ENGG5104 – Image Processing and Computer Vision – Spring 2022

Assignment 2 — Image recognition

Deadline: Mar 16, 2022 (Wednesday) 23:59

1 Task 1: Implement VGGNet

This section aims to implement a simple deep convolutional neural network called VGGNet.

1.1 Background

VGG stands for Visual Geometry Group. It is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The "deep" refers to the number of layers with VGG-16 or VGG-19 consisting of 16 and 19 convolutional layers.

The VGG architecture is the basis of ground-breaking object recognition models. Developed as a deep neural network, the VGGNet also surpasses baselines on many tasks and datasets beyond ImageNet. Moreover, it is now still one of the most popular image recognition architectures.

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					

Figure 1: VGGNet architecture.

1.2 Requirements

Only torch.nn.Conv2d, torch.nn.ReLU, torch.nn.Linear, torch.nn.MaxPool2d, torch.nn.Dropout, torch.nn.Softmax, nn.AdaptiveAvgPool2d can be directly used in your implementation. **No** other modules can be used. You can choose different stride, channels number, kernel size, padding for each layer to boost the performance. **torch.nn.Dropout is recommended in the classifier** to facilitate the training and avoid over-fitting. Note that depthwise convolution or group convolution must not be used. The computational cost should be under 200M FLOPs.

Your code for VGGNet should be put in model.py.

1.3 To Do List:

Implement VGGNet-A with 11 weight layers (class vggnet in model.py). (40 points)

1.4 Reference

1. `https://pytorch.org/docs/stable/nn.html`

2. `https://arxiv.org/pdf/1409.1556.pdf`

2 Task 2: CrossEntropy Loss

2.1 Background

With the linear classifier parameterized with W, given an image and label (X, y), x is the features from deep convolutional neural network, the loss will be as the following,

$$\mathcal{L}_{cross-entropy} = -\log \frac{\exp(w_y \cdot x)}{\sum_{i \in \{0,1,2,\ldots,n\}} \exp(w_i \cdot x)},$$

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

The graph above shows the range of possible loss values given a true observation (isDog = 1). As the predicted probability approaches 1, log loss slowly decreases. As the predicted probability decreases, however, the log loss increases rapidly. Log loss penalizes both types of errors, but especially those predictions that are confident and wrong!

Cross-entropy and log loss are slightly different depending on context, but in machine learning when calculating error rates between 0 and 1 they resolve to the same thing.

2.2 Todo List:

Implement cross-entroy loss function (class CrossEntropyLoss in loss.py). (20 points)

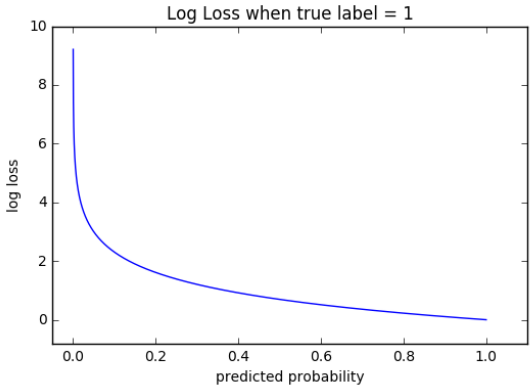

Figure 2: Cross-entropy loss curve.

After completing this part, the reference accuracy is 48(± 3)%.

2.3 Reference

1. `https://en.wikipedia.org/wiki/Cross_entropy`

3 Task 3: Conventional Augmentations

3.1 Background

Before input images into deep convolutional neural networks, we usually do some preprocessing on the images, like random flip, random padding, random crop, and normalization. In this task, you are required to implement the preprocessing operations including random flip, padding, random crop by yourself. Pytorch functions in torchvision.transforms must not be used.

3.2 Todo List:

Implement preprocessing including random flip, padding, and random crop. (10 points)

Your code should be put in transform.py.

After completing this part, the reference accuracy is 64(± 3)%.

3.3 Reference

1. `https://pytorch.org/docs/stable/_modules/torchvision/transforms/transforms.html`

4 Task 4: Image recognition challenge

4.1 Background

In this task, you are required to conduct experiments on CIFAR-10 dataset. The higher Top-1 accuracy, the better. Any other tricks can be pluged in your algorithm, like cosine learning rate schedule, data augmentations and regularization techniques.

An compulsory task here is to implement Cutout augmentation technique. Cutout, the simple regularization technique of randomly masking out square regions of input during training can be used to improve the robustness and overall performance of con- volutional neural networks. Not only is this method easy to implement, but it can be used in conjunction with existing forms of data augmentation and other regularizers to further improve model performance.

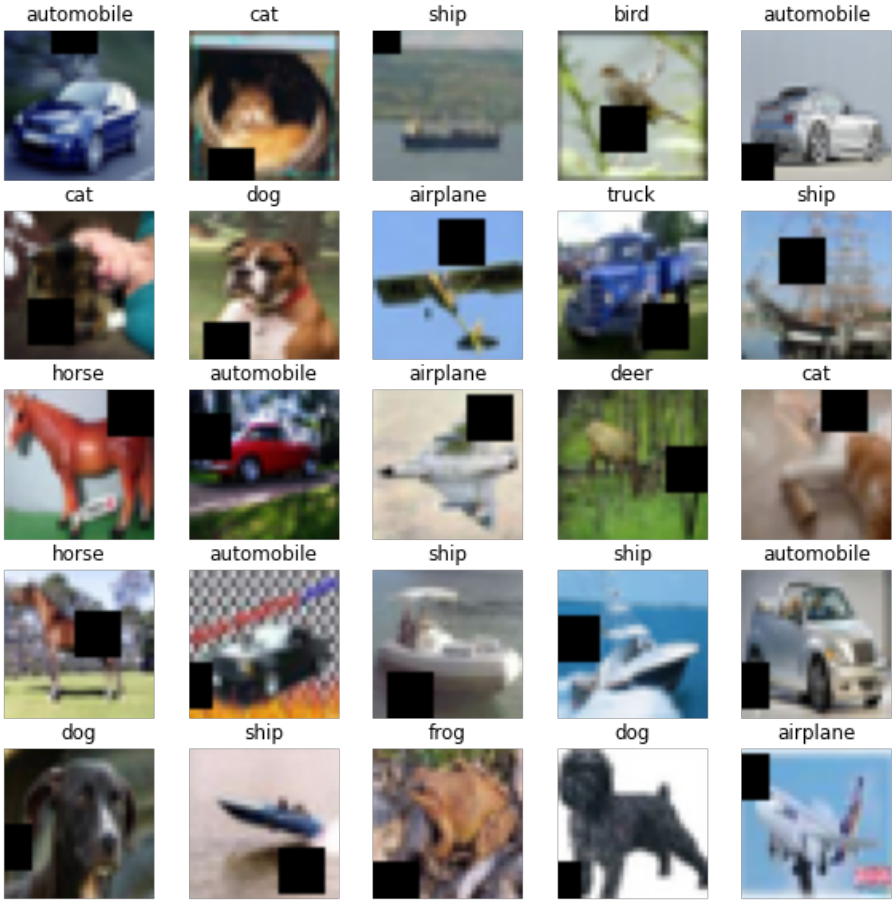

Figure 3: Cutout data augmentation.

4.2 Todo List:

1. Implement cutout augmentation (in the transform.py). After completing this part, the reference accuracy is 65(±3)%.

2. Improve your algorithm with any tricks and achieve Top-1 accuracy as high as possible (If your accuracy is higher than 67%, you can get full 100 points for the assignment.).

This is an open question, you can change the code anyway you want. The only constrain is FLOPs of model. Read the following papers for inspiration.

4.3 Reference

1. `https://arxiv.org/pdf/1708.04552.pdf`

5 General Specification

You are required to complete all functionalities according to the specification with Python. Skeleton programs are given. Do not change the files' name or the functions' name. You are **NOT** allowed to implement your program in another language (e.g. Matlab/C++) and then initiate system calls or external library calls in Python. Your source codes will be compiled and PERUSED, and the object code tested! You should not add new files for the first 4 tasks.

The folder's name, filename and code can not contain any Chinese.

- **NO PLAGIARISM!**
  You are free to design your algorithm and code your implementation, but you should not "borrow" codes from your classmates. If you use an algorithm or code snippet that is publicly available or use codes from your classmates or friends, be sure to DECLARE it in the comments of your program. Failure to comply will be considered as plagiarism.

6 Submission Guidelines

The folder you hand in must contain the following:

- **README.txt** - containing anything about the project that you want to tell the TAs, including a brief introduction of the usage of the codes.
- **code/** - directory containing all your code for this assignment, which is expected to have five .py files as shown in the skeleton.

Rename the folder as <your student ID>-Asgn2, and compress it into <your student ID>-Asgn2.zip, and upload it to blackboard system. (For example, 1155123456-Asgn2/ and 1155123456-Asgn2.zip, pay attention to the name.)

Please read the guidelines CAREFULLY. If you fail to meet the deadline because of a submission problem on your side, marks will still be deducted.

The late submission policy is as follows:

- 1 day late: -20 marks

- 2 days late: -40 marks

- 3 days late: -100 marks

Pay attention to the format before. 10% deduction for every wrong format(file name, function name, etc.).