Fine-Tuning Caffe-Net for Pill Recognition

Rohit Bhattacharya
Johns Hopkins University
Baltimore, MD, USA
rohit.bhattachar@gmail.com

Azwad Sabik
Johns Hopkins University
Baltimore, MD, USA
azwadsabik@gmail.com

Abstract

In this project, we explore the efficacy of fine-tuning neural nets. In particular, we use a model of CaffeNet that is pre-trained on the ILSVRC12 dataset and fine-tune it for the purpose of pill recognition as presented by the NIH pill recognition challenge. From our results, we see that deep neural networks learn transferable features in the lower layers, and thus, simply fine-tuning the upper layers might be sufficient for many purposes.

1. Introduction

Misidentification of drugs is a potentially fatal problem faced by the many elderly patients with several prescriptions. In order to attain familiarity with the training and implementation of neural networks while tackling this real-world problem, we used the Pill Image Recognition Challenge[1] as the basis of our project. To this end, we decided to fine-tune CaffeNet (from the Caffe[2] ModelZoo), a modified implementation of the classic AlexNet[3], for our purposes. Some papers[4] boast of the features learnt by deep neural networks as being highly transferable, and we wanted to see for ourselves to what extent this holds true. Through our project, we are able to infer some of the pros and cons of fine-tuning a neural network as opposed to learning from scratch.

2. Dataset

The pill recognition challenge provides its participants with a total of 2000 reference quality images of 1000 classes of pills (a front and back image for each pill) and 5000 consumer quality images of the same, with varying numbers of images for each pill. The reference quality images are meant to serve as examples of images from the NLM pill database and each have a uniform focus and resolution; the consumer quality images are similarly centered but have varying resolutions and levels of background space surrounding their pills. We set out on our project with the goal of finetuning a network to generate a mapping from an arbitrary image of a pill to the corresponding pill class, and thus opted to utilize all of the reference images and 4000 randomly selected consumer images as our training data-set. The remaining set of 1000 validation images was specifically held out from the remaining dataset such that it would consist of one exactly one random example image from each pill class.

Figure 1: Reference quality image

Figure 2: Consumer quality image

3. Implementation

The implementation of our project is done in Python and Caffe. We use the CaffeNet model from the Caffe Model-Zoo as our base and modify its layers for our own purposes. Since we experienced issues running Caffe on the Amazon Web cluster, we decided to run our nets instead on thin6 - a server maintained by Anton Deguet of the Laboratory for Computational Sensing and Robotics at JHU. On this server, we were able to run our net on 3 GPUs, significantly speeding up our training time.

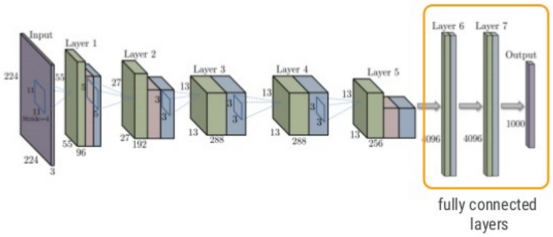


Figure 3: CaffeNet architecture

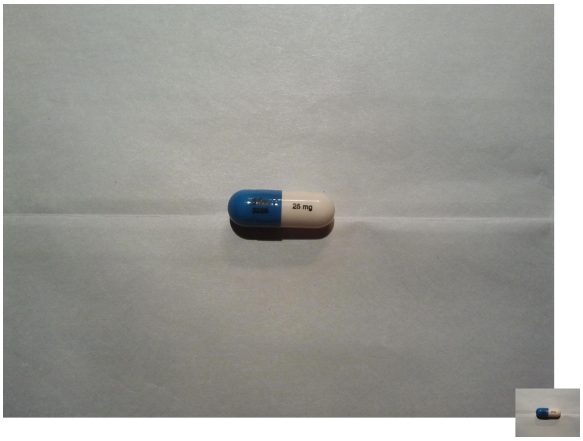


Figure 4: Preprocessing consumer quality image

4. Methods

In this section we describe the methodology used in completing this project.

4.1. Preprocessing

Given that the majority of the dataset consisted of fairly high quality (on the order of a a few megapixels) images, and given that we intended to ultimately finetune a network that would perform random crops of our data to ensure that the input image size was held fixed at 227×227, we assumed that it may be beneficial to preprocess our data via a simple series of rescales and crops. In the case of the reference images, where each pill tended to take up a large portion of its image, we simply rescaled the image such that its minimum side was rescaled to 256 pixels and the overall image's aspect ratio was maintained. We applied a similar treatment to the consumer quality images but first cropped the image down to a central region spanning half of the height and width to cut away at image background. The network followed up with these treatments by randomized crops for training, central crops for testing, and mean subtraction for images.

4.2. Fine-tuning

Fine-tuning a neural net involves utilizing some of the weights learnt by the model during previous training experiences, and completely re-learning others. The reasoning behind utilizing fine-tuning is that the lower layers of a neural net seem to learn the core, and basic features, that are easily transferable between datasets. In fact, as a preliminary experiment to motivate us to continue forward with fine-tuning a neural net, we tested CaffeNet on the pill images, as is, without any training (as ImageNet is also a 1000 class problem we did not even have to replace any layers)! The results were astounding, and quite encouraging - most clearly one could see how the net utilizes some very universal features such as colour and shape when the net classified an orange tablet as a ping pong ball!

Figure 5: Comparison of ping pong ball and orange tablet showing fine-tuning rationale

In order to implement fine-tuning, the learning rates on the layers whose weights we would like to reuse are set to zero during the fine-tuning process - this ensures that the pre-learned weights are held constant. The layers that need to be re-learnt as part of the fine-tuning, are given boosted learning rates, in order to encourage the learning of new weights for the specific task at hand.

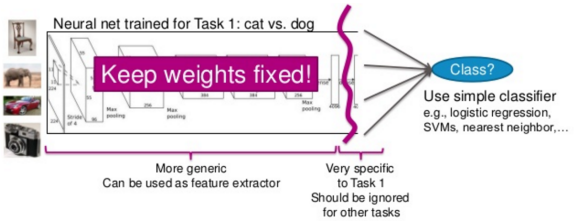
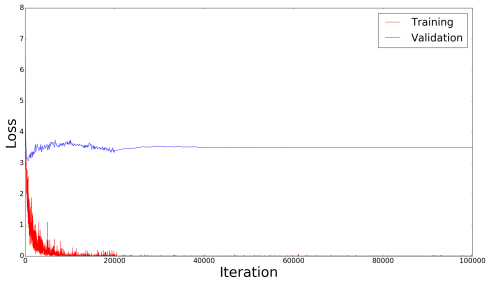


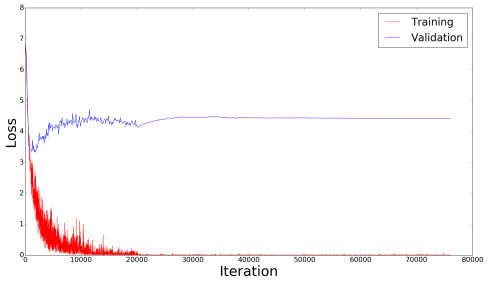
Figure 6: Fine-tuning paradigm

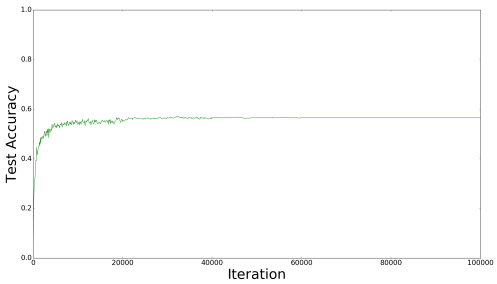
4.3. Fine-tuning CaffeNet

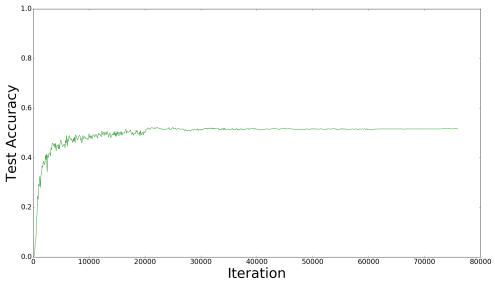
CaffeNet, as presented in the Caffe ModelZoo, is a modified version of AlexNet. It has 5 convolutional layers and 3 fully connected layers, and here, we use a version that is pre-trained on the ILSVRC12 data set. For our project, we experimented with two different modalities of fine-tuning. In the first modality (single-layer tuning), we utilized all pre-trained layers of CaffeNet, except the last fully connected layer (fc8). We boosted the learning rate of this layer tenfold, and allowed the network to re-learn it by training for 100,000 iterations. In the second modality (two-layer tuning), we experimented with re-learning two of the layers - one of them being a convolutional layer (conv3) and the other one being fc8 as before. Here too, we boosted the learning rates of the layers to be re-learnt tenfold and trained for 100,000 iterations.

5. Results









6. Discussion

6.1. Performance

Overall we were surprised at how well transfer learning can work in deep neural networks. Our network with a single fine-tuned layer achieves 56.1% accuracy while the one with two fine-tuned layers achieves only 51.6%. While these accuracies may not seem that high, we must consider that this is already a hard problem due to the high class count (1000 classes), as well as the limited amount of training data available to us (a few thousand as opposed to a million in ImageNet). This does however show the benefits, of fine-tuning - one can imagine that learning from scratch on such a small data set would not achieve nearly as high an accuracy and would also take much longer than the 50,000 iterations it took our network to converge.

6.2. Choice of fine-tuning layers

The choice of layers to be re-learned can play a huge role in determining the efficacy of the neural network. It is clear that relearning the last fully connected layer is a must, even if the output dimensions are exactly the same between two different applications. The general agreement is that, the lower layers of the neural network is where the most basic features are learnt and that these are the most transferable, so it would make sense to leave these relatively untouched. By attempting to re-learn one of the earlier convolutional layer, we were able to confirm this notion as is seen in the drop in accuracy between the 1-layer and 2-layer fine-tuned networks. Given more time however, we would liked to have also observed the impact of fine-tuning more of the higher fully connected layers, but our current finding was satisfactory enough in proving the transferability of some of the lower network layers.

7. Conclusion

This project gave us a great opportunity to learn about the inner workings and actual implementation of deep neural nets, which had so far been a mystery to us. By making and breaking various nuts and bolts such as attempting to re-learn lower layers of a neural net taught us the real value of transferability and fine-tuning of deep networks. Eventually, we hope to move on to training larger and deeper networks such as ResNet and GoogLeNet. We had in fact, tried to train these networks but ran out of memory on the server we were using. Another avenue to pursue would be deep learning even the preprocessing step - for example using a deep network to segment the pill out of each image.

References

[1] Nlm pill image recognition challenge. `http://pir.nlm.nih.gov/challenge/`. Accessed: 2016-05-11.

[2] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[4] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3320–3328. Curran Associates, Inc., 2014.