Building Convolutional Neural Networks with Caffe

Ravenna Thielstrom and Razi Shaban, Swarthmore College

I. Introduction

A traditional neural network, when trained to classify images, are usually programmed to take long strings of binary for their input, a translated representation of the actual picture. The more complex the pictures are, however, the larger the input must be in order to specify all aspects of the image (color, saturation, etc). A larger numbers of parameters in turn leads to a higher probability of overfitting. Convolutional neural networks offer a solution to this problem with a structure specialized for images, in which the network trains on multiple channels of input which each convey an aspect of the image.

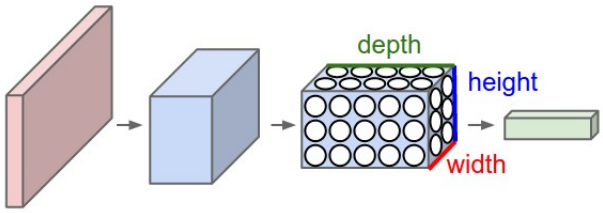

Fig. 1. Basic illustration of CNN model [1]

For example, a colored 32x32 picture like the ones used in the CIFAR-10 dataset could have three channels of input which correspond to red, green, and blue values for each pixel. This results in a depth of three for the convolutional model, meaning that both the input and the ensuing hidden layers are represented in three dimensions as opposed to one.

A significant difference between a traditional neural network and a convolutional neural network is that the layers are not fully connected to the input due to the likelihood of overfitting. Instead, each neuron is connected only to a local region of the input, building spatial relations between layers [1].

We were curious about the efficacy of a single inception module. The Inception architecture [2] stacks a series of inception modules at the top of a traditional CNN; we tried placing a single module at the top of a traditional CNN. We found that the single module did not improve our model.

II. Model

We thus tested two models: a traditional CNN and the same CNN with an additional inception module. Our traditional CNN is composed of three types of layers:

- Convolution layers take a set of filters as parameters and outputs a stack of activation maps formed by passing the filters over the spatial extent of the input
- ReLU layers apply an activation function non-linearly to the current layer
- Pooling layers downsize the spatial dimensions of the input by summarizing the outputs of adjacent neurons

Our CNN was composed of four convolution-ReLU-pool iterations followed by a single fully-connected layer. Figure 3 contains an illustration output by Caffe of our network architecture: at each iteration of the CNN, data passes through a convolutional layer, followed by a ReLU activation layer, followed by a pooling layer. This process is repeated four times with varying convolution parameters to try to catch features at different scales.

The fully-connected layer at the end of the network is connected to all neurons in the previous layer and computes its scores for each of the possible classifications of the image. The output of this fully-connected layer is then passed through a SoftMax layer to normalize output.

In addition to this basic network architecture, we experimented with adding a single inception module to see if it would help or hurt our convolutional network [2].

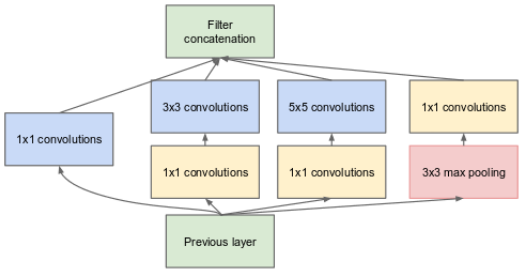

Fig. 2. An inception module [2]

The core idea of the inception module is to use 1x1, 3x3, and 5x5 convolutions in parallel with max pooling to capture a variety of structures at different levels. Adding 1x1 convolutions to each of these layers allows for dimensionality reduction to avoid a blowup in the number of parameters.

In our experimental model, we attached a single inception module to the end of fourth convolution-ReLU-pool iteration, before our fully-connected layer. This architecture can be seen in Figure 4.

III. Experiments

We trained both our normal architecture and Inception architecture on the CIFAR-10 dataset, a dataset which contains 60,000 32x32 color images which each can be classified into one of 10 classes. The network takes each image as a 32x32x3 input, with the depth of 3 accounting for the three color channels (RGB). For both architectures, our learning rate was 0.001 and our weight decay was 0.004.

Every 1,000 training iterations (2 epochs), the network carried out testing for 100 test iterations and reported its resulting accuracy.

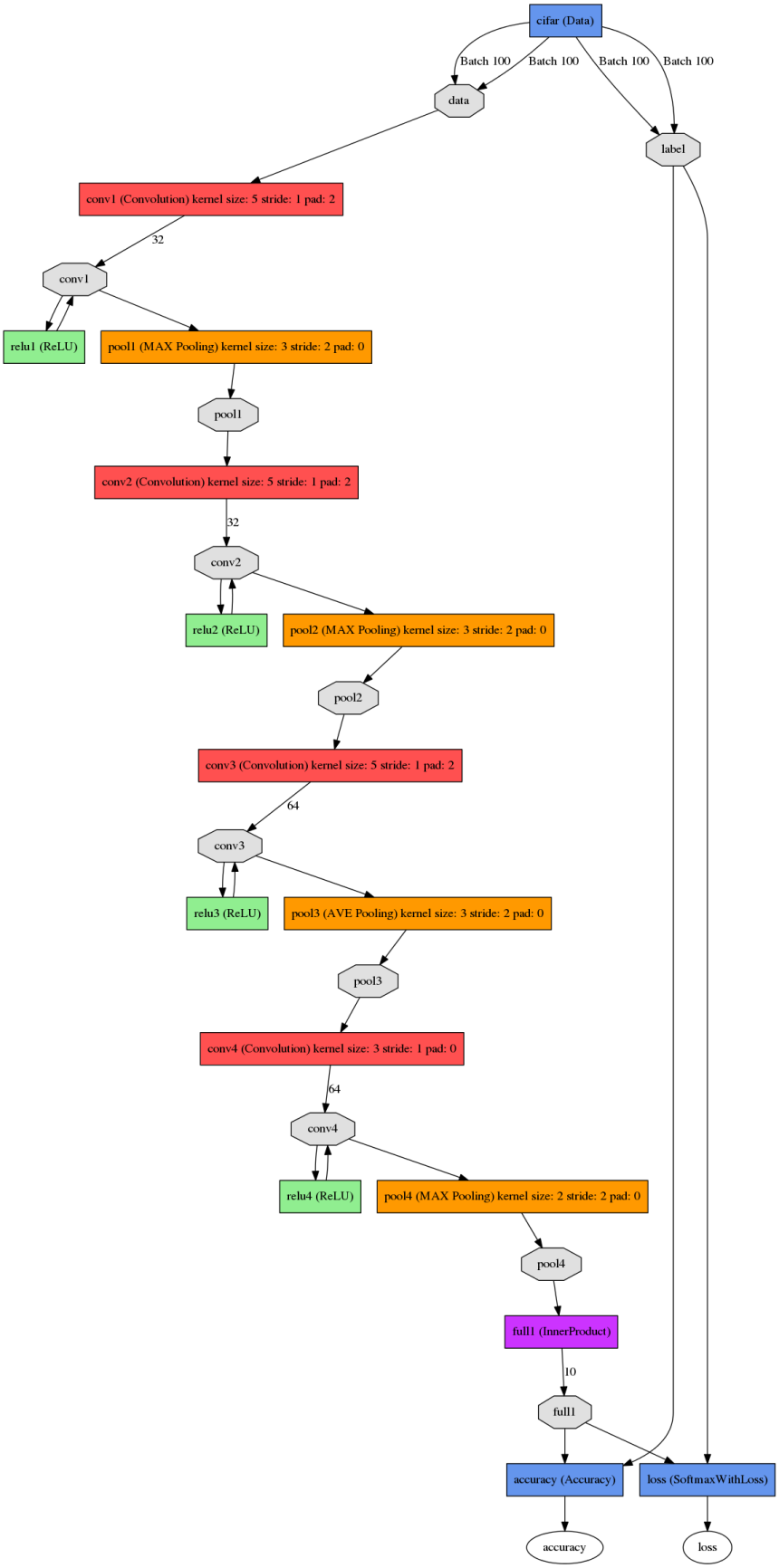

Fig. 3. Our basic CNN architecture.

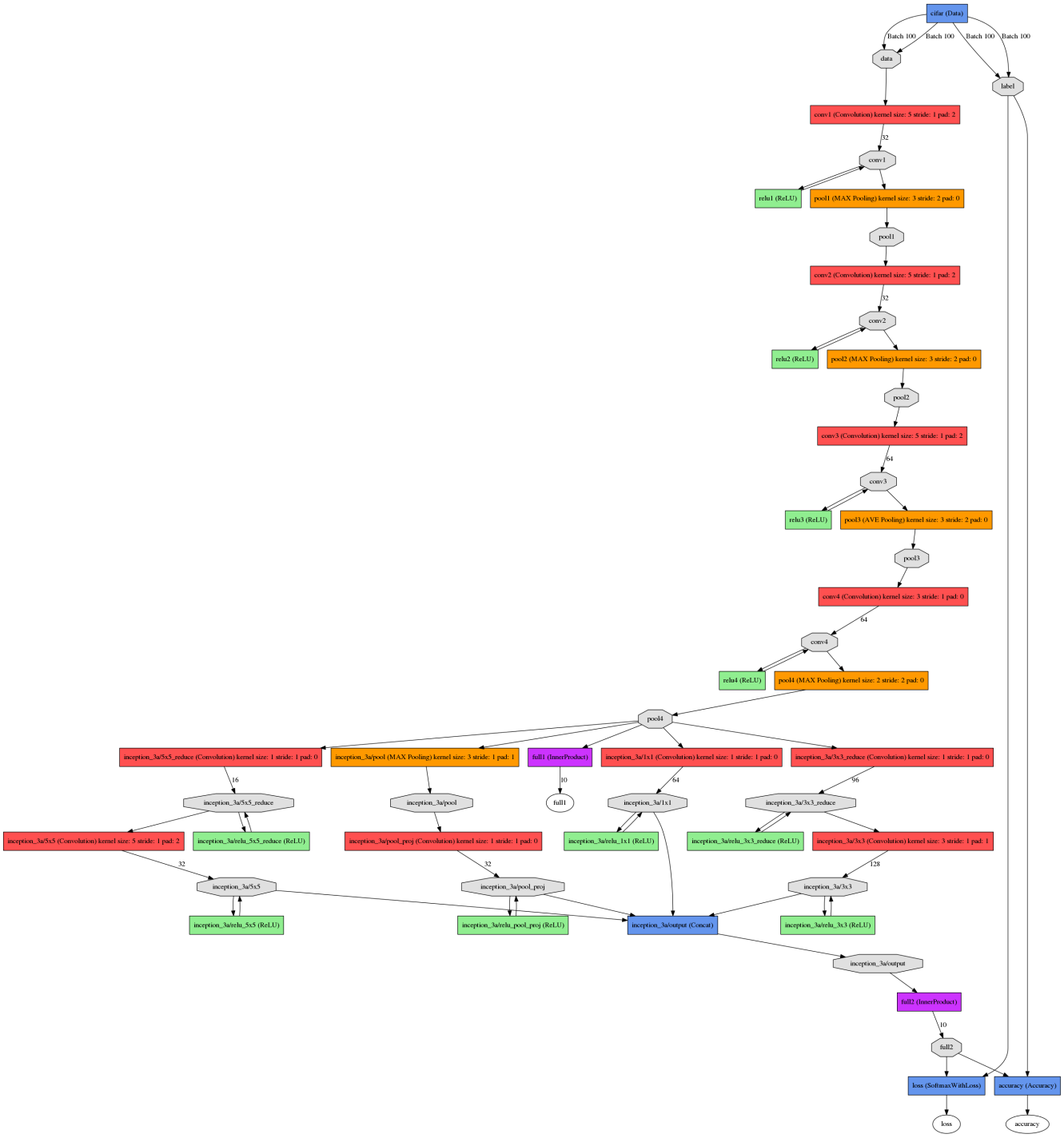

Fig. 4. Our CNN architecture with an Inception module added.

IV. Results

After 20,000 iterations (40 epochs) our regular CNN obtained an accuracy of 74.52% on test data, while our Inception CNN obtained an accuracy of 74.08% on test data. Both networks had leveled off around the 74% mark several thousands of iterations back, demonstrating no signs of increasing in accuracy any further. Although this fairly high accuracy rate confirms the success of CNNs in general, the negligible difference in accuracy between the two demonstrates that one Inception module tacked on to the end of a regular CNN will not offer much additional help during training.

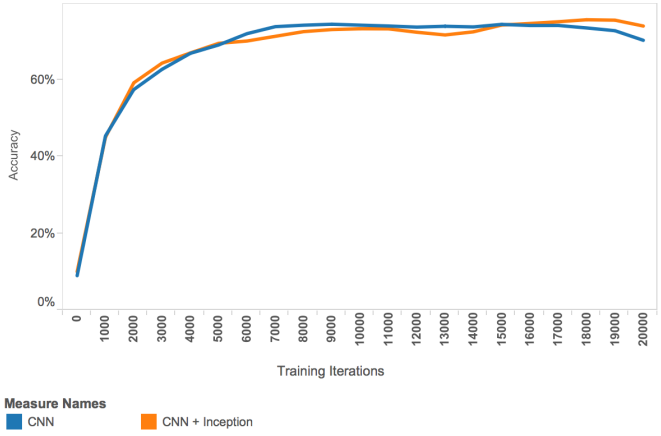

Fig. 5. Model accuracy over training

V. Conclusion

Our findings suggest that the Inception module is not itself a significant architecture, but rather allows for more efficient construction of massive serial architectures utilizing large computational resources. Notably, the effect of a single Inception module increased our epoch training time from seconds to minutes. Stacking many modules together in series as in [2] remains outside our computational access. Therefore for the moment Inception modules seem to be a tool best used to fully construct a deep network on systems that can afford the burden. As an add-on to the already perfectly acceptable vanilla CNN, an Inception module is an uneffective add-on.

References

[1] Andrej Karpathy. Convolutional neural networks (cnns / convnets).

[2] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.