Tommy Merth, Vivaan Bhatia
CSE 490G1

Tiny ImageNet

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.

Network Design

Initially, our convolutional nets did not expand in number of channels (we only performed simple convolutions and maxpooling for prototyping). We noticed that the model did not perform well, and we decided to use AlexNet and DarkNet as inspiration. After upping the number of output channels in our convolutions, we noticed that the model performed significantly better, although began to overfit. Finally, we noticed that including two linear layers (with ReLU) after the convolutions seemed to significantly improve performance.

Hyperparameters

Since we noticed overfitting, we immediately reduced the number of epochs to something more reasonable (around 30-50). We also noticed that we could increase our batch size without largely sacrificing performance. In order to make the model train faster, we increased the learning rate by a factor of 2.

Data Augmentation

We applied some simple image transformations to our training set, such as rotations and randomized horizontal/vertical flips. However, we didn't find a combination of augmentations that significantly improved accuracy without hurting performance, so we decided to focus more on network design.

What design worked the worst (but still performed better than random chance)? Provide all the same information as question 1.

Network Design

Our initial designs, as described above only achieved accuracies around 4-5%. We believe this is because the convolutional layers did not extract as many features (due to a low number of output channels). Another reason that this model underperformed could be due to it only have one output linear layer. Most likely, this does not provide enough parameters to generalize predictions based on the features extracted from the convolutional layers.

Hyperparameters

These initial, poorly performing models used approximately the default hyperparameters (although we noticed overfitting very early on and decreased the number of epochs).

Tommy Merth, Vivaan Bhatia
CSE 490G1

Data Augmentation

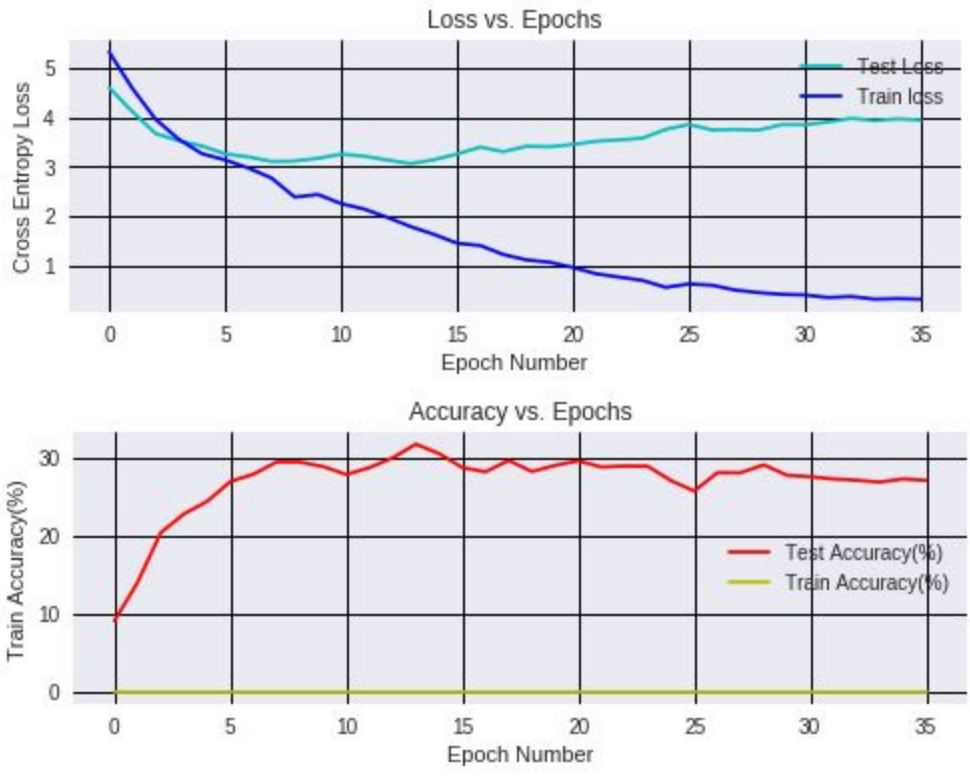
No data augmentation was used for this primitive model.

Why do you think the best one worked well and the worst one worked poorly.

Our earlier, more primitive model didn't have enough parameters to represent a powerful enough model. Because of this, the convolutional layers couldn't extract enough features, and the model's best guess is not based off of enough information.

Results from Tiny ImageNet model

Final test loss: 4.02
Final train loss: 0.314
Final accuracy (best): 34.3%



Tommy Merth, Vivaan Bhatia
CSE 490G1

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.

   Disclaimer on Full ImageNet: we attempted many times to run Google Collab, but ran into many spontaneous errors and shutdowns that seemed to be very common on this platform. Thus, we were not able to provide graphs for this section. My partner and I spent over 30 hours on this assignment and definitely learned a lot. And while we do not have graphs to show for our work, we would like to talk about our design decisions in the context of Full ImageNet and the reasoning behind those decisions.

   Network Design
   We theorized that our model from Tiny ImageNet was very capable at extracting semantic features from the images themselves. And since the images used in this dataset are merely a superset of the ones from Tiny ImageNet, we decided to follow our model closely from the first part (the first convolutional layers). However, we inevitable needed to modify our network to predict 1000 classes. To do this, we expanded the number of parameters in the linear layers to allow for a more powerful model after the feature extraction. This involved increasing the output dimension to 1000, but also more than doubling the dimension of each previous layer.

   Hyperparameters
   See disclaimer.

   Data Augmentation
   Our approach for this dataset largely resembled the augmentation from the Tiny ImageNet model. This is simply because the input data (images) are exactly the same format as from the previous.

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?

   See disclaimer.

   Although we weren't able to test this, we would expect that the depth of the network would have a marginal effect on accuracy. While the model could become more powerful, introducing more layers could make the model more prone to overfitting—and possibly negate the benefits from including more layers.

Tommy Merth, Vivaan Bhatia
CSE 490G1

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.

   With larger images, we would expect our model to perform considerably worse, since the strides/size of the convolutions aren't tuned for larger images. However, the architecture of our model probably would not change much—the convolutional layers would still need to extract the same types of features as in the Tiny and Full ImageNet datasets used here.

   The same could be said for smaller images—we would need to adjust the hyperparameters of the convolutional layers, although the architecture would most likely remain the same.