CS x476 Project 3

Zhaodong Yang
halyang@gatech.edu
zyang645
903748903
6476

Part 1: Dataloader

Report the classes and the number of instances in the dataset, both train and test. (1pt)

Both the train and test set has 15 classes. The train set has 2985 instances and the test set has 1500 instances.

Part 1: Please briefly explain the structure of ImageLoader class and how it works (how to use it to load data)? (2pts)

ImageLoader mainly consists of function load_imagepaths_with_labels(), which finds out the image path and store it in self.dataset, load_img_from_path(), which loads an image according to path, and __getitem__(), which simultaneously gets an image and its class. There's some additional function in this class. Get_classes() is to find out all the class names of test set and train set. And __len__() is to find out the length of the dataset.

Part 2:
Report the dimensions of the input that is passed to and the output you obtain the SimpleNet you created. Express the answer in variables and clearly mention what each variable specifies. (1pt)

Input_dimension = [1, 1, 64, 64], which means [batchsize, channel, image size, image size]

Output_dimention = [1, 15], which means [batchsize, class probability]

Part 3: Loss function. Why do we need a loss function? (1pt)

We need a loss function to quantify the performance of the model. Then we can minimize the loss function to make model performs best.

Part 3: Explain the reasoning behind the loss function used. (1pt)

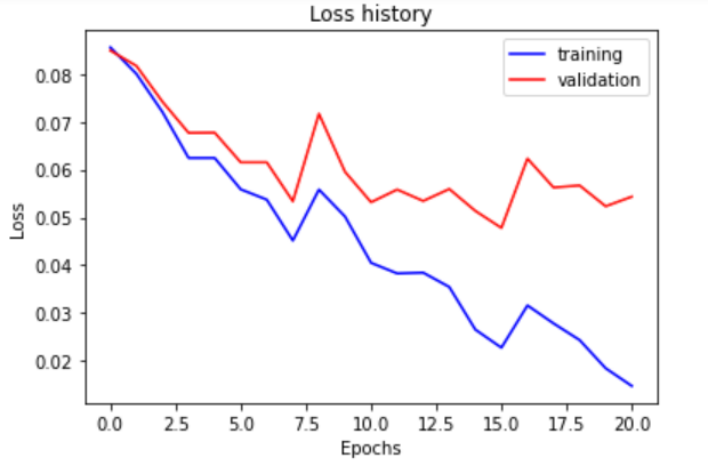
The cross entropy is multiplying the probability of true distribution and entropy of estimate. Larger the probability of true distribution is, larger the loss is. Larger the estimation probability is, smaller the loss is.

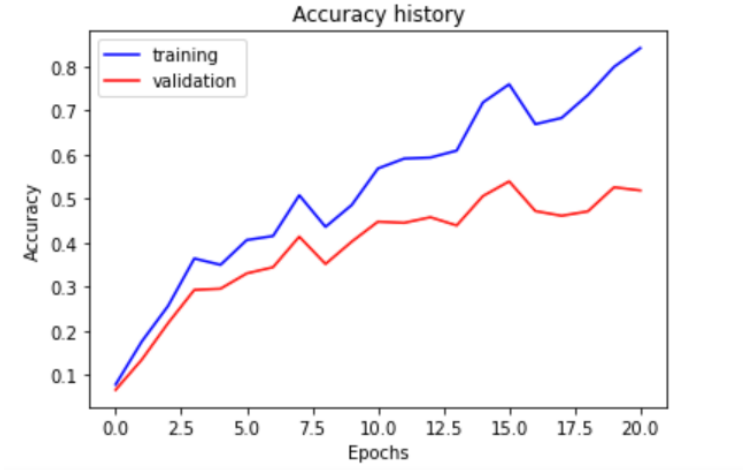
Part4: Optimizer.

Please briefly explain how an optimizer works? (2pts)

The optimizer calculate the gradient of a point and move the point in the direction where the gradient drops fastest.

Part 5: Training SimpleNet (5pts) – You need to get >45% validation accuracy as stated in notebook





Final training accuracy value: 0.8409

Final validation accuracy value: 0.5180

Part 5: Training SimpleNet (3pts)

"optimizer_type": "SGD",

"lr": 2e-1,

"weight_decay": 2e-4,

"momentum": 0.9

Part 6.1 : Screenshot of the functions you used in get_data_augmentation_transforms() (3pts)

```
return transforms.Compose([
    ##########################################################################
    # Student code begin
    ##########################################################################
    transforms.RandomHorizontalFlip(p = 0.5),
    transforms.Resize(inp_size),
    transforms.ToTensor(),
    transforms.Normalize(mean = pixel_mean, std = pixel_std)
    ##########################################################################
    # Student code end
    ##########################################################################
])
```

I really want to say that methods like RandomRotation and RandomResize somehow make the performance worse! It seems that there’s not much rotation and resize in test set I guess.

Part 6.2: Building AlexNet: why do we want to “freeze” the conv layers and some of the linear layers in pretrained AlexNet? Why CAN we do this? (0.5pt)

Because the AlexNet we use is pretrained. So it already has ability to extract the features from the images. If we unfreeze the layers, it might perform worse because our dataset is small. And it will take much more time to train it if we unfreeze the layers.

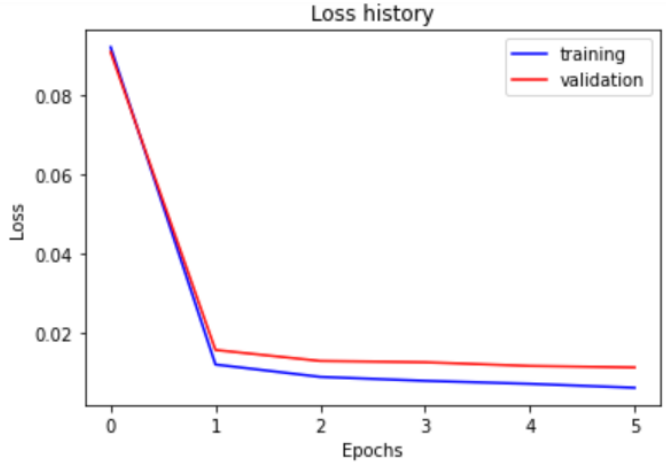
Part 6.2: Building AlexNet: Why don’t we have to freeze MaxPool and RELU layers? (0.5pt)

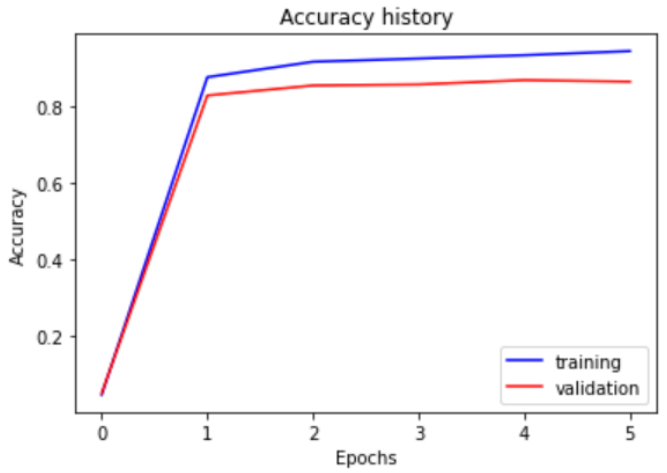
Because MaxPool and RELU do not have weights. There’s nothing we can change in them.

Part 6.3: Training AlexNet: what does fine-tuning a network mean? (1pt)

It means adjusting the hyperparameters to make the network perform best.

Part 6.3 : Training Alexnet (5pts) – you need to get >85% validation accuracy as stated in the notebook





Final training accuracy value: 0.9430

Final validation accuracy value: 0.8633

Part 6.3: Training AlexNet (3pts)

"optimizer_type": "SGD",

"lr": 2.5e-2,

"weight_decay": 1e-4,

"momentum": 0.9

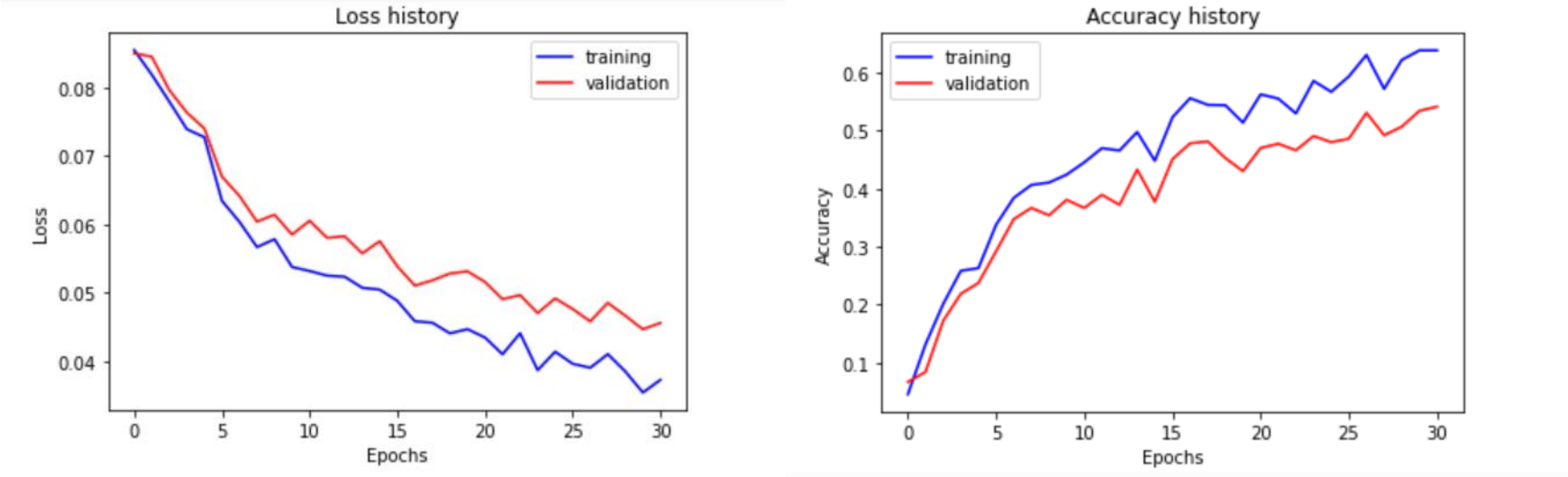
Conclusion: briefly discuss what you have learned from this project. (3pts)

I learned how to build and train a neural network from zero. I understood how loss function and optimizer worked and how the hyperparameters influenced the training of networks. Because I spent a lot of time tuning the networks, I also knew what values I should choose for the hyperparameters in the beginning. And I learned how to use colab as well.

EC1: SimpleNetDropout: How do dropout and data-augmentation help? (1pt)

Data-augmentation makes train set less biased. In this way networks work better on test set than without data-augmentation. Dropout randomly cut down connections between neurons, which make the network less dependent on some neurons.

EC1: Training SimpleNetDropout (3pts) – you need to get >52% validation accuracy as stated in the notebook.



Final training accuracy value: 0.6382

Final validation accuracy value: 0.5413

EC1: Training SimpleNetDropout (1pt)

"optimizer_type": "SGD",

"lr": 2e-1,

"weight_decay": 2e-4,

"momentum" : 0.9

EC1: SimpleNetDropout: compare the loss and accuracy for training and testing set, how does the result compare with that of SimpleNet without Dropout? How to interpret this result? (1pt)

The difference of loss and accuracy of SimpleNetDropout between training and testing sets are smaller than that of SimpleNet. If we define r = testing_accuracy / train_accuracy, r(SimpleNetDropout) = 0.8482 and r(SimpleNet) = 0.6160. We can tell r(SimpleNetDropout) is bigger. This is because dropout randomly cut down connections between neurons, which make the network less dependent on some neurons.

EC2: Quantization. Paste the code of the quantize function here. (3pts)

<Screenshot here>

EC2: Quantization. Briefly discuss the steps you followed. You cannot use code and should describe the intuition behind each step. (3pts)

<text answer here>

EC2: Quantization. Paste your results here (3pts) – If you satisfy the below limits.

Size comparison: <text answer here> { should be >50% reduction}

Processing Time comparison: <text answer here> { should be >10% reduction}

Accuracy comparison: <text answer here> { should be <5% reduction}