AIML425 ASSIGNMENT 2

Benjamin McEvoy

300579954

1. INTRODUCTION

This document revises the concepts of objective functions, regularisation and Stochastic Gradient Descent (**SGD**) applied within deep learning networks to classify the MNIST database of rotated images in a Multi-Class Neural Network. To access the code-base this, repository or this google drive link: https://colab.research.google.com/drive/1Tlb_phSrlLsVHJTgHJvw9dytN8L2YVDd

2. THEORY

This section is to outline the background knowledge, required for this paper which includes the descriptions and working of objective functions, regularisation and Stochastic Gradient Descent.

2.1. Objective Function

An objective function is a mathematical function that measures how well a model's predictions match the actual values. Its purpose is to quantify the "error" or "cost" associated with a particular set of model parameters, guiding the optimisation process to improve the model's performance.

2.1.1. Multi-class Classification

In the context of Multi-class classification, an objective function would follow:

Each class label is represented as a binary vector using One-hot encoding. For example, if there are five classes (a, b, c, d, e), an observation belonging to class c would be represented as $[0, 0, 1, 0, 0]$.

The network produces a probability distribution over the classes. For the same five-class example, the output might be $[q(y_a = 1 \mid x), q(y_b = 1 \mid x), q(y_c = 1 \mid x), q(y_d = 1 \mid x), q(y_e = 1 \mid x)]$, where each $q(y_j = 1 \mid x)$ is the predicted probability that the observation belongs to class j.

The objective function used is typically the Categorical Cross-Entropy. It measures the difference between the predicted probability distribution and the true one-hot encoded distribution.

The Cross-Entropy Loss for a single observation is calculated as:

$$L = -\sum_{i=1}^{d} y_i \cdot \log(q(y_i = 1 \mid x))$$

where y_i is the true label (0 or 1) and $q(y_i = 1 \mid x)$ is the predicted probability for class i. For the true class, y_i is 1, and for others, it is 0.

The objective function is minimised during training to adjust the model parameters θ such that the predicted probabilities q are as close as possible to the true labels p. This is done using optimisation algorithms like gradient descent.

2.2. Regularisation

Regularisation involves adding penalty terms that help control the complexity of the model.

2.2.1. Dropout

Dropout is a regularisation technique that is used during model training, adding robustness to the network and reducing the chances of over-fitting. During each training update, the dropout randomly "drops" (deactivates) a certain percentage of connections in a layer, setting the outputs to zero. In example, if the dropout is set to 25%, every update 1/4 of the connections in the layer will be dropped. This introduces prevention from reliance on singular nodes and encourages the generalisation of features. During inference, there is no dropout, and weights are scaled by multiplying them using $1-p$ where $\boldsymbol{p}$ is the dropout probability (i.e. 0.1). The scaling ensures the network predictions are consistent.

3. EXPERIMENTS

This section outlines the setup, configuration, architecture and discussion of the results; providing insight into the utilisation of the objective functions and SGD on the classification of the MNIST Dataset.

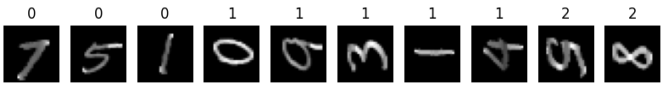
3.1. Setup

3.1.1. Preprocessing

The assignment mentions the usage of the MNIST dataset and no particular version, variant or link to the preferred

dataset. In this experiment, an assumption was made to use the MNIST dataset integrated into TensorFlow Datasets [1] sufficing with the arrangements of brief. This experiment aims to construct a network that effectively classifies images to a rotational class based on the dataset provided, seeking convergence and stability in accuracy maintaining posture to not under-fit or over-fit on the dataset.

To preprocess the data, NumPy is used to perform 90-degree rotations on the datasets. The process involves splitting the data into three parts, rotating each part by the specified axis, and assigning class labels to the rotated arrays. These parts are then concatenated back together to update the dataset with the new values as seen below in Figure 1.



(a) Samples after rotation, and Normalisation.

Fig. 1: Indices of Rotated Images.

Normalization methods provided by TensorFlow Keras are applied to the arrays to facilitate faster convergence and ensure consistency during training. These methods scale features to a range of 0 and ensure a standard deviation of 1.

For validation, the data is split into an 80/20 ratio, with 80% allocated to the training set and 20% to the validation set. This provides a sufficient amount of data for validation while keeping a substantial-sized dataset for training. One-hot encoding is used to label the samples, due to the categorical nature of the dataset (left, right or, un-flipped), converting it into a binary matrix.

3.1.2. Model Architecture

The model architecture of the network was not clearly defined, a simple sequential fully connected network with four layers was used as the model architecture, which featured a flattened layer for the image that converts it into a 1D array, and a dense layer of 128 neurons and relu for the hidden layer, and the objective function dropout layer to ensure data isn't overfit, and another dense layer with 3 neurons using a Softmax activation function.

For compilation of the model, and specified by the brief. The use of SGD was used as the optimiser of the network, with a learning rate of 0.01 as this is the default value for machine learning tasks with the optimiser [1]. Categorical Cross Entropy is also used as the loss function in this task, as it is specifically designed for multi-class classification problems such as the brief outline; calculating the loss between the true labels and predicted possibilities. It is also a loss function that fits the one-hot encoding format that is pre-processed as each element in the dataset is a binary vector with a correct class (left, right, or un-flipped).

3.2. Results

Experimentation was conducted through the previously stated sequential model, fed into this model was a dropout of 10% for regularisation to improve on possible over-fitting of the dataset. In the appendix (Figure 2), we can see that after a few epochs, the data starts to converge to the accuracy of the dataset indicating that the network is not under-fitting nor over-fitting meaning that the structure of the model generalises well.

Before the results above, experiments with dropout regularisation (seen in Appendix Figure 3.) of 0% and 40% caused the model to introduce either under-fitting or over-fitting on the validation, meaning that the data had higher amounts of bias or variance meaning that it would cause poor performance and predictive ability.

The final test accuracy of the dropout regularisation of 10% is: **97.24%**; compared to 0% (97.46%) and 40% (97.33%). These stay around the 97% range, however, this could be due to the test dataset being small. Whether the test accuracy remains the same on a larger test set is unknown, and must be tested further, or the data is simply too similar that it masks the over-fitting or under-fitting if the model.

4. TOOLS

This assignment used the tools of Numpy [2], Matplotlib [3], SKLearn [4] and Tensorflow [1] to carry out experimentation and construction of the neural network. Tensorflow is responsible for the model architecture with layers, models, utilities and optimisers using SGD. Numpy is used for calculations, rotations and normalisation; Scikit Learn for preprocessing, and Matplotlib is used to plot the data in a visual format.

Google Colab [5] is where the code base is hosted and is provided with any available machine during runtime allocation.

Besides the utilisation of these tools [6], I declare that the code and paper provided is my work.

5. CONCLUSION

In conclusion, the results show that the model created in the code-base generalises well according to Figure 1. in the appendix. However, issues remain raised as there is speculation that the training data size may be too small to conclusively decide the effectiveness of the regularisation function. The model itself, on the MNIST dataset shows a converging validation accuracy with the training set accuracy and indicates itself to being a well-generalised design. To gain more understanding, other evaluations could be conducted with precision and recall functions; however the code-base delivers the required deliverables of the assignment.

6. REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[2] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al., "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

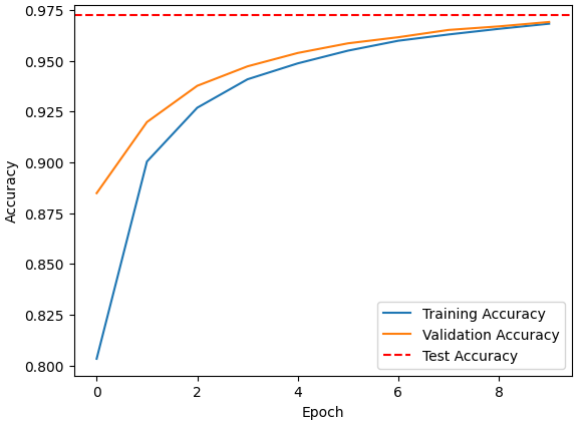
[3] John D Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[4] 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*, vol. 12, pp. 2825–2830, 2011.

[5] Google, "Google colaboratory," Online, 2023, Available at: https://colab.research.google.com.

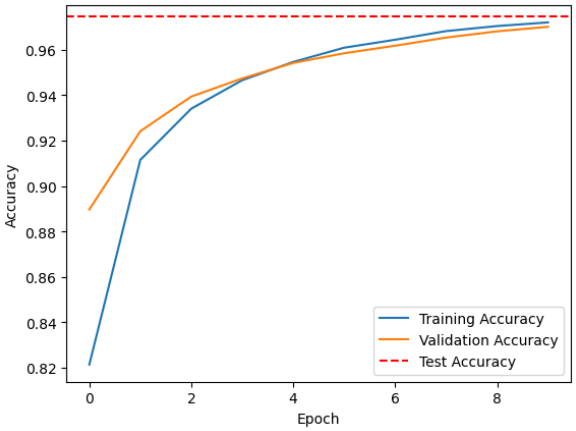
[6] Inc. Grammarly, "Grammarly," https://www.grammarly.com, 2024, Accessed: 2024-08-12.

Appendix

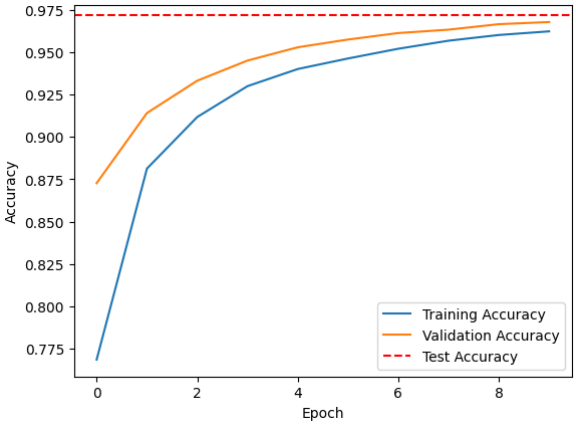


(a) Training and Validation Accuracy - 10%

Fig. 2: Training and Validation Accuracy - 10%



(a) Training and Validation Accuracy - None.



(b) Training and Validation Accuracy - 40%.

Fig. 3: Prior Model Accuracy.