Image And Video Analysis: Automatic Detection Of Pain Level

Antonio Acunzo
Università degli Studi di Firenze
antonio.acunzo@stud.unifi.it

Jacopo Bartoli
Università degli Studi di Firenze
jacopo.bartoli@stud.unifi.it

Abstract

Analyzing sequences of data is a challenging task nowadays. There are a lot of models capable of handling this task; the most famous are LSTM and Transformers. The goal of this project is to predict the VAS value, a measurement used for subjective characteristics. In our scenario it measures the pain experienced by a patient. The predictions need to be done based on a video thus a sequence of frames. In particular we used an architecture based on a combination of Transformers and LSTM.

1. Introduction

As we mentioned before the analysis of sequences of data is a trending topic in the AI area. In particular the need to create models capable of handling Natural Languages Processing tasks led to many studies on sequence analysis. In particular two models are the most used in this field: Long Short-Term Memory(LSTM) and Transformers.

As described in [1] by Hochreiter et al. LSTM is model capable of learning to store information over extended time intervals. It can learn much faster than other older techniques, like real-time recurrent learning and can lead to more successful runs. LSTM also solves complex and artificial long-time-lag tasks that have never been solved by previous recurrent network algorithms. The LSTM structure can be seen in Figure 1

The Transformers are a model based on attention as described by Vaswani et al. in [4] . Attention mechanisms have become a part sequence modeling in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences. Transformer is a model architecture that eschewes recurrence rather than relying entirely on an attention mechanism to draw global dependencies between input and output [4]. This model structure is shown in Figure 2.

Progress in the AI area brought to the development of efficient techniques for the medic field. Architectures and models that can achieve really high scores on certain tasks like object detection and segmentation in medic images can lend a hand to perform diagnosis or for treatment prescription.

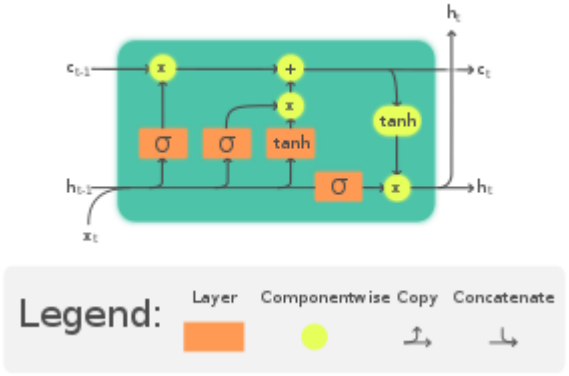

Figure 1. LSTM Structure

In this work we want to investigate an architecture that can detect the VAS value of a patient pain analyzing a sequence of frames. In these videos the patient performes an action that leads to some degree of pain. This can be of help to detect the degree of pain in subject that have some communication issues.

The datasets used for our experiment is UNBC-McMASTER Shoulder Pain Expression [2] and BioVid Heat Pain Database.

In Section 2 we introduce the considered datasets. In Section 3 we describe the architecture used in the experiments, which are presented in Section 5. In Section 4 we briefly describe the code implemented. Finally in Section 6 we described the results obtained and in Section 7 are contained some brief conclusions about this work.

2. Dataset

The dataset UNBC-McMASTER Shoulder Pain Expression contains 200 video sequences representing facial expressions of 25 different subjects. Each sequence can be composed by a different number of frame. Each frame is

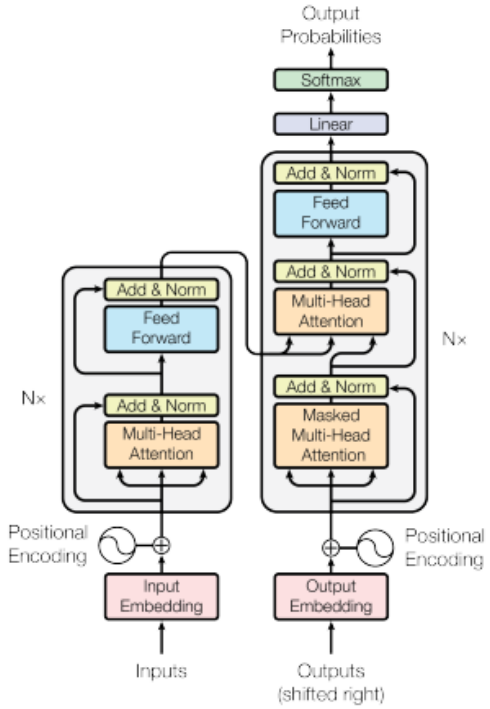

Figure 2. Transformer architecture

represented by the x and y coordinates of the facial landmarks.

Each sequence has a label that represents a value in $[0, 10]$ that describes the intensity of the pain that the subject experienced.

The dataset BioVId Heat Pain contains 8700 video sequences representing facial expressions of more than 100 different subjects. For each subject there are about 20 sequences associated with each of the 5 label values, in fact the labels in this dataset can have a value in $[0, 5]$. Each sequence is composed by 138 frames and all frames are represented by the x and y coordinates of the facial landmarks.

Both datasets are composed by two parts. The first part contains for each sequence its name, number of frame and label. The second parts contains the coordinates of the landmarks for each frame and the associated sequence id.

In Table 1 we can see the distribution of the labels in the UNBC-McMASTER dataset.

3. Architecture

Our architecture is inspired by the one described in [5]. In particular we used the regression architecture without performing the unsupervised pretraining task. The model showed in Figure 3 uses only the encoder portion of a Transformer to perform the regression and classification tasks. The authors suggested to concatenate all the output vectors

VAS Value	Frequence
0	35
1	42
2	24
3	20
4	21
5	11
6	11
7	6
8	18
9	10
10	2

Table 1. VAS Values Distributions on UNBC-McMASTER dataset

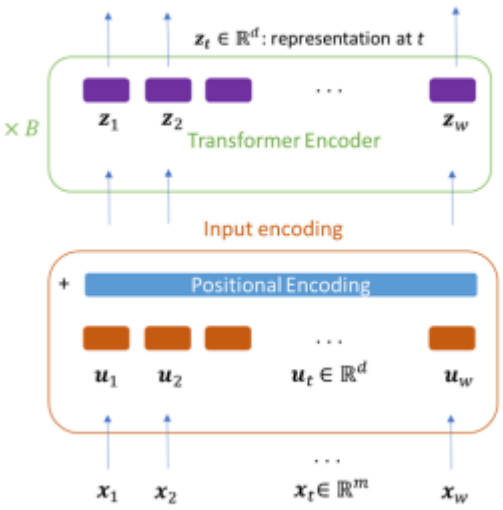

Figure 3. Regression Model Proposed by Zerveas et al.

of the encoder and feed them to a linear layer. Instead of this linear layer we tried to use a LSTM layer to perform the regression. Another difference with the referred paper is the use of a non-learned Positional Encoding. The Positional Encoding used is the sine and cosine encoding described in [4].

In the training pipeline we also used some Upsampling and Downsampling techniques to fit the data in our model input format.

4. Code

The code is all implemented using Google Colab. We divided our code between four different notebooks:

- dataset_generation.ipynb;

- preprocessing.ipynb;
- train.ipynb;
- test.ipynb.

4.1. dataset_generation.ipynb

In this notebook we generated two .csv, one for training and one for testing, starting from the raw data of UNBC-McMASTER dataset described in Section 2. In addiction you can select what landmarks take into account for the dataset creation. The main columns of the output are:

- 'Sequenza': it represents the ID of the sequence.;
- 'Frame': it represents the ID of a frame inside a sequence;
- Columns that represent the features: it can be of variable length;
- 'Label: it represents the VAS value associated to each sequence. The value is the same for each item of the same sequence.

4.2. dataset_generation_Biovid.ipynb

In this notebook we generated one .csv, for training or for testing based on the user's choice, starting from the raw data of BioVid dataset described in Section 2. The main columns of the output are:

- 'Sequenza': it represents the ID of the sequence.;
- 'Frame': it represents the ID of a frame inside a sequence;
- Columns that represent the features: it can be of variable length;
- 'Label: it represents the VAS value associated to each sequence. The value is the same for each item of the same sequence.

4.3. preprocessing.ipynb

In this notebook we applied the Oversampling and Undersampling operations. The normalization is also applied here. The notebook needs as input two .csv file that one represents the train set and the other one the test set. As output it will give two files, one for the test set and one for the train set.

4.4. train.ipynb

In this file is defined our architecture and the train set is divided between train and validation set. The notebook also computes the training of our model, saves the trained model and the metrics needed for the performance analysis.

Hyperparameters	Value
Optimizer	Adam
Learning Rate	0.001
Dropout	0.1
Epochs	30
Sequence Length	230
Attention Heads	6
Encoding Layers	3
Dimension FeedForward	256

Table 2. Training Setup

4.5. test.ipynb

This notebook computes the evaluation of the model on the test set. It saves the metrics needed for the performance analysis too.

5. Experiments

To build and train our model we used Tensorflow and Keras libraries, which provide easy and ready-to-use tools to train custom models. Concerning the training setup we used the one described in the Table 2.

As features we decided to not use the positions of the landmarks, but the x and y speed of the single landmarks instead. For our experiments we utilized all the landmarks.

During our training on the first dataset we tried different values for the dimension of the hidden representation. In particular we used 2, 4, 8, 16 and 32. The batch size used is very small, only 8, due to the small size of the dataset. As training loss we used the Mean Sqared Error as suggested in [5], while as error we used the Mean Absolute Error to compare the results with the previous work on this dataset.

During the training on the second dataset we tried to do some more hyperparameters tuning. In particular we took in to account: Batch Size, Learning Rate, number of Attention Heads, number of Attention Layers and the dimension of the hidden representation. We found a limited number of values for these hyperparameters deemed relevant.

Regarding the normalization of our datasets, we used the Standard Scaler from the library sklearn [3]. It subtracts at each feature the feature mean across the dataset and divides for the feature variance. To avoid numerical problems we scaled the label between 0 and 1.

6. Results

As we can see in Figure 46 the train loss of almost all the models decrease sharply. The only exception is the model with dimension of the hidden representation equals to 2. Looking at the graph in Figure 47 we see that the models with lower value of the dimension of the hidden representation have the worst performances. While the performance

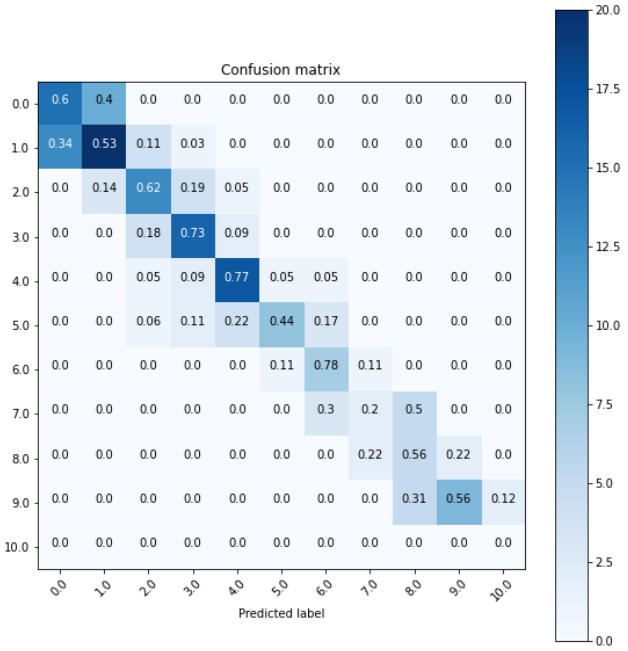

Figure 4. Confusion Matrix with hidded dimension 8

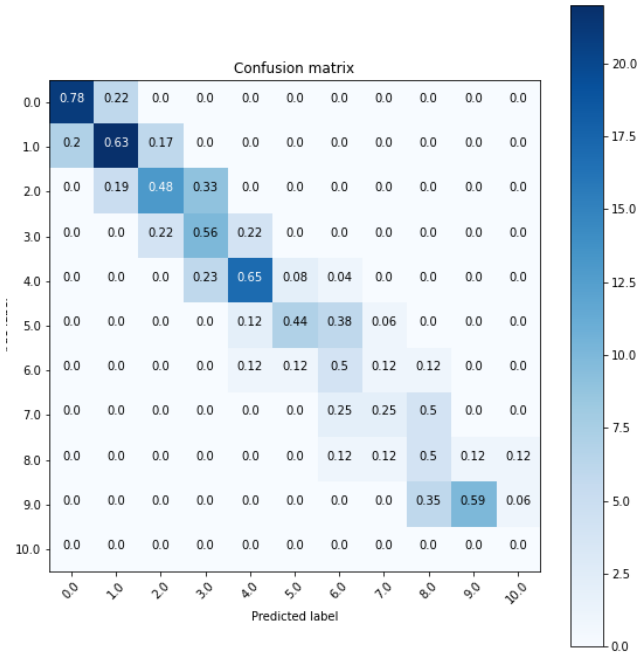

Figure 5. Confusion Matrix hidded dimension 16

Hidden Representation Dimension	Mean Absolute Error
8	0.1756
16	0.1553
32	0.1783

Table 3. Test Error

differences of the model with dimension of the hidden representation equals to 8, 16 and 32 are pretty slim.

In Figures 4, 5 and 6 we can see the confusion matrixes of the better performing models during the training phase. We can see that with a higher dimension of the hidden representation the model has better performances. We can see that the worst results are on the high VAS values. This is probably due to the lower number of examples with that values in the dataset. In Table 3 we can see that the best absolute error is with the dimension of the hidden representation equals to 16. It's important to remember that the Mean Absolute Error is computed on the label scaled between $[0, 1]$

Regarding the experiments on the second dataset we observed that the model overfit even after few epochs. In 4 we can see the score of all the configuration tested. In particular the second and the third starting from the bottom are the best performing. The first configuration seems to have the highest score but if we take a look at its confusion matrix the results are very poor.

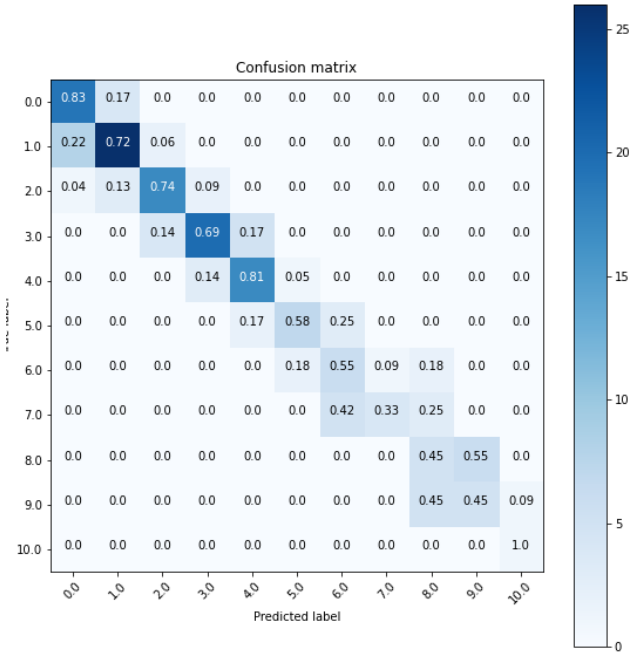

Figure 6. Confusion Matrix hidded dimension 32

7. Conclusions

In conclusions we can say that the performances of the model are satisfiying. In particular we observed that at higher values of the dimension of the hidden representation better results are achieved. The dataset taken into account is unbalanced and the number of the examples is pretty low.

Learning Rate	Batch Size	Attention Heads	Attention Layers	Hidden Representation Dimension	Error
10^{-2}	256	6	3	32	0.2977
10^{-3}	256	6	3	32	0.342
10^{-4}	256	6	3	32	0.3468
10^{-4}	256	3	3	32	0.3445
10^{-4}	256	12	3	32	0.3527
10^{-4}	256	3	5	32	0.3488
10^{-4}	256	3	1	32	0.3557
10^{-4}	256	3	5	64	0.3319
10^{-4}	256	3	5	128	0.3299
10^{-4}	256	3	5	16	0.3404
10^{-4}	256	3	5	256	0.3310
10^{-4}	128	3	5	128	0.3305
10^{-4}	64	3	5	128	0.3435

Table 4. Test Mean Absolute Error

With higher number of examples, or more sequences with high VAS values, the model can probably improve its performances.

The second dataset has a higher number of values but also in this scenario the model has some issues. In particular as we said above the network is overfitting very fast, so it's needed to apply some regularization. The hyperparameters that can help in this task are the number of Attention Layers and the dimension of the Hidden Representation. More test on the combination of this value can lead to better results.

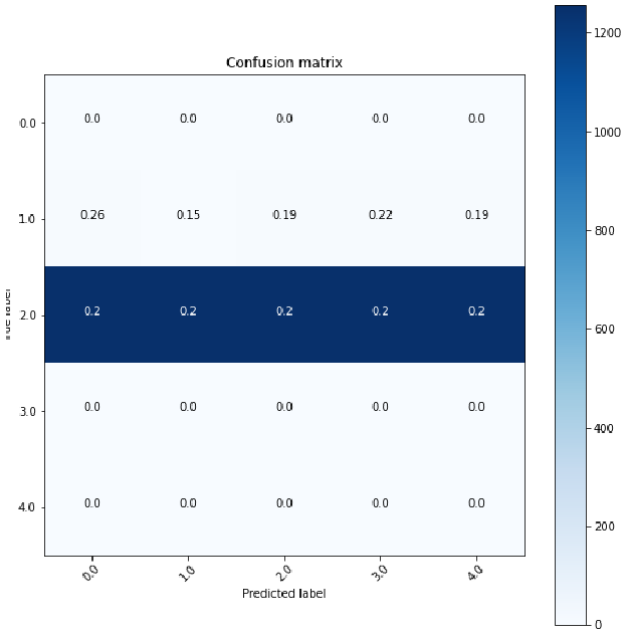

Figure 7. Confusion Matrix of Training with LR 0.01

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[2] Patrick Lucey, Jeffrey F Cohn, Kenneth M Prkachin, Patricia E Solomon, and Iain Matthews. Painful data: The unbc-mcmaster shoulder pain expression archive database. In *2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG)*, pages 57–64. IEEE, 2011.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[5] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

8. Appendix

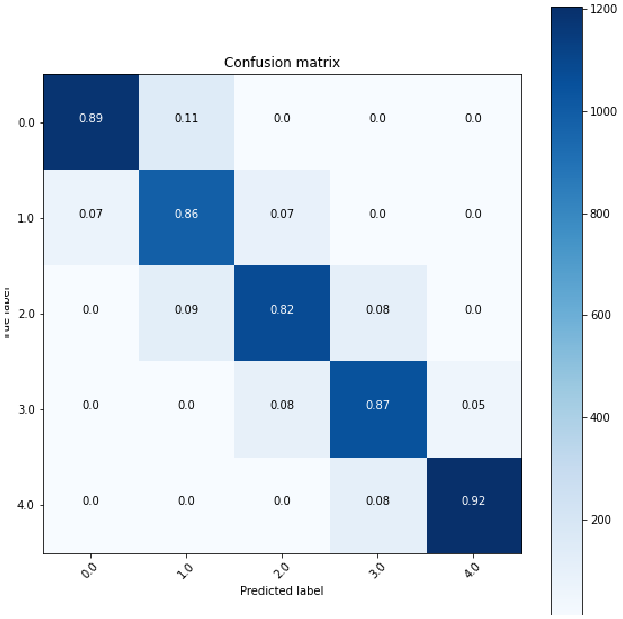

Figure 8. Confusion Matrix of Training with LR 0.001

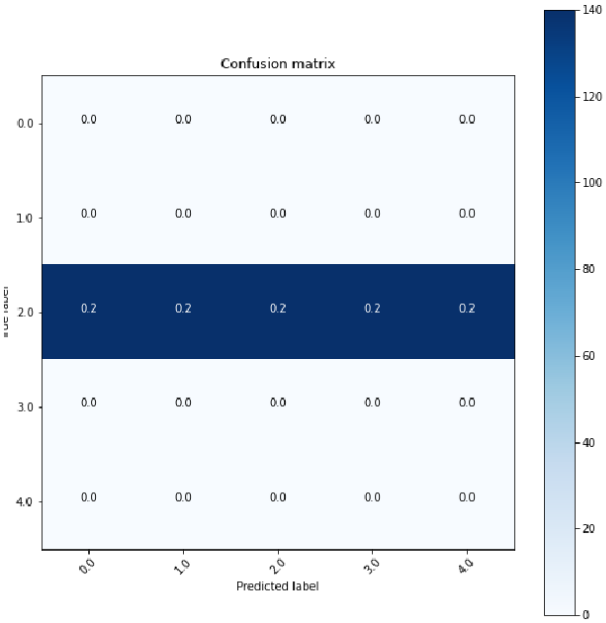

Figure 10. Confusion Matrix of Validation with LR 0.01

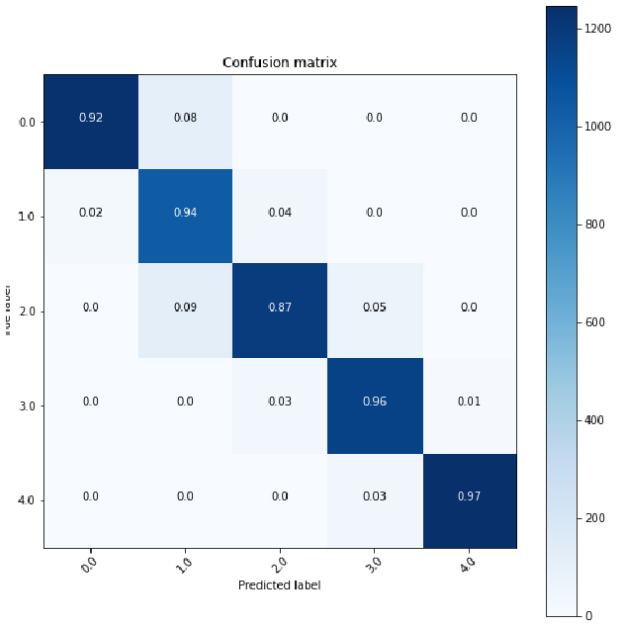

Figure 9. Confusion Matrix of Training with LR 0.0001

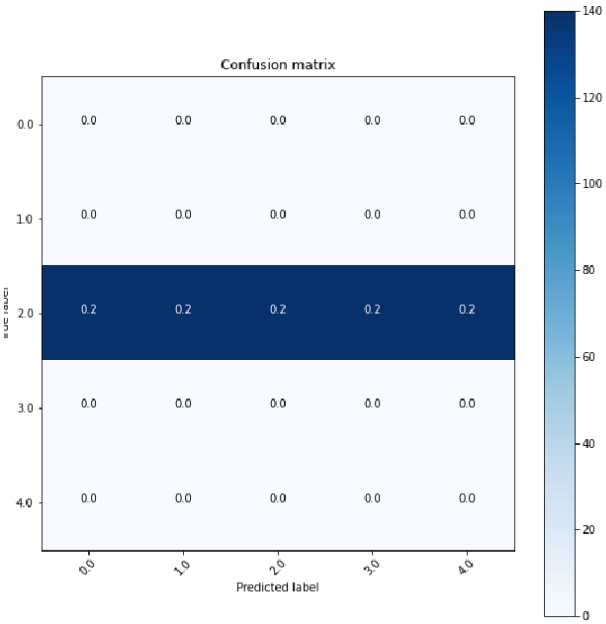

Figure 11. Confusion Matrix of Validation with LR 0.001

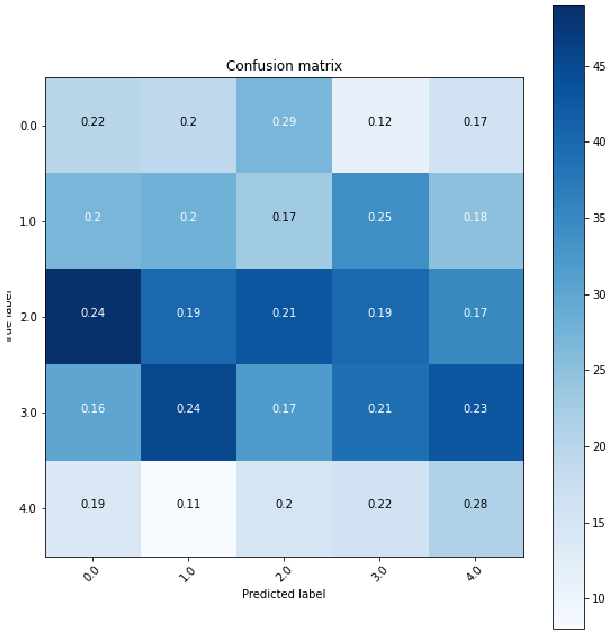

Figure 12. Confusion Matrix of Validation with LR 0.0001

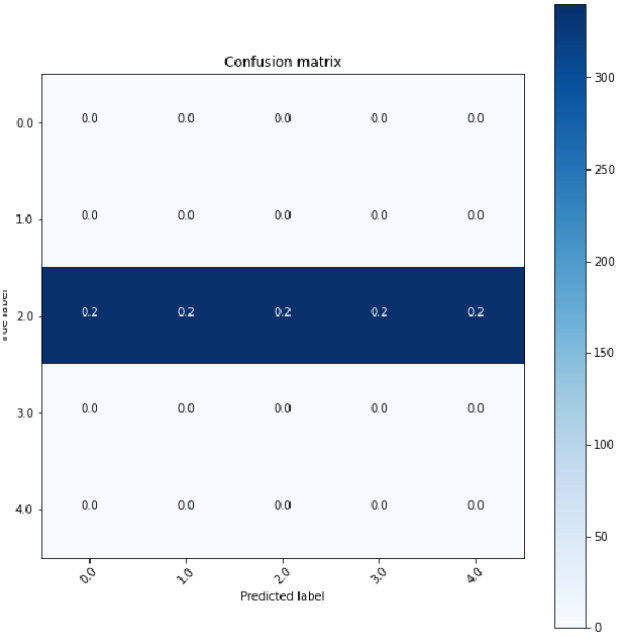

Figure 13. Confusion Matrix of Test with LR 0.01

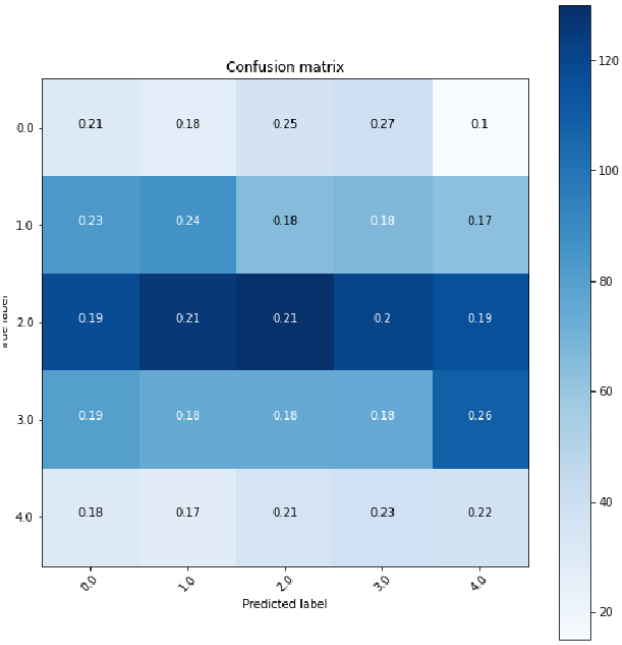

Figure 14. Confusion Matrix of Test with LR 0.001

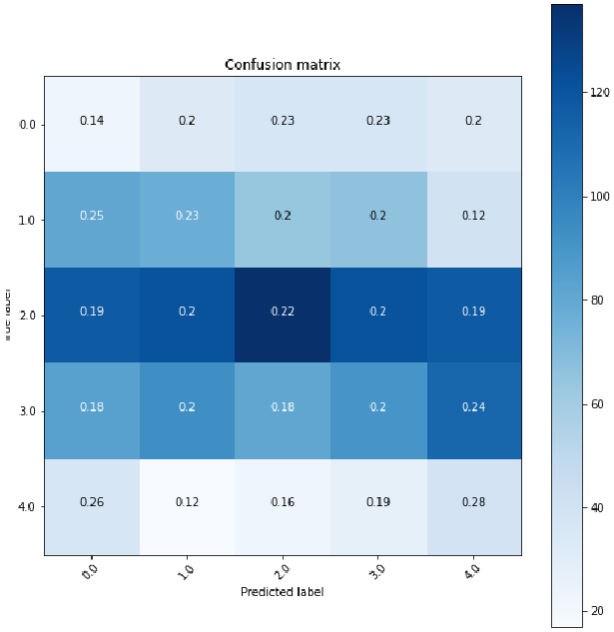

Figure 15. Confusion Matrix of Test with LR 0.0001

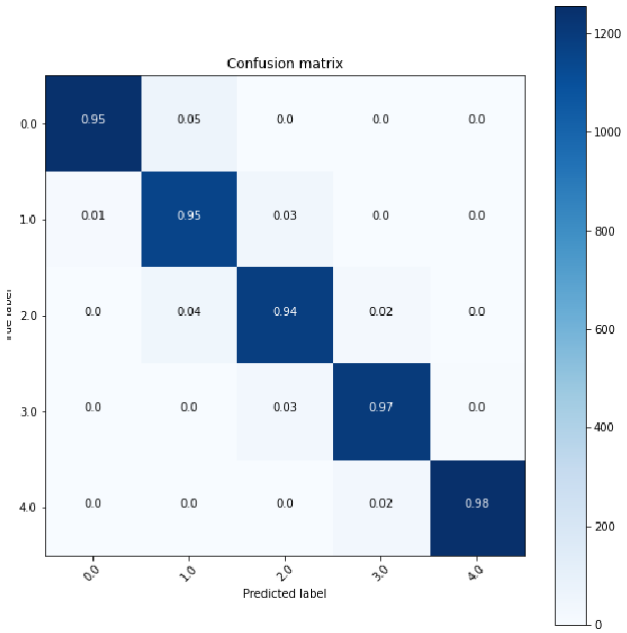

Figure 16. Confusion Matrix of Training with 12 Attention Heads

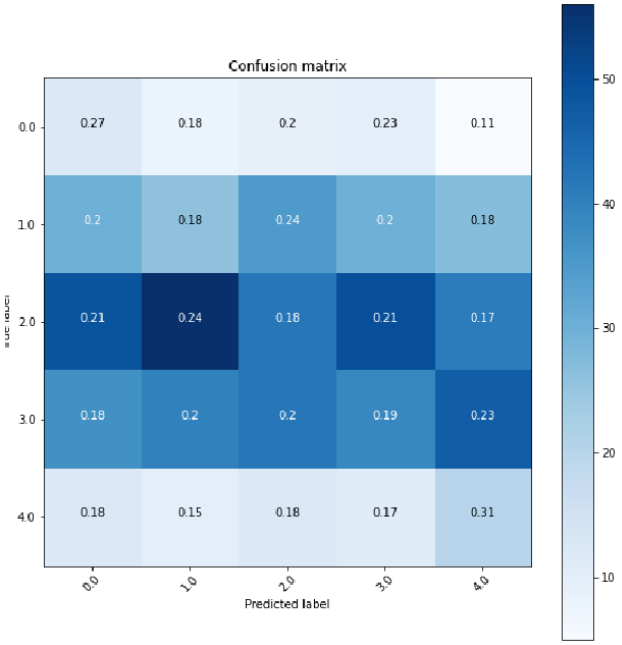

Figure 18. Confusion Matrix of Validation with 12 Attention Heads

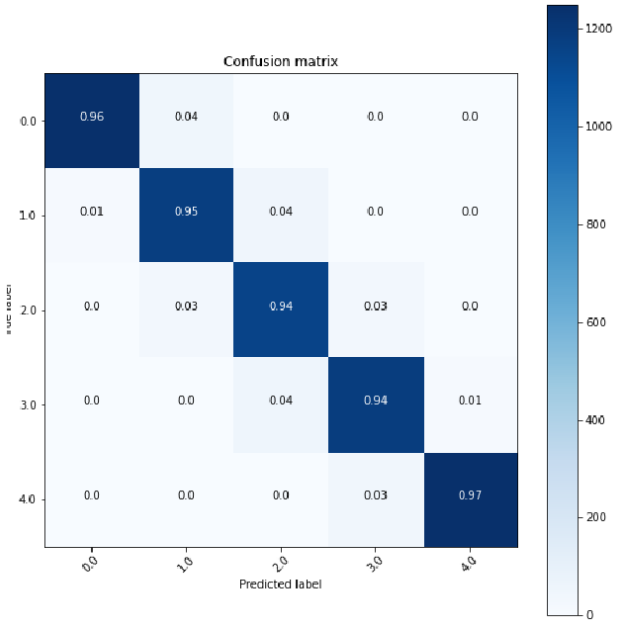

Figure 17. Confusion Matrix of Training with 3 Attention Heads

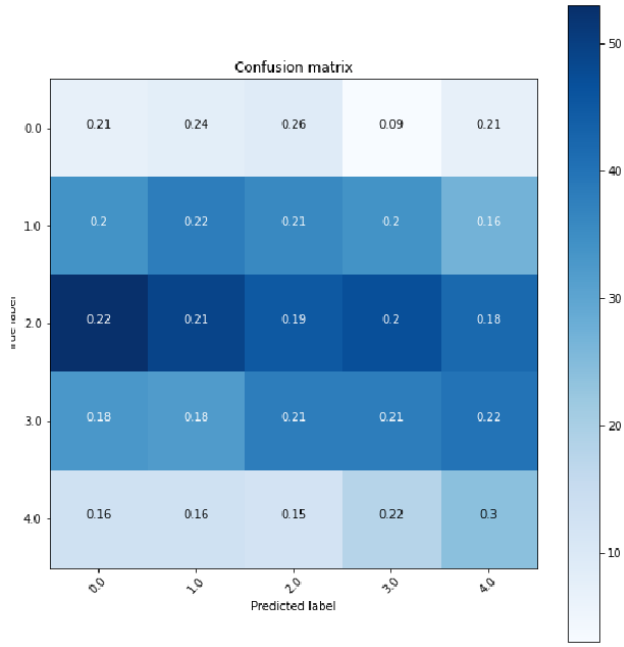

Figure 19. Confusion Matrix of Validation with 3 Attention Heads

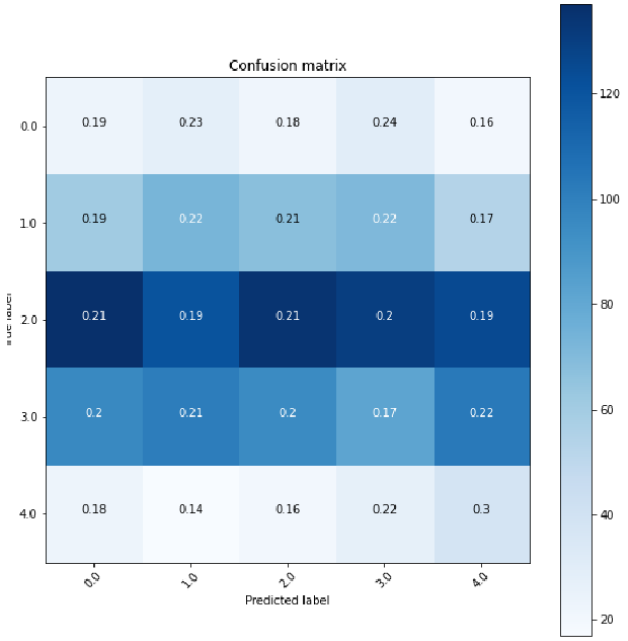

Figure 20. Confusion Matrix of Test with 12 Attention Heads

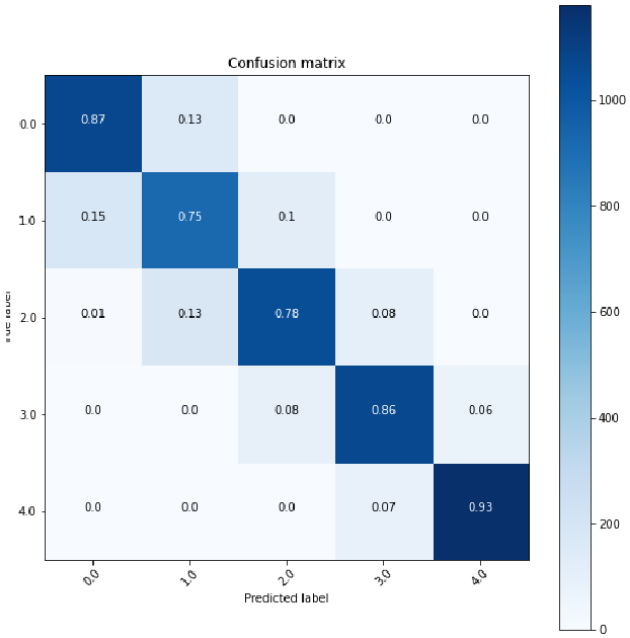

Figure 22. Confusion Matrix of Training with 1 Attention Layer

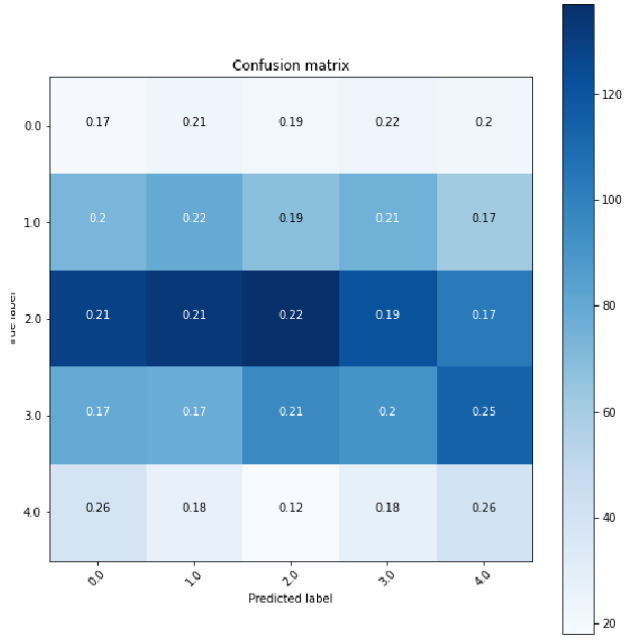

Figure 21. Confusion Matrix of Test with 3 Attention Heads

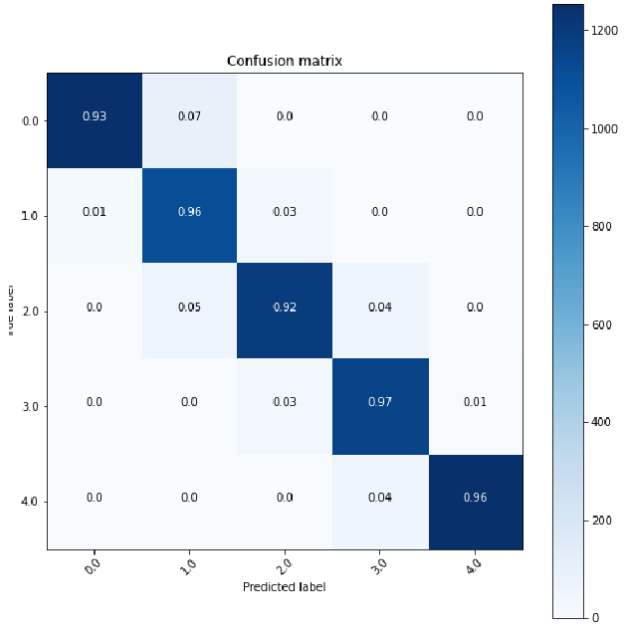

Figure 23. Confusion Matrix of Training with 5 Attention layers

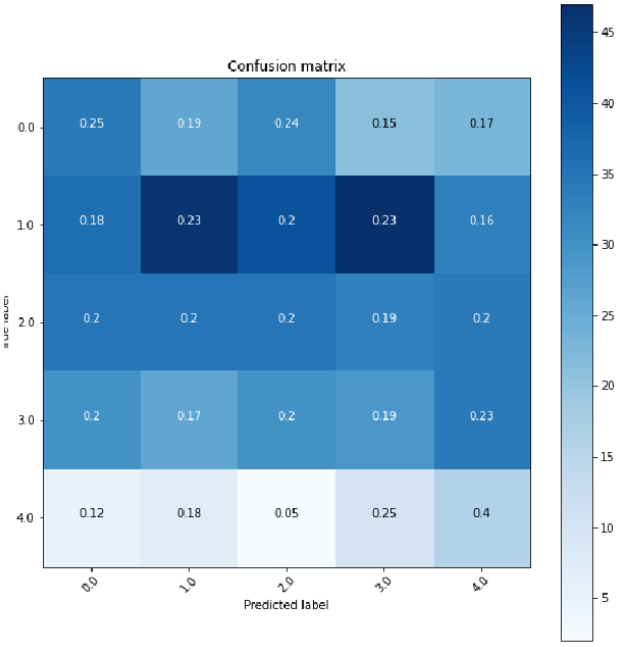

Figure 24. Confusion Matrix of Validation with 1 Attention Layer

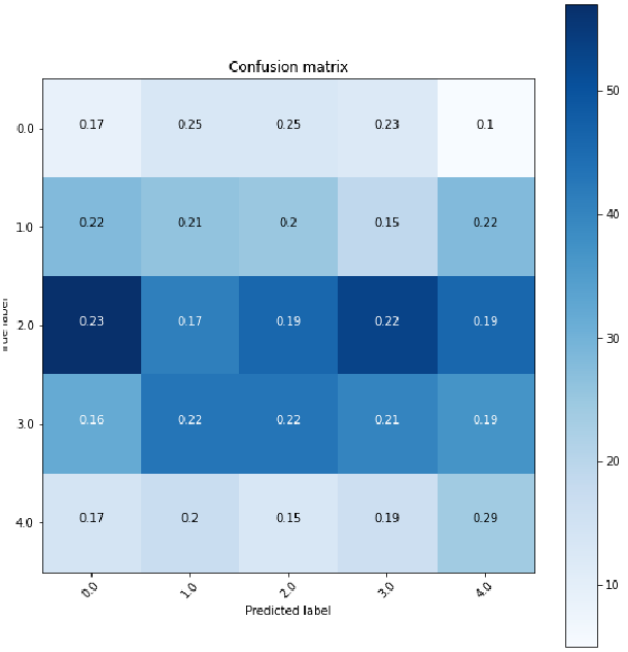

Figure 25. Confusion Matrix of Validation with 5 Attention layers

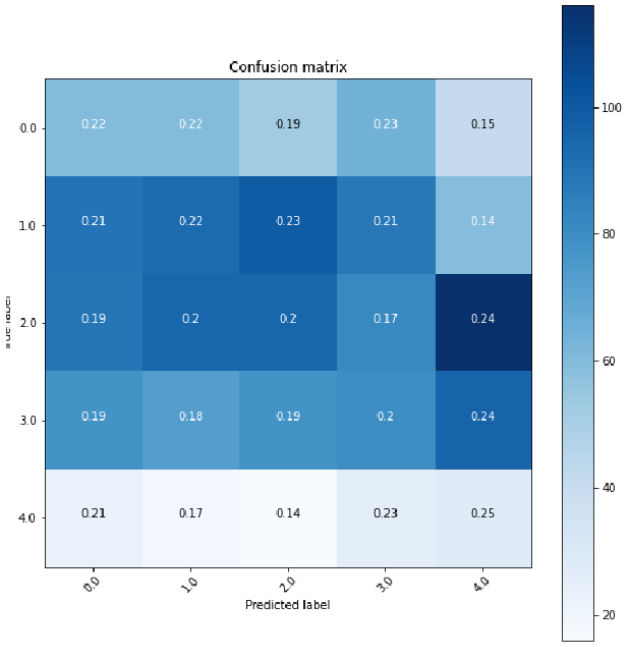

Figure 26. Confusion Matrix of Test with 1 Attention Layer

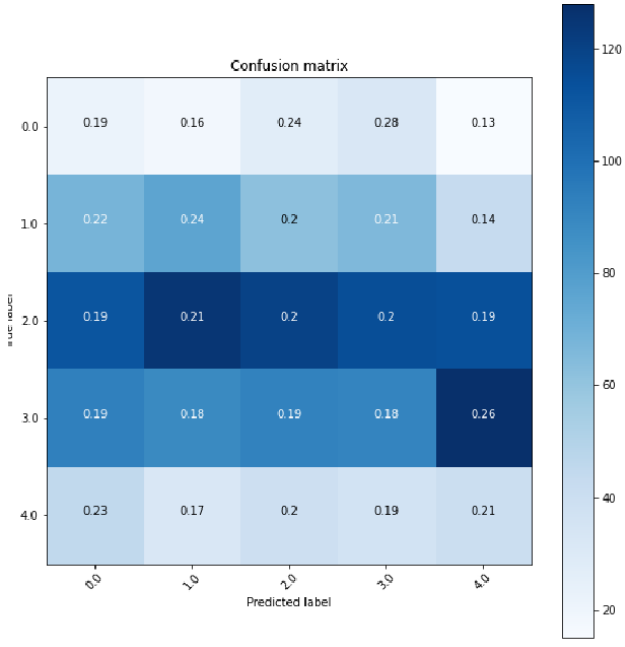

Figure 27. Confusion Matrix of Test with 5 Attention layers

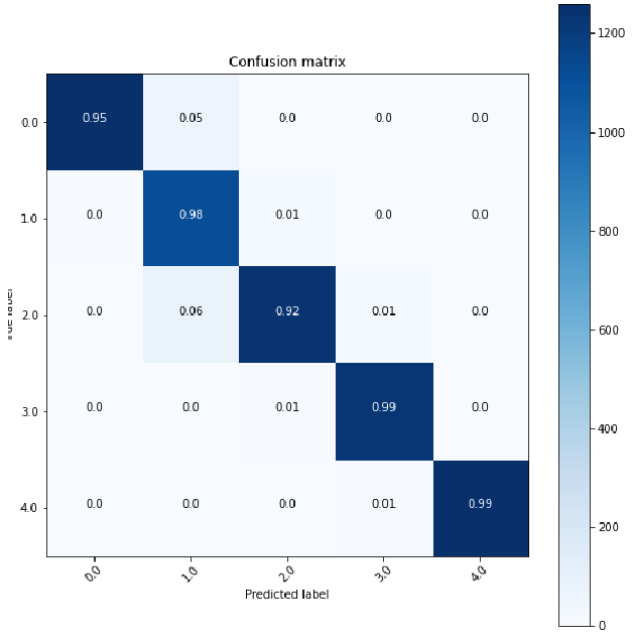

Figure 28. Confusion Matrix of Training with dimension of the Hidden Representation 64

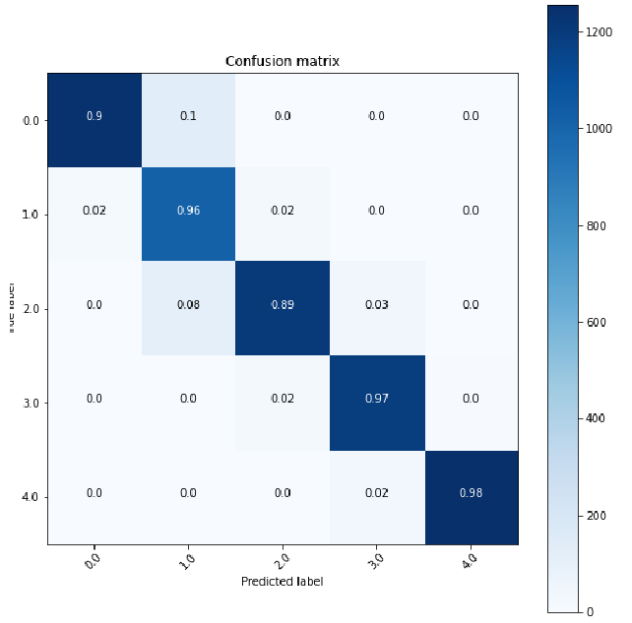

Figure 29. Confusion Matrix of Training with dimension of the Hidden Representation 128

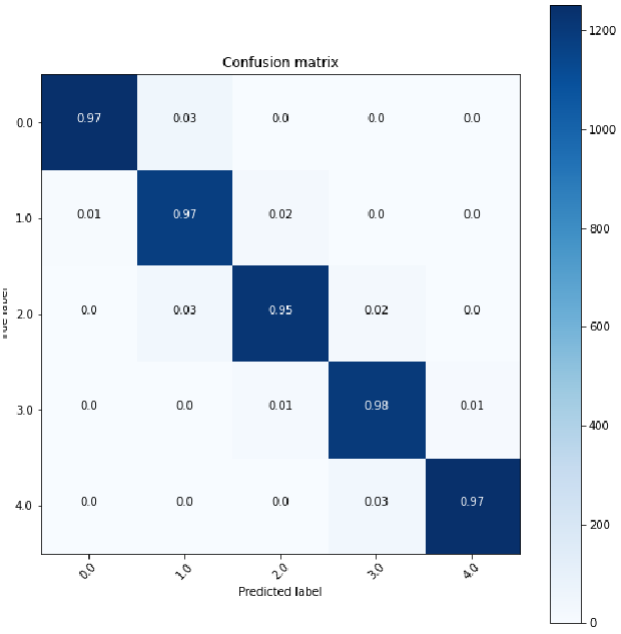

Figure 30. Confusion Matrix of Training with dimension of the Hidden Representation 16

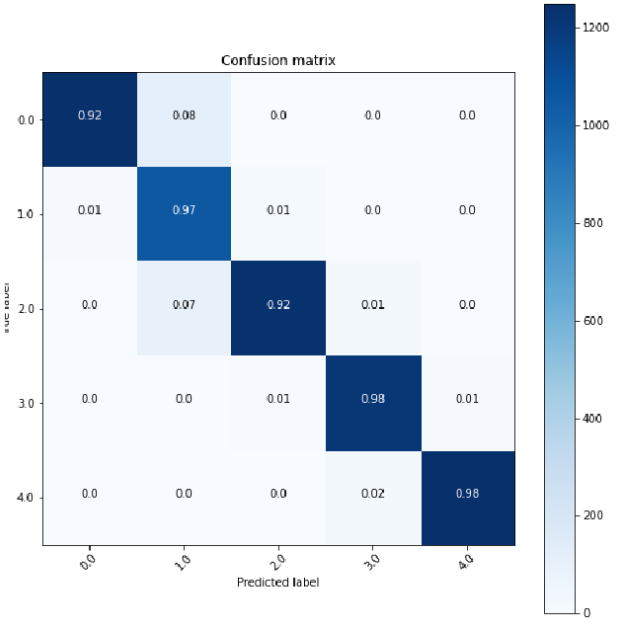

Figure 31. Confusion Matrix of Training with dimension of the Hidden Representation 256

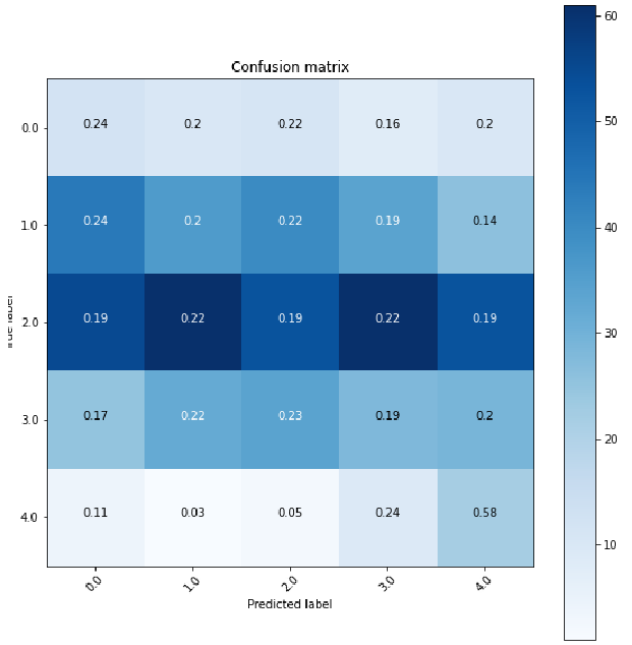

Figure 32. Confusion Matrix of Validation with dimension of the Hidden Representation 64

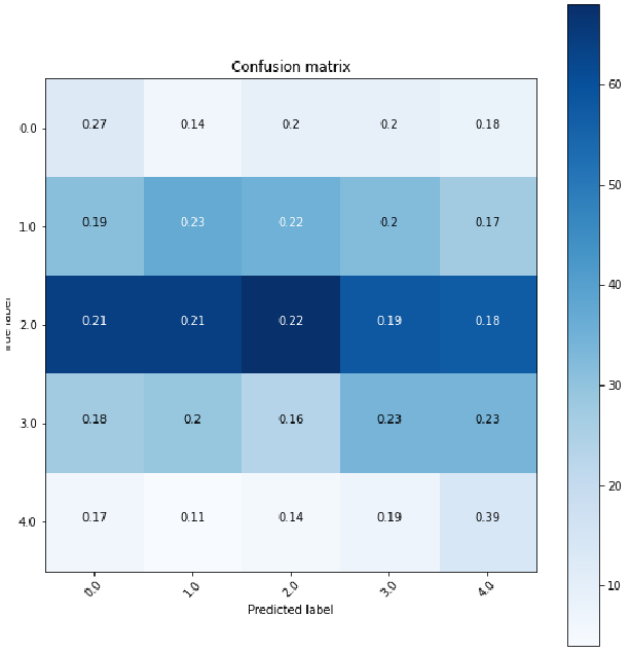

Figure 33. Confusion Matrix of Validation with dimension of the Hidden Representation 128

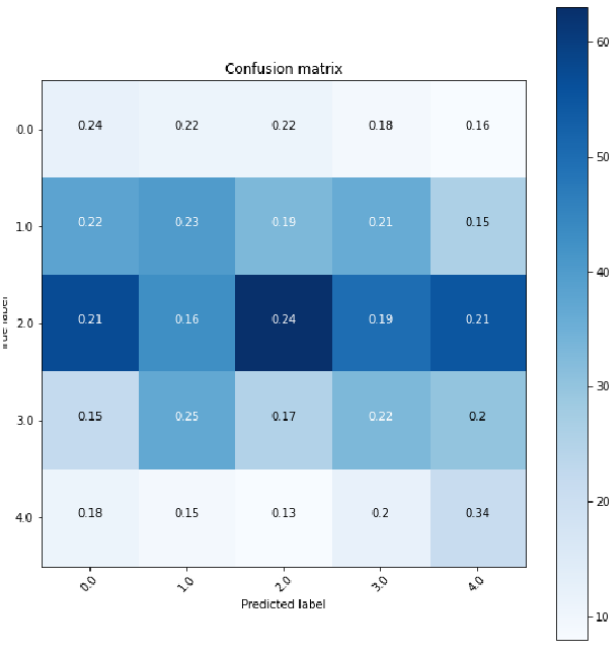

Figure 34. Confusion Matrix of Validation with dimension of the Hidden Representation 16

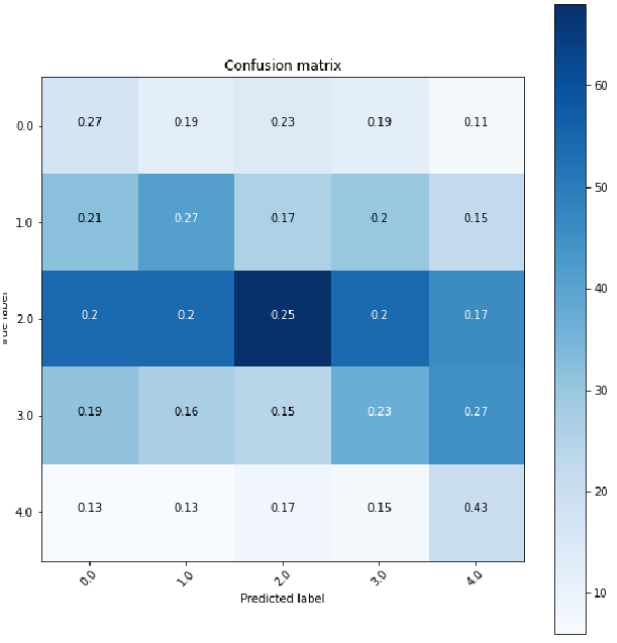

Figure 35. Confusion Matrix of Validation with dimension of the Hidden Representation 256

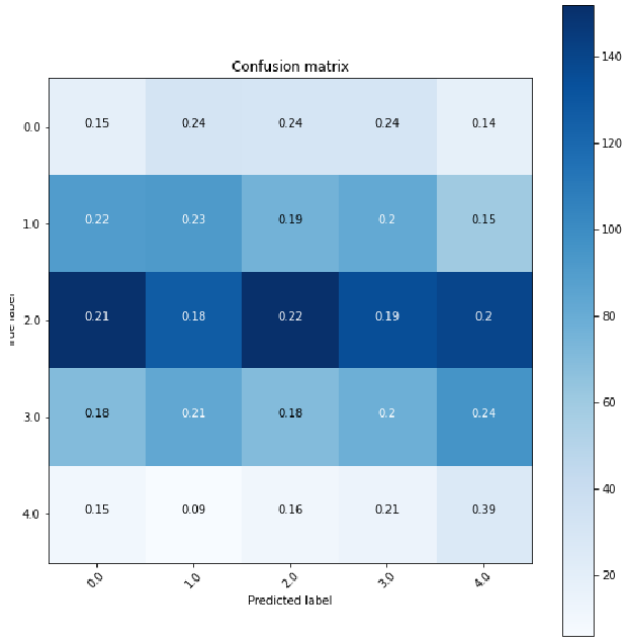

Figure 36. Confusion Matrix of Test with dimension of the Hidden Representation 64

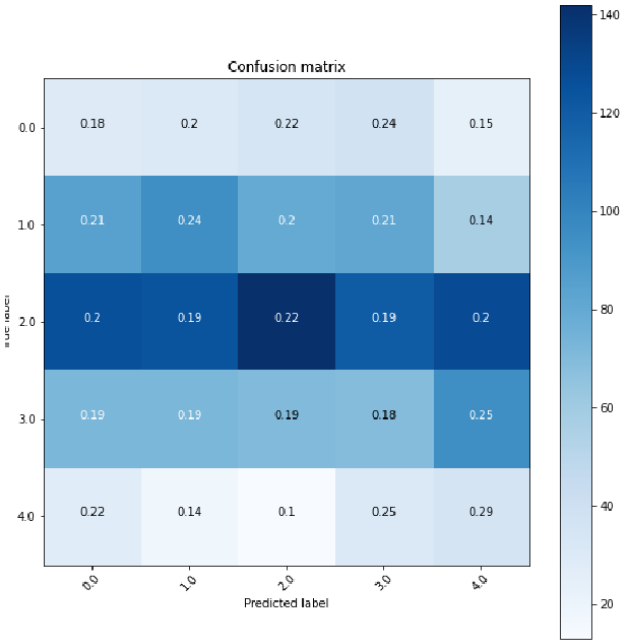

Figure 38. Confusion Matrix of Test with dimension of the Hidden Representation 16

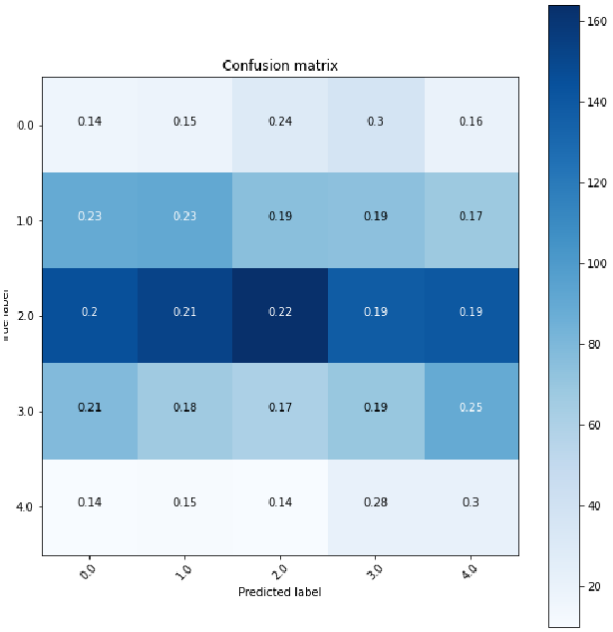

Figure 37. Confusion Matrix of Test with dimension of the Hidden Representation 128

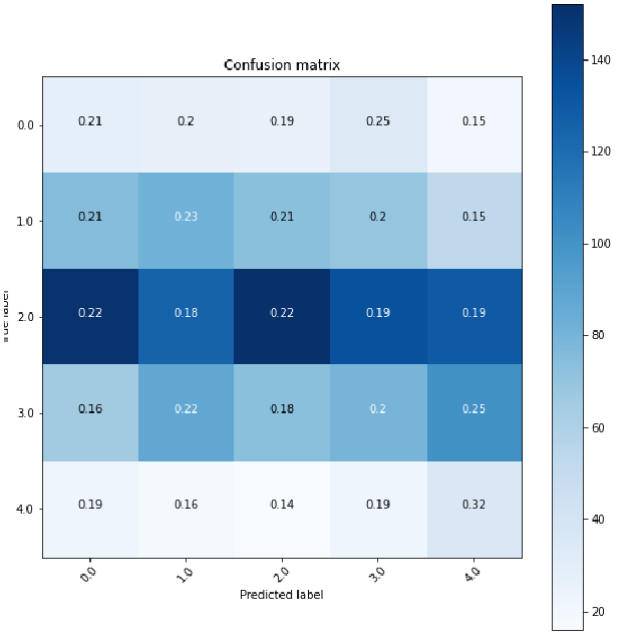

Figure 39. Confusion Matrix of Test with dimension of the Hidden Representation 256

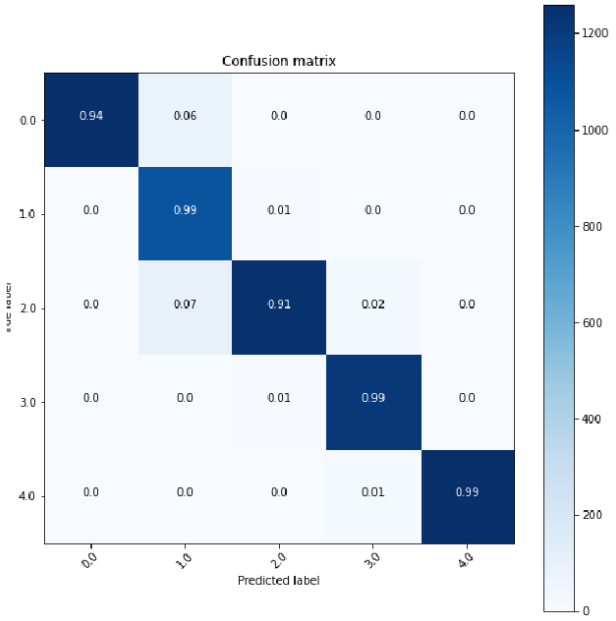

Figure 40. Confusion Matrix of Training with Batch Size 128

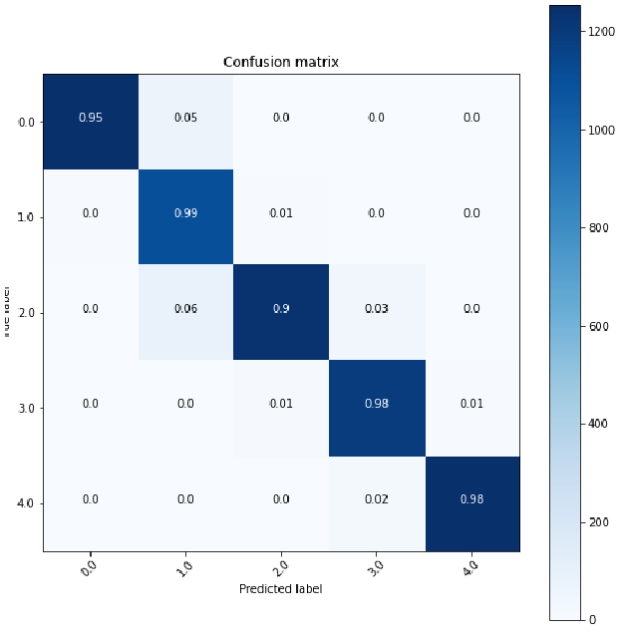

Figure 41. Confusion Matrix of Training with Batch Size 64

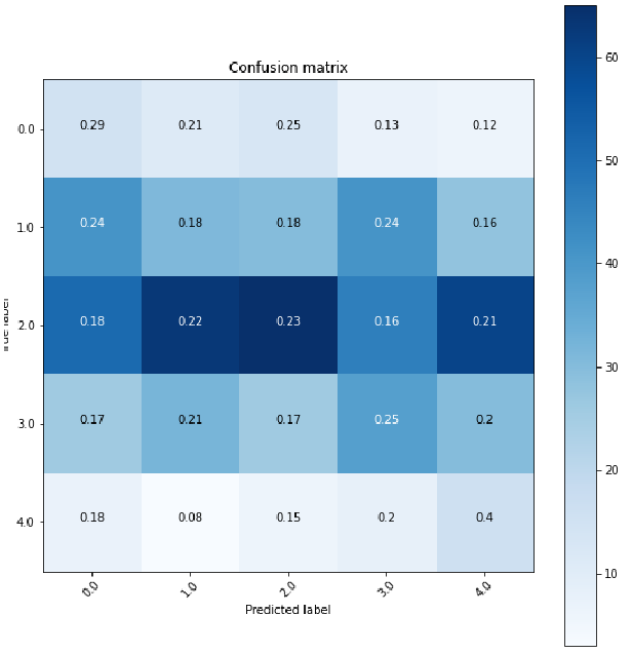

Figure 42. Confusion Matrix of Validation with Batch Size 128

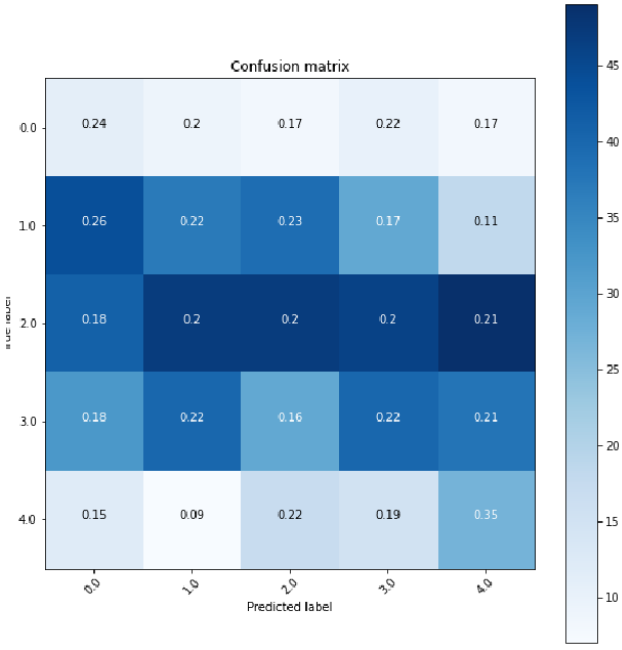

Figure 43. Confusion Matrix of Validation with Batch Size 64

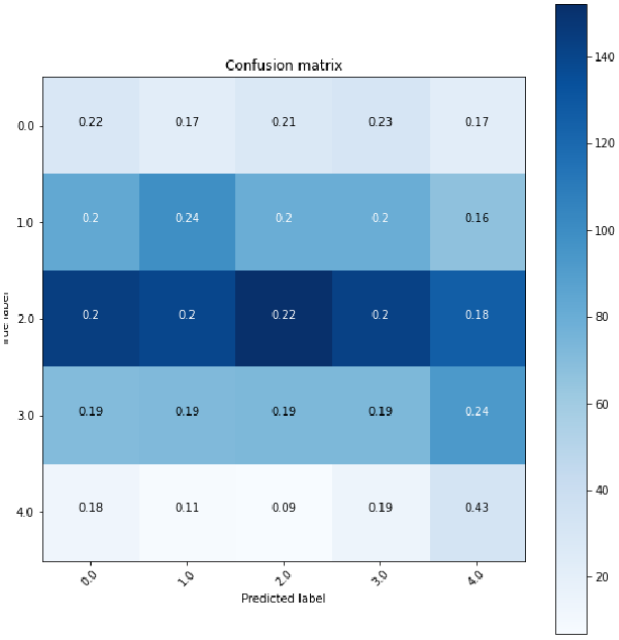

Figure 44. Confusion Matrix of Test with Batch Size 128

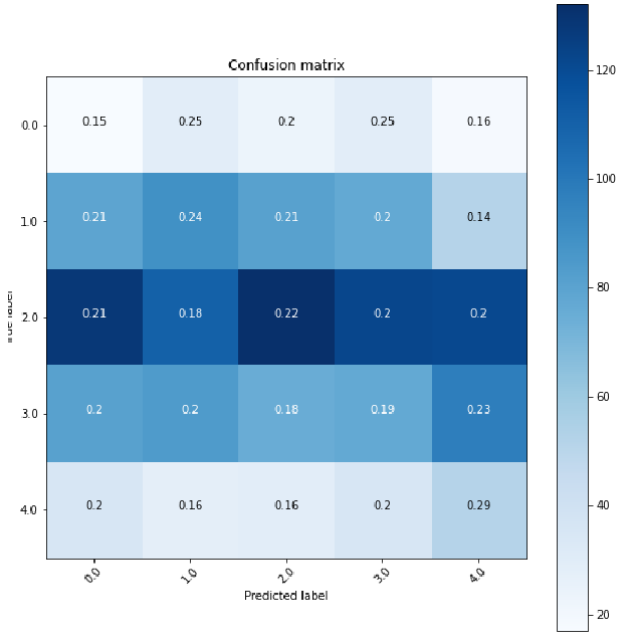

Figure 45. Confusion Matrix of Test with Batch Size 64

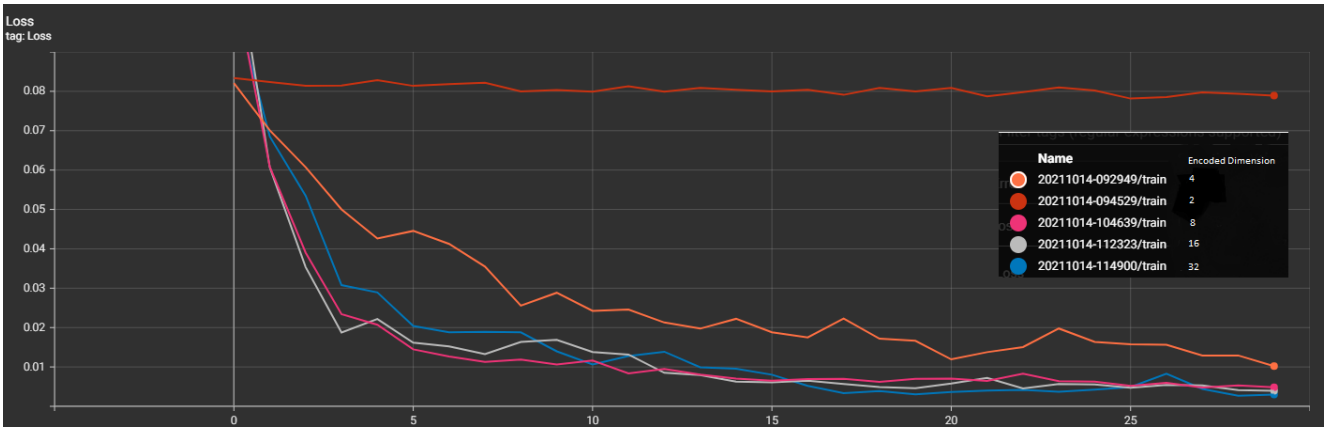

Figure 46. Train Loss

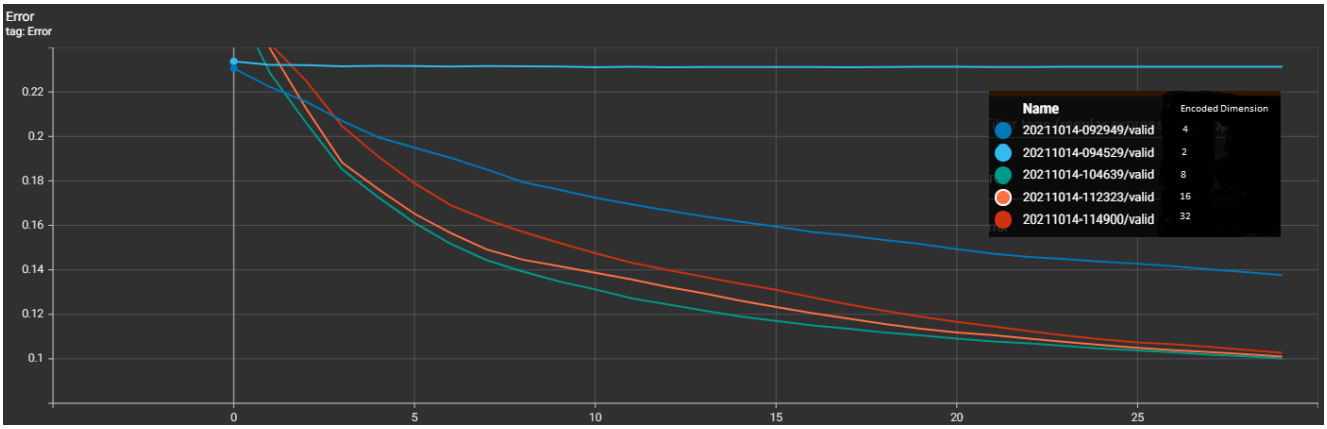

Figure 47. Validation Error

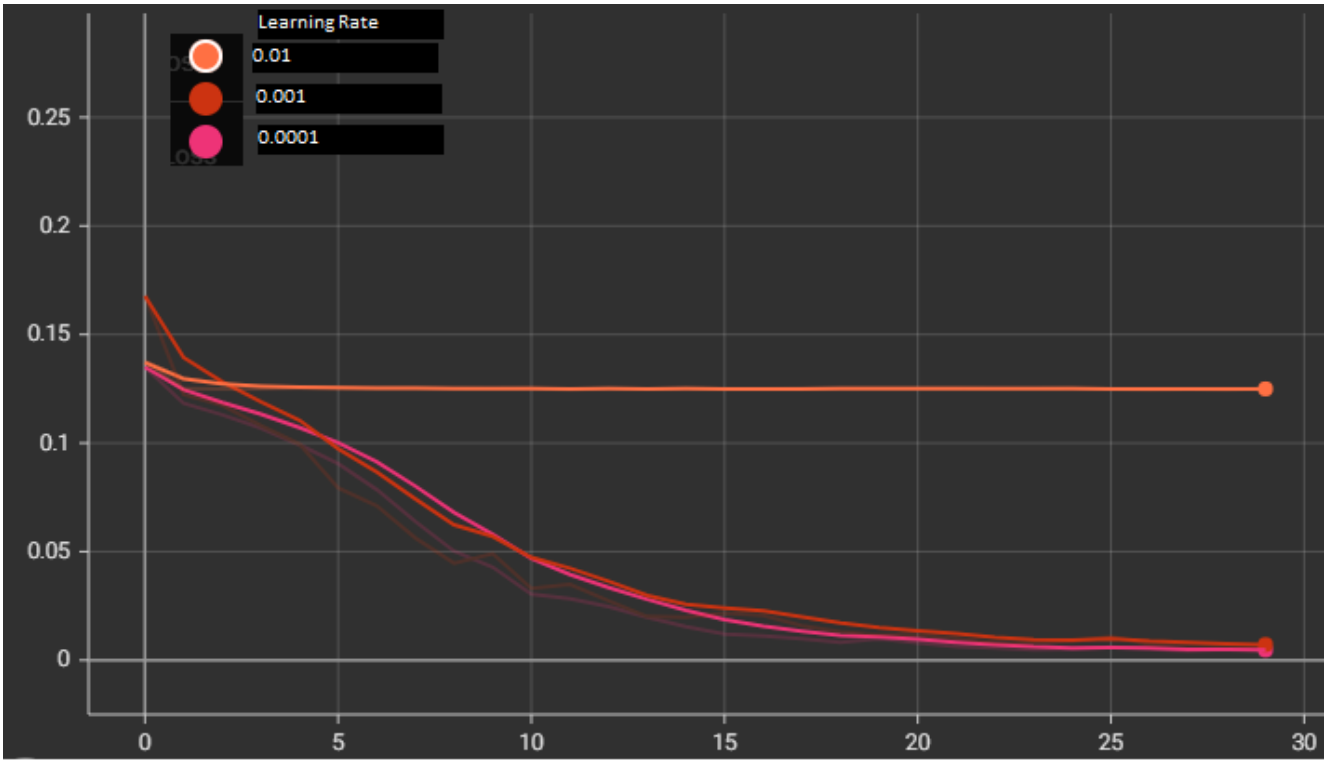

Figure 48. Training Loss changing Learning Rate

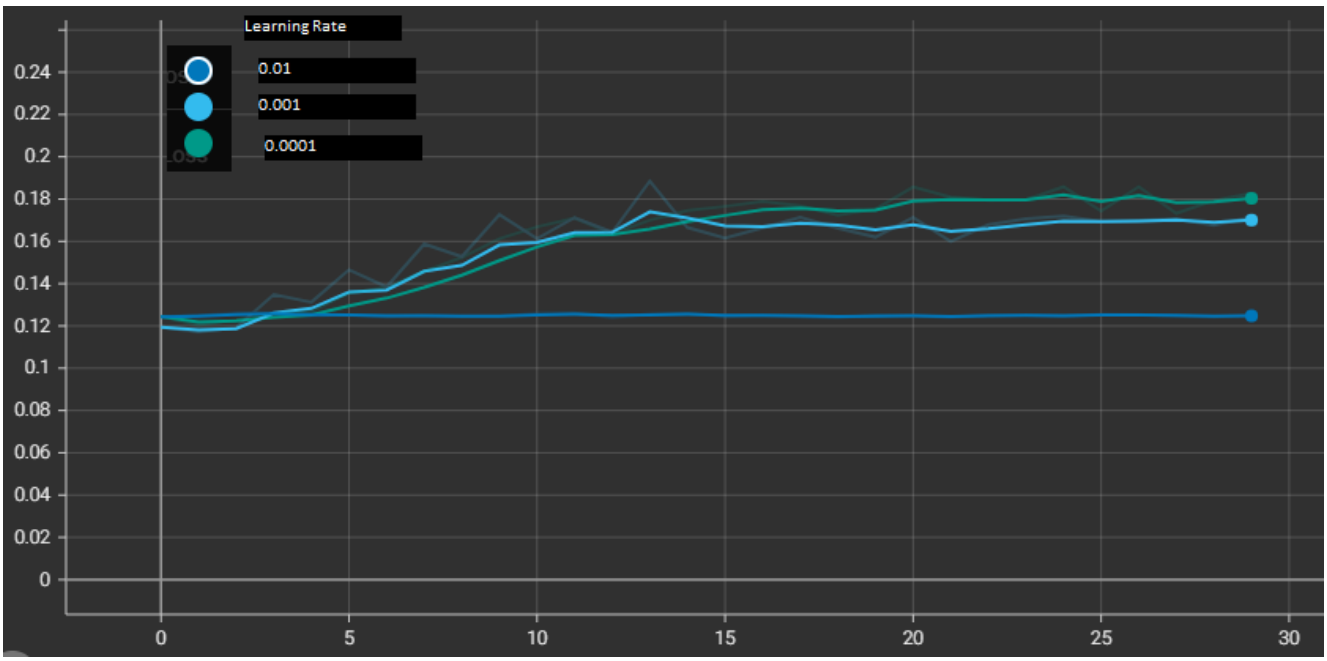

Figure 49. Validation Loss changing Learning Rate

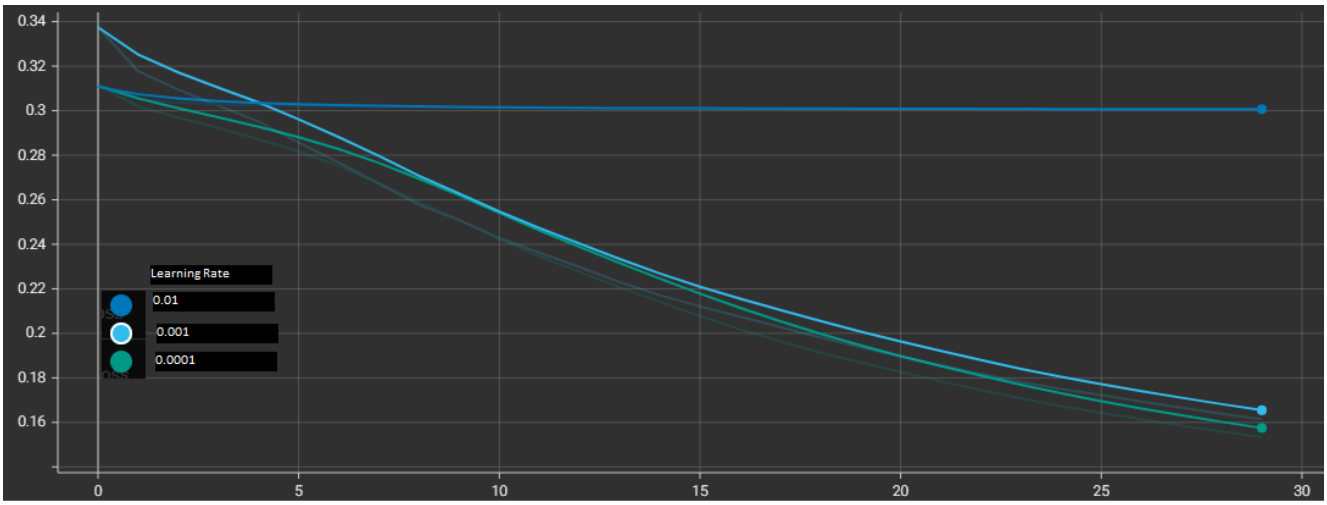

Figure 50. Validation Mean Absolute Error changing Learning Rate

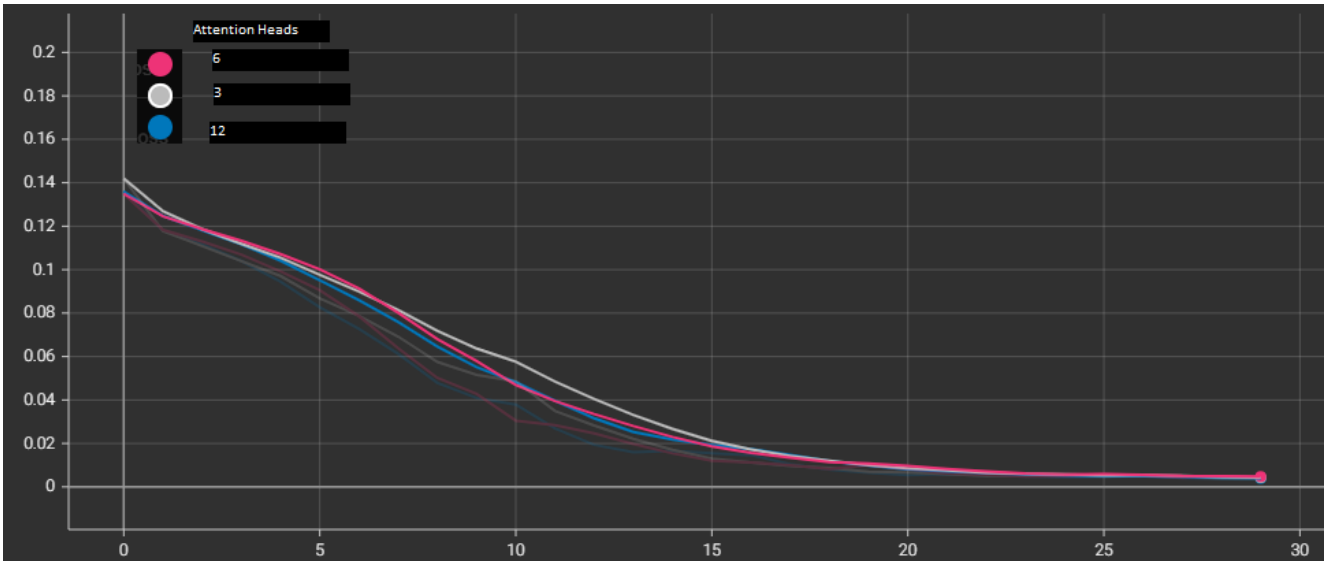

Figure 51. Training Loss changing the number of Attention Heads

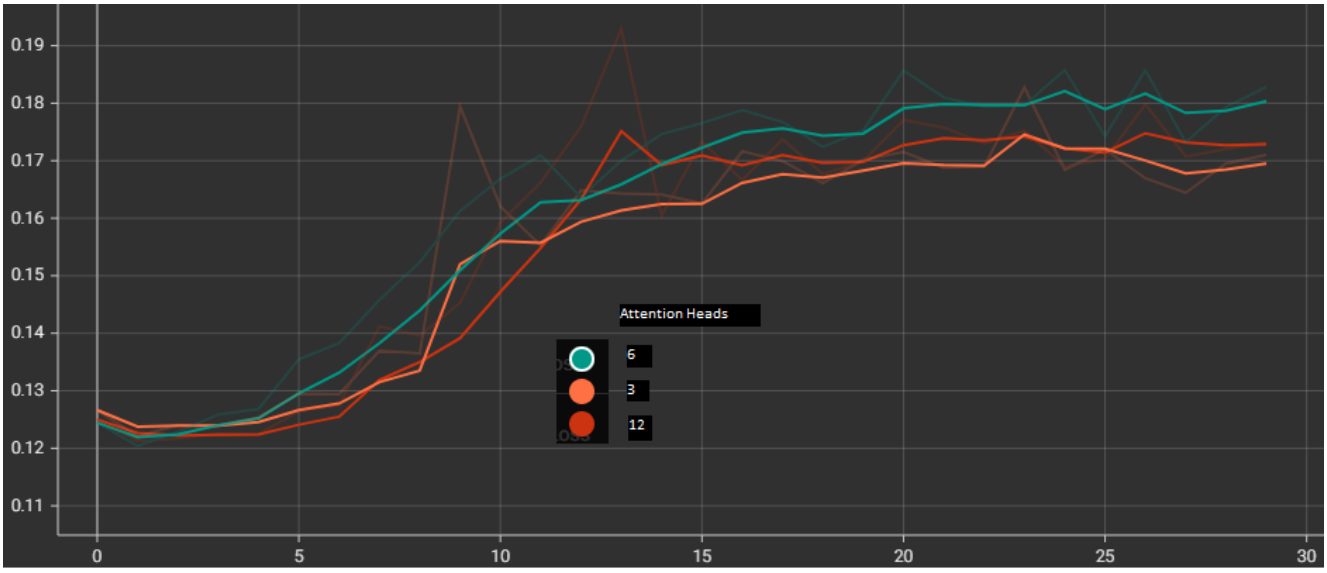

Figure 52. Validation Loss changing the number of Attention Heads

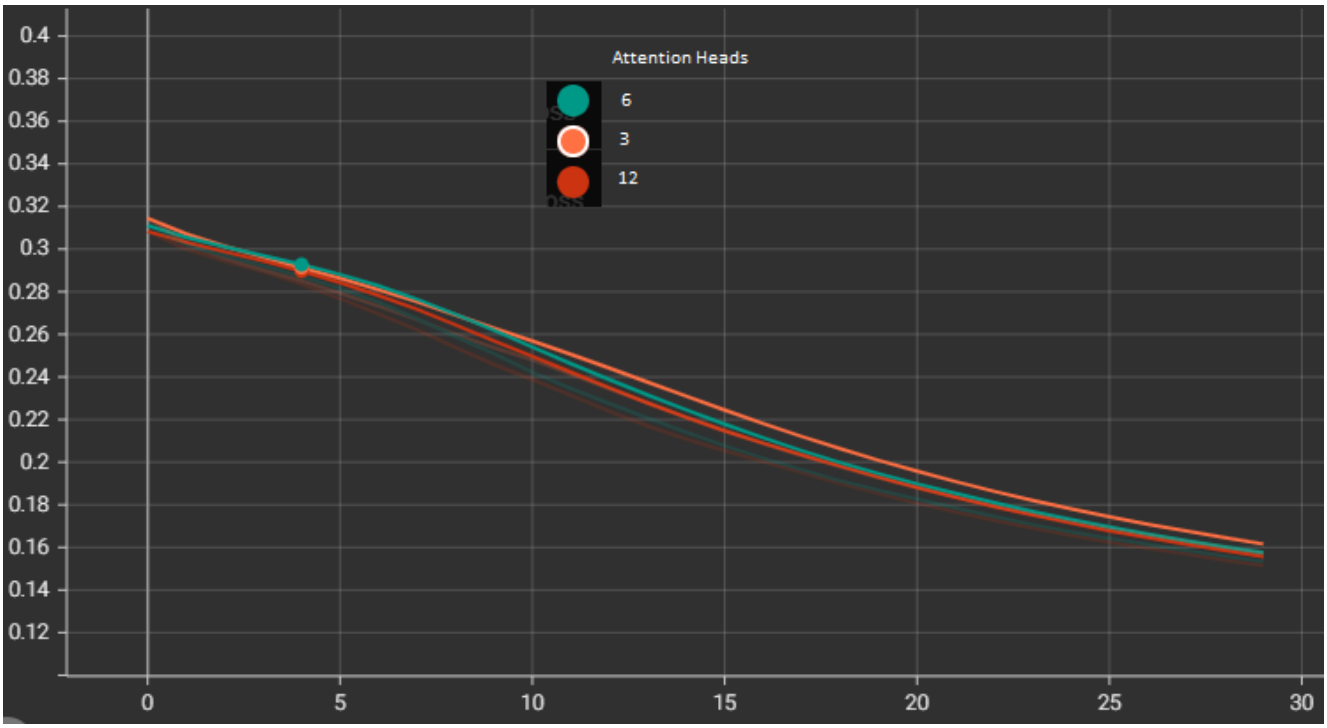

Figure 53. Validation Mean Absolute Error changing the number of Attention Heads

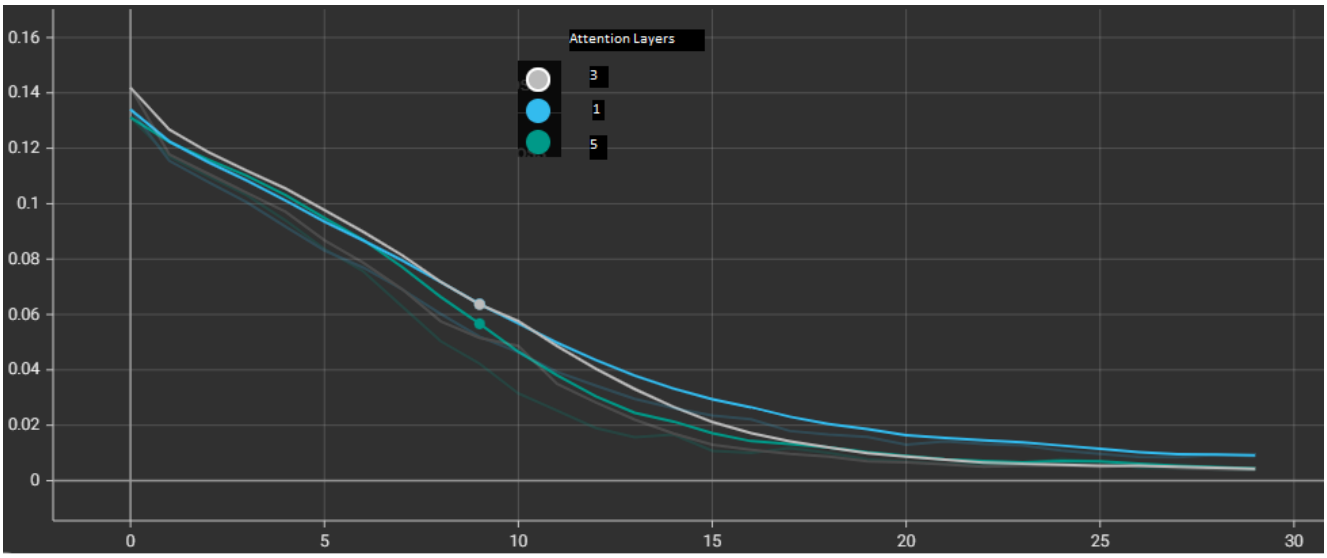

Figure 54. Training Loss changing the number of Attention Layers

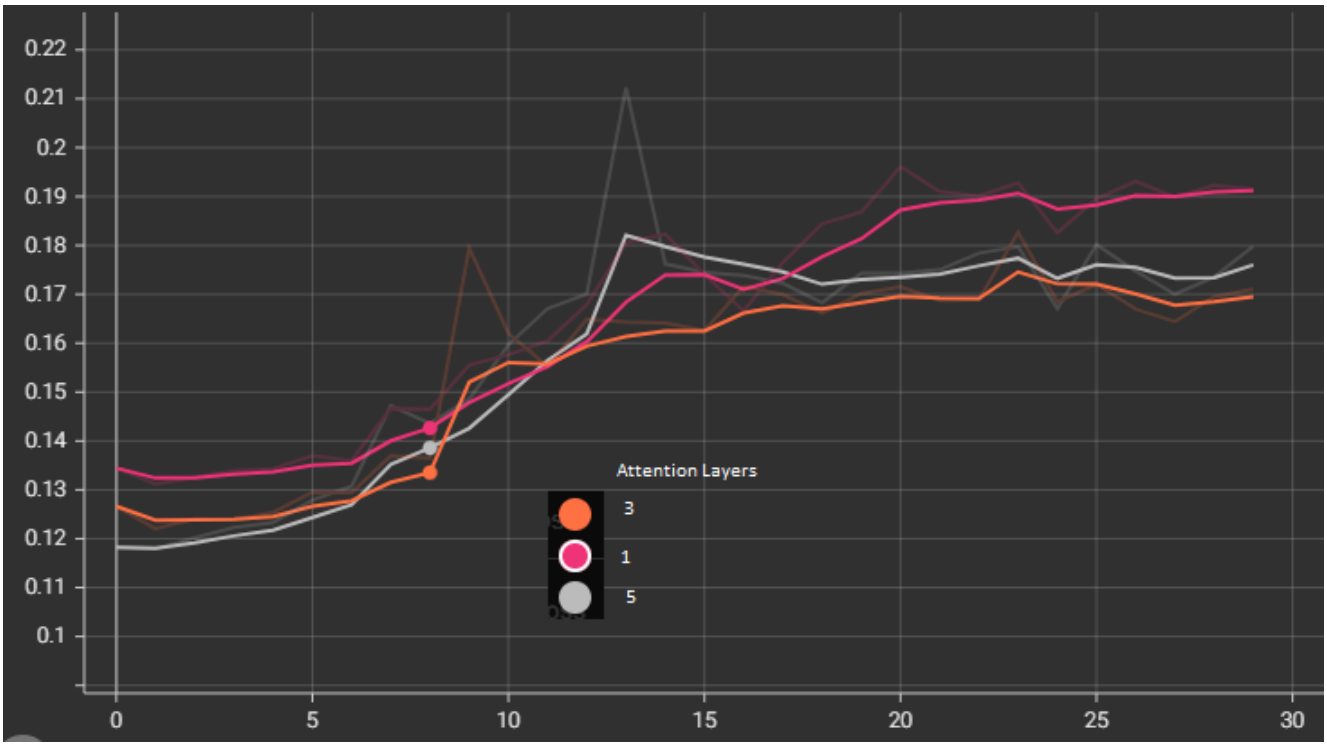

Figure 55. Validation Loss changing the number of Attention Layers

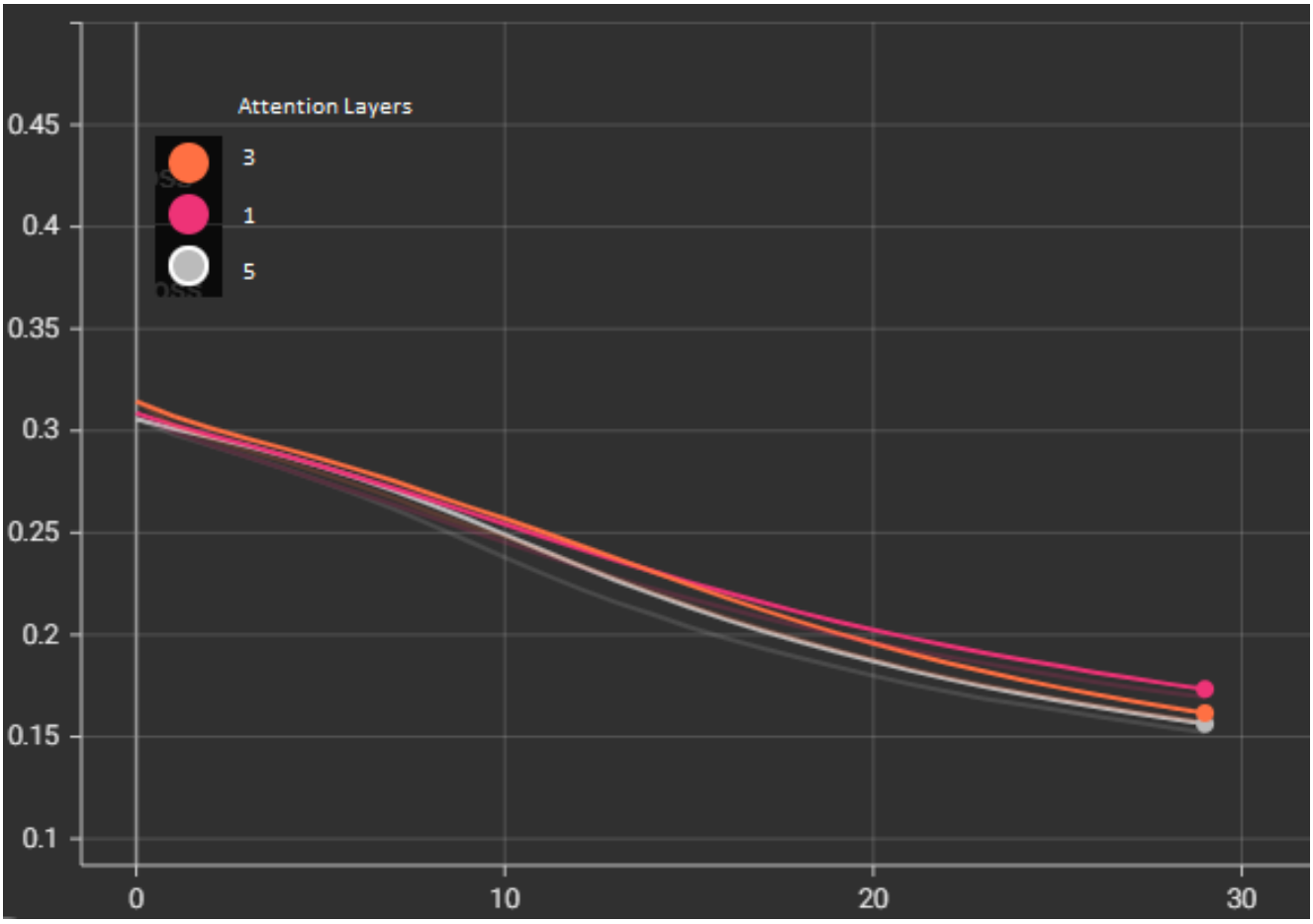

Figure 56. Validation Mean Absolute Error changing the number of Attention Layers

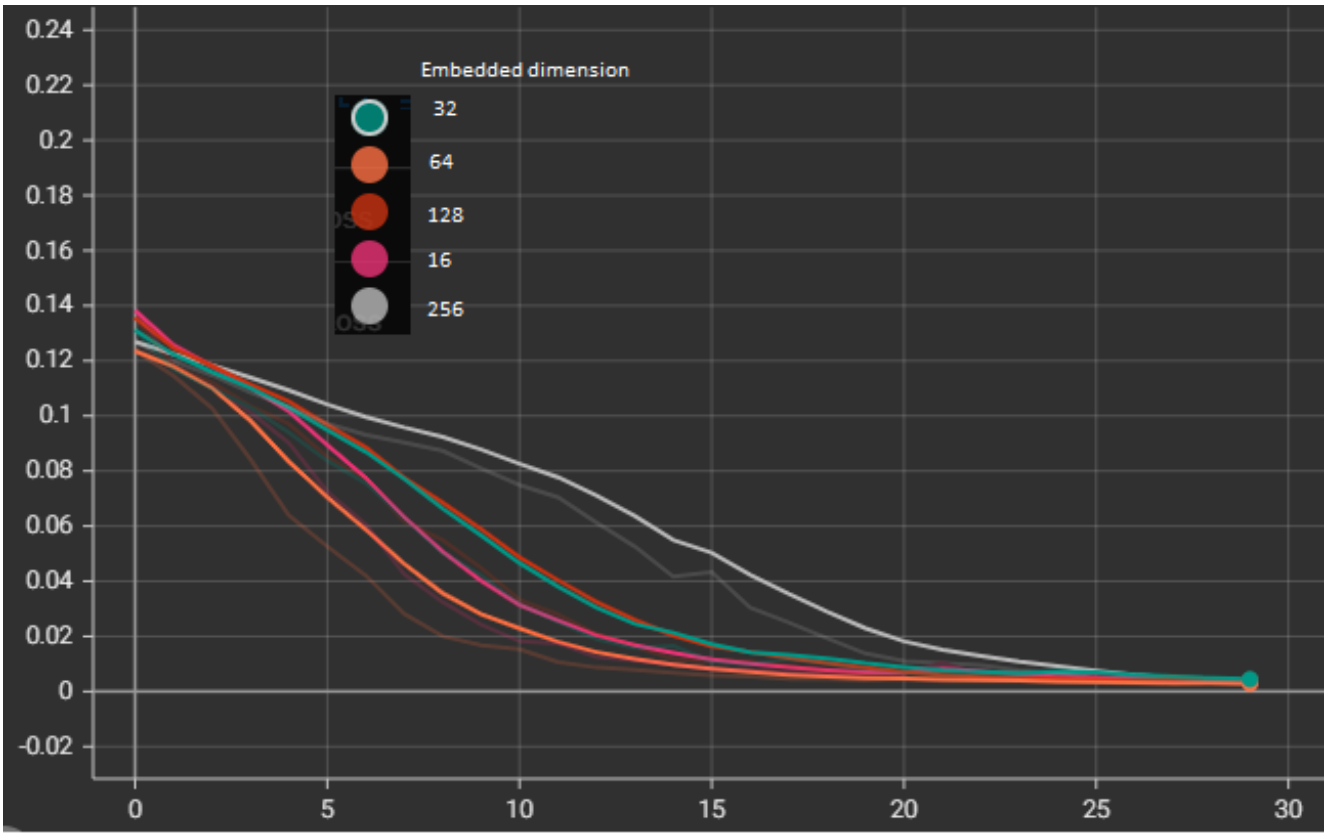

Figure 57. Training Loss changing the dimension of the Hidden Representation

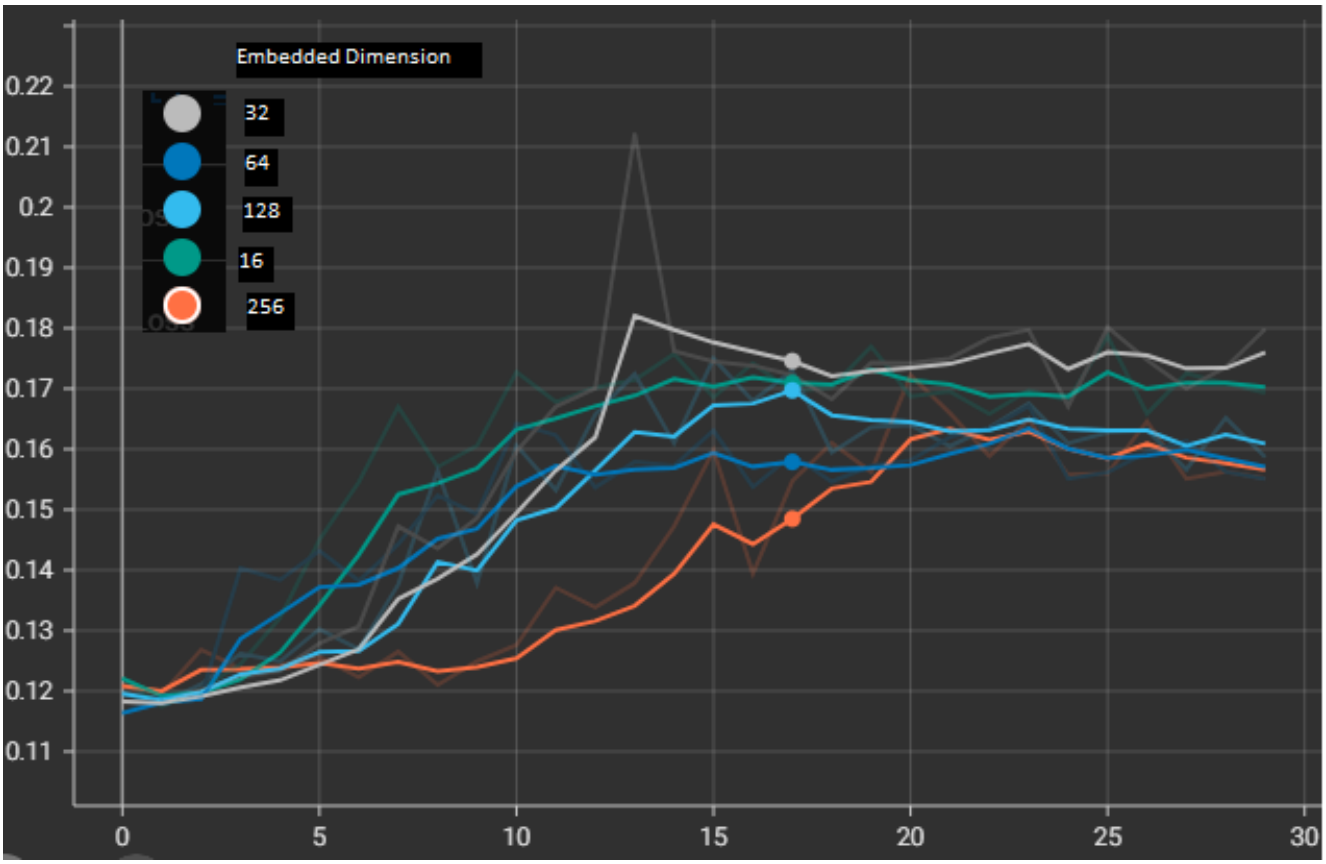

Figure 58. Validation Loss changing the dimension of the Hidden Representation

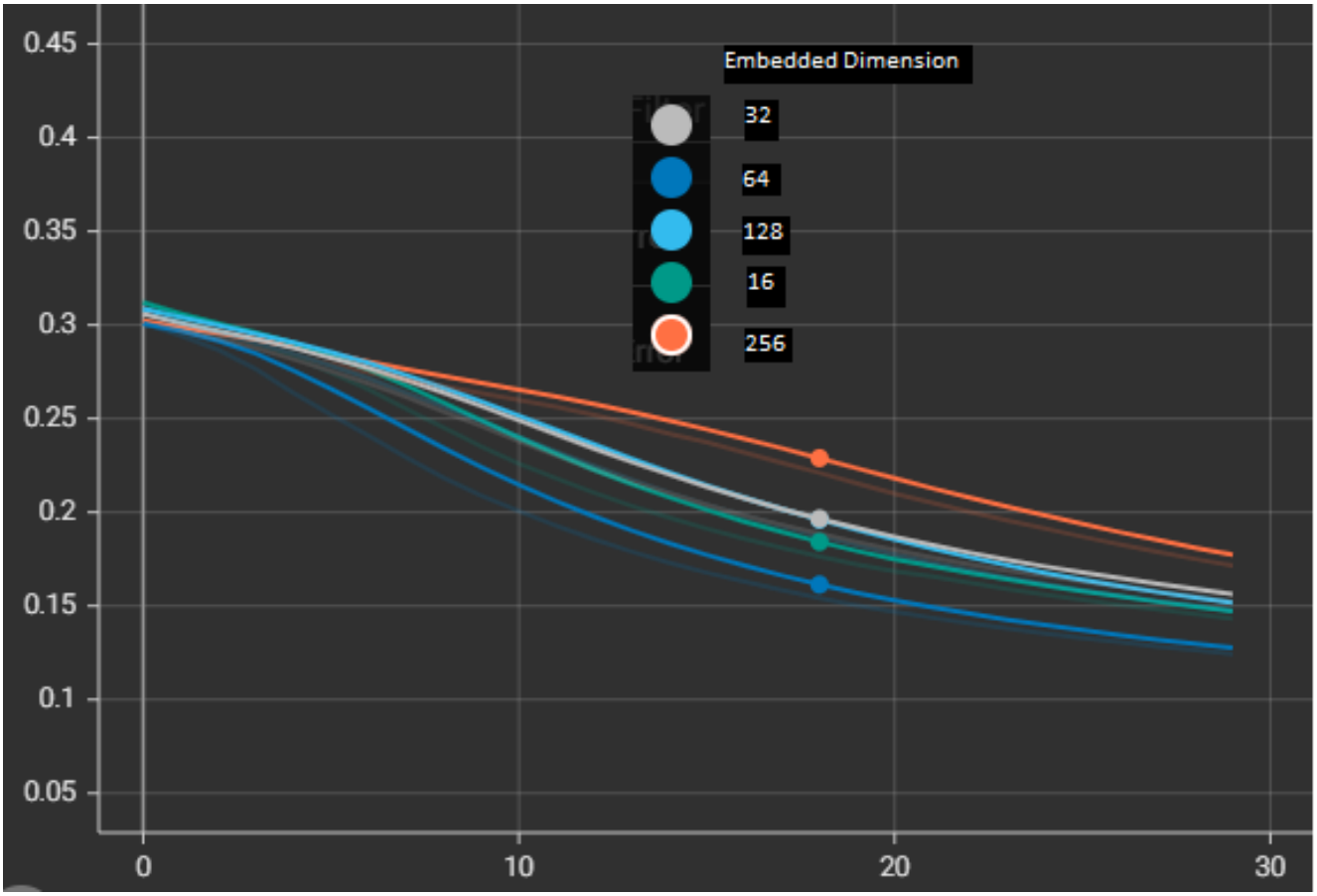

Figure 59. Validation Mean Absolute Error changing the dimension of the Hidden Representation

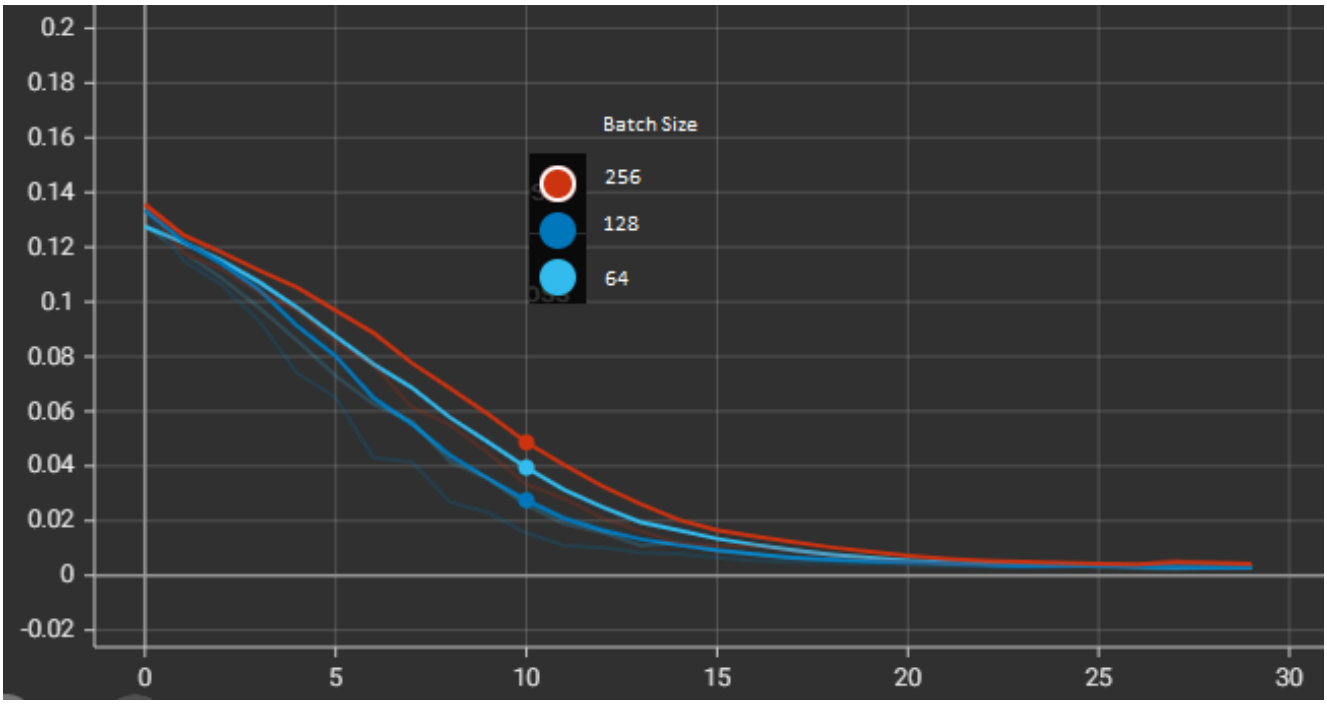

Figure 60. Training Loss changing the Batch Size

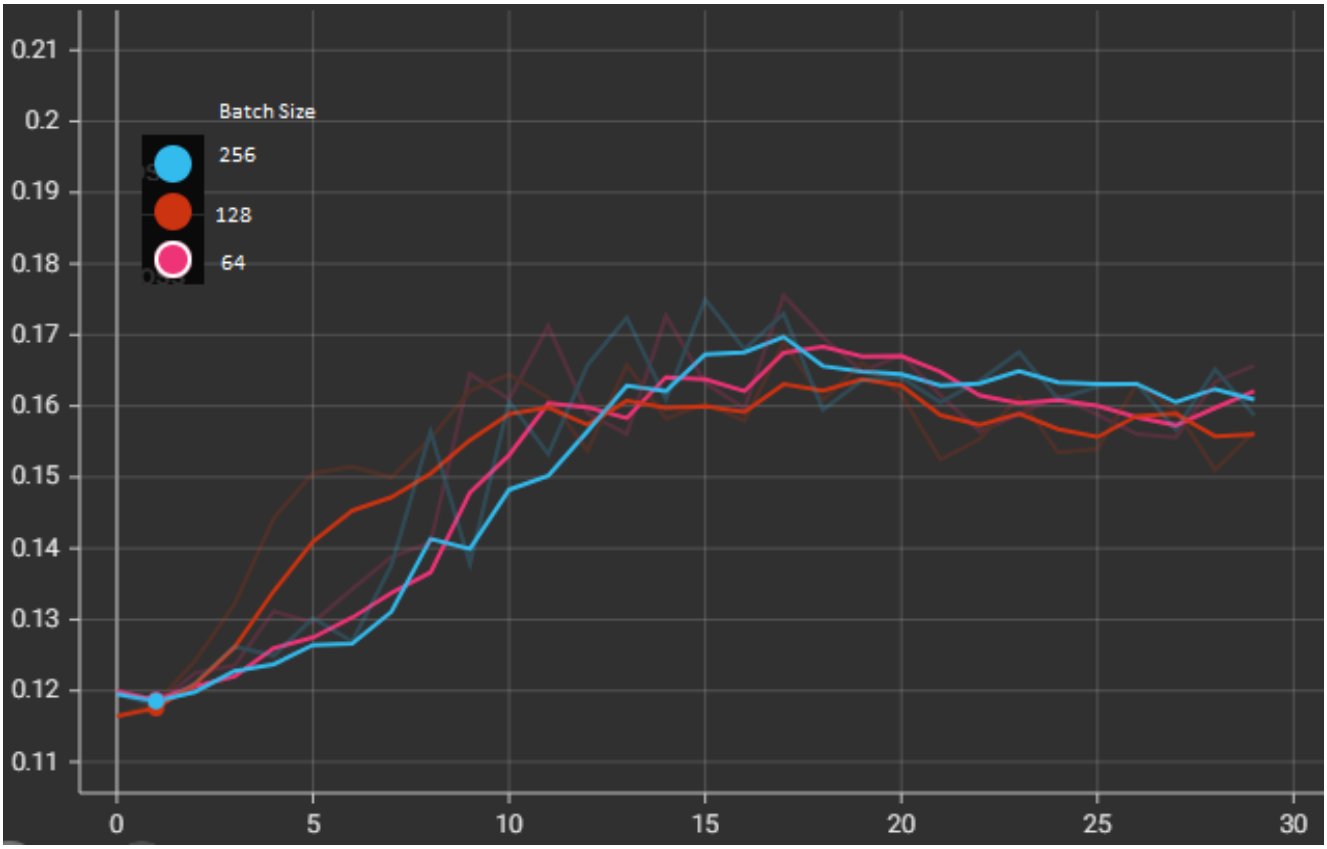

Figure 61. Validation Loss changing the Batch Size

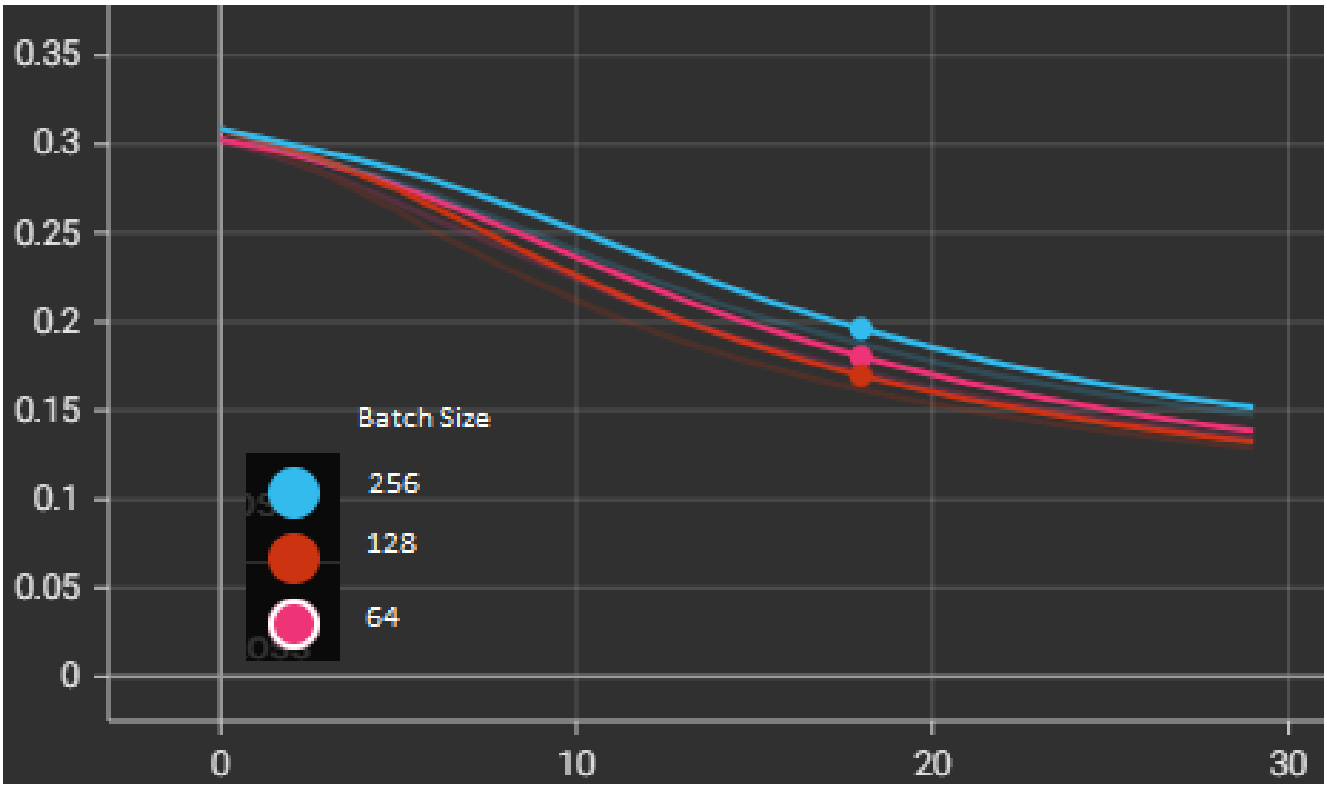

Figure 62. Validation Mean Absolute Error changing the Batch Size