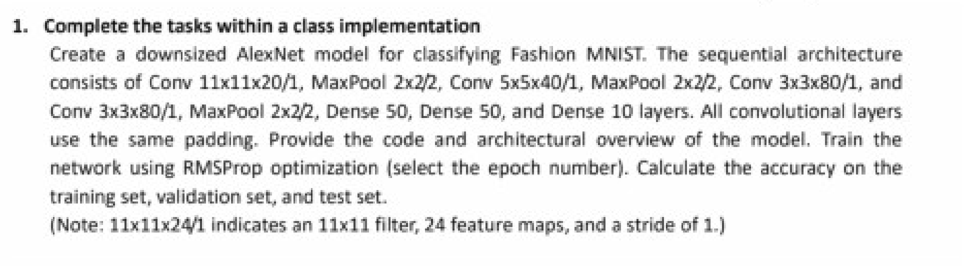
```python
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import os

# The exact path you provided
image_path =
"/var/folders/_7/f441syp12hxc66fp7hpbg3j40000gn/T/TemporaryItems/NSIRD
_screencaptureui_LJjnro/Screenshot 2025-04-14 at 4.40.58 PM.png"

try:
    # Check if the file actually exists at the path
    if not os.path.exists(image_path):
        print(f"Error: File not found at the specified path:")
        print(image_path)
    else:
        # Read the image file into an array
        print(f"Loading '{os.path.basename(image_path)}'...")
        img_data = mpimg.imread(image_path)

        # Create a figure and axes to display the image
        fig, ax = plt.subplots()
        ax.imshow(img_data)

        # Optional: Hide axes and labels for a cleaner look
        ax.axis('off')
        ax.set_title(os.path.basename(image_path)) # Show filename as
title

        # Show the plot window
        print("Displaying image in a Matplotlib window...")
        plt.show()

except FileNotFoundError:
    print(f"Error: File not found (double check): {image_path}")
except Exception as e:
    print(f"An error occurred while trying to load or display the
image with Matplotlib:")
    print(e)
    print("\nEnsure the file is a valid image format supported by
Matplotlib (like PNG, JPG).")
    print("Make sure you have Matplotlib installed (`pip install
matplotlib`).")
```

```
Loading 'Screenshot 2025-04-14 at 4.40.58 PM.png'...
Displaying image in a Matplotlib window...
```

Screenshot 2025-04-14 at 4.40.58 PM.png



1. **Complete the tasks within a class implementation**

   Create a downsized AlexNet model for classifying Fashion MNIST. The sequential architecture consists of Conv 11x11x20/1, MaxPool 2x2/2, Conv 5x5x40/1, MaxPool 2x2/2, Conv 3x3x80/1, and Conv 3x3x80/1, MaxPool 2x2/2, Dense 50, Dense 50, and Dense 10 layers. All convolutional layers use the same padding. Provide the code and architectural overview of the model. Train the network using RMSProp optimization (select the epoch number). Calculate the accuracy on the training set, validation set, and test set.

   (Note: 11x11x24/1 indicates an 11x11 filter, 24 feature maps, and a stride of 1.)

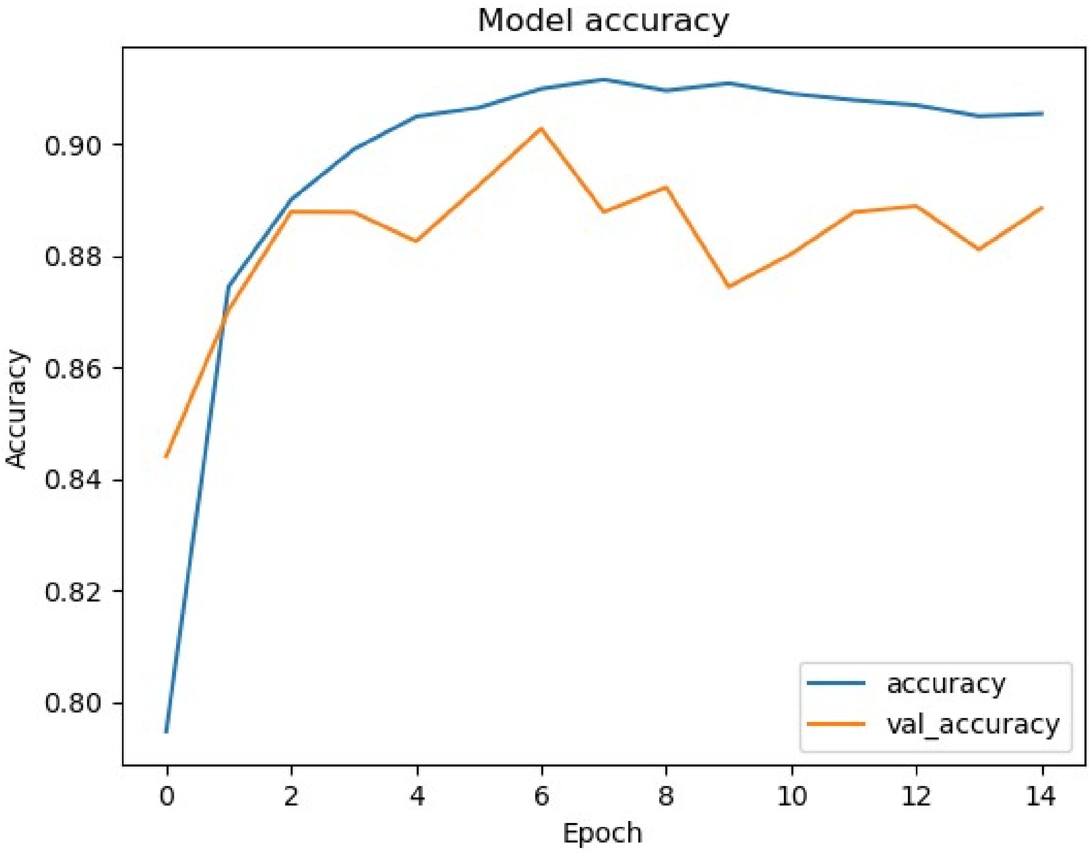
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split

#Define the model
def models():
    Alexnet = keras.Sequential([
        layers.Conv2D(filters=20, kernel_size=(11, 11), strides=1,
padding = 'same', activation='relu',data_format="channels_last",
input_shape = (28, 28, 1), name='conv1'),
        layers.MaxPooling2D(pool_size=(2, 2), strides =2,
name='pool1'),
        layers.Conv2D(filters=40, kernel_size=(5, 5), strides=1,
padding='same', activation='relu', data_format="channels_last",
name='conv2'),
        layers.MaxPooling2D(pool_size=(2, 2), strides =2,
name='pool2'),
        layers.Conv2D(filters=80, kernel_size=(3, 3), strides=1,
padding='same', activation='relu', data_format="channels_last",
name='conv3'),
        layers.Conv2D(filters=80, kernel_size=(3, 3), strides=1,
padding='same', activation='relu', data_format="channels_last",
name='conv4'),
        layers.MaxPooling2D(pool_size=(2, 2), strides =2,
name='pool3'),
        layers.Flatten(name='flatten'),
        layers.Dense(50, activation='relu', name='fc1'),
        layers.Dense(50, activation='relu', name='fc2'),
        layers.Dense(10, activation='softmax', name='output')
    ])
    #summary of the model
    Alexnet.summary()
    return Alexnet
```

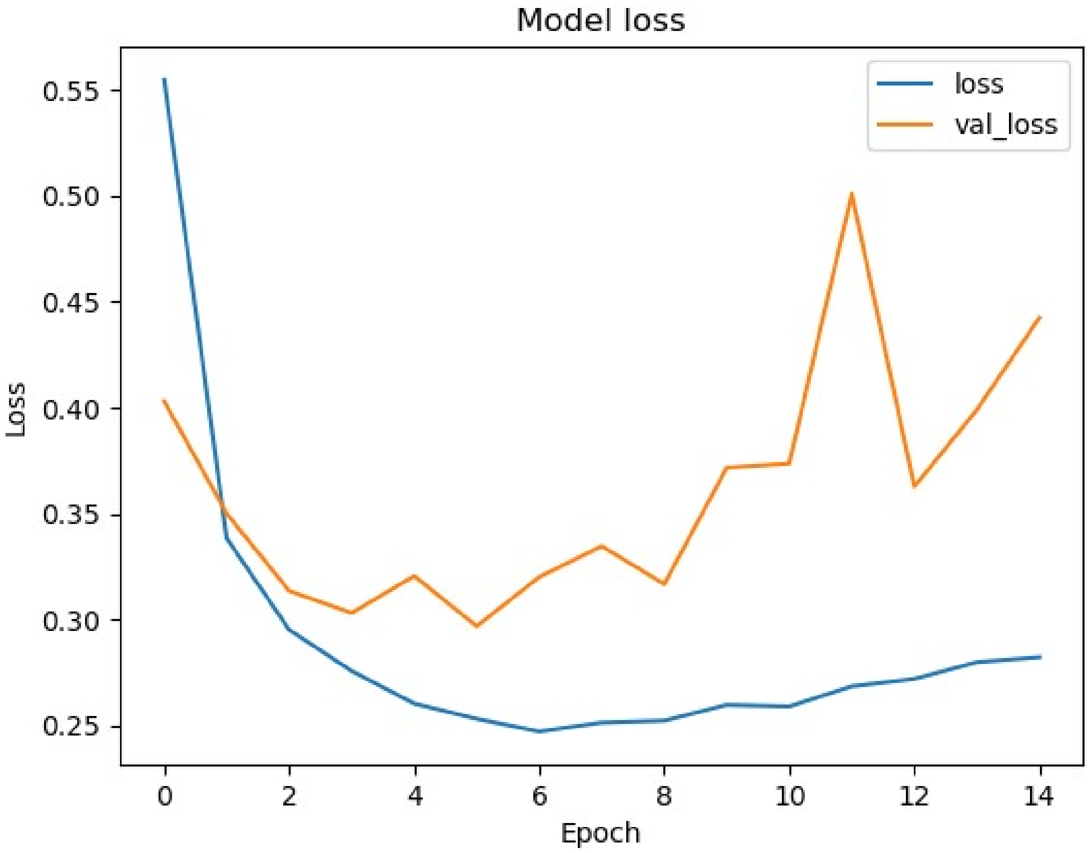
```
#Load the data and preprocessing

def load_dt():

    (x_train, y_train), (x_test, y_test) =
keras.datasets.fashion_mnist.load_data()
    x_train = x_train.astype('float32') / 255.0
    x_test = x_test.astype('float32') / 255.0

    #splitting the data into training and val sets
    x_train, x_val, y_train, y_val = train_test_split(x_train,
y_train, test_size=0.25, random_state=42)

    return x_train, x_val, y_train, y_val, x_test, y_test

def compile_model(alx,op,ep,bs):
    #Compile the model

    alx.compile(optimizer= op, loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
    #load the data
    x_train, x_val, y_train, y_val, x_test, y_test = load_dt()
    #Train the model
    history=alx.fit(x_train, y_train, epochs=ep, batch_size=bs,
validation_data=(x_val, y_val))

    plt.plot(history.history['accuracy'], label='accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()
    plt.plot(history.history['loss'], label='loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()

    return history, alx
```

```python
#Set the parameters
op = keras.optimizers.RMSprop()
ep = 15
bs = 32
#Compile the model
alx = models()
history= compile_model(alx,op, ep, bs)
#Plot the training and validation accuracy and loss
```

```
WARNING:absl:At this time, the v2.11+ optimizer
`tf.keras.optimizers.RMSprop` runs slowly on M1/M2 Macs, please use
the legacy Keras optimizer instead, located at
`tf.keras.optimizers.legacy.RMSprop`.

Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv1 (Conv2D)              (None, 28, 28, 20)        2440      
                                                                 
 pool1 (MaxPooling2D)        (None, 14, 14, 20)        0         
                                                                 
 conv2 (Conv2D)              (None, 14, 14, 40)        20040     
                                                                 
 pool2 (MaxPooling2D)        (None, 7, 7, 40)          0         
                                                                 
 conv3 (Conv2D)              (None, 7, 7, 80)          28880     
                                                                 
 conv4 (Conv2D)              (None, 7, 7, 80)          57680     
                                                                 
 pool3 (MaxPooling2D)        (None, 3, 3, 80)          0         
                                                                 
 flatten (Flatten)           (None, 720)               0         
                                                                 
 fc1 (Dense)                 (None, 50)                36050     
                                                                 
 fc2 (Dense)                 (None, 50)                2550      
                                                                 
 output (Dense)              (None, 10)                510       
                                                                 
=================================================================
Total params: 148150 (578.71 KB)
Trainable params: 148150 (578.71 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/15
```

```
1407/1407 [==============================] - 75s 53ms/step - loss:
0.5548 - accuracy: 0.7947 - val_loss: 0.4029 - val_accuracy: 0.8441
Epoch 2/15
1407/1407 [==============================] - 51s 36ms/step - loss:
0.3383 - accuracy: 0.8745 - val_loss: 0.3499 - val_accuracy: 0.8703
Epoch 3/15
1407/1407 [==============================] - 51s 36ms/step - loss:
0.2951 - accuracy: 0.8901 - val_loss: 0.3134 - val_accuracy: 0.8879
Epoch 4/15
1407/1407 [==============================] - 51s 36ms/step - loss:
0.2757 - accuracy: 0.8991 - val_loss: 0.3030 - val_accuracy: 0.8878
Epoch 5/15
1407/1407 [==============================] - 51s 36ms/step - loss:
0.2603 - accuracy: 0.9050 - val_loss: 0.3205 - val_accuracy: 0.8826
Epoch 6/15
1407/1407 [==============================] - 57s 41ms/step - loss:
0.2531 - accuracy: 0.9066 - val_loss: 0.2967 - val_accuracy: 0.8926
Epoch 7/15
1407/1407 [==============================] - 53s 38ms/step - loss:
0.2471 - accuracy: 0.9100 - val_loss: 0.3200 - val_accuracy: 0.9028
Epoch 8/15
1407/1407 [==============================] - 55s 39ms/step - loss:
0.2513 - accuracy: 0.9116 - val_loss: 0.3346 - val_accuracy: 0.8879
Epoch 9/15
1407/1407 [==============================] - 57s 40ms/step - loss:
0.2522 - accuracy: 0.9096 - val_loss: 0.3166 - val_accuracy: 0.8923
Epoch 10/15
1407/1407 [==============================] - 52s 37ms/step - loss:
0.2596 - accuracy: 0.9109 - val_loss: 0.3716 - val_accuracy: 0.8745
Epoch 11/15
1407/1407 [==============================] - 51s 36ms/step - loss:
0.2589 - accuracy: 0.9091 - val_loss: 0.3736 - val_accuracy: 0.8803
Epoch 12/15
1407/1407 [==============================] - 52s 37ms/step - loss:
0.2684 - accuracy: 0.9079 - val_loss: 0.5009 - val_accuracy: 0.8878
Epoch 13/15
1407/1407 [==============================] - 54s 39ms/step - loss:
0.2719 - accuracy: 0.9070 - val_loss: 0.3626 - val_accuracy: 0.8889
Epoch 14/15
1407/1407 [==============================] - 61s 43ms/step - loss:
0.2798 - accuracy: 0.9050 - val_loss: 0.3989 - val_accuracy: 0.8811
Epoch 15/15
1407/1407 [==============================] - 55s 39ms/step - loss:
0.2821 - accuracy: 0.9054 - val_loss: 0.4424 - val_accuracy: 0.8885
```

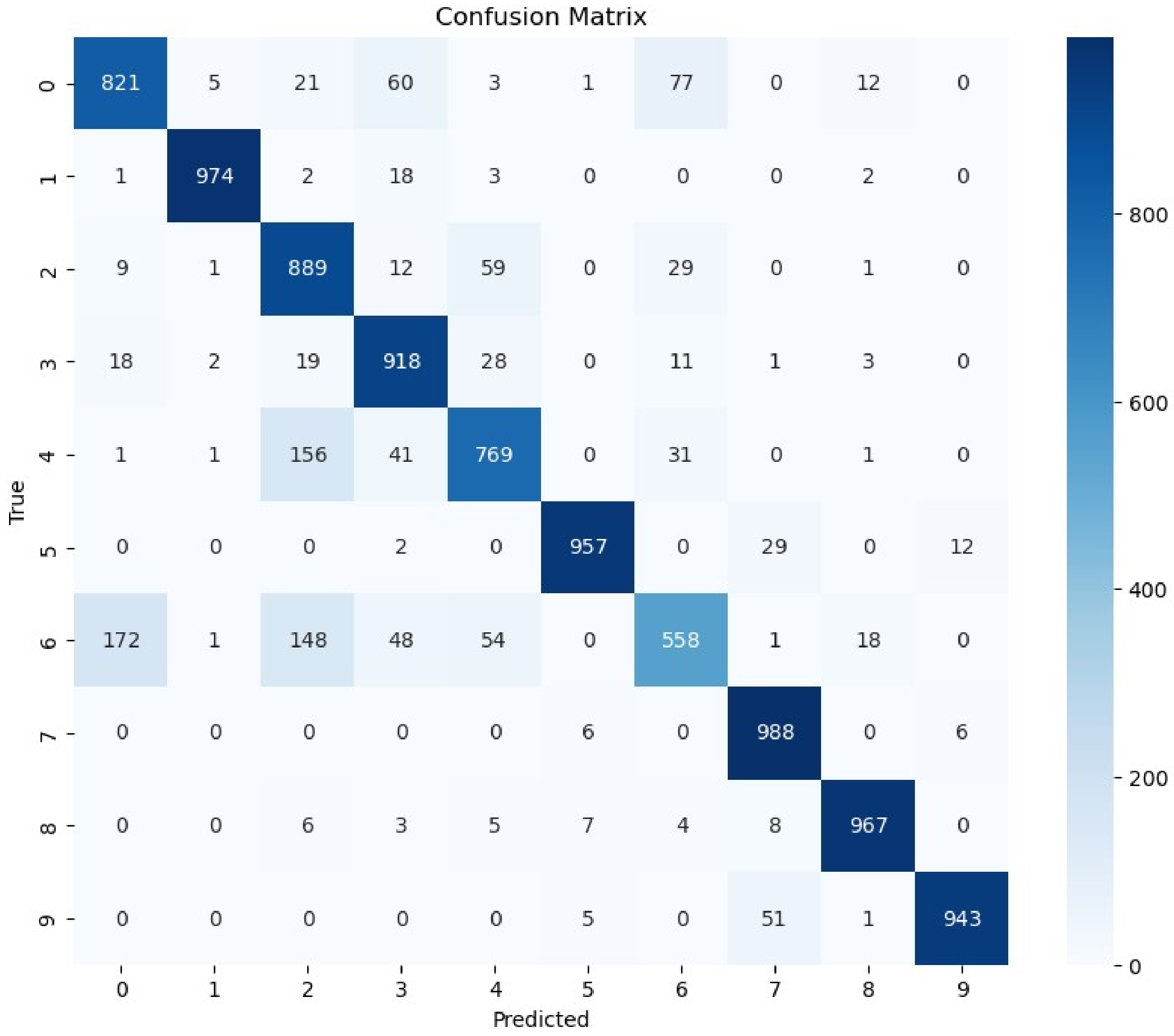

Model accuracy
Accuracy
0.80
0.82
0.84
0.86
0.88
0.90
Epoch
0
2
4
6
8
10
12
14
accuracy
val_accuracy



```
#Evaluate the model
from sklearn.metrics import confusion_matrix
import seaborn as sns
#Evaluate the model

def evaluate_model(model, x_test, y_test):
    test_loss, test_acc = model.evaluate(x_test, y_test)
    print(f"Test accuracy: {test_acc:.4f}")
    print(f"Test loss: {test_loss:.4f}")

#Predict the model
    predictions = model.predict(x_test)
    predicted_classes = np.argmax(predictions, axis=1)
    cm = confusion_matrix(y_test, predicted_classes)
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

    return test_loss, test_acc
x_train, x_val, y_train, y_val, x_test, y_test = load_dt()
```

```
test_loss, test_acc=evaluate_model(alx, x_test, y_test)
```

```
313/313 [==============================] - 4s 13ms/step - loss: 0.5185
- accuracy: 0.8784
Test accuracy: 0.8784
Test loss: 0.5185
313/313 [==============================] - 3s 11ms/step
```



```
x_train, x_val, y_train, y_val, x_test, y_test = load_dt()


#show the data
print("Training data shape: ", x_train.shape, y_train.shape)
print("Validation data shape: ", x_val.shape, y_val.shape)
print("Testing data shape: ", x_test.shape, y_test.shape)
```

```
#show the data
plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.imshow(x_train[i], cmap='gray')
    plt.title(y_train[i])
    plt.axis('off')
plt.show()
```

```
Training data shape:  (45000, 28, 28) (45000,)
Validation data shape:  (15000, 28, 28) (15000,)
Testing data shape:  (10000, 28, 28) (10000,)
```

6
8
9
3
8
0
5
6
6
7
6
9
4
0
3
1
8
7
5
8
6
9
7
3
4