Assignment 4
Convolutional Neural Network

Yun-Yang Huang
Po-Chih Kuo

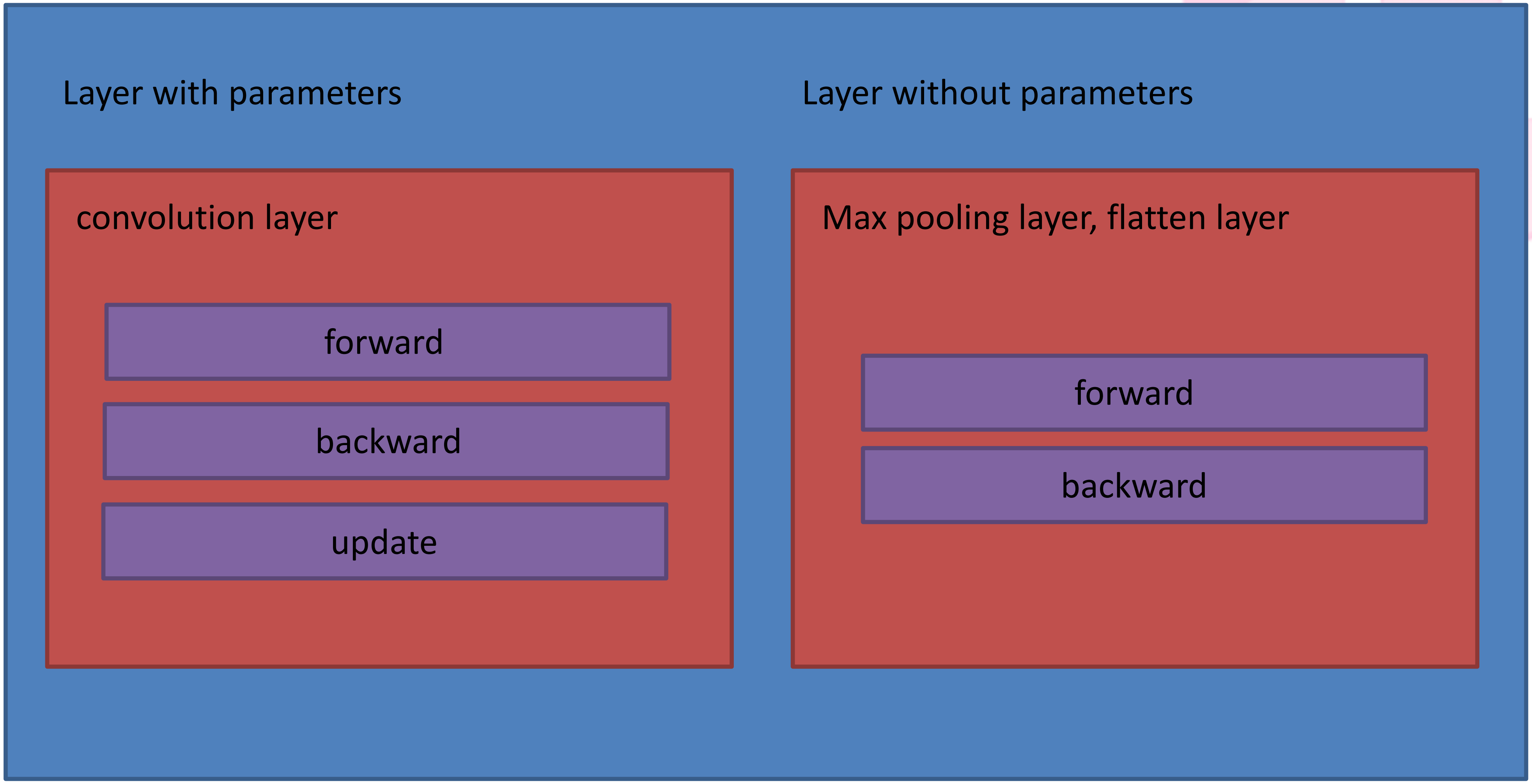
Goal

- Build your own convolutional neural network step by step
- Extend your previous NN to “C”NN
- Implement certain functions required to build a convolutional neural network
- Understand the how convolution layer and max pooling layer work, including forward propagation, backward propagation and update.
- Build a convolutional neural network to predict the pulmonary disease of patients from their chest X-ray (CXR) images.
- Be familiar with existing DL tools (TensorFlow)

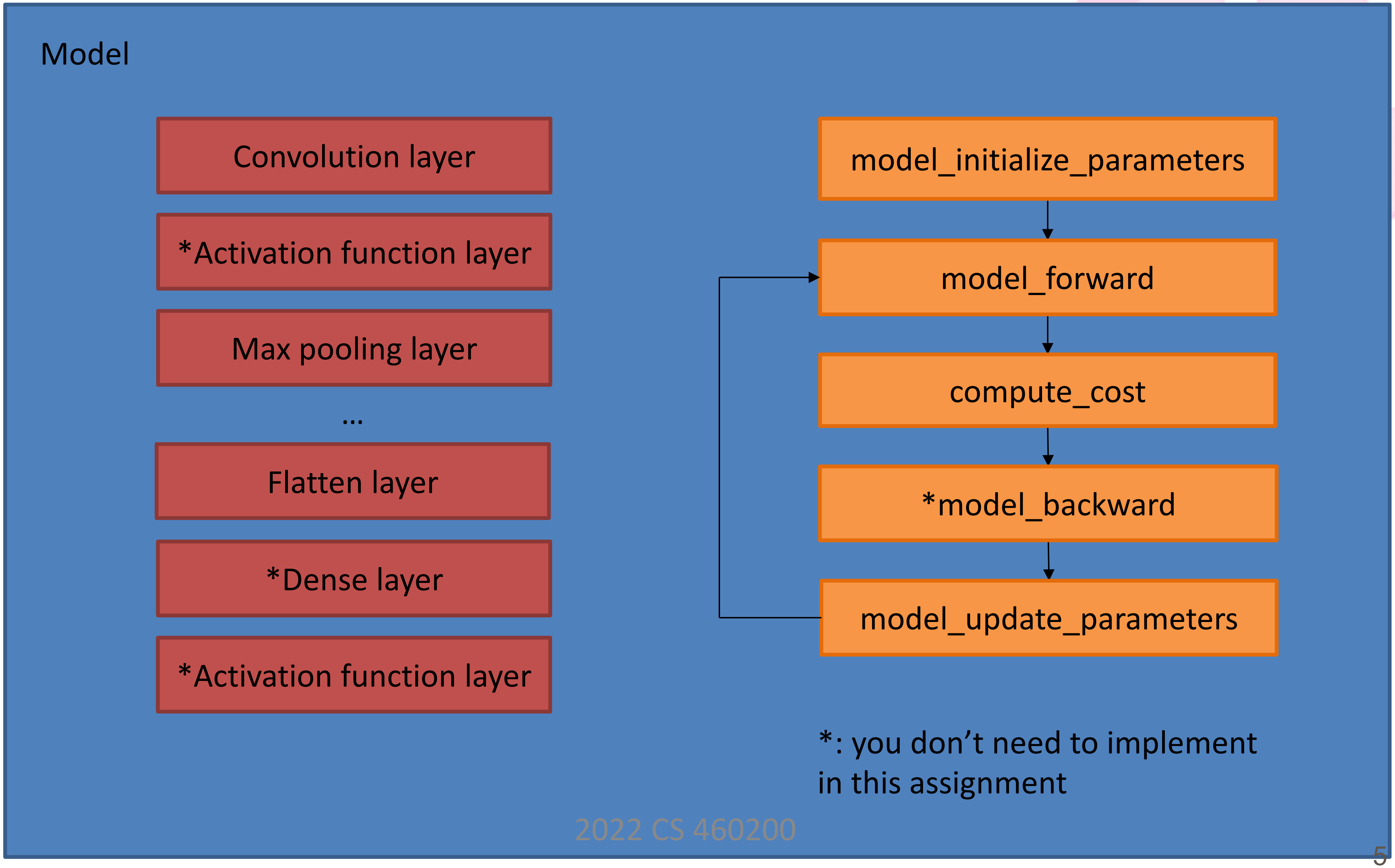
Grading Policy

Item	Score
Basic Implementation	65%
Advanced Implementation	25%
Basic & Advanced Report	10%

Overview


Layer with parameters
convolution layer
forward
backward
update
Layer without parameters
Max pooling layer, flatten layer
forward
backward

Overview

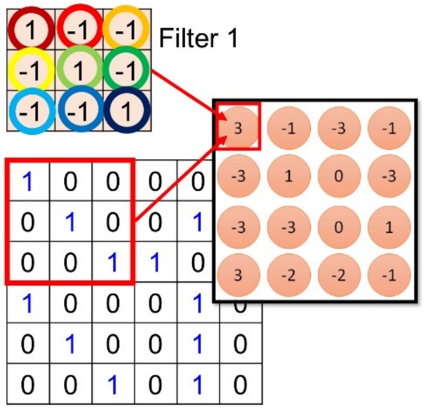


*: you don’t need to implement in this assignment

Basic Implementation (65%)

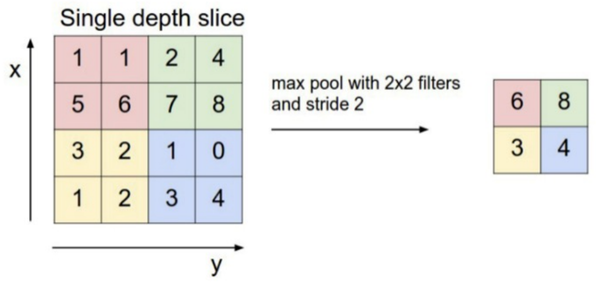
Convolution layer

1. Initialize the weight of the layer (1%)
2. Implement Zero-Padding function. (3%)
3. Implement conv_single_step() function. (5%)
4. Implement forward pass. (15%)
5. Implement convolution update parameters. (1%)



Max pooling layer

1. Implement forward pass. (15%)



Flatten layer

1. Forward pass (5%)
2. Backward pass (5%)

Basic Implementation (65%)

Model

1. Implement model forward, backward and update. (5%)

Designing a binary classifier (10%)

1. Implement a binary classifier and try to get a good performance.
2. You can only use the functions you implement in the previous step.
3. You will get all 10% if your prediction achieves accuracy greater than 0.55 in the testing set.

Basic Implementation (65%)

- For the basic part, if you want to use any other packages that are used for **speeding up** your training, you must ensure that the final output you submitted must match the output format we specify in the ppt, or otherwise, zero marks will be given for that code implementation.

- If you use any other packages, you must write the reason why you want to do so in the report, where you use them and the result of using these packages.

- Although you don't need to implement the backward pass of the convolutional neural network, you still can try to understand it. It will give you a deeper understanding of deep learning.

Advanced Implementation (25%)

Designing a binary classifier in **TensorFlow**

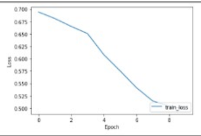
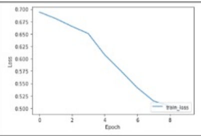
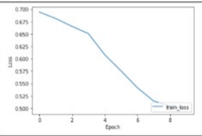
1. Build a convolutional neural network in TensorFlow.
2. You can import any packages in this part.
3. You will get 15 % if your prediction achieves accuracy greater than 0.7 in the testing set.
4. The rest 10 % will be graded by your rank.
5. Some TensorFlow tutorials:

   There is no limitation!!!

   https://www.tensorflow.org/tutorials/images/cnn
   https://www.tensorflow.org/tutorials/images/data_augmentation
   https://www.tensorflow.org/tutorials/images/transfer_learning

Report (Basic and Advanced) (10%)

- Do a simple experiment to find out whether your convolutional neural network exceeds your linear model (under the same number of epochs). And below are some evaluation metrics you need to finish:
  - epochs(2%)
  - training time (1%)
  - accuracy (both training and valid) (1%)
  - number of parameters (2%)
  - training loss curve (2%)
  - others(optional)

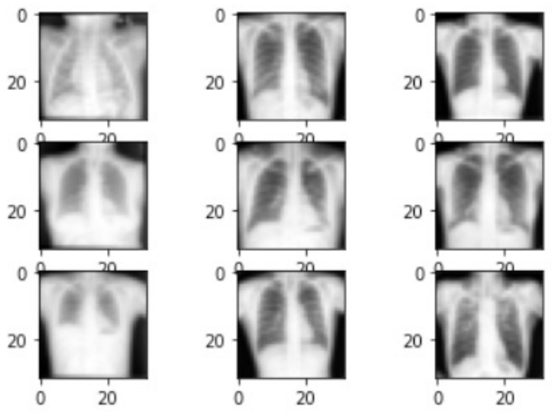
	linear model	your CNN model	TensorFlow CNN model
epochs	10	10	10
training time	1 hour 10 min	1 hour 30 min	20 min
accuracy	0.75	0.80	0.85
num of parameters	100	10000	10000
training loss curve			
others			

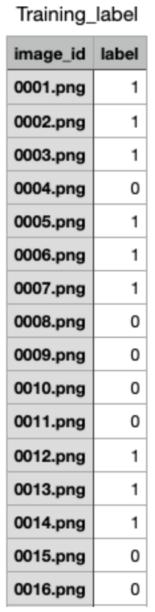
- For the advanced part: Describe how you design or choose your own model architecture, and how you choose loss function and optimizer. (2%)
- Do not exceed 2 pages!

Data (Basic and Advanced)

Binary classification: Chest X-rays images

1. You will get 600 samples as training data. (300 normal and 300 abnormal) and 60 samples as testing data.
2. Use the training data to predict whether the patient in test data is normal or not.
3. 0:normal; 1:abnormal
4. The first column of the Traing_lable.csv file corresponds to the file name of the images.
5. The shape of X_train: (600, 32, 32, 1); shape of y_train: (600, 1)
   X_test(60, 32, 32, 1)




Training_label

image_id	label
0001.png	1
0002.png	1
0003.png	1
0004.png	0
0005.png	1
0006.png	1
0007.png	1
0008.png	0
0009.png	0
0010.png	0
0011.png	0
0012.png	1
0013.png	1
0014.png	1
0015.png	0
0016.png	0

Items for you

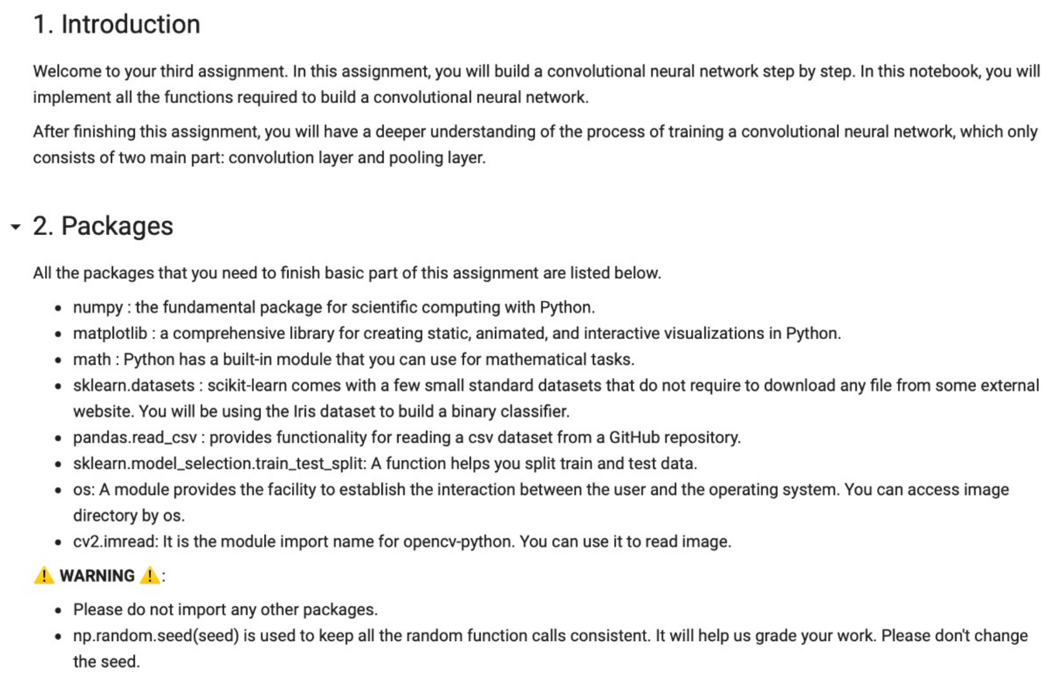
- Template: **hw4_template.ipynb**
- Some files: **Dense.py, Activation.py, Loss.py, Predict.py**. You can paste the code you wrote in HW3 into these py files and import these files to ipynb file. Bellows are the functions you need to copy-and-paste:

Dense.py class Dense():	**Activation.py class Activation():**	**Loss.py**	**Predict.py**
initialize_parameters()	forward()	compute_BCE_cost()	
forward()	backward()		
backward()			
update()			

- Training data: **Training_data.zip**, **Testing_data.zip**, **Training_label.csv**
- Sample output: **sample_output.npy**

Template

Except for the imported packages in the template, you cannot use any other packages (ex: tqdm).
Remember to save the code file to **hw4.ipynb**



1. Introduction

Welcome to your third assignment. In this assignment, you will build a convolutional neural network step by step. In this notebook, you will implement all the functions required to build a convolutional neural network.

After finishing this assignment, you will have a deeper understanding of the process of training a convolutional neural network, which only consists of two main part: convolution layer and pooling layer.

2. Packages

All the packages that you need to finish basic part of this assignment are listed below.

- numpy : the fundamental package for scientific computing with Python.
- matplotlib : a comprehensive library for creating static, animated, and interactive visualizations in Python.
- math : Python has a built-in module that you can use for mathematical tasks.
- sklearn.datasets : scikit-learn comes with a few small standard datasets that do not require to download any file from some external website. You will be using the Iris dataset to build a binary classifier.
- pandas.read_csv : provides functionality for reading a csv dataset from a GitHub repository.
- sklearn.model_selection.train_test_split: A function helps you split train and test data.
- os: A module provides the facility to establish the interaction between the user and the operating system. You can access image directory by os.
- cv2.imread: It is the module import name for opencv-python. You can use it to read image.

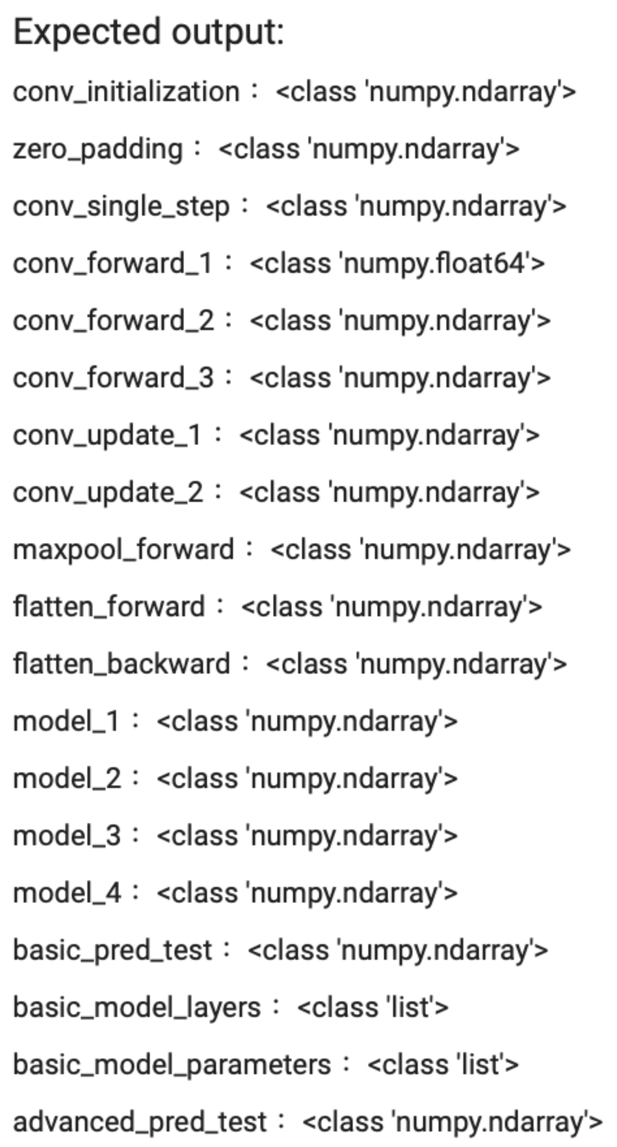
⚠️ **WARNING** ⚠️:

- Please do not import any other packages.
- np.random.seed(seed) is used to keep all the random function calls consistent. It will help us grade your work. Please don't change the seed.

Output NPY File Format

- Named as “**hw4_output.npy**”
- This file is a dictionary that stores your output for each function. Note that the hyperparameter and weights of classifier in basic part will also be stored in the output file to check whether your predictions come from the same classifier you submitted.
- The dictionary should have the following keys: 'conv_initialization', 'zero_padding', 'conv_single_step', 'conv_forward_1', 'conv_forward_2', 'conv_forward_3', 'conv_update_1', 'conv_update_2', 'maxpool_forward', 'flatten_forward', 'flatten_backward', 'model_1', 'model_2', 'model_3', 'model_4', 'basic_pred_test', 'basic_model_layers', 'basic_model_parameters', 'advanced_pred_test'

Output NPY File Format



Expected output:

```
conv_initialization : <class 'numpy.ndarray'>
zero_padding : <class 'numpy.ndarray'>
conv_single_step : <class 'numpy.ndarray'>
conv_forward_1 : <class 'numpy.float64'>
conv_forward_2 : <class 'numpy.ndarray'>
conv_forward_3 : <class 'numpy.ndarray'>
conv_update_1 : <class 'numpy.ndarray'>
conv_update_2 : <class 'numpy.ndarray'>
maxpool_forward : <class 'numpy.ndarray'>
flatten_forward : <class 'numpy.ndarray'>
flatten_backward : <class 'numpy.ndarray'>
model_1 : <class 'numpy.ndarray'>
model_2 : <class 'numpy.ndarray'>
model_3 : <class 'numpy.ndarray'>
model_4 : <class 'numpy.ndarray'>
basic_pred_test : <class 'numpy.ndarray'>
basic_model_layers : <class 'list'>
basic_model_parameters : <class 'list'>
advanced_pred_test : <class 'numpy.ndarray'>
```

The Evaluation Metric

In the basic implementations, you will get a full score if your output is exactly the same as our answer, or otherwise you will get a zero mark for each function implementation.

For the binary classifier, we will use **accuracy** to evaluate your model.

- Basic part: If your accuracy reaches 0.55, you will get all 10%.
- Advanced part: If your accuracy reaches 0.7, you will get 15%. You will be compared with others who submit the advanced prediction to get the rest 10%.

Requirement

- Do it individually! Not as a team! (team is for final project)
- Announce date: 2022/12/1
- Deadline: 2022/12/14 23:59 (Late submission is not allowed!)
- Hand in your files in the following format (Do not zip the files!)
  - hw4.ipynb (Please keep your execution output)
  - hw4_output.npy
  - hw4_report.pdf
- Assignment 4 will be covered on the next exam.

Penalty

0 points if any of the following conditions:

- Plagiarism
- Late submission
- Not using template or import any other packages
- Incorrect input/output format
- Incorrect submission format
- Predictions mismatch (your predictions did not come from the same classifier you submitted)

Questions?

- TA: Yun-Yang Huang (dan89092989@gmail.com)
- **No debugging service**