Task

In this programming assignment, you will implement a CNN for an image classification task on the CIFAR-10 dataset.

The CIFAR-10 image dataset contains 60k 32×32 color images of objects for 10 classes as shown in Figure 1. The dataset consists of 50,000 training images and 10,000 testing images. Each data point is a 32x32x3 color image falling into one of the 10 categories. Detailed information on CIFAR-10 can be found here.

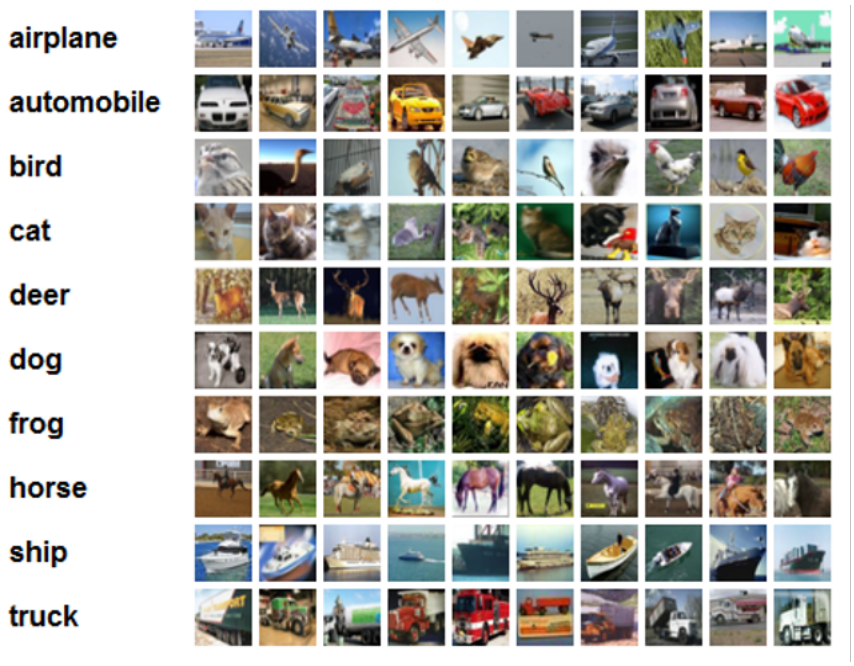

Figure 1: CIFAR-10 sample images and their labels
Source: http://www.cs.toronto.edu/~kriz/cifar.html

Implementation outlines

Data loading

We have already downloaded the data from the website and selected a subset for this programming assignment. Please use the provided data.

We provide $50,000$ training images and $5,000$ testing images. We will evaluate your model with the remaining testing images which are unobtainable to you. The data are stored in the NumPy array format and can be loaded with the following code:

```
import numpy as np
```

```
training_data = np.load('training_data.npy')
training_label = np.load('training_label.npy')
```

Please preprocess the data by dividing the intensities of each image by 255 and converting the label with one-hot encoding.

Convolutional layers

The students are expected to build a model, which consists of three convolutional layers, with functions below:

```
tf.keras.layers.Conv2D(filters, kernel_size, padding, name)
tf.keras.layers.MaxPool2D(pool_size, strides, padding, name)
```

Detailed information of each Conv layer and pooling layer is provided below:

1. Convolutional layer1: 16 5×5 filters. No zero padding with the stride as 1.
2. Max pooling layer1: The pooling area is 2×2 with the stride as 2.
3. Convolutional layer2: 32 5×5 filters. No zero padding with the stride as 1.
4. Max pooling layer2: The pooling area is 2×2 with the stride as 2.
5. Convolutional layer3: 64 3×3 filters. No zero padding with the stride as 1.

Note that ReLU activation is applied to all convolutional maps.

Fully connected layer and the loss

The students are expected to connect the output of convolutional layers with a fully connected layer, which has 500 nodes, with a ReLU activation function. The output layer has 10 nodes, resulting from applying softmax functions. Functions such as

```
tf.keras.layers.Dense(output_units, activation, name)
tf.keras.layers.Flatten()
tf.nn.softmax_cross_entropy_with_logits()
```

may help you construct the model and compute the loss.

Backpropagation

The students are allowed to use tf.GradientTape to compute gradients. The refer to the following implementation:

```
with tf.GradientTape(persistent = True) as T:
    output = your_model.forward(input_data)
    loss = compute_loss_function(output, input_labels)
model_gradient = T.gradient(loss, your_model.trainable_variables)
your_optimizer.apply_gradients(zip(model_gradient, your_model.trainable_variables))
```

Model definition and saving

Please define the model in the same format with the following implementation:

```
class your_model(tf.keras.Model):
    def __init__(self, your_arguments):
        super(your_model, self).__init__()
        self.conv_layer_1 =  # declare conv layers and pooling layers

    def forward(your_input):
        # forward propagation
        return outputs
```

Then use the following code to save the trained model:

```
your_model.save("trained_model")
```

Please do not change the file name.

Supplement instruction of programming assignment 3

In this section, we provide more details about our expectations of your implementation and reports.

Implementation/code

In Programming assignment 3, any function in Tensorflow2.X is allowed as long as the students believe that it will help them train the models. However, we expect the students to follow the following instructions carefully:

- The model structure is fixed.
  We decide to unify the model structure due to the fairness in evaluating the model performance. However, the students are encouraged to explore the influence of optimization methods, batch size, and other hyper-parameters.

- Please do not commit plagiarism.
  Please do not copy codes from the internet, code examples from previous classes, and each other.

- Please be specific about the question.
  From TA's experience with PA2, students expected the TA to go through their implementation and found the bugs. It is neither efficient nor effective. Debugging the code and identifying the issues are good practice for students to learn how to apply the deep learning techniques. Please try to debug your code and identify the possible issues first, then come to TA with an actual question.

- Please start early.
  Please start early on the programming assignments. If you need extra help from TA, please let her know in advance. It is not a good idea to ask for emergent meeting a few hours before DDL.

Reports

A detailed technical report is required as well as the implementation. The majority of the scores for this assignment depend on the goodness of the report. A large penalty will be applied if the report is handwritten. We suggest the students use Latex or Markdown to write the report.

Things should be included in the report

- Problem statement: A brief introduction to the problem you are going to solve. Please summarize the task in your own words. Do not copy the sentences from the instruction.

- Model description: Clearly describe your model including the structure. (What are the components of the model? What are the map sizes? What are the activation functions? Please include an illustration of the architecture.)

- Plot the figure of training and testing losses versus epochs. Plot the figure of training and testing accuracy versus epochs.

- Report classification error for each class (no plot) on the test dataset. Report the average classification error on the test dataset.

- Visualization of the filters in your first convolutional layer. Please refer to example in Figure 2.

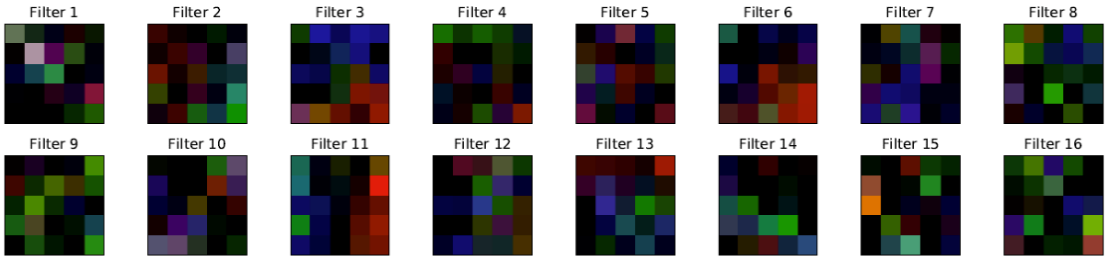

Figure 2: The visualization of the 16 5×5 filters in the first convolutional layer of the model. The images in this Figure are merely examples. Please do not regard them as the weights of a trained model.

- Discussion (Not only summarize the empirical results but also describe the influence of hyper-parameters on model performance.)

Submission

Things need to be included in the submission (the TA prefers separate files instead of a zip file.):

- A report (Well-written, in the format of a technical report).
- Code (Please submit .py file instead of .ipynb file.)
- Saved model (Use the given code. Please do not change the given file name.)

Grading policy

Programming Assignment 3: Total 100 points.
Report: 65 points
Problem statement, 10 points.
Model description, 10 points.
Experiment results, 30 points: 5 points for training losses and testing losses versus epochs; 5 points for training accuracy and testing accuracy versus epochs; 10 points for classification errors for each digit versus epochs; 5 points for final test classification average error; 5 points for weights visualization.
Discussion, 10 points.
5 points for the quality of writing.

Code: 20 points
No bug and no error. The output results are consistent with your reported results.

Results on validation dataset: 15 points
validation accuracy above 70%, 15 points.

Contact Method

If you are still confused about the instruction and supplement materials. Please send emails to the TA (yinn2@rpi.edu) or come to her office hour. Please make good use of TA OH.