Image Recognition: Dog or Horse

Convolutional Neural Layer Fusion Via Majority Voting Nearest Neighbors

Jared Seifter
Johns Hopkins University
May 2018

Abstract

Neural networks for image recognition use convolutional neural layers inspired by similar designs in the human body for vision to extract features from images that are relatively invariant to changing conditions such as lighting or angle of view. Decision fusion of neural networks has been used to reduce misclassification and can also allow for a steeper learning gradient for smaller training sets. While biologically inspired convolutional neural networks have been found empirically to perform best for object recognition, the feed forward portion trained via supervised learning is highly prone to overfitting on small datasets. Therefore, an optimal image recognition algorithm for small datasets should combine multiple convolutional neural layers for the most generalizable feature extraction possible, but use a classifier optimized for small data, nearest neighbor, as the final step to classify and fuse those results via majority voting. When examining this strategy to distinguish between dogs and horses with fewer than 100 training sample each, the machine learning algorithm was able to perform with an accuracy of 78.38%.

Introduction

Classical artificial intelligence started in the 1950s by taking a cognitive based approach to model constrained problems such as playing games. However, this method proved unsuccessful when applied to larger real-world problems with uncertainty and error. Artificial intelligence took ideas from statistics to build machine learning algorithms that can train on real world problems and account for noise and incomplete information. These machine learning algorithms were incredibly powerful because it was not necessary to understand and include each variable as collections of related variables were learned via the data. However, machine learning algorithms require large quantities of training data to properly account for edge cases and to strengthen the induction hypothesis inherent within the methodology (*Machine Learning*, 2018).

The high dimensionality of the linear algebra required makes training and testing large datasets difficult even when it is available. Many machine learning algorithms are constrained by the compute requirement as well as the working memory requirement. The simplest solution is horizontal expansion of processors into large clusters to map and reduce the training sets but often the I/O losses do not translate well for calculations that cannot be constrained within isolated chunks of data. One approach is to create dedicated hardware built to increase speed for specific machine learning approaches, but a more generalizable strategy leverages the graphical processing unit (GPU) to accomplish similar speed ups without the vendor lock-in and reusability issues. GPUs allow for the parallelism of large clusters while utilizing graphical memory to minimize I/O constrains. In addition, GPUs are built for imprecise calculations

which are perfect for complex probabilistic linear algebra with floating points as precision of a specific variable is not a concern (Beckerman , 2011).

Large, multilayer artificial neural networks have become possible by the increasing power and availability of GPUs and the tools to use them. NVIDIA cuDNN is a library for building neural network layers explicitly using CUDA and is optimized specifically for taking advantage of the large amounts of graphical memory and parallel compute on an NVIDIA GPU (Catanzaro, 2014). The deep neural networks made possible using cuDNN and other libraries are incredibly valuable versus other less expensive machine learning approaches that can also utilize the GPU because of the strength of its automated feature detection which decreases the need for (but does not eliminate) domain expertise and reduces bias inherent within human written feature detection algorithms (Glorot, 2010).

Image Recognition with Few Training Examples

Neural networks for image recognition use convolutional neural layers inspired by similar designs in the human body for vision to extract features from images. These layers are valuable because they process the image which can be considered a three-dimensional matrix of intensities and colors to create features that are relatively invariant to changing conditions such as lighting or angle of view (Kheradpisheh, 2016). When a human being is looking at a dog and it trying to recognize that it is a dog, from a matrix of color intensity perspective, the dog could be a completely different shape. When it is dark out, that dog will not appear to be the same color during the day and even colors of nearby objects can reflect off the dog and cause contamination of the ground truth. However, human being and other animals can immediately recognize a dog in changing conditions, even if the individual has never seen a dog in a similar situation previously. There is no cognitive or reasoning step, the biological neural network perceives the dog as the "same" due to the implicit feature extraction occurring on the original image. The same concepts and neural structure is applied in machine learning to create convolutional neural layers which have been shown empirically to perform well for image classification and object recognition tasks (Krizhevsky).

Convolutional neural layers have been used successfully for proper feature extraction of images for object recognition, but combining results from multiple different networks can allow for more robust feature detection. Decision fusion of neural networks has been used to reduce misclassification by 30% in optical character recognition and can also allow for a steeper learning gradient for smaller training sets because of the expanded correlative domains inherent within using multiple feature strategies (Rogova, 1994). Majority voting as the mechanism for decision has been shown successful in a variety of domains and is particular valuable for smaller datasets because of its lack of complexity which can reduce overfitting (Suen, 2000)

While biologically inspired convolutional neural networks have been found empirically to perform best for object recognition because of the strong optical feature extraction, the feed forward portion trained via supervised learning is highly prone to overfitting because of the complexity. The high number of nodes allow for the modeling of incredibly complex problems but it also makes it prone to building an overly complex model to fit the data available instead of a worse fit that is more generalizable. This is shown below in Figure 1 when an n-degree fit has

a higher accuracy on the training data but is less representative and will likely perform worse on test data (*Over-fitting?*, 2018).

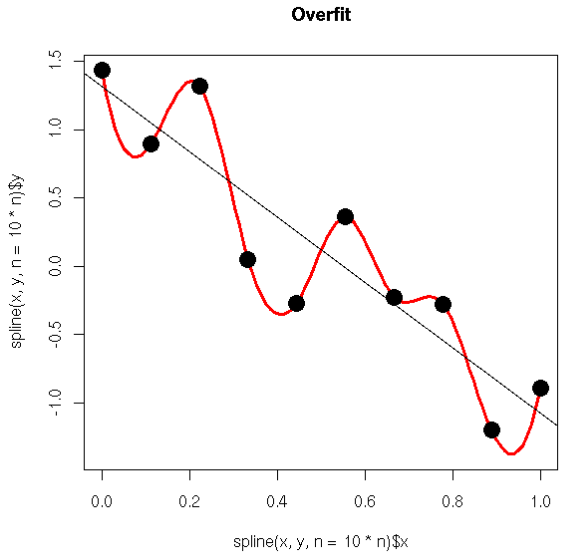

Figure 1: Graph showing an example of overfitting

Convolutional neural layers have been successfully split from the remaining network to keep the strong feature extraction for images but leverage a different algorithm suited to the problem (Yosinski, 2014). Therefore, an optimal image recognition algorithm for small datasets should combine multiple convolutional neural layers for the most generalizable feature extraction possible, but use a different classifier as the final step to classify and fuse those results via majority voting. K-nearest neighbor was chosen as the machine learning classifier because it uses the simplest method possible to compare the features found to incoming data with no underlying complexity as it is just finding the closest matches.

Method and Results

For this project, small datasets of 107 dogs and 106 horses were used to train and test a decision fusion algorithm using majority voting for nearest neighbors of convolutional neural layers. The horse photos were mined from unsplash.com and the dog data was obtained from a Kaggle competition to distinguish dogs and cats. To create the multiple convolutional neural layers, it was first necessary to use a library to read in the images easily in CUDA. OpenCV is the industry standard for image read and processing but had compatibility issues with CUDA so it was necessary to use LodePNG. A python script leverages OpenCV to convert all the data into PNG files which are then put in the appropriate directory to be read in my the CUDA file. The CUDA file uses cuDNN to explicitly build out convolutional neural layers that can detect contours via three dimensional kernels around channels based on the two dimensional template kernels:

	{1, 1, 1},		{1/16, 1/8, 1/16}
kernel 1	{1, -8, 1},	kernel 2	{1/8, 1/4, 1/8},
	{1, 1, 1}		{1/16, 1/8, 1/16}

These layers are run discontinuously as the feature detection portion of the machine learning algorithm because it can be compute intensive and take a long time to run. Results of the layers were outputted to the appropriate folders based on the kernel. A graphical representation of the extracted features versus for the first kernel and the original image can be seen in Figure 2 below:

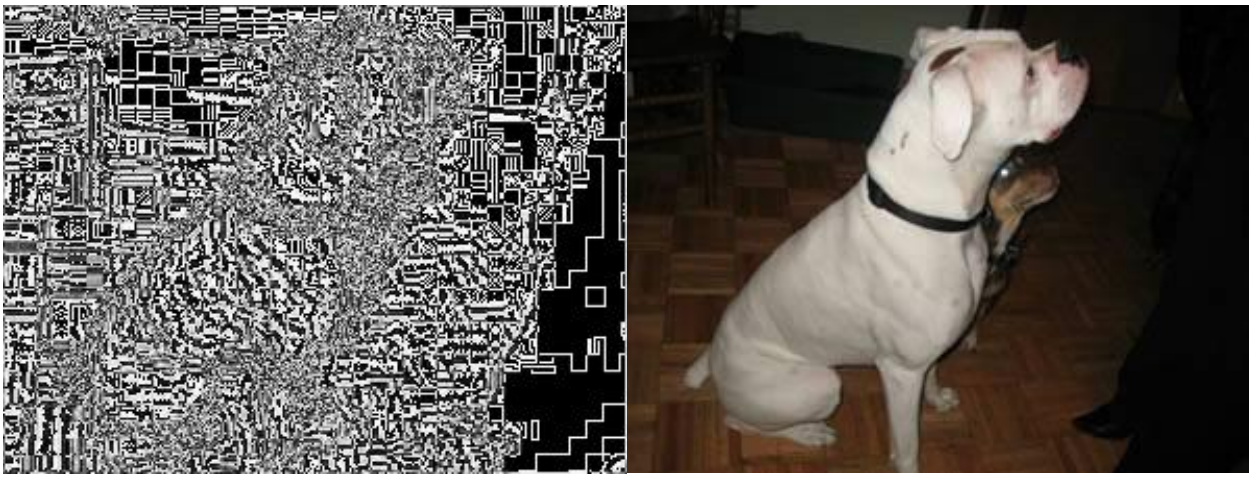

Figure 2: Dog.19 using the first convolutional neural layer and the original dog.

These results are then read into a python script with the raw data and then reduced to a 255x255 signal which is the flattened. These three signals for each image are used for training of three separate nearest neighbor algorithms via scikit-learn with 1/6 of the data left for testing purposes. The number of k used for the nearest neighbor was varied between 1, 5, 10, and 20 and the resulting accuracies were then compared.

K	1	5	10	20
Dog Accuracy	94.74%	94.74%	100.00%	89.47%
Horse Accuracy	55.56%	61.11%	38.389%	38.89%
Total Accuracy	**75.68%**	**78.38%**	**70.27%**	**64.86%**

Discussion

With training sets of less than one hundred for both dogs and horses, the machine learning algorithm was able to obtain 78.38% accuracy when distinguishing between the two when k is set to 5. The algorithm seemed overly sensitive towards dog classification at there was consistently high accuracy when identifying dogs than horses. When k is set to 5, which has the overall highest accuracy, dogs were correctly identified 94.74% of the time.

However, when k is set to 10, no dogs were misclassified. While overall this resulted in lower accuracy because of the little over one third chance of correctly identifying horses, in many cases the k equals 10 scenario may be optimal. In a facial recognition scenario for example, a system would prefer to be overly sensitive and allow a human to select if it is the person or not whether the use case is auto tagging friends on Facebook or identifying criminals based on camera footage. When machine learning is brought to work together with human expertise, there is often one class that is more crucial to being detected and the overall accuracy is not as relevant because of the extra manual verification.

Horses may have had a lower accuracy than dogs because the data scraped from unsplash.com may not be as representative as the pictures of dogs specifically curated for an image recognition challenge by Kaggle. There were more pictures showing large landscapes with horses less of a focus of the image which may have made it more difficult to distinguish between the two. Weighting could have been used on the nearest neighbor classifiers to increase horse sensitivity though this would likely result in an even greater decrease in dog sensitivity. The best mechanism to increase horse classification performance is to increase the training set size or find more representative samples.

Regardless of the weaker results for horses, there was greater than chance classification of horses while maintaining near 100% accuracy for dogs, all with such a small dataset. This shows that generalizable feature detections in images coupled with decision fusion can be used to replicate the human ability to quickly extrapolate angle and lighting independent features for rapid recognition in heterogeneous future encounters. Further work could explore this methodology with larger datasets to see how that affects performance. In addition, the algorithm can be used to train on multiple types of animals as well as alternative classifiers in the decision fusion step. Some models allow for a projected probability of classification which could be used to create weighted voting decision fusion so that more valuable features are further reinforced over others.

References

Bekkerman, R., Bilenko, M., & Langford, J. (Eds.). (2011). *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press.

Brown, L., Catanzaro, B., Seward, C., Brown, L., & Appleyard, J. (2014). *Accelerate Machine Learning with the cuDNN Deep Neural Network Library*. *NVIDIA Developer Blog*. Retrieved 13 May 2018, from https://devblogs.nvidia.com/accelerate-machine-learning-cudnn-deep-neural-network-library/

Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249-256).

Kheradpisheh, S. R., Ghodrati, M., Ganjtabesh, M., & Masquelier, T. (2016). Deep networks can resemble human feed-forward vision in invariant object recognition. *Scientific reports*, *6*, 32672.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Rogova, G. (1994). Combining the results of several neural network classifiers. *Neural networks*, *7*(5), 777-781.

Suen, C. Y., & Lam, L. (2000, June). Multiple classifier combination methodologies for different output levels. In *International workshop on multiple classifier systems* (pp. 52-66). Springer, Berlin, Heidelberg.

What is Machine Learning? | Machine Learning. (2018). *Mlplatform.nl*. Retrieved 13 May 2018, from http://www.mlplatform.nl/what-is-machine-learning/

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. In *Advances in neural information processing systems* (pp. 3320-3328).