# Exploring Methods on Tiny ImageNet Problem

Huseyin Atahan Inan
Stanford University
hinan1@stanford.edu

## Abstract

*In this project, we are interested in building a neural network architecture for the image classification task on the Tiny ImageNet dataset. We first modified the last layer (fully-connected layer) of the well-known ResNet architecture to obtain a good performance as a baseline. We further tried incorporating LSTM's on the last two convolutional layers of the ResNet architecture. We explored different such architectures involving LSTM's and compared the results of these networks.*

## 1. Introduction

Tiny ImageNet Challenge is very similar to the well-known ImageNet Challenge (ILSVRC). The goal is to achieve the best possible performance for the Image Classification problem. Tiny ImageNet Challenge is a subset of the ImageNet Challenge where it contains 200 classes instead of 1000 classes. Each class has 500 training images, 50 validation images, and 50 test images. The final network is given a test image and should output a class prediction out of 200 classes.

We approach the problem first by doing the literature review and understanding the approaches introduced for the ImageNet Challenge. Since the problem and the dataset is very similar to the ImageNet Challenge (ILSVRC), it is very natural to consider that these methods will perform very good in the Tiny ImageNet Challenge as well. Given that the modern ConvNets take 2-3 weeks to train across multiple GPUs on ImageNet, it proves to be better to utilize the pretrained models and try to improve upon them by coming up with novel ideas. For this purpose, we first read the Transfer Learning section from the lecture notes [1] since it provides many practical and important tips.

We next focused on the ResNet model [2] which is built on very interesting and intuitive idea and it is the winner of the 2015 ImageNet Challenge. We modified the last layer, i.e., the fully-connected layer by making it output to 200 class scores instead of 1000 class scores and trained this modified network to get a baseline performance.

We next considered applying LSTM's to the final convolutional layers of ResNet to see whether we can get better performance by incorporating LSTM's. We tried different combinations and compared the performance results among each other and also with the baseline. Unfortunately, we were unable to beat the baseline performance in the end.

## 2. Related Work

One can say that the success of the deep convolutional networks started with the AlexNet introduced in [3] that won the 2012 ImageNet Challenge (ILSVRC) by a huge margin (achieving a top 5 test error rate of 15.4% compared to the next best entry achieving an error of 26.2%). The layout introduced in [3] consists of 5 conv layers along with max-pooling layers, dropout layers, and 3 fully connected layers.

In 2013, the winner of the competition was ZF Net introduced in [6] achieving an error rate of 11.2%. It has a similar structure to the AlexNet but one main and important difference is that they use $7 \times 7$ sized filters instead of $11 \times 11$ in the first layer to retain pixel information in the input. ZF Net paper further provides visualization approach that helps understanding the inner workings of the network.

VGG Net was introduced in [4] in the year of 2014 that has become popular due to having a simple yet deep and powerful network with small sized filters ($3 \times 3$). This allowed the network to be more efficient in terms of the number of parameters.

The winner of ILSVRC 2014 was GoogLeNet which is a 22 layer CNN that achieved an error rate of 6.7%. The paper [5] introduces the idea of an Inception module that allows to perform convolution and pooling operations at the same time (in parallel) by using $1 \times 1$ convolutions which help reducing the dimensionality of the input. This network also does not use any fully connected layers which saves a huge number of parameters. They obtain the final output using an average pool instead.

The last network that we mention is the Microsoft ResNet introduced in [2] that won ILSVRC 2015 with an error rate of 3.6%. It goes as deep as 152 layer network architecture. It achieves a good performance with such deep

network due to the Residual Block idea that basically adds the input itself to the output of conv-relu-conv layers. Here the network only needs to compute the term that one has to add to the input itself and the claim is that it is easier to optimize the residual mapping. Note that it will be much easier for the network to learn nothing which will result in an identity mapping, hence, worst case the performance will stay the same.

## 3. Approach

In this section, we provide the approaches taken for the image classification problem aforementioned. We start with taking the ResNet model and modifying the last layer, i.e., the fully-connected layer such that it provides scores for 200 classes instead of 1000 classes. Figure 1 depicts the modified network.

We next incorporate LSMT's to this network as follows. We consider the output of last two layers (conv4 and conv5) as sequences with respect to their depths and we separate them into groups of length 64. Note that the depth for conv4 is 256 which will provide 4 such sequence and the depth for conv5 is 512 which will provide 8 such sequence. Therefore, in total, we will have 12 sequences of length 64 and there will be one LSTM corresponding to each sequence, hence, there will be in total of 12 LSTM's. We keep the hidden size same as the input size, therefore, the LSTM's corresponding to sequences coming from conv4 has hidden size $14 * 14 = 196$ and the LSTM's corresponding to sequences coming from conv5 has hidden size $7 * 7 = 49$. In the end, we concatenate all the last hidden vectors of the LSTM's which will provide us a vector of size $49*8+196*4 = 1176$ and we use a fully-connected layer to output a vector of size 200 corresponding to the scores of the classes. Figure 2 depicts the corresponding network.

## 4. Experiment

In this section, we will explain the experiments we conducted and provide the results. We start with the discussion of the dataset we have used.

Tiny ImageNet dataset contains 200 classes where each class has 500 training images, 50 validation images, and 50 test images. We train the network using the training images while observing the performance on validation set. The test images do not have labels, therefore, the final predictions are obtained using test images and sent to the online server for the performance on the test set. We evaluate our results based on the accuracy, i.e., the fraction of correct labels corresponding to the images.

The baseline network where we modified the fully-connected layer of the ResNet provided us with the test error of 0.338. We note that we did not spend much time training this network since we wanted to quickly get a good baseline accuracy and move on to trying our own ideas, therefore, one can get better performance by spending more time on fine-tuning and choosing learning rate carefully etc.

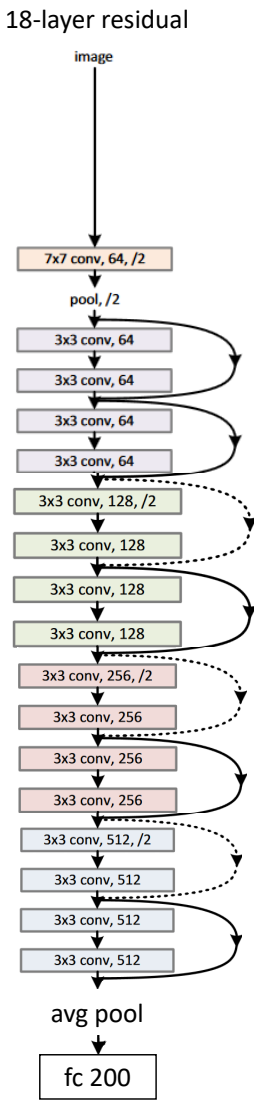


Figure 1. The baseline network. The image is modified from Figure 3 of [2].

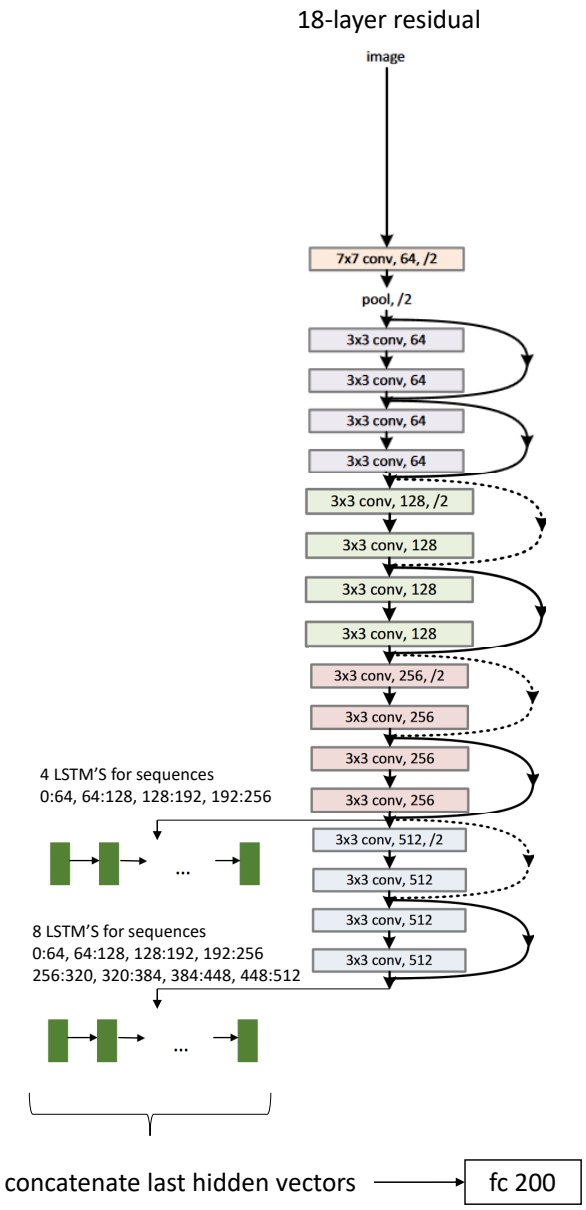


Figure 2. Incorporating LSTM's. The image is modified from Figure 3 of [2].

We next continue with the ResNet model where we incorporate LSTM's as explained in the previous section. We first kept all the convolutional layers constant while training the LSTM's and the final fully-connected layer and then when we reached at a flat region in terms of accuracy, we trained the whole network. One can observe in the plots that we obtained a jump in the performance when we went to training the whole network. Figure 3-6 shows the loss and accuracy for training set and val set. The final accuracy for the validation set is 0.6659. With this network, we achieved an error rate of 0.386 on the test set which is unfortunately worse than the baseline performance. We also tried adding LSTM's to the earlier layers or using different sequence lengths, for instance, 128 instead of 64 but they gave similar performances.

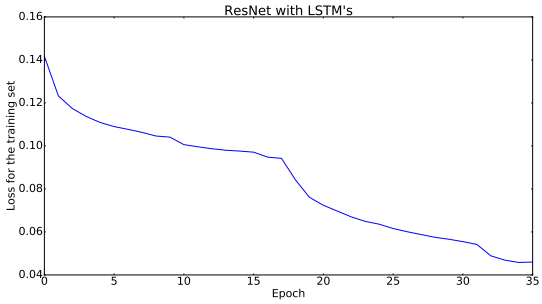


Figure 3. Results for the ResNet model with LSTM's

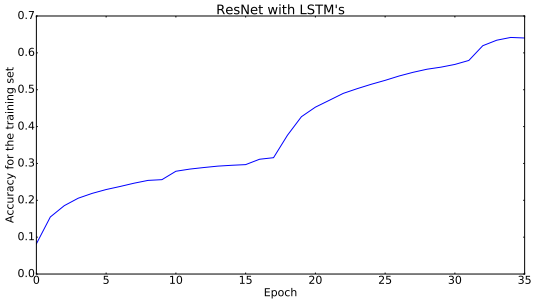


Figure 4. Results for the ResNet model with LSTM's

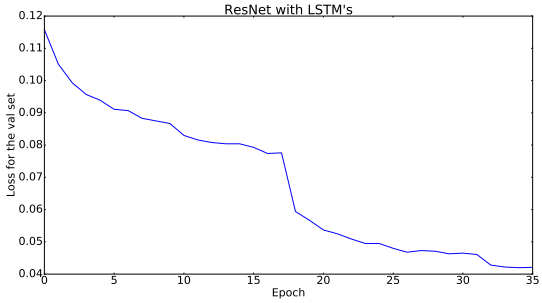


Figure 5. Results for the ResNet model with LSTM's

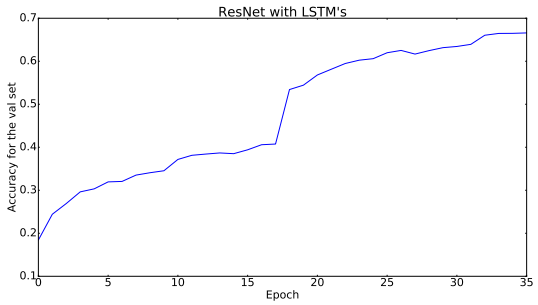


Figure 6. Results for the ResNet model with LSTM's

We finally took the ensemble of this network and the baseline where we took the average of the scores produced by these networks and that gave us an error rate of 0.329 in the test set.

## 5. Conclusion

In this project, we were interested in the image classification task on the Tiny ImageNet dataset. We learned how to take a pretrained model and modify it and fine-tune to obtain good performance for the problem that we had. We also experimented incorporating LSTM's on top of ResNet model to obtain better performance. Unfortunately, we were unable to beat the baseline performance, however, we believe that one can come up with interesting techniques that could potentially provide better performance.

## References

[1] Transfer learning. *http://cs231n.github.io/transfer-learning/*.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *ArXiv e-prints*, Dec. 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[6] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.