Lab3: Leukemia Classification

TA 劉子齊 Jonathan

Lab Objective:

In this lab, you will need to analysis acute lymphoblastic leukemia (急性淋巴性白血病) in the following three steps.

First, you need to write your own custom DataLoader also design your own data preprocessing technique through PyTorch framework.

Second, you need to classify acute lymphoblastic leukemia via the ResNet architecture [1].

Finally, you have to upload your prediction result to the Kaggle competition, also plot the confusion matrix and the accuracy curve to evaluate the performance.

Important Date:

1. Closure of the Kaggle competition: 8/3 (Thu.) 11:59 a.m.
2. Model weight upload deadline: 8/3 (Thu.) 11:59 a.m.
3. Experiment report submission deadline: 8/3 (Thu.) 11:59 a.m.
4. Demo date: 8/3 (Thu.)

Model Weight Upload Form

ResNet18 & ResNet50: https://forms.gle/F1pQn9osaZ5JzW5Y8

ResNet152: https://forms.gle/6gCvxuvFdA12Yzpz9

Turn in Experiment Report (.pdf) & Source code (.py)

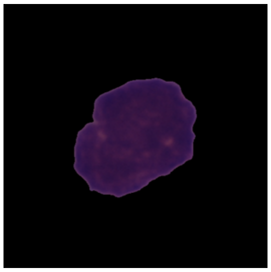
Notice: zip all files in one file and name it like 「DLP_LAB3_YourStudentID_name.zip」, ex: 「DLP_LAB3_311605004_劉子齊.zip」

Requirements:

1. Implement the **ResNet18, ResNet50, ResNet152** architecture on your own; **do not call the model from any library**.
2. Train your model from scratch, **do not load parameters** from any pretrained model.
3. Compare and **visualize** the accuracy trend between the **3 architectures**, you need to plot each epoch **accuracy (not loss)** during training phase and testing phase.
4. Implement your own custom **DataLoader**
5. Design **your own data preprocessing method**
6. Calculate the **confusion matrix** and plotting

Dataset - Leukemia Classification (Kaggle):

Acute lymphoblastic leukemia (ALL) is the most common type of childhood cancer and accounts for approximately 25% of pediatric cancers.These cells have been segmented from microscopic images and are representative of images in the real world because they contain some staining noise and illumination errors, although these errors have largely been fixed in the course of acquisition.



Format: .bmp

Reference :
https://www.kaggle.com/datasets/andrewmvd/leukemia-classification

Implementation Details:

Prepare Data

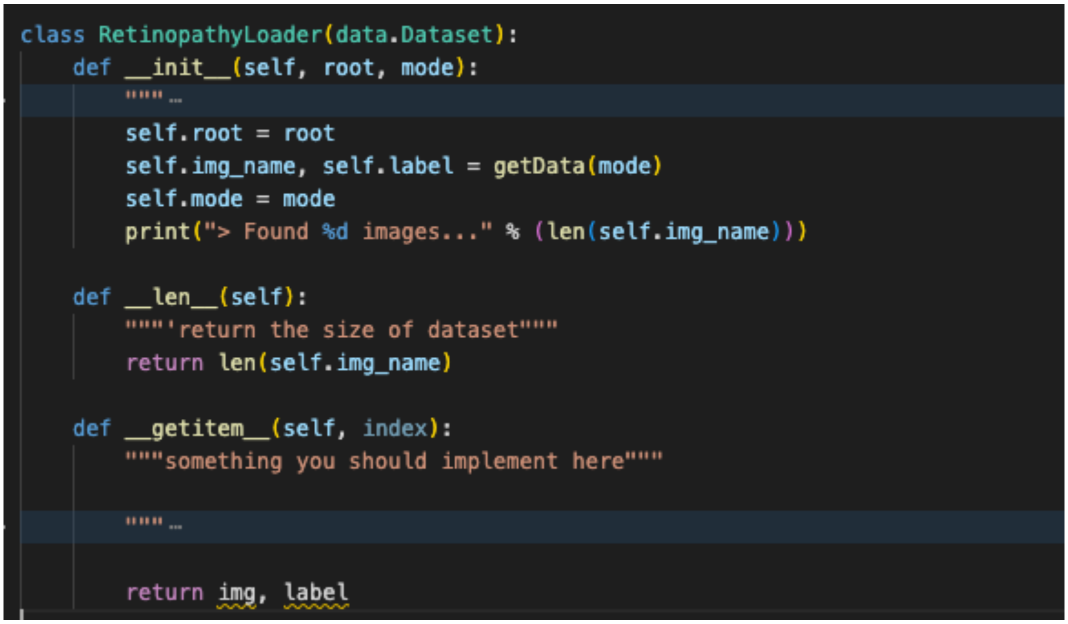
The dataset contains 10,661 images, we divided dataset into 7,995 training data, 1,599 validating data, and 1,067 testing data. In total, there are 10,661 images with two labeled classes:

- 0: Normal cell
- 1: Leukemia blast

The images’ resolutions are the same, which is 450 * 450 pixels.

Custom Dataloader

- This is the skeleton that you have to fill to have a custom dataset, refer to “dataloader.py”



```
class RetinopathyLoader(data.Dataset):
    def __init__(self, root, mode):
        """"…
        self.root = root
        self.img_name, self.label = getData(mode)
        self.mode = mode
        print("> Found %d images..." % (len(self.img_name)))

    def __len__(self):
        """'return the size of dataset"""
        return len(self.img_name)

    def __getitem__(self, index):
        """something you should implement here"""

        """"…

        return img, label
```

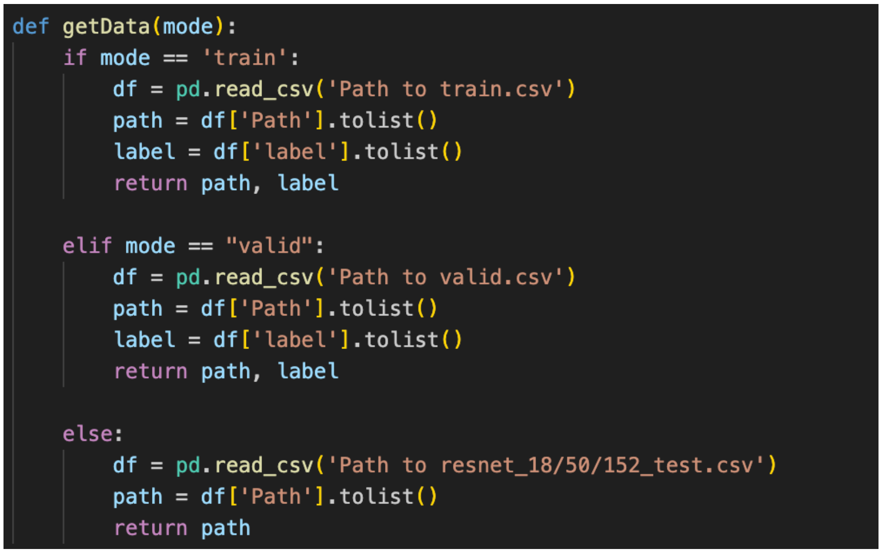
- The __init__ function is where the initial logic happens like reading a csv, assigning transforms etc.

- The__getitem__ function returns the data and labels and you can process the data like loading image, preprocessing, transforming image before returns.

- The index parameter where in the __getitem__ function, it is the nth data/image you are going to return.

- Reference: https://pytorch.org/tutorials/beginner/data_loading_tutorial.html https://github.com/utkuozbulak/pytorch-custom-dataset-examples

- You can use the following getData function to read all images' path and ground truth.



```python
def getData(mode):
    if mode == 'train':
        df = pd.read_csv('Path to train.csv')
        path = df['Path'].tolist()
        label = df['label'].tolist()
        return path, label

    elif mode == "valid":
        df = pd.read_csv('Path to valid.csv')
        path = df['Path'].tolist()
        label = df['label'].tolist()
        return path, label

    else:
        df = pd.read_csv('Path to resnet_18/50/152_test.csv')
        path = df['Path'].tolist()
        return path
```

ResNet

ResNet (Residual Network) is the Winner of ILSVRC 2015 in image classification, detection, and localization, as well as Winner of MS COCO 2015 detection, and segmentation [2].

- **Degradation problem:** the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Not overfitting, it's the vanishing/ exploding gradient problem.

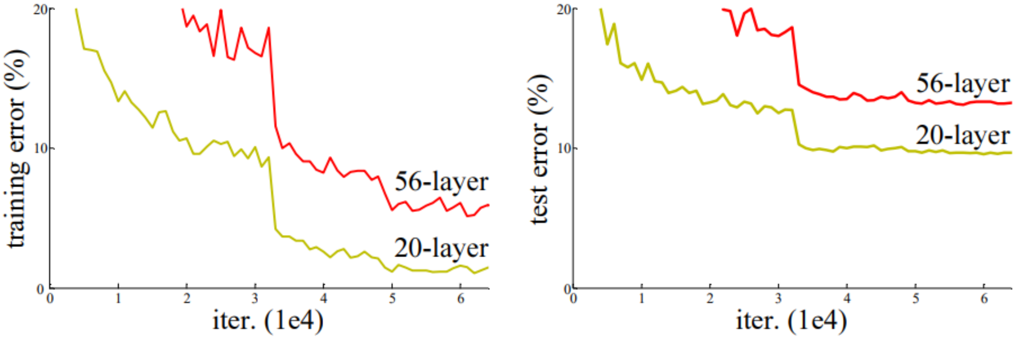
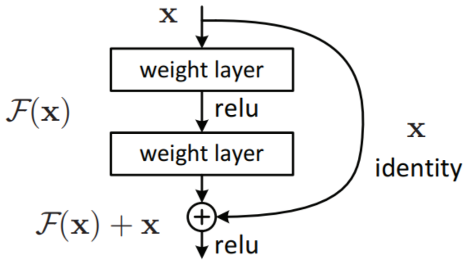


Figure. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error.

- **Skip/ Shortcut connection:**
  To solve the problem of vanishing/exploding gradients, a skip/ shortcut connection is added to add the input x to the output after few weight layers as below [2]

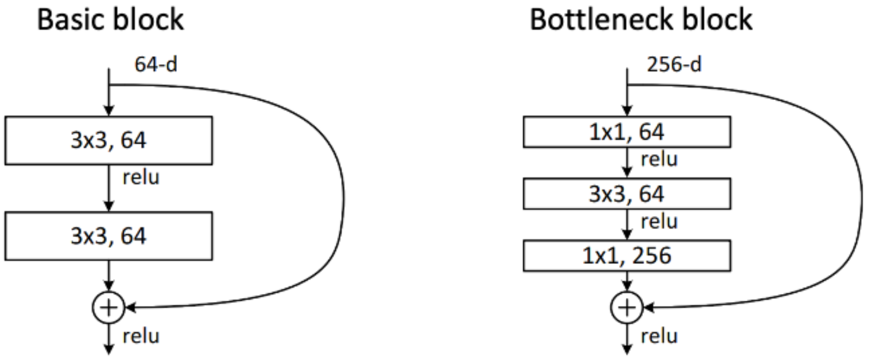


- Why ResNet can avoid vanishing gradient problem?

$$\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i),$$

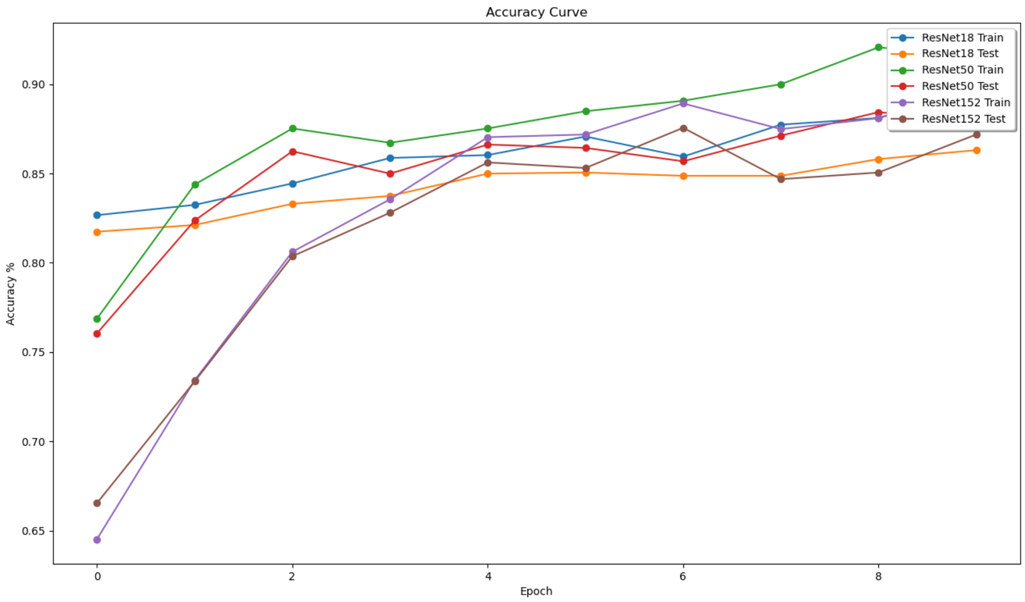
$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\left(1 + \frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)\right)$$

- Building residual basic block and bottleneck block:



- Using pretrained model and reinitialize the specific layers
  https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html

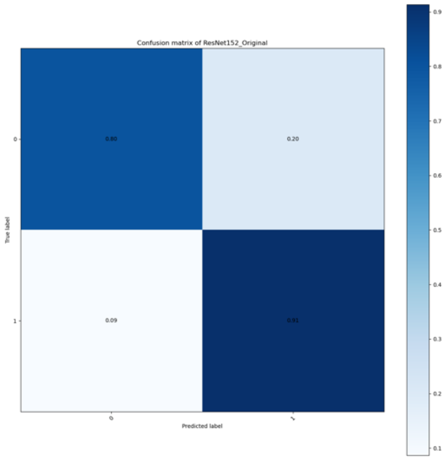
- Your visualization figure of accuracy should like example as below.



Confusion matrix

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It is extremely useful for measuring Recall, Precision, Specificity, Accuracy and most importantly AUC-ROC curve [3].

- The confusion matrix shall be similar as the following example:



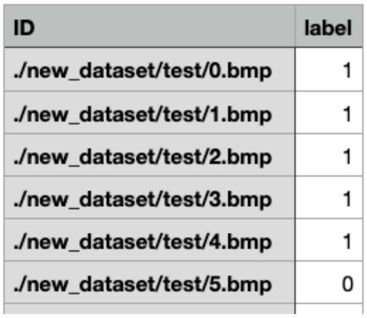
Report Spec

1. Introduction (10%)
2. Implementation Details (30%)
    A. The details of your model (ResNet)
    B. The details of your Dataloader
    C. Describing your evaluation through the confusion matrix
3. Data Preprocessing (20%)
    A. How you preprocessed your data?
    B. What makes your method special?
4. Experimental results (10%)
    A. The highest testing accuracy
        - Screenshot
        - Anything you want to present
    B. Comparison figures
        - Plotting the comparison figures
        - **(RseNet18, ResNet50, ResNet152)**
5. Discussion (30%)
    A. Anything you want to share

Kaggle Competition

Result Format

- The result should be a .csv file
- The file should contain the image id and the predicted label, which should look like the following figure:



ID	label
./new_dataset/test/0.bmp	1
./new_dataset/test/1.bmp	1
./new_dataset/test/2.bmp	1
./new_dataset/test/3.bmp	1
./new_dataset/test/4.bmp	1
./new_dataset/test/5.bmp	0

Evaluation

- We use Accuracy Score metric to evaluate your prediction result on the Kaggle website
- Since there are 3 different model architectures, there are 3 Kaggle competitions:
  - Leukemia Classification Competition - ResNet18
    - https://www.kaggle.com/t/b37aaf945a3d4529ae4a9a3e7bdc7c99
  - Leukemia Classification Competition - ResNet50
    - https://www.kaggle.com/t/fb3c03bfe8e04755a54493b8fb60aec1
  - Leukemia Classification Competition - ResNet152
    - https://www.kaggle.com/t/19fa714d7c3e4297a2a8f3acb2553e8d
- Please set your team's name as your student ID
  - Otherwise, you'll get no points in the competition
  - Also, you'll be kicked out of the competition, and get no points

---- Criterion of result (30%) ----

For each ResNet competition (10%)

- 1st : 10 pts
- 2nd : 9 pts
- 3rd : 8 pts
- Top 20%: 7 pts
- Top 40%: 6 pts
- Others above baseline: 5 pts
- Below baseline: 0 pts
- The total score is the sum of 3 competition ranking result.
- The baseline is on the leaderboard on the kaggle website

Score: 30% experimental results + 30% report + 40% demo score

P.S If the zip file name or the report spec have format error, it will be penalty (-5).

In demo phase, you need to load trained model and show your results of the 3 different models. If you failed to do so, you will get 0 pts on the model that you couldn't successfully demonstrate.

If you're found faking your model predicting result on Kaggle, you'll get 0 pts in this lab. Also, we will take disciplinary action in accordance with school regulations.

Please make sure the code and the model weight you uploaded can be successfully executed by the teaching assistant.

Reference:

[1] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[2] Review: ResNet
https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8

[3] Understanding Confusion Matrix
https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62