MLP Coursework 1: Activation Functions

s1764997

Abstract

This paper evaluates performance of relatively shallow fully connected neural networks on EMNIST dataset. It establishes a baseline model with a certain architecture and parameters and then investigates different activation functions and their effect on the model's performance. The activation functions used and tested in this paper are ReLU, Leaky ReLU, Parametric ReLU, Randomized ReLU and ELU. It further evaluates whether non-linear activation functions are indeed needed in neural networks. During the whole course of the paper, it introduces the covered topics theoretically and also provides experimental results.

1. Introduction

EMNIST dataset, investigated in this paper, is a large dataset of hand-written digits and letters. There are 47 possible classes in this dataset - 10 for digits and 37 for letters. The reason for 37 letters classes is that it covers both lower- and upper-case letters of English alphabet. However, for certain letters, it is really challenging to distinguish between their lower- and upper-case form and are hence put into one class instead. Each digit/letter is represented by 28x28 pixels. Training dataset has size 100000 x 784, validation and testing set have each size 15800 x 784.

The first task is to come up with a reasonable baseline that is using only ReLU activation function and will have 1 to 3 hidden layers with 32, 64 or 128 units per hidden layer. Different learning rates have been explored to find the right baseline with sufficient performance.

The second part is dealing with using different activation functions mentioned in the abstract. This part only allows tuning of hyper-parameters of the activation functions rather than of the whole network. These were tested and model with the best performance on validation set was used to record its performance on the testing set to get an estimate for the general accuracy of our model on future, unseen data.

The last part investigates whether and why non-linear activation functions are needed in neural networks. It gives a theoretical argument justified by experimental results.

2. Baseline

In order to choose a good baseline model, several neural network architectures were tested. Fully connected neural networks were used, with varying number of hidden layers (1, 2, 3) and number of units per hidden layer (32, 64, 128). Note that for a chosen number of units per hidden layer, this number is the same for each hidden layer in the neural network that is tested.

Stochastic Gradient Descent optimiser was used for the course of all experiments. This is because it seems to be producing better generalisation results than ADAM optimizer and overall produces very accurate results [1]. All experiments in this paper are carried out using softmax cross-entropy loss function. This ensures that the outputs of the final layer will sum up to one which is desirable for classification tasks. ReLU activation function is used for all experiments in this section.

In order to choose appropriate learning rate, the following ones were tested: 0.0001, 0.001, 0.01, 0.1, 1. Due to random initialisation of the weights, our experimental results would not be reproducible. Hence, every result is the mean of 3 experiments. The following tables present test set classification accuracies. Please note that # HL refers to the number of hidden layers in the network. This notation is used for practical purposes throughout the whole paper.

# HL	32 UNITS	64 UNITS	128 UNITS
1	78.66%± 0.20%	80.73%± 0.40%	81.36%± 0.15%
2	79.15%± 0.28%	80.70%± 0.21%	81.10%± 0.28%
3	77.88%± 0.23%	80.47%± 0.40%	81.34%± 0.15%

Table 1. Test set classification accuracies. Learning rate was 0.1.

# HL	32 UNITS	64 UNITS	128 UNITS
1	78.29%± 0.34%	81.41%± 0.22%	82.98%± 0.07%
2	78.75%± 0.22%	81.89%± 0.17%	83.42%± 0.15%
3	79.11%± 0.15%	81.36%± 0.14%	83.17%± 0.02%

Table 2. Test set classification accuracies. Learning rate was 0.01.

# HL	32 UNITS	64 UNITS	128 UNITS
1	67.94%± 0.34%	70.39%± 0.40%	72.51%± 0.08%
2	68.85%± 1.23%	73.12%± 0.40%	75.34%± 0.11%
3	68.28%± 0.23%	74.03%± 0.03%	76.81%± 0.09%

Table 3. Test set classification accuracies. Learning rate was 0.001.

To keep this report concise and to the point, tables present results only for learning rates where the highest classification accuracies were obtained. This is because we are trying to optimise our baseline model, and so our learning rate needs to be chosen wisely.

When the learning rate was set to 0.0001, all of the classification scores obtained were in general lower than

results in tables presented above. This is most likely due to the models hitting a local optimum where the weights get stuck due to the learning rate being too small. Another possible explanation could be that the models did not converge. However, considering that the network is relatively shallow, this should not happen when being trained for 100 epochs. I am therefore more inclined towards the former explanation. When the learning rate was set to 1, the model did not converge but ended up diverging instead and unusable results were produced. When the learning rate is too high, which was most likely the case, the optimal points where the loss function is minimised are jumped over and the model is not stable.
The experiments indicate that the more complicated our model is, that is, the more hidden units per layer and more hidden layers are used, the better the results. This is logical as a deeper neural network with more units can better grasp inherent features of objects of different classes from the images which in turn results in obtaining higher classification scores. Indeed, it seems that very impressive results can be obtained without using CNNs as the following paper proves [2]. The authors obtained best results when using more hidden layers, with decreasing number of units throughout the layers. However, networks they tested are a lot more robust and would require incomparable amount of computing power and time to train.
Some of our experiments indicate that a more complex model is not always better. When the learning rate was 0.1 and 0.01, the number of layers did not have a clear impact on the test set classification accuracies obtained. It could be argued that our experiments did not have a sufficient range of hidden layers and units per hidden layer to show this relation and that a lot of noise is present in the experiments due to the random weight initialisation.
Note that the tables present test set classification accuracies, where the highest score was obtained with 2 hidden layers and 128 units per hidden layer.

In order to pick the baseline, validation set classification scores are used. This is to ensure we would not over-fit on the testing set. These were different in comparison to test set results. The best validation set classification scores obtained were 84.06% ± 0.09% and 84.11% ± 0.02%. Both were obtained when using with 3 hidden layers and 128 units per hidden layer. The former score was obtained using 0.1 learning rate, the latter one using 0.01 learning rate. I decided to choose 0.01 learning rate for our baseline. It produces the best validation set classification accuracy. My decision was also effected by literature I came across [3] where this value was suggested for similar experiments. Also, 0.1 is a rather large learning rate and could result in more unstable results. After all, the default value in coursework specification is 0.001 and thus value 0.1 seemed too high.

3. Activation Function Comparisons

The baseline used only ReLU activation function. This part assesses performance of the following activation functions: Leaky ReLU (LReLU), Random ReLU (RReLU), Parametric ReLU (PReLU) and ELU.

$$f(x) = \begin{cases} \alpha x & x \leq 0 \\ x & x > 0 \end{cases} \qquad f'(x) = \begin{cases} \alpha & x \leq 0 \\ 1 & x > 0 \end{cases}$$

LReLU (formula above) introduces a new parameter α, due to which the gradient $f'(x)$ obtains value α when the input $x <= 0$ instead of 0, which was the case for ReLU. Producing zero gradient is a problem because during the back-propagation, due to this zero gradient, the weights of the neural net are not updated and so the net is not learning. This gives LReLU a clear advantage over classical ReLU activation. Note that α value is the same for every unit in the neural network, is chosen before the learning process and stays constant afterwards.

RReLU has the same formula as LReLU and on the outside looks exactly the same. However, the important distinction is that for RReLU α is not a single scalar, but a vector of uniformly drawn scalars. Each unit in the network has its own α parameter. The α vector is drawn at every forward propagation stage.

PReLU, similarly as LReLU, introduces a scalar parameter α and has the same equation. However, in this case α is not being constant during the whole training process, but instead it is being learnt during the back-propagation process. PReLU has a chance to update this value during the training process which gives it an advantage over LReLU.

$$f(x) = \begin{cases} \alpha(e^x - 1) & x < 0 \\ x & x >= 0 \end{cases} \qquad f'(x) = \begin{cases} \alpha e^x & x < 0 \\ 1 & x >= 0 \end{cases}$$

ELU, similar to LReLU, has a single α value for every unit in the network and this value is chosen before the training process and stays constant. However, as can be seen above, the underlying mathematical formula is different for this activation function. ELU tends to converge the loss value faster to zero and produces better results [4]. On the other hand, due to the nature of the function, for large negative x, it saturates to 0.

In this part of the coursework, the models evaluated were trained on 50 epochs. When comparing multiple diagrams of the baseline performance, the model seemed to be well converging already after around 30 epochs. I chose to train on 50 epochs because at this point it produced almost the same results as on 100 epochs in half the time. The following graph shows how performance of our baseline model was changing as a function of the number of epochs and provides visual justification why 50 epochs

are sufficient for future training.

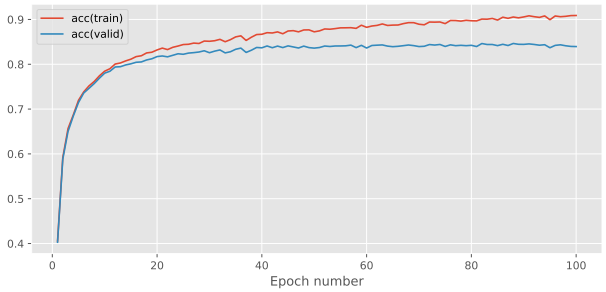

Figure 1. Training and validation classification accuracies obtained by the baseline model as functions of the number of epochs the model was trained on. It indicates that validation accuracy does not seem to change much after around 30 epochs which is why future training is performed on 50 epochs to save time and computational resources.

Following tables present experiments carried out on the validation set. Every result is the mean with its associated standard deviation of 3 experiments. Model architecture, learning rate and optimizer stay the same as for the baseline. Only impact of different activation functions is investigated.

ACTIVATION FUNCTION	CLASSIFICATION ACCURACY
LEAKY RELU	84.16%± 0.19%
RANDOM RELU	83.65%± 0.21%
PARAMETRIC RELU	82.35%± 0.19%
ELU	83.74%± 0.05%

Table 4. Evaluating different activation functions. Default parameters were used for each activation function.

For the default parameters of the activation functions, LReLU obtained the highest validation set classification accuracy, while PReLU obtained the lowest. I considered this initially a bit surprising, however, this could be due to the initial values of the hyper-parameters of the activation functions. Those are explored below.

α VALUE	CLASSIFICATION ACCURACY
0.001	83.92%± 0.17%
0.01	84.16%± 0.19%
0.1	84.03%± 0.01%
0.2	84.02%± 0.21%
0.3	83.69%± 0.07%

Table 5. Evaluating Leaky ReLU activation function with various α values.

α values were explored within the suggested range. The default α value Tensorflow is using for this activation function is 0.2, Keras is using 0.3. Hence, these two values were tested as well. According to our experiments, the best validation score was still obtained when α was set to its default value 0.01.

For RReLU, instead of specifying a single α value, the range where α is uniformly drawn from is specified. This is interesting because if we make this range sufficiently small, such that lower bound is almost the same as the upper bound, RReLU should start to behave like LReLU. This is because all the α values drawn at each forward propagation would be almost the same, approaching a single value that LReLU would be set to. However, our focus was to get the best performance out of this activation function which was when this range was larger, and thus the alpha values varied more. The best validation set classification accuracy was obtained when the lower boundary of the range was set to 0.001 and upper boundary was set to 0.2.

LOWER BOUND	UPPER BOUND	CLASSIFICATION ACCURACY
0.001	0.2	83.91%± 0.05%
0.01	0.2	83.68%± 0.10%
0.001	0.01	83.83%± 0.05%

Table 6. Evaluating RReLU activation function with different lower and upper bounds.

α VALUE	CLASSIFICATION ACCURACY
0.01	NON-APPLICABLE
0.1	NON-APPLICABLE
0.25	83.92%± 0.15%
0.5	81.78%± 0.46%

Table 7. Evaluating PReLU activation function with different initial α values.

PReLU started producing foolish results when the initial α value was set to anything below 0.1. It is interesting that even with α values lower than 0.1, the model was still producing meaningful results up to around 20-25 epochs and then its accuracy suddenly dropped to almost 0 and error exploded. This is shown in the figure below.

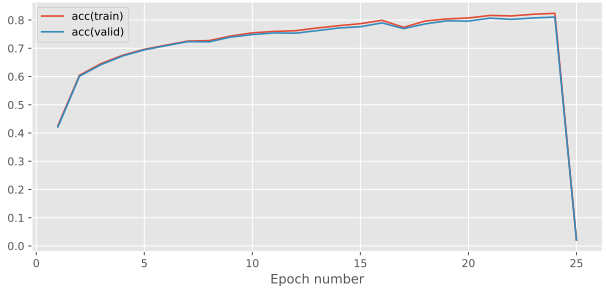

Figure 2. Training and validation classification accuracies obtained when using PReLU activation with initial alpha value set to 0.1. Model produces stable results during the first 24 epochs and then its accuracy suddenly drops to almost zero.

The number of epochs where this "drop" in accuracy occured varied across different experiments as well as different alpha values. The lower the initial value of alpha, the sooner (after fewer epochs) this drop happened. I questioned implementation of this activation function, however, after going through it again, it still seemed correct and stayed unchanged. For now, I could not come up with a the-

oretical explanation for this problem and it requires further investigation.

α VALUE	CLASSIFICATION ACCURACY
0.001	83.92%± 0.15%
0.01	83.74%± 0.05%
0.1	84.22%± 0.13%
0.2	84.34%± 0.02%

Table 8. Evaluating ELU activation function with different α values.

ELU obtained the highest validation set classification accuracy when its α value was set to 0.2. This is also the highest classification score obtained over all the models that were tested on the validation set. LReLU with α set to 0.01 was really close. Let us compare these 2 models again, this time using another metric - the error these models obtained.

ACTIVATION	α VALUE	ACCURACY	ERROR
LRELU	0.01	84.16%± 0.17%	0.4837 ± 0.0050
ELU	0.2	84.34%± 0.13%	0.4842 ± 0.0020

Table 9. LReLU and ELU validation set results.

The comparison is interesting because even though ELU obtains a higher classification accuracy, it also obtains a slightly higher error value. This sounds a bit counter-intuitive. A possible explanation is that ELU performed better overall, however, for some specific samples/classes it performed a lot worse than LReLU. This could account for achieving overall higher error value even though model with ELU's activation got higher classification accuracy.
In general, it is a good practice to choose only one model to be evaluated on the testing set. If we chose more models, it could happen that our test set results would indicate that a model that did worse on the validation model did better on the test set. Choosing then the model that performed best on the testing set would lead to possible over-fitting on the test results which is not desirable. It makes our estimate of the generalization error less accurate.
However, as we are interested in comparing these models, I decided to run both models on the testing set and output the obtained results.
Note that this time both models are trained on 100 epochs and then evaluated on the test set in contrast to the 50 epochs that were used previously. This is because our baseline is trained on this number of epochs. Thus, in order to objectively compare our results to the baseline, this parameter should stay the same.
To summarize, both models use 3 hidden layers with 128 units per hidden layer. Learning rate is set to 0.01 and SGD optimizer is used. The first model is using LReLU activation function with α set to 0.01, the second model is using ELU with α set to 0.2.

Our baseline model obtained on the test set the following classification accuracy: 83.17% ± 0.02%.
Our two best models, that both outperformed the baseline on the validation set, performed worse on the test set as is indicated in the table above. This is quite ironic. However, it does not necessarily mean that our baseline is a better model and should be used instead. It shows that all of these models performed very similarly and are thus likely to do similarly well on future, unseen data too.
In order to estimate the generalisation performance, model with the best performance on validation set should be evaluated on the test set and this result should be used. The model had 3 hidden layers, 128 units per hidden layer, learning rate set to 0.1, used SGD optimizer and was trained on 100 epochs. The model used ELU activation function with α set to 0.2 and obtained 82.88%± 0.23% classification accuracy and 0.5458 ± 0.0096 cross-entropy error on the test set. These are our estimated general metrics we approximately expect to see on the future data.

ACTIVATION	α VALUE	ACCURACY	ERROR
LRELU	0.01	83.14%± 0.01%	0.5582 ± 0.0050
ELU	0.2	82.88%± 0.23%	0.5458 ± 0.0096

Table 10. LReLU and ELU test set results.

4. Linear vs Non-linear

The intuitive reason why we need non-linear activation functions is the following: imagine several linear transformations are applied to the original inputs that were supplied to the neural network. Because all of these transformations are linear, their overall result will be a combination of linear transformations on the inputs and thus this can be replaced by a single linear transformation.
In addition, using only linear transformations, our neural network would not be capable of modelling more complicated, non-linear functions. I therefore expect that using only linear transformations will lead to worse classification scores obtained by the model.
A simple way to compare linear vs non-linear results is to run the same experiments that were run for the baseline, but this time without using the ReLU activation function.

# HL	32 UNITS	64 UNITS	128 UNITS
1	69.47%± 0.17%	69.83%± 0.17%	69.78%± 0.20%
2	69.11%± 0.08%	69.73%± 0.21%	69.88%± 0.03%
3	68.78%± 0.25%	69.46%± 0.12%	69.27%± 0.08%

Table 11. Validation set classification accuracies. No activation function was used. Every experiment was carried out 3 times on 100 epochs and SGD optimizer was used. Learning rate was set to 0.01.

Two interesting things can be observed from the table above. The first one is that results obtained without using any non-linear activation function are in general worse in comparison to results obtained using non-linear activations. The second one is that all of the results are very similar, and it seems that the number of hidden layers and units per hidden layer do not have an impact on the model. This is because applying additional linear transformations does not make our model more sophisticated and can be replaced by

applying a single linear transformation. Thus, even a more complex model would likely still provide similar results to the ones presented above.
It is also expected that similar results would be derived for other learning rates where the model would converge. I ran experiments for the following learning rates [0.0001, 0.001, 0.01, 0.1] and this was indeed the case. Only when the learning rate was 0.0001, the classification accuracies were in general lower. However, this was also the case when selecting the baseline model in part 1. This could be because the neural network hit a local optimum which is often the case for a too small learning rate. Another explanation could be that it did not have sufficient time to converge as the learning rate was too small. However, considering that the network is relatively shallow, 100 epochs should be sufficient for convergence and would therefore say that the former is the true reason.

5. Conclusion

Results of our experiments analysed below share these characteristics: they all consist of three fully connected hidden layers with 128 units per hidden layer. The learning rate was set to 0.01. These parameters came from investigation in the first part of this paper which dealt with finding a good baseline model. SGD optimizer was used as it seems to be generalising very well according to the literature that has been used [1]. As the weights initialization is random, in order to produce more stable results, each reported result is the mean of 3 experiments that were run under the same conditions.
The best performing model obtained validation set classification accuracy of value 84.34%± 0.13% but scored rather worse on the test set, obtaining accuracy 82.88%± 0.23%. This model used ELU activation function with α value set to 0.2. The second-best performing model (with reference to the validation set classification accuracy) obtained the following testing set classification accuracy: 83.14%± 0.01%. This model used LReLU activation function with α value set to 0.01.
These results are a bit disappointing as our baseline outperformed them on the testing set, obtaining classification accuracy of 83.17%± 0.02%, using ReLU activation function.
There does not seem to be a clear theoretical explanation why baseline model with the ReLU activation should be outperforming the other two models on the testing set. The fact that the baseline model did worse on the validation set shows that there is always a level of unpredictability present in machine learning. In addition, because our weights were initialized randomly, there is also a level of randomness present in our experiments. This was mitigated by performing all experiments three times under the same conditions and taking the average of these experimental results.
It is important to note that the baseline's performance on the validation set was 84.11%± 0.02% which is very close to the validation set performances of the other tested models. Thus, all these 3 models performed so similarly, such that using any of them for future experiments would likely lead to similar results on future data.
That being said, in order to choose only one model and estimate generalisation classification accuracy, model with the ELU activation function and α=0.2 would be used. This is because it performed best on the validation set which is the standard procedure for selecting a model for future evaluation.
If larger models (deeper neural nets) could be tested, then it is likely that the different activation functions and tuning of their hyper-parameters could account for bigger differences in the models' performances.

References

[1] Wilson, A., Roelofs, R., Stern, M., Srebro, N., Recht, B. (n.d.). The Marginal Value of Adaptive Gradient Methods in Machine Learning. Retrieved from https://arxiv.org/pdf/1705.08292.pdf

[2] Ciresan Dan Claudiu, Meier, U., Gambardella, L., Schmidhuber, J. (n.d.). Deep Big Simple Neural Nets Excel on Hand- written Digit Recognition. Retrieved from https://arxiv.org/pdf/1003.0358.pdf

[3] Bengio, Y. (n.d.). Practical Recommendations for Gradient-Based Training of Deep Architectures. Retrieved from https://arxiv.org/pdf/1206.5533.pdf

[4] Clevert, D.-A., Unterthiner, T., Hochreiter, S. (n.d.). FAST AND ACCURATE DEEP NETWORK LEARNING BY EXPONENTIAL LINEAR UNITS (ELUS). Retrieved from https://arxiv.org/pdf/1511.07289.pdf

[5] Xu, B., Wang, N., Chen, T., Li, M. (n.d.). Empirical Evaluation of Rectified Activations in Convolution Network. Retrieved from https://arxiv.org/pdf/1505.00853.pdf

[6] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Retrieved from https://arxiv.org/pdf/1502.01852.pdf

[7] https://datascience.stackexchange.com/questions/54258/using-leakyrelu-as-activation-function-in-cnn-and-best-alpha-for-it