Neural Networks from Scratch

Hunor Laczko and Smriti Joshi

I. INTRODUCTION

Deep learning has proved to be a useful tool capable of modelling complex functions present in real-life problems and providing accurate results. This laboratory report presents the analysis of the performance of basic supervised neural networks. Three types of neural networks are evaluated: Binary classification using single neuron, binary classification using single hidden layer and multiclass classification using single hidden layer. This report has the following structure: Tools used to train and evaluate the network are discussed in Section II, analysis of the data is presented in Section III, the methods and implementations are discussed in Sections IV and V and the results are summarized in section VI. Further, the report is concluded and potential future work is discussed in section VII.

II. PRELIMINARY

This section discusses the cost function and metrics used for training and evaluating the network.

A. Cost function: Cross Entropy

Cross entropy is a popular choice for loss function, it measures the similarity between two vectors or distributions. In a typical multiclass classification application it is used to measure the similarity between the output of a softmax layer (the probability of object belonging to each class) and the one-hot-encoded ground truth label. Mathematically, it is represented by Equation 1.

$$CE = -\sum_{i=1}^{C} (A_i \log B_i) \tag{1}$$

For binary classification, a simplified version called the binary cross entropy is used. Mathematically it can be expressed as follows:

$$BCE = -\sum_{i=1}^{C=2} (A_i \log B_i) = -A_i \log B_i - (1-A_i)\log(1-B_i) \tag{2}$$

In Equation 1. and 2., C is the total number of classes, A is the true label, B is the output probability.

B. Evaluation metric

1) Confusion matrix: Confusion matrix is a table used to present the performance of binary classifiers in the case where true labels are known. This table is shown in Figure 1. False negative, also called Type II error, refers to the case when the training example belongs to positive class but is incorrectly classified as negative class. False positive, called as Type I error, refers to the case when a training example belongs to negative class but is incorrectly labelled as positive class.

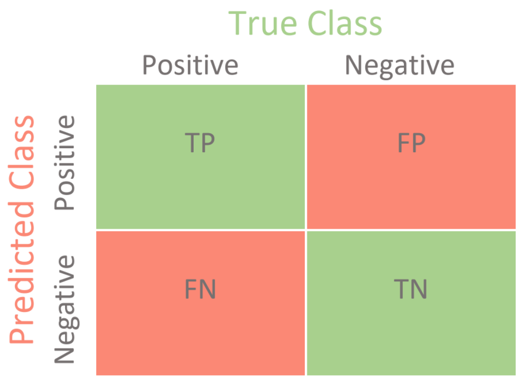


Fig. 1: Confusion Matrix (TP: True positive, FP: False positive, FN: False Negative, TN: True Negative) Source: Towards Data Science

2) Accuracy: Accuracy is the number of instances classified correctly over the total number of instances. Mathematically, it is given by:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

This is not useful for imbalanced datasets because the majority class is usually predicted correctly because of higher number of training examples and leads to higher accuracy.

3) F1-Score: This metric measures the performance of the network from the values of precision and recall in binary classification. This is chosen over accuracy as evaluation metric due to high data imbalance for first two training networks. Precision is the number of correctly identified positive results divided by the number of all positive results, including those not identified correctly. Recall is the number of correctly identified positive results divided by the number of all samples that should have been identified as positive. This can be visualised in Figure 2. Finally, F-1 score is mathematically given by:

$$F1Score = 2\left(\frac{precision * recall}{precision + recall}\right) = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{4}$$

C. Optimizers

1) Stochastic Gradient Descent (SGD): This algorithm is an iterative method to optimise the loss function. It

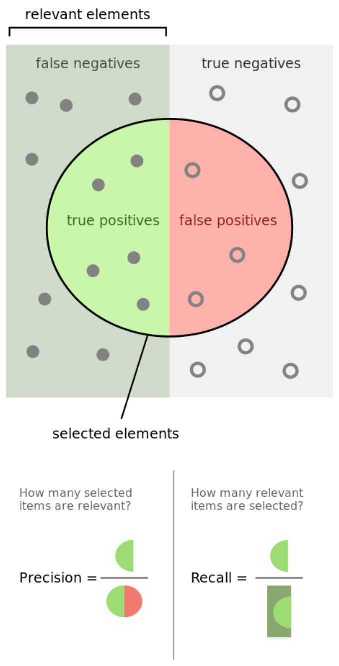


Fig. 2: Visualisation of metrics. Source: Wikipedia

introduces stochastic approach in gradient descent by replacing the gradient with the estimation of gradient from randomly selected batch of data. This decreases the computational burden achieving faster iteration rates.

2) RMSProp: RMSProp [1] is another stochastic technique for mini-batch learning. RMSProp optimises the loss function by a moving average of squared gradients to normalize the gradient. It is also known for handling the problem of vanishing/exploding gradients as this normalization balances the step size (momentum), decreasing the step for large gradients to avoid exploding, and increasing the step for small gradients to avoid vanishing. In practical terms, this can be understood as having learning rate as a variable parameter.

3) Adam: Adam optimiser is also an iterative method and extends on the idea of RMSProp. The learning rate in this case is adapted not only on the basis of mean of gradients but also on the average of uncentered variance. In the paper proposing adam, the optimizer is presented as increasing the speed of training considerably.

D. Activation functions

Activation functions are popular to introduce non linearity in the network to fit complex distributions. In the networks, they are used to determine if the neuron should be fired and to what extent. Two kinds of activation functions are used in this assignment. They are explained in more detail below.

[1]https://deepai.org/machine-learning-glossary-and-terms/rmsprop

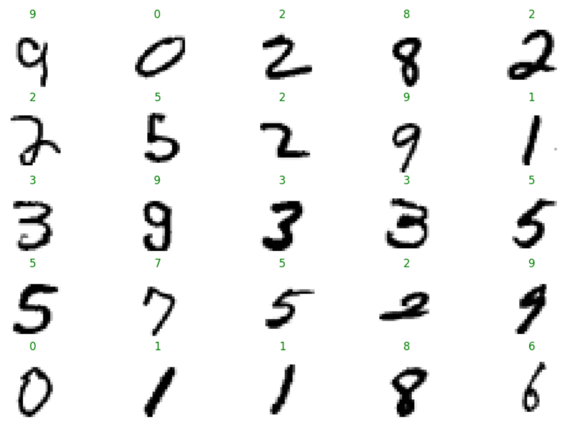


Fig. 3: Subset of dataset used for training for multiclass classification network

1) Sigmoid: Sigmoid activation, also known as logistic action, maps the input in range 0 to 1. In the final layer, it outputs the probability which can be thresholded for binary classification. Mathematically, it is given by:

$$S(x) = \frac{1}{1+e^{-x}} \tag{5}$$

This function is differentiable and monotonic. However, its derivative is not monotonic.

2) Tanh: This function is understood as rescaled sigmoid function and outputs values in range [-1,1]. This is believed to have advantage over sigmoid as negative values are mapped to strongly negative and zero values are mapped to zero. Mathematically, it is given by:

$$T(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{6}$$

This function is differentiable and monotonic. However, its derivative is not monotonic.

3) ReLU: ReLU stands for Rectified Linear Unit. It has advantage over sigmoid/tanh function mainly because they tend to saturate at the extreme values and only are sensitive to changes in the middle of their respective ranges. ReLU solves this problem by using a different kind of non-linear function. Mathematically, it is defined as:

$$f(x) = \begin{Bmatrix} x & if x > 0, \\ 0 & otherwise. \end{Bmatrix} \tag{7}$$

4) Softmax: Softmax function can be thought of as extension of logistic function to multiple dimensions. In this assignment, it is used as the last layer of multiclass-classification network to predict the probability distribution over all classes. Mathematically, this can be obtained with the following equation:

$$S(x) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{8}$$

where the summation is over probabilities for all inputs.

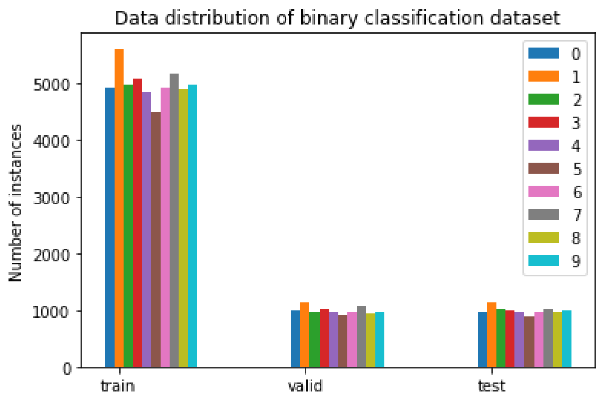


Fig. 4: Graph showing distribution of dataset for multiclass classification

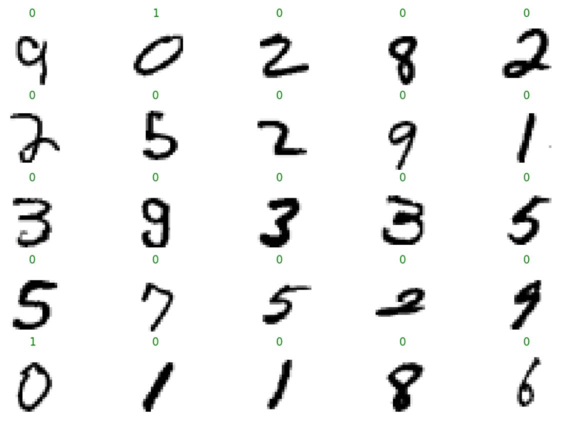


Fig. 5: Subset of dataset used for training for binary classification networks

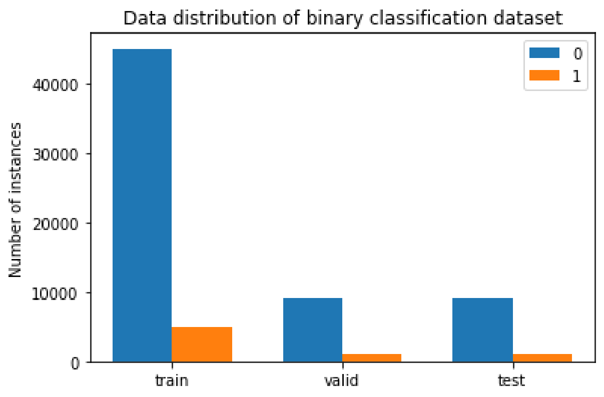


Fig. 6: Graph showing imbalanced dataset for binary classification

III. DATASET

The dataset used for training and evaluating the two networks mentioned in section IV is MNIST dataset [1]. MNIST stands for Modified National Institute of Standards and Technology. This dataset contain 60,000 images of size 28 x 28 square pixels containing handwritten numbers from 0 to 9. The classes for these numbers are also 0 to 9 respectively. This can be visualised in Figure 3. The title of each image displays the ground truth label. For each class, there are roughly equal number of instances in train, validation and test set as seen in the bar graph presented in Figure 4. To train the network, these labels are one-hot encoded in the multiclass case. For the binary classification problem, the labels are adjusted such that label for number '0' is 1 and for numbers '1-9' is 0. Figure 5 displays a subset of this dataset for visualisation. The title of each image displays the ground truth label. However, this leads to high imbalance in dataset. This imbalance can be visualised in Figure 6 which shows that there are almost 10 times more instances for '0' class than for '1' class all the three sets. This is why F-Score is evaluated for the binary classification networks.

IV. METHOD

This section discusses the methods for networks implemented in this assignment.

A. Binary Classification

1) Single Neuron: In this assignment, single neuron is used for binary classification of the created dataset explain in the previous section. Single neuron network can be visualised in Figure 7. where the input can be an image or a batch of images as discussed in Section V later. The output is the probability which is thresholded to give either true or false. Mathematically, single neuron operation looks as follows:

$$\hat{y} = f(\sum_i W_i x_i + b) \tag{9}$$

where $\hat{y}$ is the predicted output, f refers to the activation function, W_i refers to the the weights associated with the input pixels and b is the bias.

2) Single Hidden Layer: Single hidden layer in binary classification network replaces the neuron in the previous section by a layer of more than one neuron. This is usually called hidden layer. Mathematically this changes the computation as there are weights from input layer to hidden layer as well as from hidden layer to output layer. All these parameters have to be updated at each iteration. [2]

B. Multiclass Classification

Multiclass classification follows a similar routine as binary classification. The differences are as follows: The last layer of the network is replaced by softmax layer which outputs probability of each class. The loss function used to train this network is cross entropy discussed in section II-A. On MNIST dataset, the labels for each image are

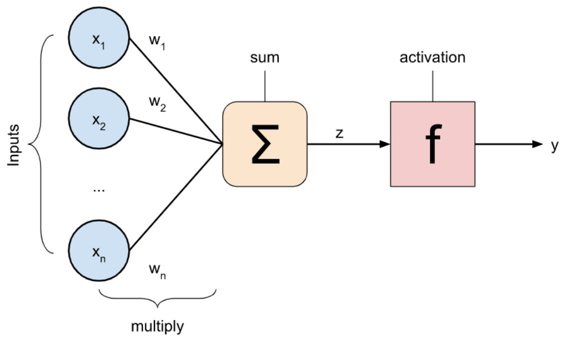


Fig. 7: Single Neuron Network. Source: Towards Data Science

one-hot encoded. Further, different architectures have been experimented with to get a higher accuracy. The parameters adjusted are discussed in Section V.

V. IMPLEMENTATION

These networks are written in Python with Tensorflow and Keras library. The development platform used is: Windows using the Jupyter Notebook and Visual Studio Code. The training set as discussed in Section III is further split into training and validation set with a fixed split size of 0.2. As metrics, accuracy was used and evaluated at each iteration of training. For binary classification, F1-Score was used as additional metric to account for the imbalance in the dataset.

A. Parameters

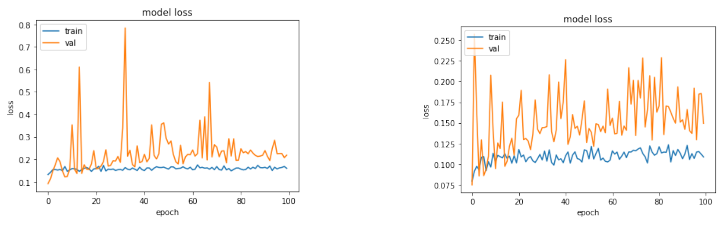
All of the methods presented above have several tunable parameters. These can help achieve a better performance. Each will be be presented next section.

1) Batch size: Batch size refers to the number of training images utilised in one iteration i.e before updating the parameters of the model. Different batch sizes are experimented with single neuron network and results are evaluated later in the report.

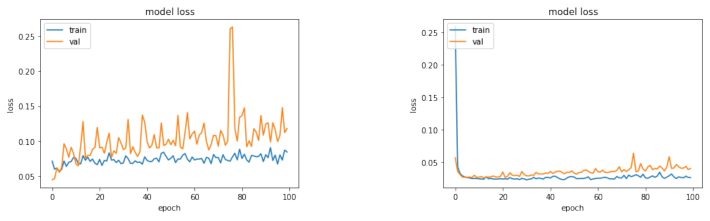
2) Number of neurons in hidden layer: The suggested number of neurons are usually between the number of inputs and number of outputs. Here, the input is image vector of size 784 and output is either one label or one-hot encoded vector of size 9. Different number of neurons are experimented with single hidden layer network and results are evaluated later in the report.

3) Learning Rate: Learning rate is used to defined the size of steps taken during optimization. In the current application, initial learning rate is 0.01 for all the three networks. This parameter varies according to the optimizer during training. Different optimizers are experimented with in the multiclass neural network.

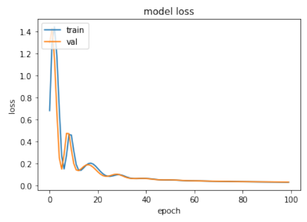
4) Iterations: Number of iterations used in the code is '100' for all the three networks as suggested in the assignment description.



(a) 32 (b) 64



(c) 128 (d) 1024



(e) whole training set

Fig. 8: Variation of loss for different batch sizes in single neuron model for binary classification

	Ground Truth		
Predicted Class		Positive	Negative
	Positive	8972	48
	Negative	37	943

TABLE I: Confusion Matrix for Single Neuron

VI. RESULTS

A. Binary Classification

1) Single Neuron: Since the dataset is simple, and only a binary classification is needed, a single neuron already performs above 99% accuracy. This network is evaluated for various batch sizes - 32, 64, 128, 1024, 48000 (Whole dataset). The reasons for using smaller batch sizes are as follows:

- Less memory is required because less number of training samples are used per iteration.
- Network converges faster because weights are updated more often.

The results for the network in Table II show that highest accuracy and F-score are obtained when the whole dataset is used as one batch.. This takes the lowest time (0.04 s) because the dataset is transferred into memory all at once. However, in Figure 8, the variation of loss shows that higher batch sizes converge slower than the smaller batch sizes. For example, the training loss for batch size 64 converges to range 0.075-0.100 within first five epochs whereas for the whole training set, convergence is achieved at almost 40th epoch. Further, from these graphs, the irregularities in

Batch Size	Accuracy	F-Score	Time per epoch(s)
32	0.9870	0.9143	5
64	0.9906	0.9506	2
128	0.9903	0.9486	1
1024	0.9893	0.9466	0.3
All	**0.9916**	**0.9573**	**0.04**

TABLE II: Results on test set for Single Neuron Network

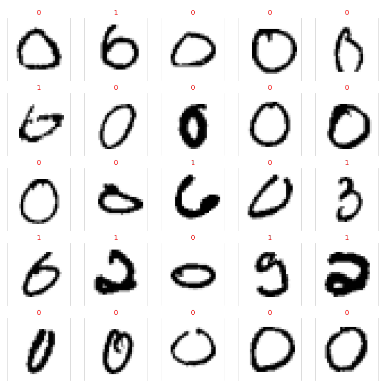
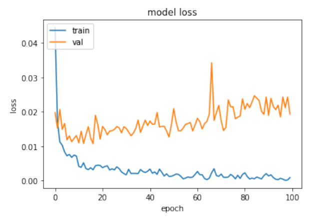
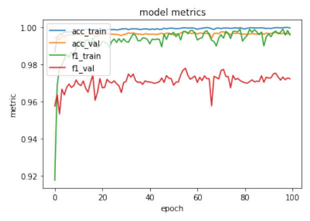


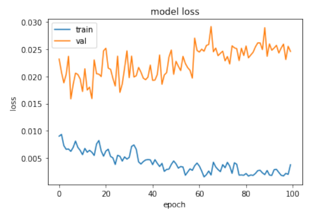
Fig. 9: Subset of wrongly classified instances with single neuron network



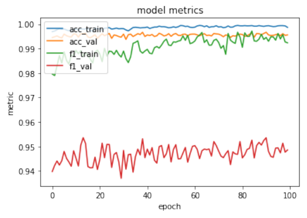
(a) Sigmoid loss



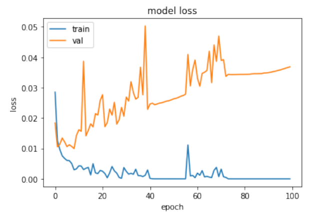
(b) Sigmoid metrics



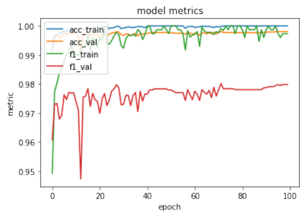
(c) Tanh Loss



(d) Tanh metrics



(e) ReLU loss



(f) ReLU metrics

Fig. 10: Variation of loss and metrics(accuracy,FScore) Single hidden layer for binary classification

validation loss for batch sizes 32, 64 and 128 can also be observed. This is probably because the updating the weights based on a small batch is more noisy. Although this noise can be good by helping to jerk out of local optima, the same noise and jerkiness can prevent the descent from fully converging to an optima at all. For larger batch sizes of 1024 and 48000, the validation loss follows the training loss closely and the curves are smoother. Figure 9 shows few instances that are wrongly classified by the network. It is seen that lot of numbers with label '1' have a generally curved structure resembling the number 0 itself.

The confusion matrix for the best batch size can be seen in Table I. There are slightly more false positives than false negatives. This can be expected, since the dataset is highly imbalanced having more positive instances.

2) Single Hidden Layer: In this section, binary classification with single hidden layer is discussed. Different kinds of activation functions and number of neurons are evaluated for this network. The batch size is selected as 64. As can be seen in Table III, the values of accuracy are more than 99.5 % and F-score is more than 98% for all the three activation functions. Looking at the validation loss in Figure 10, it is observed that it is higher than training loss. For ReLU, the increase is more drastic which can indicate overfitting. Similarly, it is seen that the F-Score is lower for validation set than for training by about 3-4 %. According to the aforementioned curves, sigmoid shows more stable training process with less noise. Further in the analysis, using the ReLU function, different number of neurons in the hidden layer are tried and the results are presented in Table IV. It is observed the with increasing number of neurons, the accuracies and F-Scores also increase. This is expected as more number of neurons let the network model increasingly

Hidden Layer Activation	Accuracy	F-Score
sigmoid	0.9964	0.9822
tanh	0.9970	0.9838
relu	**0.9974**	**0.9866**

TABLE III: Results for different activation function for single hidden layer binary classification on test set

complex relationships. The loss and metric seen in Figure 15. shows interesting curves for hidden layer with 1024 neurons. It can be seen that validation loss is increasing which can be indicative of overfitting but the difference between the losses is very small and therefore, the results on test set are similar. Further, after 60th epoch, the training accuracy becomes 1 and training loss becomes 0. Because of this, the network is not updated anymore and validation loss becomes constant. Figure 12 shows certain instances where the network performs wrong classification. As seen before, lot of handwritten zeros with label '0' have a deshaped form-pen-ups, circular shape. Similarly, lot of number having label '1' have generally curved structure resembling the number 0 itself.

B. Multiclass Classification

For this problem, original dataset with one hot encoded labels is used as training and test data. Different optimizers and network architectures are used to maximise the accuracy obtained. The default optimizer used to test the architectures is SGD. First of all, a simple architecture with single hidden

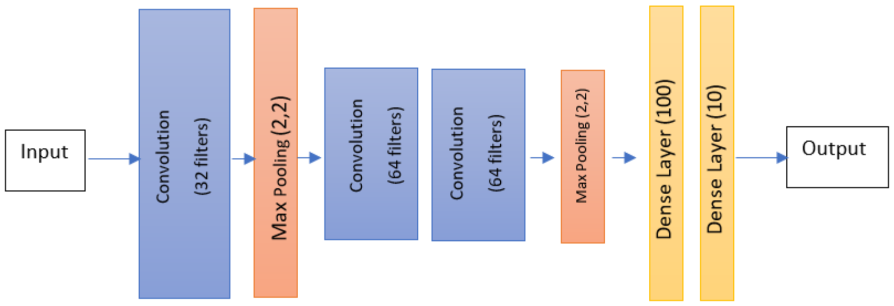


Fig. 11: CNN architecture

Hidden Layer Activation	Accuracy	F-Score
16	0.9972	0.9857
64	0.9974	0.9866
1024	**0.9983**	**0.9912**

TABLE IV: Results for different number of neurons for single hidden layer binary classification on test set

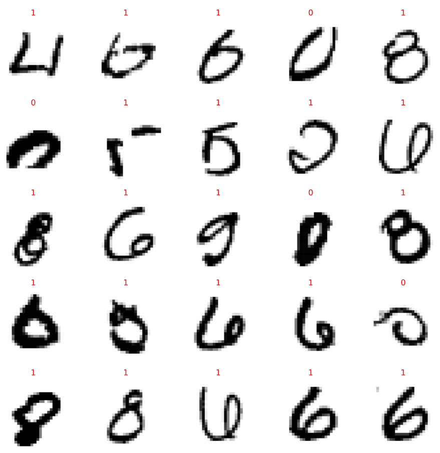
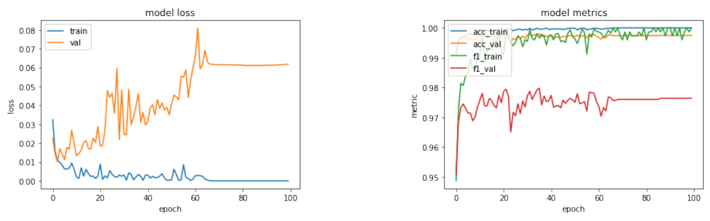


Fig. 12: Subset of wrongly classified instances with single hidden layer network

layer with 64 neurons is evaluated. This method gives an accuracy of 95.85 % on the test set. To increase the accuracy, rather than fully connected layer, convolutional blocks are used to train the network. The architecture of this network can be seen in Figure 11. On the test data, considerably higher accuracy of 99.37 % is obtained by this network architecture. This is because CNNs are able to extract specific features from the image on which the network can be trained more efficiently.

Further, to test if the accuracy can be increased, RMSProp was used to train the data. The network obtained achieved 96.63% accuracy on test set. With Adam optimizer, it achieves accuracy of 96.92%. The loss corresponding to each case can be seen in Figure 14. It is seen that for Adam optimiser, the loss values are very irregular and for small



(a) loss (b) metrics

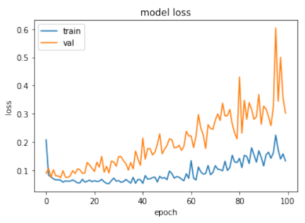
Fig. 13: Variation of loss and metrics(accuracy, F-Score) Single hidden layer for binary classification for 1024 neurons

increase in training loss, validation loss increases by a large amount. Similarly for RMSProp, the validation loss/metrics take huge jumps during the training process. This is better than adam since the two losses vary together. But the best loss is obtained for SGD where it is the lowest amongst the three and shows smooth curves. After 20th epoch, an accuracy of 1 is obtained on training data and the losses are almost constant. Therefore, CNN-architecture with SGD optimizer remained to be the highest- accuracy combination.

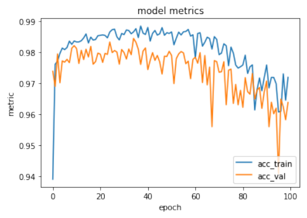
Inspecting the confusion matrix seen in Figure 15b. it can be seen that there are very few misclassifications. In these few cases the similar numbers are wrongly classified. Figure 15a shows these results obtained from multiclass classification from this network. These examples are such that they are even difficult to classify by human observer because of resemblance to other digits.

VII. CONCLUSIONS

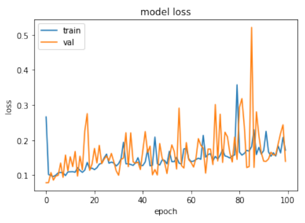
In conclusion, all main aspects of training a deep neural network were investigated in this report. First, it was shown how batch size affects the training times and accuracies, then different activation functions and hidden layer sizes were tried out. Lastly, several optimizers were investigated. With the results from all these experiments a convolutional neural network was built using the best parameters defined earlier which outperformed previous methods in multiclass classification. As future work, other datasets offering more challenging scenarios can be trained and evaluated with simple and complex network architectures.



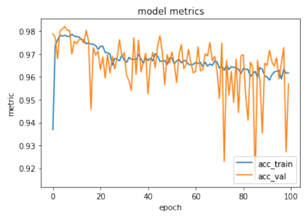
(a) Adam loss



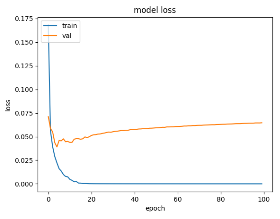
(b) Adam metrics



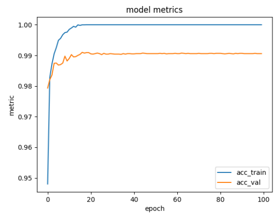
(c) Rmsprop Loss



(d) Rmsprop metrics

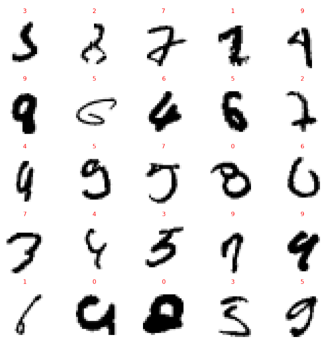


(e) SGD loss

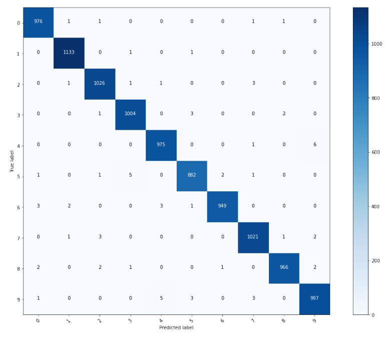


(f) SGD metrics

Fig. 14: Variation of loss and metrics(accuracy ,F-Score) CNN in multiclass classification with different optimizers



(a) Misclassifications



(b) Confusion Matrix

Fig. 15: Multiclass Classification CNN

References

[1] LeCun, Y. & Cortes, C. (2010). MNIST handwritten digit database

[2] Source: Sánchez-Monedero, Javier. (2013). Challenges in ordinal classification: artificial neural networks and projection-based method