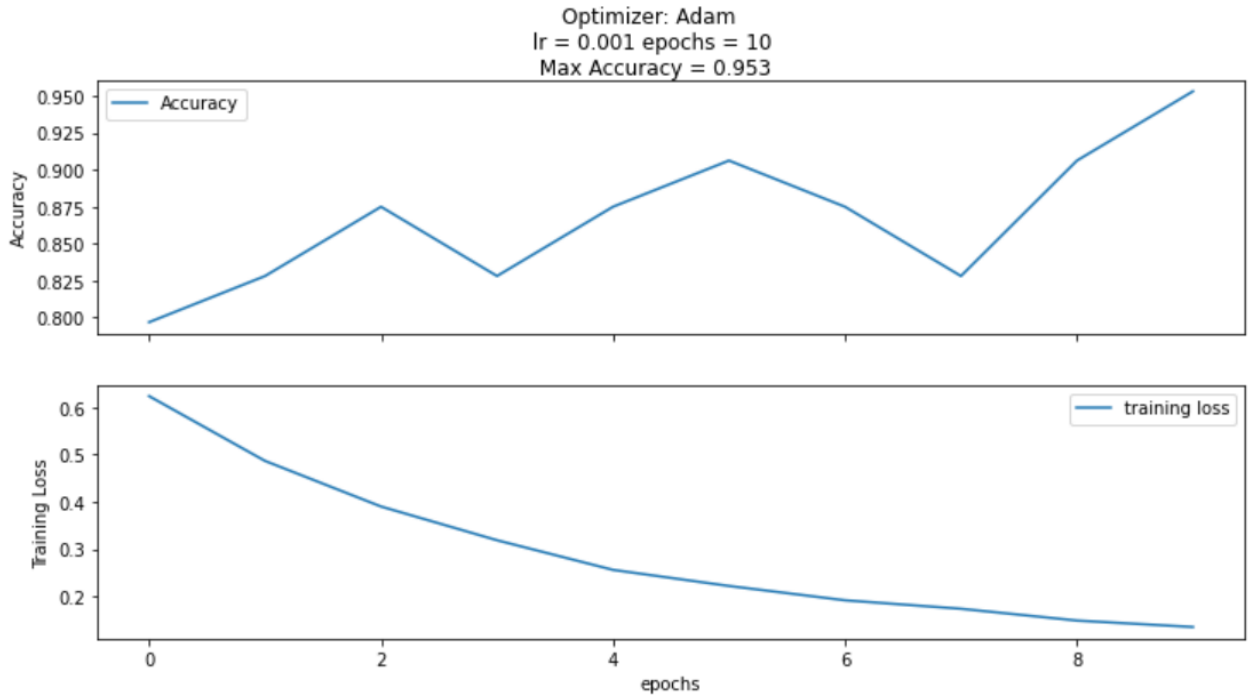
Eric Gagliano (egagli)
Inferring Structure from Complex Systems

Lab 5

Github link: https://github.com/egagli/amath563/blob/main/labs/5/Lab5_Notebook.ipynb

1. Implement AlexNet in PyTorch, achieve greater than 85% accuracy within 10 epochs on the Dogs and Cats classification task.

```
10%|▉         | 1/10 [02:40<24:04, 160.49s/it]Epoch: 1, Training Loss: 0.62379, Accuracy: 0.79688
20%|█▊        | 2/10 [05:23<21:35, 161.93s/it]Epoch: 2, Training Loss: 0.48675, Accuracy: 0.82812
30%|██▋       | 3/10 [08:07<19:01, 163.01s/it]Epoch: 3, Training Loss: 0.39034, Accuracy: 0.87500
40%|███▌      | 4/10 [10:50<16:16, 162.77s/it]Epoch: 4, Training Loss: 0.31904, Accuracy: 0.82812
50%|████▌     | 5/10 [13:39<13:45, 165.10s/it]Epoch: 5, Training Loss: 0.25638, Accuracy: 0.87500
60%|█████▍    | 6/10 [16:28<11:06, 166.64s/it]Epoch: 6, Training Loss: 0.22238, Accuracy: 0.90625
70%|██████▎   | 7/10 [19:18<08:22, 167.63s/it]Epoch: 7, Training Loss: 0.19208, Accuracy: 0.87500
80%|███████▏  | 8/10 [22:09<05:37, 168.54s/it]Epoch: 8, Training Loss: 0.17419, Accuracy: 0.82812
90%|████████  | 9/10 [24:58<02:48, 168.69s/it]Epoch: 9, Training Loss: 0.14925, Accuracy: 0.90625
100%|█████████| 10/10 [27:48<00:00, 166.80s/it]Epoch: 10, Training Loss: 0.13563, Accuracy: 0.95312
```



AlexNet has an architecture of 8 layers, with 5 convolution layers, 3 max pooling layers, and 3 fully connected layers. AlexNet uses ReLU activation, and outputs 1000 categories (though for our purposes, we use 2 categories). AlexNet takes input of 227x227x3, so we resize our cat and dog dataset to these specifications. The dataset we downloaded from kaggle. Since I'm using google collab, I've staged these images on google drive for easier access. My implementation worked well, and I tried using nn.Sequential() this time around which made the overall coding process simpler with less to define in the forward pass. With Adam with a learning rate of 0.001 and 10 epochs and a batch size of 64, I was able to achieve 95% accuracy. As we'll see in

problem 3, the network performed very well with the test images. Time wasn't too bad either, I used collab with GPU acceleration, so by sending the model and input data to the GPU, I was able to get through 10 epochs in 27 min.

2. What values did you use for following hyperparameters when training AlexNet that meets the accuracy requirement?

1) Learning Rate

Using Adam with a learning rate of 0.001.

2) Batch Size

Training and Test batch sizes of 64.

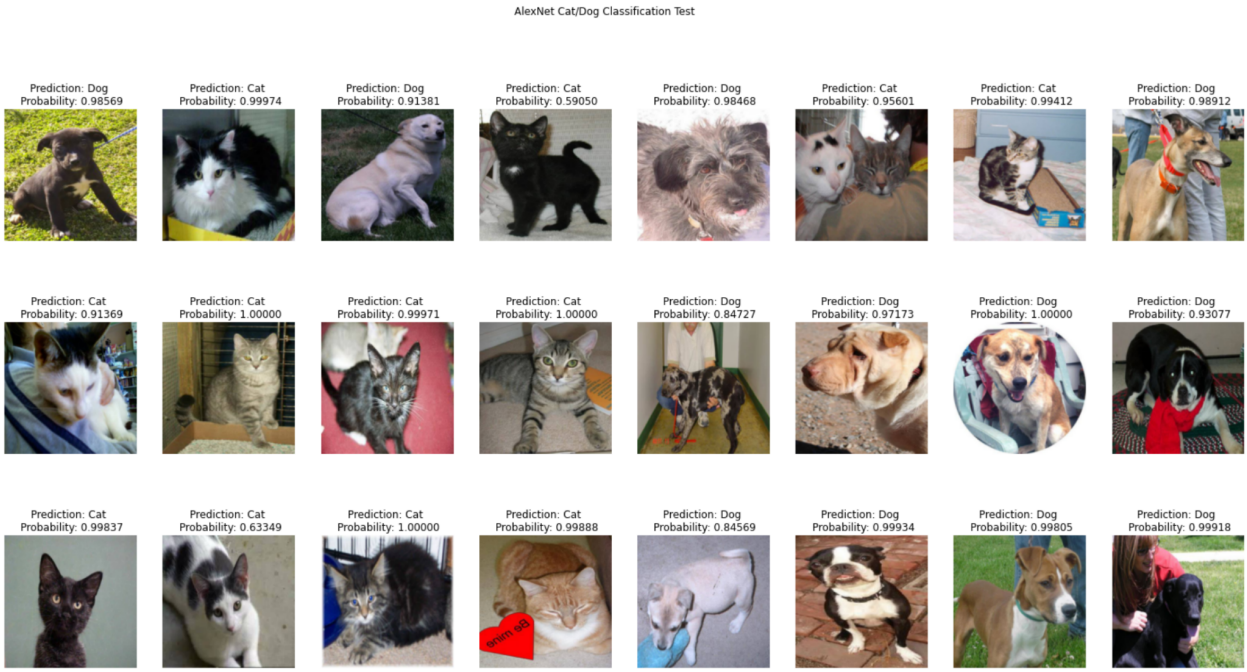
3) Dropout (if any) after your convolutional layers

Dropout used in the first two FC layers as described by the AlexNet paper.

4) Any other hyperparameters you set manually (regularization, normalization, etc.)

I use 10 epochs. No other manually set hyperparameters! I applied a resize, random horizontal flip, and normalization transformation to the images (though in displaying the images in the next question, I have used the un-normalized versions).

3. Plot the first 10 test images with a title that shows the probability of the predicted class.

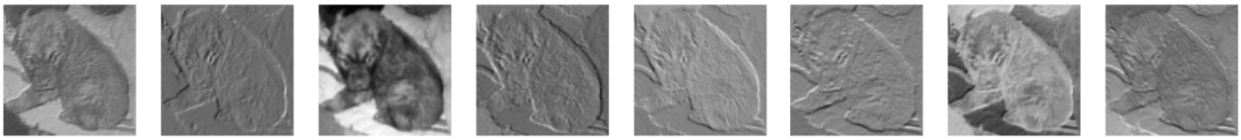


Looks like all cats and dogs are classified correctly. This makes sense as our calculated accuracy by the last epoch was 95%, so in a set of 24 images it is very plausible that the images are all classified correctly (back of the napkin calculation shows a probability of ~30% that this happened given a true accuracy of 95%). The classification probabilities of these test images are super high, and for the few images where classification probability is <90%, it makes sense that the network might be a bit confused. For example, the 5th image in the bottom row is a dog, and is correctly identified as a dog, but the probability is 85%. I could understand this because of the size and proportions of the dog, as well as the texture seems a bit cat-like.

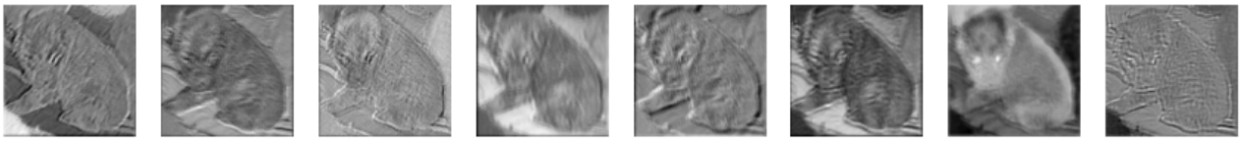
4. Visualize the feature maps of some images from the Dogs and Cats classification dataset.

Here are 4 example images with respective feature maps, 8 displayed features per layer:

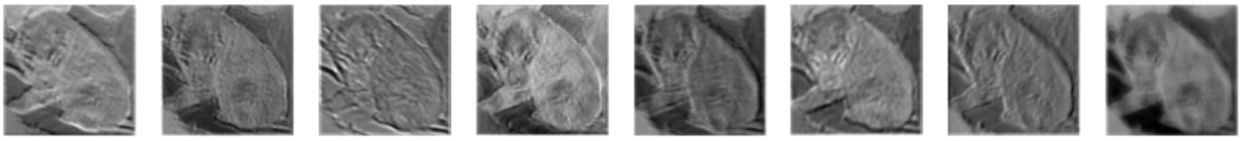
Convolutional Layer 1



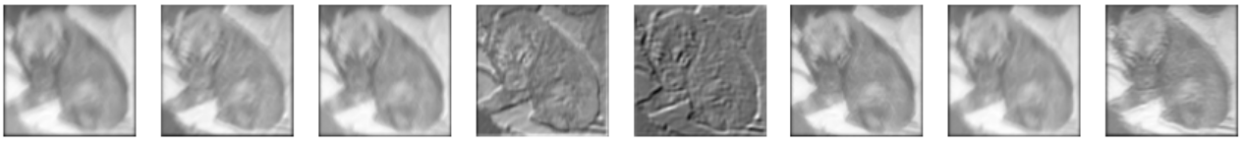
Convolutional Layer 2



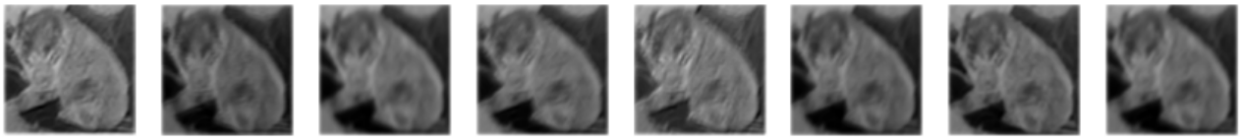
Convolutional Layer 3



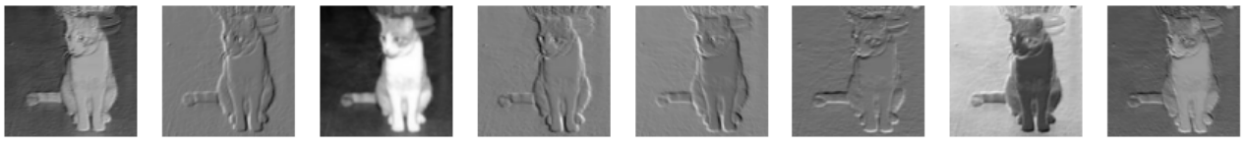
Convolutional Layer 4



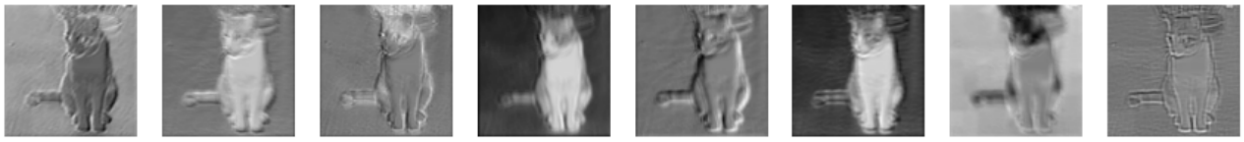
Convolutional Layer 5



Convolutional Layer 1

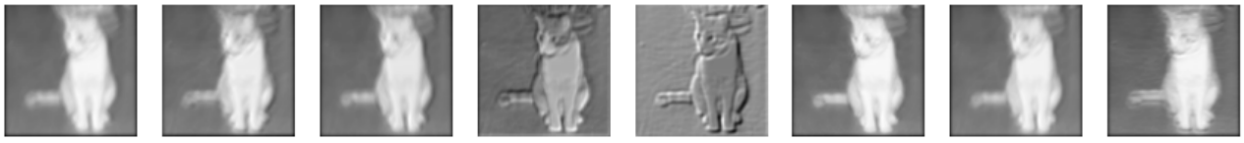


Convolutional Layer 2

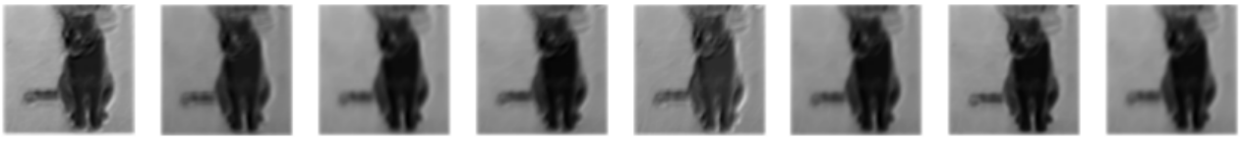


Convolutional Layer 3

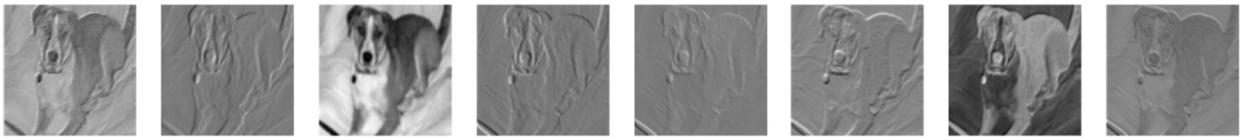
Convolutional Layer 4



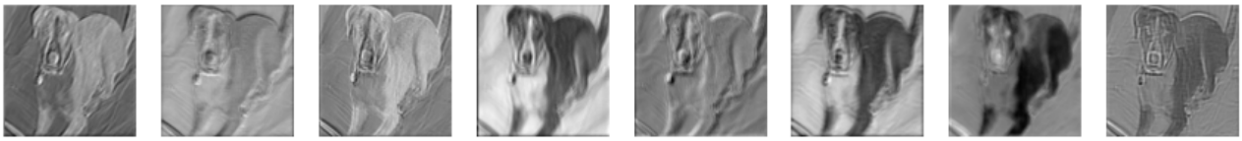
Convolutional Layer 5



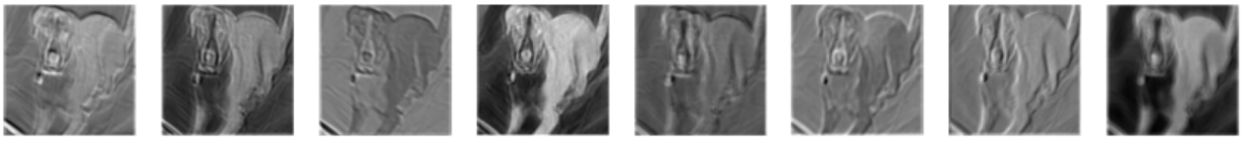
Convolutional Layer 1



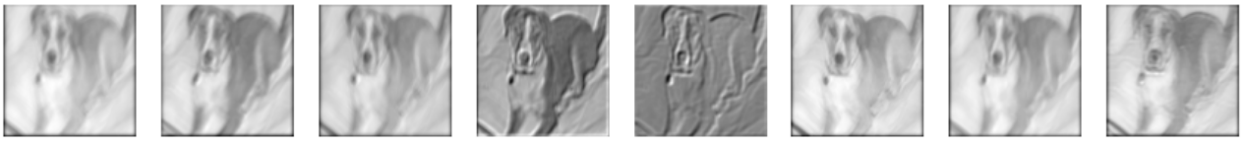
Convolutional Layer 2



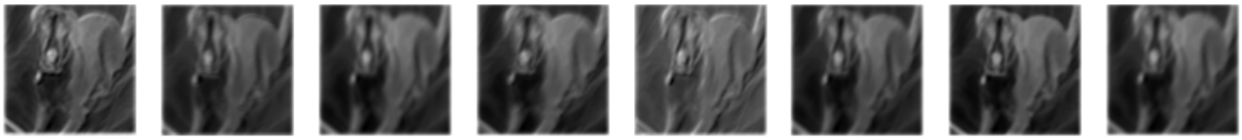
Convolutional Layer 3

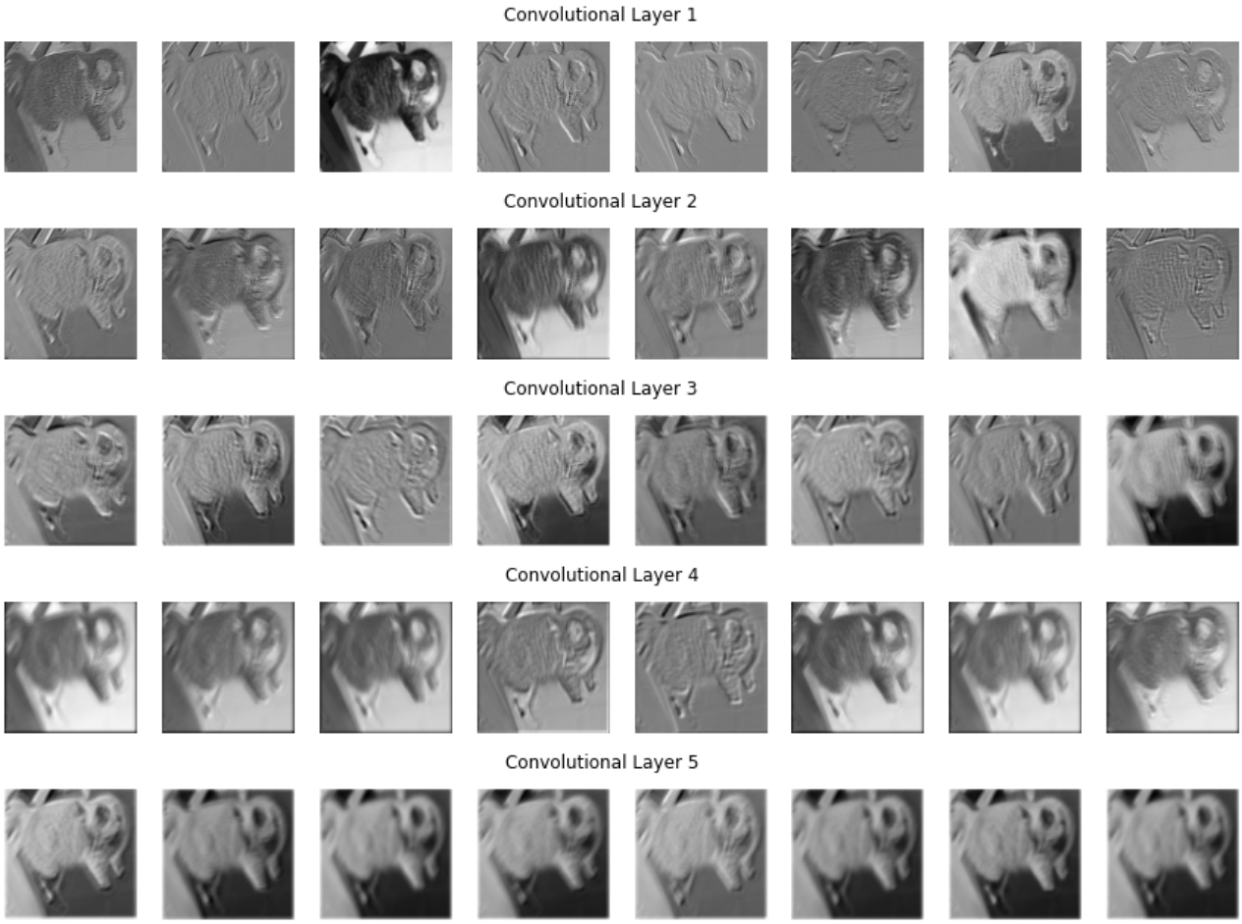


Convolutional Layer 4



Convolutional Layer 5





It’s neat to see what the different layers seem to be picking out and which features “focus” on different things. This helps me understand which features are looked for and how our model is distinguishing dogs from cats. This can also help identify blind spots and edge cases, or more generally, under which conditions and environments our model might perform well / not well.