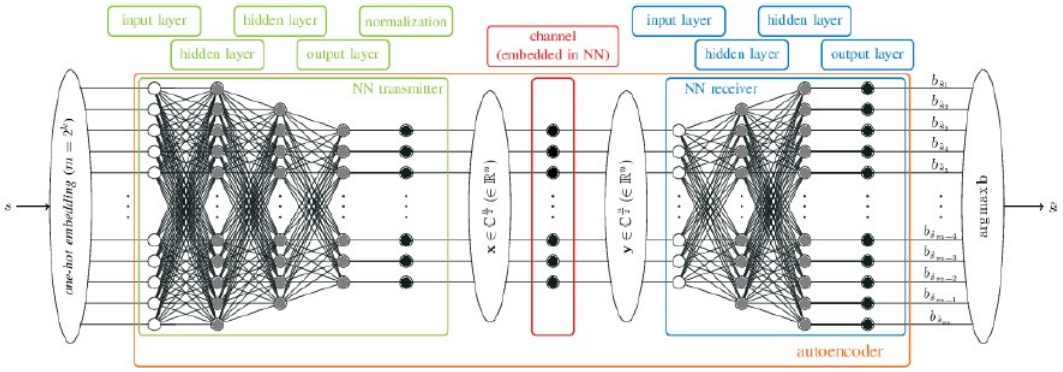


FIGURE 8 – Scheme of the overall process

So, one question I asked myself is the following : How to optimize the model with the back propagation as we have the medium model which is random. To solve this problem there is a trick called **reparametrisation trics** which consists in adding a layer for the noise.

To optimise the model, we once again use cross-entropy.

$$\hat{\Theta} = argmin_{\Theta} - \mathbf{E}_{s\sim p(s|y)}[log(s|y;\Theta)]$$

where $\hat{\Theta}$ is the parameter of the model to be estimated which corresponds to the weight of the neural network.

Another important thing to remember is that it is important to find a good compromise between the communication rate R and the bit error rate BER. Indeed, if you increase R then you will also increase BER.

In this assignment, we will try to send an input alphabet of dimension M = 16 and to find a code of length n = 2. The inputs of our model will be $s \in \{0, ..., 15\}$. We will use one hot encoding :

$$1 = \begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix} \quad 2 = \begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix} \quad ... \quad 15 = \begin{bmatrix}0\\0\\0\\\vdots\\1\end{bmatrix}$$

IV.3 Study of the network for the autoencoder

Most of the time, I have seen so far that for autoencoders, the design of the transmitter and the receiver are symmetrical.

Struture neural network for the transmitter :

Layer	Shape	amount param
dense	64	1088
activation (relu)	64	0
dropout (0.3)	64	0
dense	128	8320
activation (relu)	128	0
dropout (0.3)	128	0
dense	2	258
activation (tanh)	2	0
		total params
		9,666

Here, having the tanh activation at the end is not necessary because afterwards we have a normalization batch. I tried to remove the tanh function. The results do not differ so the choice is not important a priori. So, we can delete it to have a better back propagation. The only small difference that could be noted is when the SNR is large. Without tanh the model predicts values that are much more dispersed but this does not improve the results. I also used dropout - 0.3 layers to avoid overfitting.

I have also displayed some results without tanh (In APPENDIX). Visually, I find the results with tanh have a better shape but I am not a specialist.

Struture of neural network for the receiver :

Layer	Shape	amount param
dense	128	384
activation (relu)	128	0
dropout (0.3)	128	0
dense	64	8256
activation (relu)	64	0
dropout (0.3)	64	0
dense (softmax)	14	1040
		total params
		9,680

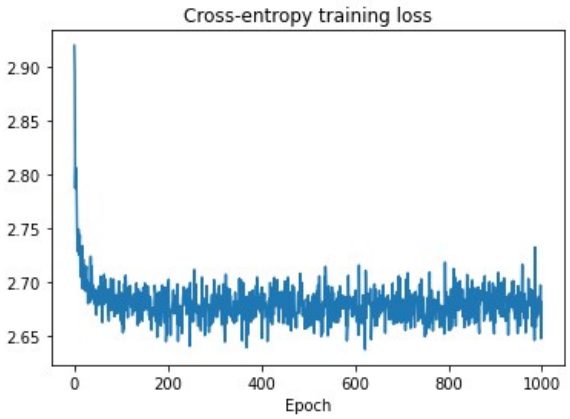
For all the layers execpt the last one I use relu and it works pretty well. For the last layer I chose to put softmax because we are looking for a probability for each class. I also use drop out - 0.3 for the same reasons.

Overall the model studied is relatively simple so I think that many architectures can achieve good results. So it is complicated to give pros and cons of the choices.

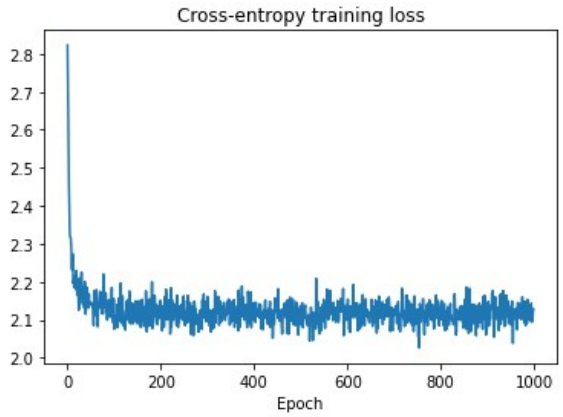
IV.4 Study of latent codes

We will now look at the latent codes. This allows us to learn things about the data, which is also one of the forces of autoencoders. If the autoencoder is well trained we should observe a structure. Here the space of the internal representation is n = 2 (real and imaginary part). So to best distinguish the different inputs, we would like to separate the data in this space as best as possible with a constraint on the signal power.

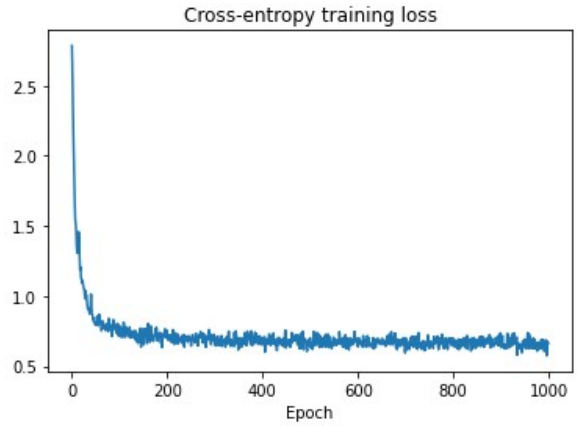
As before we will make a comparison for the SNR in dB from -20 to 20.



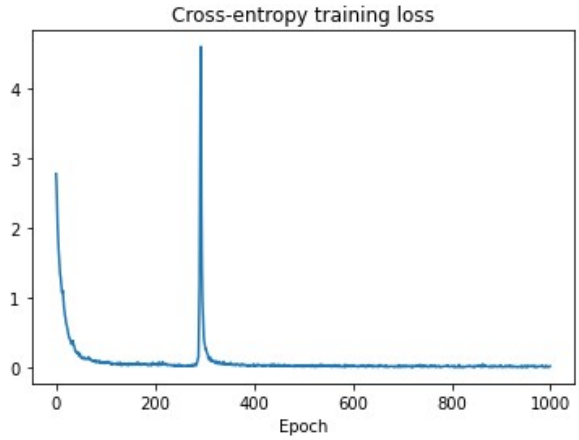
(a) loss function SNR = -10



(b) loss function SNR = 0



(a) loss function SNR = 10



(b) loss function SNR = 20

We can see that when the SNR in dB is low then the model fails to learn. This is because the noise is too high. The more the noise decreases, the lower the loss.

(a) bler in function of SNR

Our Bler tends towards 0 as the SNR decreases. We achieve quite the same results as in assignment 1 because here we optimise both the transmitter and receiver. However, these results are

really better because in assignment 1, we started with a 4-bit representation. Then we added redundancy by giving a 7-bit representation so we had a high rate but a small Ber. In contrast, with the autoencoder we represent on only 1 complex symbol so we increase the rate but the error remains low. Thus, the autoencoder seems to be a better model in our modelling if we can also optimise the transmitter.

Below we can see the evolution of the latent code (before and after the medium model) as a function of the SNR. The first figure corresponds to the centre of the clusters before the medium. As the noise is Gaussian then the inputs will be distributed around the cluster centres of their class according to a normal distribution (2D).

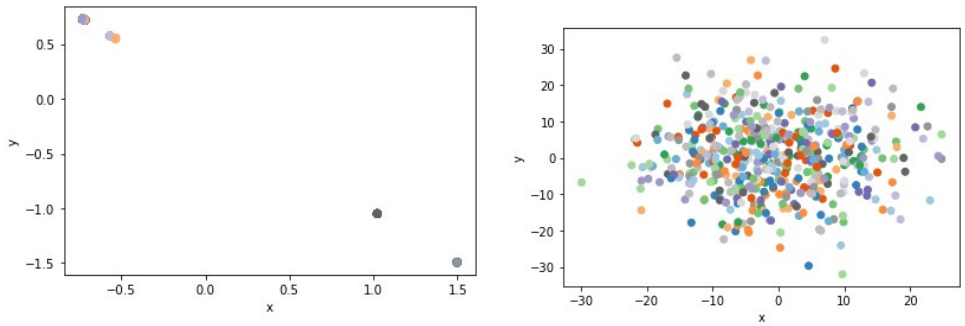


FIGURE 12 – latent code SNR = -20 dB

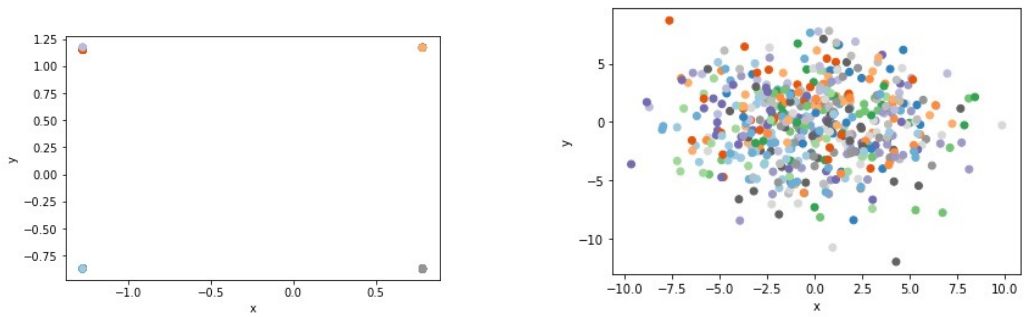


FIGURE 13 – latent code SNR = -10 dB

At first, we can see that the model has difficulty in separating the inputs before the medium. There is no clear structure. There are some clusters, but these are made up of many different inputs. Thus, it is difficult to separate the points after the medium chanel because the noise has scattered all the points according to a normal distribution. It is not possible to distinguish and therefore estimate the messages from this representation. This is due to low value of SNR (in dB).

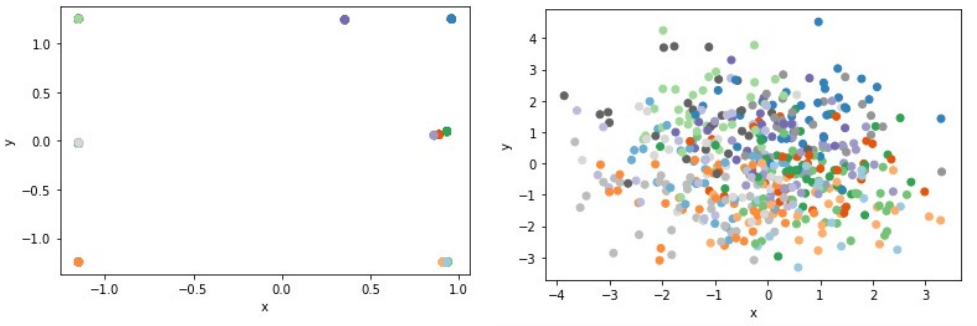


FIGURE 14 – latent code SNR = 0 dB

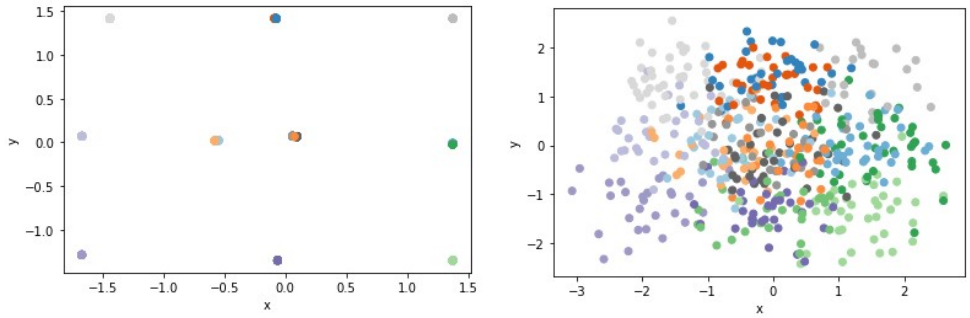


FIGURE 15 – latent code SNR = 5 dB

Gradually, it is possible for the model to find a representation that separates the inputs better and better. This is possible because the noise decreases. So for SNR = 0 dB and SNR = 5dB, we see that there is a structure. Between 9 and 10 clusters appear. Ideally, we would like to have 16 clusters.

We start to see that after the medium the messages of a message class start to be grouped together. The messages are still highly scattered and even if the cluster centres are better the scatter plots overlap. This allows the receiver to estimate more correctly the messages received but it is still not perfect.

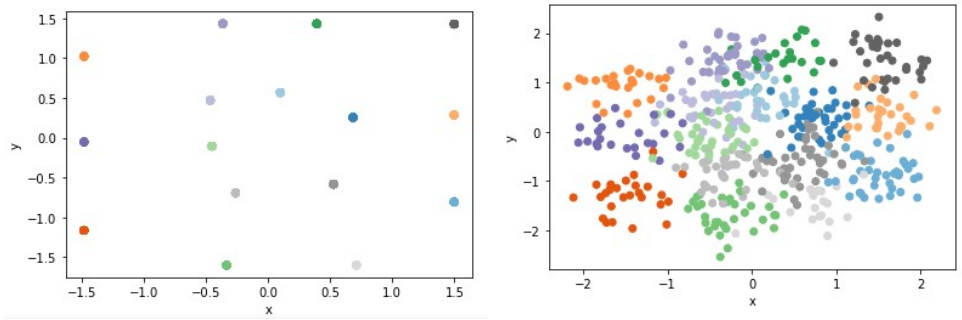


FIGURE 16 – latent code SNR = 10 dB

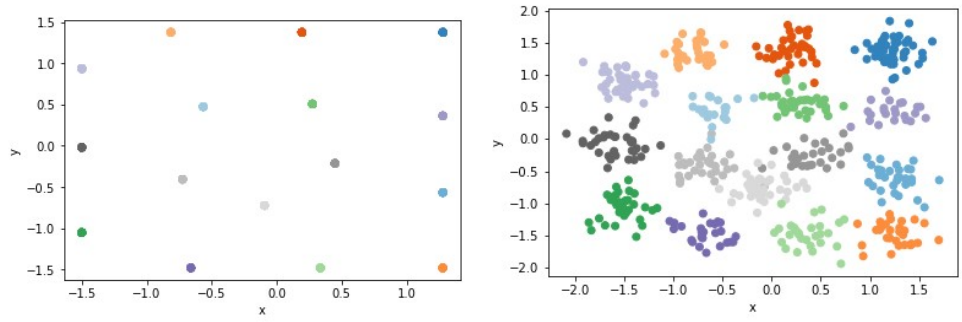


FIGURE 17 – latent code SNR = 15 dB

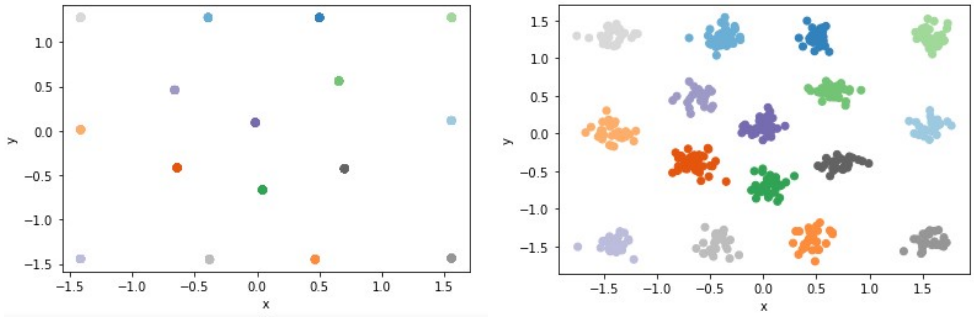


FIGURE 18 – latent code SNR = 20 dB

For SNR = 10 dB, SNR = 15 dB and SNR = 20 dB, all 16 clusters appear distinctly. However, the distance between the clusters is not optimal for SNR = 10 dB and after the medium the distributions overlap. At SNR = 20 dB, the ideal location is beginning to be found, and this becomes apparent after the medium, as the clusters are very clearly distinguishable and the distributions do not overlap anymore. Indeed, for this SNR the bler is null so we have an accuracy of 100 %.

We have seen the power of this model in the context of communication. We were able to optimise both the receiver and the transmitter and also get an idea of the constellation in the latent space. It is a very complete model and yet it is not very complex.

V Assigmnent 3

V.1 Presentation of the problem to be solved

So far, we have managed to optimise the transmitter as well as the receiver using deep learning algorithms. The only part we have not tried to model is the medium and therefore the noise.

In communication, GANs are excellent tools for modelling the medium.

If we can model the medium with GANs then this model can be introduced into an autoencorder to model the entire transmission chain.

V.2 Presentation of the model

The model we are going to use is of course the conditionnal Generative Adversarial Networks (GANs).GANs is an very interesting model in its construction. It consists of two neural networks

which are adversarial. The first neural network is called the generator, it is the network that interests us to make our fake predictions. The purpose of this network is to learn how to generate data that is not distinguishable from true values. Then, we will also train a discriminator which will have for goal to differentiate the data generated by the generator and the true data. In this way, both algorithms will progress at the same time and allow better data to be generated.

From a mathematical point of view :

— the generator will try to generate new data x' with distribution p(x)

— the discriminator will try to learn to distinguish x' and the collected data x.

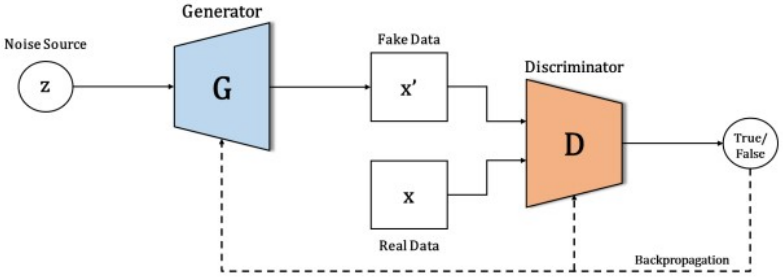


FIGURE 19 – GANs

We want to find G and D such that :

$$min_G max_D V(D,G) = \mathbf{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbf{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

The training of this model is very interesting. We will train the two models alternately.

First, we fixe the generator and the discriminator is trained. Then the discriminator weights are fixed and the whole network is trained. As we have fixed the discriminator weights, only the generator weights are updated during the back propagation.

Knowing the generator, the expression above is minimised when :

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_{gen}(x)}$$

It can also be shown that when we have convergence $p_{gen}(x) = p_{data}(x)$ and $D^*(x) = \frac{1}{2}$. It means that the prediction of the dicriminator are random (it is like throwing a coin)

In this assigment we will try to use a simple 4-QAM scheme to simulate the transmission through an AWGN channel and generate the real received data $y_k = x_k + n_k$.

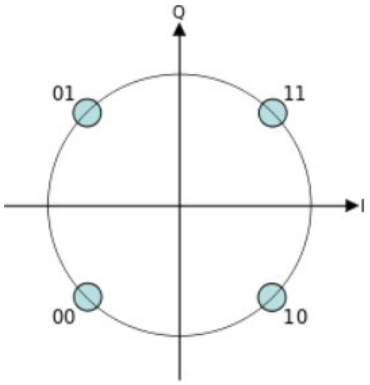


FIGURE 20 – 4_QAM

V.3 Network study for GAN

For the creation of a GAN, both the generator and the discriminator must be designed. Care must be taken to ensure that both models progress at the same speed. Indeed, if the discriminator is too good then the generator will not be able to operate and will not progress. The other extreme is that the dicriminator is too bad and so the generator will learn to generate bad data because it will pass the discriminator test even if it is not good.

The first iteration is also very important, it depends strongly on the batch size. Indeed, if we show too many examples to the discriminator then it will take too much advance on the generator and we will have a lot of difficulty to converge. This is why we recommend not to have batch$\geq$64.

Here is the structure I have chosen for the two models :

The size of the latent representation is 2. If we put more than two then the model manages to create densities around the centroids but the assignment is completely random. Thus, I noticed that it was easier to separate the data when dim(z) = 2. However, I had to increase the number of channels per layer.

For the discriminator :

Layer	Shape	amount param
dense	1028	3084
activation (leaky_re_lu - 0.2)	1028	0
dense	512	526848
activation (leaky_re_lu - 0.2)	512	0
dense	1	513
activation (sigmoid)	1	0
		total params
		530,445

The discriminator wants to make a single continuous prediction between (0,1). Thus, it is natural to consider the sigmoid function as the activation function. Then, I chose for the other activation functions to use the function $leaky\_re\_lu - 0.2$ which allows to slightly authorize the negative values. For the shape of the layers, I used multiples of 2 and I tried with different configurations. For this part, I mainly observed the results. I noticed for example that having more than 4 dense

layers includung the last dense with shape = 1 did not give good results and the only advantageous configuration I found was to have 3 layers including the last dense with one neurone.
For the generator :

Layer	Shape	amount param
dense	512	2560
activation (leaky_re_lu - 0.2)	512	0
dense	1028	527364
activation (tanh)	1028	0
dense	2	2058
		total params
		531,982

For the generator we wanted predictions around the points (0,0), (1,0), (0,1) and (1,1). As we want a normal distribution, the values can be greater than 1 and less than 0. Thus, we should not put an activation function at the very end after the dense layer. Instead, to control the values we can put an activation function tanh just before the last dense layer to bound the values between (-1,1). For the activation function after the first dense layer, I chose to put a leaky_re_lu to avoid the vanishing gradient problem and to allow slightly negative values. Finally, for the structure we start with 4 neurons and we want to predict 2 values (real and imaginary parts). I also chose to build the generator and discriminator as dual. Thus, the shape of the layers was also chosen by trial and error.

V.4 Result

It is very complicated to converge a GAN. I noticed this even on a very simple example like this assignment. At first I was getting results that looked good because I had normal distributions appearing around the points (0,0), (1,0), (0,1) and (1,1). However, I did not look at whether the points were deterministically going to the right cluster. When I looked to see if this was the case, I realised that the generator had learned nothing. There were normal densities but the points were randomly distributed among the 4 clusters.

So it was difficult to find a good structure from this point on. To see if the predictions were good, I also plotted the histograms on the x- and y-axis to compare them with the marginal normal distribution N(0,eps) and N(1,esp).

Then I let the model run for about 150 000 epochs. I printed the results every 1000 epochs and also saved the separate points of the different models. To get better results I probably should have made more regular backups

Here are the best models (it runs for more than 4 hours) :

The figure on the left compares the true (with colors) and estimated distributions(in blue). The figure in the centre shows whether the estimated data are well distributed around the correct class (the previous blue points are displayed with the color of their class). I made two figures because it was complicated to see the shape of the cast and the class at the same time. Finally, the figure on the right gives the histogram estimate of the marginal law distributions. I have also plotted the true marginal normal distribution N(0,eps) and N(1,eps). The blue and orange hitograms are the histograms of the projections on the two axes.

Note : The figures on the right and in the middle do not have the same scale in x and y. This is why the clouds appear to be stretched along the x-axis.

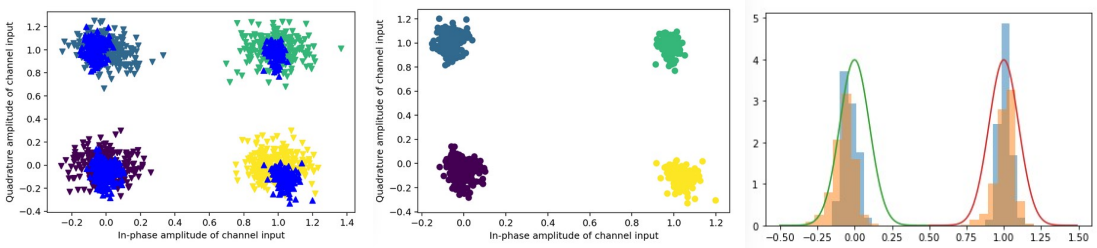


FIGURE 21 – epoch = 74000

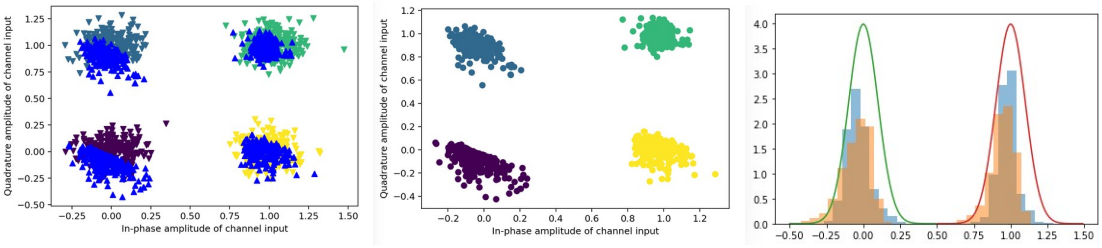


FIGURE 22 – epoch = 84000

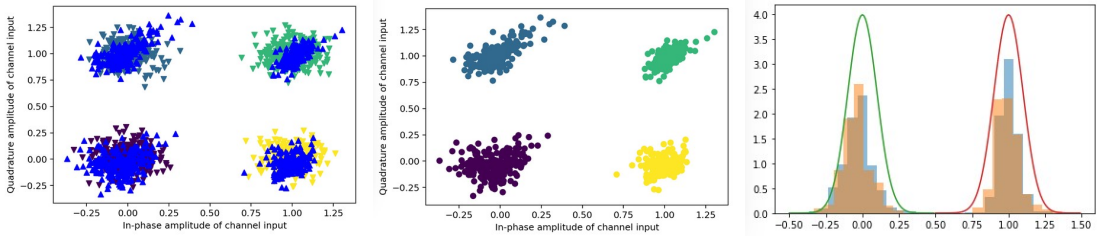


FIGURE 23 – epoch = 146000

To make our choice, we can look at the marginal law distribution. According to me, we can also look at the figure on the left which gives a good idea of the overall shape of the distribution representation of the scatter plot compared to the real data.

We can see that for most of the models the data are well centred.
The first model is good because it gives a good dispersion and a good mean. However, the dispersion around the mean is too low. The second model seems to estimate the dispersion better but there is a linear trend in each scatter plot. Finally the third model seems to be a compromise between the two other models. This is why I chose this third model.

These are good results, but I think that by saving the models more regularly (100 epochs) then it would be possible to have better results. Also, it would be interesting to decrease the learning rate in Adam as the model gets closer to the optimum. Another point could be to create your own metric. Thus, the training of the model will stop if it falls below a certain threshold for this metric.

VI Conclusion

During these assignments we were able to really understand how useful deep learning could be in the field of communication. Indeed, we saw how little by little the whole transmission and reception chain could be modelled and optimised. The first model was very useful to introduce neural networks in this domain. Then autoencorders and GAN came to improve the process by optimising the transmitter and receiver. The GAN can be used to model the last part not modelled in the autoencoder which is the noise. I found the format of these assignments to be very good. The fact of having a skeleton is useful when you have never used the deep learning library. However, the missing parts are left to important parts that also allow to really understand and ask the right questions. Moreover, we are relatively free as for the choice of layers, activation functions etc... This will be very useful for my continuation.

VII APPENDIX

VII.1 Assignment 2

Some results without, tanh for the transmitter

Layer	Shape	amount param
dense	64	1088
activation (relu)	64	0
dropout (0.3)	64	0
dense	128	8320
activation (relu)	128	0
dropout (0.3)	128	0
dense	2	258
		total params
		9,666

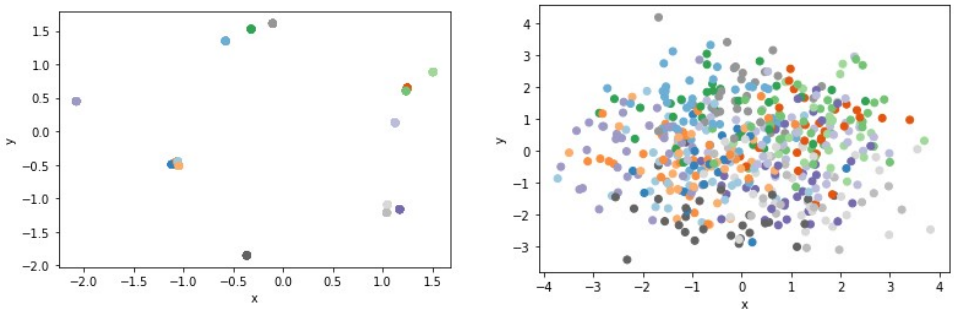


FIGURE 24 – SNR = 0

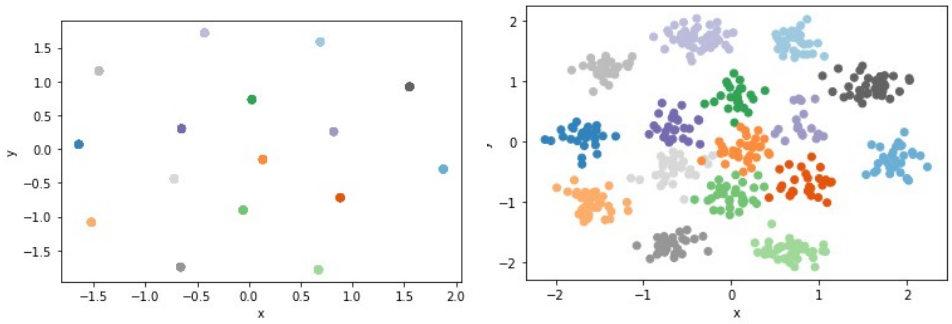


FIGURE 25 – SNR = 15

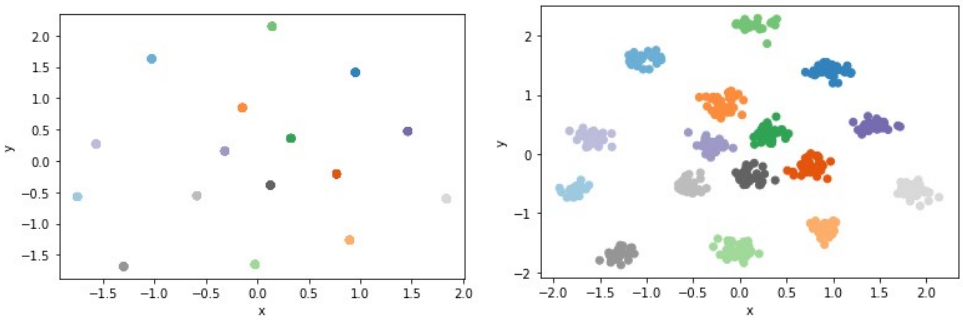


FIGURE 26 – SNR = 20

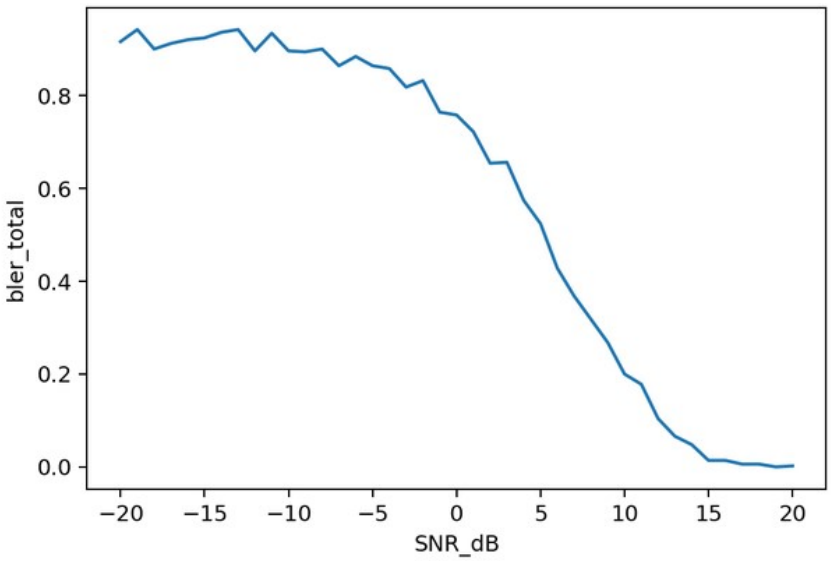


FIGURE 27 – BLER

.