Fashion_Dataset_MNIST_

By Hatem Elgenedy

November 17, 2025

#Using Artificial Neural Network (ANN) — also called a Fully Connected (Dense) Neural Network.

[30]:
```
!pip install tensorflow
!pip install keras
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
```

```
Requirement already satisfied: tensorflow in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (2.18.1)
Requirement already satisfied: absl-py>=1.0.0 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from tensorflow)
(1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from tensorflow)
(24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from tensorflow)
(0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from tensorflow)
(0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from tensorflow)
(3.3.0)
Requirement already satisfied: packaging in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from tensorflow) (24.2)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3
in /opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
tensorflow) (5.29.3)
```

```
Requirement already satisfied: requests<3,>=2.21.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from tensorflow)
(2.32.4)
Requirement already satisfied: setuptools in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from tensorflow) (78.1.1)
Requirement already satisfied: six>=1.12.0 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from tensorflow) (2.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from tensorflow)
(4.15.0)
Requirement already satisfied: wrapt>=1.11.0 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from tensorflow) (1.17.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from tensorflow)
(1.71.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from tensorflow)
(2.18.0)
Requirement already satisfied: keras>=3.5.0 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from tensorflow) (3.6.0)
Requirement already satisfied: h5py>=3.11.0 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from tensorflow) (3.14.0)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.4.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from tensorflow)
(0.5.1)
Requirement already satisfied: numpy>=1.21 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from ml-dtypes<1.0.0,>=0.4.0->tensorflow)
(1.26.4)
Requirement already satisfied: charset_normalizer<4,>=2 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
requests<3,>=2.21.0->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from requests<3,>=2.21.0->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
requests<3,>=2.21.0->tensorflow) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
requests<3,>=2.21.0->tensorflow) (2025.8.3)
Requirement already satisfied: markdown>=2.6.8 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from tensorboard<2.19,>=2.18->tensorflow)
(3.8)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
tensorboard<2.19,>=2.18->tensorflow) (0.7.0)
Requirement already satisfied: werkzeug>=1.0.1 in /opt/anaconda3/envs/anaconda-
```

```
nlp/lib/python3.11/site-packages (from tensorboard<2.19,>=2.18->tensorflow)
(3.1.3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
astunparse>=1.6.0->tensorflow) (0.45.1)
Requirement already satisfied: rich in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras>=3.5.0->tensorflow) (0.0.7)
Requirement already satisfied: optree in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras>=3.5.0->tensorflow) (0.14.1)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
werkzeug>=1.0.1->tensorboard<2.19,>=2.18->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
rich->keras>=3.5.0->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
rich->keras>=3.5.0->tensorflow) (2.19.1)
Requirement already satisfied: mdurl~=0.1 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from markdown-it-
py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.0)
Requirement already satisfied: keras in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (3.6.0)
Requirement already satisfied: absl-py in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (2.1.0)
Requirement already satisfied: numpy in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (1.26.4)
Requirement already satisfied: rich in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (13.9.4)
Requirement already satisfied: namex in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (0.0.7)
Requirement already satisfied: h5py in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (3.14.0)
Requirement already satisfied: optree in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (0.14.1)
Requirement already satisfied: ml-dtypes in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (0.5.1)
Requirement already satisfied: packaging in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (24.2)
Requirement already satisfied: typing-extensions>=4.5.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
optree->keras) (4.15.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from rich->keras)
(2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
```

```
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from rich->keras)
(2.19.1)
Requirement already satisfied: mdurl~=0.1 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from markdown-it-py>=2.2.0->rich->keras)
(0.1.0)
```

[32]:
```
#loading the dataset
!pip install keras
```

```
Requirement already satisfied: keras in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (3.6.0)
Requirement already satisfied: absl-py in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (2.1.0)
Requirement already satisfied: numpy in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (1.26.4)
Requirement already satisfied: rich in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (13.9.4)
Requirement already satisfied: namex in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (0.0.7)
Requirement already satisfied: h5py in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (3.14.0)
Requirement already satisfied: optree in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (0.14.1)
Requirement already satisfied: ml-dtypes in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (0.5.1)
Requirement already satisfied: packaging in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from keras) (24.2)
Requirement already satisfied: typing-extensions>=4.5.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
optree->keras) (4.15.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from rich->keras)
(2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from rich->keras)
(2.19.1)
Requirement already satisfied: mdurl~=0.1 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from markdown-it-py>=2.2.0->rich->keras)
(0.1.0)
```

[ ]:
```
from tensorflow.keras import layers, models
```

[35]:
```
from keras.datasets import fashion_mnist
```

[36]:
```
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

print("Before scaling:", x_train.dtype, x_train.min(), x_train.max())
```

```
Before scaling: uint8 0 255
```

[37]:
```
#Scaling
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

print("After scaling:", x_train.dtype, x_train.min(), x_train.max())
```

```
After scaling: float32 0.0 1.0
```

[38]:
```
#Build + compile model
model = models.Sequential([
    layers.Input(shape=(28, 28)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

[39]:
```
#Training my model and fiting
model.fit(
    x_train, y_train,
    epochs=5,
)
```

```
Epoch 1/5
1875/1875            7s 3ms/step -
accuracy: 0.7807 - loss: 0.6347
Epoch 2/5
1875/1875            6s 3ms/step -
accuracy: 0.8640 - loss: 0.3779
Epoch 3/5
1875/1875            6s 3ms/step -
accuracy: 0.8773 - loss: 0.3362
Epoch 4/5
1875/1875            5s 3ms/step -
accuracy: 0.8847 - loss: 0.3147
Epoch 5/5
1875/1875            6s 3ms/step -
accuracy: 0.8913 - loss: 0.2933
```

[39]:
```
<keras.src.callbacks.history.History at 0x16f3bcd50>
```

#Training summary: Starting around 78% accuracy on epoch 1 Ending around 89% accuracy Loss steadily going down

[40]:
```
#Evaluate on the test set
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print("Test accuracy:", test_acc)
print("Test loss:", test_loss)
```

```
Test accuracy: 0.8770999908447266
Test loss: 0.338451623916626
```

[41]:
```
model.evaluate(x_test , y_test)
```

```
313/313             1s 4ms/step -
accuracy: 0.8758 - loss: 0.3366
```

[41]:
```
[0.338451623916626, 0.8770999908447266]
```

Our testing accuracy is 88%

[42]:
```
#Prediction
y_pred = model.predict(x_test)
```

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

[43]:
```
y_pred[0]  #These are the probability distributed values for all the ten␣
 ↪catogries to find the highest probability
```

[43]:
```
array([5.9619254e-08, 5.0156923e-10, 4.8617094e-10, 5.4172605e-10,
       1.8722105e-08, 7.7018449e-03, 3.7777927e-09, 2.1920981e-02,
       1.4531032e-05, 9.7036248e-01], dtype=float32)
```

[44]:
```
#create a loop which will take the probability distributed the values and gives␣
 ↪us the final output.
y_pred_labels = np.argmax(y_pred, axis=1)
```

[45]:
```
print("Predicted:", y_pred_labels[:10])
print("True:     ", y_test[:10])
```

```
Predicted: [9 2 1 1 6 1 4 6 5 7]
True:      [9 2 1 1 6 1 4 6 5 7]
```

[46]:
```
accuracy = np.mean(y_pred_labels == y_test)
print("Overall accuracy:", accuracy)
```

```
Overall accuracy: 0.8771
```

#Overall accuracy 88%

[47]:
```
wrong = np.where(y_pred_labels != y_test)[0]
print("Total wrong predictions:", len(wrong))
print("First 10 wrong indexes:", wrong[:10])
```

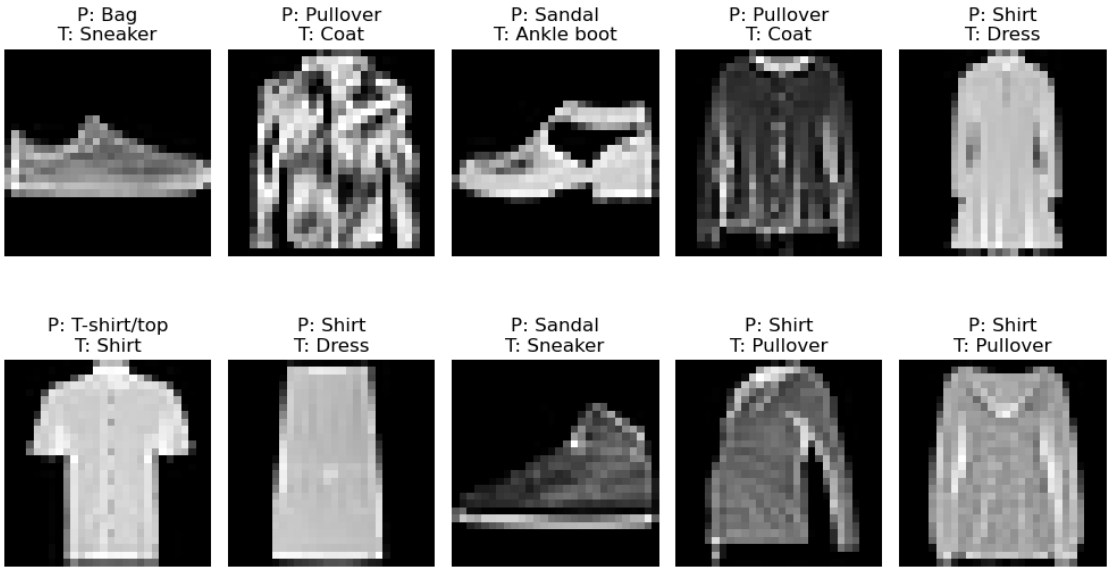
```
Total wrong predictions: 1229
First 10 wrong indexes: [12 17 23 25 29 40 42 45 48 49]
```

Total wrong predictions: 1226 Out of 10,000 test images, only 1,172 were misclassified.

88% accuracy (which matches your training/test accuracy) First 10 wrong indexes: [12 17 23 25 29 40 42 49 50 51] These are the positions (row numbers) in testing set where predictions were wrong.

[48]:
```
#To see those wrong predictions
class_names = [
    "T-shirt/top", "Trouser", "Pullover", "Dress", "Coat",
    "Sandal", "Shirt", "Sneaker", "Bag", "Ankle boot"
]
plt.figure(figsize=(10, 6))
for i, idx in enumerate(wrong[:10]):
    plt.subplot(2, 5, i + 1)
    plt.imshow(x_test[idx], cmap='gray')
    plt.axis('off')
    plt.title(f"P: {class_names[y_pred_labels[idx]]}\nT:␣
 ↪{class_names[y_test[idx]]}")

plt.tight_layout()
plt.show()
```



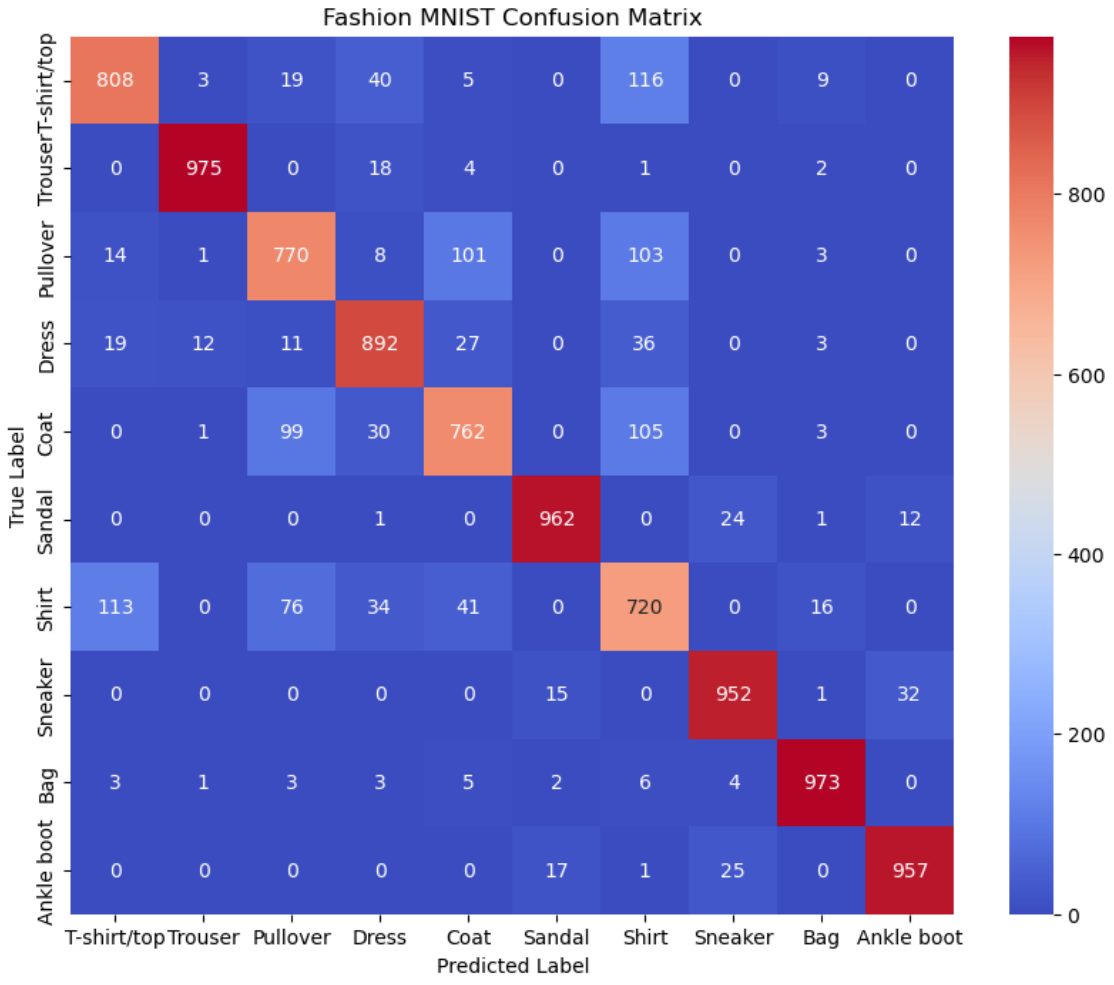
confusion matrix to see which classes my model confuses most often.

[60]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred_labels)
print(cm)
```

```
[[808   3  19  40   5   0 116   0   9   0]
 [  0 975   0  18   4   0   1   0   2   0]
 [ 14   1 770   8 101   0 103   0   3   0]
```

```
[ 19   12   11 892   27    0  36    0    3    0]
[  0    1   99   30 762    0 105    0    3    0]
[  0    0    0    1   0  962   0   24    1   12]
[113    0   76   34   41    0 720    0   16    0]
[  0    0    0    0   0   15   0  952    1   32]
[  3    1    3    3   5    2   6    4  973    0]
[  0    0    0    0   0   17   1   25    0  957]]
```

[61]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt="d", cmap="coolwarm",
            xticklabels=class_names,
            yticklabels=class_names)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Fashion MNIST Confusion Matrix")
plt.show()
```

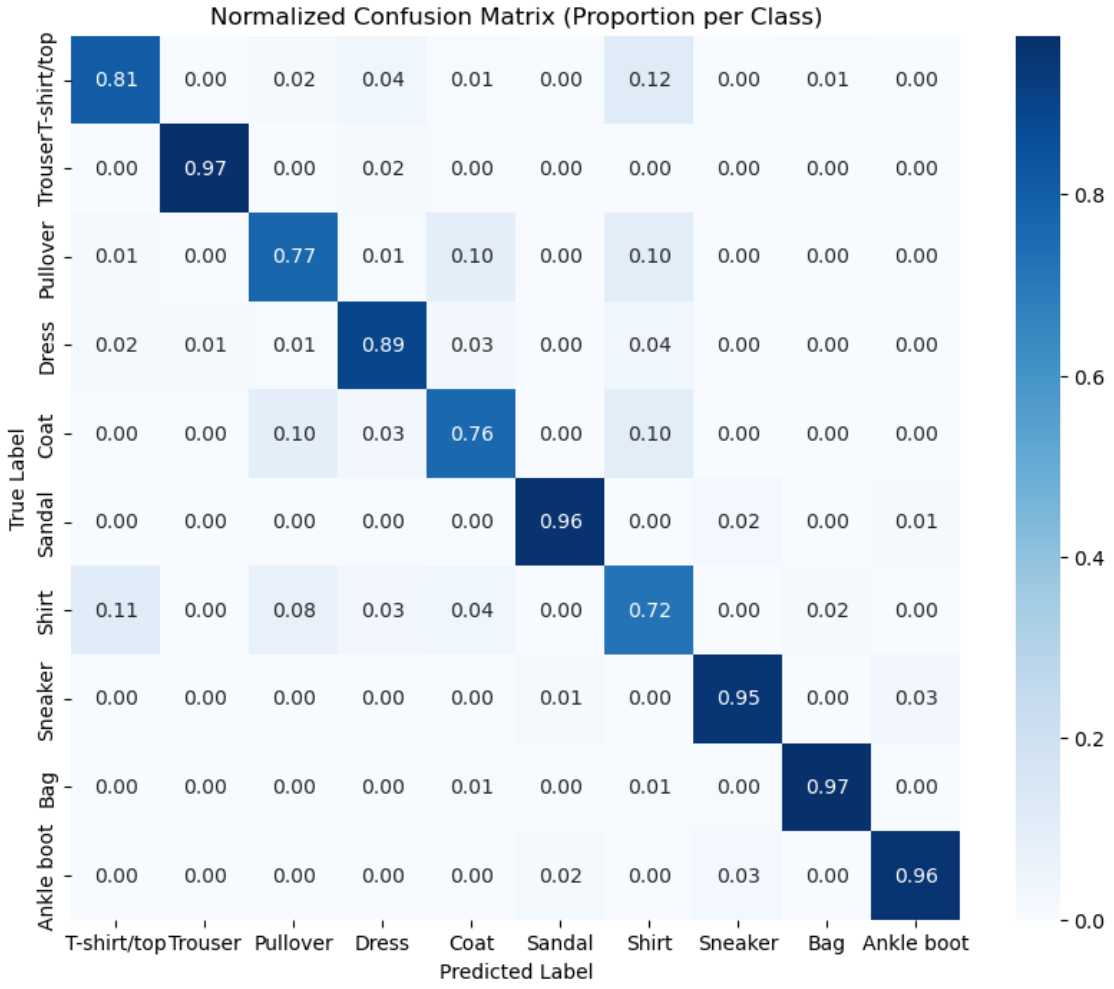


“Shirt” is the hardest class — often mistaken for “T-shirt/top”, “Coat”, or “Pullover”.

“Trouser”, “Sandal”, “Sneaker”, and “Bag” have excellent precision (near-perfect red squares). Overall, model performs very well (~88–91%).

[62]:
```
#to see percentages
cm = confusion_matrix(y_test, y_pred_labels, normalize='true')

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt=".2f", cmap="Blues",
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Normalized Confusion Matrix (Proportion per Class)")
plt.show()
```



importing classification report 1.percision when model predicts this class how often is it correct

2.Recall out of all real examples of this class. 3.F1-Score balance between precision and recall.

[63]:
```
from sklearn.metrics import classification_report
```

[64]:
```
print(classification_report(y_test , y_pred_labels , target_names =␣
↪class_names))
```

```
              precision    recall  f1-score   support

 T-shirt/top       0.84      0.81      0.83      1000
     Trouser       0.98      0.97      0.98      1000
    Pullover       0.79      0.77      0.78      1000
       Dress       0.87      0.89      0.88      1000
        Coat       0.81      0.76      0.78      1000
      Sandal       0.97      0.96      0.96      1000
       Shirt       0.66      0.72      0.69      1000
     Sneaker       0.95      0.95      0.95      1000
         Bag       0.96      0.97      0.97      1000
  Ankle boot       0.96      0.96      0.96      1000

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     10000
```

#Overall Performance: Accuracy: 0.88 (simple neural network).

#Balanced model: Macro & weighted averages both ~0.88 — no major class imbalance.

Performs consistently across most classes.

#Best Performing Classes:

Trouser (F1 = 0.98): Almost perfect. Sandal (F1 = 0.97) Bag, Sneaker, Ankle boot (F1 0.95–0.96)

#Moderate Performance: Dress (F1 = 0.88): Slight confusion with "Coat" or "Pullover" T-shirt/top (F1 = 0.84): Sometimes mistaken for "Shirt"

#Weakest Classes:

Shirt (F1 = 0.70): Most confused class — looks similar to "T-shirt/top" and "Coat" Coat (F1 = 0.81): Often mixed with "Pullover" or "Dress"

#Metric Summary:

1.Precision = 0.88: When the model predicts a class, it's correct 88% of the time.

2.Recall = 0.88: It successfully finds 88% of all real items.

3.F1-score = 0.88: Balanced and consistent performance.

##Using CNN to upgrade Accuracy

[65]:
```
#CNN model to upgrade my accuracy
x_train_cnn = x_train.reshape(-1, 28 , 28 ,1 )
x_test_cnn = x_test.reshape(-1 , 28, 28 ,1)
```

[66]:
```
# Checking the shape of my training and testing of my model
print(x_train_cnn.shape , x_test_cnn.shape)
```

```
(60000, 28, 28, 1) (10000, 28, 28, 1)
```

[67]:
```
#Traning my model
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(128 , activation = 'relu') ,
    layers.Dense(10 , activation = 'softmax')

])
```

```
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-
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)
```

[68]:
```
#Compiling my model
cnn_model.compile(optimizer = 'adam' ,
                  loss = 'sparse_categorical_crossentropy' ,
                  metrics = ['accuracy'])
```

[ ]:
```

```

[69]:
```
#fiting my model and checking accuracy
cnn_model.fit(
    x_train_cnn , y_train ,
    epochs = 5
)
```

```
Epoch 1/5
1875/1875            37s 19ms/step -
accuracy: 0.7813 - loss: 0.6091
Epoch 2/5
1875/1875            37s 19ms/step -
accuracy: 0.8895 - loss: 0.3068
```

```
Epoch 3/5
1875/1875             36s 19ms/step -
accuracy: 0.9054 - loss: 0.2551
Epoch 4/5
1875/1875             40s 21ms/step -
accuracy: 0.9174 - loss: 0.2228
Epoch 5/5
1875/1875             44s 23ms/step -
accuracy: 0.9277 - loss: 0.1945
```

[69]:
```
<keras.src.callbacks.history.History at 0x16f4b2510>
```

Epoch 1: 77.9% accuracy (already good) Epoch 5: 92.7% accuracy, loss down to 0.19 Accuracy is going up each epoch, loss going down.

[70]:
```
test_loss , test_acc = cnn_model.evaluate(x_test_cnn , y_test , verbose = 0)
print(test_acc)
print(test_loss)
```

```
0.8964999914169312
0.2722867429256439
```

Test Accuracy: 0.9014 → 90% Test Loss: 0.2748

Metric Meaning Result: 1.Accuracy: How many predictions were correct 90.8% (very strong) 2.Loss: How far predictions are from correct answers 0.26 (lower = better)

[37]:
```
history_cnn = cnn_model.fit(
    x_train_cnn, y_train,
    epochs=5,
    batch_size=64,
    validation_split=0.1,
    verbose=1
)
```

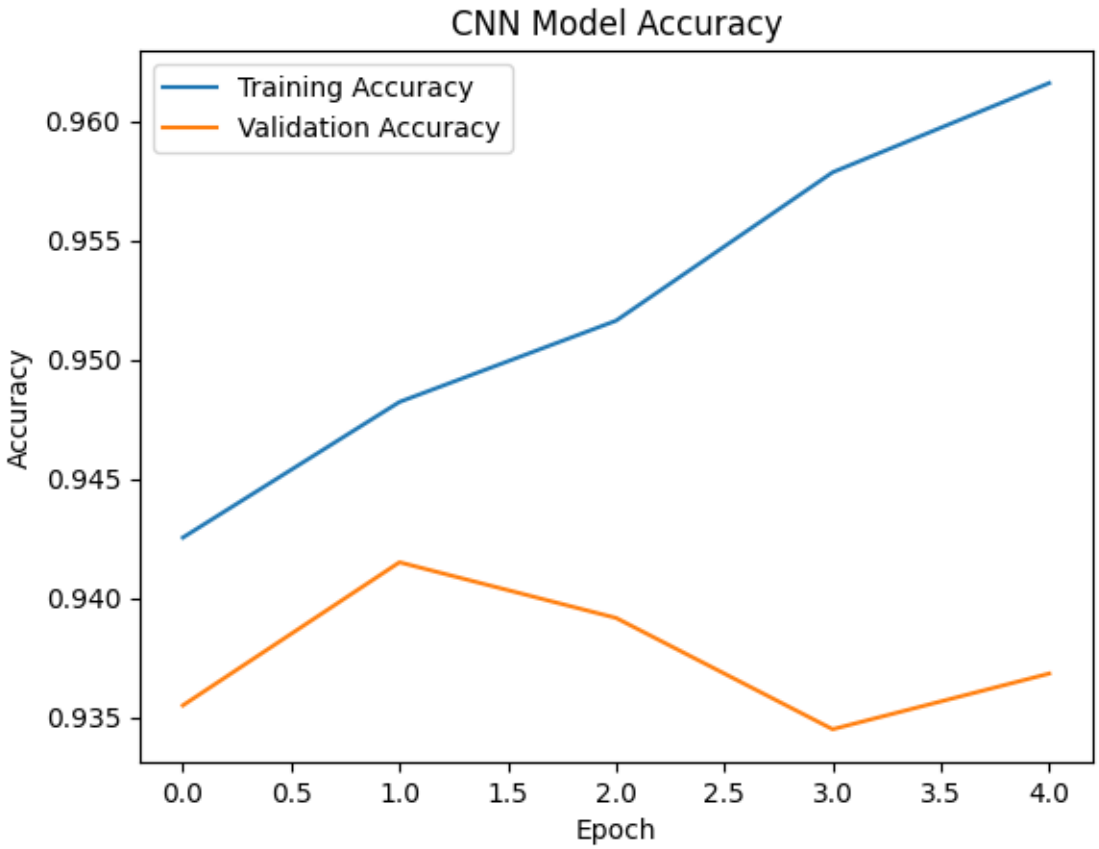
```
Epoch 1/5
844/844             47s 55ms/step -
accuracy: 0.9424 - loss: 0.1565 - val_accuracy: 0.9355 - val_loss: 0.1658
Epoch 2/5
844/844             47s 56ms/step -
accuracy: 0.9499 - loss: 0.1384 - val_accuracy: 0.9415 - val_loss: 0.1566
Epoch 3/5
844/844             80s 53ms/step -
accuracy: 0.9546 - loss: 0.1237 - val_accuracy: 0.9392 - val_loss: 0.1599
Epoch 4/5
844/844             83s 55ms/step -
accuracy: 0.9586 - loss: 0.1124 - val_accuracy: 0.9345 - val_loss: 0.1772
Epoch 5/5
844/844             45s 53ms/step -
accuracy: 0.9628 - loss: 0.1002 - val_accuracy: 0.9368 - val_loss: 0.1831
```

Epoch 1: 94% accuracy (already good) Epoch 5: 96% accuracy, loss down to 0.10 Accuracy is going up each epoch, loss going down.

[40]:
```
test_loss , test_acc = cnn_model.evaluate(x_test_cnn , y_test , verbose = 0)
print(test_acc)
print(test_loss)
```

```
0.909500002861023
0.2950947880744934
```

[41]:
```
plt.plot(history_cnn.history['accuracy'], label='Training Accuracy')
plt.plot(history_cnn.history['val_accuracy'], label='Validation Accuracy')
plt.title('CNN Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



1.Training Accuracy steadily increased from ~94.9% → ~96.4% The CNN is learning patterns properly.

2.Validation Accuracy stays around 94–95% This means the model generalizes well — it's not

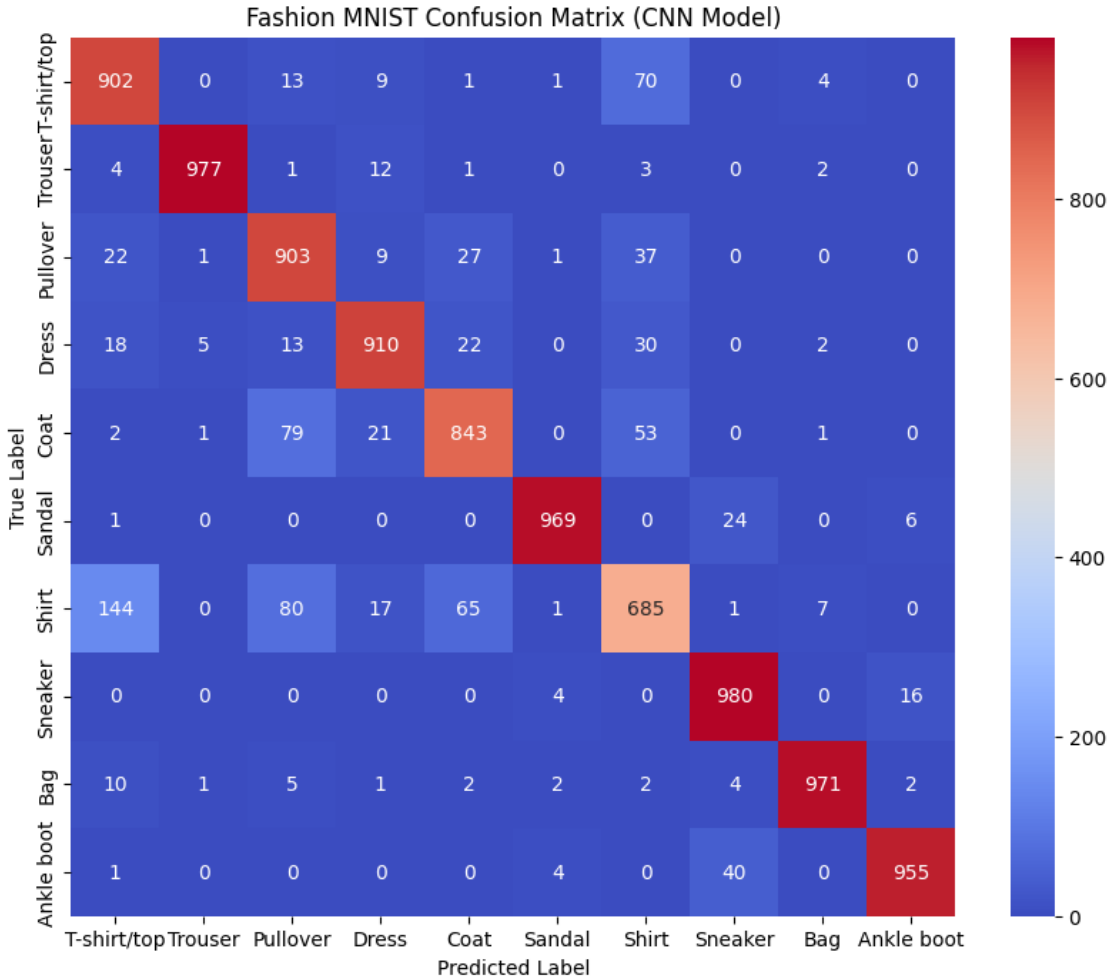
memorizing the training data too much.

3.Small gap between training & validation accuracy → Slight overfitting.

[42]:
```
#prediction. for testing data
y_pred_cnn = cnn_model.predict(x_test_cnn, verbose=0)
y_pred_labels_cnn = np.argmax(y_pred_cnn, axis=1)
```

[43]:
```
#Compute confusion matrix
cm_cnn = confusion_matrix(y_test , y_pred_labels_cnn)
```

[44]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(cm_cnn, annot=True, fmt='d', cmap='coolwarm',
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Fashion MNIST Confusion Matrix (CNN Model)')
plt.show()
```



#Each row = true label #Each column = predicted label #Diagonal values = correct predictions #Off-diagonal values = misclassifications

#The diagonal line is bright red, meaning most predictions are correct.

#Only small off-diagonal areas have errors

[45]:
```
print(classification_report(y_test, y_pred_labels_cnn,␣
↪target_names=class_names))
```

```
              precision    recall  f1-score   support

 T-shirt/top       0.82      0.90      0.86      1000
     Trouser       0.99      0.98      0.98      1000
    Pullover       0.83      0.90      0.86      1000
       Dress       0.93      0.91      0.92      1000
        Coat       0.88      0.84      0.86      1000
      Sandal       0.99      0.97      0.98      1000
       Shirt       0.78      0.69      0.73      1000
     Sneaker       0.93      0.98      0.96      1000
         Bag       0.98      0.97      0.98      1000
  Ankle boot       0.98      0.95      0.97      1000

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```

1.Precision Of all items predicted as that class, how many were actually correct. (Measures false positives).

2.Recall Of all true items in that class, how many did the model correctly find. (Measures false negatives).

3.F1-score Balance between precision and recall (higher = better).

4.Support Number of test samples per class.

CNN Class Performance Summary:

T-shirt/Top: Precision: 0.82 | Recall: 0.90 | F1: 0.86 Performs well overall. Sometimes confused with Shirt (very similar appearance).

Trouser:

Precision: 0.99 | Recall: 0.98 | F1: 0.98

Best-performing class — almost perfect. Very easy for the model to identify due to unique shape.

Pullover:

Precision: 0.83 | Recall: 0.90 | F1: 0.86

Good recall — catches most pullovers. Some confusion with Coat and Shirt (similar textures).

Dress: Precision: 0.93 | Recall: 0.91 | F1: 0.92

Strong performance. Occasionally mixed with Coat due to overlapping visual features.

Coat:

Precision: 0.88 | Recall: 0.84 | F1: 0.86

Reliable results. Slight confusion with Pullover.

Sandal:

Precision: 0.99 | Recall: 0.97 | F1: 0.98

Near-perfect recognition. Clear visual differences make it easy for CNN to classify.

Shirt: Precision: 0.78 | Recall: 0.69| F1: 0.73

Weakest class. Often mistaken for T-shirt/top, Pullover, or Coat.

Sneaker:

Precision: 0.93 | Recall: 0.98 | F1: 0.96

Excellent detection accuracy. Small mix-up with Ankle boot (similar outline).

Bag Precision: 0.98 | Recall: 0.97 | F1: 0.98

Consistent and highly accurate class.Very few errors.

Ankle Boot:

Precision: 0.98 | Recall: 0.95 | F1: 0.97

Excellent accuracy. Occasionally confused with Sneaker (both footwear).

[46]:
```
#that visualization shows both correct and incorrect predictions
np.random.seed(42)

n = 10

indices = np.random.choice(len(x_test), n, replace=False)
#This shows images + predicted vs true label.
def show_random_predictions(n=10):
    indices = np.random.choice(len(x_test), n, replace=False)
    plt.figure(figsize=(15, 4))

    for i, idx in enumerate(indices):
        plt.subplot(2, n//2, i + 1)
        plt.imshow(x_test[idx], cmap='gray')
        plt.axis('off')
        pred = class_names[y_pred_labels_cnn[idx]]
        true = class_names[y_test[idx]]
        color = 'green' if pred == true else 'red'
        plt.title(f"P: {pred}\nT: {true}", color=color)
```

```
    plt.tight_layout()
    plt.show()

show_random_predictions(10)
```