Venkata Sri Satya Sai Kiran Ayyagari - 11713213

Step - 1 - Preprocessing

```
import numpy as sk
import tensorflow as sktf
from tensorflow.keras.datasets import fashion_mnist
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.utils import shuffle
(x_train,y_train),(x_test,y_test)=fashion_mnist.load_data()
skx_data=sk.concatenate((x_train,x_test))
sky_data=sk.concatenate((y_train,y_test))
skx_data=skx_data.astype('float32')/255.0
skx_data=skx_data.reshape(-1,28,28,1)
skx_data,sky_data=shuffle(skx_data,sky_data,random_state=42)
X_train,X_test,y_train,y_test=train_test_split(skx_data,sky_data,test_
size=0.3,random_state=42)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

Step - 2 - Models

```
from tensorflow.keras import layers,models
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
def build_cnn():
  model=models.Sequential([
      layers.Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)),
      layers.MaxPooling2D((2,2)),
      layers.Conv2D(64,(3,3),activation='relu'),
      layers.MaxPooling2D((2,2)),
      layers.Flatten(),
      layers.Dense(64,activation='relu'),
      layers.Dense(10,activation='softmax')
      ])

model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
```

```
metrics=['accuracy'])
  return model
skcnn=build_cnn()
skcnn.fit(X_train,y_train,epochs=10,validation_split=0.2,batch_size=64
)
skcnn_preds=skcnn.predict(X_test)
skcnn_preds_classes=sk.argmax(skcnn_preds,axis=1)

skX_train_flat=X_train.reshape(-1,28*28)
skX_test_flat=X_test.reshape(-1,28*28)
skscaler=StandardScaler()
skX_train_scaled=skscaler.fit_transform(skX_train_flat)
skX_test_scaled=skscaler.transform(skX_test_flat)

sksvm=SVC(kernel='rbf',probability=True)
sksvm.fit(skX_train_scaled,y_train)
sksvm_preds_proba=sksvm.predict_proba(skX_test_scaled)
sksvm_preds=sksvm.predict(skX_test_scaled)

skrf=RandomForestClassifier(n_estimators=100)
skrf.fit(skX_train_flat,y_train)
skrf_preds_proba=skrf.predict_proba(skX_test_flat)
skrf_preds=skrf.predict(skX_test_flat)

skensemble_proba=(sksvm_preds_proba+skrf_preds_proba+skcnn_preds)/3
skensemble_preds=sk.argmax(skensemble_proba,axis=1)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/10
613/613 ━━━━━━━━━━━━━━━━━━━━ 40s 62ms/step - accuracy: 0.7129 - loss:
0.8110 - val_accuracy: 0.8661 - val_loss: 0.3774
Epoch 2/10
613/613 ━━━━━━━━━━━━━━━━━━━━ 33s 54ms/step - accuracy: 0.8651 - loss:
0.3760 - val_accuracy: 0.8789 - val_loss: 0.3305
Epoch 3/10
613/613 ━━━━━━━━━━━━━━━━━━━━ 42s 55ms/step - accuracy: 0.8835 - loss:
0.3173 - val_accuracy: 0.8871 - val_loss: 0.3074
Epoch 4/10
613/613 ━━━━━━━━━━━━━━━━━━━━ 41s 55ms/step - accuracy: 0.8949 - loss:
0.2864 - val_accuracy: 0.8918 - val_loss: 0.2893
Epoch 5/10
613/613 ━━━━━━━━━━━━━━━━━━━━ 40s 54ms/step - accuracy: 0.9045 - loss:
0.2601 - val_accuracy: 0.8981 - val_loss: 0.2813
```

```
Epoch 6/10
613/613 ━━━━━━━━━━━━━━━━━━━━ 40s 53ms/step - accuracy: 0.9176 - loss:
0.2321 - val_accuracy: 0.8973 - val_loss: 0.2760
Epoch 7/10
613/613 ━━━━━━━━━━━━━━━━━━━━ 44s 57ms/step - accuracy: 0.9205 - loss:
0.2184 - val_accuracy: 0.8959 - val_loss: 0.2944
Epoch 8/10
613/613 ━━━━━━━━━━━━━━━━━━━━ 39s 55ms/step - accuracy: 0.9270 - loss:
0.2004 - val_accuracy: 0.9056 - val_loss: 0.2628
Epoch 9/10
613/613 ━━━━━━━━━━━━━━━━━━━━ 32s 52ms/step - accuracy: 0.9341 - loss:
0.1784 - val_accuracy: 0.9058 - val_loss: 0.2740
Epoch 10/10
613/613 ━━━━━━━━━━━━━━━━━━━━ 44s 57ms/step - accuracy: 0.9413 - loss:
0.1668 - val_accuracy: 0.9073 - val_loss: 0.2685
657/657 ━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step
```

Step - 3 - Evaluation

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
def evaluate_model(name,y_true,y_pred):
  print(f"\n{name} Evaluation:")
  print("Accuracy:",accuracy_score(y_true,y_pred))

print("Precision:",precision_score(y_true,y_pred,average='weighted'))
  print("Recall:",recall_score(y_true,y_pred,average='weighted'))
  print("F1 Score:",f1_score(y_true,y_pred,average='weighted'))
evaluate_model("CNN",y_test,skcnn_preds_classes)
evaluate_model("SVM",y_test,sksvm_preds)
evaluate_model("Random Forest",y_test,skrf_preds)
evaluate_model("Ensemble",y_test,skensemble_preds)


CNN Evaluation:
Accuracy: 0.9078095238095238
Precision: 0.9089163327295683
Recall: 0.9078095238095238
F1 Score: 0.9079932595034714

SVM Evaluation:
Accuracy: 0.8917619047619048
Precision: 0.8909199440577197
Recall: 0.8917619047619048
F1 Score: 0.8910984346128933

Random Forest Evaluation:
Accuracy: 0.8841428571428571
Precision: 0.8828581146035642
```

```
Recall: 0.8841428571428571
F1 Score: 0.8828232772080516

Ensemble Evaluation:
Accuracy: 0.9120952380952381
Precision: 0.9116952149682691
Recall: 0.9120952380952381
F1 Score: 0.9117702477060698
```