Selected Topics in Visual Recognition using Deep Learning

Homework 1

Kuan-Chung Lin 0856735

Project repository: https://github.com/purpleFar/car-brand-classification

---

Introduction

The proposed challenge is a car images classification task with vehicle brand. The Cars dataset contains 16,185 images of 196 classes of cars. The data is split into 11,185 training images and 5,000 testing images. Classes are typically at the level of Make, Model, Year, ex. 2012 Tesla Model S or 2012 BMW M3 coupe.

The difficulty with this challenge is that the appearance of some pictures of different cars is easy to be very similar. For example: Acura TL Sedan 2012 and Acura TSX Sedan 2012.

Acura TL Sedan 2012

Acura TSX Sedan 2012

The classification model must learn the tiny details from the training data to get high accuracy. There are many other factors in the test data, such as low resolution, poor shooting angles and watermarks covering the car body.

Difficult to classify pictures in test data

Environment

- System: Windows 10
- CPU: Intel® Core™ i5-10300H CPU @ 2.50GHz 2.50GHz
- GPU: NVIDIA GeForce GTX 1660 Ti
- Python: 3.6.12
- Extra modules: Pytorch-1.5.0, Imgaug-0.4.0

Data Processing

First, I split the train dataset into 15-fold, fourteen as the training set, and one as the validation set. But the training set and validation set are not fixed all the time. Each fold will be the validation set once. In order to balance number of classes in each fold, I cut each category into 15-fold before merging them together. I will explain how I used these 15-fold in the next chapter.

Secondly, I perform the common processes of computer vision on images of train dataset, such as resizing, adding Gaussian noise, flipping, rotating, and performing the above processing in random order. Then, I map all the pixels of the image of the train and valuation dataset from 0~255 to 0~1. Finally, I normalize the pixels according to the channel where each pixel is located. Because my model is pre-trained from ImageNet, I use the following parameters for normalization:

Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

Model

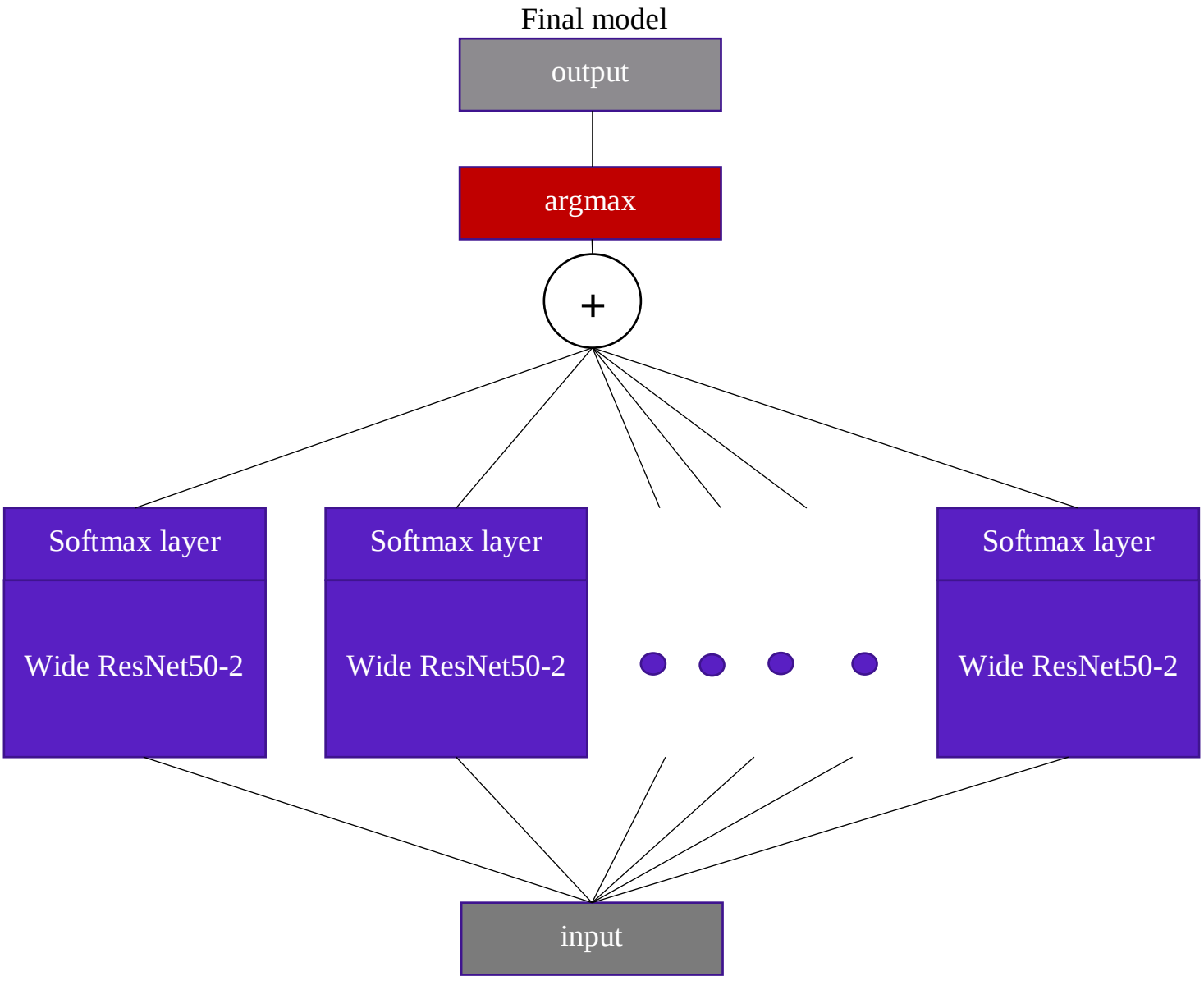
I use the best performing model in ImageNet, which is "Wide ResNet 50-2" as my model architecture. Because my GPU memory is only 6GB, I didn’t use " Wide ResNet 101-2".

ImageNet 1-crop error rates (224x224)

Network	Top-1 error	Top-5 error
AlexNet	43.45	20.91
VGG-11	30.98	11.37
VGG-13	30.07	10.75
VGG-16	28.41	9.62
VGG-19	27.62	9.12
VGG-11 with batch normalization	29.62	10.19
VGG-13 with batch normalization	28.45	9.63
VGG-16 with batch normalization	26.63	8.50
VGG-19 with batch normalization	25.76	8.15
ResNet-18	30.24	10.92
ResNet-34	26.70	8.58
ResNet-50	23.85	7.13
ResNet-101	22.63	6.44
ResNet-152	21.69	5.94
SqueezeNet 1.0	41.90	19.58
SqueezeNet 1.1	41.81	19.38
Densenet-121	25.35	7.83
Densenet-169	24.00	7.00
Densenet-201	22.80	6.43
Densenet-161	22.35	6.20
Inception v3	22.55	6.44
GoogleNet	30.22	10.47
ShuffleNet V2	30.64	11.68
MobileNet V2	28.12	9.71
ResNeXt-50-32x4d	22.38	6.30
ResNeXt-101-32x8d	20.69	5.47
Wide ResNet-50-2	21.49	5.91
Wide ResNet-101-2	21.16	5.72
MNASNet 1.0	26.49	8.456

However, when I trained the model and submitted the answer to Kaggle, I thought my score was not high enough, even though it passed the baseline. In order to increase my accuracy I trained fifteenth models with different training set and valuation set. Just like cross-validation, I use the fold of different index to make different training set and validation set. Unfortunately, my program crashed when it is training the 13th model. But it took me so much time that I can't train again from the beginning.

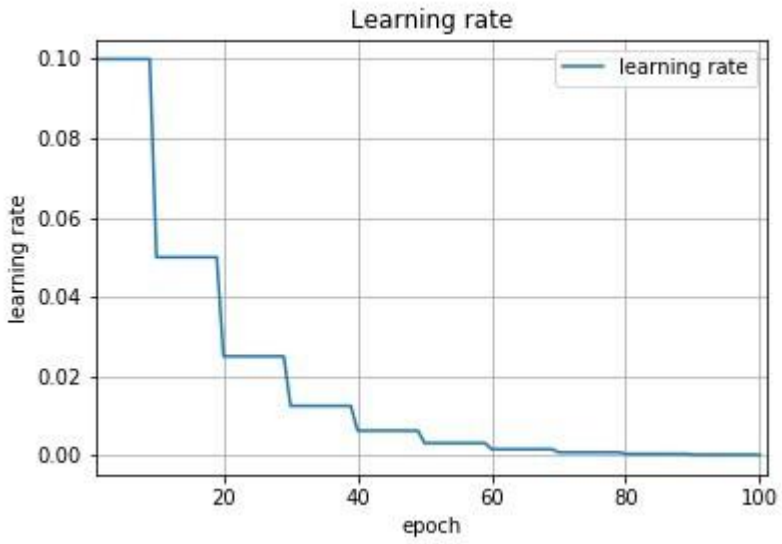
Final, I merge these 13th model as the below image then I got my best submission.



Other technical details

I used SGD and decrease learning rate every 10 epoch to optimize my model.

Decrease learning rate for each epoch



Besides, I write the early stopping algorithm by myself.

```
def train_early_stop(
    net,
    trainloader,
    valloader,
    optimizer,
    n_steps=800,
    p=10,
    savefile="./best_model.pt",
    show_acc=False,
    special_item=None,
    return_log=False,
    device=device,
):
    r"""

    Arguments:
    ---------
        net: (nn.Module)
            The model be trained.
        trainloader: (torch.utils.data.DataLoader)
            Torch dataloader in training.
        valloader: (torch.utils.data.DataLoader)
            ---
        optimizer:
            ---
        n_steps: (int)
            Get val accuracy and loss in each n_steps (default: ``100``).
        p: (int)
            Patience (default: ``6``).
        savefile: (string)
            The path of best model's parameters save (default: ``'./best_model.pt'``).
        show_acc: (boolean)
            Show valuation accuracy (default: ``False``).
        special_item: (function)
            f(model, inputs, labels)-->loss, outputs (default: ``None``).
        device: (string)
            'cpu' or 'cuda' (default: ``'cuda'``).
    """
```

Summary of Results

I got 90.92% in one Wide ResNet.

I got 93.4% when I merge all models.