Introduction

In this part, you are going to design a CNN to perform classification on MNIST and CIFAR10 dataset. The MNIST dataset has been included in the `mnist` folder. The CIFAR10 dataset is too large to be included, so you need to download it yourself from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz, decompress it and place the folder `cifar-10-batches-py` into `cifar10` folder.

You will most likely use `Conv2D`, `Dense`, `MaxPooling2D`, `Dropout` and `Activation` layers to construct the network. Please read https://keras.io/layers/convolutional/#conv2d, https://keras.io/layers/core/#dense, https://keras.io/layers/pooling/, https://keras.io/layers/core/#dropout, and https://keras.io/layers/core/#activation. As you have seen in Part 1, you'll need an optimizer and a loss function to train the network. Please read https://keras.io/losses/ and https://keras.io/optimizers/.

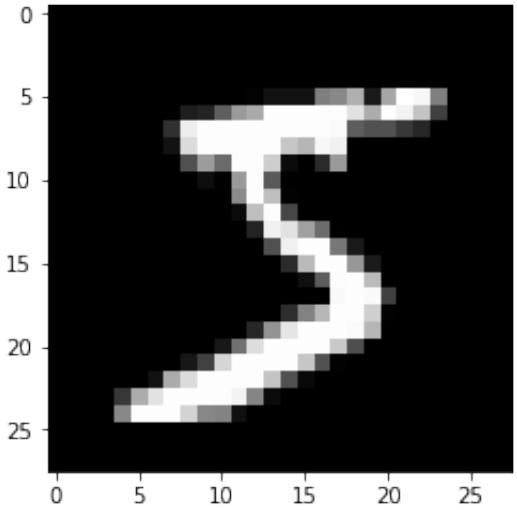
We replaced some parts of the code with ??? and ..., it is your job to read keras' documentation and figure out what happened here.

Loading MNIST and CIFAR10 Dataset

The MNIST loader

The MNIST reader function is in `assign2_utils_p2.py` as `mnist_reader()`. The loaded data can be directly used in training. An example of using it:

```
from assign2_utils_p2 import mnist_reader
train_x, train_y, test_x, test_y, class_name = mnist_reader()
import matplotlib.pyplot as plt
plt.set_cmap('gray')
plt.show(plt.imshow(train_x[0,:,:,0]))
print('The label is {}'.format(class_name[list(train_y[0]).index(1)]))
```

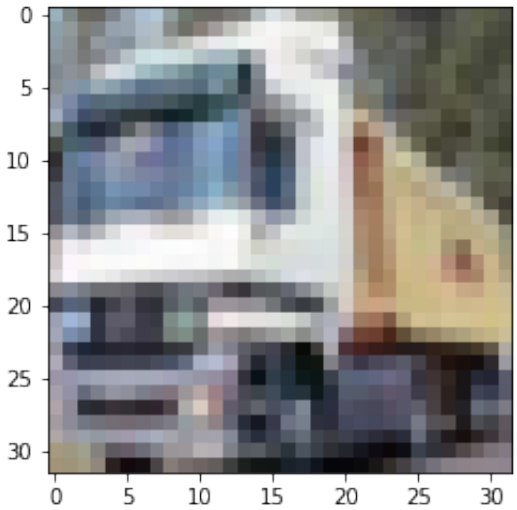



```
The label is 5
```

The CIFAR10 loader

The MNIST reader function is in `assign2_utils_p2.py` as `cifar10_reader()`. An example of using it:

```
from assign2_utils_p2 import cifar10_reader
train_x, train_y, test_x, test_y, class_name = cifar10_reader()
plt.show(plt.imshow(train_x[1,:,:,:]))
print('The label is {}'.format(class_name[list(train_y[1]).index(1)]))
```



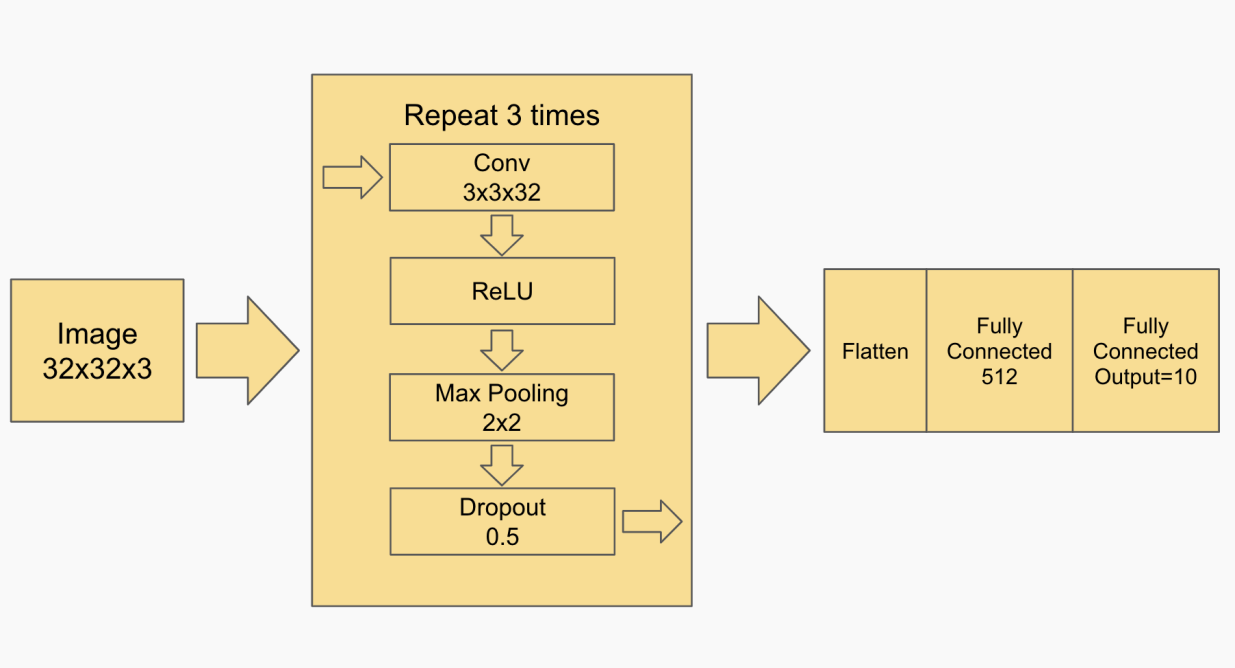

```
The label is truck
```

The Problem Set

Task 1: Train a small three-layer CNN

In this task, you are to train a CNN defined by us. The network structure is described as follows:

- It gets a batch of input with size BxHxWxC, where B is the batch size, (H,W) is the shape of input image (H=W=28 for MNIST and 32 for CIFAR10), C is the number of channels (C=1 for grayscale images, 3 for RGB colored images).
- The CNN has three identical computation blocks. Each of them has a group of 3x3 convolution filters, and produces 32 output feature maps. The padding is "valid". The output feature maps are activated by ReLU function. The output is then fed into a 2x2 max-pooling layer. The padding is also 'valid'. Finally each of the blocks has a dropout layer, which randomly set its input units to zero with a probability of 50%
- After the three blocks, the data is flattened into an 1-D vector. Then the vector is fed into a fully-connected layer, which has 512 output neurons. The data then goes through the final fully-connected layer which outputs 10 scores.
- The scores are processed by 'softmax' function, so that we can use cross-entropy loss.



There is an incomplete definition `example_network` in `example_network.py`, you are to complete it. When you finish network definition, the following code shows a possible way of training it.

```
from keras import optimizers, losses
from example_network import example_network
model = example_network(input_shape=(32,32,3))
sgd = optimizers.???(...)
model.compile(...)
model.fit(x=train_x, y=train_y, epochs=1, verbose=1)
loss, acc = model.evaluate(x=test_x, y=test_y)
print('Test accuracy is {:.4f}'.format(acc))
```

If you write the code correctly, you should get a test accuracy of at least 92% on MNIST and 41% on CIFAR10.

You should do the follow for this task:

1. Complete network definition in `example_network.py`
2. Train the CNN on MNIST and CIFAR 10, write about its accuracy on test set in the report.
3. Save your **model structure as a json file** and submit it. Read the *Saving/loading only a model's architecture* part at https://keras.io/getting-started/faq/#how-can-i-save-a-keras-model.

Task 2: Train your own network

Though quite high, ~92% is not even close to the best result we could get on MNIST dataset. In this task, you are to design your own CNN. Be creative and tune as much as possible, try train a network that outperform the model in Task 1, both on **MNIST** and **CIFAR10**!

You should do the follow for this task:

1. Define your own network, write your design intuition in the report.
2. Train the CNN on MNIST and CIFAR 10, write about its accuracy on test set in the report.
3. Save your **whole model** and submit it. Read the *Saving/loading whole models (architecture + weights + optimizer state)* part at https://keras.io/getting-started/faq/#how-can-i-save-a-keras-model. **The maximum size of the model is 25MB, any model larger than this is considered invalid.**

Task 3: Try advanced networks (optional)

Keras offers a variety of pre-trained models that can be called with a few lines of code. You are strongly encouraged to read https://keras.io/applications/ and try some of those models. Use it to do whatever you like and think about why it is designed so. Note that if you want to train those model from scratch, a GPU is recommended.

You are not required to submit anything for this task. However, if you like, you may write something in the report or share with us in the sharing session.

Hints and Notes

1. There's no need to do Task 1 and Task 2 on GPU machines.
2. The code may run for 30+ seconds, depending on for how many epochs you prefer the model to be trained.

Submit Your Results

We've provided an example folder named `Axxxxxxxx` in the homework package. Rename this folder to your matriculation number. Inside this folder you can see all the example of files that you need to submit. Here's a list of things to submit:

For Part 1

1. A report `Axxxxxxxxx/part1/Axxxxxxxx.pdf`, whose length should be less than two pages (including figures and references).
2. A file `Axxxxxxxxx/part1/Axxxxxxxx.pkl` containing your answer to Part 1. The structure of this file has been explained in Part 1.
3. Any source code you wish to present in `Axxxxxxxxx/part1/source/`. No file name limits.

For Part 2

1. A file `Axxxxxxxxx/part2/Axxxxxxxx.json` containing your model structure in Part 1 - Task 1.
2. A file `Axxxxxxxxx/part2/Axxxxxxxx_mnist.h5` containing your trained model in Part 1 - Task 2 for model trained on MNIST, and another file `Axxxxxxxxx/part2/Axxxxxxxx_cifar10.h5` for the same model but trained on CIFAR10.
3. A report `Axxxxxxxxx/part2/Axxxxxxxx.pdf`, whose length should be less than two pages (including figures and references).
4. Any source code you wish to present in `Axxxxxxxxx/part1/source/`. No file name limits.

When you are ready

Zip the folder `Axxxxxxxx` into a **zip** archive `Axxxxxxxx.zip`, and upload it to IVLE.

Your submission will be graded by an auto-grading program. **All file and variable names are case-sensitive. Please strictly follow the file name format, directory and data structure. There will be 0 mark for corrupt submissions.**