Tiny ImageNet

1. What design that you tried worked the best? This includes things like network design, learning rate, batch size, number of epochs, and other optimization parameters, data augmentation etc. What was the final train loss? Test loss? Test Accuracy? Provide the plots for train loss, test loss, and test accuracy.
    a. The model based on VGG16 got the highest accuracy for me. It is identical to the VGG16 model except that the spatial dimensions of my net are smaller. I also don't have multiple fully connected layers at the end, I have just one. The images start at 64x64 and end at a 2x2x512 in the last conv layer instead of 7x7x512 in vanilla VGG16. I added batch normalization after every conv layer. I used a learning rate of 0.01 and weight decay of 0.001.
    b. The final train loss is 0.0104
    c. The final test loss is 2.366
    d. The final test accuracy is 48.0

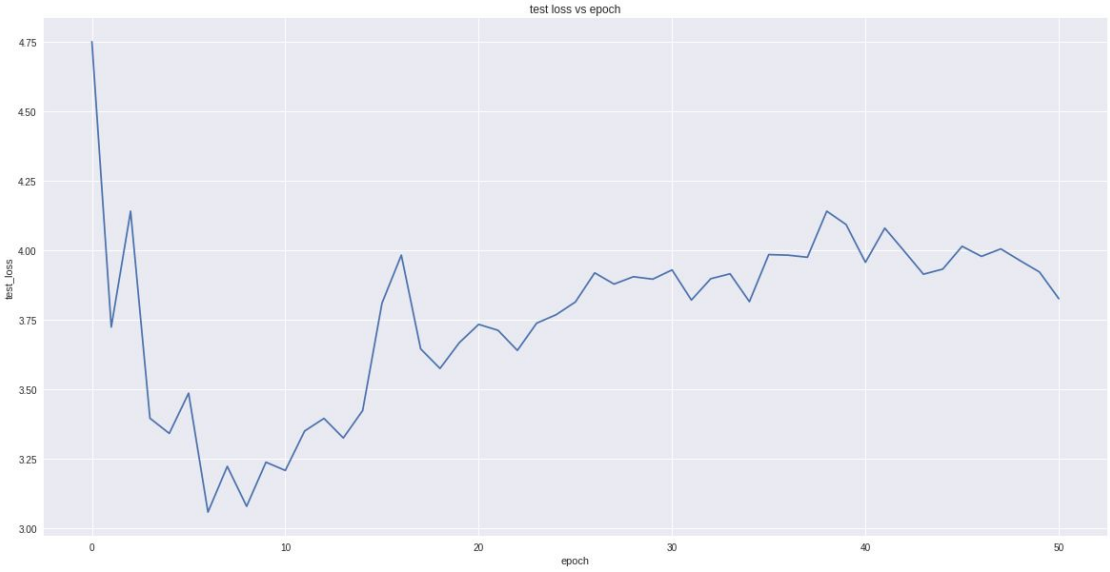
    e.

f.

g.

2. What design worked the worst (but still performed better than random chance)? Provide all the same information as question 1.
    a. The final train loss is 0.365
    b. The final test loss is 3.825
    c. The final test accuracy is 27.72

d.


test loss vs epoch
test_loss
epoch
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
0
10
20
30
40
50

e.

f.

3. Why do you think the best one worked well and the worst one worked poorly.
    a. The VGG16 net is deeper and has more convolution layers than the darknet reference network. This is probably why it outperforms the darknet reference net. However, it is also much slower. It takes an order or magnitude longer to run one epoch.
    b. VGG16 has more than one convolutions before a maxpool which might allow better gathering of information than having a maxpool after every convolution layer.

Full ImageNet

1. What design that you tried worked the best? How many epochs were you able to run it for? Provide the same information from Tiny ImageNet question 1.
    a. Of the models I tried, the darknet reference model worked best. It was a combination of how accurate the model was as well as how long it took to train it. The VGG16 I used for TinyImagenet would have probably performed better, but it took multiple hours to run even a single epoch. The darknet reference model performed much faster because it had much fewer layers. However, its accuracy was not as low proportionally. This might be because of the balanced nature of the model with each convolution doubling the number of channels followed by a maxpool that reduces the spatial dimensions.
    b. The final train loss is 2.874
    c. The final test loss is 3.340
    d. The final test accuracy is 30.46

e.

f.

g.

2. Were you able to use larger/deeper networks on Full ImageNet than you used on Tiny ImageNet and increase accuracy? If so, why? If not, why not?
    a. No, using a deeper network on full ImageNet was too slow to run effectively on Google Colab. This might be because the large test set makes every epoch much larger. This ends up taking too long to run. However, since the task is harder than tiny ImageNet, using a larger network would probably perform better than a smaller network
3. The real ImageNet dataset has significantly larger images. How would you change your network design if the images were twice as large? How about smaller than Tiny ImageNet (32x32)? How do you think your accuracy would change? This is open-ended, but we want a more thought-out answer than "I'd resize the images" or "I'd do a larger pooling stride." You don't have to write code to test your hypothesis.
    a. Here are a few things I might do for larger images:
        i. Use a larger convolution size on the initial layers because the net might need to look at a larger space to infer the basic patterns. This will probably have a higher accuracy than an identical net with a smaller convolution size in the initial layers.
        ii. Add dropout layers. Since the image is larger than before, it might contain extraneous information that might not be necessary for the network.
    b. Here are a few things I might do for smaller images:
        i. Reduce the number of maxpool layers without reducing the number of convolution layers. Since there isn’t that much spatial information to begin with, to make the net deep enough, the number of maxpool layers needs to be decreased.