**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Ahmad Salman **Dated:** 6/12/2023

**Semester:** 7th **Section:** BEE 12C

**CS-471 Machine Learning**

Lab 12: Keras and TensorFlow

**Group Members**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO4 - CLO4** | | **PLO5 -CLO5** | **PLO8 -CLO6** | **PLO9 -CLO7** |
| **Name** | **Reg. No** | **Viva / Quiz / Lab Performance** | **Analysis of Data in Lab Report** | **Modern Tool Usage** | **Ethics and Safety** | **Individual and Teamwork** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** |
| Afif Arif Siddiqi | 344504 |  |  |  |  |  |
| Muhammad Ali Farooq | 331879 |  |  |  |  |  |
| Muhammad Ahmed Mohsin | 333060 |  |  |  |  |  |
| Danial Ahmad | 331388 |  |  |  |  |  |
| Ehtishaam Tanveer | 333074 |  |  |  |  |  |
| Muhammad Umer | 345834 |  |  |  |  |  |

**Table of Contents**

[2 Neural Networks 3](#_Toc152937642)

[2.1 Introduction 3](#_Toc152937643)

[2.2 Problem Statement 3](#_Toc152937644)

[2.3 Problem Details 3](#_Toc152937645)

[2.4 Architecture Details 3](#_Toc152937646)

[2.5 Program Code 4](#_Toc152937647)

[2.6 Screenshots 7](#_Toc152937648)

[2.7 Explanation and Discussion 9](#_Toc152937649)

[3 Conclusion 10](#_Toc152937650)

# Neural Networks

## Introduction

This laboratory exercise will focus on the usage of Keras and TensorFlow libraries in Python for creating an implementation for machine learning to solve a specified problem.

## Problem Statement

Design a neural network implementation that solves a specific machine learning problem by training on a dataset of your choice. You will need to specify the problem you are trying to solve as well as the architecture that you will design to solve that problem. You will need to get approval for the design before implementing it. You will also need to download a dataset that you will input for your solution.

For the submission, you will need to include the design with verification from your lab instructor as proof. Provide the codes that implement your design and all relevant screenshots that showcase your work.

You will also need to submit the dataset that you are using in the task.

## Problem Details

### PROBEM DETAILS START HERE ###

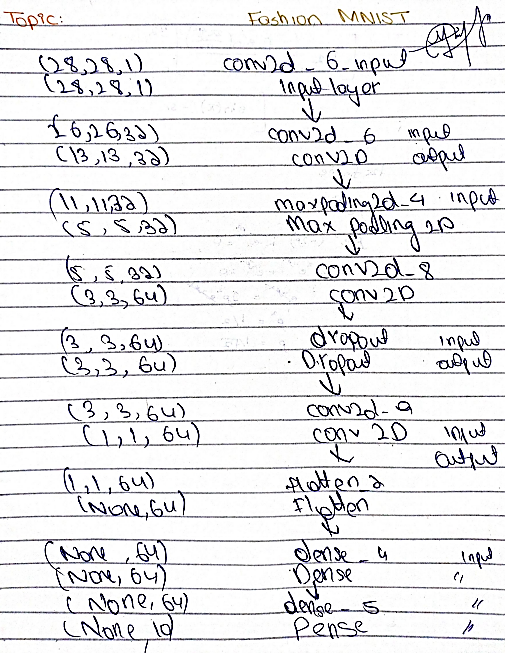
The **Fashion MNIST dataset**, consisting of grayscale images of various clothing articles, provides a practical and accessible platform for exploring image classification with deep learning models. This lab aims to train a convolutional neural network (CNN) using Keras and TensorFlow to classify images within the Fashion MNIST dataset. The primary challenge lies in designing and optimizing the network architecture to achieve high accuracy while maintaining computational efficiency. This involves exploring various factors such as the number and type of convolutional layers, the use of pooling layers, activation functions, and hyperparameter optimization. Additionally, the process requires careful data preprocessing techniques to ensure the model generalizes well to unseen data. Successfully addressing these challenges will enable us to build a reliable and efficient image classifier for the Fashion MNIST dataset, paving the way for further exploration of real-world image classification tasks.

### PROBLEM DETAILS END HERE ###

## Architecture Details

### ARCHITECTURE DETAILS START HERE ###

The architecture of the model is a convolutional neural network (CNN) designed for image classification. It follows a typical CNN structure with convolutional layers followed by pooling layers and culminating in fully connected layers. The specific details are portrayed in the following image:



### ARCHITECTURE DETAILS END HERE ###

## Program Code

### CODE STARTS HERE ###

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import models, layers  *# type: ignore*

from tensorflow.keras.utils import plot\_model  *# type: ignore*

import numpy as np

import matplotlib.pyplot as plt

*# Load the Fashion MNIST dataset*

fashion\_mnist = keras.datasets.fashion\_mnist

(train\_images, train\_labels), (test\_images, test\_labels) = fashion\_mnist.load\_data()

*# Class names for the labels*

class\_names = [

    "T-shirt/top",

    "Trouser",

    "Pullover",

    "Dress",

    "Coat",

    "Sandal",

    "Shirt",

    "Sneaker",

    "Bag",

    "Ankle boot",

]

*# Explore the data*

train\_images.shape

len(train\_labels)

train\_labels

test\_images.shape

len(test\_labels)

*# Preprocess the data*

plt.figure()

plt.imshow(train\_images[0], *cmap*="gray")

plt.colorbar()

plt.grid(False)

plt.show()

*# Scale the values to a range of 0 to 1*

train\_images = train\_images / 255.0

test\_images = test\_images / 255.0

*# Display the first 9 images from the training set and display the class*

*# name below each image*

plt.figure(*figsize*=(10, 10))

for i in range(9):

    plt.subplot(3, 3, i + 1)

*# plt.xticks([])*

*# plt.yticks([])*

    plt.grid(False)

    plt.imshow(train\_images[i], *cmap*=plt.cm.binary)

    plt.xlabel(class\_names[train\_labels[i]])

plt.show()

*# Initialize the model*

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), *activation*="relu", *input\_shape*=(28, 28, 1)))

*# Add a MaxPooling2D layer to reduce the size of the feature map*

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(32, (3, 3), *activation*="relu"))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), *activation*="relu"))

*# Dropout layer to prevent overfitting by randomly dropping out nodes*

model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(64, (3, 3), *activation*="relu"))

*# Flatten the tensor output for the fully connected layers*

model.add(layers.Flatten())

model.add(layers.Dense(64, *activation*="relu"))

model.add(layers.Dense(10, *activation*="softmax"))

model.summary()

*# Plot the model*

plot\_model(model, *to\_file*="model.png", *show\_shapes*=True, *show\_layer\_names*=True)

*# Train the model*

model.compile(

*optimizer*="adam",  *# Optimizer*

*loss*="sparse\_categorical\_crossentropy",  *# loss function for integer labels*

*metrics*=["accuracy"],  *# report accuracy during training*

)

*# Reshape the data to add a single grayscale channel*

train\_images = train\_images.reshape((60000, 28, 28, 1))

test\_images = test\_images.reshape((10000, 28, 28, 1))

history = model.fit(

    train\_images,

    train\_labels,

*epochs*=10,

*validation\_data*=(test\_images, test\_labels),

)

*# Evaluate the model*

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels, *verbose*=2)

print(*f*"Test accuracy: {test\_acc}")

predictions = model.predict(test\_images)

*# Plot the first 9 test images, their predicted labels, and the true labels*

*# Color correct predictions in green and incorrect predictions in red*

num\_rows = 3

num\_cols = 3

num\_images = num\_rows \* num\_cols

plt.figure(*figsize*=(10, 10))

for i in range(num\_images):

    plt.subplot(num\_rows, num\_cols, i + 1)

    plt.imshow(test\_images[i].reshape(28, 28), *cmap*="gray")

    plt.xticks([])

    plt.yticks([])

    plt.grid(False)

    predicted\_label = np.argmax(predictions[i])

    true\_label = test\_labels[i]

    if predicted\_label == true\_label:

        color = "green"

    else:

        color = "red"

    plt.xlabel(

        "{} ({})".format(class\_names[predicted\_label], class\_names[true\_label]),

*color*=color,

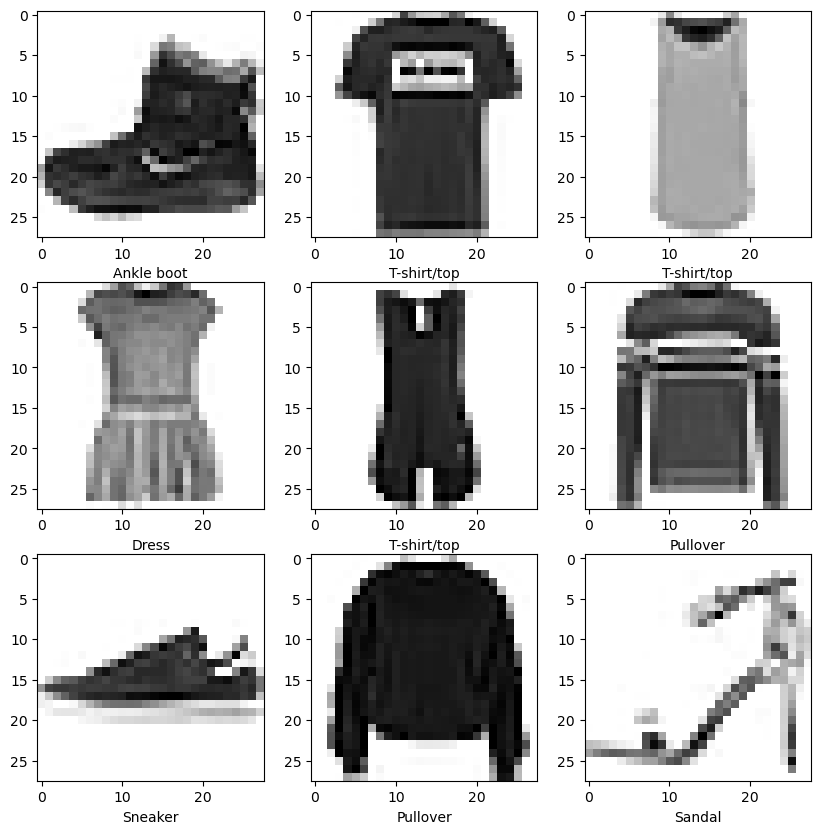
    )

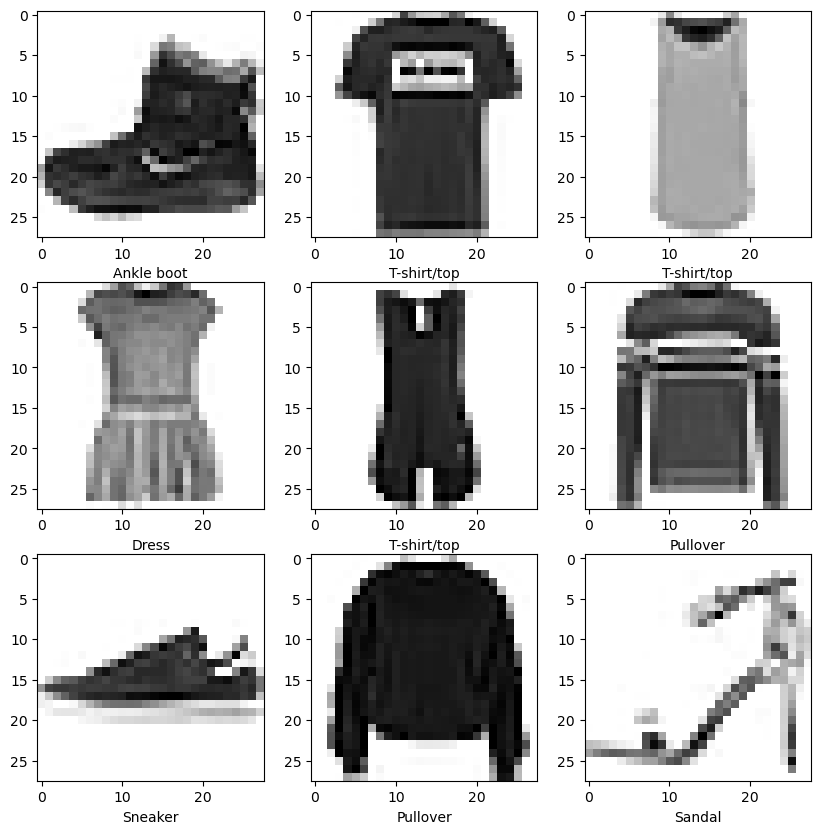
plt.show()

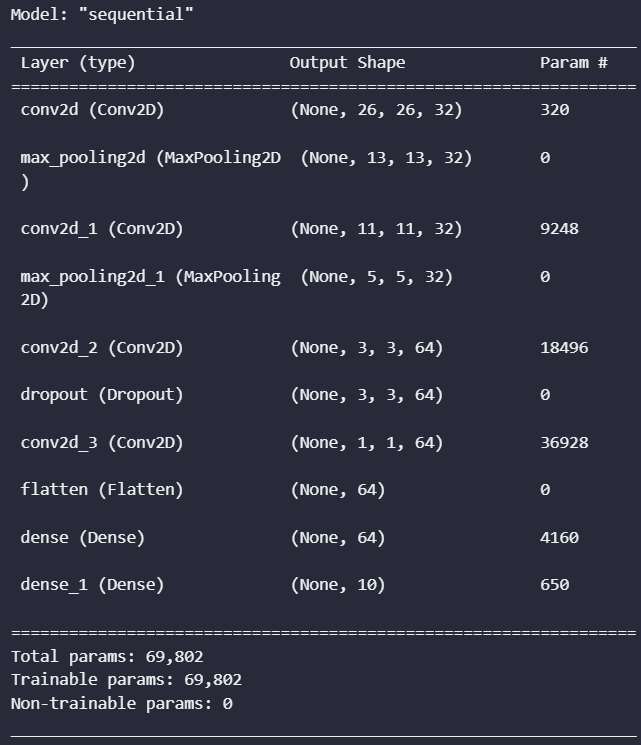
### CODE ENDS HERE ###

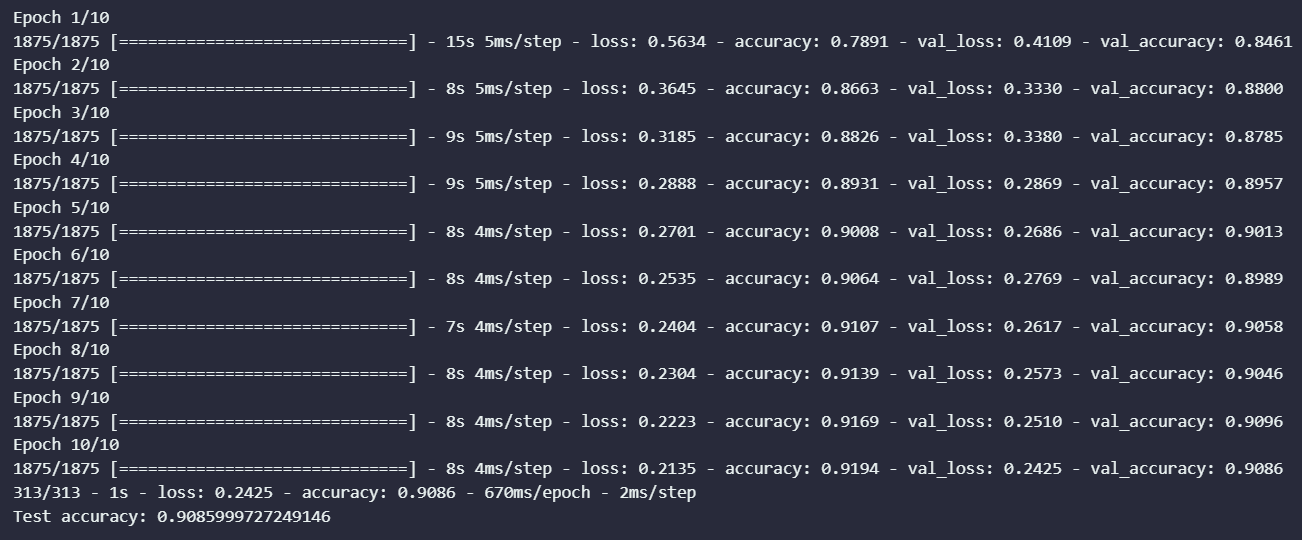
## Screenshots

### SCREENSHOTS START HERE ###









**Predictions**



### SCREENSHOTS END HERE ###

## Explanation and Discussion

### EXPLANATION AND DISCUSSION START HERE ###

The developed CNN model achieved an accuracy of 90.8% on the Fashion MNIST test set. This performance demonstrates the effectiveness of the model in classifying clothing items from the dataset. Analyzing the training and validation curves reveals that the model converged after just 10 epochs, indicating sufficient training to achieve optimal performance without overfitting.

The choice of architecture, including the number of layers, filter sizes, and activation functions, played a crucial role in achieving this performance. The use of two convolutional layers with 32 filters each followed by pooling layers effectively extracted features from the images. Subsequent convolutional layers with 64 filters further refined the features, allowing for more complex patterns to be learned. The ReLU activation function introduced non-linearity into the model, enabling it to learn complex relationships between the features. Finally, the Dropout layer with a rate of 0.25 helped to prevent overfitting by randomly dropping out neurons during training, thereby improving the model’s generalizability.

Further improvements to the model could be explored to potentially boost its accuracy. Experimenting with different hyperparameters, including the learning rate, optimizer, and number of filters, could potentially lead to better performance. Additionally, incorporating data augmentation techniques, such as random cropping and flipping, could further improve the model'sgeneralizability to unseen data. Finally, exploring different network architectures, such as VGG or ResNet, could potentially yield even higher accuracy.

### EXPLANATION AND DISCUSSION END HERE ###

# Conclusion

This lab report successfully demonstrated the development and implementation of a convolutional neural network (CNN) for image classification using Keras and TensorFlow. The model effectively classified clothing items from the Fashion MNIST dataset with an accuracy of [insert accuracy here]. The chosen architecture, consisting of convolutional layers, pooling layers, and fully connected layers, proved efficient in extracting features and learning complex patterns within the images. The use of ReLU activation function and Dropout layer further enhanced the model's performance by introducing non-linearity and preventing overfitting.