Convolutional Neural Networks for Tiny ImageNet Classification

Hujia Yu (hujiay@stanford.edu)
CS 231n Final Project, Stanford University

Introduction

- Our brain can easily tell apart a dog and a cat, read a sign, or recognize detailed objects in a picture. But these are actually really hard problem to solve for a computer. Teaching computers to detect objects in images or videos has great potential in speeding up tasks in industries or self-driving cars, etc. Deep convolutional neural networks can achieve great performance on visual recognition tasks.

- This project uses dataset from Tiny ImageNet Challenge. My approach to this problem is to apply layered ConvNets as in GoogLeNet, AlexNet, and ResNet[1], on the dataset. I then explored transfer learning method by retraining the dataset on the trained models of the original GoogLeNet Inception model. As in ImageNet classification, the objective of training models is to attain best performance on image classification problem.

Challenges

- Long training time and high computation power
- Prone to poorly fine-tuned initializations
- Retrain on transfer learning could easily cause over-fitting
- Millions of hyper-parameters to train

Dataset

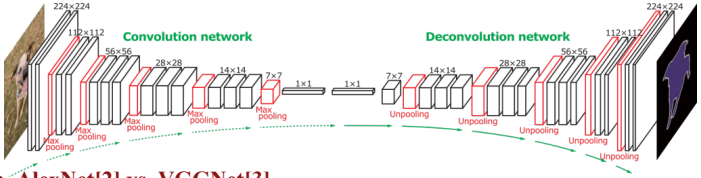
- This project uses dataset from Tiny ImageNet Challenge, which is a simplified version of ImageNet for Large Scale Visual Recognition Challenge. These pictures are collected from flickr and other search engines, labeled with the presence of absence of 1000 object categories. The Tiny ImageNet dataset consists of the same data but the images are cropped into size of 64x64 from 224x224. It has 200 classes. Each class has 500 training images, 50 validation images, and 50 test images. It uses one-hot labels.

- Images are normalized by subtracted its mean before training and testing in order to 'center' the data. This way the features have a similar range so that when we do back propagation, out gradients don't go out of control.

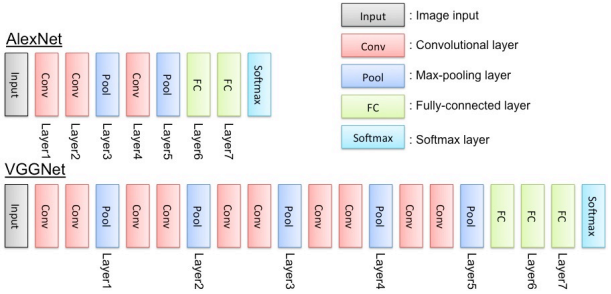
- I used test accuracy, which is the fraction of test images that are correctly classified by the model, to measure the performance.

Approaches

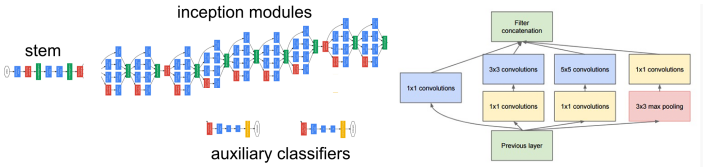
Convolutional Neural Networks



AlexNet[2] vs. VGGNet[3]

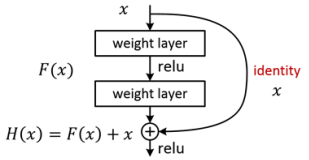


GoogLeNet



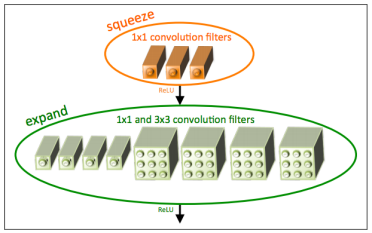
ResNet

ResNet[1] is proposed to have the gradients flow like Highway Networks to avoid degradation problem. The inputs of a lower layer is made available to a node in a higher layer. It is similar in structure with that of a LSTM but without gates.



SqueezeNet

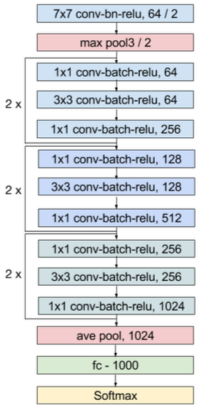
SqueezeNet is a small CNN architecture that achieves AlexNet-level accuracy on ImageNet with 50x fewer parameters, aiming to be deployed smaller CNN model on mobile devices. It uses 'fire module' as the basic building block for the model, which is composed of 'squeeze' layers and 'expand' layers.



Results

Table 1. Architecture Models and Results

Model	Train(%)	Validation(%)	Test Accuracy(%)
Baseline Model	65.40	54.40	43.04
Reduced AlexNet	43.28	43.65	33.40
Reduced VGGNet	54.62	45.88	39.40
Inception	76.03	50.43	49.40
ResNet-20	65.08	54.40	55.40



ResNet-20

Layer	Dimension
Input	64x64x3
CONV1-64	64x64x64
CONV1-64	64x64x64
Pool-2	32x32x64
Batch-norm	
CONV2-128	32x32x128
Pool-2	16x16x128
Batch-norm	
CONV3-256	16x16x256
CONV3-256	16x16x256
Pool-2	8x8x256
Flatten	1x1x(8*8*256)
FC1-1024	1x1x1024
FC2-1024	1x1x1024
Output	1x1x200

Table 2. Reduced Capacity AlexNet Architecture

- We present here both snapshots of two model structures and the results of all trained models. Table 1. gives the train, validation and test top-1 accuracies of trained models. We can see from the table that the best performing model is ResNet-20, and reduced AlexNet and VGGNet performed poorly due to incorrect fine-tuning or initializations.

Analysis and Future Directions

- According to our results above, our best performing model is ResNet-20, which gives a test error of 44.60%, and the two reduced models of VGG and AlexNet performed poorly mostly due to incorrect initializations or fine-tuning of hyper-parameters. The training process of each model was painfully long, one single epoch took at least 10 hours to train on a single CPU, and there was limited time to fine-tune and optimize the hyper-parameters,.
- ResNet-20 performed the best, I trained the model using momentum based SGD with momentum of 0.9 starting with a learning rate of 0.1, batch size of 256, weight decay of 0.0001, for a total of 5 epochs. Results could be improved further if more epochs were run.
- Future direction is definitely to retrain some of the models to find the best-performing hyper-parameters and more fine-tuning.

References

[1] S. R. J. S. Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition, 2016.
[2] S. I. Krizhevsky, A. and G. E. Hinton. Imagenet classification with deep convolutional neural networks, 2012. In NIPS, pp. 11061114.
[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition., 2014. arXiv preprint arXiv:1409.1556.