Rahul Ghosh
Student ID: 5476965

Assignment#3
CSci 5525

Q2: LogReg on MNIST dataset

MODEL DESCRIPTION

For Multiclass Logistic Regression, the class posterior is given as:

$$P(C_k|k) = \pi_k(w_k; x) = \frac{exp(a_k)}{\sum_j exp(a_j)}$$

$$NegativeLoglikelihhod = -\sum_{n=1}^{N}\sum_{n=1}^{N} y_{nk} log \pi_{nk}$$

$$Gradient = -\sum_{n=1}^{N} (\pi_{nk} - y_{nk})\, x_n$$

In the MNIST dataset each image is of the shape (28x28) which are flattened and stored in the form of vectors of length 784. The weights and the biases of the logistic regression model are initialized using a random initializer of the shape [784, 10] and a constant value respectively. Given the weights and biases,

$$logits = wx + b$$

The loss is calculated using the softmax_cross_entropy_with_logits and finally the GradientdescentOptimizer is used to minimize the calculated loss.

TRAINING DETAILS

```
image_size = [28, 28]
learning_rate = 0.01
batch_size = 128
n_epochs = 50
```

RESULTS

Total time: 62.4106068611145 seconds
Accuracy: 0.9134

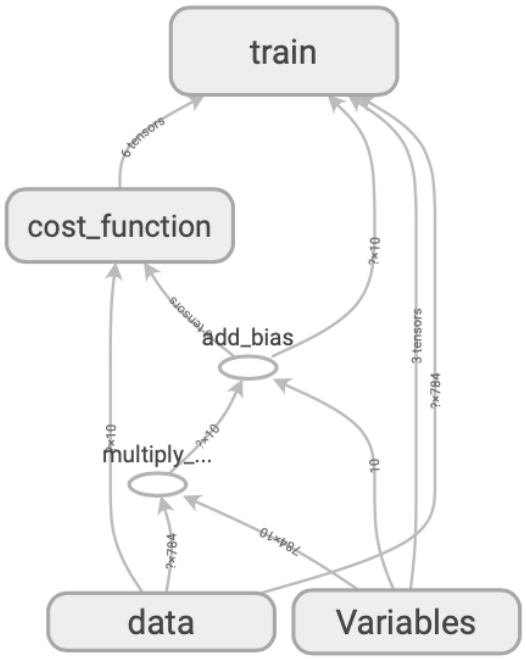

Figure 1: Logistic Regression Model

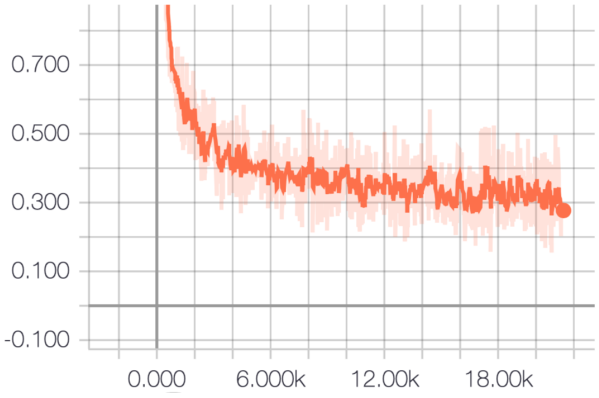

Figure 2: Model Loss vs Global_step

Q3: ConvNet on MNIST dataset

MODEL DESCRIPTION

Convolution acts as the main idea behind the working of a CNN, producing filter maps stacked over one another. A CNN is a feed forward network with such convolution layers in which non linearity is added through activation functions such as the rectified linear units. The model used in this task comprises of two convolution-pooling layer combinations followed by a fully connected layer and finally a logistic regression classifier.

$$s[i,j] = (I * K)[i,j] = \sum_m \sum_n I[m,n]K[i-m, j-n]$$

DATASET AND MODEL DETAILS

The image of size 28x28 is passed to the first convolution layer which has 32 filters of size [5, 5]. The output is then passed to a max-pooling layer which does the pooling in a 2x2 window. The output of the pooling layer is then passed to the second convolution layer which now has 64 filters of again size [5, 5] followed by a similar max-pooling layer. The output is flattened and passed to a fully connected layer of 1024 nodes, from where it is classified using the same logistic regression classifier as the previous question. The weights and the biases of the logistic regression model are initialized using a random initializer of the shape [784, 10] and a constant value respectively. Given the weights and biases, the loss is calculated using the softmax_cross_entropy_with_logits and finally the GradientdescentOptimizer is used to minimize the calculated loss.

The term dropout refers to dropping out units (both hidden and visible) in a neural network. It refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random. It is a regularization approach in neural networks which helps reducing interdependent learning amongst the neurons. Dropout is added to the fully connected layer with a probability if 0.75.

```
image_size = [28, 28]
learning_rate=0.0001
n_classes=10
batch_size=128
n_epochs = 100
dropout = 0.75
```

RESULTS

Total time: 57806.42195415497 seconds
Accuracy 0.4121

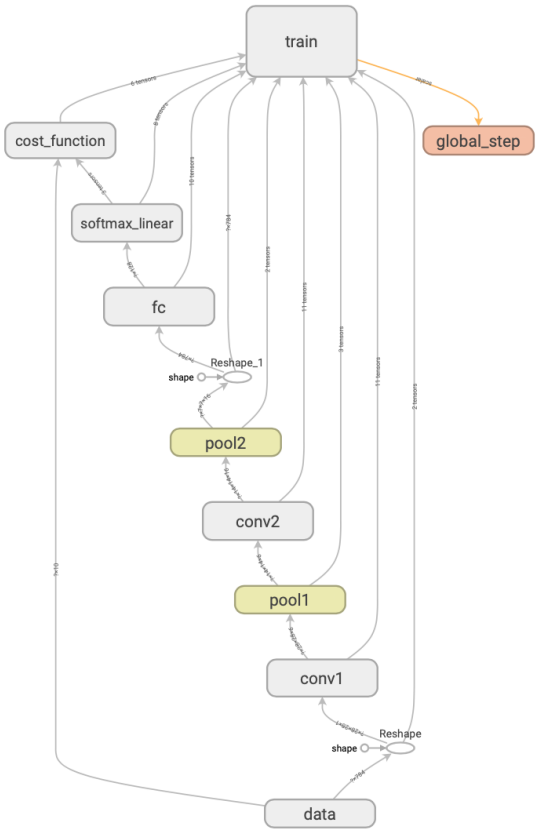

Figure 3: Convolutional Neural Network Model

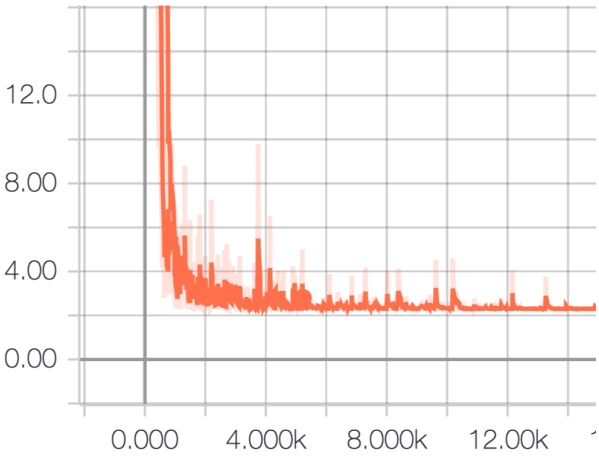

Figure 4: Model Loss vs Global_step

Q4: MyConvNet on MNIST dataset

MODEL DESCRIPTION

A similar model as that of Question3 is used with difference being in the number of parameters. The CNN model of Question3 has too many filters which results in a large number of parameters, thus leading to the model not getting converged even after 100 epochs. This results in bad test-set performance.

To overcome this the first convolution layer now has 6 filters of size [5, 5] followed by a maxpooling layer of window size 2x2 and stride 2x2. Next to focus more on the local information and take less context into consideration, the filters of the second convolution layer are reduced to size [3, 3]. There are 16 such filters and the output is then again passed to a maxpooling layer with similar configuration as that of the first one. The output of the second maxpooling layer is passed to a reduced fully connected layer with only 128 nodes as compared to 1024 nodes used in Question3.

These tweaks result in much faster convergence and also reduces the computation time of a singe epoch. The improvement in results is shown below.

DATASET AND MODEL DETAILS

```
image_size = [28, 28]
learning_rate=1e-2
n_classes=10
batch_size=128
n_epochs = 25
```

RESULTS

Total time: 610.0318353176117 seconds
Accuracy 0.9744

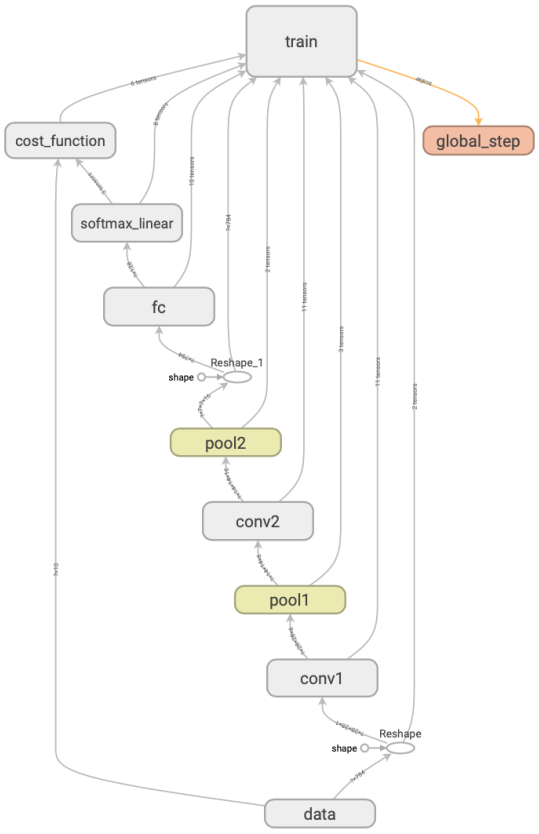

Figure 5: My Convolutional Neural Network Model

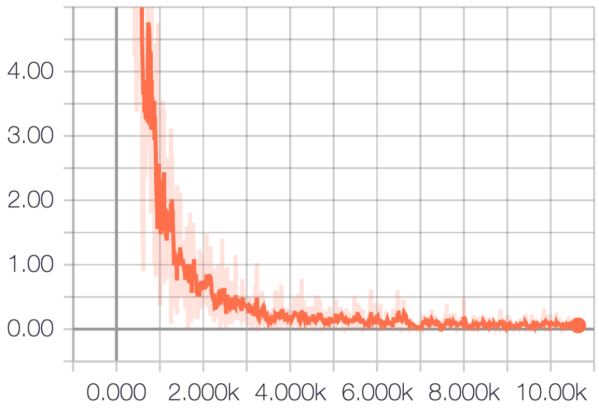

Figure 6: Model Loss vs Global_step

Q5: VanillaLSTM on MNIST dataset

MODEL DESCRIPTION

LSTMs are generally used to model sequences and operate over sequences of vectors. The input can be a sequence of vectors as well as the output. A LSTM when unrolled through time can be visualized as a feed forward network at each time step which takes a fixed size input vector and outputs a fixed size vector.

LSTMs is different from a regular RNN as they perform a bunch of operations inside each feedforward unit to take care of the long term dependencies. Instead of a single neural network layer, there are four layers interacting in a very special way through gates as shown in Figure 4 [1].

The MNIST data consists of images with size 28x28. For fitting the MNIST data into the model, we consider each row of the image as the input to the network at time t. Thus we have time series of vectors with 28 timesteps and the dimension of each vector being 28. The whole input is passed to the LSTM cell and the ouput generated by the network is a list of tensors of shape [batch_size,28] with 28 being the dimension of each output corresponding to the input at time t. The length of the list is 28 which is the number of time steps through which network is unrolled. We consider only the output of the final timestep as the prediction will only be done when all the rows are supplied to the network. The output is passed through a logistic regression layer for classification.

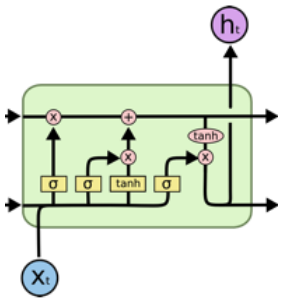

Figure 7: LSTM Cell

DATASET AND MODEL DETAILS

```
image_size = [28, 28]
time_steps=28
num_units=128
n_input=28
learning_rate=0.001
n_classes=10
batch_size=128
n_epochs = 25
```

RESULTS

Total time: 955.0442140102386 seconds
Accuracy 0.9823

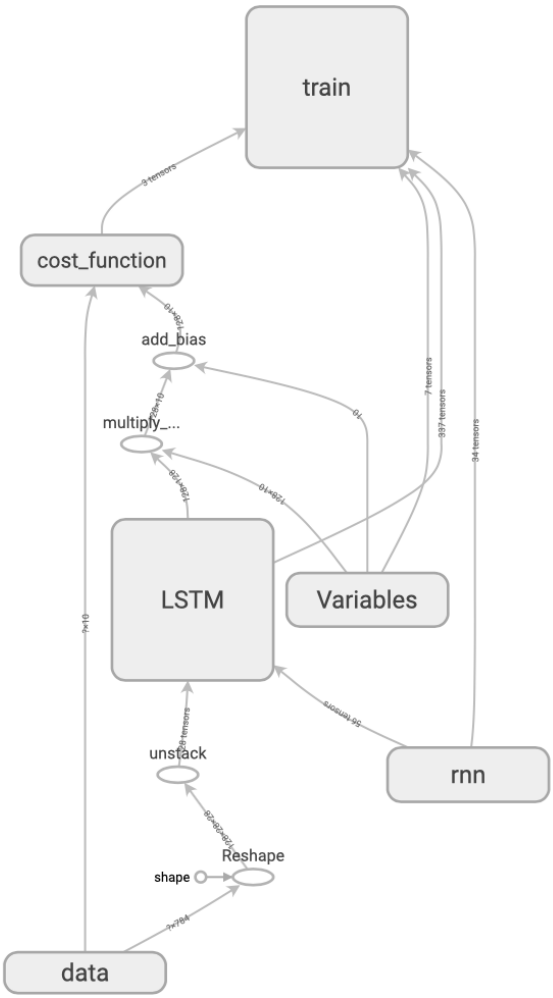

Figure 8: LSTM Model

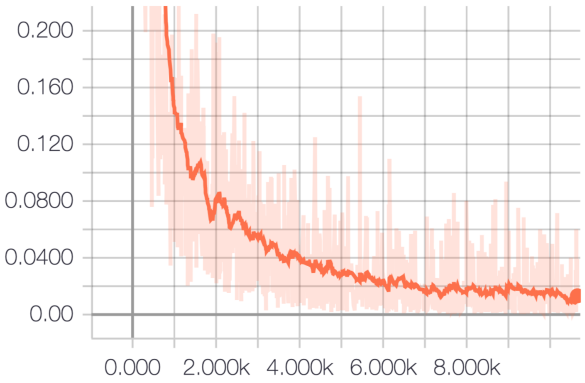

Figure 9: Model Loss vs Global_step

References

[1] http://colah.github.io/posts/2015-08-Understanding-LSTMs/