In [4]:
```
import pandas as pd
import numpy as np
from tensorflow.keras.datasets import fashion_mnist
import tensorflow as tf
from tensorflow.keras.layers import Conv2D,Dense, Flatten, Dropout, BatchNorma
from tensorflow.keras.models import Sequential
```

In [5]:
```
(X_train,Y_train),(X_test,Y_test)=fashion_mnist.load_data()
X_train,X_test = X_train/255.0,X_test/255.0

X_train = X_train.reshape(X_train.shape[0],-1)
X_test = X_test.reshape(X_test.shape[0],-1)
```

MLP model

In [7]:
```
model1_mlp = Sequential([Dense(128,activation='relu',input_shape=(784,)),
                         Dropout(0.2),
                         Dense(10,activation='softmax')
                        ])

model1_mlp.compile(optimizer ='adam',loss='sparse_categorical_crossentropy',me
model1_mlp.fit(X_train,Y_train,epochs=5,validation_data=(X_test,Y_test))
tst_loss,tst_accuracy_mlp =model1_mlp.evaluate(X_test,Y_test)
print("Accuracy of MLP model is:",tst_accuracy_mlp*100)
```

```
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 816us/step - accuracy: 0.7637 - loss: 0.6672
- val_accuracy: 0.8344 - val_loss: 0.4522
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 777us/step - accuracy: 0.8507 - loss: 0.4111
- val_accuracy: 0.8605 - val_loss: 0.3928
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 821us/step - accuracy: 0.8657 - loss: 0.3678
- val_accuracy: 0.8676 - val_loss: 0.3687
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 837us/step - accuracy: 0.8740 - loss: 0.3442
- val_accuracy: 0.8667 - val_loss: 0.3614
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 794us/step - accuracy: 0.8811 - loss: 0.3231
- val_accuracy: 0.8735 - val_loss: 0.3567
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 308us/step - accuracy: 0.8768 - loss: 0.3459
Accuracy of MLP model is: 87.34999895095825
```

In [8]:
```
X_test = X_test.reshape(-1, 784)
model1_mlp_predict = model1_mlp.predict(X_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 320us/step
```

In [9]:
```
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_sco
mlp_predict = np.argmax(model1_mlp_predict, axis=1)
precision_mlp = precision_score(Y_test, mlp_predict, average='weighted')
recall_mlp=recall_score(Y_test, mlp_predict, average='weighted')
f1_mlp = f1_score(Y_test, mlp_predict, average='weighted')

print("MLP Precision score is:", precision_mlp*100)
```

```
print("MLP Recall score is:",recall_mlp*100)
print("MLP F1 score is:",f1_mlp*100)
```

```
MLP Precision score is: 87.45829357439729
MLP Recall score is: 87.35000000000001
MLP F1 score is: 87.31504155952774
```

CNN model

In [10]:
```
(X_train, Y_train), (X_test, Y_test) = fashion_mnist.load_data()
X_train, X_test = X_train / 255.0, X_test / 255.0

X_train = np.expand_dims(X_train, axis=-1)
X_test = np.expand_dims(X_test, axis=-1)
```

In [11]:
```
model2_cnn = Sequential([Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1
                     BatchNormalization(),
                     MaxPooling2D((2,2)),
                     Dropout(0.25),

                     Conv2D(32,(3,3),activation='relu'),
                     BatchNormalization(),
                     MaxPooling2D((2,2)),
                     Dropout(0.25),

                     Flatten(),
                     Dense(256,activation='relu'),
                     BatchNormalization(),
                     Dropout(0.5),
                     Dense(10,activation='softmax')])


model2_cnn.compile(optimizer='adam',loss='sparse_categorical_crossentropy',met
model2_cnn.fit(X_train,Y_train,epochs=5,validation_data=(X_test,Y_test))

tst_loss_cnn,tst_accuracy_cnn =model2_cnn.evaluate(X_test,Y_test)
print("Accuracy of CNN model is:",tst_accuracy_cnn*100)
```

```
Epoch 1/5
/Applications/anaconda3/lib/python3.11/site-packages/keras/src/layers/convolut
ional/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(shap
e)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 27s 14ms/step - accuracy: 0.7492 - loss: 0.7529
- val_accuracy: 0.8574 - val_loss: 0.3866
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 29s 15ms/step - accuracy: 0.8539 - loss: 0.4047
- val_accuracy: 0.8826 - val_loss: 0.3269
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 29s 15ms/step - accuracy: 0.8677 - loss: 0.3599
- val_accuracy: 0.8875 - val_loss: 0.3046
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 28s 15ms/step - accuracy: 0.8761 - loss: 0.3398
- val_accuracy: 0.8881 - val_loss: 0.2904
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 28s 15ms/step - accuracy: 0.8821 - loss: 0.3176
- val_accuracy: 0.8996 - val_loss: 0.2695
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9015 - loss: 0.2729
Accuracy of CNN model is: 89.96000289916992
```

In [13]:
```
model2_cnn_pred = model2_cnn.predict(X_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
```

In [14]:
```
model2_cnn_predict = np.argmax(model2_cnn_pred, axis=1)
precision_cnn = precision_score(Y_test, model2_cnn_predict, average='weighted'
recall_cnn=recall_score(Y_test, model2_cnn_predict, average='weighted')
f1_cnn = f1_score(Y_test, model2_cnn_predict, average='weighted')

print("CNN model Precision score is:", precision_cnn*100)
print("CNN model Recall score is:",recall_cnn*100)
print("CNN model F1 score is:",f1_cnn*100)
```

```
CNN model Precision score is: 89.92893969788803
CNN model Recall score is: 89.96
CNN model F1 score is: 89.84775527869068
```

Simple RNN

In [15]:
```
(X_train, Y_train), (X_test, Y_test) = fashion_mnist.load_data()
X_train, X_test = X_train.astype('float32') / 255.0, X_test.astype('float32') /

X_train = X_train.reshape(-1, 28, 28)
X_test = X_test.reshape(-1, 28, 28)
```

In [16]:
```
model3_rnn = Sequential([SimpleRNN(128, input_shape=(28,28), return_sequences='
                    SimpleRNN(64),
                    Dense(10,activation='softmax')])

model3_rnn.compile(optimizer='adam',loss='sparse_categorical_crossentropy',met
model3_rnn.fit(X_train,Y_train,epochs=5,validation_data=(X_test,Y_test))

tst_loss_rnn,tst_acc_rnn = model3_rnn.evaluate(X_test,Y_test)
print("Accuracy of Simple RNN model is:",tst_acc_rnn*100)
```

```
Epoch 1/5
/Applications/anaconda3/lib/python3.11/site-packages/keras/src/layers/rnn/rnn.
py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a la
yer. When using Sequential models, prefer using an `Input(shape)` object as th
e first layer in the model instead.
  super().__init__(**kwargs)
```

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 13s 7ms/step - accuracy: 0.7056 - loss: 0.8081
- val_accuracy: 0.7826 - val_loss: 0.5932
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 7ms/step - accuracy: 0.8035 - loss: 0.5473
- val_accuracy: 0.8204 - val_loss: 0.4946
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 6ms/step - accuracy: 0.8239 - loss: 0.4942
- val_accuracy: 0.8241 - val_loss: 0.5064
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 6ms/step - accuracy: 0.8295 - loss: 0.4716
- val_accuracy: 0.8181 - val_loss: 0.4906
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 6ms/step - accuracy: 0.8317 - loss: 0.4654
- val_accuracy: 0.8392 - val_loss: 0.4533
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8443 - loss: 0.4472
Accuracy of Simple RNN model is: 83.92000198364258
```

In [17]:
```
model3_rnn_pred = model3_rnn.predict(X_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

In [18]:
```
model3_rnn_predict = np.argmax(model3_rnn_pred, axis=1)
precision_rnn = precision_score(Y_test, model3_rnn_predict, average='weighted'
recall_rnn=recall_score(Y_test, model3_rnn_predict, average='weighted')
f1_rnn = f1_score(Y_test, model3_rnn_predict, average='weighted')

print("Simple RNN Precision score is:", precision_rnn*100)
print("Simple RNN Recall score is:",recall_rnn*100)
print("Simple RNN F1 score is:",f1_rnn*100)
```

```
Simple RNN Precision score is: 83.79454302032066
Simple RNN Recall score is: 83.91999999999999
Simple RNN F1 score is: 83.7507801642537
```

Committe of models

In [19]:
```
ensemble_pred = (model1_mlp_predict + model2_cnn_pred + model3_rnn_pred) / 3

ensemble_labels = np.argmax(ensemble_pred, axis=1)
ensemble_acc = np.mean(ensemble_labels == Y_test)
print("Accuracy of the ensemble model by averaging is:", ensemble_acc*100)
```

```
Accuracy of the ensemble model by averaging is: 89.21
```

In [21]:
```
prec_ensemble = precision_score(Y_test, ensemble_labels, average='weighted')
recall_ensemble=recall_score(Y_test, ensemble_labels, average='weighted')
f1_ensemble = f1_score(Y_test, ensemble_labels, average='weighted')


print("Precision score of ensemble(avg):", prec_ensemble*100)
print("Recall score of ensemble(avg):",recall_ensemble*100)
print("F1 score of ensemble(avg):",f1_ensemble*100)
```

```
Precision score of ensemble(avg): 89.157351235714
Recall score of ensemble(avg): 89.21
F1 score of ensemble(avg): 89.08995710646495
```

Committee of models by multiplication

In [22]:
```
ensemble_predictions = model1_mlp_predict * model2_cnn_pred * model3_rnn_pred
ensemble_labels = np.argmax(ensemble_predictions, axis=1)
ensemble_accuracy = np.mean(ensemble_labels == Y_test)
print("Accuracy of the ensemble model by multiplication is:", ensemble_accuracy
```

```
Accuracy of the ensemble model by multiplication is: 89.32
```

In [23]:
```
prec_ensemble1 = precision_score(Y_test, ensemble_labels, average='weighted')
recall_ensemble1=recall_score(Y_test, ensemble_labels, average='weighted')
f1_ensemble1 = f1_score(Y_test, ensemble_labels, average='weighted')


print("Precision score of ensemble(multiply):", prec_ensemble1*100)
print("Recall score of ensemble(multiply):",recall_ensemble1*100)
print("F1 score of ensemble(multiply):",f1_ensemble1*100)
```

```
Precision score of ensemble(multiply): 89.27451270609113
Recall score of ensemble(multiply): 89.32
F1 score of ensemble(multiply): 89.21987389558109
```

In [ ]:

In [ ]: