CSC3022F - Computer Science

MNIST Image Classification

Momelezi Mchunu

Preface

Recognising handwritten digits might be an easy task for human brains, but computers need to be fed data and trained in order to recognise or try to recognise the given handwritten digit.

Acknowledgements

As most of the work is readily available online, the inspiration was drawn from multiple online sources

Abstract

Recognising handwritten digits might be an easy task for human brains, but computers need to be fed data and trained in order to recognise or try to recognise the given handwritten digit. Implementing a neural network that solves the problem was a challenge because there are so many ways to go about doing it. The solution was implemented using python, specifically the pytorch module. The best performing classifier I have achieves a little bit above 90% accuraccy

Keywords

Neural Network; topology;loss function; Optimizer

Contents

1 Introduction

In classifying real world images, accuracy is important. With a model that incorrectly classifies image, it maybe rendered useless or even dangerous and training machines to be able to classify handwritten digits is no exception. The report outlines in details how I went about implementing the classifiers. They were implemented using different number of layers, optimizers and activation functions. The assignment classifies 28 by 28 greyscale handwritten images. It does by taking each pixel and feeding it into the artificial neural network, computation is done and results are obtained.

2 Pre-processing Data

Each image has a 2D structure since all the images are made up a 28 by 28 structure. The input layer of the artificial neural network is linear meaning it cannot accept an input so the 2D vector is flattened to obtain a 1D vector where each row represents the data that should go into that neuron, this is done by using the torch.Flatten(input). Each pixel in the image has a value ranging between 0 and 255 representing intensity. When the image is being converted into a tensor, the values are then changed to range between 0 and 1, this is done by using the ToTensor() method. The normalised data is then separated into batches of 64 and then fed into the model, in order to train it.

3 Network Topology

There are 3 neural networks that I conducted my tests on, each network has one or 2 paramaters that were changed in order to see which network provided the highest accuracy percentage while not overfitting. These parameters are either the activation function, optimizer or the number of layers in the network.

3.1 Network with 3 layers using SGD optimizer

The figure shows a neural network made up of 784 input neurons, 512 hidden layer neurons and 10 output layer neurons. The network uses the ReLU activation function in between the input layer and the hidden layer. The ReLU function is used over the traditional because they saturate easily [1] 2.The figure also shows that the Softmax activation is used at the output layer to squash values to add up to one.The "new" output now has a values that add up to one and the value that corresponds to the input has the highest value. Using the SGD optimizer with a learning rate of 0.001 yielded an accuracy slightly less than 50%

NOTE: The diagram mentions 2 activation functions when there is only one ,i.e f1 = ReLU(x)

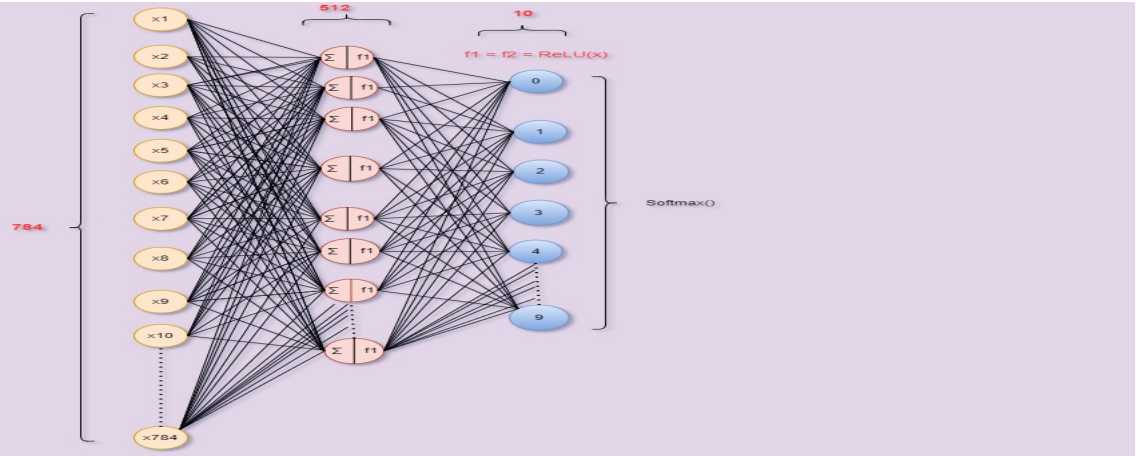


Figure 1: SGD Optimizer

3.1.1 Layers: Input Layer, Hidden layer(s) , Output layer

Input Layer

The input layer takes in the 784 tensor values and forward propagates them to the hidden layer when the necessary values have been done, this layer is standard in every neural network, this layer is fully connected to the hidden layer.

Hidden layer

The hidden layer takes in values from the input layer after they have been passed in to the ReLU activation function. In more complicated networks with multiple hidden layers each layer might get a refined output of the data from the previous layer. With each hidden layer identifying a certain feature in the data, e.g a line in a number. The neural network with one hidden layer had better the lowest lost values. The hidden layer is fully connected to the output layer and to the

output layer.The connectedness was chosen because it does not make any special assumptions about the input it receives from the input layers.

Output layer

The output layer takes the values from the fully connected hidden layer, and perform calculation on the values using forward propagation, values are then checked against the required output and if they are not as expected they are fed back into the network using backpropagation

3.1.2 ReLU function

$$\text{f(x)} = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

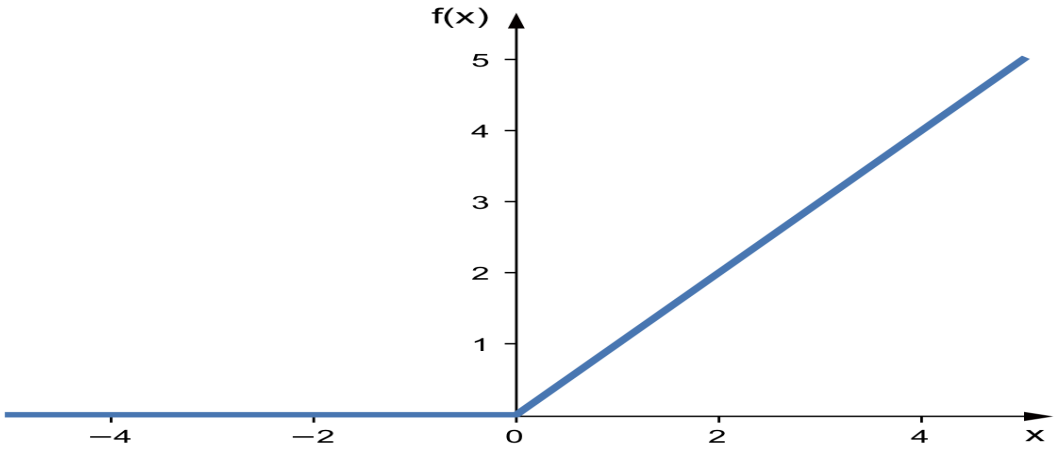


Figure 2: ReLU function

To train deep neural networks using stochastic gradient descent with backpropagation of errors, an activation function that looks and acts like a linear function but is actually a nonlinear function is required, allowing complex relationships in the data to be learned.In addition, the function must be more sensitive to the activation sum input and avoid saturation [1]. The solution had been floating around in the field for a while, but it wasn't brought to light until studies in 2009 and 2011 did.The rectified linear activation function, or ReL for short, is the answer.A rectified linear activation unit, or ReLU for short, is a node or unit that implements this activation function [1].

3.2 Network with 3 layers using Adam optimizer

The structure of this network is the same as the one in section 3.1.1 Network with 3 layers using SGD optimizer ,the figure shows a neural network made up of 784 input neurons, 512 hidden layer neurons and 10 output layer neurons. The network uses the ReLU activation function in

between the input layer and the hidden layer. The ReLU function is used over the traditional because they saturate easily [1] 2.The figure also shows that the Softmax activation is used at the output layer to squash values to add up to one.The "new" output now has a values that add up to one and the value that corresponds to the input has the highest value. Using the Adam optimizer with a learning rate of 0.001 yielded an accuracy above than 95%

3.2.1 Layers: Input Layer, Hidden layer(s) , Output layer

Input Layer

The input layer takes in the 784 tensor values and forward propagates them to the hidden layer when the necessary values have been done, this layer is standard in every neural network, this layer is fully connected to the hidden layer.

Hidden Layer

The hidden layer of this network is the same as the one in section 3.1.1 Network with 3 layers using SGD optimizer, they differ in a way that this network use the Adam optimizer to adjust weights and learning rate. This layer is fully connected to the output layer.

Output layer

The output layer takes the values from the fully connected hidden layer, and perform calculation on the values using forward propagation, values are then checked against the required output and if they are not as expected they are fed back into the network using backpropagation

3.3 Network with 4 layers using SGD optimizer

The figure shows a neural network made up of 784 input neurons, 2 hidden layers neurons containing 512 and 64 neurons respectively and 10 output layer neurons.he network uses the ReLU activation function in between the input layer and the first hidden layer.The Tanh activation function is used in between the first and second hidden layer. The combination of the tanh and ReLU activation function means that the weighted sum of their combination does not change the fact that they are monotone non-decreasing functions [2]. This network also obtained accuracy values below 50% and less than those of the one inn section 3.1.

NOTE: The diagram mentions 2 activation functions which use the same activation function where as that is not true,i.e f1 = ReLU(x) and f2=tanh(x) and after the output layer the Softmax activation function is applied

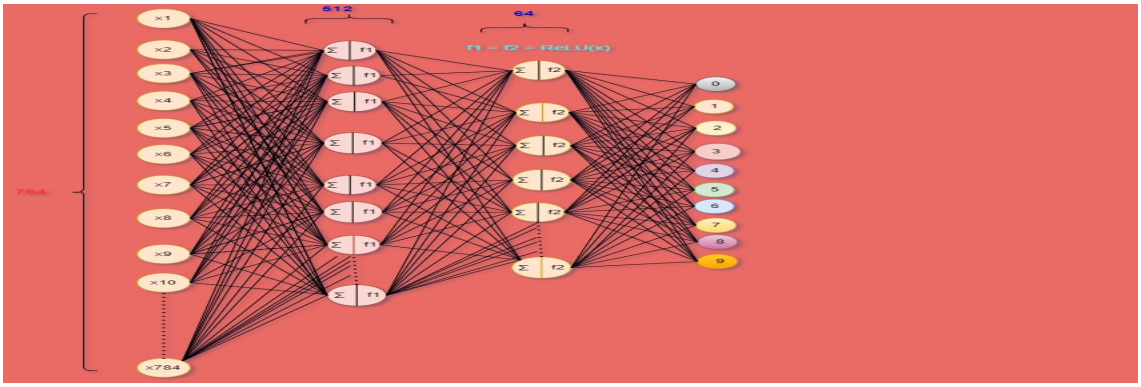


Figure 3: 2 hidden layer network

3.3.1 Layers: Input Layer, Hidden layer(s) , Output layer

Input Layer

The input layer takes in the 784 tensor values and forward propagates them to the hidden layer when the necessary values have been done, this layer is standard in every neural network, this layer is fully connected to the hidden layer. The values are passed through the ReLU activation function.

First Hidden Layer

The first hidden layer consists of 512 neurons, it is fully connected to the next and previous neuron. The values that are pass from this layer go through the Tanh activation.

Second Hidden Layer

This layer consists of 64 neurons. The layer is fully connected to the previous layer hidden layer and also to the output layer.

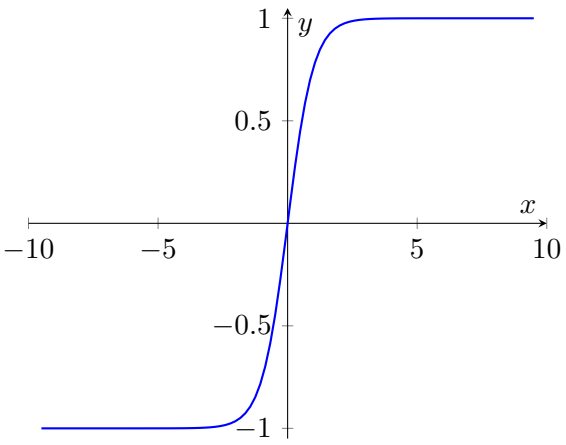
Output Layer

The output layer takes the values from the fully connected hidden layer, and perform calculation on the values using forward propagation, values are then checked against the required output and if they are not as expected they are fed back into the network using backpropagation. The values that go through to the output layer are between -1 and 1 due to tanh being the activation function in the previous hidden layer. After the calculations are done, these values are then passed into the Softmax activation function

3.3.2 Tanh Activation Function

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Figure 4: tanh formula



The function takes any real value as input and outputs values in the range -1 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

3.4 Model Performance graphs

The graphs below show the accuracy and loss values mentioned in the network topology section.

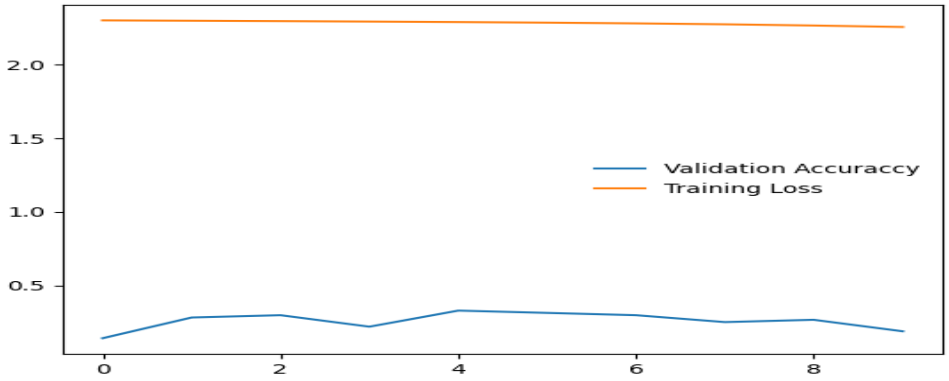


Figure 5: Accuracy and Loss of section 3.1

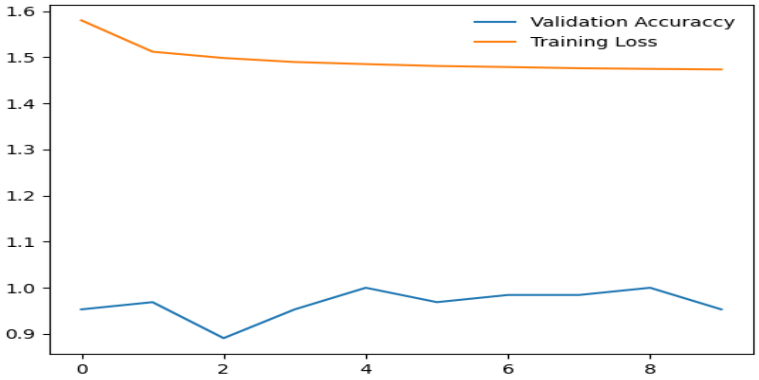


Figure 6: Accuracy and Loss of section 3.2

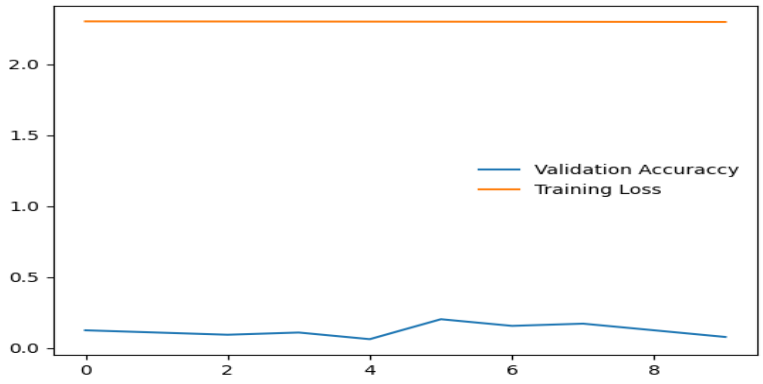


Figure 7: Accuracy and Loss of section 3.3

4 Loss functions

4.1 Cross Entropy Loss

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

Figure 8: log loss formula

The performance of a classification model whose output is a probability value between 0 and 1 is measured by cross-entropy loss, also known as log loss. As the projected likelihood differs from the actual label, cross-entropy loss grows. The log loss was picked because of its robustness as compared to other loss functions like the MSE that check every value.

5 Optimizers

5.1 Adam Optimizer

Adam is not the same as traditional stochastic gradient descent.For all weight updates, stochastic gradient descent maintains a single learning rate (called alpha), which does not fluctuate during training.Each network weight (parameter) has its own learning rate, which is adjusted individually as learning progresses [3]. Adam combines the finest features of the AdaGrad and RMSProp methods to create an optimization technique for noisy problems with sparse gradients.

$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$
$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$
$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$
$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

Figure 9: Adam formula

5.2 SGD

Stochastic gradient descent (SGD) in contrast performs a parameter update for *each* training example $x^{(i)}$ and label $y^{(i)}$:

$$\theta = \theta - \eta \cdot \nabla_\theta J\left(\theta; x^{(i)}; y^{(i)}\right).$$

Figure 10: SGD formula

6 Discussion

The main challenge I faced were to implemement an early stop loss as a way to mitigate overfitting and I ended up not implementing it. As an alternative I trained the model with varying values epochs and I ended up settling for 10 epoch as on average they produced better accuracy values but this is not consistent because the training data is shuffled which means the values for some data the model might start to converge after 6 iterations and not the best performing model will be obtained. This assignment could have been improved by adding an early stop loss to obtain the best model, and more feature engineering.

7 Conclusion

The classifier which uses the Adam optimizer with 3 layers achieves a better accuracy rate than the other 2 classifiers. It achieves accuracy values of above 95% where as the other 2 could not reach above 50. The model used also classifies values correctly. This assignment could have been extended in a way that I could have used other different Loss functions to see what they achieved as compared to my best performing classifier.

References

[1] J. Brownlee, "A gentle introduction to the rectified linear unit (relu)." 2019. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[2] X. Li, Z. Hu, and X. Huang, "Combine relu with tanh," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1, 2020, pp. 51–55.

[3] J. Brownlee, "Gentle introduction to the adam optimization algorithm for deep learning." 2020. [Online]. Available: https://medium.com/swlh/fully-connected-vs-convolutional-neural-networks-813ca7bc6ee5