IT3071 – Machine Learning and Optimization Methods

Lab Sheet 06

Part A

01) Import Required Libraries

- Import numpy, matplotlib, keras utilities, and sklearn.model_selection.
- Briefly explain the role of each (e.g., NumPy for arrays, Matplotlib for visualization, Keras for deep learning).

02) Load the Dataset

- Use the **Fashion MNIST dataset** from keras.datasets.
- Print the shapes of training and testing data.
- Why is it useful to check dataset shapes before model training?

03) Visualize a Sample

- Display the first training image with its label.
- Why is visualization of data important before training a model?

04) Data Reshaping and Normalization

- Reshape training and test images to include the channel dimension (28, 28, 1).
- Convert pixel values to float and normalize to the range [0, 1].
- Explain why normalization is important in deep learning.

05) Convert Labels to One-Hot Encoding

- Convert categorical class labels into one-hot vectors using to_categorical.
- Show an example of original vs one-hot encoded labels.

06) Create a Validation Set

- Use train_test_split to divide training data into train and validation sets (80/20 split).
- Why is validation data needed in addition to testing data?

Part B

01) Set Hyperparameters

- Define batch_size, epochs, and num_classes.
- What does each of these control in the training process?

02) Build a CNN Model

- Construct a **Sequential model** with the following:
    - Conv2D (32 filters, 3×3, linear activation, same padding) + LeakyReLU
    - MaxPooling2D (2×2)
    - Conv2D (64 filters) + LeakyReLU + MaxPooling2D
    - Conv2D (128 filters) + LeakyReLU + MaxPooling2D
    - Flatten
    - Dense (128 units, linear activation) + LeakyReLU
    - Dense (num_classes, softmax output)
- Draw a diagram of the architecture.

03) Compile the Model

- Compile the model using **Adam optimizer** and **categorical crossentropy** loss.
- Justify the choice of optimizer and loss function for this multi-class classification task.

Part C

01) Fit the Model

- Train the model on the training set with the defined hyperparameters.
- Use validation data to monitor performance.
- Plot training and validation accuracy/loss curves.

02) Evaluate on Test Data

- Evaluate the trained CNN using the test dataset.

- Report the test accuracy and test loss.
- Why might test accuracy differ from validation accuracy?

03) Predictions (Optional Extension)

- Predict labels for a few test images and visualize them with predicted class titles.
- Discuss any misclassifications observed.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from keras.datasets import fashion_mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, LeakyReLU
from keras.losses import categorical_crossentropy
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
```

Load dataset

In [ ]:
```
(train_X, train_Y), (test_X, test_Y) = fashion_mnist.load_data()

print('Training data shape : ', train_X.shape, train_Y.shape)
print('Testing data shape : ', test_X.shape, test_Y.shape)
```

Display one sample

In [ ]:
```
plt.figure(figsize=(2,2))
plt.imshow(train_X[0,:,:], cmap='gray')
plt.title("Label : {}".format(train_Y[0]))
plt.show()
```

Reshape to include channel dimension

In [ ]:
```
train_X = train_X.reshape(len(train_X), 28, 28, 1)
test_X = test_X.reshape(len(test_X), 28, 28, 1)
```

Normalize

In [ ]:
```
train_X = train_X.astype('float32') / 255
test_X = test_X.astype('float32') / 255
```

One-hot encode labels

In [ ]:
```
train_Y_one_hot = to_categorical(train_Y)
test_Y_one_hot = to_categorical(test_Y)
```

Split into train/validation

In [ ]:
```
train_X, valid_X, train_label, valid_label = train_test_split(
    train_X, train_Y_one_hot, test_size=0.2, random_state=0)
```

Hyperparameters

In [ ]:
```
batch_size = 64
epochs = 20
num_classes = 10
```

CNN Architecture

In [ ]:
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='linear',
                 input_shape=(28,28,1), padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D((2, 2), padding='same'))
model.add(Conv2D(64, (3, 3), activation='linear', padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
model.add(Conv2D(128, (3, 3), activation='linear', padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
model.add(Flatten())
model.add(Dense(128, activation='linear'))
model.add(LeakyReLU(alpha=0.1))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=categorical_crossentropy, optimizer=Adam(),
              metrics=['accuracy'])
model.summary()
```

CNN Fitting

In [ ]:
```
results = model.fit(train_X, train_label, batch_size=batch_size,
                    epochs=epochs, validation_data=(valid_X, valid_label))
```

CNN Evaluation

In [ ]:
```
test_eval = model.evaluate(test_X, test_Y_one_hot)
print("Test Loss:", test_eval[0])
print("Test Accuracy:", test_eval[1])
```