

Background

Image classification is a core task within Computer Vision that continues to be improved upon as more novel Convolutional Neural Network (CNN) architectures are designed and benchmarked through the ImageNet Challenge[1]. This project is a combination of experiments with varying architectures to benchmark on this year's Tiny ImageNet Challenge – a smaller version of the ILSVRC which has been used as the standard benchmark for a multitude of computer vision tasks.

Problem

The challenge in image classification is extracting quantifiable features from a 3 channel pixel matrix -- in the past algorithms were written to manually detect edges and attempt to extrapolate shapes that could be used for classification. However, these solutions do not offer the scalable solutions. Starting in 2012, Convolutional Neural Networks began dominating the image classification space -- and while deep learning with ConvNets has yielded amazing results, we continue to look for more computationally efficient, more accurate, and more descriptive models to apply to this task.

Algorithms

The primary algorithm leveraged to train deep neural networks is back-propagation -- an efficient way to leverage the chain rule in order to compute gradients by calculating a forward and backward pass through the network.

Given that our problem is a multi class classification, the standard loss function used is the Softmax or cross-entropy loss. Note that all convolutional layers also include l2 regularization. As such we define the loss function as:

$$L = \underbrace{\frac{1}{N} \sum_i L_i}_{\text{data loss}} + \underbrace{\frac{1}{2} \lambda \sum_k \sum_l W_{k,l}^2}_{\text{regularization loss}} \quad L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

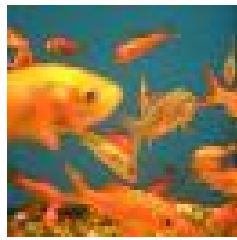
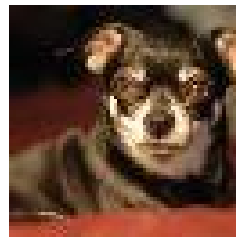





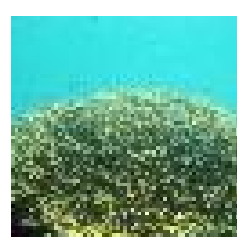
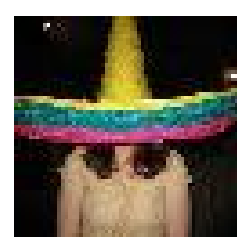

Finally, given our loss, we minimize it by leveraging the stochastic optimization algorithm, Adam - which yields the following update rule:

$$\theta_{t+1} = \theta_t - \frac{\alpha m_{t+1}}{\sqrt{\hat{g}_{t+1} + \epsilon}}$$

Data

To train, validate, and test the models, I used the Tiny ImageNet Dataset -- composed of 3x64x64 pixel images across 200 classes. The training data was preprocessed by subtracting the mean image. During training, the training data was augmented randomly by rotating up to 45 degrees, flipping horizontally, and shifting height and width ranges.

Dataset Breakdown

Train	100,000					
Validate	10,000					
Test	10,000					

Sample Images

Model Architectures

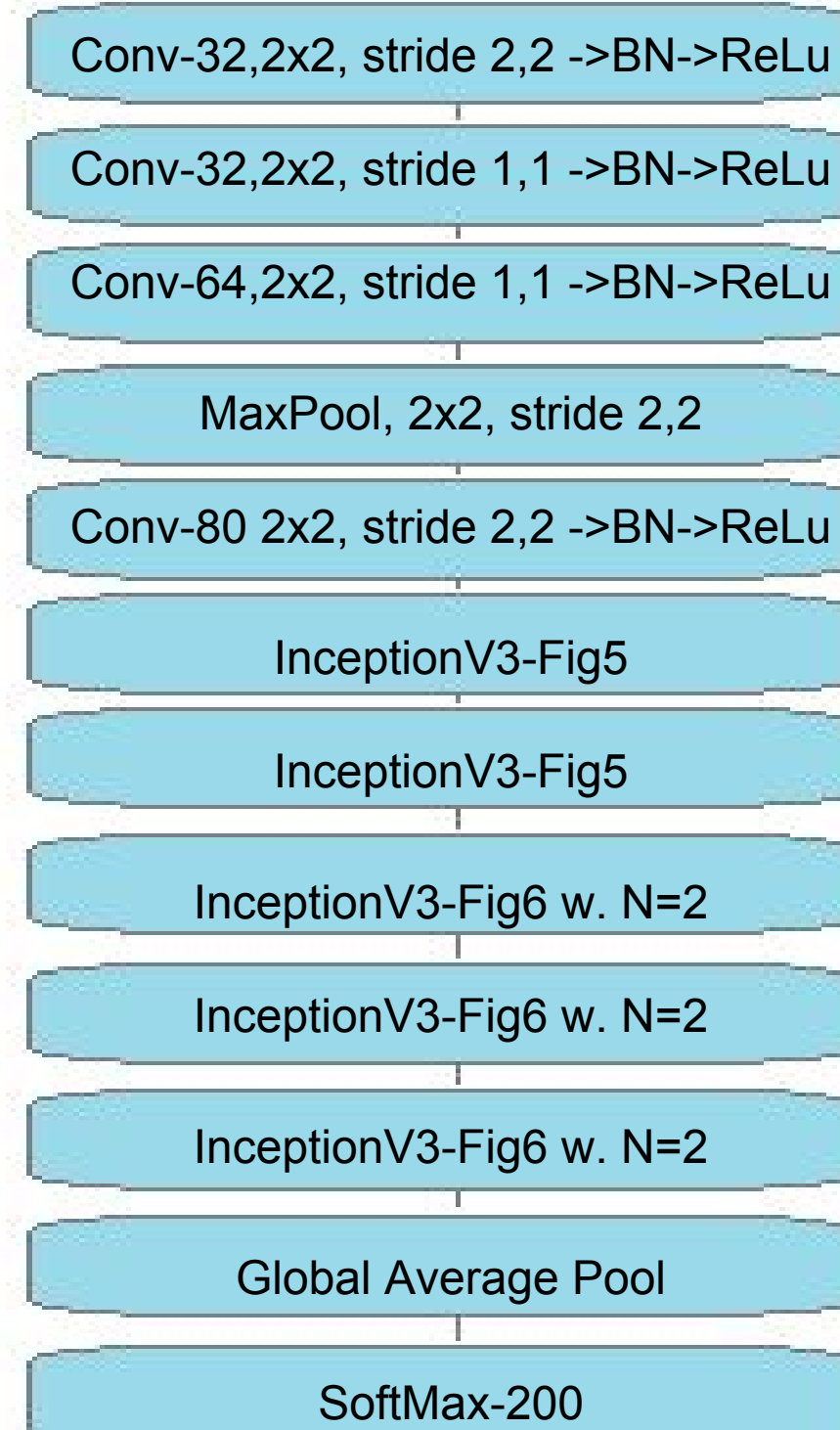
VGG-Like

This model was inspired by the VGGNet architecture -- due to the smaller resolution images, the filter sizes and strides were reduced, as well as the number of layers.



InceptionV3-Like

This model was inspired by Google's Inceptionv3 framework -- once again, all filters were reduced to 2x2, and the number of layers and inception modules used was reduced as well.



Metrics & Results

To benchmark the models, the hold-out 10,000 image validation set was used. Test set performance is being held out until further iteration. The primary metrics used are the cross entropy loss defined in Algorithms, and the validation accuracy given by: (True Positive + True Negative) / (Total Examples). To measure accuracy, a one vs all model was applied to the multi class classification.

*****models will be trained further and graphs will be added before poster session****

Classification Metrics				
Model	VGG-Like		Inceptionv3-Like	
Metric	Train	Val	Train	Val
Loss	2.5019	2.4997	2.5019	2.4997
Accuracy	0.3679	0.3453	0.4073	0.4071

Challenges

Both VGG and Inception models went through multiple iterations -- the most common problem during training was over-fitting. Adding Batch Normalization instead of DropOut, reducing the filter sizes and number of layers helped produce better results. Also, learning rate was adjusted manually throughout training to improve generalization.

Future

Prior to testing on the test set, and in turn the live competition server, I will be training some slight variants of the VGG and Inception models so as to build ensembles off of them -- in order to average the predictions. I intend to also continue training the models for further epochs so as to continue minimizing loss.

Time permitting, I would like to revisit other architectures such as ResNet and Fractal.