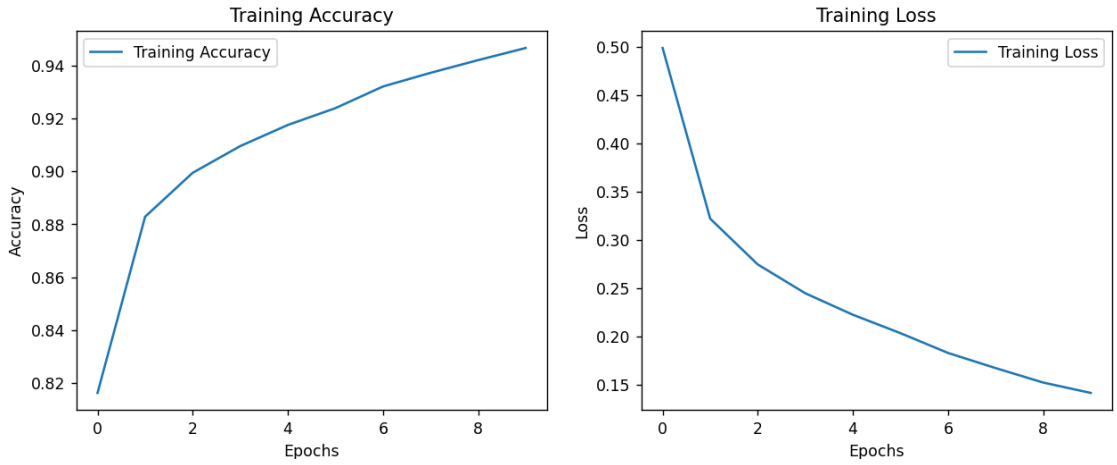
Assignment 2

Exercise 1

i) I have designed a CNN architecture for classifying our dataset that is as follows:

- Input Layer: 28x28 grayscale images.
- Convolutional Layer: 32 filters, 3x3 kernel, ReLU activation.
- Max Pooling Layer: 2x2 pooling.
- Convolutional Layer: 64 filters, 3x3 kernel, ReLU activation.
- Convolutional Layer: 64 filters, 3x3 kernel, ReLU activation.
- Max Pooling Layer: 2x2 pooling.
- Flatten Layer: Convert 2D to 1D.
- Fully Connected Layer: 128 units, ReLU activation.
- Output Layer: 10 units (for the 10 classes), softmax activation
- Loss Function: Categorical Crossentropy (since we have multiple classes).
- Optimizer: Adam (popular for fast convergence).
- Metric: Accuracy
- 10 epochs

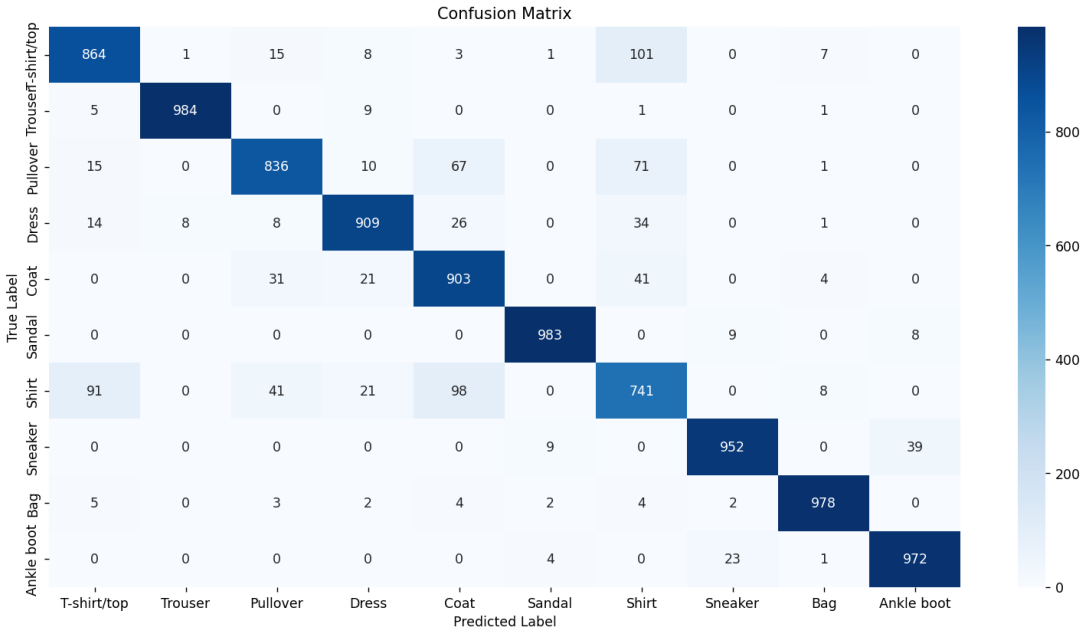
ii) Here are plots and metrics that evaluate the model's performance.



Training Accuracy and Loss Plot:

Training Accuracy: The accuracy improves steadily across the epochs, starting from around 82% and reaching above 94% by the 10th epoch. This indicates that the model is learning effectively during training.

Training Loss: The loss decreases progressively, suggesting that the model is minimizing errors as it is trained.



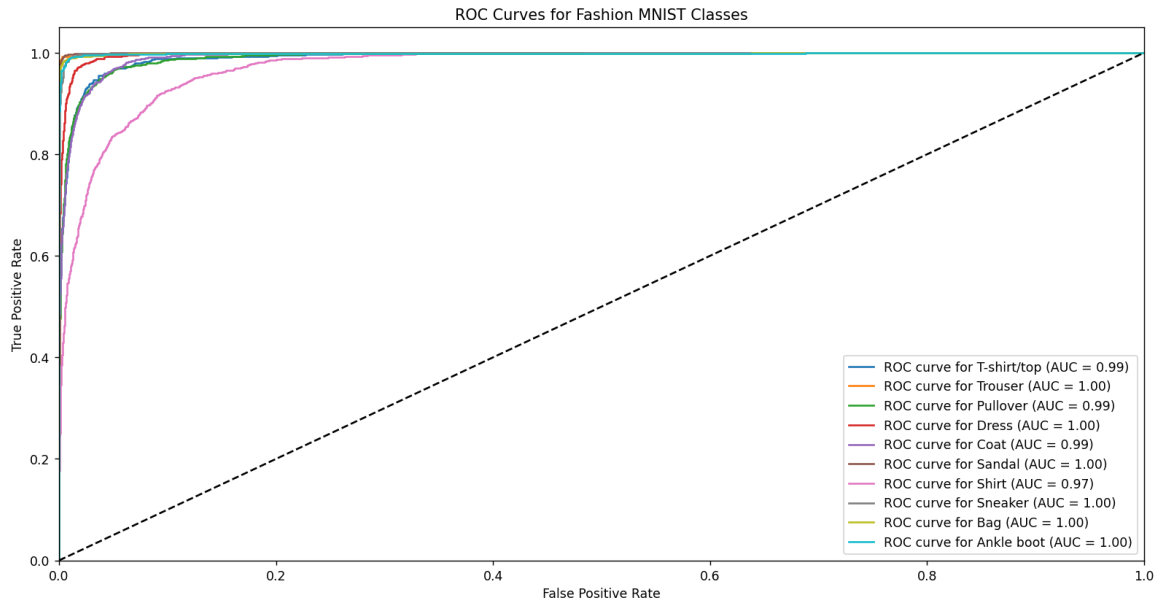
Confusion Matrix

The model performs well in most classes, but there are a few notable misclassifications:

Class T-shirt/top is often confused with Shirt: There are 101 misclassifications, likely because these items have similar visual features (both involve upper-body clothing).

Class Shirt shows significant confusion with Class T-shirt/top and Class Coat.

For most other classes, such as Sneaker, Bag , and Ankle Boot, the model performs extremely well with minimal misclassifications.



ROC Curves for Fashion MNIST Classes

The model has nearly perfect performance for most classes, as reflected by the high AUC scores close to 1.0.

Class Shirt has the lowest AUC (0.97), which aligns with the confusion observed in the confusion matrix. This means the model struggles slightly more with distinguishing shirts from other classes.

Overall, the model has excellent performance based on the ROC curves. The challenge lies mainly in distinguishing between certain visually similar categories (e.g., shirts and t-shirts/tops)

```
Test Accuracy: 0.9121999740600586
Test Loss: 0.257387638092041
```

```
Classification Report:
              precision    recall  f1-score   support

 T-shirt/top       0.87      0.86      0.87      1000
     Trouser       0.99      0.98      0.99      1000
    Pullover       0.90      0.84      0.86      1000
       Dress       0.93      0.91      0.92      1000
        Coat       0.82      0.90      0.86      1000
      Sandal       0.98      0.98      0.98      1000
       Shirt       0.75      0.74      0.74      1000
     Sneaker       0.97      0.95      0.96      1000
         Bag       0.98      0.98      0.98      1000
  Ankle boot       0.95      0.97      0.96      1000

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

Average AUC-ROC Score: 0.99
```

Classification Report

Overall Accuracy: The model achieved an accuracy of 0.91, indicating that 91% of the predictions were correct.

Precision, Recall, and F1-Score: These metrics are consistently high (around 0.91) across all categories, suggesting balanced performance in terms of precision (correct positive predictions), recall (correctly identified actual positives), and the harmonic mean of both (F1-score).

Support: Each category has a support value of 1000, meaning the model was tested on 1000 instances per category.

AUC-ROC Score: The average AUC-ROC score is 0.99, indicating excellent performance in distinguishing between classes.

These results suggest that the model is performing well across all categories with high precision, recall, and F1-scores, and an excellent AUC-ROC score.

Exercise 2

i)

Residual connections, introduced in ResNet (Residual Networks), allow a model to "skip" one or more layers during forward and backward propagation. The central idea is to enable a shortcut

connection, where the input to a set of layers is added directly to the output, bypassing the intermediate layers.

The vanishing gradient problem occurs when gradients become very small, especially in deep networks, preventing the network from learning effectively. Residual connections help in several ways:

Easier Gradient Flow: The direct shortcut (residual connection) allows the gradient to bypass intermediate layers during backpropagation, making it easier for the network to propagate gradients back to earlier layers. This improves gradient flow and mitigates the vanishing gradient problem, enabling deeper networks to train more effectively.

Identity Mapping: In the residual connection, if the transformation F(x) learns to output values near zero, the identity mapping (x+0=x) will pass the input forward unchanged. This prevents the degradation of performance in very deep networks by allowing the network to "skip" layers that aren't needed, making training more stable and efficient.

Improved Training for Deep Networks: Deep networks without residual connections can suffer from diminishing performance as more layers are added, due to optimization difficulties. Residual connections allow much deeper networks to be trained without this degradation in performance.

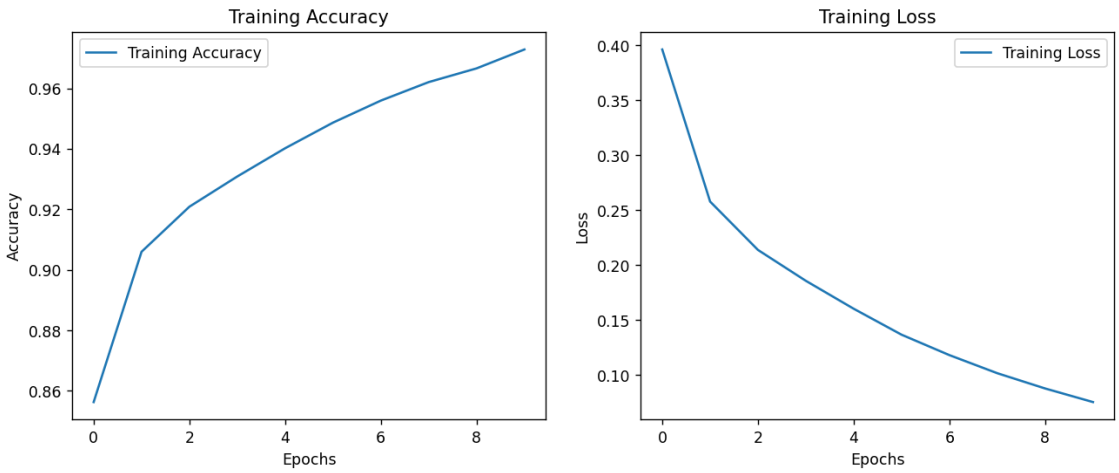
ii)

Residual blocks are added in my model using the following code.

```
# Input layer
input_layer = layers.Input(shape=(28, 28, 1))

# First convolutional layer
conv1 = layers.Conv2D( filters: 32,  kernel_size: (3, 3), activation='relu', padding='same')(input_layer)
pool1 = layers.MaxPooling2D((2, 2))(conv1)

# Second convolutional layer with a residual connection
conv2 = layers.Conv2D( filters: 64,  kernel_size: (3, 3), activation='relu', padding='same')(pool1)
pool2 = layers.MaxPooling2D((2, 2))(conv2)

# To add a residual connection, we need to match the dimensions:
# We upsample pool1 to match the dimensions of pool2 using a Conv2D layer
residual1 = layers.Conv2D( filters: 64,  kernel_size: (1, 1), padding='same')(pool1)
residual1 = layers.MaxPooling2D((2, 2))(residual1)  # Match dimensions

# Add the residual connection
residual_output1 = layers.add([pool2, residual1])

# Third convolutional layer with another residual connection
conv3 = layers.Conv2D( filters: 64,  kernel_size: (3, 3), activation='relu', padding='same')(residual_output1)

# Flatten the feature maps and apply fully connected layers
flatten = layers.Flatten()(conv3)
dense1 = layers.Dense( units: 64, activation='relu')(flatten)
output_layer = layers.Dense( units: 10, activation='softmax')(dense1)
```

```
# Define the model
model = models.Model(inputs=input_layer, outputs=output_layer)

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(train_images, train_labels, epochs=10)
```

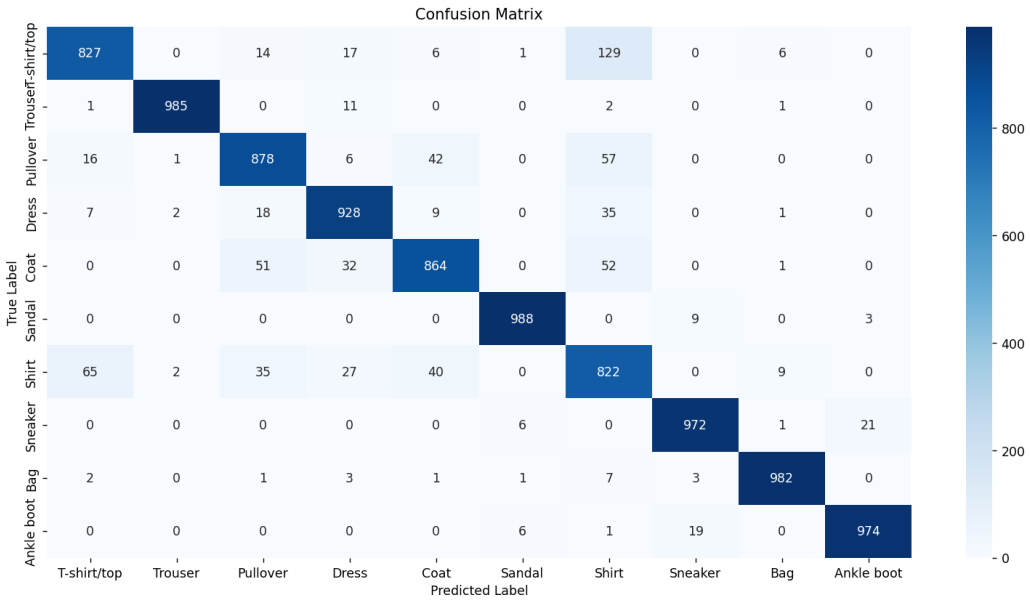
iii) Here are plots and metrics that evaluate the model's performance.



Training Accuracy and Loss Plot:

Training Accuracy: The accuracy improves steadily across the epochs, starting from around 85% and reaching well above 96% by the 10th epoch, showing an improvement from the previous model. This indicates that the model is learning effectively during training.

Training Loss: The loss decreases progressively reaching a lower number than the previous model, which shows an improvement. This suggesting that the model is minimizing errors as it is trained.



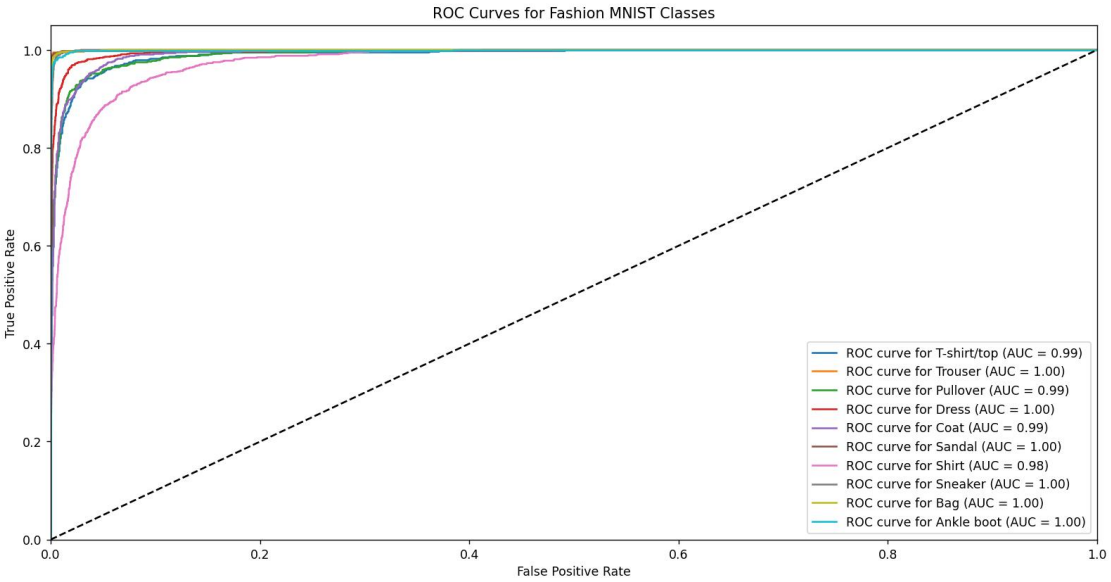
Confusion Matrix

The model performs well in most classes, but there are a few notable misclassifications:

Class T-shirt/top is still confused with Class Shirt like the previous model.

Class Shirt shows less confusion with Class T-shirt/top and Class Pullover and Class Coat than the previous model.

For most other classes, such as Sneaker, Bag , and Ankle Boot, the model again performs extremely well with minimal misclassifications.



ROC Curves for Fashion MNIST Classes

The model has nearly perfect performance for most classes, as reflected by the high AUC scores close to 1.0.

Class Shirt has the lowest AUC (0.98), which aligns with the confusion observed in the confusion matrix. This means the model struggles slightly more with distinguishing shirts from other classes. Here we have a slight improvement from the last model where the lowest AUC was 0.97.

Overall, the model has excellent performance based on the ROC curves. The challenge lies mainly in distinguishing between certain visually similar categories (e.g., shirts and t-shirts/tops)

```
Test Accuracy: 0.9219999940940094
Test Loss: 0.3039233088493347
```

```
Classification Report:
              precision    recall  f1-score   support

 T-shirt/top       0.90      0.83      0.86      1000
     Trouser       0.99      0.98      0.99      1000
    Pullover       0.88      0.88      0.88      1000
       Dress       0.91      0.93      0.92      1000
        Coat       0.90      0.86      0.88      1000
      Sandal       0.99      0.99      0.99      1000
       Shirt       0.74      0.82      0.78      1000
     Sneaker       0.97      0.97      0.97      1000
         Bag       0.98      0.98      0.98      1000
  Ankle boot       0.98      0.97      0.97      1000

    accuracy                           0.92     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.92      0.92      0.92     10000

Average AUC-ROC Score: 0.99
```

Classification Report

Overall Accuracy: The model achieved an accuracy of 0.92, indicating that 92% of the predictions were correct. This is a slight improvement from the previous model.

Precision, Recall, and F1-Score: These metrics are consistently high (around 0.92) across all categories, suggesting balanced performance in terms of precision (correct positive predictions), recall (correctly identified actual positives), and the harmonic mean of both (F1-score). Here we can see again an improvement when compared to the previous model.

Support: Each category has a support value of 1000, meaning the model was tested on 1000 instances per category.

AUC-ROC Score: The average AUC-ROC score is 0.99, indicating excellent performance in distinguishing between classes.

These results suggest that the model is performing well across all categories with high precision, recall, and F1-scores, and an excellent AUC-ROC score.

Overall the results of the model which uses residual connections show an improvement when compared to the initial model that does not use residual connections.