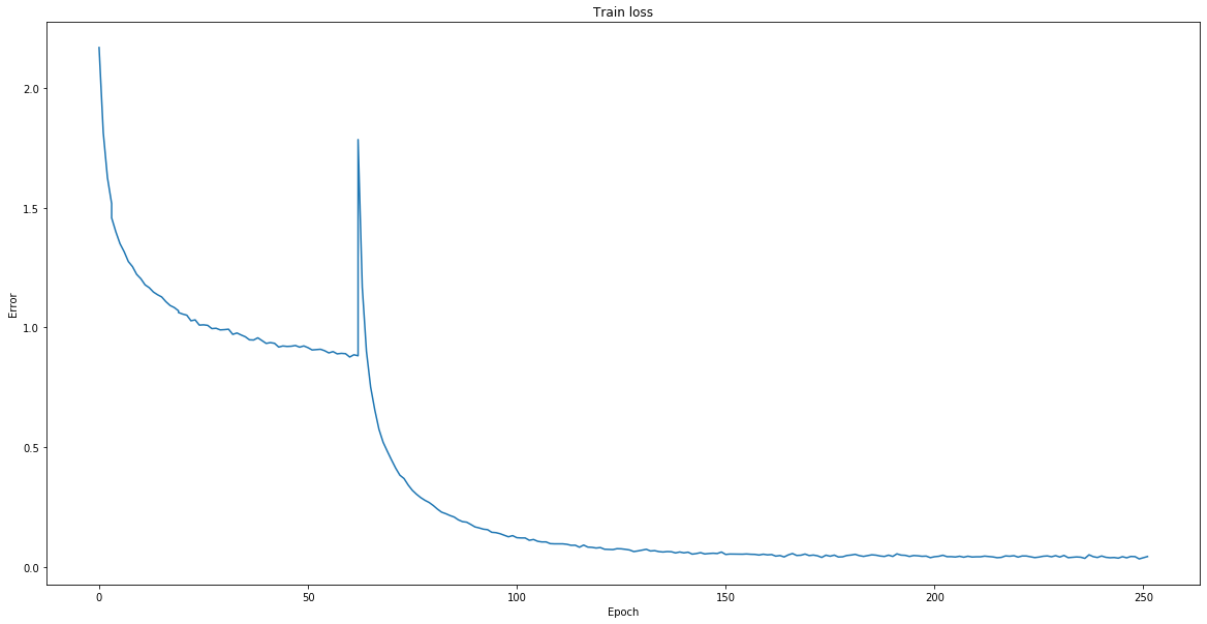
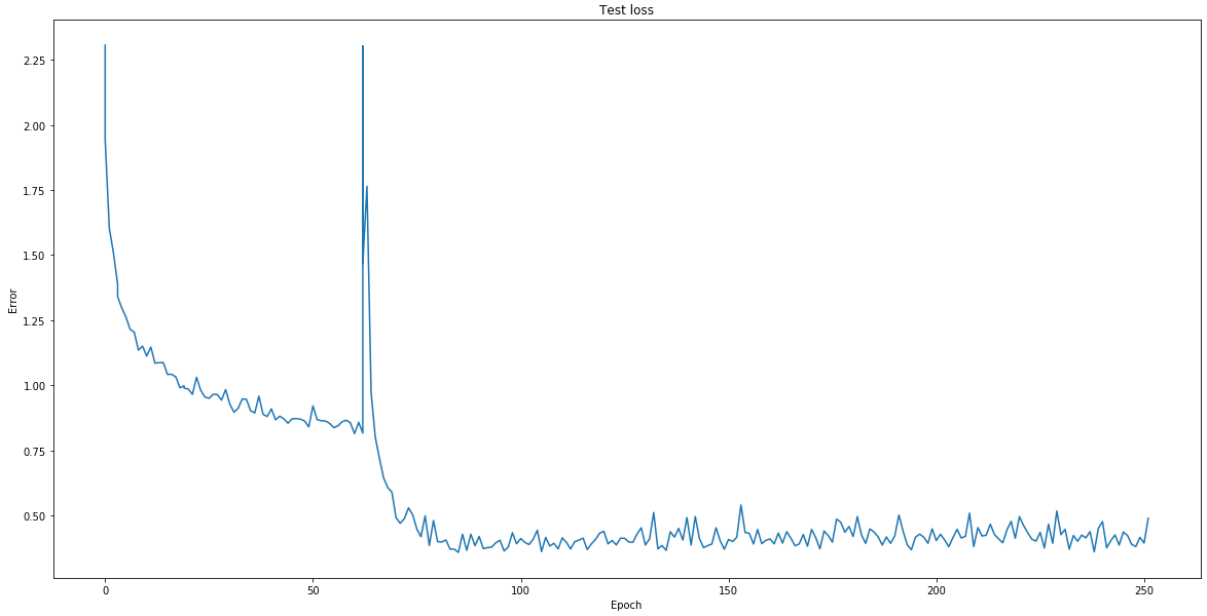
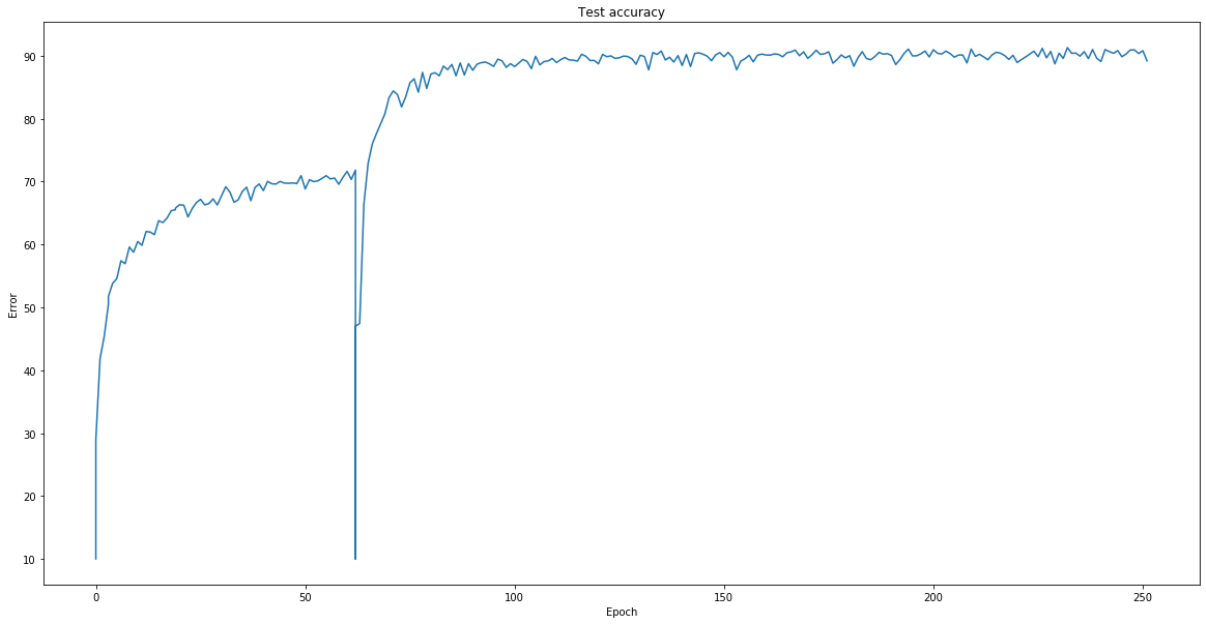
Cifar answers:

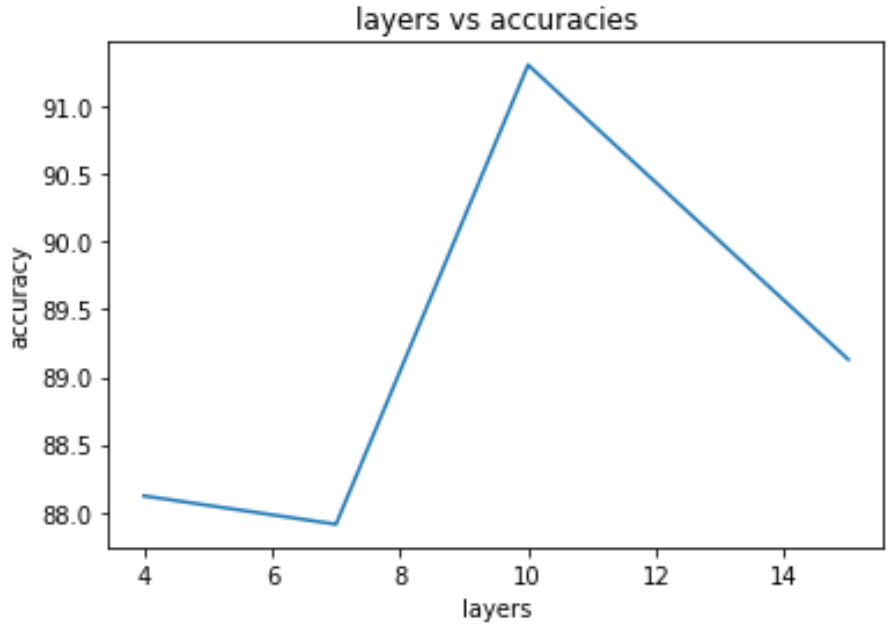
1) I got 91.31% with 10 convolutional layers, using batch norm and dropout. Network mostly converged after the 100th epoch. The following are the graphs that I got for train and test losses as well as test accuracy.







I tried different layer sizes and their results are as follows:



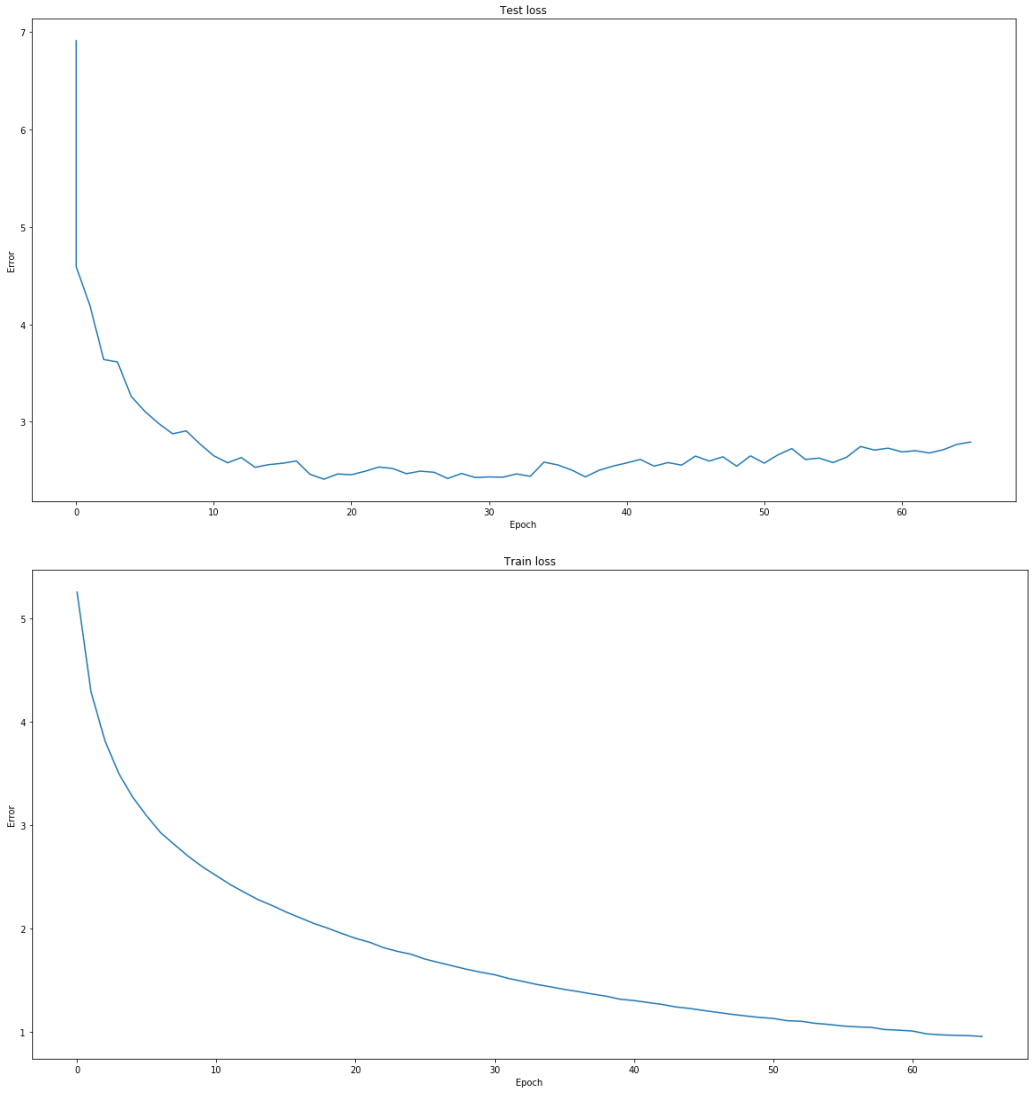
It is apparent that, on Cifar, the network started to get overfitting after 10 layers and underfits before 10.

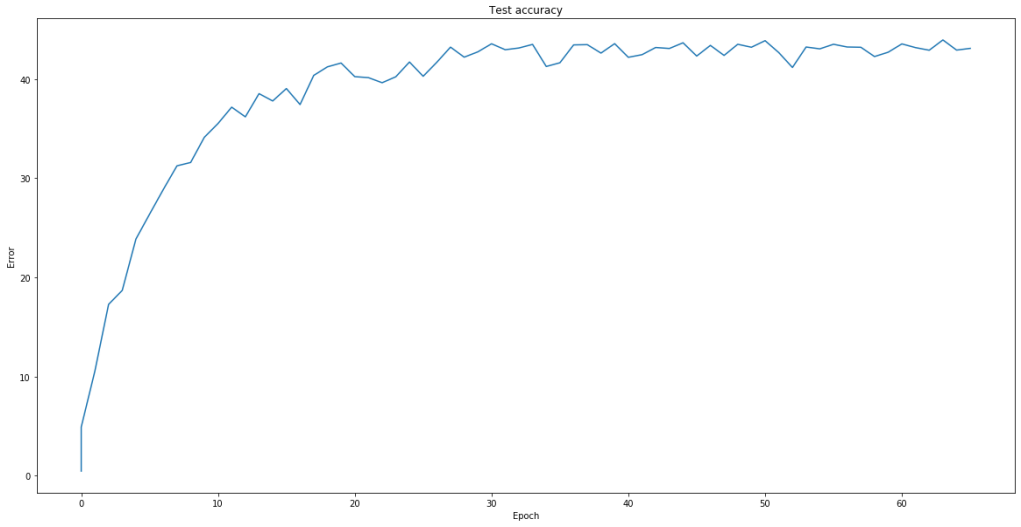
2) Using a small number of fully connected neurons worked worst. Because even though the convolutional base learns relevant features, the network has no power to use their non-linear combinations in its FC layer if the number of neurons in the FC layer is small. Also using a bigger learning rate (0.1) proved to be bad. Batch size smaller than 256 did reduce the accuracy. I also tried out the ADAM optimizer which did not improve the accuracy.

3) Because the worsts ones either learn the bias or variance of the data, i.e. they underfit or overfit. The best one just generalizes the train data well enough to do well on the test data. When one uses many layers, the network gets very flexible which ends up with severe overfitting if it is not fed with enough data. This happened in the worst case when 15 conv layers are used as the training accuracy hit nearly 100% and test accuracy stopped improving.

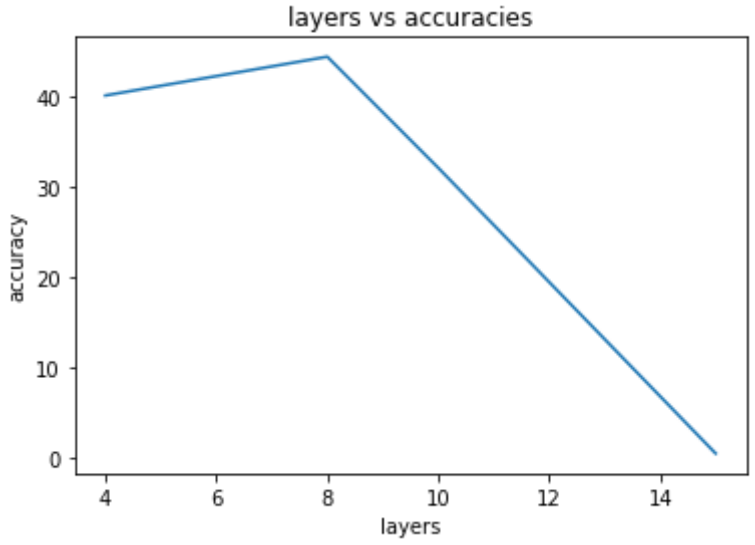
TinyImageNet answers:

1) I was able to achieve 44.4% with 8 conv layers, using batch norm and dropout. Results shown below:





Imagenet case is harder compared to the Cifar because we have 20 times more classes (10 vs 200) vs 2 times more samples (50000 vs 100000). Also, the classes in Cifar are distinct from one another and thus easy to separate. The same thing does not apply for Imagenet where we have many classes and I suspect our dataset does not have enough entropic capacity for us to fully exploit to map those classes to different regions in the feature space because of the input dimensionality being small (64x64). So a 10 layer CNN didn't perform well than that of 8 layer model in Imagenet and when you go up to 15 layers, it completely fails. Using 15 layers for Cifar was also overfitting but not as severe as in the Imagenet case.



Following are the results of my best model:

2) Larger images mean classes might become separable with stronger models so I might try deeper models on that. As in the native Imagenet case, people generally

use input dimensions that are higher than 256x256 (current state of the art uses 600x600 inputs and yields about 97.1% in top 5 accuracy https://kobiso.github.io/Computer-Vision-Leaderboard/imagenet.html) and having high accuracies with deeper architectures. So I expect 128x128 inputs to yield better results on deeper networks.

3) Larger images mean classes might become separable with stronger models so I might try deeper models in that case. As in the original Imagenet case, people generally use input dimensions that are higher than 256x256 (current state of the art uses 600x600 inputs and yields about 97.1% in top 5 accuracy https://kobiso.github.io/Computer-Vision-Leaderboard/imagenet.html) and having high accuracies with deeper architectures. So I expect 128x128 inputs to yield better results on deeper networks compared to 64x64 inputs. As for the smaller input dimensionality case, maybe using fully convolutional networks would help because we already have a small dimension that we can use in the Softmax layer without using any max pooling and fully connected layers. But I strongly doubt that 32x32 inputs have enough entropic capacity for us to exploit so regardless of which model that we use, all models will yield poor results.