Evaluation of CNN Fine-Tuning for Image Classification

Zachary Maybury
University of Massachusetts, Lowell
91.523 - Computer Vision, Fall 2015
Zachary_Maybury@Student.uml.edu
Code: https://github.com/zmaybury/91-523-f15-project

Abstract

This paper details the evaluation of convolutional neural network (CNN) fine-tuning with the goal of learning a set of general guidelines for successfully fine-tuning a heavily trained CNN to solve an image classification task. Most datasets are relatively small when compared to the data size required for successfully training CNN's, and therefore it is difficult (if not impossible) to utilize this machine learning method on these datasets. This work focuses on evaluating methods for fine-tuning CNN's to solve both class categorization tasks (categorizing flower, bird, human, etc.) as well as sub-class categorization (species of birds), tasks which traditionally require expert (domain specific) knowledge in domains where datasets are underrepresented and the collection and annotation of new data is prohibitive. Several common fine-tuning parameters are evaluated to determine the role they play in network fine-tuning, and include the number of samples used for training and validation, the base network learning rate, the number of fine-tuning iterations, and whether to fine-tune the entire network or only the last two convolutional layers. Baselines are calculated for evaluation by training new CNN's for each dataset using no previous information or fine-tuning. These baselines are then compared to CNN's fine-tuned from the "BVLC Caffenet CNN"[1] using the previously detailed fine-tuning parameters. Overall network accuracy on previously unseen test data is used as the key evaluation metric of network performance. In addition, "per class accuracy" confusion matrices are utilized to provide further insight into network performance. This work concludes by providing general fine-tuning guidelines discovered from the above evaluation that are consistent across many fine-tuning tasks, including those for the categorization of category and sub-category. Researchers can utilize these guidelines to effectively adapt a heavily trained network, such as the BVLC Caffenet CNN, to many domains and datasets.

1. Introduction

Convolutional Neural Networks achieve state of the art performance for many image classification tasks. They are, however, "data hungry." In domains with well defined, easily distinguishable categories, and well curated, exceedingly large datasets, CNNs achieve maximum performance, as an enormous amount of data is required to iteratively train the millions of internal parameters (or weights). For example, the Imagenet dataset used to train the (at the time) state-of-the-art BVLC Caffenet CNN model contained upward of fifteen million images and ten thousand classes. Additionally, due to the dimensionality of these networks, they are ripe for classic machine learning pitfalls, such as overfitting.

For classification tasks, such as specific scenes, styles, or trends, it would be exceedingly useful to harness the power of CNNs to solve these problems, however, these datasets are usually small, consisting of only hundreds of examples per class and rarely more than one hundred thousand total samples, making them poor candidates for new CNN modes. Rather than attempt to train an entire network to solve classification problems with new datasets, fine-tuning can be utilized to adapt the features found in a large CNN such as Imagenet to new classification tasks.

Fine-tuning is the process by which weights learned from a previously trained CNN can be "adapted" to new domains and datasets. Fine-tuning has been shown to achieve state-of-the-art performance for a number of classification tasks consisting of small datasets, but the methods and parameters used to fine-tune these models has been largely unexplored. We will formally evaluate the role key fine-tuning parameters play in network performance. This evaluation will inform a set of general guidelines for fine-tuning, in an attempt to take the "trial and error" out of CNN fine-tuning and provide a deeper understanding about the role that the number of samples used for training and validation, the base network learning rate, and the number of fine-tuning iterations play in fine-tuning CNNs for classification tasks with small datasets. We will additionally explore how the fine-tuning topology (whether to fine-tune the entire network or a subset of it) effects the accuracy of fine-tuned CNNs. By taking the "guess work" out of fine-tuning (and providing a deeper understanding of the fine-tuning parameters) this work will provide a framework for the straightforward adoption of deep learning to many image domains and datasets.

2. Related Work

CNN fine-tuning was previously explored by Karayev et al. in [2] where CaffeNet was fine-tuned to detect the style of images based on 45 style categories, including

image styles such as vintage and noir, and art genres such as cubism and impressionism. Using less than one hundred thousand samples, state-of-the-art accuracies for style recognition were achieved using their fine-tuned CNN. The tuning parameters remained a largely unexplored aspect of [2] and it can be concluded that this network's parameters were chosen by trial and error. The method described in [2] is very similar to the method of fine-tuning utilized in this paper. The detection of image style comprises images containing content from a wide variety of classes represented in Imagenet, making it a good candidate for fine-tuning. In fact, the authors of [2] hypothesize that "Style is content-dependent," meaning that certain objects and features learned by Caffenet are found in only certain styles. This work will take this conclusion a step farther, by applying fine-tuning to sub-category classification tasks.

The features of deep neural networks were also researched by Yosinski et al. in [3] where the authors sought to determine how transferable the features of a deep neural network were to other tasks. The key findings of this paper showed that, "the transferability of features decreases as the distance between the base task and target task increases, but that transferring features even from distant tasks can be better than using random features."[3] This means that lower layers in a network, those similar to Gabor filters and color blobs, remain applicable to any dataset, while the later layers, which become more specific to the task at hand, become increasingly less useful. Therefore, if lower level features are generically applicable to any dataset, it begs the question of whether these features should be fine-tuned at all. While [3] only hints at the answer to this question, our work will evaluate the role the decision to fine-tune these features or not plays in network performance. Additionally, this work will confirm the findings in [3], that transferring features, even from distant tasks, is better than using random features.

Recognition using CNN features has also been explored by Razavian et al. in [4] where features extracted from the Caffenet network were used "as a generic image representation" and applied as features to train a SVM to solve a diverse range of recognition tasks. [4] achieves consistent results across classification tasks that are superior to highly tuned state-of-the-art systems. Like [3], the results detailed in [4] imply that the features learned by highly trained CNNs are extremely transferrable, even when the task to solve is extremely disjoint from the original task and data. The transferability illustrated in [4] showcases the power of CNN features, and the need for a set of general guidelines for CNN fine-tuning to apply them to new image tasks and domains.

3. Approach

First we will present a framework for creating fine-tuned models adapted from the Caffenet CNN (Section 3.1), then we present a framework for the evaluation of models to determine performance (Section 3.2), and finally our approach for identifying important CNN parameters for fine-tuning (Section 3.3) is presented.

3.1. Creating Fine-Tuned Models

Using the Caffe [5] framework, several steps need to be accomplished in order to train our network including:

1. Creation of training and validation leveldb image databases
2. Creation of solver prototxt file, specifying network fine-tuning parameters
3. Creation of training prototxt file, specifying network topology for training and validating the network
4. Creation of deployment prototxt file, specifying network topology for network to use in production
5. Use the above files with Caffe and Caffenet CNN model to fine-tune network with specified parameters

The above steps are time consuming and must be repeated for any combination of the training parameters we wish to test. Therefore, we devised a program to automatically execute the above for user provided network fine-tuning parameters. For example, as the number of samples is changed, the program automatically randomizes the dataset, constructs the appropriate training, validation, and testing split of the data, and constructs the image databases to be used by the network solver. Similarly, if the base learning rate or training iterations change, the program creates a new solver prototxt file with the appropriate parameter changes. For each set of parameters, the program creates two networks, one network where the entire network in fine-tuned and one where only the final two layers are fine-tuned. This framework allows us to fine-tune a network for any arbitrary set of parameters to be used later in our evaluation.

3.2. Evaluation Framework for Fine-Tuned Models

For the purpose of evaluation, we created a C++ framework to test an arbitrary model created by the program in Section 3.1. This program utilizes the Caffe library to:

1. Load the Network
2. Process the test set of image through the CNN
3. Calculate network performance
4. Determine per-class performance and create confusion matrix

The program is based loosely on the "cpp_classification" program included with the distribution of Caffe, turning it into an evaluation framework for an arbitrary model. Currently, this framework implements network accuracy as the only performance network, however the framework could be

easily extended to provide, precision, recall, or any other arbitrary calculation.

3.3. Identifying Important Network Parameters

With the frameworks detailed in Sections 3.1 and 3.2 in place, we are now able to set up several experiments analyzing the effects of fine-tuning parameters on network performance. This evaluation will cover that we believe to be the parameters most important to fine-tuning: the number of training and validation samples, the base learning rate, the number of training iterations, and the network fine-tuning topology.

Given the number of internal parameters in CNNs, it follows instinctively that the number of samples will be an important parameter that we should test. As more samples are provided, we would expect to be able to more accurately calculate the internal parameters to solve the specified task.

Base learning rate was also identified as an important parameter, chiefly because a learning rate that is too large will "blow away" the previously learned rates and potentially diverge, while a learning rate that is too small will make the network training extremely rigid, preventing us from learning important domain specific features.

As we are iteratively tuning the network, we determined the number of iterations to be an integral parameter to evaluate. Determining how many iterations it takes for the network converges is important, as theoretically there will be point where the network weights have achieved a minimum and the learning rate is so small that the network will no longer move regardless of further iterations. When CNNs are trained they generally are run for hundreds of thousands of iterations, at which point they are assumed to be relatively stable. However, fine-tuned networks will likely converge much more quickly than networks trained from scratch. Therefore, an evaluation of how quickly these networks converge will be useful to researchers interested in fine-tuning.

Based on the research in [-3], there is a possibility that a network fine-tuning topology with the early layers held constant may perform as well, or potentially even better than a network where all layers are fine-tuned. This is very important to know, as the execution speed of network fine-tuning is directly correlated with the number of layers that need to be tuned. A network requiring all weights to be tuned takes considerably more time for each iteration than a network requiring only some weights to be tuned. Therefore an evaluation of how the performance of the network differs given these different fine-tuning topologies will be valuable to the community.

4. Dataset

This work utilizes four datasets, consisting of face, bird, and flower images. The first dataset, "PubFig" [6] is a real-world face dataset consisting of nearly sixty thousand images of two hundred celebrities collected from the internet. Each image in this multi-class dataset is annotated with a label detailing the name of the celebrity pictured. The pictures in this dataset are extremely varied, and include images of celebrities on set, in public, on television, and in other real-world locations.

The "CUB-200 2011"[7] dataset was also used, and is a real-world bird dataset made up of nearly twelve thousand images of two hundred different bird species. Each image is labeled based on the bird species contained in the image. The pictures in this dataset include birds in a variety of environments. This dataset will hereafter be referred to as the "Bird" dataset.

The third dataset utilized by this work is the "102 Flower" [8] dataset, which consists of over eight thousand images of one hundred and two different flower species. The flowers in this dataset are common to the United Kingdom, and therefore the images share many common backgrounds and settings.

The final dataset used with this work was constructed by combining the above three datasets into a three-class dataset categorized by people, bird, and flower classes. This dataset was constructed primarily to create a dataset with very well represented classes (closer to Imagenet) to analyze how larger sample size contributes to network performance. This dataset will hereafter be referred to as the "Combined" dataset. Figure 1 shows some examples of the images in the above datasets.

Figure 1: Example images from the PubFig (top), CUB-200 2011 (left), and 102 Flower (right) datasets

5. Evaluation

First we will show the result of our experiments varying the number of samples used for fine-tuning (Section 5.1). Next we will detail the performance of networks when the base learning rate is varied (Section 5.2). Then we will discuss our findings when experimenting with the number of iterations (Section 5.3). Finally, we will showcase the

performance of our best network, utilizing the ideal fine-tuning parameters we discovered, which achieves performance on par with current state-of-the-art shown in [9] (Section 5.4).

5.1. Experiment 1: Number of Samples

The primary limiting factor when applying neural network machine learning techniques to a given dataset is a lack of data. Thus, our first evaluation concerns the performance of the network as the number of samples is increased. Table 1 shows the accuracy of three baselines, random guessing, training a new CNN with no fine-tuning, Strong DPM, and Fine-Grained CNN with no parts. These will be used for comparison to the accuracy of the fine-tuned networks. Additionally, we include the accuracies for the best network found in our sample count experiments. Random guessing for our Combined dataset, which consists of 3 classes, will achieve an accuracy of 33%. Similarly, random guessing for the bird dataset will achieve an accuracy of 0.5%. In order to validate that fine-tuning for domain adaptation is in fact better than training a new CNN, we trained CNNs for each dataset using no fine tuning. As shown in Table 1, these networks perform similarly to random guessing, when presented with the small number of samples available in the training sets. A method utilizing Strong DPM detailed in [9] achieves the best non-CNN classification accuracy, improving 38.48% over random guessing. [9] also details a fine-grained CNN model with no parts that achieves an improvement of 49.67% over random guessing. In all of the following experiments, we will keep half of the samples for testing.

Figures 2 and 3 show the results of our experiment varying the number of samples for training and validation on the Combined and Bird datasets respectively. It can be observed in the results for the combined dataset, that after 500 samples, or about 5% of the total samples, the network achieves nearly perfect accuracy. Conversely, 5% of the Bird dataset, only 5 samples, is clearly not enough, as the network continues to improve, and still has much room for improvement, after utilizing all available samples. These figures illustrate that, as expected, the more examples the are available when fine-tuning, the better the dataset will perform, and informs our first guideline when fine-tuning CNNs: Use as many examples as possible. While this rule may seem like it is not helpful for tasks where only small datasets are available, one can observe in Table 1 that near state-of-the-art accuracy is achieved for the Bird dataset after only 50 samples per class. This is not a prohibitively large number of samples, meaning that, if a dataset does not exist for a particular task, one could reasonably be constructed and used with a CNN fine-tuning method.

Classification Method	Combined Dataset Accuracy	Bird Dataset Accuracy
Random Guessing	33%	0.5%
New CNN with no fine-tuning	**31.3%**	**0.51%**
Strong DPM [9]	N/A	38.02%
Fine-Grained CNN no parts [9]	N/A	50.17%
Fine-tuned CNN	**99.6%**	**44.68%**

Table 1. Baseline Classification Accuracy for Combined and Bird datasets. (Our approaches in **bold)**

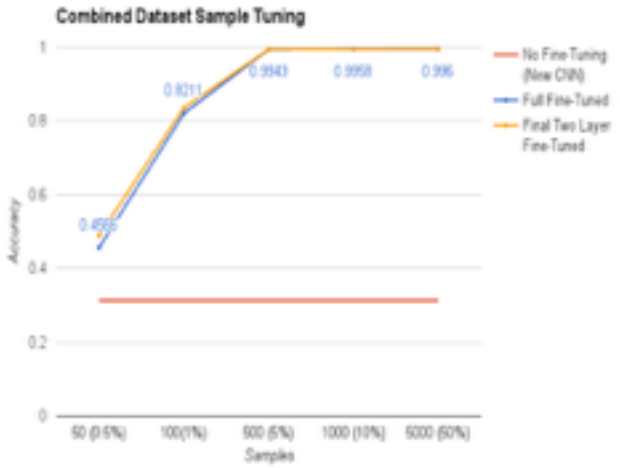

Figure 2: Combined dataset sample tuning for a fully fine-tuned network (blue) and fine-tuning only the final two layers (yellow)

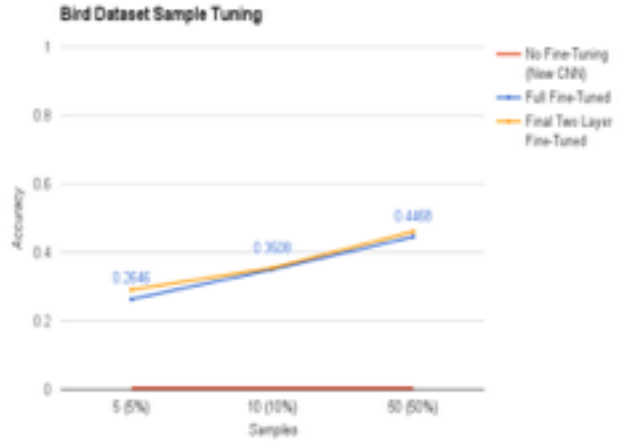

Figure 3: Bird dataset sample tuning for a fully fine-tuned network (blue) and fine-tuning only the final two layers (yellow)

As can be seen in Figures 2 and 3, the fully fine-tuned network performs slightly worse than when fine-tuning only the last two layers. This confirms the finding in [-3] that the low level features contained the first layers are extremely transferrable across tasks. Additionally, fine-tuning actually performs slightly worse when the entire network is tuned. This implies that the low level features learned from a massive dataset do not need any further tuning, allowing us to speed fine-tuning processing time (and improve performance!) by only training the final two layers representative of the high level, in domain, concepts.

5.2. Experiment 2: Base Learning Rate

As previously indicated, the base learning rate is also an important factor in CNN fine-tuning. Figures 4 and 5 show the result of varying the base learning rate for fine-tuning for the Combined and Bird datasets respectively. In each of these experiment, all available training and validation samples were utilized during fine-tuning, as this resulted in maximum performance as detailed in Section 5.1.

As can be seen in the figures, choosing an appropriate base learning rate is extremely important when fine-tuning a network. For larger learning rates, these networks quickly diverge, yielding a useless network. Additionally, if the learning rate is too small the network becomes too rigid, and suffers to learn appropriate features. In some cases, we observed a learning rate that was too small also leading to network divergence.

Similar to Experiment 1, we see superior performance for both datasets when only fine-tuning the final two layers of the network. Additional, fully fine-tuned networks are extremely less robust to a poor base learning rate choice. In fact, across tasks and regardless of other parameters, we found a base learning rate of 0.0001 to achieve maximum fine-tuning accuracy. The only remaining constant across our experiments in the Caffenet model we fine-tuned from, therefore we conclude that this value is well suited for any fine-tuning tasks using the Caffenet model. These findings inform our second guideline when fine-tuning CNNs: Use a base learning rate at or near 0.0001.

5.3. Experiment 3: Number of Iterations

Our final experiments analyze how the number of iterations effects fine-tuning performance. Figures 6 and 7 show the result of varying the number of iterations for fine-tuning for the Combined and Bird datasets respectively. In these experiments, all available training and validation samples where utilized during fine-tuning, as this results in maximum performance as detailed in Section 5.1. Additionally, a base learning rate of 0.0001 is used as this also results in maximum performance as detailed in Section 5.2.

As can bee seen in the figures, each network achieves maximum performance between 1000 and 5000 iterations,

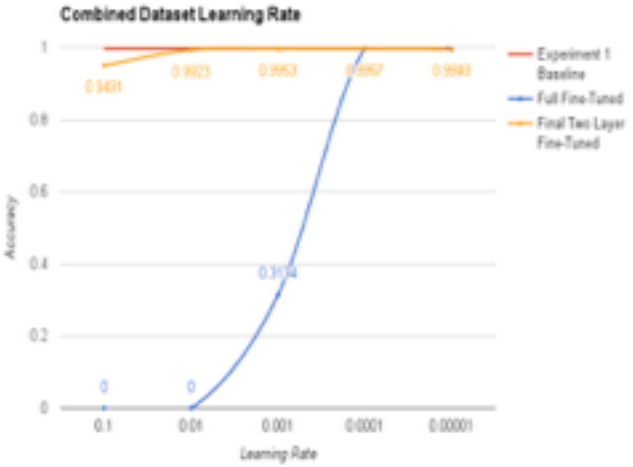

Figure 4: Combined dataset base learning rate tuning for a fully fine-tuned network (blue) and fine-tuning only the final two layers (yellow)

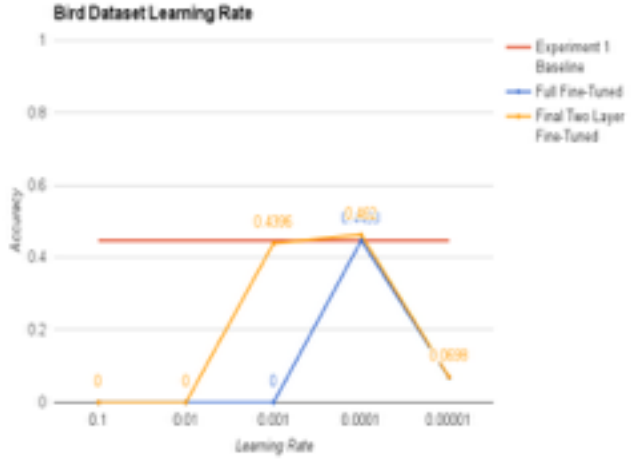

Figure 5: Bird dataset base learning rate tuning for a fully fine-tuned network (blue) and fine-tuning only the final two layers (yellow)

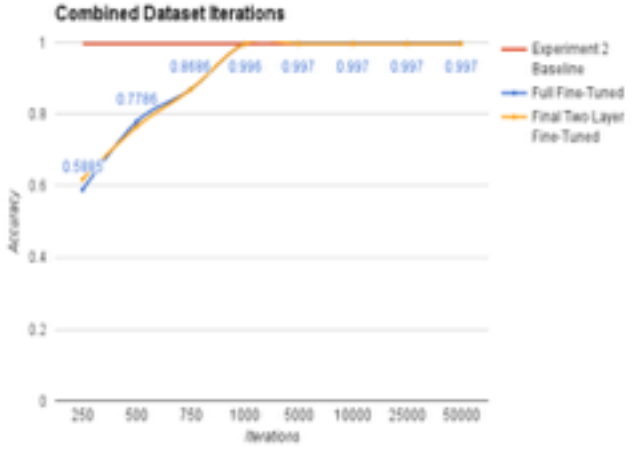

Figure 6: Combined dataset iterations tuning for a fully fine-tuned network (blue) and fine-tuning only the final two layers (yellow)

at which point further training produces no additional gains. These findings inform our third guideline when fine-tuning CNNs: Maximum performance is achieved between 1000 and 5000 fine-tuning iterations. In each case, after 5000 iterations the network weights are well established, the learning rate has become extremely small, and the network has achieved a global minimum. At this point, further training becomes unproductive and unnecessary.

Similar to experiments 1 and 2, we see superior performance for both datasets when only fine-tuning the final two layers of the network. The fully fine-tuned network consistently underperforms the network where only the final two layers are fine-tuned, which in concert with our findings in experiments 1 and 2, informs the fourth, final, and possibly most important guideline when fine-tuning CNNs: Hold lower levels constant, only fine-tune final layers. By following this guideline, we can create a hybrid CNN for our task, maintaining the diverse low level features of the heavily trained CNN and learning the high level features necessary to solve our domain specific task.

5.4. Best Networks

Using the above guidelines, we fine-tuned CNNs for both the Combined and Bird datasets, the later of which achieves near state-of-the-art [9] accuracy. Utilizing the above guidelines, these networks use all available training samples, have a base learning rate of 0.0001, fine-tune for 5000 iterations, and utilize a fine-tuning training topology where only the last two layers are fine-tuned. When using these parameters we are able to achieve exceptional performance for tasks involving our Combined and Bird datasets.

Table 2 shows a confusion matrix for our Combined fine-tuned CNN. Interestingly, most model confusion comes from confusing birds for flowers or vice-versa. This is not surprising as bird and flowers are found in common natural environments, while the person image samples come from a wide range of environments. Nevertheless, when tested against a test set of almost 20 thousand images, the network achieves a classification accuracy of 99.7%.

Figure 8 shows the confusion matrix for our Bird fine-tuned CNN. While there is still some cross-class confusion, overall classification accuracy on our test set is 63.76%, in line with the state-of-the-art non parts CNN model detailed in [9]. This result is achieved with less than 50 samples per class, which is on par with the number of per-class samples seen in many small datasets. The success shown by this model, utilizing so few training examples to solve an expert classification task, provides great promise for applying these techniques to similar, underrepresented datasets.

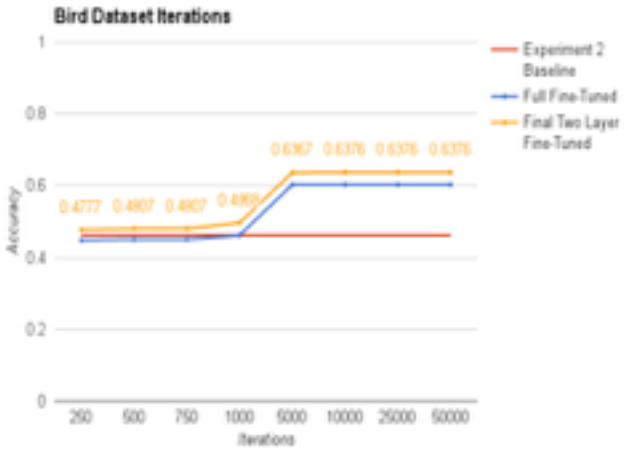

Figure 7: Bird dataset iterations tuning for a fully fine-tuned network (blue) and fine-tuning only the final two layers (yellow)

Class	Bird	Flower	Person
Bird	0.9937	0.0065	0.0022
Flower	0.0014	0.9977	0.0002
Person	0.0013	0.0011	0.9974

Table 2: Confusion matrix for Combined fine-tuned CNN.

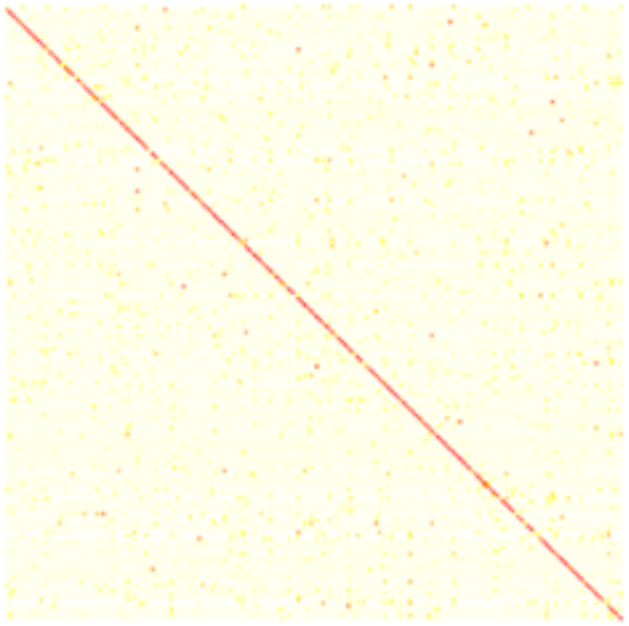

Figure 8: Confusion matrix for Bird fine-tuned CNN. Top 3 classification accuracies are Green Violetear (90.48%), Geococcyx (88.1%), and American Three Toed Woodpecker (86.67%).

6. Conclusion

In this paper, an evaluation of CNN fine-tuning for

image classification is presented, and several general guidelines for successful fine-tuning are outlined. These guidelines include utilizing as many samples as possible, using a base learning rate of 0.0001, fine-tuning for at most 5000 iterations, and most importantly only fine-tuning the final two layers of the network. This paper concludes by showing the successful application of these guidelines to two classification tasks, achieving results comparable to state-of-the-art.

Future work includes further validation for the guidelines described in this paper, specifically how they perform in other domains, especially those farther from the Caffenet model used for fine-tuning.. Additionally, there are many other parameters involved in CNN fine-tuning that, while seemingly unimportant, could prove to be very important. Finally, further analysis of fine-tuning training network topology is required, but was beyond the scope of this work. Here we compared fine-tuning entire networks vs. fine-tuning only the last two layers. While fine-tuning of the last two layers outperformed fine-tuning the entire network, it is possible that even better performance may be achieved by fine-tuning only the last layer, fine-tuning the last three layers, or some other combination of layers.

References

1. BVLC Caffenet, https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet
2. Karayev, et Al. Fine-tuning CaffeNet for Sytle Recognition on “Flickr Style” Data, 2014. http://sergeykarayev.com/files/1311.3715v3.pdf
3. Yosinski et Al. How transferable are features in deep neural networks?. In *NIPS Advances in Neural Information Processing Systems*, 2014
4. Razavian et Al. CNN Features off-the-shelf: an Astounding Baseline for Recognition, 2014. http://arxiv.org/abs/1403.6382
5. Caffe: Deep Learning Framework, http://caffe.berkeleyvision.org
6. PubFig: Public Figures Face Database, http://www.cs.columbia.edu/CAVE/databases/pubfig/explore/
7. Caltech-UCSD CUB-200 2011 Database, http://www.vision.caltech.edu/visipedia/CUB-200-2011.html
8. 102 Flower Dataset, http://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html
9. Zhang et Al. Part-based R-CNNs for Fine-grained Category Detection, 2014. http://arxiv.org/pdf/1407.3867.pdf