Neural Networks for Shape Recognition

Dalin Wang

1. Introduction:

Training artificial Neural networks has become the de facto method for image recognition problem, which is a very important topic in the machine learning field as it can support lots of interesting applications in many fields including medicine, art and finance. Recent years have seen an explosion of proprietary/open source projects aimed at building a better machine learning platform, such as scikit-learn, tensor-flow, and Teano, to name a few [1]. And as a result, we have seen more and more applications of machine learning in our daily interaction with technology---Alexa voice assistant, Apple photos and self-driving cars. For the task of image recognition, researchers have demonstrated steady progress against ImageNet[2], a benchmark problem for computer vision, and each year at the ImageNet competition, we witness a new construction of neural networks become the state-of-art: QuocNet[3], AlexNet[4], Inception(GoogleLeNet) [5], BN-Inception-v2[6]. The ImageNet is a database of 1000 different classes of images, like "Leopard, container ship and motor scooter".

In this study, I developed a Convolution Neural Network(CNN) model using TensorFlow on the BabyAIShape Datasets[7]. The task is to distinguish between 3 basic shapes—rectangle, eclipse, and triangle—in 32 x 32 images with one shape and uniform colors (one for the shape, and one for the background). The motivation for this study is for one, it is a simple task for beginners to learn about artificial neural networks; second, the model can be used in many interesting applications, e.g. an app for teaching children basic intuition of shapes in early education.

The paper is divided into 5 sections. Section 2 introduces the dataset and how we prepare and process the dataset. In section 3, I explained the three 3 steps of training a neural network with TensorFlow and the architecture of my CNN. Section 4 discusses the decision on hyperparameters like learning rate and batch size, and the different types of optimization methods. In section 5, I provided the results of my model and discussed some thoughts on aspects of potential improvements.

2. Dataset

In this study, I have used publicly available dataset BabyAIShapeDatasets. Therer are several datasets in the link provided, but the particular dataset I have used is the shape2_1cspo_2_3 dataset group, which is composed of 10000 samples of training data, 5000 samples of validation data and 5000 samples of testing data. In the github repo[9], you can find the corresponding python files responsible for generating the *amat* file, which is the file format later used by the CNN model. To generate the amat files, you need to install python[10], and then in the terminal (assuming you are using Mac OS), simply type in the following command:

```
$ python shapeset2_1cspo_2_3.10000.train.py write_formats amat
```

I also included the amat files inside the *input* folder for convenience. The .amat file is of ascii format. The value separator is the space (ascii code 0x20) It is organized as follows:

- The first line is the number of examples and the number of values per line (1031).
- On each subsequent line, the first 1024 values represent the gray tone of each pixel (or the color, depending on how you interpret it) as a floating point value between 0 and 1, inclusively (32 lines of 32 pixels). The next 7 values are:
- The shape: 0=rectangle, 1=ellipse and 2=triangle
- The color of the shape: this is actually an integer between 0 and 7. Divide by 7 to get the corresponding gray tone.
- The x coordinate of the centroid of the shape, between 0 (leftmost) and 256 (rightmost).
- The y coordinate of the centroid of the shape, between 0 (top) and 256 (bottom).
- The rotation angle of the shape, between 0 (no rotation) and 256 (full circle). This can probably not be learnt reliably because there the reference point is ambiguous (for instance, there is currently no way to know relatively to which side the triangle was rotated).
- The size of the shape, between 0 (a point) and 256 (the whole area). There is a lower bound and an upper bound.
- The elongation of the shape, between 0 (at least twice as wide as tall) and 256 (at least twice as tall as wide).

2.1 Preprocessing the data

To preprocess the data, I first normalized the data since the color is merely a distraction when training a model to recognize shapes. By iterating over the normalize function, the pixels belong to the shape would have a value of 0.5, and the pixels of the background would have a value of -0.5. Since the training and validation data only has 15000 data points, it is too few to train an accurate model. To solve this problem, I flipped the image along the x-axis and y-axis and then translated the image to the left, right, top and bottom by 3 pixels to gain 24000 more sample data. At the end, I have a total of 30000 training and validation data combined for training.

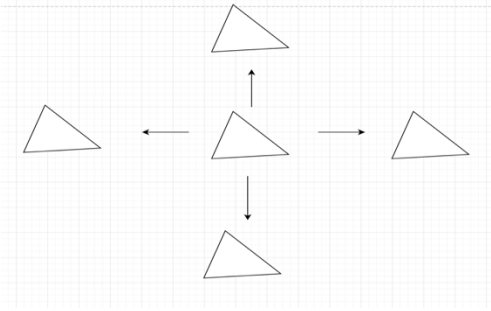

Figure 1Populate The Shape Dataset with translation operations

3. Architecture

As you can see from figure 2, we use three main types of layers to build our CNN model: Convolutional Layer, Pooling Layer and Fully-Connected Layer. Specifically, our CNN model consists of two layers of convolutional layer, each followed by a max-pooling layer. The output of the second pooling layer is passed in a fully connected hidden layer, which is followed by an output layer, which is also a fully connected layer. Notice there is also a dropout layer in between the hidden layer and the output layer.

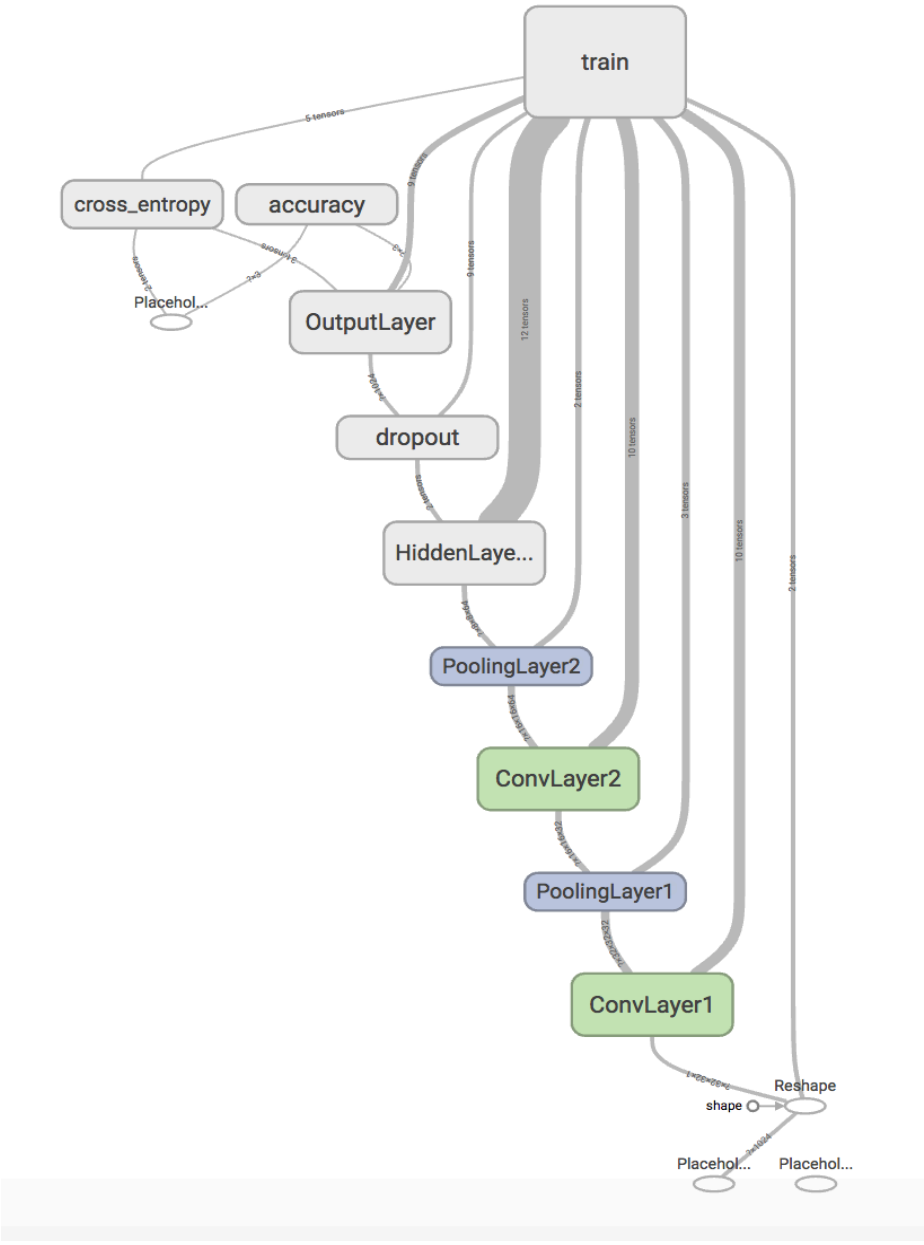

Figure 2 Tensor Graph generated with Tensor board

The convolutional layer is at the forefront of CNN model. Recall that the input of the convolutional layer is a batch of [32 x 32 x 1] images of shapes with a unified shape color and background, the default batch size is 50. The convolutional layer will compute the output of the neurons that are connected to local regions in the input by passing the local region of pixels through a filter. Figure 2 is an image I found online that can demonstrate how the convolutional layer filter interacts with the input. The input image here is of size [7x7x3], and the filter is [3x3x3] and they have applied 2 distinct filters to the input volume and the filter stride is 2. In contrast, in my CNN model, my first convolutional layer used filter of size [5x5] with one input channel and 32 output channels with 0 padding and stride of 1; my second convolutional layer used the same filter size of [5x5] with 32 input channels and 64 output channels with 0 padding and stride of 1.

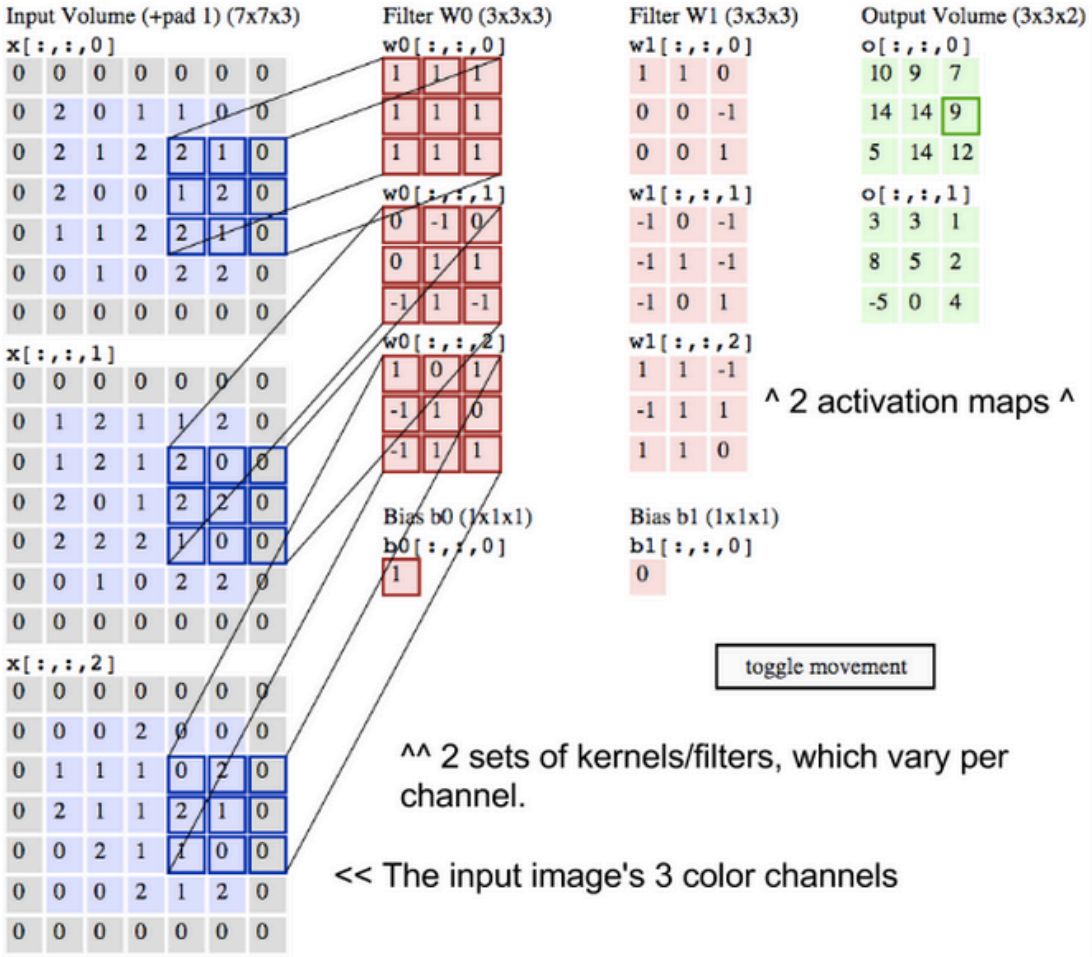

Figure 3 Credit for this excellent animation goes to Andrej Karpathy

Next layer is max-pooling layer. When applied, the max-pooling layer takes the largest value from a block of numbers in the input matrix, and places it to a new matrix next to other max values from other blocks. Lesser information is lost with this operation, however, it has the advantage of avoiding over-fitting and reducing processing time and storage.

Last type of layer is fully-connected layer, which just means it has full connections to all activations in the last layer. The activations in fully-connected layer is only computed with matrix multiplication with a bias offset. As in the case of our model, we have two fully-connected layers, the hidden layer and the output layer. It's worth noting that the only difference between the fully-connected layer and the convolutional layer is the weights in the convolutional layer is connected to local regions whereas the fully-connected layer's weights are committed to all parameters in the input matrix. As a result, the weights dimension for our hidden layer is [8x8x64, 1024], which is connected to the activations after the second pooling layer which downs the image to [8x8] with 64 feature maps for each parameter in the input matrix. Our output matrix has weights of size [1024, 3], which maps the 1024 features input 3 classification classes.

There are many commonly used activation functions in machine learning. In this project, we chose the ReLU(Rectified Linear Unit) which computes the function f(x) = max(0, x). In other words, the activation function simply thresholds the input at 0. The advantage of activation is that it was found that "it greatly accelerates the convergence of stochastic gradient descent compared to sigmoid/tanh functions due to its linear and non-saturating form"[8].

4. Softmax Classifier and Adam Optimizer

   The softmax classifier is of the form:

$$L_i = -\log\left(\frac{e^{f_{yi}}}{\Sigma_j e^{f_j}}\right)$$

   The softmax classifier normalized the class scores to class probabilities that sum to 1 and then applies the results to a cross entropy loss function. Therefore, our model is essentially a softmax classifier that tries to minimize the cross-entropy between true class distributions and estimated class distributions.

   Last, we want to introduce the gradient descent method I used for my project: "Adam Optimizer (Adaptive Moment Estimation). Adam is a method that computes the adaptive learning rate for each parameter. It is found that Adam optimizer works well in practice and compares favorably among other optimizers."[9]

5. Results:

   After training on training and validation datasets for 6000 iterations on batches of 50 samples per iteration, I ran the model on test dataset of 5000 shape images, and achieved an accuracy of 0.9941. Here is a snapshot of the accuracy on the testing dataset from tensorboard.

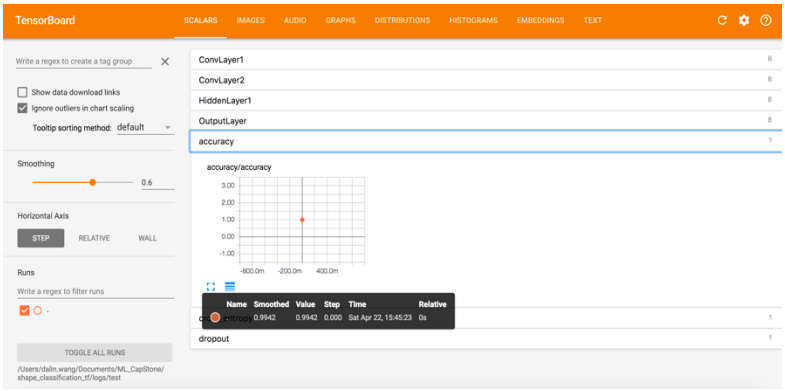

Figure 4 Accuracy on testing dataset

References:

1. Top 20 Python Machine Learning Open Source Projects in 2016: (*http://www.kdnuggets.com/2016/11/top-20-python-machine-learning-open-source-updated.html*)
2. ImageNet: (http://www.image-net.org/)
3. QuocNet: (http://static.googleusercontent.com/media/research.google.com/en//archive/unsupervised_icml2012.pdf)
4. AlexNet: (http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf)
5. GoogleLeNet: (https://arxiv.org/abs/1409.4842)
6. BN-Inception-v2: (https://arxiv.org/abs/1502.03167)
   Gradient Descent: (*http://cs231n.github.io/optimization-1/*)
7. BabyAIShapeDatasets: (http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/BabyAIShapesDatasets)
8. ImageNet Classifcation: (http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf)
9. Adam Optimizer: (http://sebastianruder.com/optimizing-gradient-descent/index.html#adam)