Convolutional Neural Network (CNN)

100 points

This programming assignment is to use the Keras/Tensorflow to construct, train, and test a CNN. The data set we will be using is the `CIFAR10` images that you can load from `Keras`. Read this webpage for the details about this data set. Documentation of relevant APIs used in this project can be found at the Keras/Tensorflow.

Load the data set

Use the following statements to load and preprocess the data set:

```
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
train_x = x_train.astype('float32') / 255  # normalization
test_x = x_test.astype('float32') / 255
train_y = to_categorical(y_train)   # create label vectors
test_y = to_categorical(y_test)
```

Construct an CNN using Keras/Tensorflow

Feel free to choose/try/test/find a good set of conv/pooling layers of different hyperparameters that will produce a validation set accuracy of at least **70%**. The reason we ask for 70% accuracy is that we don't have much hardware resource and time to wait for the finish of the training of a very large (deep) CNN that will produce high accuracy. This homework is to serve as the proof of concept on how a CNN can be quickly constructed and then trained/used using an existing ML framework for real-world image classification.

Use the following statements to load the relevant framework packages:

```
from tensorflow.keras import layers
from tensorflow.keras import models
```

We use a sequential model:

```
model = models.Sequential()
```

You will need to specify the input image/volume size for the first conv layer. Below is an example from the CNN that we constructed during class time for the MNIST data set. You need to change the input image size so that it can work with the new data set. (We are not sharing the complete notebook constructed during the class time for the MNIST data, in order for you to get more opportunities to read and learn the Keras APIs.)

```
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 2
8, 1)))
```

The Keras framework will automatically uses the output volume size of a previous layer as the input volume size of the follow-up conv layer, so you don't need to specify the input volume size of follow-up conv layers.

A typical max pooling layer can be added using a statement like the follow example:

```
model.add(layers.MaxPooling2D((2, 2)))
```

After choosing a set of conv/pooling layers, you will also need to add:

- one flatten layer: `model.add(layers.Flatten())`
- one or two hidden dense layer(s): `model.add(layers.Dense(64, activation='relu'))`.
- one dense output layer: `model.add(layers.Dense(10, activation='softmax'))`.

Note:

- You can try and decide the size of the dense layers.
- The `CIFAR10` data set has a total of **10 classes**.
- Every conv layer and the hidden dense layer(s) use the `relu` activation function. The output layer uses the `softmax`.
- You can use `model.summary()` to get a summary of your NN.

Browse through the Keras/Tensorflow website whenever you want to know and use a relevant API, and make sure you know the meaning of the API's parameters.

Compile, train, and test your model

- Compile your model.

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

- Train and test your CNN using the training/test data sets.

```
history = model.fit(
    train_x,
    train_y,
    batch_size=64,
    epochs=5,
    validation_data=(test_x, test_y),
)
```

Save your model

Use the following statement to save your trained model

```
model.save('keras_CNN_CIFAR10.model')
```

Misc.

If you want to, you can use read and plot things saved within the saved `history` to visualize out and see what the accuracy/error trend looks like during the training.

Submission

Your submission will include:

- A single **ONE** notebook file, named as `keras_CNN_CIFAR10.ipynb`. Your notebook must **include all of its output**. The grading of your work is to check your code and its output. Please know that the TA and the instructor won't have time to re-run your notebook, so your submitted notebook must include its output that shows the accuracy of your CNN in its predication for the training set as well as the validation set.

- The saved model folder `keras_CNN_CIFAR10.model`.
- Compress the notebook and the model folder together into one zip file, named as `keras_CNN_CIFAR10.zip`.
- Submit the zip file.