Tiny ImageNet Visual Recognition Challenge

Ya Le
Department of Statistics
Stanford University
yle@stanford.edu

Xuan Yang
Department of Electrical Engineering
Stanford University
xuany@stanford.edu

Abstract

In this work, we investigate the effect of convolutional network depth, receptive field size, dropout layers, rectified activation unit type and dataset noise on its accuracy in Tiny-ImageNet Challenge settings. In order to make a thorough evaluation of the cause of the peformance improvement, we start with a basic 5 layer model with 5×5 convolutional receptive fields. We keep increasing network depth or reducing receptive field size, and continue applying modern techniques, such as PReLu and dropout, to the model. Our model achieves excellent performance even compared to state-of-the-art results, with 0.444 final error rate on the test set.

1. Introduction

Convolutional neural networks have demonstrated recognition accuracy better than or comparable to humans in several visual recognition tasks, including recognizing traffic signs, faces, and handwritten digits. In particular, an important role in the advance of deep visual recognition architectures has been played by the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [12], which has served as a testbed for a few generations of large-scale image classification systems.

In this project, we work on the Tiny ImageNet Visual Recognition Challenge. This challenge runs similar to the ImageNet Challenge (ILSVRC). The goal of the challenge is to do as well as possible on the Image Classification problem. The performance is measured by the test set error rate, the fraction of test images that are incorrectly classified by the model.

We experiment with various convolutional neural networks, which differ in depth, receptive field size, nonlinearity layer and number of dropout layers. Our results show that deeper networks with smaller receptive fields and larger number of channels usually perform better. Some of our networks achieve excellent performance even compared to state-of-the-art results.

The rest of the paper is organised as follows. We discuss relevant literature in Section 2 and describe our problem and data set in Section 3. The details of our technical approaches are presented in Section 4. In Section 5, we describe our convolutional neural network configurations, implementation details and experiment results. Section 6 concludes the paper.

2. Related Work

Recently there have been tremendous improvements in recognition performance, mainly due to advances in two technical directions: building more powerful models, and designing effective strategies against overfitting.

Neural networks are becoming more capable of fitting training data. This is majorly due to the development of the following areas: sophisticated layer designs [17, 4]; new nonlinear activations [11, 10, 20, 16, 3, 5]; and increased complexity, including increased depth [17, 14], enlarged width [19, 13], and the use of smaller strides [19, 13, 1, 14].

Since networks are more complex, it's necessary to have better strategies for model generalization. This is achieved by large-scale data [2, 12], aggressive data augmentation [9, 7, 14, 17], and effective regularization techniques [6, 15, 3, 18].

3. Problem Statement

The goal of this challenge is to estimate the content of photographs for the purpose of retrieval and automatic annotation using a subset of the large hand-labeled ImageNet dataset (10,000,000 labeled images depicting 10,000+ object categories) as training. Test images will be presented with no initial annotation – no segmentation or labels – and algorithms will have to produce labelings specifying to which category the images belong. New test images are collected and labeled especially for the original ImageNet Challenge and are not part of the previously published ImageNet dataset. The general goal is to identify the main objects present in images and to make image classification.

3.1. Data

The TinyImageNet dataset is a subset of the ILSVRC-2012 classification dataset. It consists of 200 object classes, and for each object class it provides 500 training images, 50 validation images, and 50 test images. All images have been downsampled to $64 \times 64 \times 3$ pixels. The training and validation sets are released with images and annotations, including both class labels and bounding boxes. But the main goal of this project is to predict the class label of each image without localizing the objects. The test set is released without labels or bounding boxes.

3.2. Evaluation

For each image, algorithms will produce one label which has largest estimated probability. The quality of a labeling will be evaluated based on the ground truth label for the image. The error of the algorithm for that image would be 1 if the predicted label doesn't match the ground truth label, and 0 otherwise. The overall error score for an algorithm is the average error over all test images.

4. Technical Approach

4.1. Model Architecture

In the last few years, mainly due to the advances of deep learning, more concretely convolutional networks, the quality of image recognition and object detection has been progressing at a dramatic pace. Therefore we will focus on a deep neural network architecture for this project. We will experiment with the following architectures with several dropout layers inserted in the network and use the validation set to tune the hyperparameters, including filter size, number of filters, network depth, learning rate and regularization strength, etc.

1. [conv-relu-pool]×N - conv - relu - [affine]×M - [softmax or SVM]

2. [conv-relu-pool]×N - [affine]×M - [softmax or SVM]

3. [conv-relu-conv-relu-pool]×N - [affine]×M - [softmax or SVM]

4.2. Reduce Overfitting

Since there are only 500 training images for each class, which is a much smaller training set compared to the ILSVRC-2012 classification dataset, it's insufficient to learn a deep neural network without considerable overfitting. We consider the following two primary ways to reduce overfitting.

4.2.1 Data Augmentation

The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformations. We will employ four distinct forms of data augmentation at training time, all of which allow transformed images to be produced from the original images with very little computation, so the transformed images do not need to be stored on disk. At training phase, the network random choose from ten transformed images of the given test image, which are the four corner crops and the center crop, as well as their horizontal reflections. At test time, the network makes a prediction based on the center crop.

1. Random crops. The images of Tiny ImageNet Challenge are $64 \times 64 \times 3$. Instead of feeding our training images directly to the convnet, at training time we will randomly crop each training image to $56 \times 56 \times 3$ and train our network on these extracted crops.

2. Random flips. We will randomly flip half of the training images horizontally.

4.2.2 Dropout

The recently-introduced technique "dropout" [15] is a very efficient version of model combination that only costs about a factor of two during training. While training, dropout is implemented by only keeping a neuron active with some probability p, or setting it to zero otherwise. At test time, we will use all the neurons but multiply their outputs by p.

Since the dropped-out neurons do not contribute to the forward pass and do not participate in back-propagation, the neural network samples a different architecture every time an input is presented, while all these architectures share weights. In this way, a neuron cannot rely on the presence of particular other neurons. Therefore, the droput technique reduces complex co-adaptations of neurons, and is forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

For this project, we will use the inverted dropout, which performs the scaling at train time, leaving the forward pass at test time untouched, with $p = 0.5$.

4.3. Other Techniques

4.3.1 Model Ensemble

One reliable approach to improving the performance of Neural Networks is to train multiple independent models, and at test time average their predictions. Usually the improvements are more dramatic with higher model variety in the ensemble. We will use the following two ways to form an ensemble. Using which one will depend on the complexity of the network.

1. Top models discovered during cross-validation. We will use cross-validation to determine the best hyper-parameters, then pick the top few models to form the ensemble. This can be easier to perform since it doesn't require additional retraining of models after cross-validation.

2. Different checkpoints of a single model. For complex networks which are expensive to train, we will take different checkpoints of a single network over time and use those to form an ensemble.

4.3.2 Parametric Rectifiers

We will experiment with the network that replaces the parameter-free ReLU activation by a learned parametric activation, called Parametric Rectified Linear Unit (PReLU) [5]. This activation function adaptively learns the parameters of the rectifiers, and improves accuracy at negligible extra computational cost.

Formally, the activation function is defined as $f(y_i) = max(0, y_i) + a_i \min(0, y_i)$, where y_i is the input of the nonlinear activation f on the ith channel, and a_i is a coefficient controlling the slope of the negative part, which allows the nonlinear activation to vary on different channels. If $a_i = 0$, it becomes ReLU; if a_i is a small and fixed value, PReLU becomes the Leaky ReLU (LReLU) in [10]. The number of extra parameters that PReLu introduces is equal to the total number of channels, which is negligible when considering the total number of weights.

5. Experiment

5.1. Model Configuration

We experiment with nine different models. The configurations of these convolutional neural networks and their number of parameters are outlined in Table 1, one per column. Particularly, Model 9 is inspired by VGG model [14] and modified to fit our dataset. These model configurations differ in the following ways:

1. Depth: The depth ranges from 5 layers in Model 1 to 16 layers in Model 9.

2. Receptive field size: Most conv layers in Model 1-4 have 5×5 receptive fields, while all conv layers in Model 5-9 use 3×3 receptive fields.

3. Number of kernels: The number of kernels ranges from 128 in Model 1 to 512 in Model 9.

4. Dropout. As the neural net gets deeper, we add more dropout layers.

5. ReLU layer: Each convoluational layer in Model 1-4 and Model 9 is followed by a ReLU layer, while convolutional layer in Model 5 is followed by a PReLU layer, and each convolutional layer in Model 6-8 is followed by a LReLu layer.

5.2. Implementation Details

We implemented our models using the publicly available C++ Caffe toolbox [8] on GPUs provided Amazon Web Services. We modified the caffe code to add PLeRu layers (and LReLu is just a special case of PReLU). Here we focus on the implementation details of Model 8 and Model 9 as these two models have the best performances and they are evaluated on the test set.

5.2.1 Training Methodology

The training is carried out by minimizing the regularized multinomial logistic regression loss using mini-batch gradient descent with momentum. All the training images are randomly cropped to $56 \times 56 \times 3$ images.

The original Imagenet Challenge has input dataset as 224x224, but the Tiny Imagenet Challenge only has input size 64x64. With cropping the input image, some objects are located in the corner. During data augmentation, with random crop, the object will be even further away from the center of our view, or even outside the crop. This makes the image become an invalide training data, and we need to eliminate them from the training dataset. With the provided object bounding box, this becomes feasible. We check the object location of each image, if less than 1/4 of the object is a random crop, we will remove the image from the dataset. Specifically, Model 1-4 are trained using all training images, while Model 5-9 are trained using screened images.

For all of our models, the batch size was set to 200, momentum to 0.9 and dropout ratio to 0.5. The learning rate was initially set to 0.02 for Model 8 and 0.01 for Model 9, and then decreased by a factor of 0.1 and 10 respectively when the validation set accuracy stopped improving. The weight decay (the L2 penalty multiplier) was set to 1×10^{-4} for Model 8 and 5×10^{-4} for Model 9.

For deep networks, it's important to have good initializations due to the non-convexity of the learning objective. We adopted the training strategy in [14], where we began with training first few convolutional layers and the last three fully connected layers so that the network is shallow enough to be trained with random initialisation. Then we initialised the first few convolutional layers and the last three fully connected layers with the weights derived from previous training, and initialised the intermediate layers randomly. For random initialisation, we use the caffe xavier algorithm to fill weights, which automatically determines the scale of

Table 1. **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (Model 1) to the right (Model 9), as more layers are added. The convolutional layer parameters are denoted as conv (receptive field size)-number of channels. The ReLU activation function is not shown for brevity.

Input Size	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9
56	conv5-64	conv5-64	conv5-64	conv5-64	conv3-64	conv3-64	conv3-128	conv3-128	conv3-64
		conv3-64	conv3-64	conv5-64	conv3-64	conv3-64	conv3-128	conv3-128	conv3-64
	maxpool								
28	conv5-64	conv5-128	conv5-128	conv5-128	conv3-128	conv3-128	conv3-256	conv3-256	conv3-128
		conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-256	conv3-256	conv3-128
								dropout	
	maxpool								
14	conv5-128	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-512	conv3-512	conv3-256
			conv3-256	conv3-256	conv3-256	conv3-256	conv3-512	conv3-512	conv3-256
									conv3-256
				dropout		dropout	dropout	dropout	
	maxpool								
7									six conv3-512 layers
FC1	256	1024	1024	1024	1024	1024	2048	2048	4096
					dropout				
FC2							2048	2048	4096
							dropout	dropout	dropout
FC3					200				
					softmax				
depth (conv+fc)	5	7	8	8	8	8	9	9	16
parameters (M)	1.97	13.74	14.33	14.33	14.20	14.20	60.56	60.56	138

initialization based on the number of input and output neurons, and we simply initialize the bias as 0.

5.2.2 Testing Methodology

At test time, the test image is cropped from center to size $56 \times 56 \times 3$ without any flipping. The trained convolutional neural network is then applied to the cropped image to obtain the soft-max class posteriors for the image.

5.3. Results

We perform the experiments on the 200-class Tiny-ImageNet. The results are measured by top-1 error rate. We only use the provided data for training. All the results are evaluated on the validation set, except for the final results, which is evaluated on test set. The validation accuracy is listed in Table 2, and the final accuracy of the ensemble model on the test set is 0.556.

5.3.1 Comparison between Dropouts

In Table 2, we compare the effects of adding dropout layers. We used one dropout layer in Model 3, but two dropout layers in Model 4. We can see that there is more than 1% top-1 accuracy improvement. The improvement is mainly because the dropout layers can help reduce overfitting. The difference bewteen training loss and testing loss is smaller for Model 4 than Model 3, which confirms the regularization effect of dropout.

Table 2. **Convolutional neural network performance on validation set.**

Model Name	Depth	Nonlinear	Screen	Val Accuracy (%)
Model 1	5	ReLu	N	37.27
Model 2	7	ReLu	N	28.20
Model 3	8	ReLu	N	38.29
Model 4	8	ReLu	N	39.55
Model 5	8	PReLu	Y	29.00
Model 6	8	LReLu	Y	40.19
Model 7	9	LReLu	Y	42.07
Model 8	9	LReLu	Y	46.45
Model 9	16	ReLu	Y	59.50
Ensemble	Model 8+Model 9		Y	61.23

5.3.2 Comparison between ReLU, LReLU and PReLU

We implemented Parametric ReLU in caffe and used it in model 5, whose network architecture is the same as that of model 4 except that model 4 uses ReLU. Results in Table 2 show that parametric ReLU does not help to improve the accuracy. The reason could be that PReLu introduces more parameters in the model, resulting in an even worse overfitting.

Since PReLU does not work well in our case, we replace ReLU with LReLU and set the slope to 0.01. Comparing Model 6 with Model 3, or Model 7 with Model 4, we can see there is around 2% accuracy improvement. But this improvement may also result from smaller receptive fields or training data screening.

5.3.3 Comparison between Screening Dataset Results

When the object only has less than 1/4 part in a random crop, we treat this image as noise in the dataset. With filtering out this kind of images, we get a slightly performance improvement. The improvement is smaller than we expected mainly due to two reasons. One is we only removed 1535 images whose object is at the corner. And statistics show that beside those invalide images we still have 1891 images whose object area is smaller than 8×8. In this case, it will also be difficult for human to see, but we have not filter those images out. The other reason is that many images, whose object is at corner or small, are in the same class such as basketball. If we remove all of them from the dataset, it will also drop the classification peformance for that class. Due to this trade-off, we did not remove the images that has small objects. And we see only a small accuracy increase.

5.3.4 Comparision between Single-model Results

Next we compare single-model results. We start with a basic 5 layer model, and track three directions: increasing kernel numbers, reducing kernel window and increasing network depth. Comparing model 1 with model 3 or model 4, the only difference is we increase the network depth, and we see around 2% accuracy improvement. However, increasing network depth does not always make peformence between, for example, model 2 get higher error rate.

With more smaller kenel window, we see around 2% accuracy improvement by comparing model 3 with model 6 or model 4 with model 7. But this improvement may also comes with LReLU and screening dataset.

With more kernels and adding another fully-connected layer, we see more than 4% accuracy improvement if we compare model 7 with model 8. However, it needs much larger number of parameters and takes longer to train.

5.3.5 Comparision between Multi-model Results

We ensembled the Model 8 and Model 9 in Table 2. For the time being we have only one model for each architectrue, except architecture 3, we have two models based on different regularization. Since the last few models such as model 8 are much better than the smaller network such as model 1, the smaller network does not contribute to the ensembled result. We tried with ensembling all the models, and get validation accuracy as 0.54, but with combining Model 8 and Model 9, we get 0.61 validation accuracy. This indicates that combining few stronger models is better than combining more poor models. The multi-model accuracy on the test set is 0.556.

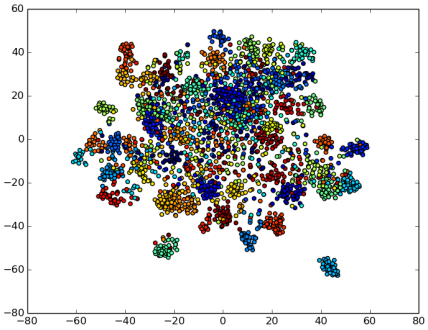

Figure 1. t-SNE graph for 50 classes of validation set using model 9.

5.4. Visualization

We visualize the classification of our model 9 by extracting the feature output of the second fullu-connected layer and plot in t-SNE graph. In t-SNE, the points with the same color belongs to the same class. If a certain cluster is further away from other points in the graph, the network is more certain about this group of points belong to a certain class. From Figure 1, we can see, most points are clustered well, except for a few points in the center. This implies we have a decent classification accuracy of model 9.

5.5. Error Analysis

While our results get close to state-of-the-art convolutional neural network performance, we still get some misclassifications shown in Figure 2. Some missclassification is because some objects have similar shapes, for example, in the first row of Figure 2, the first image (val_1153) shows a vestment, but has been recoginized as alp, and the second image(val_4311) is a bucket or pail, but has been classified as frying pan. Since convolutional neural networks are very sensitive to texture rather than shape, we are not surprised to see this kind of miss classification. There are some other miss classification that are resulted in image down-sampling or crop shown in the second row of Figure 2, such as the first image(val_7570) in that row is moving van, but has been recoginazed as pop bottle, and the second image(val_8160) is lawn mower, but has been predicted as alp, and we found even human feel hard for this kind of images.

GT: vestment
alp

GT: bucket/pail
frying pan

GT: moving van
pop bottle/soda bottle

GT: lawn mower
alp

Figure 2. Example validation images incorrectly classified by model 9. For each image, the ground-truth label and the top-1 label predicted by our method are listed.

6. Conclusion

Our results get close to the state-of-the-art convolutional neural network peformance on ImageNet Challenge. The down-sampling of the input images make the network more difficult to classify correctly due to the information loss, however, less classes of the dataset compensate that. We also found that deeper network with more kernels generally improve the accuracy. Parametric ReLU is not always better than ReLU or LReLU if not well-initilized or no regularization introduced. Last, screening dataset by the object location can also help improve peformance.

References

[1] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[3] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *arXiv preprint arXiv:1406.4729*, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[7] A. G. Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013.

[8] 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.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

[11] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575*, 2014.

[13] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[16] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber. Compete to compute. In *Advances in Neural Information Processing Systems*, pages 2310–2318, 2013.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[18] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[19] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[20] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, et al. On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3517–3521. IEEE, 2013.