**ARITHA MINDULA JAYARATNE**

**S4683873**

NIT6004 Project 1: Neural Network and Deep Learning

**CODE LINK -** [**https://colab.research.google.com/drive/15lZ\_HNsiz34MkdhZTNA17JZYXc2hSFOt?usp=sharing**](https://colab.research.google.com/drive/15lZ_HNsiz34MkdhZTNA17JZYXc2hSFOt?usp=sharing)

**Introduction.**

In the ever-changing world of artificial intelligence, image classification stands as one of the highly developed applications. These applications demonstrate the learning capabilities of neural networks. The project is based on creating a basic neural network model which is tailored to classify images from the popular MNIST data set in the machine learning community. This data set was created by Yann LeCun and used in many introductory neural network projects. The dataset consists of 28 x 28 grayscale images of handwritten numbers from 0 to 9. The main objective of the model is to develop and evaluate a MLP model and use different optimization techniques to recognize and predict the digits on the dataset accurately. The involves a set of tasks which includes importing the data set and preprocessing it to feed it to the MLP model and normalizing the pixel values. Creating the MLP model with the appropriate network layers and in the end compiling the model with given parameters (Metrics, Loss function and Optimizer). The project aims to achieve a high accuracy rate when predicting and recognizing the numbers and showcases the effectiveness of various neural networks and different optimization strategies.

**Literature review.**

The model is a neural network which is designed for image classification purposes, the model is suited for images which are 28x28 pixels. The dataset used for training and validation of this model is the famous MNIST data set. A detailed literature review is written on the image classification model which include the layers of the model, activation techniques and about the compilation and the evaluation techniques.

The first layer is the input layer and there is no learning involved. What basically happens in this layer is that the 2D 28x28 pixel image gets transferred into a 1D array of 784 pixels. This will reshape the data to be fed into the deeper denser layers of the model. Next is the 1 hidden layer of the model which has 256 neurons each with a ReLU (Rectified linear unit) activation function. This function is used to introduce nonlinearity into the model which helps the model to learn complex patterns. The 256 neurons helps the model to form, a boarder understanding of the features which are fed into the model. After the first dense layer a dropout layer is introduce into the model with a rate of 0.2. This means that 20% of the neurons are turned to zero in the training process. Dropout layer is used to regulate the model to prevent overfitting on the training data. The second hidden layer has the same activation function accept for the number of neurons which is scaled down to 128. And another dropout layer with a 20% rate is used reduce the overfitting of the data as the complexity of the model increases. The third layer has 64 neurons with the same activation function which will be helpful in the output layer. Next layer is the output layer with 10 neurons correspondent to the 10 classes (0-9). This layer uses a Softmax activation function which outputs a probability distribution in the 10 classes. Making the model suitable for multi class classification.

In the model compilation part, there are 3 ways to optimize the model. Each function will be explained in depth for the reader to understand. First one is the optimizer function. The optimizer used in this model is the “Adam” optimizer. This is an extension of the stochastic gradient decent which has been updated to improve and adjust the learning rates of the parameters (GeeksforGeeks, n.d.).

The next is the loss function which is Sparse Categorical Crossentropy. This function is used to find the loss rate of multi- class classifications where the labels are presented as numbers. This measures the probability of the predicted distribution and the true distribution. This is the best for training the model to give out the probabilities that reflect the correct class.

The next is the metrics function. the accuracy metric is a common function used to evaluate classification models. This shows the percentage of the correctly predicted instances from the total instances in the model. This provides a quick measurement on the model’s performances.

**Methods applied and results.**

**Task 1**

**A black rectangle with white text

Description automatically generated**

Import the necessary libraries need to create the model and to visualize the results.

**Task 2**

A screenshot of a computer program

Description automatically generated

I loaded the MINST data set and divided the data into train images, train labels, test images and test labels. After loading the data, then preprocess it by dividing it by 255 to normalize the images.

A screenshot of a computer program

Description automatically generated

First, I checked the image size and the label size of the training and testing data. And checked the types of the images and the labels. Image type is float64 and the unit type is unit8. And the pixel value of the images is 0.0 to 1.0.

**A screenshot of a computer

Description automatically generatedTask 3**

The classifier MLP model has three layers with 256,128 and 64 neurons respectively with Relu activation function and two dropout layers with 20% rate in between layer 1 and 2 and another in between 2 and 3. Also there is an input layer and an output layer. The out put layer has 10 neurons with a Softmax activation function. Finally, I ran this code chunk which will create the sequential model.

Task 4

In the compilation section I have set the below mentioned parameters for the model to function better and receive a good accuracy rate.

A screenshot of a computer program

Description automatically generated

Task 5

A screenshot of a computer

Description automatically generated

Trained the model for 15 Epchos on the training data set and 20% of the data set is set a side for the testing of the model to make predictions.

A computer screen shot of a program code

Description automatically generated

Accuracy of the model is 97.98% and which is a good rate for a MLP model.

A collage of numbers

Description automatically generated

Comparing the models’ predictions against the true labels of the data set.

A graph of the same type of graph

Description automatically generated with low confidence

The fluctuations in validation loss and accuracy highlight these trends, which imply that although the model is learning efficiently as seen in the training data, its capacity to generalize to new data (as seen in the validation data) may not be as strong.

**Challenges and problems during the project.**

* Quality and availability of data – High quality is a crucial for training machine learning models. However, obtaining high quality can be difficult. This issue is a challenge in real time machine learning problems.
* Overfitting and underfitting - Models may exhibit underfitting, in which they are unable to identify the underlying patterns in the data, or overfitting, in which they perform well on the training set but poorly on fresh data. It can be difficult to strike the correct balance while avoiding these problems.
* Limited resources and time - Model training and experimentation may be difficult due to time or computing resource limitations like GPU or TPU power.

**Ethical issues in this scenario.**

The privacy of the photos used to train and evaluate the classifier is one ethical issue. Making sure that these photos don't contain sensitive information or violate people's privacy is vital. To address this, we remove any personally identifiable information from the images before using them in training by implementing data anonymization techniques. Neural networks may also reproduce biases found in the training set, which could produce unfair or discriminating results. To address this, techniques like data augmentation are used to carefully compile a variety of training datasets and conduct regular audits to find and correct biases.

**References.**

* Khemani, B., Patil, S., Kotecha, K. & Tanwar, S., 2024. A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. *Journal of Big Data*, 11(18), available at: <https://doi.org/10.1186/s40537-023-00876-4> [Accessed 24 May 2024].
* GeeksforGeeks, n.d. Adam Optimizer in TensorFlow. Available at: <https://www.geeksforgeeks.org/adam-optimizer-in-tensorflow/> [Accessed Date].
* Kriegeskorte, N. and Golan, T., 2019. Neural network models and deep learning. *Current Biology*, *29*(7), pp.R231-R236.
* Canziani, A., Paszke, A. and Culurciello, E., 2016. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.
* Carpenter, G.A., 1989. Neural network models for pattern recognition and associative memory. *Neural networks*, *2*(4), pp.243-257.
* Tibshirani, R., 1996. A comparison of some error estimates for neural network models. *Neural computation*, *8*(1), pp.152-163.