Week-8:Construct AlexNet on MNIST dataset compute the performance evaluation matrices.

Code :

```
import tensorflow as tf
import numpy as np
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore')

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# Reshape data for CNN
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

# Define AlexNet for MNIST
model = models.Sequential([
    layers.Conv2D(32, (5, 5), 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(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

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

# Print model summary
model.summary()
```

```
# Train model
model.fit(x_train, y_train, epochs=5, batch_size=128,
validation_split=0.2)

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

import numpy as np
# Predictions
y_pred_probabilities = model.predict(x_test)
y_pred = np.argmax(y_pred_probabilities, axis=1)
print("Prediction ", y_pred)

# Classification report
print(classification_report(y_test, y_pred))
```

Output:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 24, 24, 32)	832
max_pooling2d (MaxPooling2D)	(None, 12, 12, 32)	0
conv2d_1 (Conv2D)	(None, 10, 10, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 128)	204,928
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 10)	650

```
Epoch 1/5375/375 ━━━━━━━━━━━━━━━━━━━━ 28s 62ms/step - accuracy: 0.81
50 - loss: 0.5899 - val_accuracy: 0.9805 - val_loss: 0.0713
Epoch 2/5375/375 ━━━━━━━━━━━━━━━━━━━━ 29s 78ms/step - accuracy: 0.97
96 - loss: 0.0655 - val_accuracy: 0.9855 - val_loss: 0.0528
Epoch 3/5375/375 ━━━━━━━━━━━━━━━━━━━━ 38s 69ms/step - accuracy: 0.98
74 - loss: 0.0422 - val_accuracy: 0.9857 - val_loss: 0.0478
Epoch 4/5375/375 ━━━━━━━━━━━━━━━━━━━━ 42s 71ms/step - accuracy: 0.99
03 - loss: 0.0319 - val_accuracy: 0.9870 - val_loss: 0.0456
Epoch 5/5375/375 ━━━━━━━━━━━━━━━━━━━━ 36s 95ms/step - accuracy: 0.99
26 - loss: 0.0239 - val_accuracy: 0.9877 - val_loss: 0.0425

313/313 - 2s - 8ms/step - accuracy: 0.9897 - loss: 0.0334
Test Accuracy: 0.9897000193595886

313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step
Prediction  [7 2 1 ... 4 5 6]

# Classification report
precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       0.99      1.00      1.00      1135
           2       0.99      0.99      0.99      1032
           3       0.98      1.00      0.99      1010
           4       0.99      1.00      0.99       982
           5       1.00      0.98      0.99       892
           6       0.99      0.99      0.99       958
           7       1.00      0.98      0.99      1028
           8       0.98      0.99      0.99       974
           9       1.00      0.97      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```