Emily Strong

INFO 7374

Assignment 2 Individual Part – NVIDIA CNN Course

**Summary Table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Part** | **Epochs** | **Learning Rate** | **Starting model** | **Architecture** | **Train Loss** | **Validation Accuracy** | **Validation Loss** | **mAP** |
| 2 | 5 | 0.01, stepped | None | AlexNet | 0.510573 | 78.9222 | 0.444362 | n/a |
| 4 | 3 | 0.0001, fixed | Part 2 | AlexNet | 0.417429 | 80.8514 | 0.417429 | n/a |
| 4 | 5 | 0.0001, fixed | Part 2 | AlexNet | 0.346343 | 81.2978 | 0.405009 | n/a |
| 4 | 8 | 0.0001, fixed | Part 2 | AlexNet | 0.485231 | 83.0198 | 0.377874 | n/a |
| 5 | 5 | 0.01, stepped | None | Fully Connected CNN | 0.383046 | 79.7353 | 0.433146 | n/a |
| 5 | 1 | 1.00E-100 | DetectNet | DetectNet | 1.08067 | n/a | 0.804193 | 13.4135 |

**GPU Task 1**

In this task, I completed two experiments. For the first one I trained the Alexnet network for 2 epochs on 16 pictures of beagles, some “Louis” and some “not Louis”. After 2 epochs it performs at chance, classifying images of both classes as Louis with slightly above 50% certainty (Figure 1A). For the second one, I trained it for 100 epochs. This time it gets the classifications correct with at or near 100% confidence (Figure 1B). However these tests are with images from the training data rather than images the model has never seen before, and the model is likely overfitting.

|  |
| --- |
|  |
| A |
|  |
| B |

**Figure 1:** A: Results after 2 epochs. Top image is of Louis, bottom image is not. B: Results after 100 epochs.

**GPU Task 2**

The purpose of this task was to learn about how validation data can be used to monitor a model’s performance with new data while it trains. In this task I trained a neural network to detect dogs versus cats. First the data has to be preprocessed to a size compatible with the AlexNet, and organized into folders where the folder name is the category label, and with 25% of the data set aside as a validation set. Then I trained AlexNet for 5 epochs. The validation data had above 80% accuracy after the 5 epochs as shown in Figure 2, and it categorized a picture of a dog correctly with greater than 90% probability.

|  |  |
| --- | --- |
|  |  |

**Figure 2:** Dogs vs. Cats with 5 epochs. The plot shows the training and validation loss and accuracy. The image of a dog is correctly classified as a dog.

**GPU Task 3**

The purpose of this task was to deploy a trained model in an application. I imported the architecture and weights from the model from task 2 into a new caffe classifier object in a jupyter notebook. The notebook calculates the predicted probabilities for each class for test images, selects the argmax and prints one statement if it a cat and another if it is a dog as shown in Figure 3. This process can be consolidated into a single script that simply shows the output instead of the image with the output.

|  |  |
| --- | --- |
|  |  |

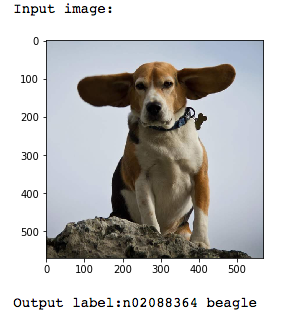
**Figure 3:** Example outputs for images of dogs and cats in the deployment application.

**GPU Task 4**

For the first part of this task, I took the model from Task 2 and trained it for more epochs to see the effects of training for a longer time. I tested 3, 5, and 8 additional epochs and found they brought only a small improvement to the validation accuracy and loss shown in the summary table. The loss and accuracy curves are in Figure 4. Then I tested using an AlexNet model pretrained on the Imagenet data set. It correctly identified a picture of Louis from Task 1 as a beagle (class 162). This can be used in an application to pair the image with its text label as shown in Figure 5.

|  |  |
| --- | --- |
| 3 | 5 |
| 8 | |

**Figure 4:** Train and validation loss and accuracy curves for 3, 5, and 8 additional epochs



**Figure 5:** Pretrained AlexNet application output for an image of a beagle

**GPU Task 5**

For this task I implemented several methods of object localization. I first created an object detector that using a sliding window method to divide the image into a grid of 256x256 sections and for each section predicts whether the image is a dog or a cat. The output is a very imprecise location for the dog in the image as shown in Figure 6.

Next I took the AlexNet architecture and changed it to be a fully connected CNN. This model did a better job of detecting where in the image the dog is, but also had false positives as show in figure 7, which also includes the accuracy and loss curves.

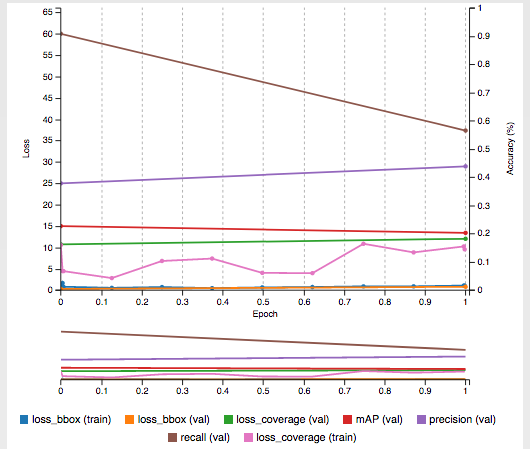
Finally I took a DetectNet architecture and fine-tuned it for one epoch with a very slow learning rate for the dogs and cats data set. Figure 8 contains the mean average precision (mAP) and loss curves, and Figure 9 contains example outputs. This application does a much better job of localizing the dog in the image, but in some images detects multiple overlapping dogs. This is consistent with the mAP of 13%.

|  |  |
| --- | --- |
|  |  |

**Figure 6:** Sliding window prediction as a method of object localization

|  |  |
| --- | --- |
|  |  |

**Figure 7:** Fully Connected CNN Accuracy and Loss Curves and Example Output



**Figure 8:** DetectNet Fine-Tuning Accuracy and Loss Curves

|  |  |
| --- | --- |
|  |  |

**Figure 9:** DetectNet Example Outputs