CS182 Project Milestone

Summary

Before we begin pruning filters and fine tune again for convolutional layers (see project proposal), we need a model that has acceptable accuracy. For the project milestone, we set up the model, visualization tools, and training framework for further improvements on the next part of our project.

Data augmentation

Since the size of the tiny imagenet dataset is limited, we applied data augmentation to enlarge the training set and to account for adversarial examples in the test set. For each image, there are a total of 9 similar images generated from using the torchvision.transforms library, namely,

- 2 images are formed by applying a random color jitter that changes their brightness, contrast, hue, and saturation
- 1 image is grayscale
- 1 image is rotated randomly between -25 and +25 degrees
- 2 images are formed by cropping and resizing to the original size, one of which is a center crop and the other is a random crop
- 1 image is formed by applying a high-probability horizontal flip and a low-probability vertical flip independently
- 2 images are formed by applying a random combination and order of the above

Through doing so, we enlarged the training set size from 100,000 to 1,000,000.

Model

We proposed a convolutional filter pruning technique that enables models with convolutional layers to be compressed and fine tuned to achieve a higher accuracy and a faster training process. To test our idea, we first need several models that have acceptable training and validation accuracy so that we can employ our methods on them.

For best use of our time, we used pretrained models on imagenet that are available on PyTorch. We tried freezing the convolutional layers for feature extraction and fine-tuning the fully connected layers, which drastically reduced training time. We also tried freezing the entire pretrained model (including the last FC layer) and adding a new FC layer that takes the 1000 output classes from imagenet and maps them to the 200 output classes of tiny imagenet. Our

motivation for adding a new FC layer is that since Tiny Imagenet's classes are a subset of Imagenet's, the pretrained final FC layer will accurately classify images and our added FC layer will adjust those classifications for the subset of classes. Hence, the model will train quicker with higher accuracy (as adding a final FC layer is smaller than replacing the pretrained final FC layer). Indeed we observe higher accuracy earlier in the training stage. (Reaching ~30% accuracy on the training data within 1 epoch, compared to 15% accuracy within 1 epoch).

Resnet-50

Our best Resnet-50 model converges at about 28% training accuracy on tiny imagenet without augmentation.

EfficientNet-b5

EfficientNet-b5 converges at about 35% training accuracy on tiny imagenet without augmentation.

EfficientNet-b5 converges at about 63% training accuracy on tiny imagenet with data augmentation.

Validation

Currently our models perform poorly on validation data that has not been augmented (~5% accuracy), as well as training data that has not been augmented (~25% accuracy), Thus we think there is a discrepancy between how the model interprets non-augmented and augmented images. We will experiment more with image augmentation since we know the test set is on adversarially augmented images.

Visualization Example

Here is an example of a training run for EfficientNet-b5, for 10 epochs, using Tensorboard.