Wide Residual Network for the Tiny ImageNet Challenge

Letícia Cordeiro
Stanford University
lrsc@stanford.edu

Abstract

The Tiny ImageNet challenge is a small scale version of the full ImageNet classification challenge. We investigate how a wide residual network performs on this dataset and compare it with a deeper but narrower residual network. In addition, with the objective of obtaining a low test error rate, we fine-tune a pretrained 18-layer residual model. The pretrained model performed well and achieved 0.324 test error and 72.8 accuracy on the validation set.

1. Introduction

The classification task in computer vision consists of assigning a particular category to an image, according to the object that it contains. It has seen significant improvements in the last decade with the development of convolutional neural networks. The role of CNNs in the advance of image classification becomes evident when we look at the results history of the classification task in the ImageNet challenge. The ImageNet challenge is a computer vision competition that occurs every year and includes the tasks of image classification, localization and detection. The ImageNet database is a very large image set with 1000 object categories.

We present here our approach to the challenge with two goals. The first goal is to investigate whether a wide residual network is appropriate for this task and how well it performs when compared to deeper residual networks. The second goal is to obtain the lowest possible error on a set of test images. For the second goal we used a pretrained ResNet with 18 layers. The choice of the ResNet, as opposed to a wide ResNet for this part was the availability of the pretrained model. Training the network with random weight initialization, not taking advantage of transfer learning, would result in a much less accurate classifier.

2. Tiny ImageNet Challenge

The Tiny ImageNet database is a small subset of the large ImageNet dataset. It consists of 100000 training images separated in 200 different classes, as opposed to more than 1 million training images from 1000 classes on the complete ImageNet set. The Tiny ImageNet challenge is a classification task with the goal of achieving the lowest possible error on a test set. On previous years of this challenge the minimum test error rate achieved was around 20%.

Figure 1 – Examples of images and classes on Tiny ImageNet

3. Residual Networks

Residual Networks [2] were designed to make it easier to train very deep networks, based on the idea that adding layers is necessary to increase classification accuracy. Training very deep networks can be difficult because accuracy tends to saturate and then quickly degrade. Residual layers make the network easier to optimize. The logic behind it is that if more layers are stacked into a model, it should not perform worse. It could be the case that

the original model with less layers achieves the best possible accuracy. In that case, adding more layers should not change the results, keeping the good performance achieved with less layers. What that means is that the additional layers would work as an identity map. Residual layers are shortcuts in the network that implement those identity maps and facilitates such identity maps to be trained by learning very small valued weights for the extra layers.

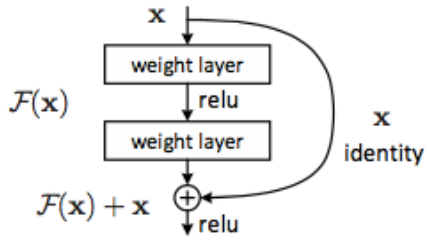

Figure 2 – Building block of a residual network [2]

A residual network model with 152 layers achieved excellent performance and won first place on the ILSVRC 2015 classification task. ResNet versions with less layers were also successful on the Tiny ImageNet challenge on the previous year.

4. Wide Residual Networks

Although it is known that very deep networks can improve accuracy of a model, it is argued in [3] that it takes too many extra layers to be rewarded with just a small improvement in accuracy. Experiments with wider, rather than deeper, networks have indicated that similar or better results than those obtained with very deep neural networks can be achieved with shallower networks with increased width. Increasing the width of a network means to increase the number of filters on the convolutional layers. We investigate the performance of a wide ResNet model on the Tiny ImageNet challenge and how it compares to a deep and narrower model.

5. Technical Approach

6.1 Deep vs. wide

In order to compare the performance of a residual network and a wide residual network, we used the architecture of ResNet18 and ResNet34 as a starting point. The ResNet18 layers were doubled in width to represent a wide Residual Network. In addition, some modifications were necessary to make the models compatible with the Tiny ImageNet dataset, as described on the following paragraphs. Pytorch framework was used for the implementation. Both models were optimized using stochastic gradient descent with 0.9 momentum.

Fully connected layer

The fully connected layer was modified to perform classification on 200 classes, instead of 1000 classes.

First convolutional layer and max pooling layer

The dimensions of the images on the Tiny ImageNet dataset are 64x64 pixels, as opposed to the 256 x 256 pixel images on the full ImageNet set. Therefore, following the steps in [1], we replace the first convolutional layer (conv1) that originally consists of 7x7 filters with stride 2 and padding 3, by 3x3 filters with stride 1 and padding 1 and remove the max pooling layer. The substitutions will keep the resulting image size of 56x56 pixels.

Data Augmentation

Tiny ImageNet contains 500 training examples per class. Compared to other datasets like ImageNet or CIFAR-10, it is a relatively small amount of data. In order to artificially increase the amount of data and avoid overfitting, we rely on data augmentation. Data augmentation was achieved by adding randomly cropped training images and horizontally flipped versions of those images. Overfitting was also avoided by setting the weight decay to an appropriate value.

Number of parameters

When comparing the results of the two models it is important to make sure that the number of parameters to be learned on each one is of the same magnitude. It would not be of interest to achieve better accuracy with a shallower network if its complexity was much higher than the deeper model. Training time is a major constrain on this challenge and a model that can be trained faster would be certainly preferred. The ResNet34 model, after applying the modifications previously mentioned contains 21379572 parameters. The wide version of the ResNet18 contains 44857672 parameters, around twice as the first model. In order to reduce the number of the parameters to learn on the wide network, two convolutional layers – from the first and last blocks – were removed. That resulted in a total of 25683784 parameters, which is comparable to the deeper model.

Learning rate

The learning rate was reduced when the training loss stopped decreasing. More specifically, there were two reductions by 10% during the 20 epochs of training for each model.

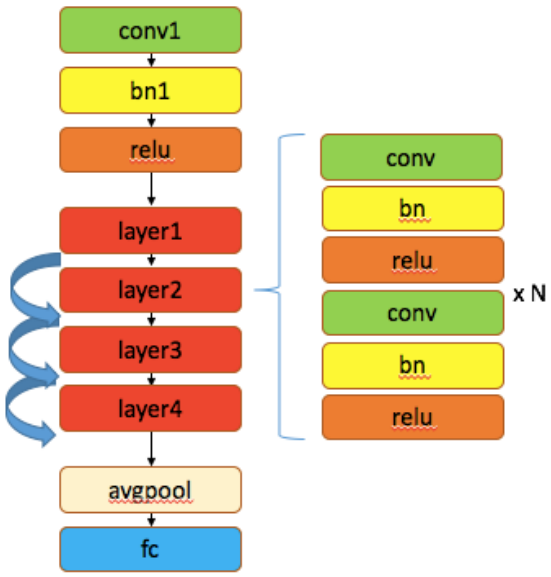

Figure 3 - Structure of residual models

6.2 Minimizing test error

For the second part of this project, the goal was to achieve the lowest possible error rate on the test set. Transfer learning was shown to work well on the Tiny ImageNet dataset, resulting in better classifiers than the ones trained from scratch. Because there were no pretrained wide residual networks available with less than 50 layers, we decided to use a pretrained ResNet18 for the challenge. The same layer substitution and data augmentation described previously were applied to the pretrained model.

Pretrained model

ResNet18 was pretrained on the full ImageNet database. Fine tuning of the parameters for the reduced dataset was performed as follows. For the first few epochs, only the modified layers (first convolutional layer and last fully connected layer) were trained and all the other weights were kept unchanged. That set the training of the weights on those layers on a good direction. After those weights were trained for a while, all parameters were allowed to be modified. This strategy allowed for fine-tuning of all weights of the model, without diverging too much from the optimized pretrained configuration.

6. Results

For the following discussion, we will refer to the models as model I, model II and model III such that:

- model I: modified ResNet34, trained from scratch
- model II: wide residual network based on modified ResNet18, trained from scratch
- model III: transfer learning from pretrained ResNet18.

For models I and II, training and validation accuracy obtained are shown on Table 1. Results show that there is not a significant advantage to use a wide, rather than deep, network on this case. Test errors are also similar. The decrease on training loss by epoch is presented on Figure 4. The expectation was to achieve better results with the wider model, but that was not the case. That can be an indication that only for very deep residual networks each added layer increases accuracy by only a small percentage and it becomes more advantageous to increase the number of filters per convolutional layer. Another possible reason for this result is that, due to time constraints, model hyperparameters were not fully optimized and that may have ended up giving some advantage to the deeper network.

Table 1 – Results for models I and II

	training accuracy	validation accuracy	Test error
model I	59.9%	53.1%	52.5%
model II (wide)	64.3%	55.4%	51.2%

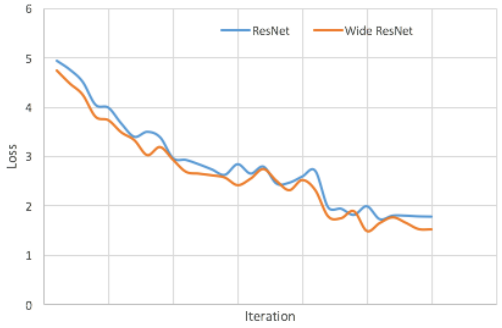

Figure 4 – Training loss for models I and II

Model III performed well and obtained 0.324 test error. Accuracy on validation set was 72.8%. It was trained for 12 epochs and the training loss history is shown on Figure 5.

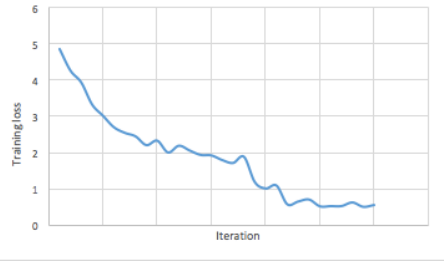

Figure 5 – Training loss for models III

7. References

[1] H. Kim, Residual Networks for Tiny ImageNet. CS231n, 2016

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

[3] S. Zagoruyko, N. Komodakis, Wide Residual Networks. arXiv:1605.07146