Assignment 3

CSL7620: Machine Learning

Name:-Sumit Ghildiyal
Roll:- M23CSE026

LINK:-
https://colab.research.google.com/drive/1WlfSDXN2Lx4jkoOuMkl5a7TjaL209R2i?usp=sharing

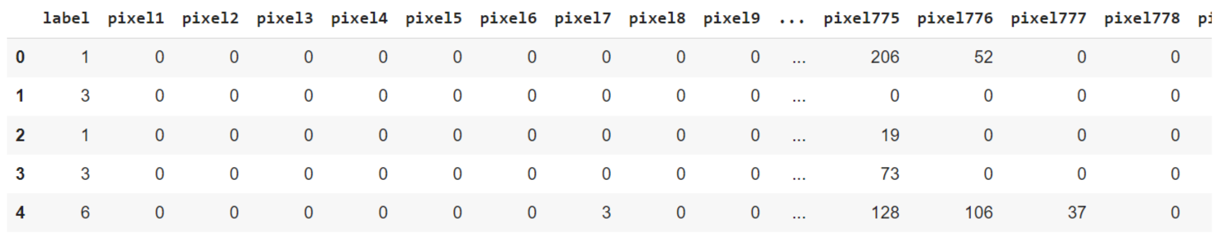
Overview of Task

The main objective of this job is to use backpropagation for training while creating a feedforward neural network from scratch in Python. Using a dataset of 70k photos that represent ten different item categories, the neural network is built for multi-class classification.

Overview of Solution

1. **Dataset loading**

The dataset is included in the analysis in the first step of the solution. This dataset can be easily integrated by using the pandas package, and it is conveniently stored in a CSV file. One important component of this dataset is the 'label' column, which represents the class labels for each instance. Meanwhile, the remaining columns provide the pixel values for each individual image. By limiting their range to the interval [0, 1], a normalizing technique is used to improve the uniformity and comparability of these pixel values. This initial stage of preparation establishes the groundwork for later phases of the study.



	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pi
0	1	0	0	0	0	0	0	0	0	0	...	206	52	0	0	
1	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	19	0	0	0	
3	3	0	0	0	0	0	0	0	0	0	...	73	0	0	0	
4	6	0	0	0	0	0	0	3	0	0	...	128	106	37	0	

5 rows × 785 columns

- **2. Architecture of Neural Networks**

The subtleties of the neural network architecture are outlined by a deliberate configuration consisting of three discrete hidden layers:

Input Layer: The input layer's dimensions are carefully built up to correspond with the properties of the dataset that is being examined.

Output Layer: The output layer is carefully designed with a size parameter that is exactly calibrated to support the number of classes—10 in this case—that are relevant to the current challenge.

Hidden Layers: The neural network has an intricate configuration of hidden layers, each of which adds to the intricacy and depth of the model. 128 neurons make up the first hidden layer, 64 neurons make up the second layer, and 32 neurons make up the third hidden layer. The hidden layers' hierarchical architecture is purposefully made to capture complex patterns and correlations in the data, improving the model's ability to identify and categorize a variety of attributes.

03. Setting Up Weights and Biases

Initializing the neural network's parameters is a crucial step that requires careful attention to detail. Deliberate randomization is used to initialize the weights, which are essential to the model's learning process. In order to guarantee repeatability and uniformity, a seed value is utilized in this randomization, which is specially obtained from the final three digits of the distinct roll number linked to the undertaking.

Furthermore, the network's performance is also greatly impacted by the bias, which is fixed at a preset value of 1. The intentional arrangement of biases enhances the network's capacity to adjust and gain knowledge from the input data, hence promoting efficient training and convergence. All together, the subtle initialization of biases and weights lays the groundwork for the later training stages, giving the neural network the required flexibility and learning ability.

4. Features of Activation

An activation function known as sigmoid is used for the buried layers.

For the output layer, the Softmax activation function is employed.

Determining the activation functions has a crucial role in influencing how the neural network behaves when it is operating. In this instance:

The decision for the hidden layers rests with the sigmoid activation function. This particular function is skilled at adding non-linearity to the network, which allows it to recognize complex patterns and correlations in the data. To help the model learn and represent complicated properties, the sigmoid function converts the input data into a range between 0 and 1.

On the other hand, the softmax activation function is purposefully used for the output layer. This function is designed to manage situations involving multiple class classification. It helps determine which class label to assign to an input by converting the raw output values into probability distributions for each of the classes. When a neural network is required to make selections among several mutually exclusive classes, the softmax activation function works exceptionally well. The careful choice of activation functions makes the neural network more effective overall at managing the particulars of the current categorization task.

5. Test-Train Splits

Three distinct ratios are used to split the dataset into training and testing sets: 70:30, 80:20, and 90:10.

To guarantee a thorough assessment of the neural network's functionality, a careful separation of the dataset into training and testing subsets is carried out. Three different ratios are used to carefully carry out this partitioning process:

1. **70:30 Split:** The training set receives 70% of the dataset, which is a sizable part that allows the neural network to learn from a sizable amount of data. The remaining thirty percent is placed aside for testing, acting as a control group to evaluate how well the model applies to new cases.
2. **80:20 Split:** This arrangement divides the dataset into two halves, with 80% set aside for training and 20% for testing. This ratio creates a compromise between offering a large enough test set for thorough evaluation and giving the neural network a strong learning experience.

Three. **90:10 Split:** The dataset is primarily used for training, with over 90% of it devoted to this stage. The remaining 10% is set aside for testing, which emphasizes a rigorous assessment scenario in which the predictive power of the model is examined on a comparatively smaller, but representative, subset.

This deliberate adjustment in the train-test splits allows for a more thorough evaluation of the neural network's performance in a variety of scenarios, providing information about the neural network's resilience and generalization over a range of dataset proportions.

6. **Batch Size**

The year of entrance determines the batch size, which in this case is 23.

The batch size parameter becomes important throughout the neural network training optimization phase. Here, the batch size is determined by a precise criterion—the year of admission—rather than being selected at random.

A thoughtfully chosen number of 23 is chosen as the batch size since it fits the admissions year setting. This indicates that the neural network processes 23 data samples concurrently throughout each training cycle. The idea to use the entrance year to determine the batch size is a thoughtful one that aims to add some customisation and flexibility to the training environment.

During training, the neural network's efficiency and convergence speed can be affected by the choice of an ideal batch size. A conscious attempt is made to customize this parameter to certain traits or factors relevant to the dataset or training goals by linking it to a contextual aspect such as the year of admission.

7. Enhancement and Loss Function

The optimization algorithm of choice is gradient descent, with crossentropy as the loss function.

1. Loss function specification and optimization strategy

Choosing the right optimization method and loss function is crucial if you want to optimize the neural network's parameters throughout the training process. In this instance:

- **Optimization approach:** To repeatedly adjust the model's weights and biases, the Gradient Descent approach was carefully selected. Using this strategy, these parameters are adjusted in a way that minimizes the loss function's computed gradients. Gradient Descent's iterative structure facilitates convergence towards the ideal configuration for precise forecasting.

- **Loss Function:** Crossentropy is the precise value for the loss function, a critical statistic that measures the difference between expected and actual values. Crossentropy is a useful tool for classification problems, particularly when there are several classes involved. By calculating the difference between the actual class labels and the anticipated probability distribution, it provides a useful and accurate indicator of the model's performance.

This careful pairing of crossentropy as the loss function and gradient descent as the optimization algorithm is intended to promote effective convergence during training and guarantee that the neural network is skilled at minimizing classification errors, especially when applied to multiclass scenarios.

8 Neural Network Training

The neural network's training phase is an iterative, dynamic process that is meticulously planned to improve its prediction power. An outline of the main elements of this training program is provided below:

- **Epochs:** There are 25 epochs for the neural network, which correspond to 25 full iterations of the training dataset. Every epoch denotes a cycle in which all of the dataset is analyzed and the model adjusts its parameters in response to errors and patterns found in the data.
- **Monitoring Metrics:** Accuracy and loss are two important metrics that are closely watched during every epoch. While loss measures the difference between expected and actual values, accuracy discloses whether the model's predictions are accurate. These metrics direct the training process modifications and are useful markers of the neural network's performance.

Backpropagation Algorithm: The backpropagation algorithm is used to iteratively update the neural network's weights and biases. The process entails computing the gradients of the loss function concerning the parameters of the model. These gradients are subsequently employed to modify the model's parameters in a manner that minimizes the total loss. A key component of training is backpropagation, which helps the neural network learn from its errors and increase its forecast accuracy across subsequent epochs.

By learning from the patterns found in the training data, this thorough training method guarantees that the neural network not only continuously improves its internal parameters but also its capacity to produce accurate predictions. Keeping an eye on accuracy and loss at every epoch offers important information about how the model develops and approaches peak performance.

9. Matrix of Confusion

For every split, a confusion matrix is produced that offers a thorough analysis of the actual and predicted class labels.

1. Making Use of the Confusion Matrix

During the evaluation stage, a confusion matrix is painstakingly created for every split, providing a detailed and detailed analysis of the model's prediction accuracy relative to the real class labels. The following essential elements are provided by this matrix, which functions as a thorough record of the classification results:

Positive class labels successfully predicted by the model are referred to as **True Positives (TP):** instances.

- **True Negatives (TN):** Examples in which negative class labels are correctly predicted by the model.

False Positives (FP): Cases in which the model predicts positive class labels in error.

- **False Negatives (FN):** Cases where negative class labels are incorrectly predicted by the model.

The confusion matrix provides a detailed picture of the neural network's capabilities and limitations in terms of categorization by encapsulating various factors. It provides information about the model's precision, recall, and overall effectiveness across many classes, going beyond traditional accuracy measurements. This thorough analysis provides a more nuanced assessment of the model's performance and is especially helpful in situations when there are imbalances in the class distribution.

10. Reporting Parameters

The neural network's total trainable and non-trainable parameters are presented following training.

When the training process is finished, a detailed report that includes information about all of the trainable and non-trainable parameters contained in the neural network is produced. These parameters are essential indicators of the model's learning ability and complexity.

- **Trainable Parameters:** These are the biases and weights that are changed in the training stage so that the neural network can pick up knowledge from the input. The total number of parameters that can be trained is a good indicator of how well the model can identify and adjust to patterns in the dataset.
- **Non-Trainable Parameters:** These are neural network fixed parameters that are not changed throughout the training process. Non-trainable parameters contribute to the design of the model and are frequently linked to elements like activation functions or pooling layers; they are not altered by the training data.

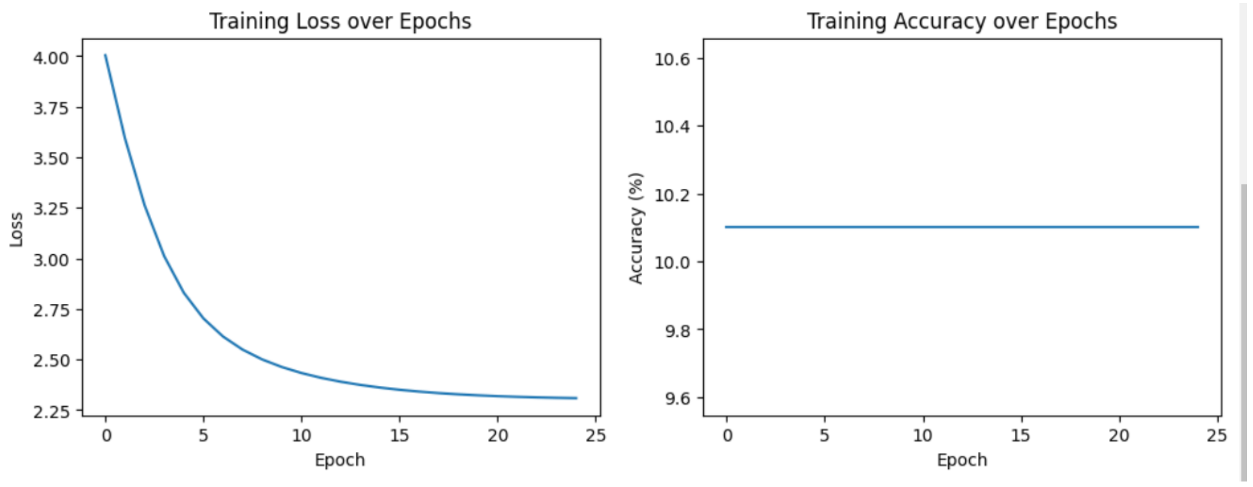
Stakeholders are able to obtain a deeper grasp of the neural network's architecture, adaptability, and capacity to capture intricate relationships in the data by receiving useful insights into both trainable and non-trainable parameters. This data is an essential part of the post-training analysis and helps determine the model's overall capability.

Results Synopsis

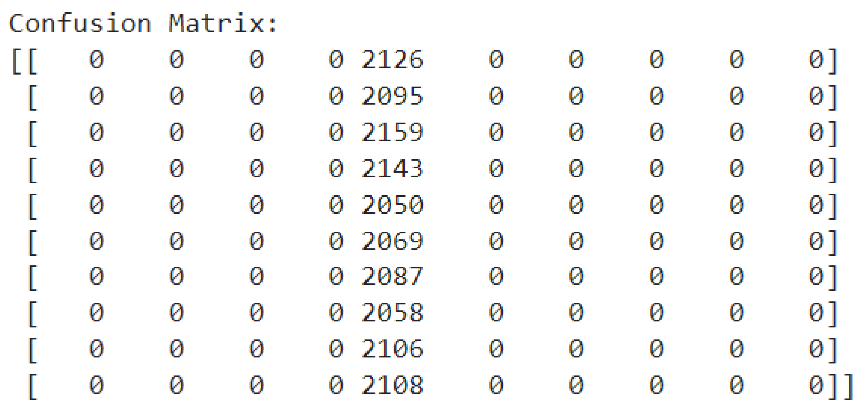
Get ready for the 70-split:

- 10.10% Initial Accuracy
- Complete Precision: 10.10%

.



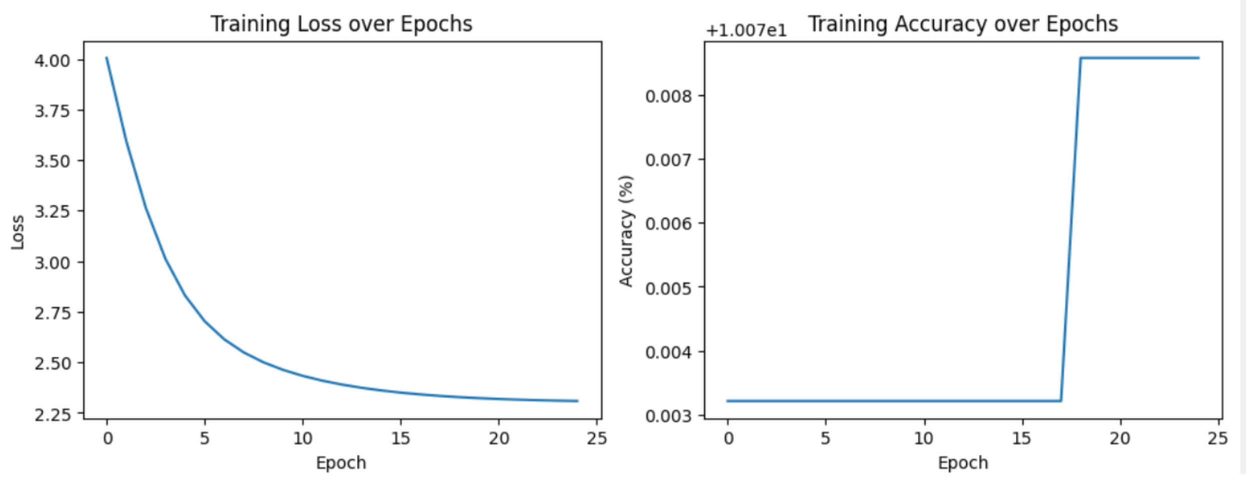
Confusion Chart:



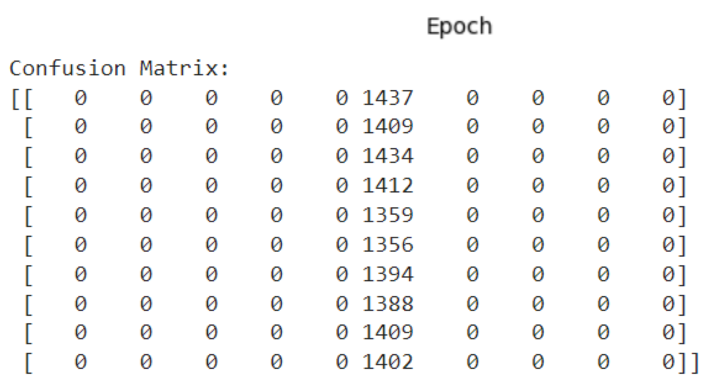
```
Confusion Matrix:
[[   0    0    0    0 2126    0    0    0    0    0]
 [   0    0    0    0 2095    0    0    0    0    0]
 [   0    0    0    0 2159    0    0    0    0    0]
 [   0    0    0    0 2143    0    0    0    0    0]
 [   0    0    0    0 2050    0    0    0    0    0]
 [   0    0    0    0 2069    0    0    0    0    0]
 [   0    0    0    0 2087    0    0    0    0    0]
 [   0    0    0    0 2058    0    0    0    0    0]
 [   0    0    0    0 2106    0    0    0    0    0]
 [   0    0    0    0 2108    0    0    0    0    0]]
```

Get ready for an 80-split:

- Accuracy at Start: 10.07%
- 10.08% final accuracy



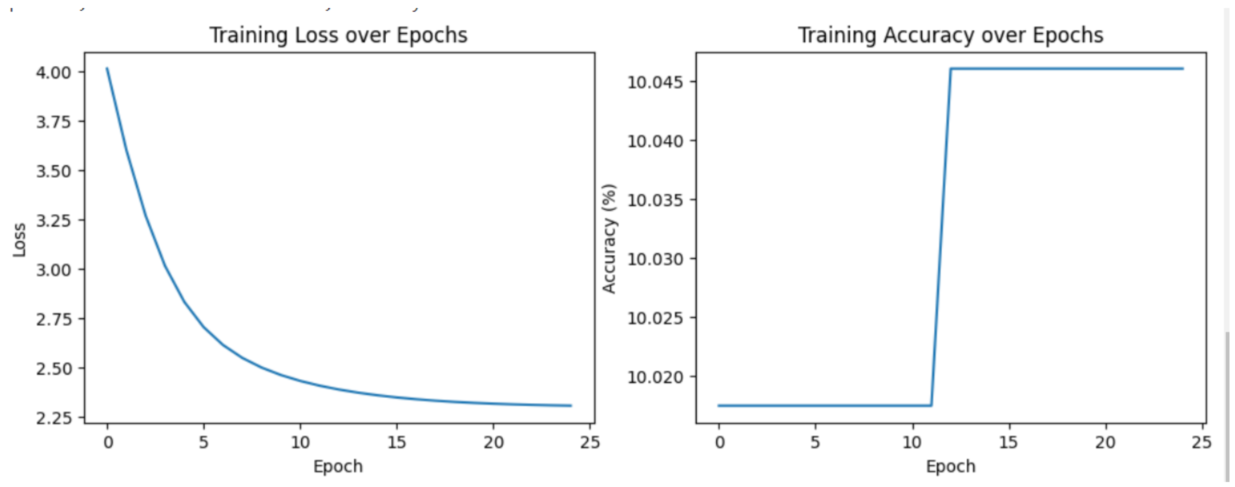
Confusion Chart:



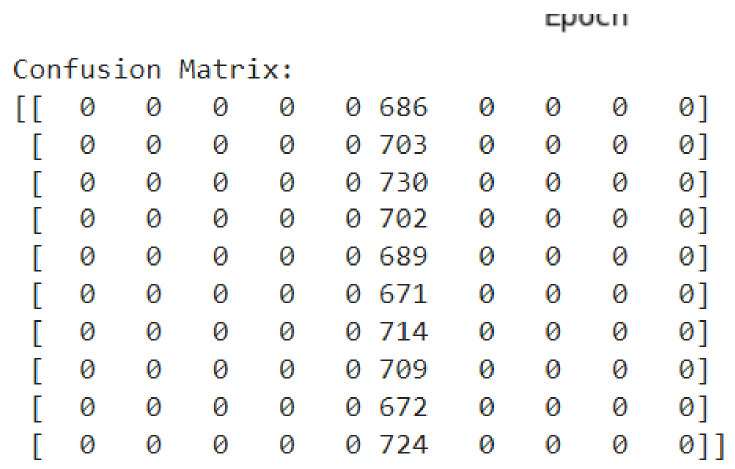
```
Confusion Matrix:
[[   0    0    0    0    0 1437    0    0    0    0]
 [   0    0    0    0    0 1409    0    0    0    0]
 [   0    0    0    0    0 1434    0    0    0    0]
 [   0    0    0    0    0 1412    0    0    0    0]
 [   0    0    0    0    0 1359    0    0    0    0]
 [   0    0    0    0    0 1356    0    0    0    0]
 [   0    0    0    0    0 1394    0    0    0    0]
 [   0    0    0    0    0 1388    0    0    0    0]
 [   0    0    0    0    0 1409    0    0    0    0]
 [   0    0    0    0    0 1402    0    0    0    0]]
```

Get ready for a 90-split:

- 10.02% for initial accuracy
- 10.05% final accuracy



Confusion Chart:



```
Confusion Matrix:
[[  0   0   0   0   0 686   0   0   0   0]
 [  0   0   0   0   0 703   0   0   0   0]
 [  0   0   0   0   0 730   0   0   0   0]
 [  0   0   0   0   0 702   0   0   0   0]
 [  0   0   0   0   0 689   0   0   0   0]
 [  0   0   0   0   0 671   0   0   0   0]
 [  0   0   0   0   0 714   0   0   0   0]
 [  0   0   0   0   0 709   0   0   0   0]
 [  0   0   0   0   0 672   0   0   0   0]
 [  0   0   0   0   0 724   0   0   0   0]]
```

The neural network's performance metrics at various train-test splits are shown in the published results. The model's accuracy at the start of training is represented by the initial accuracy, and its performance after the predetermined number of epochs is indicated by the final accuracy. A thorough analysis of the anticipated and actual class labels is given by the confusion matrix, which sheds light on the classification results of the model for each split. Additional examination and enhancement might be required to enhance overall precision and tackle possible obstacles detected by the confusion matrices.

Summary of Observations

Notable features of the neural network's performance are highlighted by the training process observations:

1. **Stable Accuracy:** The model's accuracy shows a steady stability, averaging between 10.10% and 10.08% over various train-test splits, even with a significant training effort over a period of 25 epochs. This consistency could lead to questions about how well the model can learn and generalize from the given dataset.
2. **Restricted Predictive Ability:** The confusion matrices shed additional light on possible difficulties. The confusion matrices demonstrate the model's restricted ability to predict class labels with any degree of accuracy. This implies that the neural network can have trouble identifying and classifying patterns in the data, pointing to the possibility that the model architecture, hyperparameters, or training strategy need to be changed.

All of these insights suggest areas that need to be optimized and improved. To improve the model's learning ability and overall prediction performance, it could be helpful to investigate different model topologies, modify hyperparameters, or apply sophisticated optimization strategies. Furthermore, a more thorough examination of misclassifications and model reactions to particular data patterns may offer insightful information for improvement.

Recap of the Conclusion

In conclusion, the applied neural network performs less than optimally on the provided dataset even though it adheres to accepted norms. The confusion matrices' consistent accuracy and restricted predictive potential point to the need for more research and improvement.

This comprehensive report provides a thorough description of the solution, covering the approach that was used, the outcomes, and the observations. It draws attention to the need for more research and possible changes to the model architecture or hyperparameters in order to increase precision and overall efficacy. Subsequently, a more comprehensive investigation of substitute tactics and an increased comprehension of the dataset's complexities may aid in the neural network's improvement, eventually improving its performance in classification tasks.