Using Convolutional Neural Network for the Tiny ImageNet Challenge

Jason Ting
Stanford University
jmting@stanford.edu

Abstract

In this project we work on creating a model to classify images for the Tiny ImageNet challenge. We use Convolutional Neural Networks trained on GPUs to classify images in the Tiny ImageNet data set to correctly identify the images to the labels as well as possible. We train multiple deep models to classify images on the test data set and explore using ensemble techniques to improve the test set classification accuracy.

1. Introduction

The goal of this project is to build a classifier for the Tiny ImageNet data set to accurately classify images to their label. This data set is a distinct subset of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) data set, which consists of 200 different categories. In general, we expect that similar techniques that work effectively on the ILSVRC data set would also work effectively on the Tiny ImageNet data set, so we plan to train and use similar effective models and apply techniques that were successful on ILSVRC in recent years for the Tiny ImageNet data set.

We approach problem through utilizing GPU as our primary computation device so that we can rapidly iterate and train different neural network designs for the scale and scope of this project.

1.1. Tiny ImageNet Challenge

The ImageNet [1] challenge (ILSVRC) is well-known image classification and localization benchmark for large scale data sets that has been held for six years. The ILSVRC is an image data set organized according to the WordNet hierarchy. The data set consists of 1000 different categories for image classification. The ILSVRC has a total of 1,200,000 labeled images in the training set and 150,000 labeled images in the validation and test set.

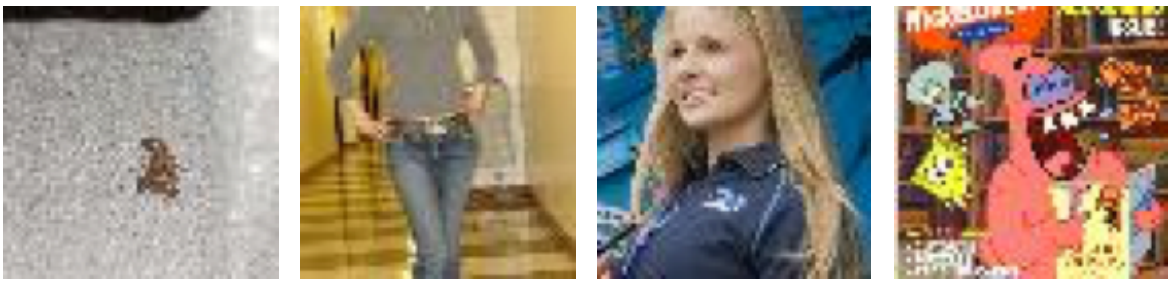
The Tiny ImageNet data set is a distinct subset of the ILSVRC data set with 200 different categories out of the entire 1000 categories from ILSVRC. The images are given in the JPEG format. Each image label has 500 training images (a total of 100,000), 50 validation images (a total of 10,000), and 50 test images (a total of 10,000). The test images are unlabeled, and bounding boxes indicating where the label is located in the image are provided for the training and validation images only, although we do not use this information to perform localized classification. The classifier's accuracy is defined as the percent of the test images which are correctly classified through uploading the labels for all of the test images produced by the classifier on the evaluation server leader board.

1.2. The Data Set

The current best model achieves an error rate of 3.57% [2] using a deep convolutional neural network. The Tiny ImageNet challenge is a simpler version of this problem, so ideally it should be possible to train a model that performs better than the best model from ILSVRC. However, the original pictures from the ImageNet data set are 482x418 pixel with an average object scale of 17.0%. Since the Tiny ImageNet data set pictures are 64x64 pixels, 13.3% pixels are removed from the original images to make the pictures a square, and then these pictures are shrunk by a factor of 6.5. This transformation alongside with how the labels are associated with the images leads to potential problems for training the model, where some example images can be seen in Figure 1.

In Figure 1(a) and Figure1(e), the label is difficult to classify by human inspection in the image since the original image was shrunk and the labeled object is scaled down in the image. In Figure 1(b) and Figure1(f), the labeled object is cropped out of the image. In Figure 1(c) and 1(g) the image label is based on the background scenery, which is different from most other images in the data set where the labels are a object in the foreground. In Figure1(d) and Figure1(h), the images are distinctively different from the other images with their respective label. Because of these problems, the models trained on the Tiny ImageNet data set do not perform as well as the top model from the ImageNet challenge.

(a) Yorkshire Terrier (b) Bow Tie (c) Bathtub (d) Banana



(e) Scorpion (f) Sunglasses (g) Umbrella (h) Comic Book

Figure 1: Some examples of images in the data set that are difficult to classify in the data set and potentially hurt the model. The features that makes these images challenging to classify is the object scaling in (a) and (e), the cropping out of crucial information in (b), (f), the image label based from the background scenery in (c), (g), and images distinctively different from other images with the same label in (d), (h).

2. Background

The ILSVRC has been held for the past six years, producing a variety of different models and techniques to classify images. In 2012, the best model Super-Vision used Convolutional Neural Network with 60 million paramters and introduced dropout to win the ImageNet challenge, which established the basic structure of Convolutional Neural Network. The Visual Geometry Group (VGG) published a paper explaining and analyzing this network [3]. In 2014 GoogLeNet achieved a 6.7% error rate for classifying images [4]. They introduce a multi-scale idea with intuitions gained from the Hebbian principle and used a 22 layer depth convolutional neural network. They acheived this depth through dimension reduction layers. By 2015 the ResNet model achieveda 3.57% error rate. This model reformulated the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions and used a 152 layer depth Convolutional Neural Network [2].

Even though the ResNet achieved the best performance of ILSVRC, the 152 layer Convolutional Neural Network took 2-3 weeks of training on 8 GPU machines, so training a model similar to the ResNet is unfeasible for this project because of resources constraint. Because of this, we plan to utilize the learning techniques mainly from VGG and GoogleLeNet, as well as other models, to develop our own Convolutional Neural Network that is available for the scale of our data and hardware.

3. Implementation

In this section we describe the packages used and the optimization algorithm used for creating and coding the convolutional neural network in details.

3.1. Theano

Theano is a library that allows users to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently [5]. Theano combines aspects of a computer algebra system (CAS) with aspects of an optimizing compiler. It can also generate customized C code for many mathematical operations, which can attain speeds rivaling hand-crafted C implementations for problems involving large amounts of data. Furthermore, Theano has transparent use of a GPU that supports CuDNN and CNMem.

For the this project we use Keras, a minimalist, highly modular neural networks library that that runs on top of Theano [6], using the NIVDIA Tesla C2070 in order to speed up training the convolutional neural networks.

Name	Architecture	Test Acc.
M1	IMG→ ConvReLU(F5-16)→ MaxPool→ ConvReLU(F3-16)→ MaxPool→ ConvReLU(F3-32)→ MaxPool → FCReLU(256) → FC(200)	23.8%
M2	IMG→ ConvReLU(F5-32)→ ConvReLU(F5-32)→ MaxPool→ ConvReLU(F3-64)→ ConvReLU(F3-64)→ MaxPool → FCReLU(256) → FC(200)	28.6%
M3	IMG→ ConvReLU(F5-32)→ ConvReLU(F5-32)→ MaxPool→ ConvReLU(F3-64)→ MaxPool→ ConvReLU(F3-128)→ MaxPool → FCReLU(256) → FC(200)	34.2%
M4	IMG→ ConvReLU(F5-32)→ ConvReLU(F5-32)→ MaxPool→ ConvReLU(F3-64)→ ConvReLU(F3-64)→ MaxPool→ ConvReLU(F3-64)→ ConvReLU(F3-128)→ MaxPool → FCReLU(256) → FC(200)	39.3%

Table 1: The 4 convolutional neural networks and their respective test accuracy. The number of parameters are given for each layer. For example, FC(200) means a fully connected layer with 200 neurons, while ConvReLU(FX-Y) is a conv-ReLU layer Y filters of size X.

3.2. Optimization Algorithm

Training the convolutional neural network is done through minimizing the softmax loss function, denoted by L. The optimization algorithms used for training are based on stochastic gradient descent (SGD). SGD updates the model parameters W through moving in the direction opposite of the gradient is denoted by the following update equation:

$$W \leftarrow W - \eta \nabla_W L$$

where η is the learning rate hyperparamter. Using the momentum update helps the Neural Network models achieve better convergence rates, where the parameter vector builds up velocity in any direction that has consistent gradient. It updates using the momentum variable μ which acts similar to the coefficient of friction. We use the Nesterov Momentum update rule in Theano when training the convolutional neural network.

4. Technical Approach

In this section we describe the neural network designs and techniques for the how we built neural network models for the Tiny ImageNet classification in detail.

4.1. Network Architecture

We trained 4 Convolutional Neural Networks that varies its filter and depth. These Convolutional Neural Networks architectures are described in detail in Table 1. Network M1 is the simplest architecture with three convolutional layers and one fully connected one. Network M2 has 4 convolutional layers, with each layer having an extra convolutional layers than M1 and larger size. The purpose of M1 and M2 is the get an idea of the hyperparamter ranges to use and generally architectures that perform decently on the test data set. Network M3 is the network that we used to combine the aspects of M1 and M2 and generate data for the study of ensemble methods. It is 4 layers deep and is larger than M2. We wanted a model that performs well and is also relatively fast to train in order to perform the ensemble method. Network M4 is the network that we used to achieve the highest test accuracy. This is the deepest network with the highest number of filters. Compared to M3, it has more and larger filters with the pooling layers are in different places.

4.2. Nonlinearities

There are several nonlinear activation functions that can be used used as the output layer of each convolutional neural network layer, such as the sigmoid function and the tanh function. The most commonly used function in modern convolutional neural networks is the rectified linear unit, or ReLU, which computes the function $f(x) = \max(0, x)$. Another activation function is Leaky ReLU, which addresses the problem where a large gradient updates the weights such that the neuron will never activate on any datapoint again. This function computes the equation $f(x) = \mathbb{1}(x < 0)(\alpha x) + \mathbb{1}(x >= 0)(x)$ where α is a small constant.

We experimented with the activation functions through comparing the performance of different activation functions on the same neural network architectures. Leaky ReLU and tanh had negligible impact in the model performance compared to the ReLU activation function, so ReLU is used for all the models for this project.

4.3. Data Augmentation

We used some image preprocessing techniques to improve training the convolutional neural networks. We used mean subtraction, which involves subtracting the mean across every individual feature in the data, and normalization, which normalizes the data dimensions so that they are of approximately the same scale. Moreover another method includes mirroring, which flip images horizontally, for the training images, which helped improve the test accuracy of

Figure 2: Training and Validation Loss of M4 vs. Epoch while training the model

Figure 3: Training and Validation Accuracy of M4 vs. Epoch while training the model

the models.

4.4. Regularization

Because deep convolutional neural networks have a lot of parameters, there are several ways of controlling the capacity of the model to prevent it from overfitting. We used L_2 regularization, which penalizes the squared magnitude of all parameters directly in the objective by adding the term $\lambda||W||_2^2$ to the loss function to each layer, where λ is the hyperparameter that determines the regularization strength and $||W||_2^2$ is the L_2-norm of the weights of the layer. We used λ values between 10^{-5} and 10^{-2} for the various networks.

Another regularization technique we used alongside L_2 regularization in the models is dropout regularization. While training, dropout is implemented by only keeping a neuron active with some probability p or setting it to 0 otherwise. For all the neural network models we used a dropout later with p values between 0.4-0.6 for all the fully connected layer except for the last layer.

5. Implementation

This section discusses the parameters and performance of the deepest and best performing model M4 in greater detail.

5.1. Training

We selected the parameters of training M4 through cross validation centered around standard effective hyperparameters for convolutional neural networks. We trained M4 using a regularization strength of $\lambda = 10^{-3}$, dropout of $p = 0.5$ for each layer for a total of 150 epoch. The learning rate was initialized to 0.01 and multiplied by 0.85 every 10 epoch to decay the learning rate over time. The μ in the momentum was set to 0.9. Using the these hyperparamters

Figure 4: Visualization of the first layer of the convolutional neural network M4. The first-layer weights are very nice and smooth with noticable patterns, indicating nicely converged network.

M4 achieved a test accuracy of 39.3%. The improvement of the model's loss value and accuracy over each epoch are shown in figure 2 and 3 respectively.

5.2. Visualization

Figure 4 shows the visualization of the first layer of M4. The weights are useful to visualize because well-trained networks usually display nice and smooth filters without any noisy patterns. Noisy patterns can indicate that the network that may not have been trained for long enough, or

possibly a very low regularization strength that may have led to overfitting.

5.3. Predicting Image Classes

We can use the trained convolutional neural network to measure the accuracy of each class separately. Since we have 50 images for each class, we can compute the fraction of the classes for each image that are classified correctly by M4. The class with the lowest accuracy is the class "wooden spoon" with an accuracy of 2% and the class with the highest accuracy can is the class "school bus" with an accuracy of 86%. Some examples of the images with the label "school bus" and images with the label "wooden spoon" can be visualized in Figure 5 and Figure 6 respectively.

The model's performance for the classes suggests some important image properties that the model uses to classify the images. The scale of the image is important for accurately predicting the class of the image. In the "school bus" class, the labeled object takes up the majority of the image, whereas the "wooden spoon" class the labeled object consists of a spectrum of different sizes, where the "wooden spoon" is not necessarily in the forefront of the image. Furthermore, the convolutional neural network relies heavily on the color composition of the image. The "school bus" class has images composed of the similar yellow color for most of the images, whereas the "wooden spoon" class has a variety of different colors that make up the images. Moreover, a property that makes it hard for the classifier to predict the right class is the shape of the object. For example in Figure 6 in the "wooden spoon" class, the shapes of the spoon in each picture vary greatly from each other, which makes the images in the class very different from each other, whereas in the "school bus" class in Figure 5 the school bus is usually rectangular.

6. Ensemble Method

This section discusses using ensemble methods, which uses various methods to combine different models in order to improve the performance of the neural networks. As the number of models in the ensemble increases, the performance typically monotonically improves, where the improvements are more dramatic with higher model variety in the ensemble.

6.1. Using Model Ensemble

We form the model ensemble through using the top models discovered during cross-validation. We used cross-validation to determine the best hyperparameters, then pick the top few models to form the ensemble. We used a set different set of hyperparameters using the M3 architecture to produce 6 models for the ensemble method. Because of resource constraint we trained the model for 75 epoch.

Figure 5: Images from the class "school bus." The best convolutional neural network classifies images from this class most accurately. Many the pictures in this class consists of a shade of yellow, the color of the class, and has the labeled object scaled largely in the image.

Figure 6: Images from the class "wooden spoon." The best convolutional neural network classifies images from this class least accurately. Many the pictures in this class consists different colors, are scaled differently in the image, have shapes that vastly differ, and usually consists of other prominent subjects in the pictures.

A simple way to implement an ensemble of models is to average the predicted probabilites for each model in the ensemble. More precisely, suppose we have models k models $m_1, \ldots, m_k$ and we want to combine them into an ensem-

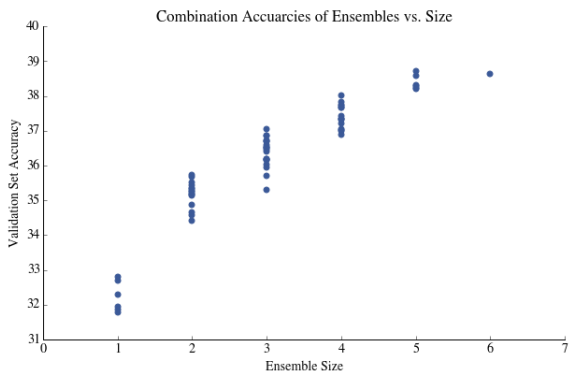


Figure 7: Validation accuracy of the ensemble model using different combination of models.

ble. If $p(x = y_i \mid m_j)$ is the probability that the input x is classified as y_i under model m_j, then the enemble predicts

$$p(x = y_i \mid \{m_1, \ldots, m_k\}) = \frac{1}{k} \sum_{j=1}^{k} p(x = y_i \mid m_j)$$

6.2. Ensemble size vs Performance

Using the 6 variations of M3, we can form many different ensembles of different sizes. For example, if we have n models and we want to form an ensemble of k models, then there are $\binom{n}{k}$ possible ensembles that we can form, where

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$

We can use these different possible ensembles to study the effect of ensemble size on ensemble performance. Figure 7 shows the performance of the validation set accuracy of all possible ensembles. The ensemble model with k = 6 has 39.8% accuracy, which is better than the M4 performance and achieved the highest accuracy..

7. Conclusion

This projects used convolutional neural networks for performing image classification on the Tiny ImageNet data set. We trained and tested the performance of several models M1-M4 which varies in degrees of complexity. We trained M4 as our deepest and best performing model, which achieved an accuracy of 39.3%. Furthermore we used ensemble methods for the model M3 to improve the overall test accuracy to get an accuracy of 39.8%. By the end of the course we were ranked 7th on the course's evaluation scoreboard.

Some ideas for further improvements in the performance include fine tuning different parameters for all the models and try out more models in the ensemble method. Given the time and resources for the project parameter optimization was limited. More importantly potential performance boosts includes using convolutional neural networks with deeper layers similar to the best performing models from the ILSVR, but hardware resources was the biggest factor of not training and utilizing deeper neural network models. As suggested earlier, images that are difficult for humans to label such as the images from Figure 1 can be removed from the data set before training the models. Moreover more types of data augmentation can be implemented such as color jittering and random cropping. The bounding box of the labels indicating where the image is located was not utilized in this project, so recurrent neural network models can potentially be used or creating a filter to help preprocess the image based on the area of the bounding box such as image cropping can potentially be implemented to improve the performance.

References

[1] "Imagenet." http://www.image-net.org/. 1

[2] K. He, X. Zhang, S. Ren, and J. a. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015. 1, 2

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *CoRR*, vol. abs/1405.3531, 2014. 2

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, , and Rabinovich, "Going deeper with convolutions," *IEEE*, 2015. 2

[5] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements." Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012. 2

[6] F. Chollet, "Keras." https://github.com/fchollet/keras, 2015. 2