An Exploration of Model Optimization with Advanced Multi-Layer Perceptron: Predicting Media Memorability task, Using Captions

Minkun Kim
Student ID: 18212693
Mail: minkun.kim4@mail.dcu.ie

ABSTRACT

In this paper, we mainly explore a neural network topology and seeking its improvement in performance by deploying various techniques such as parameter optimizations, an autoencoder, or learning rate scheduling. Although we achieved a baseline performance, particularly in predicting short-term scores, it seems that we have not discovered the appropriate network topology that works out with long-term scores.

1 INTRODUCTION

In this work, we investigated the use of caption to predict video memorability as it is hinted from the previous work that models using captions tend to give the better results. Rather than hovering over other features such as CD3, Color Histogram, HMP, HOG, ORB, etc, we focused on caption only, seeking a higher maturity level in the modelling development process.

a. Training set : This is a regression problem as we have two continuous response variables with values ranging from 0.0 to 1.0 and one-hot encoded predictors. In detail, we have a 2d array matrix with 5191 features as predictors and each of which represents words used in the whole caption while each row represents the one of 6000 unique video clips. As for response variables, we have 2 items: 6000 short-term memorability scores and 6000 long-term memorability scores that indicate the impression of each video clip. We only use 80% of this original training set.

b. Given Model : As we have a huge number of features - 5191 -, Neural Network seems a viable choice. We start with the fully connected perceptron network structure with three-layer model as a base. In this model, we feed 5191 inputs into the first 10 neurons that using 'Relu' as an activation function and L2 regularization(Ridge) with small penalty of 0.001. This is followed by another identical layer and ends up with 2 neurons and the 'Sigmoid' function in the output layer. The back propagation comes with dropout rate 0.5 and 0.6.

c. Testing set : We only take 20% of original training set and utilize them as a testing set. After training this model, we make a prediction of short/long term memorability scores in order to run pre-defined 'Get_score()' function at the end which gives us Spearman's Correlation score by comparing our prediction with a testing set.

d. Result : The learning curve of this model illustrates that as we increase the number of epoch, the model performance is growing and both training and validation accuracy converge to certain values described as follows:

- accuracy: 0.7035
- error: 0.0137

Since the error is low and the convergence is confirmed, we cannot say that the model is overfitting. As for Spearman's Correlation, it says **0.24** for short-term and **0.14** for long-term. Both indicate our predicted memorability scores are unfortunately not highly related to the real memorability scores.

Our task is to improve this model so that it renders Spearman's Correlation score as high as possible.

2 RELATED WORK

Many recent works already investigated the use of various low, high level visual features for memorability prediction but they say that captions always outperform[1],[2],[3],[4]. Rohit Gupta's Conduent labs team used ElasticNet (L1 and L2 Regularized Linear Regression)[1]. Aaron Weiss' College of New Jersey team used Early Fusion NN. This tried to address the vanishing gradient problem by pre-training layers[2]. Wensheng Sun's Michigan Technological University Team advocates the scene semantics derived from the captions using NLP techniques and RNN[3]. Ritwick Chaudhry's Adobe research team used single layered LSTM of hidden dimension of 100[4]. In our work, we decided to delve more into Vanishing Gradient problem.

3 OPTIMIZATION APPROACHES

3.0 Pre-Exploration:

In this stage, we found that the MLP model having a more articulation in the first hidden layer including overfitting prevention shows better results. It seems that earlier layers in the network are more important as they are responsible to learn and detect the patterns and become actually the building blocks of the network. If they offer wrong information, then the next layers and the complete network cannot perform

properly. Our pre-exploration to reach this point took two steps – hyperparameter tuning and K-fold Cross Validation.

a. Hyperparameter Tuning: We use Grid Search to evaluate different configurations for our MLP model. This investigation specifically focused on answering following 5 questions:

(a) What dropout to use?

(b) Which Neuron Activation Function to use?

(c) Which Learning rate Optimization Algorithm to use?

(d) Which Network Weight Initialization to use?

(e) What Batch-Size & NO.of Epoch to use?

We discovered when we drop random 30% of input nodes on purpose (to make a thinner model) and weight_constraint "3", our model gives us Spearman's correlation **0.338** for short-term and **0.163** for long-term. This is followed by Spearman's correlation **0.421** for short-term and **0.167** for long-term when we use "selu" as an activation function in the hidden layer and "adadelta" as a learning rate algorithm. Finally, "he_normal" as a Network Weight Initializer with 100 epochs yields Spearman's correlation **0.446** for short-term and **0.200** for long-term, which seems the best scores we can get from using Grid Search. When plotting a learning curve, both the training and validation loss decrease in an exponential fashion as the number of epochs was increased, suggesting that the model gains a high degree of accuracy as the epoch_size (or number of forward and backward passes) is increased.

b. K-Fold Cross-Validation: Prior to develop our model further, we would like to ensure that our model is fitting on homogeneous data so that we can prevent our prediction results from potential skewed distribution or bias. We used "cross_val_score()" function to run a 5-fold cross-validation on our neural network. These 5 fitting give us MSE: 0.0128, 0.0137, 0.0142, 0.0133, 0.0141, and it seems they are not different from each other.

3.1 Exploration 01. Greedy Layer-Wise PreTraining

There are many issues that can be addressed for a better deep learning model. The first one would be a topology of the network such as layers' size or neurons in each layer. We wanted to make sure if our model is well-structured. However, in this regards, the biggest enemy in general, might be "Vanishing Gradient". As the number of hidden layers is increased, the amount of error information propagated back to earlier layers is significantly reduced. In this case, weights in hidden layers close to the output layer are updated properly, whereas weights in hidden layers close to the input layer are updated wrong or not at all in response to errors calculated on the training dataset [5]. We wanted to tackle this issue, using Greedy Layer-wise Pretraining for a starter. This method can help this situation by iteratively deepen our model and evaluate their performances. We would expect the addition of layers to improve the performance of the model on both training set and testing set.

As a result, however, it turned out the number of layers is not decisive factor that impacts the performance of our model as the line plot of added layers against accuracy almost leveled out. So we conclude that deeper model based on simple MLP cannot be a solution for our task.

3.2 Exploration 02. Greedy Layer-Wise Pretraining with Autoencoder as a Regressor

An autoencoder, through the iterative process of training with different feature, learns the features of given data with a bottleneck at the centre and reconstructs the desired data from these learned features. In short, an autoencoder does two tasks - encoding features and then decoding them. It is usually used to de-noise the features. In our work, we tried this autoencoder in the hope that it refines our predictor matrix and features, thus yields better results. We made two approaches - fitting the autoencoder regressor with adding more layers and fitting the autoencoder regressor solely.

The result shows that adding layers does not work here as well. And for some reason, the autoencoder itself does not shine in our task as it produces lower Spearman's correlation scores – **0.441**, **0.182** -, particularly in the Long-term, than our simple MLP model.

3.3 Exploration 03. Learning Rate Schedules with Stochastic Gradient Descent

As a classical algorithm, SGD can be useful in improving the model performance and speeding up training by controlling learning rate. By using a combination of regularization term, dropout layers and learning rate scheduling, we want to directly tackle neurons' coefficient weight matrix and see how much we can improve the model performance. When it comes to the learning rate schedules, we can try two methods – Time-Based and Drop-Based. In Time-Based method, we decrease the learning rate gradually based on the epoch while in Drop-Based method, we decrease the learning rate, using punctuated large drops at specific epochs[5].

$$LearningRate \times \frac{1}{1 + decay \times epoch} \quad (1)$$

$$InitialLearningRate \times DropRate^{floor(\frac{1+Epoch}{EpochDrop})} \quad (2)$$

Figure 1. Learning Rate formula: Time Based (1) and Drop Based (2)

As a result of Time based Scheduling, we obtained Spearman's Correlation score **0.438** for short-term and **0.187** for long-term. From Drop based Scheduling, we obtained **0.433** for short-term and **0.183** for long-term. These scores are very close to what the previous autoencoder approach produced. Down below is the comparison in scores of each approach we made so far.

Approaches	Spearman
Initial Model (MLP with 3 Layers)	0.24(ST), 0.14(LT)
Hyperparameter tuning with GridSearch (MLP)	0.44(ST), 0.20(LT)
Autoencoder Regressor with AdaDelta Optimizer	0.44(ST), 0.18(LT)
LearningRate scheduling with Stochastic Gradient Descent Optimizer	0.43(ST), 0.18(LT)

Table 1: Approaches and Spearman's Scores

4 CONCLUSIONS

Constructing a perfect deep learning network is a very difficult task. Of course there are lots of attempts and achievements regarding the advanced MLP development out there, but in our work, we started from exploring the basic components of MLP(neuron, dropout, optimizer, activate function, learning rate, network topology, etc) and tried to understand their relationships, then moved to some advanced techniques such as using an autoencoder or a learning rate scheduling. Unfortunately, some reasonable scores obtained by basic hyper parameter tuning were not significantly improved by the advanced techniques we tried. In addition, although we reached the baseline performance using basic optimization techniques, it is still unclear which method would improve the long-term scores as all of them yielded scores less than 0.2.

REFERENCES

[1] Rohit Gupta (Conduent Lab India) et al, ***Linear Models for Video Memorability Prediction Using Visualand Semantic Features***, MediaEval'18, 29-31 October 2018, Sophia Antipolis, France

[2] Aaron Weiss (The College of New Jersey) et al, ***Predicting Memorability via Early Fusion Deep Neural Network***, MediaEval'18, 29-31 October 2018, Sophia Antipolis, France

[3] Wensheng Sun (Michigan Technological University) et al, ***Video Memorability Prediction with Recurrent Neural Networks and Video Titles***, MediaEval'18, 29-31 October 2018, Sophia Antipolis, France

[4] Ritwick Chaudhry (Adobe Research Team) et al, ***ViMemNet: Predicting Video Memorability***, MediaEval'18, 29-31 October 2018, Sophia Antipolis, France

[5] Jason Brownlee, **What is deep learning**, Machine Learning, mastery, 2016, p 107