Lab2 : Butterfly & Moth Classification

Lab Objective:

In this lab, you will need to analyze butterflies or moths with one hundred species, following these 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 Butterfly or moths via the VGGNet and the ResNet architectures.

Finally, you need to plot the accuracy (not loss) curve for each epoch and show the highest accuracy of two architectures.

Important Date:

1. Experiment Report Submission Deadline: 4/10 (Wed) 11:59 p.m.
2. Demo date: 4/11 (Thu)

Turn in:

1. Experiment Report (.pdf)
2. Source code

Notice : zip all files in one file and name it like「DL_LAB2_YourStudentID_name.zip」, ex: [DL_LAB2_312553037_王芷鈴.zip」

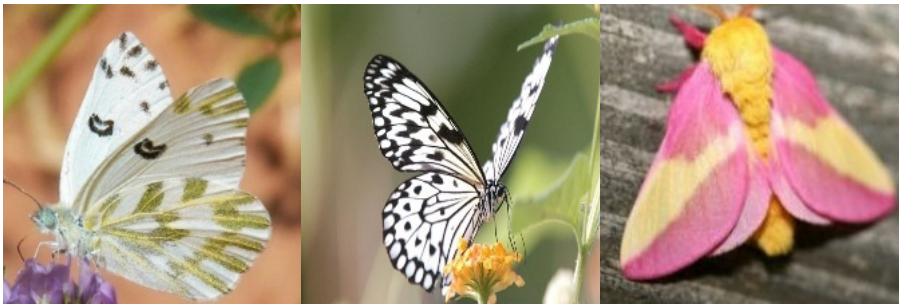
Requirements:

1. Implement the **VGG19, ResNet50** 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. Implement your own custom **DataLoader.**
4. Design **your own data preprocessing method.**

5. Compare and **visualize** the accuracy trend between the **two architectures**, you need to plot each epoch **accuracy (not loss)** during training phase and validation phase.
6. In the experiment results, you have to show the **highest accuracy** of two architectures.

Dataset – Butterfly & Moths Classification :

This dataset aims to classify 100 species of butterflies or moths. Each image within the dataset is formatted as a 224x224x3 JPG file, ensuring consistency across all samples.



Reference: https://www.kaggle.com/datasets/gpiosenka/butterfly-images40-species/data

Implementation Details:

Prepare Data

The dataset contains 13,594 images, we divided dataset into 12,594 training data, 500 validation data, and 500 testing data. In total, there are 13,594 images with one hundred labeled classes .

The images' resolutions are the same, which is 224 * 224 pixels.

Custom Dataloader

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

```python
class BufferflyMothLoader(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['filepaths'].tolist()
        label = df['label_id'].tolist()
        return path, label

    else:
        df = pd.read_csv('Path to test.csv')
        path = df['filepaths'].tolist()
        label = df['label_id'].tolist()
        return path, label
```

VGGNet

VGGNet (Visual Geometry Group Network) is the second place of ILSVRC 2014 .

- Overall visualization of the VGGNet architecture

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

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.

- **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.

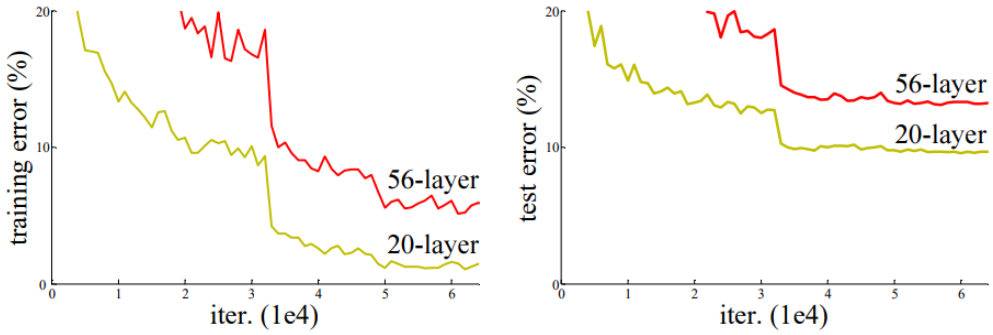
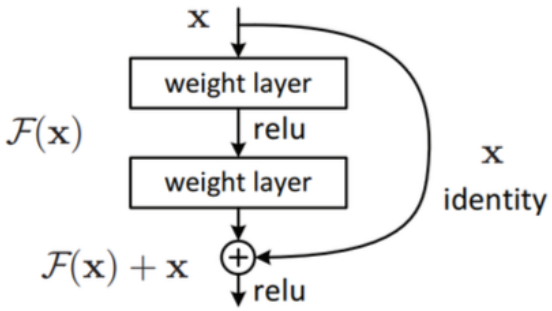


Figure. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error.

- **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]

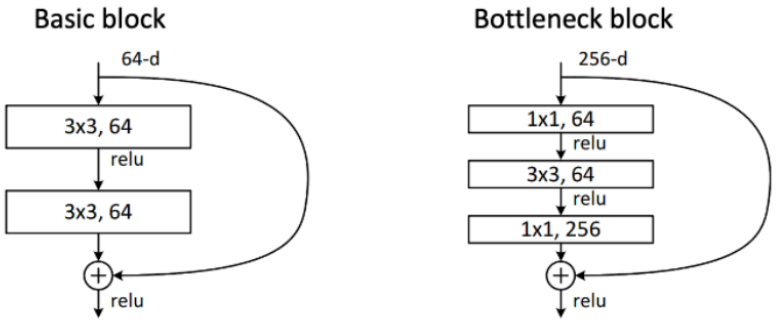


- Why ResNet can avoid vanishing gradient problem?

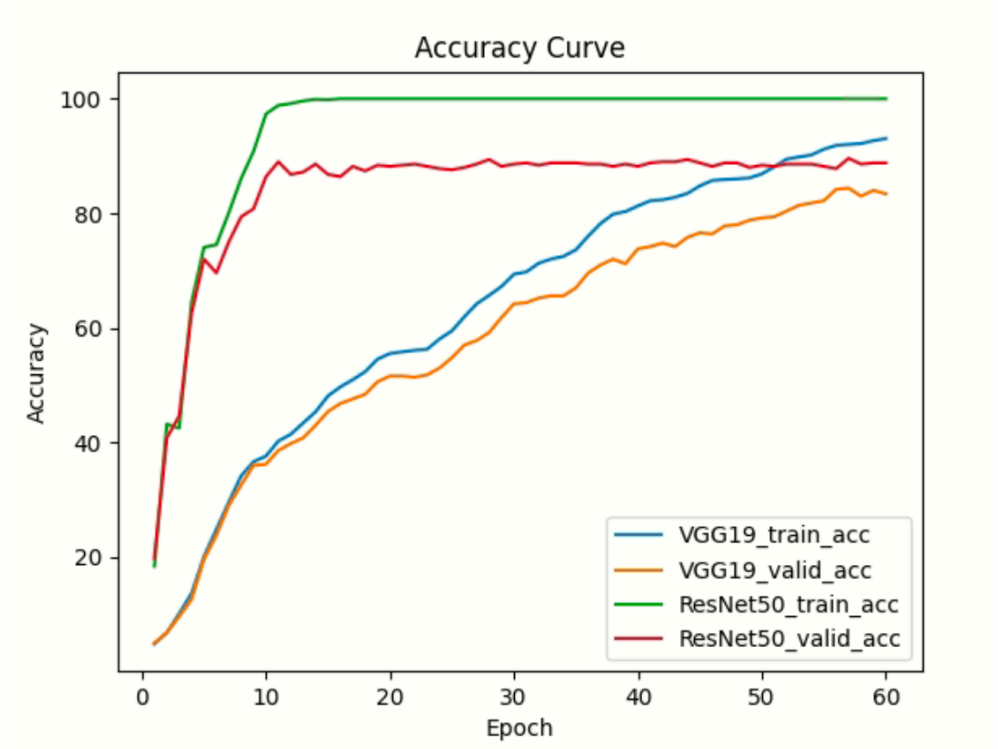
$$\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i),$$

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

- Building residual basic block and bottleneck block:



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



Report Spec (40%)

1. Introduction (10%)
2. Implementation Details (30%)
    A. The details of your model (VGG19,ResNet50)
    B. The details of your Dataloader
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
        • **(VGG19, ResNet50)**
5. Discussion(30%)
    A. Anything you want to share

Demo score (60%)

---- experimental result (20%) ----

Accuracy > = 88% = 100 pts

Accuracy 85~88% = 90 pts

Accuracy 80~85% = 80 pts

Accuracy 75~80% = 70 pts

Accuracy < 75% = 60 pts

---- question (40%) ----

Score:
60% demo score (experimental results & questions) + 40% report
If the zip file name or the report spec have format error, you will be punished (-5)

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. VGGNet - https://arxiv.org/abs/1409.1556

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