Student Name: Diego Aldo Pettorossi | utsa ID: zho125

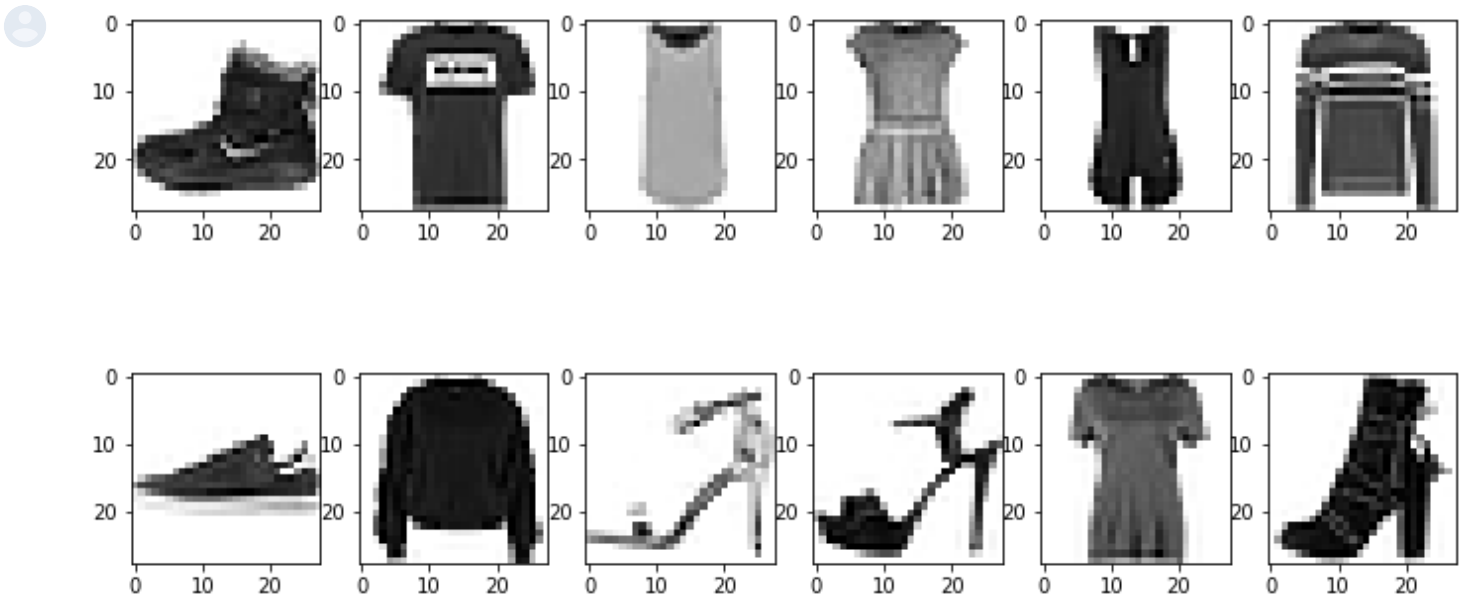
Submission Instruction: Submit a PDF file of your codes and outputs and a Google Colab shared link to your source file (.ipynb format) to Blackboard.

P1 (35pt): Write a Python code using NumPy, Matplotlib, and Keras to perform image classification for the Fashion_MINIST datset (https://github.com/zalandoresearch/fashion-mnist)

1. (5pt) Load the dataset using tf.keras.datasets.fashion_mnist.load_data() and show the first 12 images of the training dataset in two rows.

```
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.utils import to_categorical

# put your answer here
mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
plt.figure(figsize = (12, 6))
for i in range(12):
  plt.subplot(2, 6, i + 1)
  plt.imshow(train_images[i], cmap = plt.cm.binary)
plt.show()
```



2. (5pt) Add the “depth” dimension to the training/testing image data using .reshape(), use to_categorical() to transform all labels into their one-hot encoding forms, and normalize the pixel values of all images into [0, 1]. Print out the shapes of training and testing images.

- Note that the imported training/testing image data have a shape of (number_samples, image_height, image_width). You need to reshape it into the shape of (number_samples, image_height, image_width, image_depth/image_channels)

```
# put your answer here
train_images.shape, test_images.shape, train_labels.shape, test_labels.shape
```

```
((60000, 28, 28), (10000, 28, 28), (60000,), (10000,))
```

```
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

train_images = train_images / 255.0
test_images = test_images / 255.0

train_images.shape, test_images.shape
```

```
((60000, 28, 28, 1), (10000, 28, 28, 1))
```

3. (10pt) Build a CNN model using a stack of Conv2D (128 filters of size (3, 3) with ReLU activation), MaxPooling2D (pool size of (2, 2)), Conv2D (64 filters of size (3, 3 with ReLU activation), MaxPooling2D (pool size of (2, 2)), Dense (128 hidden units with ReLU activation), and output layer. Display the model architecture using .summary().

- You need to specify other parameters of the input layer and output layer.

```
# put your answer here
from keras import models
from keras import layers
model = models.Sequential()
model.add(layers.Conv2D(128, (3, 3),
                        activation = 'relu',
                        input_shape = (28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation = 'relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation = 'relu'))
model.add(layers.Dense(10, activation = 'softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 128)       1280      

 max_pooling2d (MaxPooling2D  (None, 13, 13, 128)      0         
 )                                                               

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        73792     

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0         
 2D)                                                             

 flatten (Flatten)           (None, 1600)              0         

 dense (Dense)               (None, 128)               204928    

 dense_1 (Dense)             (None, 10)                1290      

=================================================================
Total params: 281,290
Trainable params: 281,290
Non-trainable params: 0
_________________________________________________________________
```

4. (10pt) Compile and train the model for 10 epochs and batch size of 32. Set verbose = 0 during the training to compress the training progress. Draw the plot of the training accuracy w.r.t. the epoch number

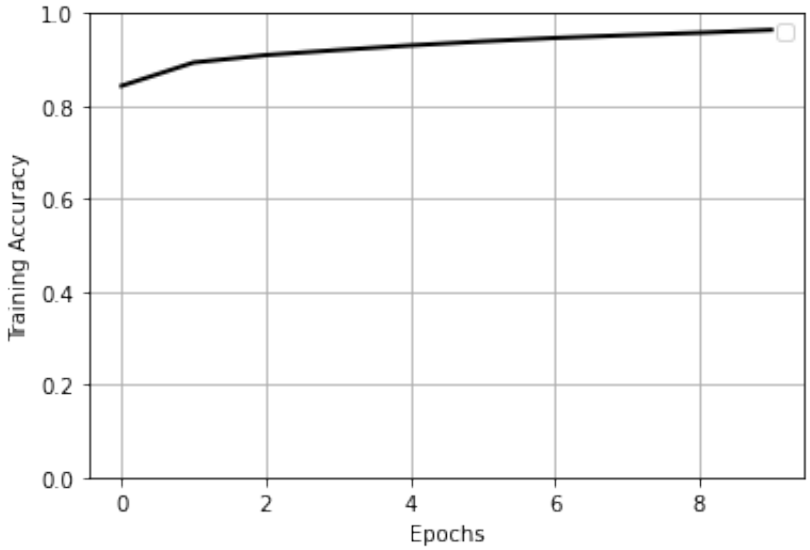
- You need to specify the right optimizer, loss function, and metrics for this task.

```
# put your answer here
model.compile(optimizer = 'adam',
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'])
history = model.fit(train_images, train_labels,
                    epochs = 10,
                    batch_size = 32,
                    verbose = 0)

plt.plot(history.history['accuracy'], c = 'k', lw = 2)
plt.xlabel('Epochs')
plt.ylabel('Training Accuracy')
plt.grid(True)
plt.ylim(0, 1.0)
plt.legend()

plt.show()
```

```
No handles with labels found to put in legend.
```



5. (5pt) Test your trained model on the testing dataset and observe the loss and accuracy using .evaluate().

```
# put your answer here
test_loss, test_accuracy = model.evaluate(test_images, test_labels)
```

```
313/313 [==============================] - 2s 5ms/step - loss: 0.3011 - accura
```

```
print('The test accuracy is: ', test_accuracy)
```

```
The test accuracy is:  0.9121999740600586
```

P2 (65pt): Write a Python code using NumPy, Matploblib and Keras to perform image classification using pre-trained model for the CIFAR-10 dataset (https://www.cs.toronto.edu/~kriz/cifar.html).

1. (5pt) Load the dataset using tf.keras.datasets.cifar10.load_data() and show the first 20 images of the training dataset in two rows.

- You will obtain the pair of feature matrix and label vector for the training dataset and the pair of feature matrix and label vector for the testing dataset at the end of this step
- Note that the CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, each with a label value within [0, 9]. In the following step, we want to partition this dataset into two training/testing pairs, one containing images with labels in [0, 4] and the other containing images with labels in [5, 9].

```
from __future__ import print_function

import tensorflow as tf
from tensorflow import keras

from tensorflow.keras.datasets import mnist,cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import backend as K
from tensorflow.keras.utils import to_categorical

import matplotlib.pyplot as plt
import random

# put your answer here
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.cifar10.load_data()

plt.figure(figsize = (20, 6))
for i in range(20):
  plt.subplot(2, 10, i + 1)
  plt.imshow(X_train[i])
plt.show()
```

```
type(X_train), type(y_train)
```

```
(numpy.ndarray, numpy.ndarray)
```

```
(y_train < 5).reshape(50000, )
```

```
array([False, False, False, ..., False,  True,  True])
```

```
X_train.shape
```

```
(50000, 32, 32, 3)
```

2. (5 pt) Reshape the label vectors in both the training and testing datasets to 1D using .reshape(), and compare them with 5 to find out the indices of images that have class labels < 5 and class labels >= 5, respectively, in the training and testing datasets.

- You will obtain four index arrays of Boolean values at the end of this step (<5 and >= 5 for training dataset and <5 and >=5 for testing dataset)
- Hint: label_vector < 5 and label_vector >= 5 will generate such indices

```
# put your answer here
y_train = y_train.reshape(50000, )
y_test = y_test.reshape(10000, )

index_train_less = (y_train < 5)
index_train_great = (y_train >= 5)

index_test_less = (y_test < 5)
index_test_great = (y_test >= 5)
```

3. (5 pt) Use the index arrays obtained in the previous step to split the training/testing dataset into two subsets (each consisting of a feature matrix and a label vector): one with class labels < 5 and one with class labels >= 5. Print out the shapes of the resulting subsets for both training and testing datasets.

- You will obtain four subsets at the end of this step: one pair of training and testing subsets of images with class labels < 5 and another pair of training and testing subsets of images with class labels >= 5.

```
# put your answer here
X_train1 = X_train[index_train_less]
y_train1 = y_train[index_train_less]
X_test1 = X_test[index_test_less]
y_test1 = y_test[index_test_less]

X_train2 = X_train[index_train_great]
y_train2 = y_train[index_train_great]
X_test2 = X_test[index_test_great]
y_test2 = y_test[index_test_great]
```

```
X_train1.shape
```

```
(25000, 32, 32, 3)
```

4. (5pt) Subtract 5 from the label vectors of the pair of training and testing subsets with class labels >= 5 so that the label vectors in this pair of subsets contains values from 0 to 4. Use to_categorical() to transform all labels into their one-hot encoding forms, and normalize the pixel values of all images into [0, 1].

```
# put your answer here
y_train2 = y_train2 - 5
y_test2 = y_test2 - 5

y_train1 = to_categorical(y_train1)
y_test1 = to_categorical(y_test1)
y_train2 = to_categorical(y_train2)
y_test2 = to_categorical(y_test2)

X_train1 = X_train1 / 255.0
X_test1 = X_test1 / 255.0
X_train2 = X_train2 / 255.0
X_test2 = X_test2 / 255.0
```

5. (5pt) Build a CNN model_1 using a stack of Conv2D (64 filters of size (3, 3) with ReLU activation), Conv2D (64 filters of size (3, 3) with ReLU activation), MaxPooling2D (pool size of (2, 2)), Dense (128 hidden units with ReLU activation), and output layer. Display the model architecture using .summary().

- You need to specify the correct hyperparameters of the input layer and output layer.

```
# put your answer here
input_shape = (32, 32, 3)
kernel_size = 3
pool_size = 2
model_1 = models.Sequential()
model_1.add(layers.Conv2D(64, kernel_size, activation = 'relu', input_shape = input
model_1.add(layers.Conv2D(64, kernel_size, activation = 'relu'))
model_1.add(layers.MaxPooling2D(pool_size))
model_1.add(Flatten())
model_1.add(layers.Dense(128, activation = 'relu'))
model_1.add(layers.Dense(5, activation = 'softmax'))
model_1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 30, 30, 64)        1792

 conv2d_3 (Conv2D)           (None, 28, 28, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 12544)             0

 dense_2 (Dense)             (None, 128)               1605760

 dense_3 (Dense)             (None, 5)                 645

=================================================================
Total params: 1,645,125
Trainable params: 1,645,125
Non-trainable params: 0
_________________________________________________________________
```

6. (10pt) Compile and train the model on the subset of training images with class labels < 5 for 20 epochs and batch size of 128. Set verbose = 0 during the training to compress the results. Draw the plot of the training accuracy w.r.t. the epoch number.

- You need to specify the correct optimizer, loss function, and metrics for this task.

```
X_train1.shape
```

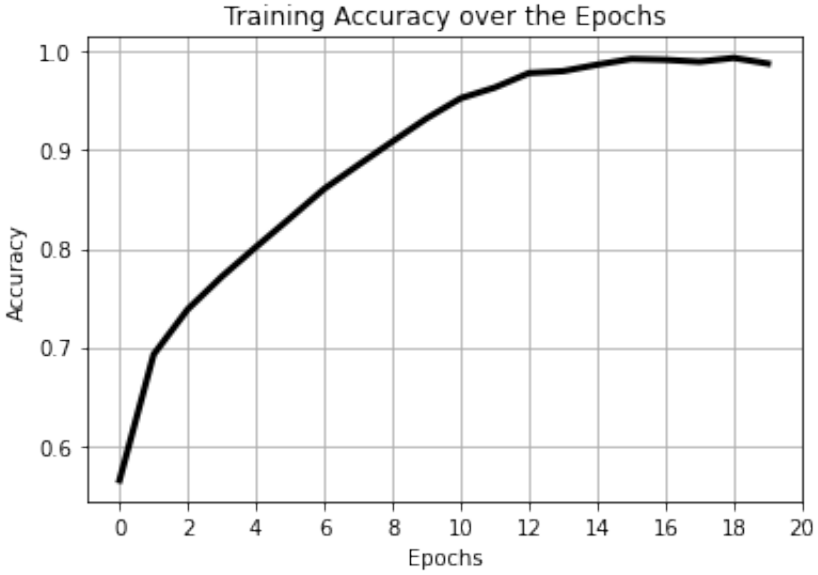
```
(25000, 32, 32, 3)
```

```
# put your answer here
model_1.compile(optimizer = Adam(lr = 0.001),
                loss = 'categorical_crossentropy',
                metrics = ['accuracy'])
model_1.fit(X_train1, y_train1, epochs = 20,
            batch_size = 128,
            verbose = 0)

plt.title('Training Accuracy over the Epochs')
plt.plot(model_1.history.history['accuracy'], lw = 3, c = 'k')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.grid(True)

plt.xticks([2 * i for i in range(11)])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWar
  super(Adam, self).__init__(name, **kwargs)
```



7. (5pt) Test your trained model_1 on the subset of testing images with class labels <5 and observe the loss and accuracy using .evaluate().

```
# put your answer here
test_loss1, test_accuracy1 = model_1.evaluate(X_test1, y_test1)
```

```
157/157 [==============================] - 1s 6ms/step - loss: 1.5570 - accura
```

```
test_accuracy1, test_loss1
```

```
(0.7495999932289124, 1.5570114850997925)
```

8. (10pt) Build a new CNN model_2 that has the same architecture as model_1 and reuse the pre-trained convolutional base layers of model_1 (i.e., all layers before applying flatten()). You need to freeze the pre-trained convolutional base layers of model_2 so that their model parameters will not be changed during the training. Display the model architecture of model_2 using .summary().

- One method to achieve the above step is as follows (You can use other methods as long as they achieve the same goal):

```
model_2 = keras.models.clone_model(model_1)

for layer in model_2.layers[:-2]:
    layer.trainable = False
```

- Other methods can be found here: https://ravimashru.github.io/100-days-of-deep-learning/days/017.html

```
# put your answer here
model_2 = keras.models.clone_model(model_1)
for layer in model_2.layers[:-2]:
  layer.trainable = False

model_2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_2 (Conv2D)           (None, 30, 30, 64)        1792      

 conv2d_3 (Conv2D)           (None, 28, 28, 64)        36928     

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 64)       0         
 2D)                                                             

 flatten_1 (Flatten)         (None, 12544)             0         

 dense_2 (Dense)             (None, 128)               1605760   

 dense_3 (Dense)             (None, 5)                 645       

=================================================================
Total params: 1,645,125
Trainable params: 1,606,405
Non-trainable params: 38,720
_________________________________________________________________
```

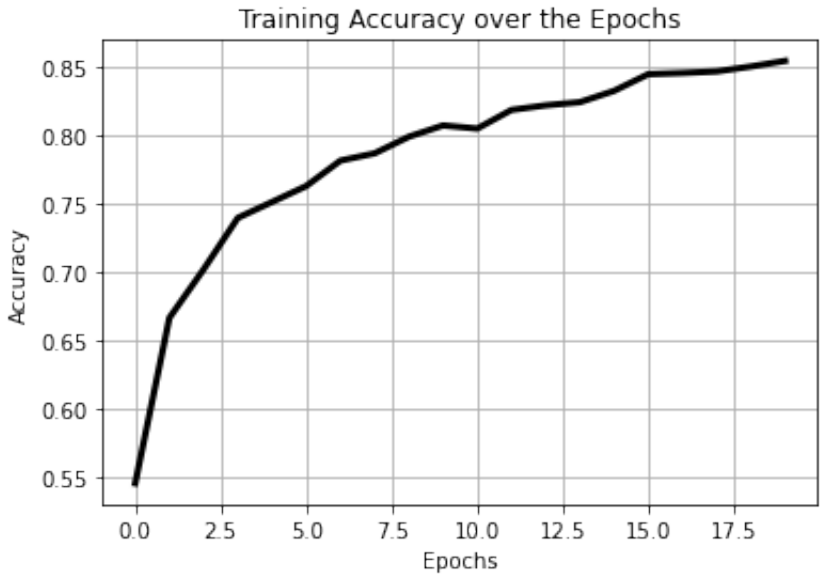
9. (10pt) Compile model_2, and train it on the subset of training images with class labels >=5 for 20 epochs and batch size of 128. Draw the plot of the training accuracy w.r.t. the epoch number.

```
# put your answer here
model_2.compile(optimizer = Adam(lr = 0.001),
                loss = 'categorical_crossentropy',
                metrics = ['accuracy'])
model_2.fit(X_train2, y_train2,
            epochs = 20,
            batch_size = 128,
            verbose = 0)

plt.title('Training Accuracy over the Epochs')
plt.plot(model_2.history.history['accuracy'], lw = 3, c = 'k')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.grid(True)

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWar
  super(Adam, self).__init__(name, **kwargs)
```



10. (5pt) Test your trained model_2 on the subset of testing images with class labels >=5 and observe the loss and accuracy using .evaluate().

```
# put your answer here
test_accuracy2, test_loss2 = model_2.evaluate(X_test2, y_test2)
```

```
157/157 [==============================] - 1s 6ms/step - loss: 0.5244 - accura
```

```
test_accuracy2, test_loss2
```

```
(0.5244149565696716, 0.8026000261306763)
```