CS x476 Project 3

Aaron Lopes
alopes@gatech.edu
alopes7
903407727
4476

Part 1: Dataloader

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

There are 15 classes for both train and test. There are 2985 instances for train and 1500 instances in the dataset for test.

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

The ImageLoader class takes in a base filepath for a given dataset, the split folder to parse, and a fundamental transform function to preprocess the dataset before feeding it to the model.

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)

The input dimensions are [N, C, 64, 64] and the obtained output is tensor([[0.0757, -0.0358, -0.0265, 0.0288, -0.0046, 0.0410, 0.0711, -0.1041, -0.0040, -0.0742, -0.1032, -0.0071, -0.0169, -0.0947, 0.0376]]), a [1, 15] tensor of which the ith value corresponds to the probability of its belonging to the ith class.

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

A loss function is needed to measure model performance.

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

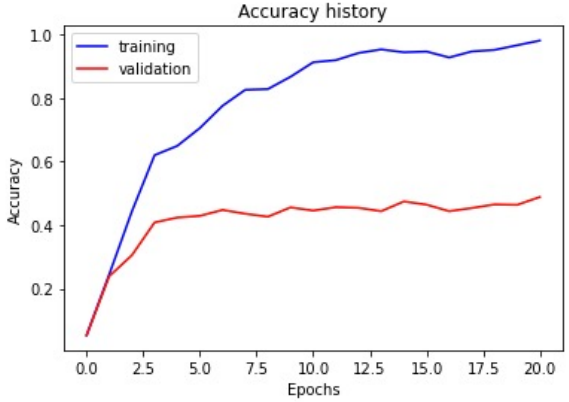
Cross entropy is used because we are measuring performance between two probability distribution models. Also, because it is independent of model and activation functions used.

Part4: Optimizer.

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

An optimizer uses computed gradients of the objective/loss function of a neural network to adjust the output layer weights and obtain better performance.

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



Final training accuracy value: 0.9812 (98.12%)

Final validation accuracy value: 0.4887 (48.87%)

Part 5: Training SimpleNet (3pts)

- "lr": 5e-4,
- "weight_decay": 5e-4

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

```python
# TODO 9
def get_data_augmentation_transforms(
    inp_size: Tuple[int, int], pixel_mean: np.array, pixel_std: np.
) -> transforms.Compose:
    """
  Returns the data augmentation + core transforms needed to be app

  Suggestions: Jittering(), Flipping(), Cropping(), Rotating() fro
  Need to add core transforms such as ToTensor() and Normalize() a

  Args:
  - inp_size: tuple denoting the dimensions for input to the model
  - pixel_mean: the mean of the raw dataset
  - pixel_std: the standard deviation of the raw dataset
  Returns:
  - aug_transforms: transforms.compose with all the transforms
  """

    return transforms.Compose(
        [
            transforms.Resize(inp_size),
            transforms.CenterCrop(10),
            transforms.ColorJitter(),
            transforms.ToTensor(),
            transforms.Normalize(mean=pixel_mean, std=pixel_std),
        ]
    )
```

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)

To prevent the weights from being retrained. We can do this because we are using a pretrained network.

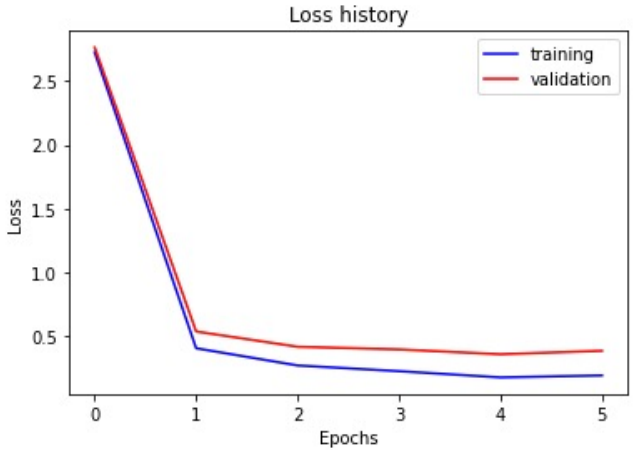
Part 6.2: Building AlexNet: Why don’t we have to freeze MaxPool and RELU layers? (0.5pt)

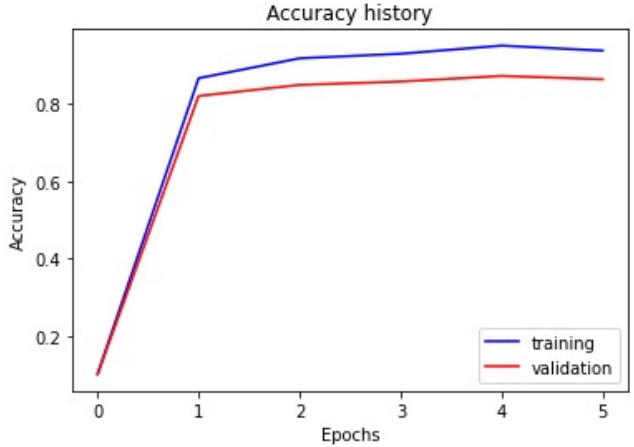
Because those are functions not associated with weights.

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

It means adjusting the hyperparameters; in this project’s case those are learning rate and weight decay.

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





Final training accuracy value: 0.9380

Final validation accuracy value: 0.8640

Part 6.3: Training AlexNet (3pts)

- "lr": 1e-2,
- "weight_decay": 5e-3

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

How learning rate and weight decay can significantly affect model accuracy. Training neural networks takes a lot of patience. How to align tensor shape as it flows through deeply connected and fully connected layers.

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

<Text solution here>

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

<Loss plot here>

<Accuracy plot here>

Final training accuracy value:

Final validation accuracy value:

EC1: Training SimpleNetDropout (1pt)

<Paste final hyperparameters here>

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)

<Text solution here>

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}