National Taiwan University **Introduction to Machine Learning and Deep Learning**
Department of Civil Engineering Instructor: C.-S. CHEN

Homework 5
Due 21:00, Thursday, December 8, 2022

Late submission within 24 hours: score*0.9;
Late submission before the post of solution: score*0.8 (the solution will usually be posted within a week); no late submission after the post of solution)

Total 90% + 20% Bonus
If you have any question in this homework, please contact TA 江郁瑄 (r10521605), and 李俊昇 (r11521608)

1. **(50%)** Convolutional filters play a key role in CNNs, and we can use packages provided by `SciPy` to see their effects. But before we can proceed further, we need to know a few basic tricks for Image IO using `matplotlib`:

   Read and display an image using matplotlib

   The following code blocks show how to use `imread()` function from `matplotlib.image` to read an image into a `numpy` array. The pixel values are between 0 and 255.

```
%matplotlib inline
import numpy as np
import matplotlib.image as mpimg
import matplotlib.pylab as plt

im = mpimg.imread("images/currency.tif")
```

   We can print the shape and type of `im` which is a `numpy` array:

```
print(im.shape, type(im))
(500, 1192) <class 'numpy.ndarray'>
```

   We can also print some contents of `im`:

```
print(im[100,350:380])
[215 212 208 207 212 208 203 212 214 173  98 103 188 209 176 102  32  46
  88  85  80  91  63  44  81 126  88  51 110 172]
```

   To display the image, we can use `imshow()` function from `matplotlib.pylab`:

```
plt.imshow(im, cmap='gray')
```

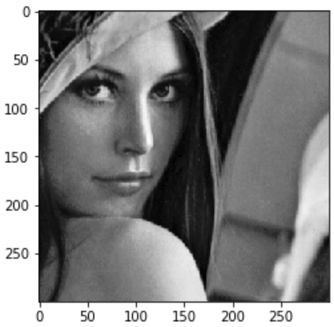
Convolution with SciPy signal's convolve2d

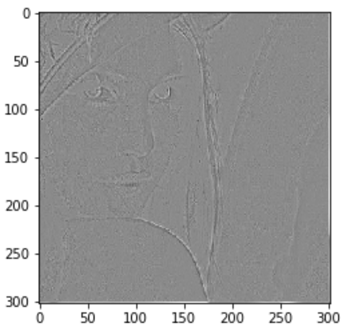
The SciPy signal module's `convolve2d()` function can be used for convolution. The following code blocks show how to use the function to perform a simple convolution:

```
import numpy as np
from scipy.signal import convolve2d as conv2
image = np.array([[1, 3, 0, -1],
                  [-2, 0, 3, 2],
                  [2, -3, -1, 4],
                  [4, 2, 0, 1]])
kernel = np.array([[2, 0], [-1, 3]])
output = conv2(image, np.flip(kernel), mode ='valid')
print(output)
```

```
[[  4  15   3]
 [-15   0  19]
 [  6  -8   1]]
```

(a) (30%) Name your Jupyter notebook `im_filter.ipynb`. Apply the above concepts and codes to read an image `lenna-gray.tif` in HW5 Supplementary.zip available from the course website and use the edge-detection kernel covered in the class to amply the edges. Below are the original and the convolved images:





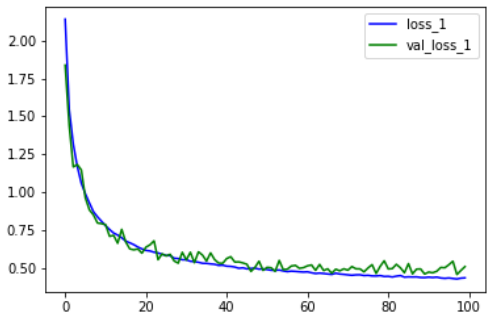
(b) (20%) Name your document `Filter_Report.pdf`. Take some photos from your phone, convert them into grayscale and apply some filters to amply their features. Document your efforts.

2. (40%) In many occasions, you will need to resume and continue your training or keep the history log. Please consult the references given at the end of this document. We will follow the scenario described below to practice this exercise. It goes like this:

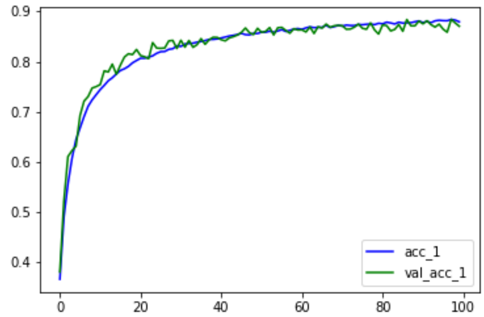
One day, TA's 200-epochs-long training task was interrupted at epoch 100 accidentally. The whole training progress was gone except the log (`training.csv`), and the model state at epoch 100 (`Prob2_epoch_100.h5`) were saved. Both files are in HW5 Supplementary.zip, available from the course website.

(1) Name your Jupyter notebook `Plot_Learning_Curve.ipynb`. Use the provided log file (`training.csv`) to plot the learning curves of model training and validation accuracy and loss, and save the figure of loss curves with the name: `problem2_1_loss.png` and the figure of accuracy curves with the name: `problem2_1_acc.png`.

Notice: You should plot the legend to specify training and validation curves. Sample plots are given below.



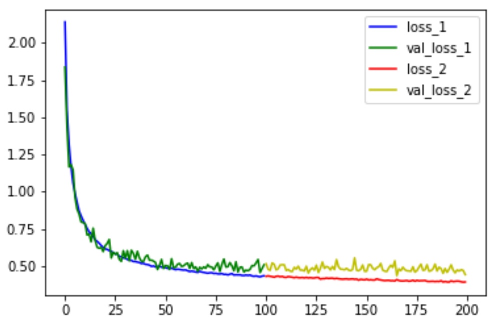
`problem2_1_loss.png`



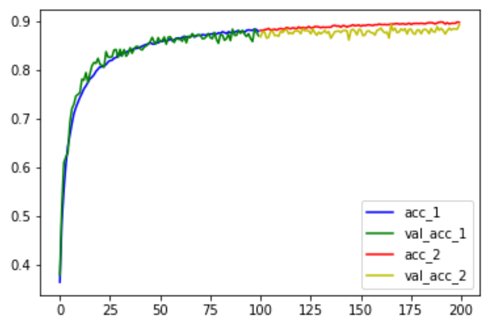
`problem2_1_acc.png`

(2) Follow `HW5_Cifar10.ipynb` in HW5 Supplementary.zip available from the course website and complete all the TODO to load the provided model file (`Prob2_epoch_100.h5`) and continue training another 100 epochs. Then concatenate the training history with the provided log file (`training.csv`) to plot the complete learning curves of model accuracy and model loss with 200 epochs. Then save the figure of loss curves with the name: `problem2_2_loss.png` and the figure of accuracy curves with the name `problem2_2_acc.png`.

Notice: You should plot the legend to specify two-stage training and validation curves. Sample plots are given below.



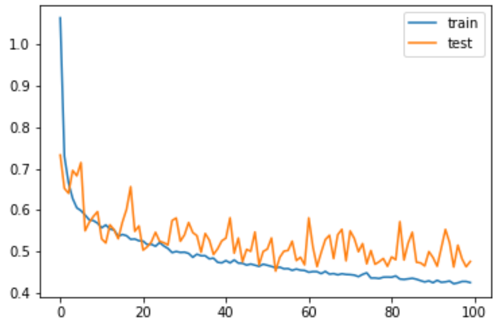
`problem2_2_loss.png`

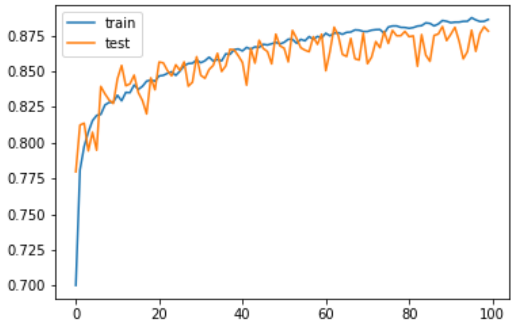


`problem2_2_acc.png`

3. **(Bonus, 20%)** The idea of transfer learning is to transfer the parameters of a trained model to another new model so that we do not need to start from scratch, or we can start with better weight values. In this problem, we will provide a pre-trained weight of a basic CNN model and a new CNN model structure.

Follow the `HW5_Cifar10_bonus.ipynb` in HW5 Supplementary.zip available from the course website and complete all the TODO to load the provided weights to a new CNN model and retrain 100 epochs from these initial weights. Plot the learning curves of model training and validation accuracy and loss with 100 epochs, and save the figure of loss curves with the name: `problem3_loss.png` and the figure of accuracy curves with the name: `problem3_acc.png`.





- **Ref:**
  https://www.tensorflow.org/api_docs/python/tf/keras/Model
  https://www.tensorflow.org/api_docs/python/tf/keras/callbacks
  https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint
  https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/CSVLogger

- **Submission Format:** You will need to submit `im_filter.ipynb`, `Filter_Report.pdf`, `Plot_Learning_Curve.ipynb`, `problem2_1_loss.png`, `problem2_1_acc.png`, your revised `HW5_Cifar10.ipynb`, `problem2_2_loss.png`, `problem2_2_acc.png`, (bonus) your revised `HW5_Cifar10_bonus.ipynb`, `problem3_loss.png`, `problem3_acc.png`. Please compress the all files into `yourStudentId_hw5.zip`, then upload it to NTU COOL.