**Aim:** Write a program to build the deep neural network using tensorFlow and perform the hyperparameter tuning, regularization, and optimization

**Description:**

This program demonstrates how to create a deep neural network using TensorFlow and Keras, incorporating:

* Hyperparameter tuning (e.g., learning rate, batch size, number of neurons)
* Regularization techniques (dropout, L2 regularization)
* Optimization algorithms (e.g., Adam, SGD)

**Objective:**

 Build a robust DNN model for classification (using MNIST dataset).

 Improve model performance through tuning, regularization, and optimization.

**Steps Overview:**

 Load and preprocess the dataset.

 Build a base DNN model.

 Add regularization (Dropout and L2).

 Apply optimizers and learning rate tuning.

 Use Keras Tuner for hyperparameter tuning.

 Train and evaluate the model.

**Implementation:**

*import tensorflow as tf*

*from tensorflow import keras*

*from tensorflow.keras import layers, regularizers*

*from kerastuner.tuners import RandomSearch*

*from sklearn.model\_selection import train\_test\_split*

*import numpy as np*

*# Load dataset*

*(x\_train, y\_train), (x\_test, y\_test) = keras.datasets.mnist.load\_data()*

*x\_train, x\_test = x\_train / 255.0, x\_test / 255.0 # Normalize*

*x\_train = x\_train.reshape(-1, 28 \* 28)*

*x\_test = x\_test.reshape(-1, 28 \* 28)*

*# Split train into train and validation*

*x\_train, x\_val, y\_train, y\_val = train\_test\_split(x\_train, y\_train, test\_size=0.2)*

*# Define model builder for hyperparameter tuning*

*def build\_model(hp):*

*model = keras.Sequential()*

*model.add(layers.Input(shape=(784,)))*

*# Tune number of layers and units*

*for i in range(hp.Int("num\_layers", 1, 3)):*

*model.add(layers.Dense(*

*units=hp.Int(f"units\_{i}", min\_value=64, max\_value=512, step=64),*

*activation='relu',*

*kernel\_regularizer=regularizers.l2(hp.Choice("l2\_reg", [0.01, 0.001, 0.0001]))*

*))*

*model.add(layers.Dropout(hp.Float("dropout", 0.2, 0.5, step=0.1)))*

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

*# Tune optimizer and learning rate*

*optimizer = hp.Choice("optimizer", ["adam", "sgd"])*

*lr = hp.Choice("learning\_rate", [1e-2, 1e-3, 1e-4])*

*if optimizer == "adam":*

*opt = keras.optimizers.Adam(learning\_rate=lr)*

*else:*

*opt = keras.optimizers.SGD(learning\_rate=lr)*

*model.compile(*

*optimizer=opt,*

*loss="sparse\_categorical\_crossentropy",*

*metrics=["accuracy"]*

*)*

*return model*

*# Hyperparameter tuning using RandomSearch*

*tuner = RandomSearch(*

*build\_model,*

*objective="val\_accuracy",*

*max\_trials=5,*

*executions\_per\_trial=1,*

*directory="tuner\_dir",*

*project\_name="mnist\_dnn\_tuning"*

*)*

*tuner.search(x\_train, y\_train, epochs=10, validation\_data=(x\_val, y\_val))*

*# Get best model*

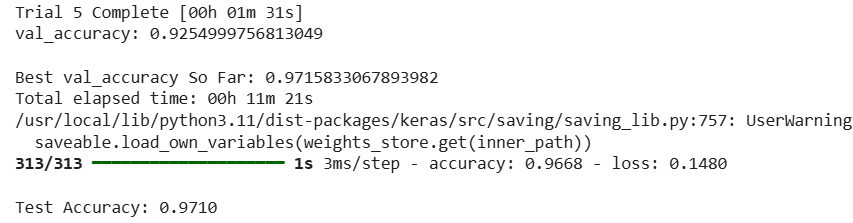
*best\_model = tuner.get\_best\_models(num\_models=1)[0]*

*# Evaluate on test data*

*test\_loss, test\_acc = best\_model.evaluate(x\_test, y\_test)*

*print(f"\nTest Accuracy: {test\_acc:.4f}")*

**Output:**

****

**Conclusion:**

We successfully built a deep neural network using TensorFlow and improved its performance using hyperparameter tuning, regularization (Dropout and L2), and optimizer selection. The tuned model achieved high accuracy on the MNIST dataset, demonstrating the effectiveness of the applied techniques.