Optimizing a Shallow Multi-Scale Network for Tiny-Imagenet Classification

Dash Bodington
Stanford University
450 Serra Mall, Stanford, CA 94305
dashb@stanford.edu

Abstract

This project explores the structural optimization of a shallow multi-scale neural network. Its dual goal is to acheive high classification accuracy on the Tiny Imagenet dataset within certain network limitations, and to obtain insight into the scale importance and properties of images. The simple multi-scale convolutional neural network implemented is able to acheive 19.1 % classification accuracy on the provided validation set, used as a test dataset. This value is found to be the limit for the network limitations imposed, but with slight modifications, similar networks can reach up to 25 % accuracy. The learned classifier is analyzed in terms of its filters, and is also applied to some non-classified fractal images, since fractal analysis is a multi-scale process as well.

1. Introduction

This project began as an exploration of fractal-like properties in the Tiny-Imagenet dataset, but has evolved to be an optimization of a specific multi-scale network for classification purposes on the same dataset.

In the measurement of fractals, or the fractal properties of images, a common analysis method is to calculate specific statistics on regions of an image which are different sizes. One of these algorithms is the box counting algorithm, a method which calculates a fractal dimension, or an amount of self-similarity in an image. Before the beginning of this project, the author was working on a regional version of the box counting algorithm which is used in image segmentation. Calculating statistics with different box sizes, however, is a technique which is not limited to fractal analysis. Similar calculations with more degrees of freedom, happen in multi-scale convolutional networks. Multi-scale networks are convnets which contain parallel processing channels containing convolutional kernels of different sizes, thus processing an image at multiple scales.

Just as fractal properties give insight into the self-similarity of an image, a multi-scale network may be able to give insight into other structures of an image, and by optimizing the structure of a small multi-scale network for classification, it may be possible to learn about specifically what structures and scales are important for image classification in the Tiny-Imagenet dataset.

The trained network will also be tested on a set of fractal images which do not belong to any of the imagenet classes, and analysis will be done to see if any interesting classifications arise. For example, with the high level of self-similarity shared by all of the fractal images, the most commonly predicted class for the classifier may be the most self-similar class in Tiny-Imagenet.

Most high performance models for classification, including most of those in the Tiny-Imagenet challenge, deal with the optimization of training, rather than optimizing the structures of networks. Data augmentation, such as adding crops, rotations, and other permutations of images, as well as filtering out bad training data covers the training data optimization, while tools like dropout and fine tuning are examples of training tricks often used. These tools, though they help performance of trained models, do not contribute to the understanding of images from the human point of view. In general, deep networks work better from a classification point of view, but they quickly become too complicated and confusing to teach us anything about the images we are attempting to classify.

With a simple, shallow, network, this project hopes to contribute to the understanding of images. By adjusting model structure parameters on a hand-tunable scale, and setting limitations on the size and training of a model, it will hopefully be possible to discover what factors contribute most strongly to the classification of an image in this architecture. With information on the optimal multi-scale network with constraints imposed, a number of larger networks based on this optimal mini-network will be tested for performance.

By the conclusion of this project, we hope to know what fractal-statistic-like calculations are optimal for image classification, what methods and structures are not optimal, and how these optimal conditions may contribute to larger net-

works.

2. Background

Multi-scale methods are not new to convolutional neural networks and are the basis of similar fractal calculations which are both used frequently when it comes to medical imaging. Previous applications of fractal calculations and shallow multi-scale convolutional neural networks fall frequently into the medical field. Fractal textures and other multi-scale method have helped distinguish growths in the lungs [7], detect microcalcifications in a breast which warn of breast cancer [2], and are involved in other analysis of medical images where fractal structures tend to be common. There are many fascinating uses of multi-scale analysis in medical and other natural growth and structures imaging, but because shallow multi-scale networks often do not perform well when it comes to large scale classification tasks, the current rage in deep learning, they are generally not used for such purposes.

This project is therefore taking an unusual approach to the application of shallow networks to a problem (Tiny-Imagenet) better suited for deep networks in the hope that it's learned parameters and properties, not it's classification accuracy, will be the interesting result.

3. Procedures

Because of the time, hardware, and iterative requirements of this project, several basic constraints had to be imposed on the models and methods involved. Constrains provide a framework for the modification, training, and testing of various models. This allows for a reasonable rate of progress, but also limits the performance of each model so that optimality only holds under the constraints imposed, and someone with more time, faster hardware, or different networks may be able to create higher-performing models.

3.1. Dataset

The Tiny-Imagnet dataset consists of 100,000 images uniformly divided in to 200 classes. The dataset also includes validation and training sets of 10,000 uniformly distributed images. For this project, the validation set is used for testing, since its true labels are known, and testing frequency is not limited by a test server.

3.2. Basic Multi-Scale Convnet Structure

The basic structure of the convnet in this project is the following: There are four parallel scale channels. The scale channels contain convolutional filters which are 9x9, 7x7, 5x5, and 3x3 in size. Each convolutional layer is followed by a rectified linear unit and a pooling layer. In the simplest model, the outputs of these four channels are concatenated and fed to two fully connected layers using a softmax loss function to calculate probabilities for each of the 200 classes.

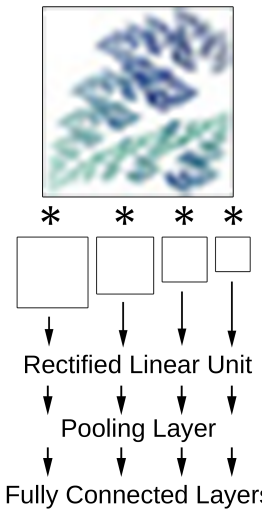

Figure 1. An illustration of the structure of the basic multi-scale network [4].

3.3. Implementation and Hardware Details

All convolutional neural networks in this project are implemented in Caffe [5], the Berkeley Vision and Learning Centers deep learning framework. Caffe allows for simple implementation and modification of convolutional neural networks and supports fast training and testing by offering acceleration on CPUs and GPUs with fast linear algebra libraries, and Nvidias CUDA and cuDNN.

Convnets implemented in Caffe were run on an Nvidia GeForce GTX 780. The GTX 780 has 3072 MB of RAM, a limiting factor when it comes to batch size, and generally allows for the acceleration of training and testing by a factor of 70 over identical CPU-only implementations.

3.4. Training Constraints

Though the GPU allows for a significant reduction in training time, fully training many versions of a network to determine optimal properties is very time consuming. For this reason, a limit of 20,000 training iterations, with a batch size of 32 images, was implemented. This limit allows for relatively fast training of a model. Depending on model size, this number of iterations takes approximately ten minutes, after which it is possible to see the convergence of the validation accuracy of the model.

Because many of the network changes made are small, learning rate does not need to be changed often. With significant changes in the network, however, the learning rate was tuned by hand to achieve a similar level of convergence for each model tested. It is important to recognize that be-

cause of these training constraints, the results of this project represent efficiency optimality, and though an amount of convergence was achieved for every model presented, it is possible that efficiency optima do not translate completely to global optima.

Training method was constant for all training. Stochastic gradient descent with momentum and learning rate decay was used. AdaGrad and Nesterov's Accelerated Gradient proved to be too slow for the training speed desired.

3.5. Model Modifications

To find optimal parameters for the structure of this convnet, many variables were changed in the structure of the network flow, in the network size, and in the details of each step in the network. The largest of the changes explored are summarized below.

- **Multi-scale depth:** The total number of convolutional layers split between the multi-scale channels was varied to find the optimal number while maintaining a small network size.
- **ReLU type:** The ReLU following each convolution was swapped for a leaky ReLU, and the negative slope of these ReLUs was modified.
- **Pooling Size/Stride/Type:** The pooling size, stride, and type were changed. Maximum and average pooling were tested.
- **Power Layer:** To examine the calculation of various statistics of an image within the fully connected structure, a power layer was added after the pooling layer, and the power was varied.
- **Number of Fully-Connected Neurons:** The number of neurons in the first fully-connected layer was modified.
- **Distribution of Fully-Connected Layers:** With the total number of convolutional layers fixed, the distribution of these layers between the various scales was varied.
- **Larger Changes:** Several larger networks containing the optimized multi-scale element were constructed, trained, and tested, to examine the properties of this element.

4. Results

The results of this project are comprised of several parts. Primarily, there is the structure optimization of the multi-scale network, which reveals the effects of changing parameters of the network, and works toward the optimal structure. Second, is a brief analysis of the learned weights of the optimal network, though time constrants have limited the depth of this section. Finally, a set of fractal images was classified with the trained network, and presented.

4.1. Network Structure

- **Multi-scale depth:** Beginning from the basic network, 16, 8, 4, and 2 filters per multi-scale convolutional channel were tested with varying numbers of training iterations.

Validation Accuracy vs. Training Iterations			
Filters per Scale	10k	20k	30k
16	0.119	0.137	
8	0.131	0.147	0.153
4	0.137	0.139	0.149
2	0.129	0.136	0.136

Figure 2. Based on the results in this table, it was decided to move forward with 8 convolutional filters per scale channel, 32 total, for the highest accuracy at the affordable training duration of 20,000 iterations.

- **ReLU type:** Modifying the rectified linear unit to be a leaky ReLU improved the network accuracy slightly, however, negative slope values above 10^5 had a negative effect on accuracy.
- **Pooling Size/Stride/Type:** First, the pooling type was changed to average, and the accuracy fell to 90
- **Power Layer:** Adding a power layer to the network between the concatenation layer and the fully connected layers was a failure. The use of any power much larger than 1 mandated that the learning rate be significantly decreased, and made the network untrainable. Powers below 1 were tested as well, but the shift required to make all values in the concatenation layer greater than zero made the network ineffective. The power layer was not used moving forward.
- **Number of Fully-Connected Neurons:** The validation accuracy of the model increased with the number of elements in the first fully connected layer, but plateaued after 100 elements, so the size was fixed to contain 100 outputs.
- **Distribution of Fully-Connected Layers:** Surprisingly, the distribution of the filters, totalling 32, did not affect the network significantly
- **Larger Changes:** Several larger networks were constructed from the finalized shallow multi-scale network, but none of them proved to be useful. The first attempt was to connect two of these networks in series, so that the concatenation layer in the first network fed into another multi-scale set of convolutional layers. This network was more difficult to train, and could not match the accuracy of the original network. Another attempt to improve this network was to double the length of the parallel multi scale channels, so that before the concatenation layer, the data had been fed through two sets of conv-relu-pools with the same

Filter Distribution and Accuracy				
Filter Sizes				
3x3e	5x5	7x7	9x9	Acc
16	8	4	2	0.191
2	4	8	16	.185
4	12	12	4	.183
1	1	1	29	.184
1	1	29	1	.184
1	29	1	1	.184
29	1	1	1	.192

Figure 3. Changing the distribution of the convolutional filters did not significantly effect the validation accuracy. If anything, the model favored small filters, but to maintain a strong multi-scale structure without much sacrifice of accuracy, a uniform distribution of filters was chosen.

properties. This modification also did not improve performance.

The most successful improvement to the network was made by repacing the first fully connected layer with a 3x3 convolutional layer with 128 filters, and then pooling 2x2 with stride 2 before the final fully connected layer. This model acheived a validation accuracy of 0.248, significantly better than the 0.191 average of the smaller network alone.

4.2. Trained Network

Because the network in this project is relatively small, it is possible to visualize all of the filters and look for interesting features.

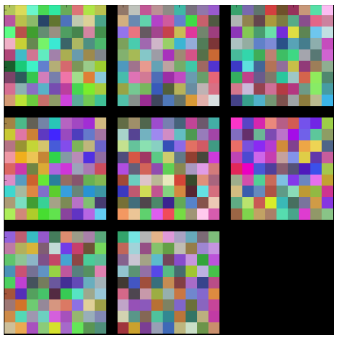

Figure 4. The 9x9 features of this network. These large filters present some clear features, best imagined as gradients accross the kernels, and mostly taking the form of diagonal structures.

5. Conclusions and Extended Results

The objective of this project was to learn about the Tiny-Imagenet dataset through trained analysis and extracted multi-scale features. Though there is more work to be done, the results so far have been interesting.

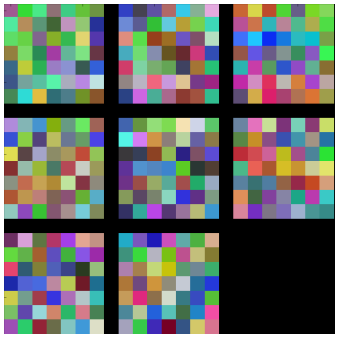

Figure 5. The 7x7 features of this network. These filters contain recognizeable gradients and patterns as well, though it is not clear exactly what they may detect.

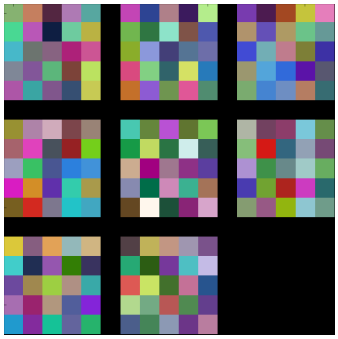

Figure 6. The 5x5 features of this network. At this size it becomes harder to distinguish patterns from what may be noise, but these filters seem more symmetrical than the previous.

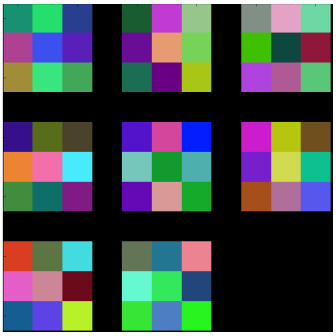

Figure 7. The 3x3 features of this network. These small filters make it difficult to imagine image features.

For a multi-scale convolutional neural network of the structure used in this project, 20 % accuracy seems to be a hard limit on its capability, no matter the distribution of filters or values of various parameters. Most of the changeable parameters for a network of this structure were explored, and while it was possible to bring the network accuracy up from an initial 12 % to 19.1 %, no amount of modification or extended training was able to increase the accuracy significantly further. In the latter stages of the project, some simple data augmentation such as random crops, mirroring and rotation, and image mean consideration were per-

formed. These tricks, however, did not yield measurable improvements in the accuracy, and in some cases slowed down training, hindering progress.

Observing the limits of this shallow netowork make it clear why deeper networks with smaller filters are often preferred, but as Figure 7 illustrates, it is difficult to find patterns and useful information in 3x3 filters. To be able to learn from the filters, we must use larger sizes, which are inherently more computationally intensive.

19.1 % accuracy is still a decent result to obtain from a shallow network, training from scratch. Unllike fine-tuning a large, deep network, the variation and exploration of shallow architectures helped develop an intuitive sense of the options in convolutional neural network construction. The greatest takeaway from this project is the intuition gained by adjusting the network structure, which may be useful when considering modifying larger convolutional neural networks as well.

5.1. Extended Results

Because this project was based loosely on fractal-type processing of images, a set of 100 fractal images which to not belong to any class in Tiny-Imagenet were classified, and some of the more interesting results are presented below.

Of the 200 classes in Tiny-Imagenet, the fractal images were distributed far from uniformly, as should be expected because only some types of images can be generated as fractals. The most popular class for the fractals was "Spider Web," which had 15 images classified under it. In general, these images contained thin lines and radial patterns which made this classification logical. Next was "Scoreboard" with 8 images, and "Centipede," "Brain Coral," and others, had 5 images. Many of the fractal images do appear somewhat similar to what they are classified as. "Brain Coral" is one of the best examples, which itself appears a bit like a fractal.

A class which was not expected to show up in the popular results was "Scoreboard." A scoreboard, while it may contain some rectangular geometric patterns which could be part of fractals, is generally inconsistent with the fractals assigned to it in Figure 9.

6. Future Work

Because of the short time frame of this project, there was a great deal of area left unexpored. Likely the most interesting analysis left undone would have been to visualilze how images pass through the shallow multi-scale network. Because drawing conclusions directly from the filters is difficult, seeing how specific images change as they are filtered, rectified, and pooled would have been fascinating. This investigation will still be done, but not within the timeframe allowed for the project.

Figure 8. "Brain Coral" fractals, resized to maximum dimension 32 without distortion [4].

This work in progress will continue, and will hopefully lead to more interesting information on what is important in images. With the exploration of a medical image database, it is hoped to directly compare fractal analysis to similarly structured, learned analysis, which likely offers improvements over current techniques due to its flexibility.

References

[1] M. N. Ahmed and A. A. Farag. Two-stage neural network for volume segmentation of medical images1. *Pattern Recognition Letters*, 18(1113):1143 – 1151, 1997.

[2] L. Bocchi, G. Coppini, J. Nori, and G. Valli. Detection of single and clustered microcalcifications in mammograms using fractals models and neural networks. *Medical Engineering Physics*, 26(4):303 – 312, 2004.

[3] R. Candela, G. Mirelli, and R. Schifani. Pd recognition by means of statistical and fractal parameters and a neural network. *Dielectrics and Electrical Insulation, IEEE Transactions on*, 7(1):87–94, Feb 2000.

[4] E. Foley. Empyreal light (www.elfractal.com), 2013.

[5] 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.

[6] S. Kerdpiboon, W. L. Kerr, and S. Devahastin. Neural network prediction of physical property changes of dried carrot as a function of fractal dimension and moisture content. *Food*

Research International, 39(10):1110 – 1118, 2006. Physical Properties {VI}.

[7] N. F. Vittitoe, J. A. Baker, and C. E. F. Jr. Fractal texture analysis in computer-aided diagnosis of solitary pulmonary nodules. *Academic Radiology*, 4(2):96 – 101, 1997.

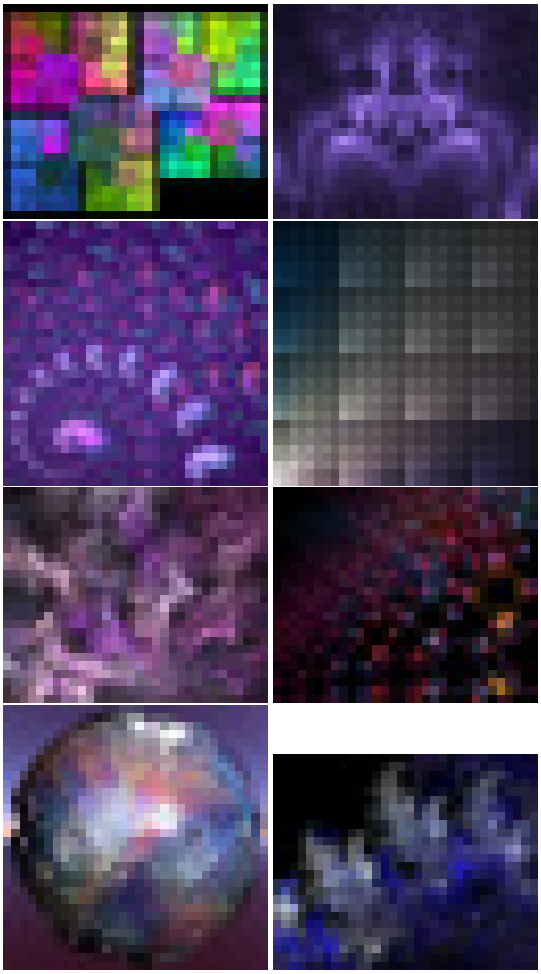

Figure 9. "Scoreboard" fractals, resized to maximum dimension 32 without distortion [4]. Only two of the fractals contain the rectangular geometries expected in a scoreboard.