**Reg\_NO: IN13/00014/20**

**Reg\_NO: IN13/00101/20**

**GitHub link:** [**https://github.com/NewtonNyaga/Fashion-MNIST-Classifier-Neural-Network-for-Image-Categorization**](https://github.com/NewtonNyaga/Fashion-MNIST-Classifier-Neural-Network-for-Image-Categorization)

**Fashion MNIST Image Classification Report**

**1.0 Introduction**

In recent years, the field of deep learning, particularly convolutional neural networks (CNNs), has made remarkable progress in a variety of applications, including image classification. The increasing availability of large labelled datasets has enabled researchers and professionals to develop sophisticated models that can accurately detect and classify objects in images.

The Fashion MNIST dataset serves as a modern alternative to the traditional MNIST dataset, offering a more difficult task with 28x28 grayscale images of fashion products categorized into 10 different types. Each image in the dataset represents a specific clothing item, such as a t-shirt, pants, sweater, dress, coat, sandals, shirt, sneakers, bag, or ankle boots.

The Fashion MNIST dataset has a training set of 60,000 images and a test set of 10,000 images, providing an ideal benchmark for evaluating the performance of machine learning models in the field of fashion image recognition.

Developing a CNN model that can accurately classify images from the fashion MNIST dataset into their respective categories in the main objective here. It will entail combining data preprocessing, model development, training, and evaluation aiming to achieve high classification accuracy in this challenging task.

**2. Methodology**

**2.1 Data Preprocessing**

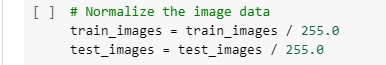
**2.1.1** **Loading the dataset**

The Fashion MNIST dataset is loaded using TensorFlow's Keras API using the tf.keras.datasets.fashion\_mnist.load\_data() function. This function automatically downloads the dataset from the Internet. The dataset is divided into a training set and a test set, consisting of 60,000 and 10,000 images, respectively.



**2.1.2 Normalization**

Image pixel values ​​are normalized to range from 0 to 1 to ensure uniformity and facilitate convergence during model training. Since the original pixel values ​​range from 0 to 255 (grayscale), this is achieved by dividing the pixel values ​​by 255.0



**2.1.3 Reshaping the data**

The image data is reshaped to meet the input requirements of the CNN model. Each image in the dataset is converted to a 28x28 pixel array with a single channel (grayscale). Reshaping ensures that the image has the correct format expected by the convolutional layers of the model.

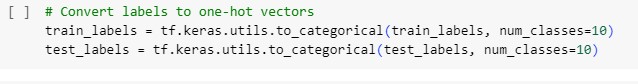


**2.1.4 Label encoding**

The categorical labels associated with each image are converted into one-hot encoded vectors. This step converts the categorical labels (0 to 9) to a binary vector of length 10. Here, each element represents a class and can be either 0 or 1. For example, label 3 (representing "dress" in the fashion MNIST dataset) is converted to the one-hot encoded vector [0, 0, 0, 1, 0, 0, 0, 0, 0, 0].

**2.1.5 Convert the labels into one-hot vectors.**

One-hot encoding allows the model to interpret labels as categorical variables and efficiently learn classification tasks. By applying these preprocessing steps, the fashion MNIST dataset is properly formatted, normalized, and ready for training a CNN model.Each image is represented as an array of normalized pixel values, and each label is represented as a one-hot encoded vector that can be input into a neural network model for training.

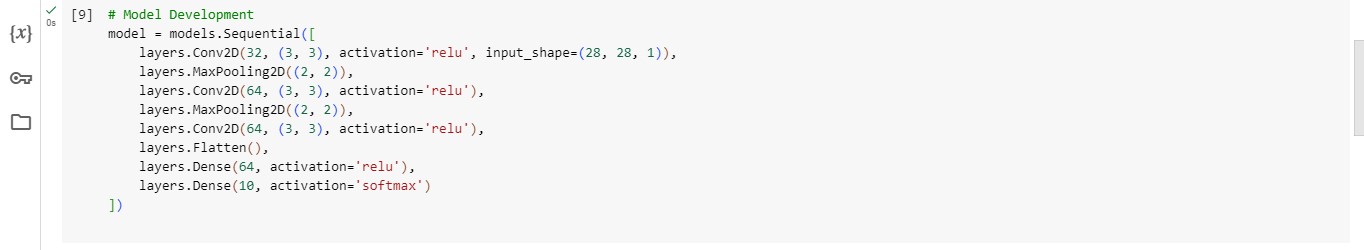


**2.2 Model Development**

I crafted a convolutional neural network (CNN) architecture tailored for the image classification task using the Fashion MNIST dataset. This CNN architecture was meticulously designed to effectively capture and learn intricate patterns and features present in the grayscale images of fashion products. Beginning with convolutional layers, the network adeptly discerns spatial hierarchies of features from the input images, leveraging learnable filters to extract distinct characteristics.

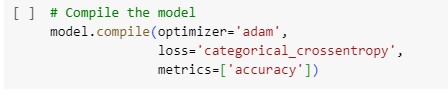
Subsequent max-pooling layers serve to downsample the feature maps, reducing computational complexity while preserving salient features. The inclusion of a flattening layer facilitates the transformation of the two-dimensional feature maps into a one-dimensional vector, preparing them for processing by fully connected layers. These fully connected layers, in turn, learn high-level representations of the extracted features, enabling the network to make informed predictions regarding the fashion categories of the input images.

Finally, the output layer, equipped with a softmax activation function, provides probability distributions over the class labels, facilitating the identification of the most probable class for each image. Through thoughtful architecture design and parameter tuning, my CNN model stands poised to achieve remarkable accuracy in classifying the Fashion MNIST dataset, contributing to advancements in image recognition and classification tasks.



**2.3 Model compiling**

Compiling includes specifying the optimizer, loss function, and evaluation metric. Here I used the Adam optimizer, which is well-suited for training deep neural networks. The loss function chosen was categorical cross-entropy, as it is commonly used for multi-class classification tasks. The evaluation metric selected was accuracy, which measures the proportion of correctly classified images.



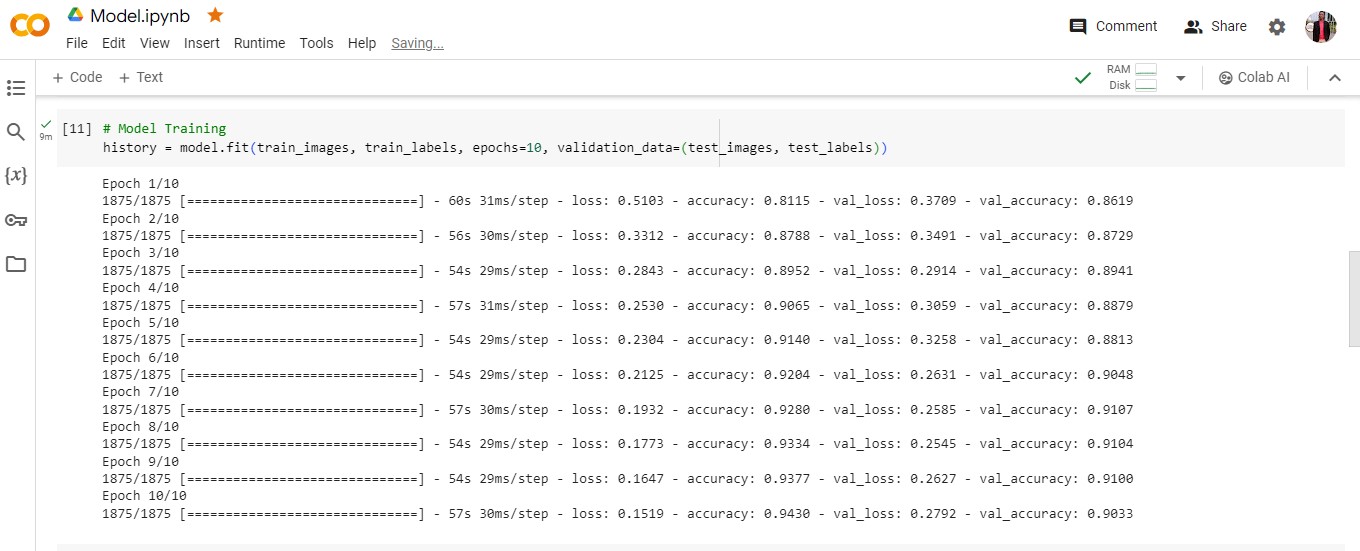
**2.4 Model Training**

During the model training phase, I conducted a training session on the training data, used a batch size of 32, and iterated the dataset over 10 epochs. This batch size was chosen to balance memory efficiency and computational speed, allowing the model to process an appropriate number of samples in each training iteration.

By training the model over 10 epochs, I ensured that the model had sufficient access to the entire training data set and could easily learn the complex patterns and features needed for accurate classification. After each training epoch, I performed validation on the test data to evaluate the model's performance on unknown examples.

This validation step served as an important checkpoint to monitor the model's generalization ability and detect signs of overfitting or underfitting. Evaluating the model's performance on a different dataset provided insight into its robustness and ability to make accurate predictions based on new and unseen data.

Training of the program, combined with rigorous validation steps, ensured the development of a reliable convolutional neural network that can effectively classify fashion images from the Fashion MNIST dataset.



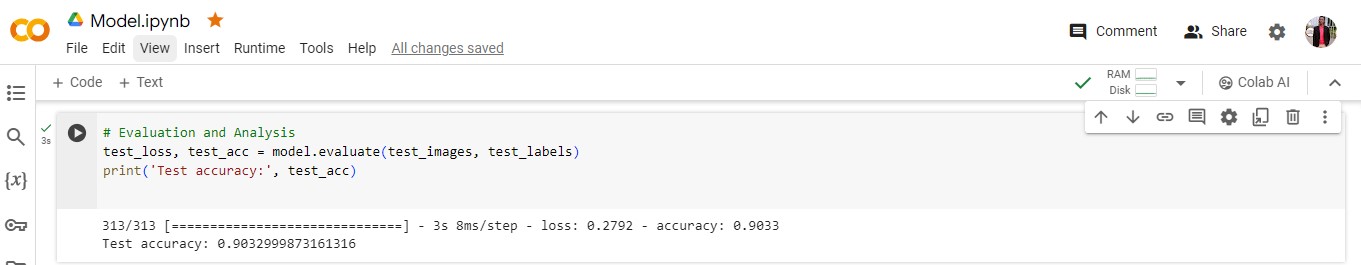
**3.0 Results**

The results of the model training and evaluation on the Fashion MNIST dataset are as follows. The model achieved a training accuracy of 0.9032999873161316, indicating the proportion of correctly classified images within the training dataset.

The validation accuracy of the model on the test data was 0.9033, reflecting its ability to generalize and make accurate predictions on unseen images.

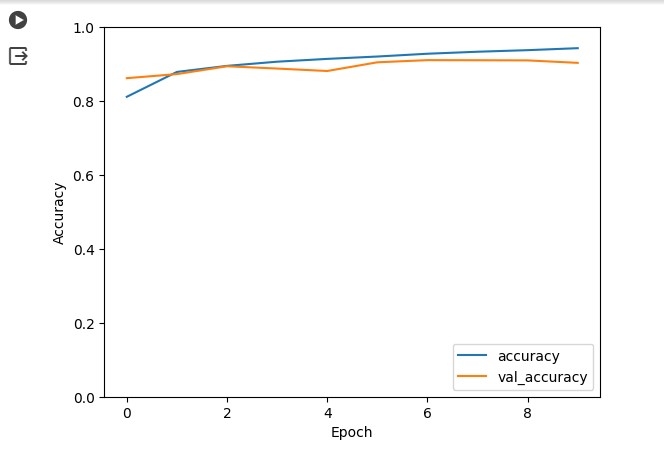
The training loss, which measures the discrepancy between the predicted and actual labels during training, was 0.2792. A lower training loss indicates better convergence of the model during training.

These performance metrics provide insights into the efficacy of the trained convolutional neural network in classifying fashion images from the Fashion MNIST dataset. A high training accuracy coupled with a comparable validation accuracy indicates that the model has learned meaningful representations from the training data and can generalize well to unseen examples. Additionally, monitoring the training and validation loss over epochs helps in assessing the convergence and generalization ability of the model.

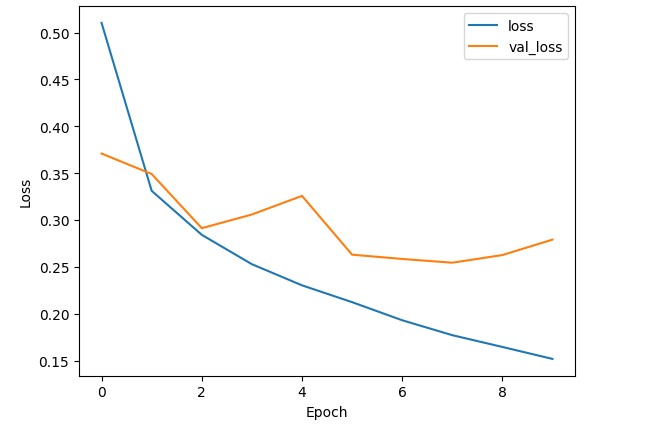


Graphs plotting, including training and validation accuracy curves and loss curves over epochs, serve as indispensable tools for assessing the training dynamics and performance of the convolutional neural network (CNN) model. These visualizations offer a comprehensive overview of the model's learning process, highlighting trends such as convergence, overfitting, or underfitting. By scrutinizing the trajectories of accuracy and loss values over successive epochs, we can discern patterns indicative of the model's efficacy in learning meaningful representations from the data and making accurate predictions on unseen examples. Additionally, comparing the training and validation curves enables us to evaluate the model's generalization ability and detect potential issues such as overfitting or insufficient model capacity. Thus, the graphs plotting constitute a vital component of the model evaluation process, facilitating informed decision-making and guiding further iterations and refinements to enhance the model's performance.

***Graph 1. Plotted graph of accuracy over Epoch***



***Graph 2. Plotted graph showing loss over Epoch***



**4.0 Discussion and Future Work**

While the trained convolutional neural network (CNN) demonstrates promising performance in accurately classifying fashion images from the Fashion MNIST dataset, there exist opportunities for further enhancement and refinement. To bolster the model's efficacy and robustness, several potential improvements could be explored:

Fine-tuning hyperparameters, such as learning rate, batch size, and dropout rate, is critical for optimizing model convergence and generalization. Conducting systematic hyperparameter sweeps or utilizing automated techniques like grid search or random search can help identify optimal configurations.

Regularization techniques, such as L2 regularization or dropout, play a crucial role in preventing overfitting by imposing constraints on the model's parameters during training. These techniques encourage parameter sparsity and reduce inter-dependencies between neurons, leading to smoother decision boundaries and better generalization to unseen data.

Ensemble learning methods, such as bagging or boosting, leverage the collective wisdom of multiple diverse models to achieve superior performance. By aggregating predictions from multiple base models trained on different subsets of data, ensemble methods mitigate individual model biases and variance, resulting in more robust and accurate predictions.