ADVANCED MACHINE LEARNING PROJECT

An analysis of IMDB and MNIST datasets using neural networks

Team Omega

Haizea Prieto de la Vega 201892873	haizea.prieto-de-la-vega.2018@uni.strath.ac.uk
Christian Hristov 201873047	christian.hristov.2018@uni.strath.ac.uk
Francesco Giuseppe Mascia 201884365	francesco.mascia.2018@uni.strah.ac.uk

Introduction

This project aims to select the best neural architecture that is appropriate to the two datasets at hand. The datasets used are IMDB and MNIST in sections 1 and 2 respectively. For each dataset an ideal architecture is proposed, along with empirical evidence to support this choice. The goal is to explore and find the architecture and configuration that gives the best accuracy.

1. IMDB Dataset

The IMDB dataset is used with the aim to classify reviews as positive or negative based on the text of the review. The dataset consists of 50,000 polarised movie reviews from the Internet Movie Database (IMDB) and it is split into 25,000 reviews for training and 25,000 reviews for testing (Tensorflow.org, 2019). The dataset comes also pre-packaged with Tensorflow meaning that the reviews have already been processed and transformed into sequences of integers representing the different words, making it easier to build our network.

The two best networks that we built are explained in the next section. Combination 1 is the best one and achieves training accuracy of 93.15% and testing accuracy of 89.02%. Combination 2 is the second-best model and achieves training accuracy of 92.1% testing accuracy of 86.06%. We also have a network which achieves an accuracy of over 88% but we have decided to select Combination 2 as the second-best network because it is the fastest network of all. We also provide a summary of four additional networks we built to attempt to increase classification accuracy.

1.1 Combination 1 – description of the network

In the case of sequential data as in the IMDB dataset, a recurrent neural network (RNN) is considered as the most adequate method to pursue a text classification task (Géron, 2017).

The network has been designed using Nielsen (2015) as a main reference, which provided a precious guideline to build the Keras code. After importing the IMDB dataset, some minor pre-processing needs to be done. We define the arguments `max_features=10000` and `maxlen=256` which tell the model to keep the 10,000 most used words and to consider reviews not longer than 256 words. The first important step is to convert the sequences of integers into tensors. A tensor can be interpreted as a container for data which gives the structure to this data. By using the `tf.keras.preprocessing` function we can pad our sequences of integers into a 2D Numpy array with the length of the `maxlen` argument. This tensor is then used for the input layer of our network which is of Embedding type with input `max_features` and an output of dimension 32. The architecture of the network is shown in Figure 1.1 below.

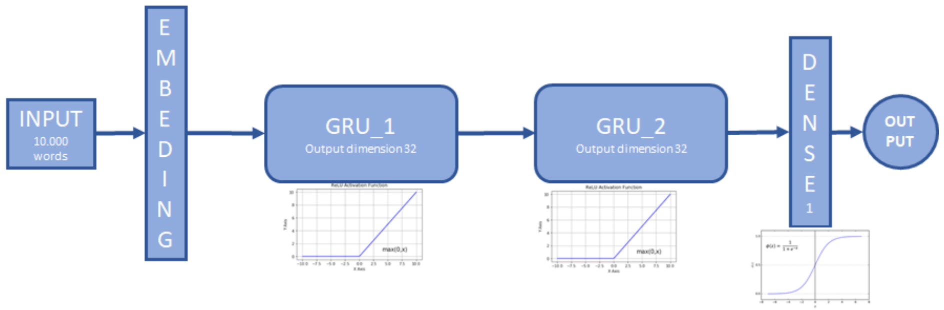


Figure 1.1 IMDB Combination 1 architecture.

The Embedding layer has the function of a "dictionary" that that takes an input of integers and processes them to return them to a corresponding integer vector. Chollet (2018) defines it as a layer that works as a dictionary lookup which takes words from a word index, processes them and associates them to the corresponding word vector.

The next step is adding two gated recurrent units (GRU) layers to process our integer sequences. The (GRU) layer type is added because, conversely to the simpler RNN layers, is able to retain information in a time *t* about inputs in steps before and avoids the problem of the vanishing gradient. A Long Short-Term Memory layer could have also been used but the

GRU layer type is faster to run. The only drawback is that GRU has slightly more representational power than LSTM, but the trade-off between speed and power is a common machine learning occurrence. The recurrent layers always have two arguments for the dropout, `dropout` for input units and `recurrent_dropout` for the recurrent units and have the function of limiting the model from overfitting. The dropouts are kept at a constant rate of 0.20 in both layers so as to propagate the learning error of the network properly through time. If the dropouts were set at a random value that would disrupt the learning process of the network (Chollet, p216). The layers are activated by the `relu` function which has the main task of cancelling out any negative values that are inputted.

Following the GRU layers, we connect a last Dense layer with `sigmoid` activation and 1 hidden unit. The layer is defined as dense because there are many inputs flowing through the network which, namely, makes it dense. The Dense layer produces 1 output because it is activated by a `sigmoid` function which has the main task of outputting a single probability value between 0-1 indicating the likelihood of the network in predicting the sentiment of the review.

Once we have the sentiment analysis network in place, we have to set an optimiser and a loss function. Binary cross-entropy is used as loss function because we are dealing with a binary classification of the output, i.e. positive and negative sentiment. Furthermore, cross-entropy is also good when the type of activation function is to give a probability as output, as it is the `sigmoid` in our case (Chollet, p72,73). The optimizer used is `rmsprop` and has the function of specifying the way in which the loss gradient is used to update the parameters (Chollet, p29). The learning rate of the optimizer is kept low at 0.001, the reason is that a low learning rate makes the training more precise but because it converges slowly towards the minimum of the function it takes time to process. With a high learning rate, we might never get to a minimum and representation would be harmed. Based on these parameters setting our network will give us how accurately we can predict the sentiment of a movie review.

This configuration was run on 263,513 parameters for 15 epochs and a batch size of 64 achieving our best result of 93.15% train accuracy and 89.02% test accuracy. This means that our network trained on the 25,000 samples of the training set and learned to predict the sentiment of reviews never seen before in the test set achieving an accuracy of 89.02%.

1.2 Further Approaches

Combination 2 is similar to Combination 1 but has a different neural network architecture. In fact, we tried to classify sentiment using a convolutional neural network (CNN) despite it being more appropriate for image classification. Naturally, a recurrent neural network (RNN) is more appropriate for text data, but the idea is that text can be treated as a one-dimensional image to capture association between neighbouring words by using a CNN with one-dimension, a Text CNN (Zhang et al, 2018).

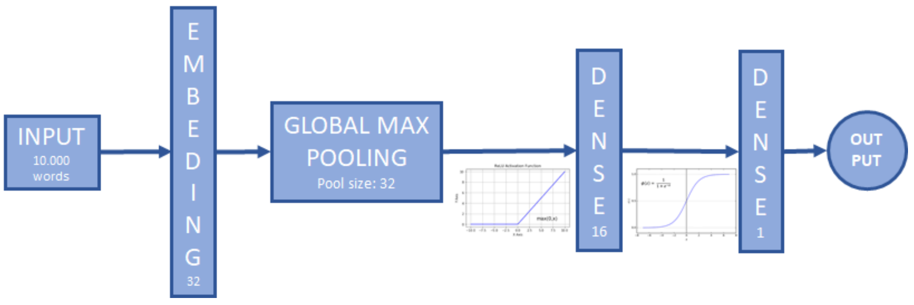


Figure 1.2 IMDB Combination 2 architecture.

Similarly to what explained in the previous section, we define the arguments `max_features=10000` and `maxlen=256` and we convert the sequences of integers into tensors. The input layer is again of Embedding type with input `max_features` and output with dimension 32 which feeds into a Max Pooling layer. This type of layer is commonly used for image feature recognition and has the task of extracting a window of input from the input feature in order to reduce the number of feature coefficients to process, thus making the network faster. The main characteristic is that for the case of text analysis a GlobalMaxPooling layer is more appropriate. This is just a normal max pooling layer which gets as pool size the size of the input, 32 in our case (StackExchange.com, 2017).

After the Max Pooling, we connect a Dense layer with `relu` activation which produces 16 output units. This is then linked to a Dense layer with `sigmoid` activation and 1 hidden unit similarly to the model in Combination 1.

The optimizer, loss function and metrics are the same as those in Combination 1: rmsprop, binary cross-entropy and accuracy respectively.

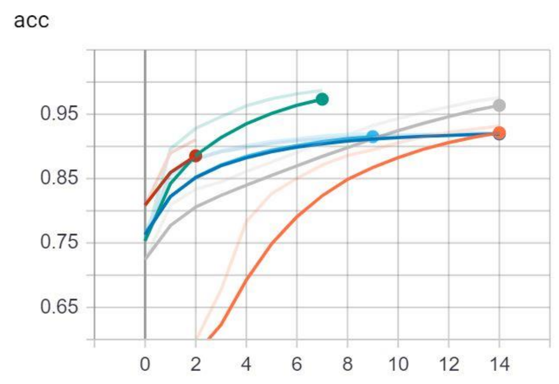
This configuration was run on 320,545 parameters for 15 epochs and a batch size of 32 achieving our second-best result of 92.1% training accuracy and 86.06% testing accuracy. This means that our network trained on the 25,000 samples of the training set and learned to predict the sentiment of reviews never seen before in the test set achieving an accuracy of 86.06%.

Table 1.1 below shows our two best combinations, four more combinations we attempted are also reported. Example c was obtained using Combination 1 but with a LSTM layer instead of the first GRU. Example d has the same architecture of Combination 2 and only the hyperparameters were changed. Example e and f derived from Combination 2 as well with an additional LSTM layer at the beginning and a 0.2 dropout between the two Dense layers. Despite having a good accuracy Example e showed to be the slowest of all, while Example f scored the lowest result.

Example	Combination	Hyperparameters	Training Accuracy	Testing Accuracy	Colour
a	**1**	**LR: 0.001, Epochs: 15, Batches:64**	**93.15%**	**89.02%**	**Orange**
b	2	LR: 0.001, Epochs: 15, Batches:32	92.1%	86.06%	Blue
c	-	LR: 0.001, Epochs: 3, Batches:32	91.04%	87.89%	Red
d	-	LR: 0.001, Epochs: 10, Batches:32	92.00%	86.38%	Green
e	-	LR: 0.001, Epochs: 15, Batches:64	98.72%	87.55%	Azure
f	-	LR: 0.001, Epochs: 15, Batches:32	97.52%	75.54%	Grey

Table 1.1 Six IMDB network combinations compared.

Our best model achieved a validation accuracy just over 89% which is considerably good in the case of text classification. Combination 1 and 2 learning was graphically represented using TensorBoard, along with Examples c, d, e and f.



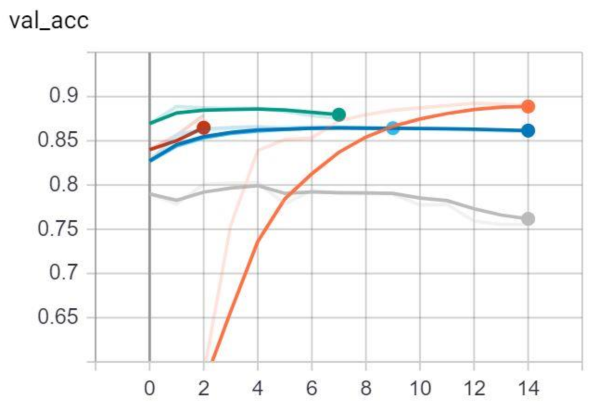


Figure 1.3 TensorBoard IMDB learning curves. Training on the left, validation on the right.

In Figure 1.3 we can observe the learning curves of the six examples. Our best model does not perform well in the early epochs, but it improves progressively until reaching the peak at 93.15% and 89.02% for training and test set respectively.

With more advanced models it is possible to achieve accuracies of up to 95% by using techniques such as LSTM for Region Embeddings, GPU kernels and semi-supervised text categorization. (Ruder, 2018).

2. MNIST Dataset

In this section, we will be using the MNIST dataset. This dataset consists of a large database of handwritten digits constructed from two datasets of the US National Institute of Standards and Technology (NIST). The training set consists of handwritten digits from 250 different people, 50% high school students, and 50% employees from the Census Bureau.

The dataset contains 70 thousand instances. Each feature vector (row in the feature matrix) consists of 784 pixels, unrolled from the original 28x28 pixels images. The target variable consists of labels from 0 to 9 representing the true number the handwritten digit means (Tensorflow.org, 2019).

The aim of this model is to correctly classify the pictures into the corresponding number. We have used a Convolutional Neural Network architecture, being the most adequate to achieve high accuracy on image classification (Géron, 2017).

Convolutional neural networks are very similar to multichannel perceptron neural networks, their advantage is that each part of the network is trained to perform a different task, this reduces the number of hidden layers that need to be used, so it can be trained faster. Furthermore, it presents invariance to the translation of the patterns to be identified. They are highly powerful for everything that has to do with image analysis because they are able to detect simple features such as edge detection, lines, etc. and composes in more complex features to detect what you are looking for (Nielsen, 2015).

We applied the CNN approach using Keras, in order to get the highest possible accuracy, we widely explored different spaces and combinations, playing with the network architecture and the hyperparameters as well. A set of six combinations was attempted to enlarge our knowledge about the topic and build a baseline that allowed the team to suggest the best model. The main model and the other approaches will be explained into more detail in the next sections.

2.1 Combination 1 – description of the network

The Combination 1 architecture was inspired by the network proposed in Gupta (2019). It consists of 4 different layers, the first and second one being convolutions with max pooling that convolutes de 28x28 sized image into a 3x3 one with 32 or 64 filters and into 2x2 with the max pooling part. Between the convolutions and the fully connected layers, there is a flattening part that transforms those filters into one. From one convolution to the other there is always a drop out procedure which speeds up the learning rate. The activation functions are respectively `relu` which avoids gradients to vanish and `softmax`, mainly used at the end of the network to give a probability number between 0 and 1.

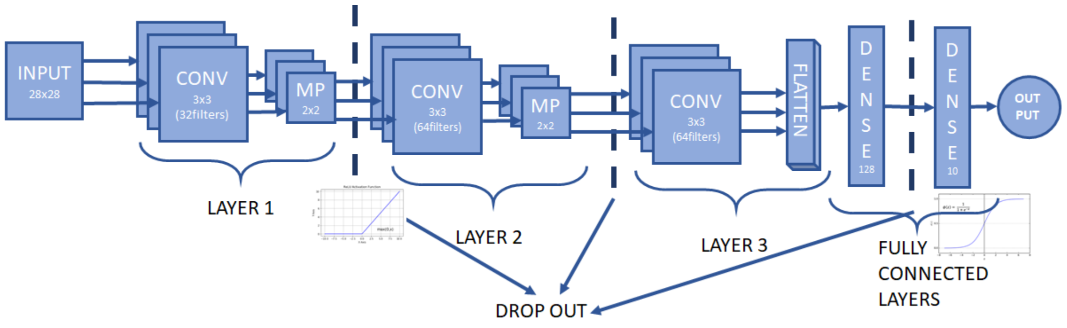


Figure 2.1 MNIST Combination 1 architecture.

This architecture achieved an accuracy of 99.23% on the validation set using a 0.001 learning rate, 20 iterations and 128 batches.

2.2 Further Approaches

The first attempt to improve the network was made by simplifying Combination 1, taking out the dropouts and only using two convolutional layers without max pooling. The graphical representation of it is displayed in Figure 2.2.

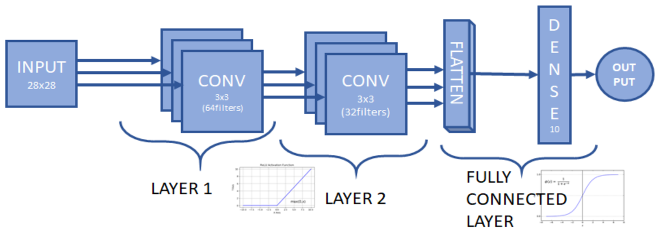


Figure 2.2 MNIST Combination 2 architecture.

This approach achieved a close result with an accuracy only 0.3% lower than Combination 1. An additional architecture was explored characterized by adding a dropout layer to Combination 2 with the idea of speeding up the architecture and getting better results. However, it did not work as expected since it performed worse than both Combination 1 and 2. Moreover, the three main models have been tested using different sets of hyperparameters to see if improvements were to be made.

Example	Combination	Hyperparameters	Training Accuracy	Testing Accuracy	Colour
a	**1**	**LR: 0.001, Epochs: 20, Batches:128**	**98.32%**	**99.23%**	**Blue**
b	2	LR: 0.001, Epochs: 5, Batches:64	99.19%	98.9%	Orange
c	3	LR: 0.001, Epochs: 10, Batches:32	98.51%	98.4%	Red
d	1b	LR: 0.001, Epochs: 10, Batches:256	99.74%	98.73%	Azure
e	2b	LR: 0.001, Epochs: 15, Batches:128	99.09%	98.38%	Pink
f	3b	LR: 0.001, Epochs: 3, Batches:256	96.18%	98.18%	Green

Table 2.1 Six MNIST network combinations compared.

As in section **1.2**, the six combinations were output to TensorBoard in order to be compared in an easy and intuitive way, a graphical representation of the learning curves is provided in Figure 2.3.

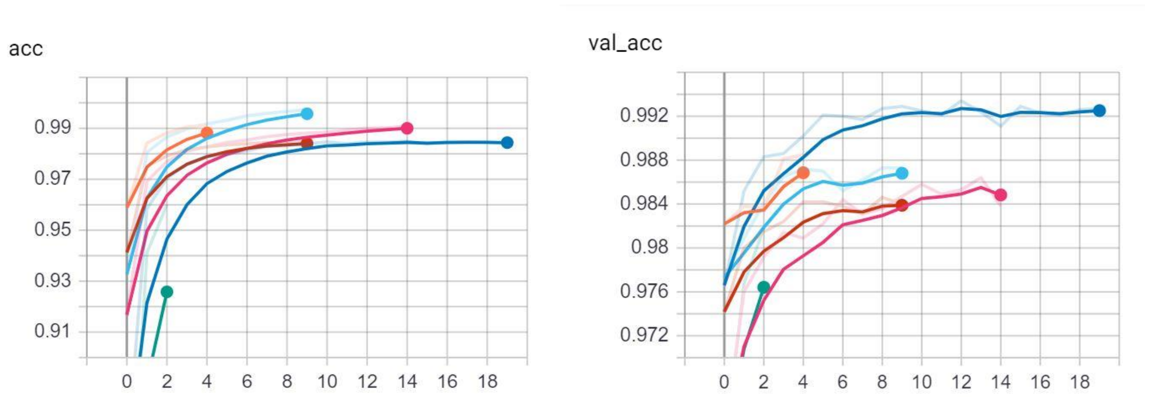


Figure 2.3 TensorBoard MNIST learning curves. Training on the left, validation on the right.

On the left, Example b (orange) and d (azure) are the ones the highest train accuracy, with a fairly low number of iterations, 5 and 10 respectively. However, when fitted on the validation data, is the Combination 1 that performs better.

In conclusion, although the highest accuracy has been obtained with Combination 1 (blue), it requires a sequence of various complex layers which can increase the cost of the final architecture as well as the time to train the whole network. On the other hand, Combination 2 is much simpler and can achieve a very similar accuracy in very little time.

For future improvement a variety of methods can be used to obtain better and better accuracies, up to 99.97%. According to Deotte (2019) such a result can be reached using 15 convolutions and getting additional data due to data transformation techniques such as rotating the original images.

References

1. Chollet, F. (2018). Deep Learning with Python. Shelter Island: Manning.
2. Deotte, C. (2019). Kaggle.com. 25 Million Images! [0.99757] MNIST | Kaggle. [online] Available at: https://www.kaggle.com/cdeotte/25-million-images-0-99757-mnist [Accessed 8 Apr. 2019].
3. Géron, A. (2017). Hands-On Machine Learning with Scikit-Learn & Tensorflow. 5th ed. Sebastopol, US. O'Reilly Media, Inc.
4. Gupta, R. (2019). Kaggle.com. Digit Classification with Keras (>99% Accuracy) | Kaggle. [online] Available at: https://www.kaggle.com/raahat98/digit-classification-with-keras-99-accuracy [Accessed 8 Apr. 2019].
5. Nielsen, M. (2015). Neural Networks and Deep Learning. Determination Press, pp.167-200.
6. Ruder, S. (2018). Sentiment analysis. [online] Available: http://nlpprogress.com/english/sentiment_analysis.hlintml [Accessed 8 Apr. 2019].
7. StackExchange.com (2017). What is global max pooling layer. [online] Available at: https://stats.stackexchange.com/questions/257321/what-is-global-max-pooling-layer-and-what-is-its-advantage-over-maxpooling-layer. [Accessed 8 Apr. 2019]
8. Tensorflow.org (2019). Tensorflow.org/datasets/datasets. [online] Available at: https://www.tensorflow.org/datasets/datasets [Accessed 9 Apr. 2019]
9. Zhang, A. et al. (2018). Text Sentiment Classification: Using Convolutional Neural Networks (textCNN). In: Dive into Deep Learning.