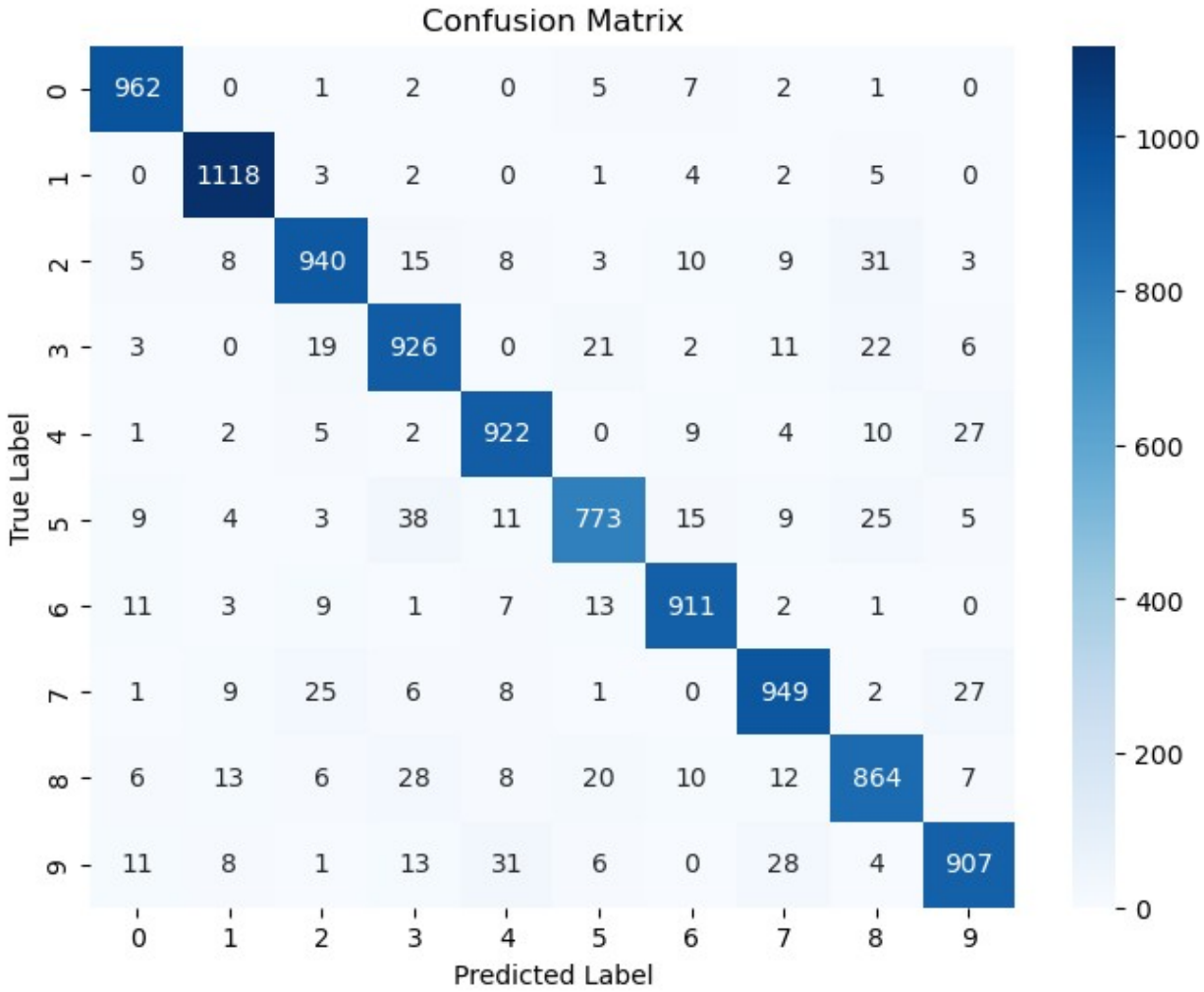
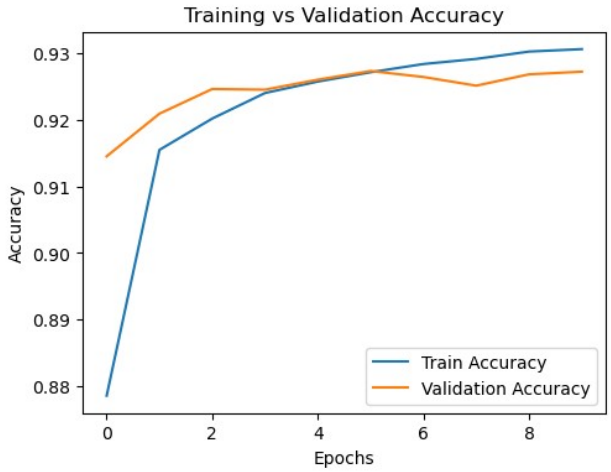
```python
#ANN Without regularisation
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) =
keras.datasets.mnist.load_data()

# Normalize the pixel values
x_train, x_test = x_train / 255.0, x_test / 255.0

# Flatten the images (Since no hidden layers, direct input to output)
x_train = x_train.reshape(-1, 28*28)
x_test = x_test.reshape(-1, 28*28)

# Build the basic Perceptron model
model = keras.Sequential([
    layers.Dense(10, activation='softmax', input_shape=(28*28,))
])

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

# Train the model
history = model.fit(x_train, y_train, epochs=10, batch_size=32,
validation_data=(x_test, y_test))

# Evaluate model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"Test Accuracy: {test_acc:.4f}")

# Predict labels
y_pred = np.argmax(model.predict(x_test), axis=1)

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=range(10), yticklabels=range(10))
plt.xlabel("Predicted Label")
```

```
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

# Plot training vs validation accuracy/loss
plt.figure(figsize=(12, 4))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training vs Validation Accuracy')

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training vs Validation Loss')

plt.show()
```

```
Epoch 1/10

C:\Newanaconda\Lib\site-packages\keras\src\layers\core\dense.py:87:
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)

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8146 - loss:
0.7197 - val_accuracy: 0.9145 - val_loss: 0.3086
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 902us/step - accuracy: 0.9137 -
loss: 0.3116 - val_accuracy: 0.9209 - val_loss: 0.2826
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 813us/step - accuracy: 0.9202 -
loss: 0.2843 - val_accuracy: 0.9246 - val_loss: 0.2705
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 881us/step - accuracy: 0.9229 -
loss: 0.2776 - val_accuracy: 0.9245 - val_loss: 0.2709
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 886us/step - accuracy: 0.9281 -
loss: 0.2603 - val_accuracy: 0.9260 - val_loss: 0.2654
```

```
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 897us/step - accuracy: 0.9269 -
loss: 0.2654 - val_accuracy: 0.9273 - val_loss: 0.2627
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 988us/step - accuracy: 0.9273 -
loss: 0.2619 - val_accuracy: 0.9264 - val_loss: 0.2646
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 988us/step - accuracy: 0.9297 -
loss: 0.2497 - val_accuracy: 0.9251 - val_loss: 0.2675
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9317 - loss:
0.2491 - val_accuracy: 0.9268 - val_loss: 0.2627
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9295 - loss:
0.2540 - val_accuracy: 0.9272 - val_loss: 0.2629
313/313 - 0s - 897us/step - accuracy: 0.9272 - loss: 0.2629
Test Accuracy: 0.9272
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 961us/step
```



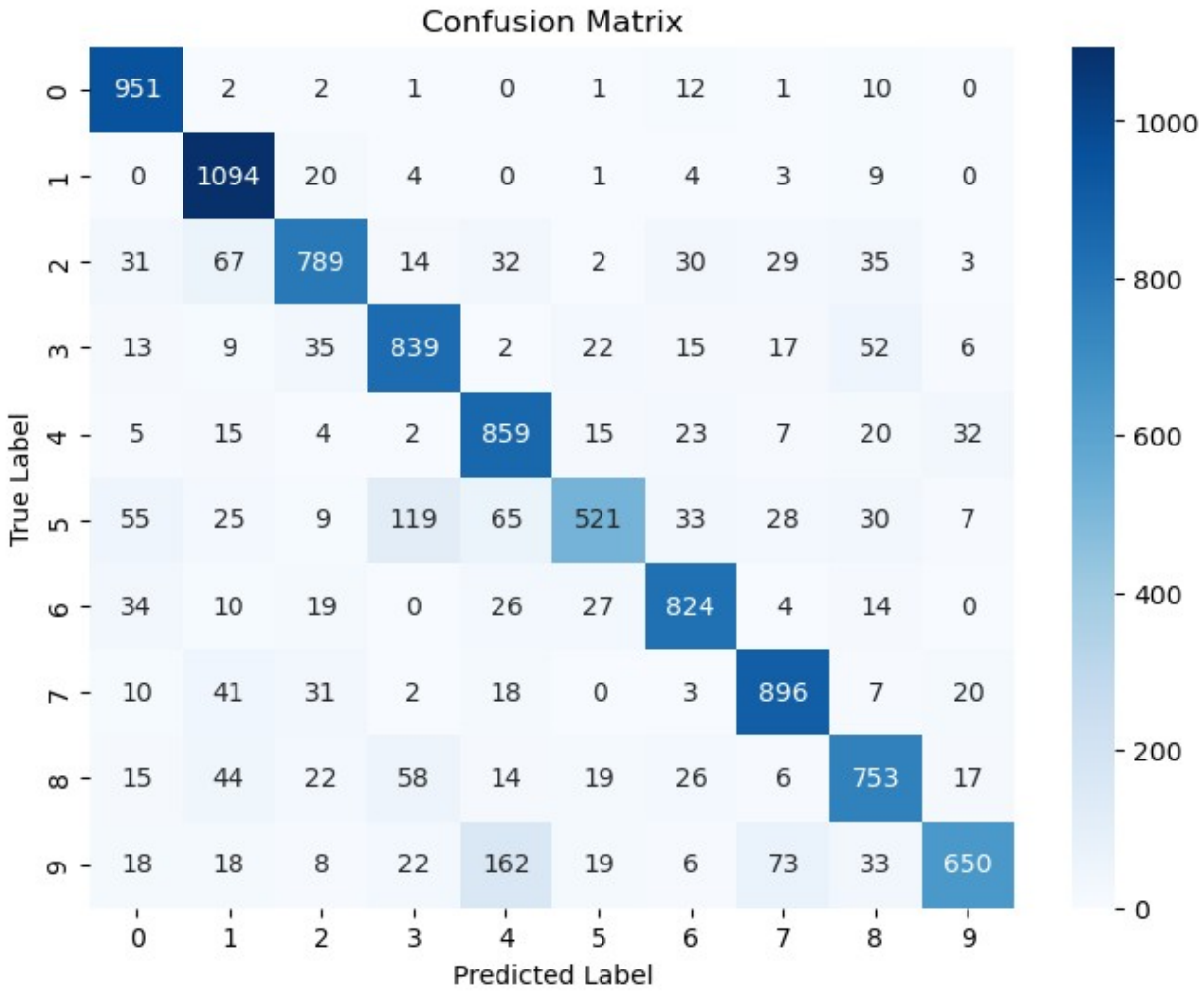


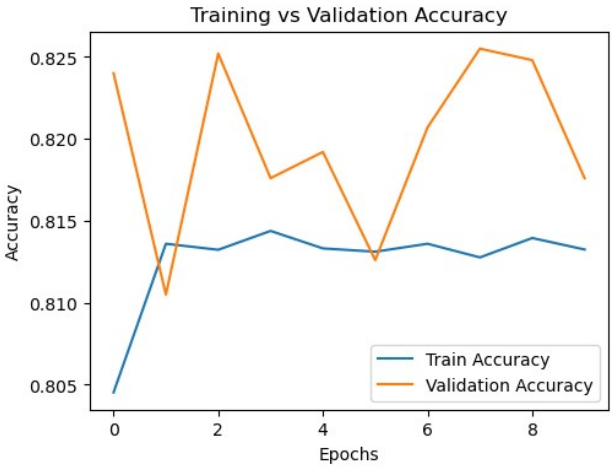

```
#With regularisation
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, regularizers
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) =
keras.datasets.mnist.load_data()

# Normalize the pixel values
x_train, x_test = x_train / 255.0, x_test / 255.0

# Flatten the images (Since no hidden layers, direct input to output)
x_train = x_train.reshape(-1, 28*28)
x_test = x_test.reshape(-1, 28*28)

# Build the Perceptron model with L1 & L2 regularization
model = keras.Sequential([
    layers.Dense(10, activation='softmax', input_shape=(28*28,),
                 kernel_regularizer=regularizers.l1_l2(l1=0.01,
l2=0.01))  # L1 & L2 regularization
])

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

# Train the model
history = model.fit(x_train, y_train, epochs=10, batch_size=32,
validation_data=(x_test, y_test))
```

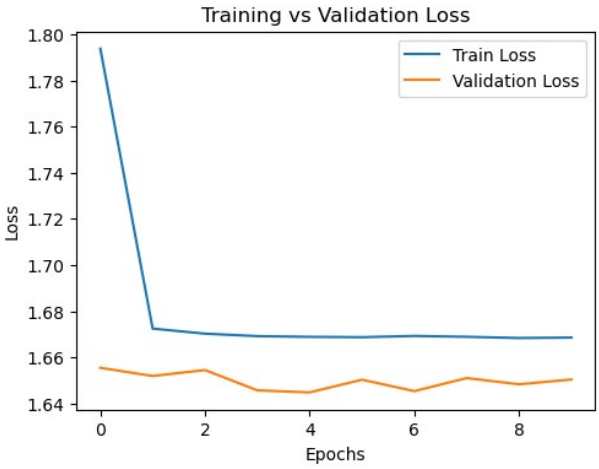
```python
# Evaluate model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"Test Accuracy: {test_acc:.4f}")

# Predict labels
y_pred = np.argmax(model.predict(x_test), axis=1)

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=range(10), yticklabels=range(10))
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

# Plot training vs validation accuracy/loss
plt.figure(figsize=(12, 4))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training vs Validation Accuracy')

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training vs Validation Loss')

plt.show()
```

```
C:\Newanaconda\Lib\site-packages\keras\src\layers\core\dense.py:87:
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
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.7664 - loss:
2.1332 - val_accuracy: 0.8240 - val_loss: 1.6555
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8151 - loss:
1.6729 - val_accuracy: 0.8105 - val_loss: 1.6519
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8114 - loss:
1.6733 - val_accuracy: 0.8252 - val_loss: 1.6545
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8143 - loss:
1.6725 - val_accuracy: 0.8176 - val_loss: 1.6457
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8105 - loss:
1.6707 - val_accuracy: 0.8192 - val_loss: 1.6448
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8119 - loss:
1.6702 - val_accuracy: 0.8126 - val_loss: 1.6503
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8111 - loss:
1.6719 - val_accuracy: 0.8207 - val_loss: 1.6454
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8109 - loss:
1.6705 - val_accuracy: 0.8255 - val_loss: 1.6510
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8142 - loss:
1.6690 - val_accuracy: 0.8248 - val_loss: 1.6484
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8159 - loss:
1.6697 - val_accuracy: 0.8176 - val_loss: 1.6504
313/313 - 0s - 861us/step - accuracy: 0.8176 - loss: 1.6504
Test Accuracy: 0.8176
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 899us/step
```


Confusion Matrix
True Label
Predicted Label
0 951 2 2 1 0 1 12 1 10 0
1 0 1094 20 4 0 1 4 3 9 0
2 31 67 789 14 32 2 30 29 35 3
3 13 9 35 839 2 22 15 17 52 6
4 5 15 4 2 859 15 23 7 20 32
5 55 25 9 119 65 521 33 28 30 7
6 34 10 19 0 26 27 824 4 14 0
7 10 41 31 2 18 0 3 896 7 20
8 15 44 22 58 14 19 26 6 753 17
9 18 18 8 22 162 19 6 73 33 650
0 1 2 3 4 5 6 7 8 9
1000
800
600
400
200
0


Training vs Validation Accuracy
Accuracy
Epochs
Train Accuracy
Validation Accuracy
0.825
0.820
0.815
0.810
0.805
0
2
4
6
8


Training vs Validation Loss
Loss
Epochs
Train Loss
Validation Loss
1.80
1.78
1.76
1.74
1.72
1.70
1.68
1.66
1.64
0
2
4
6
8