Assignement 2

Development Environment

Software:

- Python
- Pytorch / Keras depending on the exercise
- VScode
- Intellisense python debuguer

Hardware:

- Laptop running linux
- no GPU

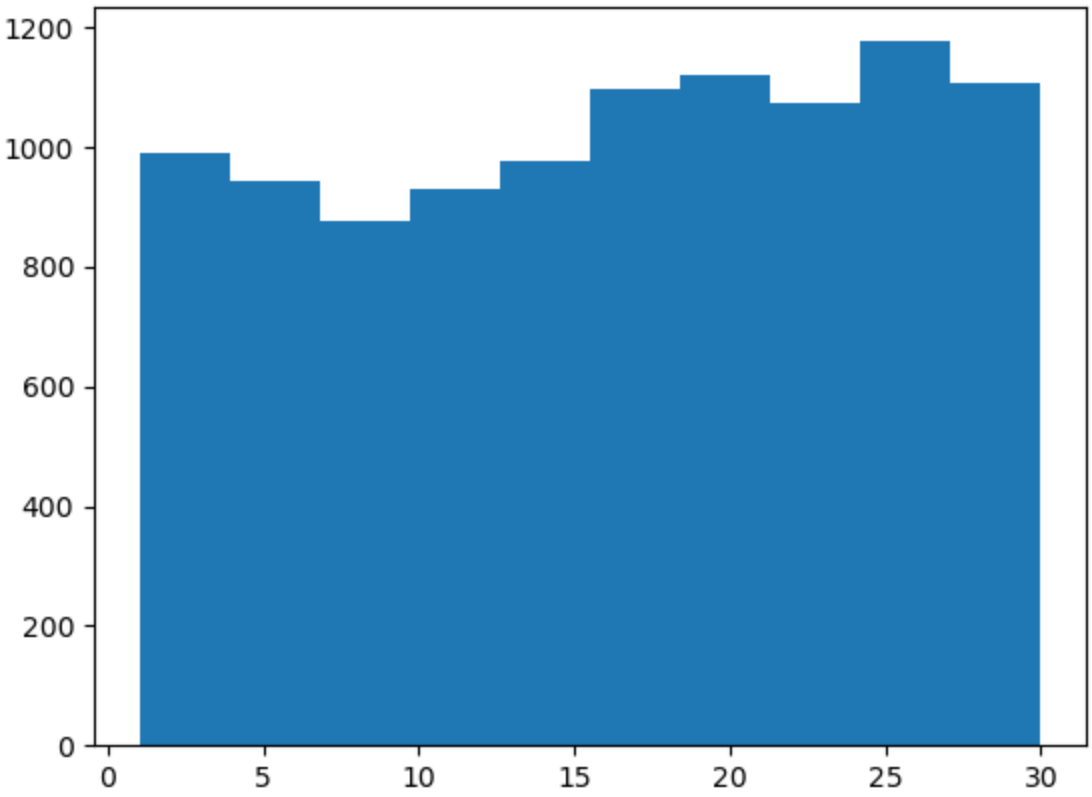
I use ChatGPT once for the data preprocessing of the second exercice, my prompt was :

```
what is th best way to normalize a date for machine learning
```

Task 1

Data Preprocessing

For this the data was already normalized so I just extracted it from the different files in wich it was stored and transformed it into a tensor.



Subject representation in the data

Model Architecture

1. the pytorch LSTM (nn.LSTM)
2. a hidden layer made using nn.Linear
3. a dropout Layer to avoid overfitting (nn.Dropout)
4. The output layer(also nn.Linear)

Choice of hyperparameters

For this part I choose to for classical hyperparameters because they were working perfectly fine:

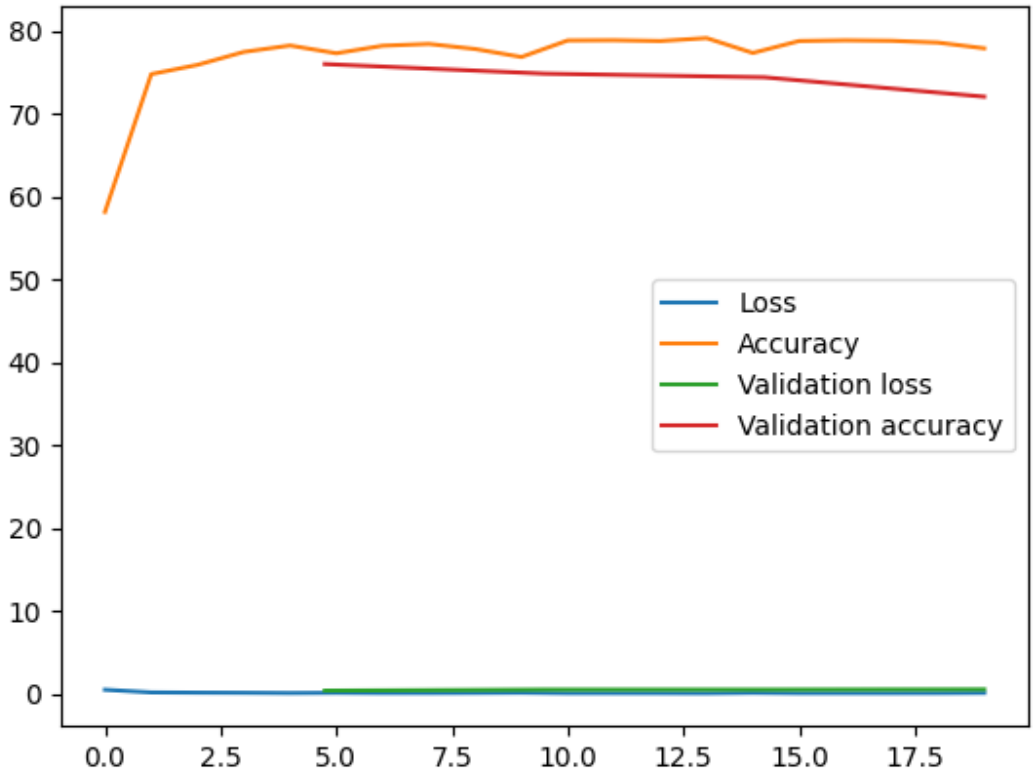
epoch: 20

learning rate: 0.01

batch size: 128

I could have used an adaptative learning or an auto-stop but the model doesn’t seem to overfit on the training data

Learning Curves



Summary

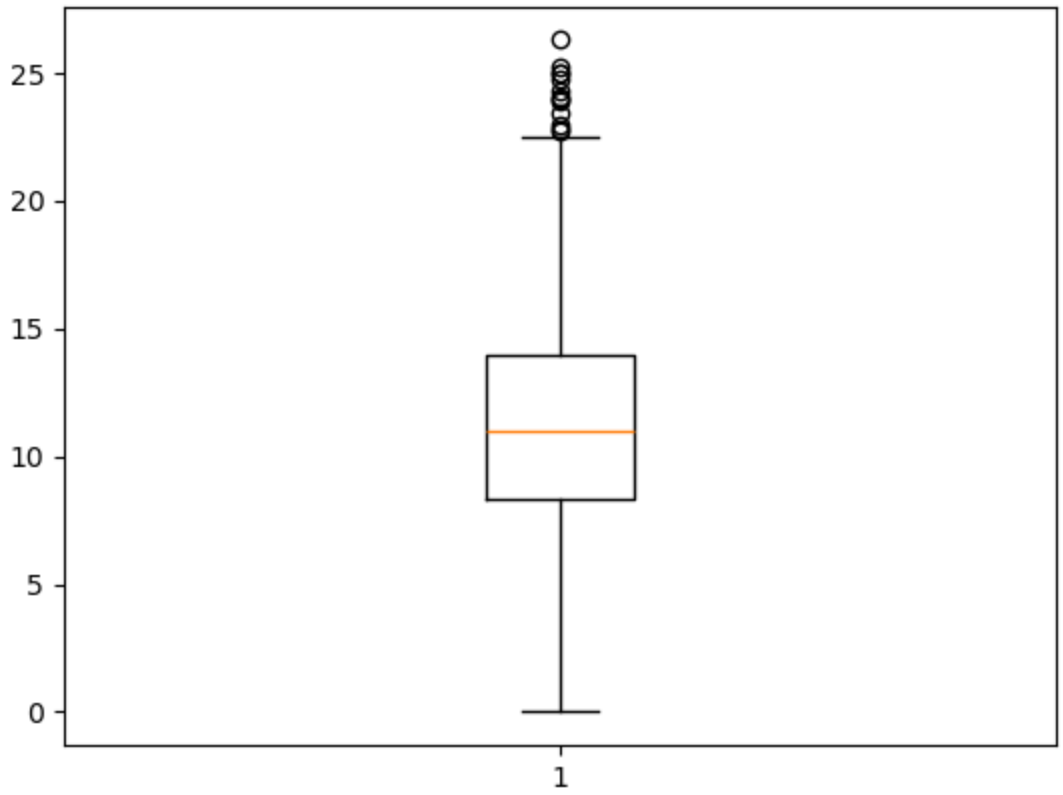
This task was my first try at implementing a LSTM but as pytorch is basically providing all the documentation it wasn’t the hardest part of the assignment

---

Task 2

Data Preprocessing

for the data preprocessing at first I tried to split each date into three different number: one for the month, one for the day and one for the year and to normalize each of them between 0 and 1 but then the 31st of december was the farthest value from the 1st of january which is not realistic. As said before I asked chatGPT for help on this and he gave me an interesting idea: using cos and sin to represent the month so that the month value will make a loop.



Boxplot representation of the temperature

Model Architecture

1. a gru model imported from nn.GRU
2. a Linear hidden layer made from nn.Linear
3. The dropout Layer
4. The second Linear (output) layer

Since this one was supposed to output a floating number and not a class i used Mean Squared Error loss as Criterion

Choice of hyperparameters

Once again the classical hyperparameters were working fine:

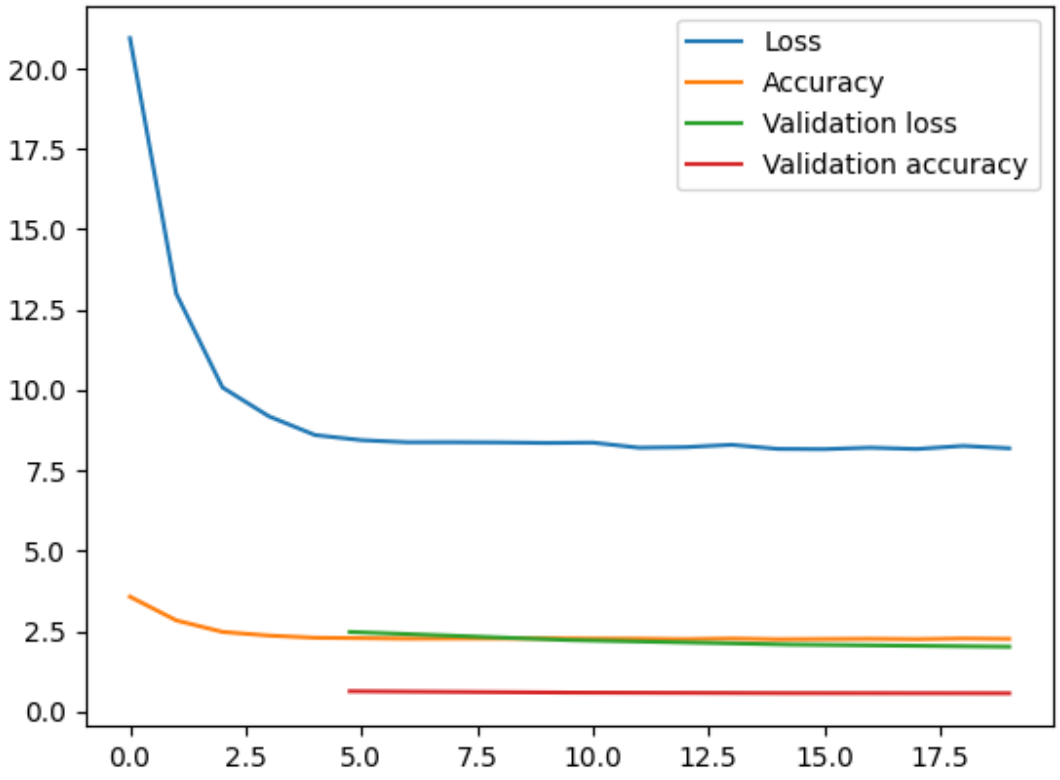
epochs: 20

learning rate: 0.01

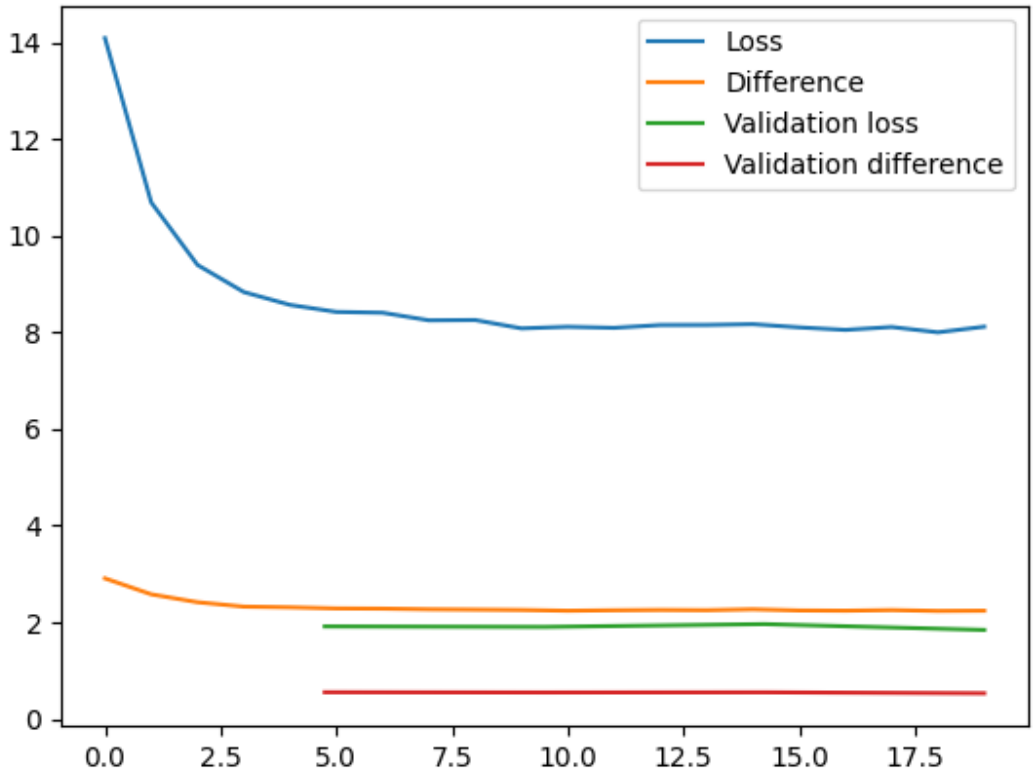
for the dropout I tried different things but it seems that for this particular model 0.25 was the one with the better result

Learning Curves

No use of cos/sin in preprocessing



Use of cos/sin in preprocessing



The two curve are very likely but the start of the first one is higher that's the impact of the difference in data-preprocessing.

Summary

This task was really interesting because even if the model is similar to the one in the previous task, the data processing was interesting and I think from now on, I will pay more attention at the way I normalize data.

Task 3

Data Preprocessing

The data was parsed from olivetti_face on sklear so it was already normalized as array, I just transformed my array into tensor. Then I created two different part of the dataset

using sklearn.train_test_split(faces,test_size=0.2) and had my 80/20% of training/validation dataset.

Model Architecture

As an autoencoder the model is made of two mirrored part:

The encoder is use to shrink the image:

1. a convolutional 2d layer with a relu activation
2. A max pooling layer
3. a second convolutional layer with a relu activation
4. a second max pooling layer

The decoder will upsize the image using:

1. The inverse of the encoder (3) in a conv2d with a relu activation
2. an upsampling layer to inverse the (4) max pooling layer
3. The inverse of the encoder (1) in a conv2d with a relu activation
4. an upsampling layer to inverse the (2) max pooling layer
5. A sigmoid activation function

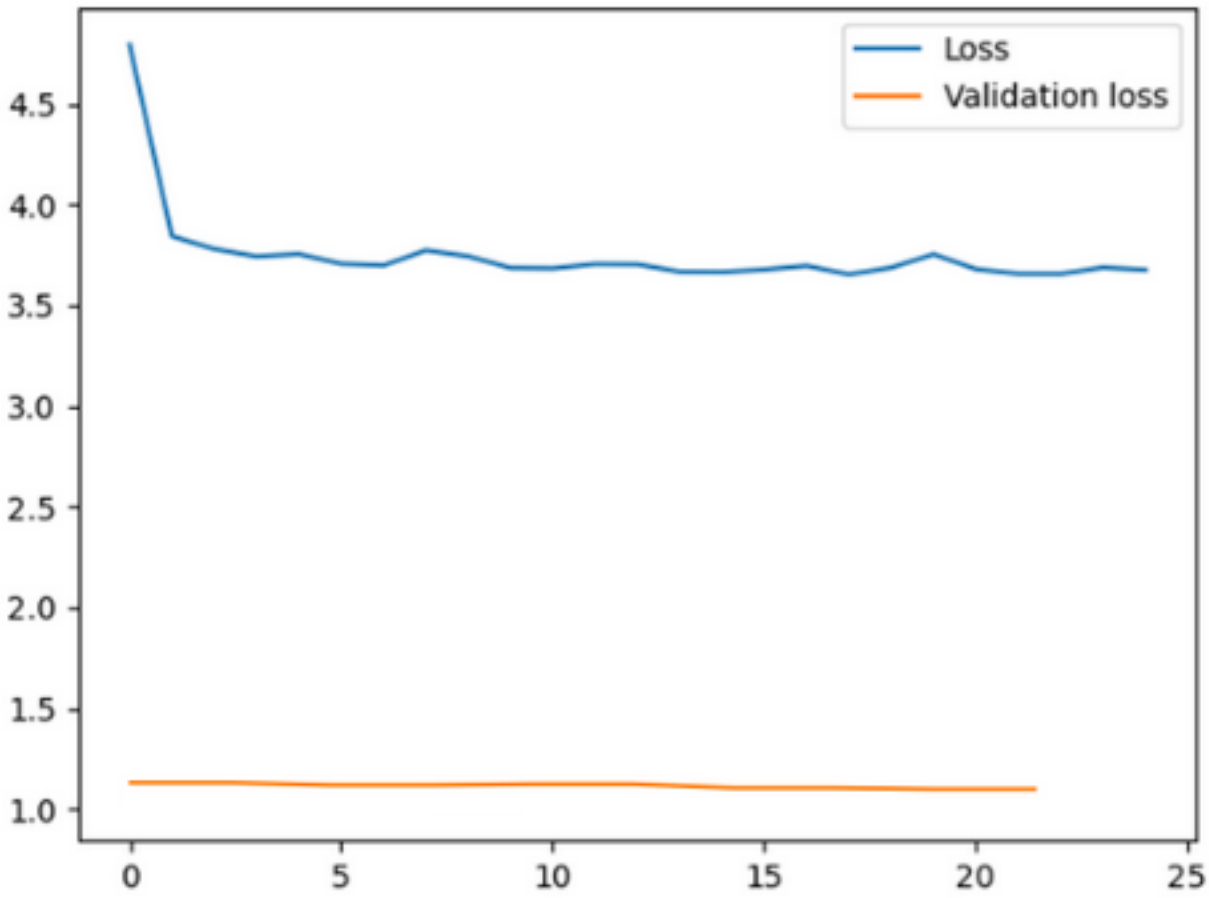
All the layer where created using modules from torch.nn

Choice of hyperparameters

The hyperparameters for this one are a bit different:

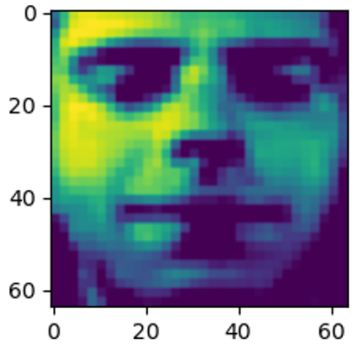
since there is an auto-stop in case wich the loss is not going down anymore I can use more epoch and cut down if the learning slow too much so I have 25 epoch. The only fitting learning rate was 0.01 with other learning rate the model was not returning anything good. I tried to add some dropout but whatever was the probabily I tried the model gave better result without so I removed it

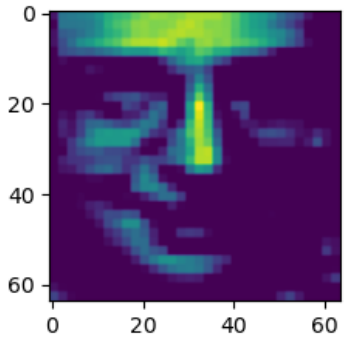
Learning Curves



This curve append in a case where the auto stop was deactivated to create a longer curve.

You cans also see below two reconstructed images.





Summary

This case was the trickiest because we are not using number anymore but plain images, you can’t change the data as you like and you need to pay attention to the dimensionality of those. This task made me learn a lot about weight initialisation, I tried to implement xavier’s one but I think it’s not really well-made because sometimes the loss start at 10 and never goes down. Also the images in output are a bit weird, I think it’s because pyplot is interpreting the channel as green instead of grey.