Rahul Ghosh
Student ID: 5476965

Assignment#4
CSci 5525

Solution of Question 1

a. Adagrad Algorithm

Adagrad is an algorithm that dynamically incorporates knowledge of the geometry of the data such that the updates are smaller for parameters associated with frequent features and larger for unique features. The parameters with large gradients has a small learning rate whereas the the parameters with small gradients has a large learning rate.

Gradient of the i^{th} parameter θ_i at iteration t is given by,

$$g_t^i = \nabla_{\theta_i} L(\theta_t)$$

The adaptive learning rate used by Adagrad is given by,

$$\eta_t^i = \frac{\eta_0}{\sqrt{\sum_{s=0}^{t}(g_s^i)^2}}$$

The stochastic gradient descent along with Adagrad,

$$\hat{g} \leftarrow \nabla_{\widetilde{\theta}} \frac{1}{m} \sum_{i=1}^{m} L(f(x_i; \widetilde{\theta}_t), y_i)$$

$$r_t \leftarrow r_{t-1} + \hat{g}_t \odot \hat{g}_t$$

$$\Delta\theta \leftarrow -\frac{\eta_0}{\delta + \sqrt{r_t}} \odot \hat{g}_t$$

Advantages

Selection of the optimal learning rate is one of the problems of the stochastic gradient descent. With the use of Adagrad algorithm, the need to manually tune the learning rate is eliminated as the algorithm dynamically chooses the learning rate. Moreover the fixed learning rate for all the dimensions in SGD can cause problems which Adagrad addresses with a different learning rate for each feature.

Limitations

The accumulation of the squared gradients in the denominator is a weakness of the Adagrad algorithm as the accumulated sum keeps growing during training which results in the learning rate to shrink and eventually become infinitesimally small, at which point the algorithm is no longer able to acquire additional knowledge.

The hyper-parameter η_0 needs to be chosen manually and additionally if the initial gradients are large, then the subsequent training will be slow.

b. Exploding and Vanishing Gradient

As we move backwards through the hidden layers, the gradients have to go through multiple matrix multiplications due to chain rule. If the gradients are less than 1, then their multiplication tend to get smaller. This results in the initial hidden layers learn much slowly than the neurons in the later layers. This is known as the problem of vanishing gradients. Similarly, if they are greater than 1, the error gradient can accumulate during an update, they get larger and eventually blow up. This in turn results in large updates to the network weights leading to an unstable network known as the problem of exploding gradients.

These problems occur more in the deeper layers closer to the input as the gradients accumulate over the layers resulting in either very large or very small updates.

Solution of Question 2

iii.

Problem Statement

Given the fashionmnist dataset and a set of time series of fashion choices from a set of users, the objective is to classify the items and predict the user's future choices

Background

This problem statement have resemblance to the next word predictor problem in which given a sequence of words we have to predict the next word in the sentence. A Recurrent Neural Network architecture is used for this where the inputs are the sequence of words. Word embeddings are used to vectorize the words to be given as input to the model. Word embeddings are generated such that similar words have vectors with high similarity. To draw the analogy, predicting the future preferences of the user given the time series of his past preferences can be seen as predicting the next word in the sentence. Therefore we need a method to vectorize the pictures of the fashion choices of the user such that the similar category of products have similar representation in the vector space. Once we have a vector representation given a picture, these vectors can be fed to the LSTM to predict the future choices of the user. We also need an architecture to classify the items.

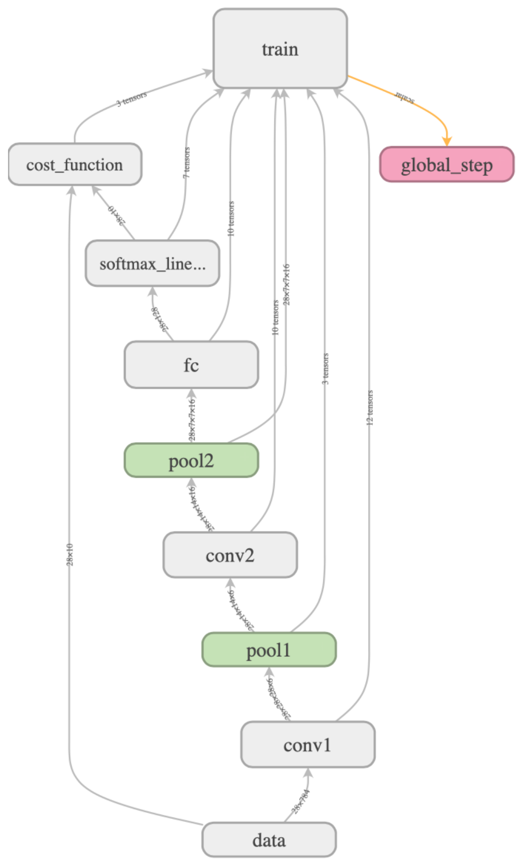

Figure 1: Convolutional Neural Network model for image feature vector generator

Solution

Given the fashionmnsit dataset, we train a CNN architecture to predict the class of the data. In this way we can train the CNN model to learn a vector representation of the dataset. Moreover, since the CNN is trained to classify similar images into same class, it will have similar representation of the images of a class.

Once the CNN architecture is trained the logistic regression softmax layer is removed to use the CNN as a feature generator. Given a sequence of images of items preferred by the user, each image is considered as an input at a single time stamp and its corresponding vector from the CNN is input to the LSTM cell. The RNN is trained to predict the next image given a sequence of images. Thus for every image the RNN should give the time shifted image as output.

Complete Architecture

The CNN architecture is shown in Figure 1. Note that we can use any complex CNN architecture to generate image descriptors. The combined architecture is shown in Figure 2.

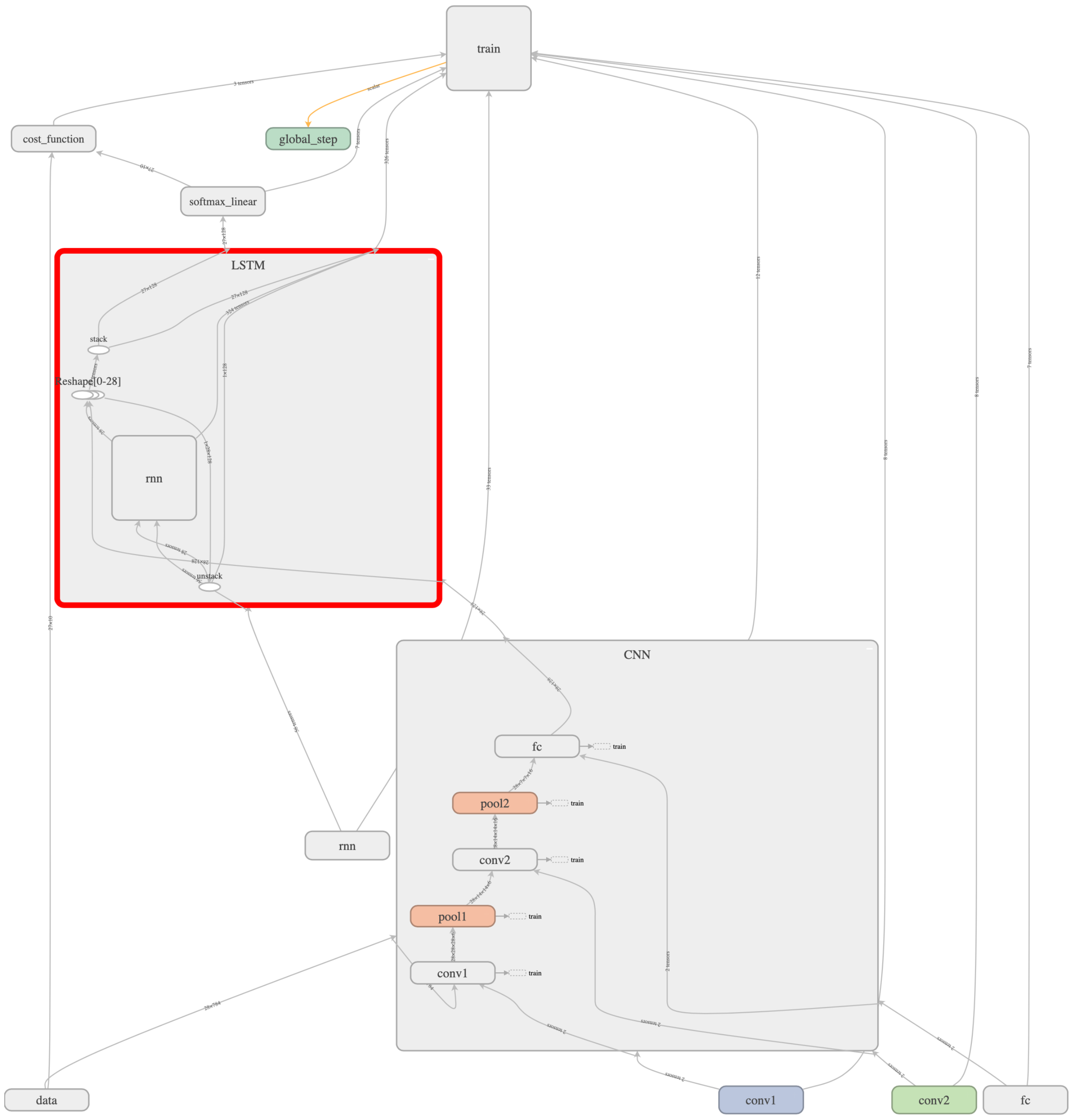

Figure 2: Combined Network Model

Objective Function and Optimization

For training the CNN we calculate the softmaxwithlogits as the loss between the logits of the model and the label of the image. For training the combined architecture.

Code Usage

In the code placeholders X and Y are used to train the CNN on the MNIST dataset. After training the CNN, the sequence of the user choices are fed into the same place holder X with batch_size now kept as the number of timesteps of the user choices and a time shifted version of the same sequence is taken as the target (shifted by one).