Artificial Intelligence Final Report Assignment 問題 1 (Problem 1)
レポート解答用紙（Report Answer Sheet）

Group Leader
学生証番号（Student ID):18521068
名前(Name):Trần Bình Luật

Group Members
学生証番号（Student ID):18521284
名前(Name):Huỳnh Ngọc Quân

学生証番号（Student ID):18521285
名前(Name):Nguyễn Minh Quân

問題１ (Problem 1)のレポート

I. Introduction

1. Image Classification

Image Classification is a fundamental task that tries to understand an entire image. The goal is to classify the image by assigning it to a specific label. For example, you may train a model to recognize photos representing three different types of animals: rabbits, hamsters, and dogs. This is one of the core problems in Computer Vision, and it has a large variety of practical applications (E.g Face ID).

Our input in Image Classification tasks consists of a set of N images, each one was labeled with one of K different classes. We refer to this data as the training set. Then we'll use this set to learn what every one of the classes looks like. After a long journey, our model can predict the class that an image belongs to as an output.

2. Dataset

The CIFAR-10 dataset (Canadian Institute For Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms. The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes.The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class. There are 50000 training images and 10000 test images.

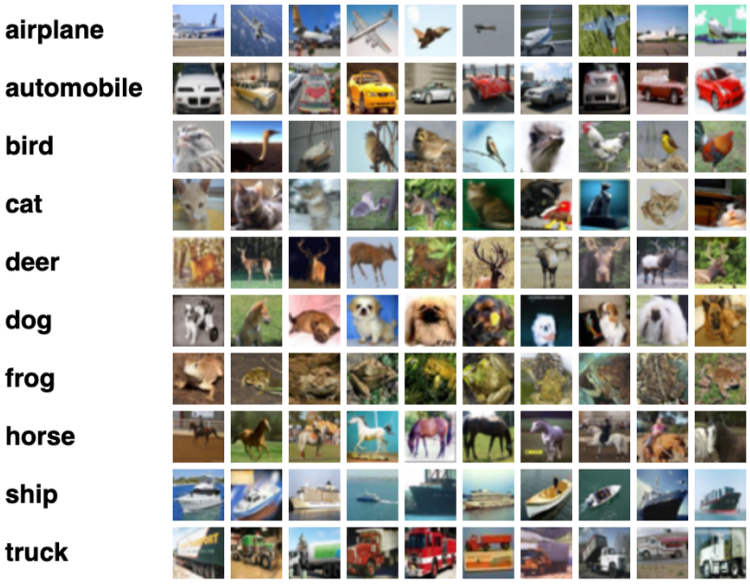

Figure 1. An example in CIFAR-10 dataset

II. Method

In this task we will use VGG model to solve this problem. VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU’s.

1. Architecture

The input of VGG is set to an RGB image of 224x244 size. The average RGB value is calculated for all images on the training set image, and then the image is input as an input to the VGG convolution network. A 3x3 or 1x1 filter is used, and the convolution step is fixed. . There are 3 VGG fully connected layers, which can vary from VGG11 to VGG19 according to the total number of convolutional layers + fully connected layers. The minimum VGG11 has 8 convolutional layers and 3 fully connected layers. The maximum VGG19 has 16 convolutional layers. +3 fully connected layers. In addition, the VGG network is not followed by a pooling layer behind each convolutional layer, or a total of 5 pooling layers distributed under different convolutional layers. The following figure is VGG Structure diagram:

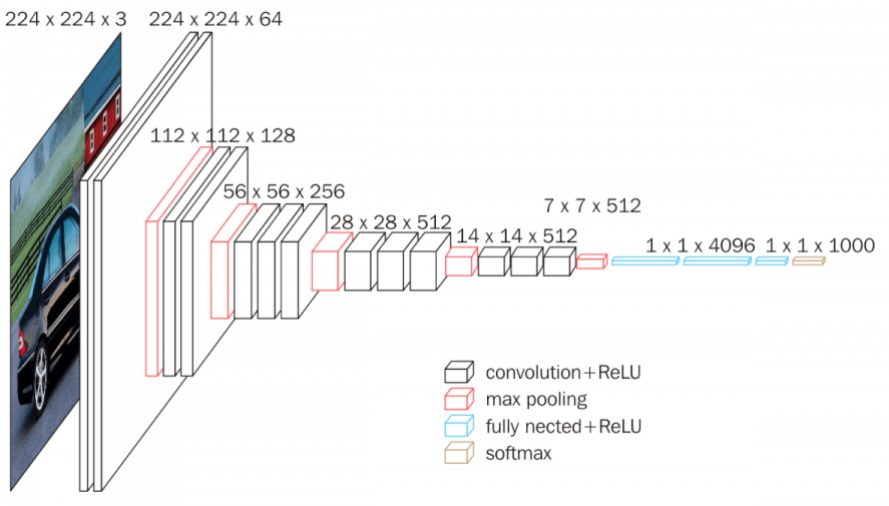

Figure 2. VGG16 Architecture

2. Configurations

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**
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**
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**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 1. ConvNet Configuration

The ConvNet configurations are outlined in table 1. The nets are referred to their names (A-E). All configurations follow the generic design present in architecture and differ only in

the depth: from 11 weight layers in the network A (8 conv. and 3 FC layers) to 19 weight layers in the network E (16 conv. and 3 FC layers). The width of conv. layers (the number of channels) is rather small, starting from 64 in the first layer and then increasing by a factor of 2 after each max-pooling layer, until it reaches 512.

III. Experiments

In this task, in order to achieve the best results, we have adjusted the parameters in turn such as: the number of epochs, the number of layers of the VGG model, the batch size of the data, etc.

In the first step, we train the model for 10 epochs and get 81%. However, we realize that the model can be even better based on the decrease of the loss function, we adjust the number of epochs from 10 to 30 and draw a few notes:

- In the first epochs, the loss function has a very strong decrease in frequency and tends to decrease slowly as the latter epochs.
- Accuracy also increased strongly in the first epochs and gradually saturated at the 10th epoch (81%) and reached its limit at the 25th epoch (85%).

Next, we adjust the number of layers. We tested with 1 layer, 2 layers and 3 layers in turn and then compared the results.

- For single layer models, in the first epoch Accuracy will tend to be 2 layers and 3 layers taller (figure 3 & figure 4). Specifically, 0.5 for 1 layer, 0.3 for 2 layers and 0.42 for 3 layers.
- The loss function of 1 layer will also be lower than that of 2 layers and 3 layers, although the reduction is also uniform in 3 layers through each epoch and gradually equals in the last epoch.
- We found that changing, adding layers did not bring better results for the model. Specifically, the model's peak Accuracy is also at 86.6% in the 3-layer model. However, Accuracy tends to increase slowly in 3 layers and does not have a decreasing trend like in 1 layer.

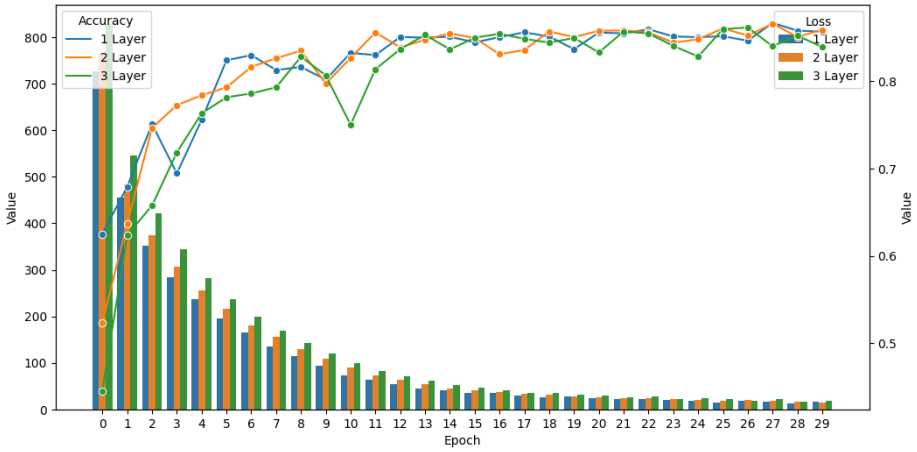

Figure 3. Result visualization bacth_size = 100

Realizing the model's saturation, we decided to test the data by adjusting the batch_size of the data from 100 to 30 and see the results.

- When we set the batch size of data at 30, we can see that in the first epochs, the Loss function of the model is quite high. However, it has a relatively strong decrease in the next epochs.

In this adjustment, we did not succeed in helping the model increase accuracy, but we did stabilize the model compared to before. Specifically, from epoch 20 to epoch 30, accuracy is always maintained at 86% instead of fluctuating like batch_size at 100.

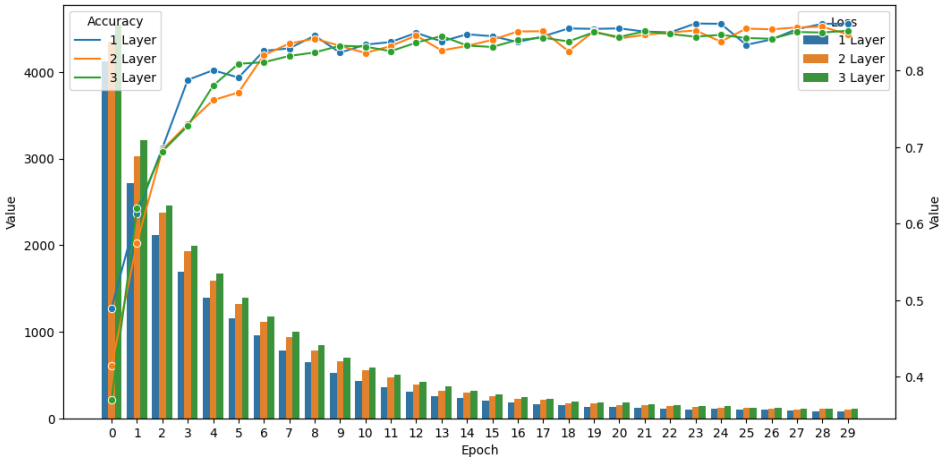

Figure 3. Result visualization bacth_size = 30

IV. Conclusions Future Work

In this work we evaluated very deep convolutional networks (up to 19 weight layers) for largescale image classification. It was demonstrated that the representation depth is beneficial for the classification accuracy, and that state-of-the-art performance on the ImageNet challenge

dataset can be achieved using a conventional ConvNet architecture (LeCun et al., 1989; Krizhevsky et al., 2012) with substantially increased depth. In the appendix, we also show that our models generalise well to a wide range of tasks and datasets, matching or outperforming more complex recognition pipelines built around less deep image representations. Our results yet again confirm the importance of depth in visual representations.

In general, VGG achieved quite good results in the image classification task of the CIFAR-10 dataset. We used VGG16 and helped us achieve an accuracy of up to 86,6% and pass the requirements set by the teacher. In this problem 1, we didn't encounter any difficulty. This also gives us a great inspiration to continue learning about other tasks of Computer Vision in the future.

In the future, we will try to use more models such as shufflenet, resnet, googlenet,... to test the accuracy and learn more methods to fine tune the parameters to get the best performance for the model.