STB600 Lab 6: Convolutional Neural Network (CNN)

Atieh Sahraeidolatkhaneh: atieh.sahraeidolatkhaneh@hv.se

Yongcui Mi: yongcui.mi@hv.se

The goal of this laboratory is to use convolutional neural network to classify images into specified groups.

1. Task 1: Prerequisites

For this task, you need to use TensorFlow an open-source library for training your model. Upload the TensorFlow to your working directory:

```
#
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
#
```

2. Task 2: Download and prepare your dataset

You are free to choose your dataset. You can choose a dataset from the TensorFlow webpage (https://www.tensorflow.org/datasets). TensorFlow Datasets is a collection of datasets ready to use. Make sure to learn about your dataset and its contents; The total number of images that it has, number of classes and the number of images in each class. The number of images in training and test sets.

```

# make sure that the pixel values are normalized
# divide the images by 255 to normalize the pixel values
```

3. Task 3: Verify the dataset

To verify the dataset, plot the first 20-25 images from the training set and display the class name of each image below each of them.

4. Task 4: Create the base convolutional network

You can use the code below as an example of the convolutional base using a common pattern: a stack of Conv2D and MaxPooling2D layers. Display the architecture of your model.

```
# replace the values with your own image dimensions and convolutional
    kernel sizes you choose for your model.
model = models.Sequential()
model.add(layers.Conv2D(Img_dimension, (3, 3), activation='relu',
    input_shape=(dimension, dimension, #channel)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

5. Task 5: Add dense layers to your model

To complete the model, feed the last output tensor from the convolutional base into one or more Dense layers to perform classification. Dense layers take vectors as input (which are 1D), while the current output is a 3D tensor. First, flatten the 3D output to 1D, then add one or more Dense layers. Choose the number of final Dense layers, based on the number of classes in your dataset. Display the architecture of your model after modifications.

```
# This code is an example
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

6. Task 6: Compile and train the network

```
# This code is an example
model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=10,
    validation_data=(test_images, test_labels))
```

7. Task 7: Evaluate the model

Plot the accuracy values of the model for train and validation sets and print the accuracy of the test set.

References