Natural Language Understanding with Distributed Representations - Assignment 1

Urjitkumar Patel
up276@nyu.edu

1 CNN Model for Text Classification and Evaluation

As part of this Assignment, using the skeleton code given, I implemented two different architectures for text classification,

1. One Layer CNN Model (less complex , less deeper)

2. Two Layer CNN Model (More complex , Mode deeper)

As our dataset contain only 10.5K samples, There were very high chances of over fitting with more complex model. I observed the same thing during my experiments which I will present in following sections.

1.1 Model Architecture

Here below is the explanation of two setups I used.

- **Simple : One Layer CNN Model**:
  This model contains one convolution layer with Relu as Non linearity layer, one max pooling layer, one dropout layer to avoid over fitting, and it ends with fully connected layer with Softmax to find the probability scores for our two possible labels. In the next section, I will present the experiments I performed with this architecture.

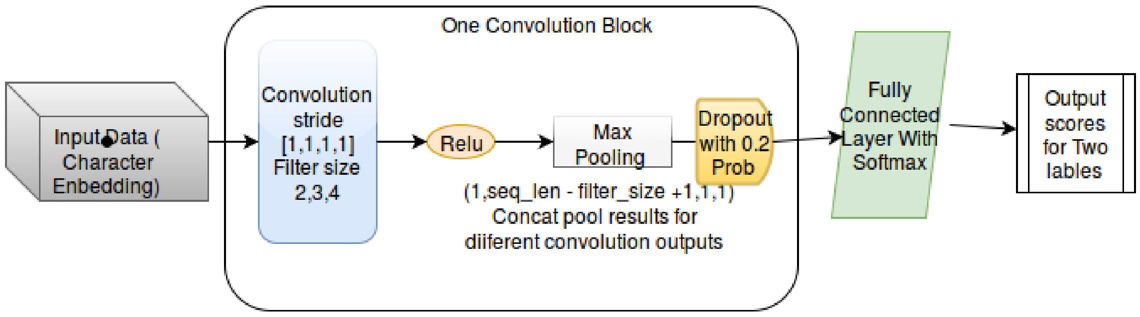
One Convolution Block
Input Data (Character Enbedding)
Convolution stride [1,1,1,1] Filter size 2,3,4
Relu
Max Pooling
Dropout with 0.2 Prob
(1,seq_len - filter_size +1,1,1) Concat pool results for diiferent convolution outputs
Fully Connected Layer With Softmax
Output scores for Two lables

Figure 1: One Layer CNN Model

- **Complex : Two Layer CNN Model**:
  I also implemented more complex CNN having two blocks of Convolution with Relu, Max Pooling, and Dropout. I did different experiments with this architecture too, but due to limitation of Data we have, this complex model failed to give better performance then previous less complex model.

1.2 How did I improve the Accuracy?

- **Baseline**: I started my experiments with Baseline Model as given in the skeleton code. With this very basic model I got 0.72 accuracy. I noticed very high amount of over fitting. Training set accuracy reached to 1.00 very quickly.
- **Changing Non-Linearity from tanh to Relu**: As asked in assignment, I replaced tanh with Relu. Because with the Relu, only about 50% of the activations are non zero, It gives sparse activation. Hence although it makes the computation really fast, It increases the chances of over fitting with less data. At the same time model was not able to achieve accuracy more than 0.69.
- **Adding Dropout**: Adding Dropout was really crucial for this data set and current model structure as we were facing really high amount of over fitting. I started with 0.5 as keep probability for dropout, and noticed really big impact on the performance. I achieved 0.73 accuracy with less over fitting. I continued trying different values for dropout parameter and noticed continuous increase in accuracy as I decreased the value of this parameter from 0.5 to 0.2.
- **Playing with Other Hyper Parameters**: As a next step, with Relu and Dropout, I tried to tune other hyper parameters such as Filter size, number of filters, Regularization parameter, learning rate and embedding dimensions. Due to less data, I tried to make the model robust using high regularization value. I noticed that, with the increment in learning rate, I was able to get the relative best results very quickly with tuned Regularization and Dropout.
- **Initializing weights with variable scalling initializer**: I used variable scalling initializer over xavier initializer to initialize the weights, It works relatively well with Relu. It keeps the scale of the input variance constant, so it does not explode or diminish by reaching the final layer.
- **Adding one more layer of convolution**: I was really skeptical about the performance of deep networks due to the limitation of the data. Though I added one block of convolution with relu, pooling and dropout to see the results. I changed the max pooling kernel size in first layer to get more activation to perform second layer convolution. I also did experiments with hyper parameters, though failed to achieve better performance then single convolution layer. Because of the depth, it takes time to reach the maximum accuracy point with low dropout keep probability and learning rate.

Here Below is the parameters which worked best for me for two setups:

parameter	**Baseline**	**One Layer CNN**	**Two Layer CNN**
Batch Size	64	64	64
Embedding dimensions	64	64	64
Filter1 Sizes	3 4 5	2 3 4	4 5 6
Filter2 Sizes	-	-	2 3 4
No of layer1 Filters	64	128	128
N0 of layer2 Filters	-	-	64
Non-linearity	tanh	Relu	Relu
Drop Out Keep Prob	-	0.2	0.3
L2 Reg. LAMBDA	0	3	3
learning rate	1	0.005	0.005
no.of epochs	50	50	50

1.3 Results

Here below is the best results I achieved with two setups I described,

Network	**Dev Accuracy**
Baseline	aprox $\tilde{0}.72$
One Layer CNN	aprox **0.76**
Two Layer CNN	aprox $\tilde{0}.70$

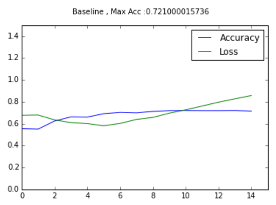

Figure 2: Baseline

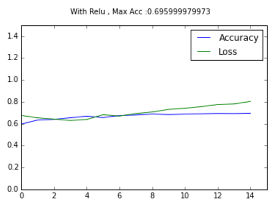

Figure 3: With Relu

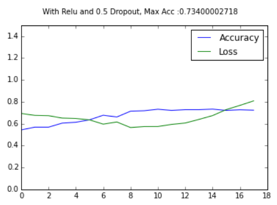

Figure 4: with Relu + Dropout

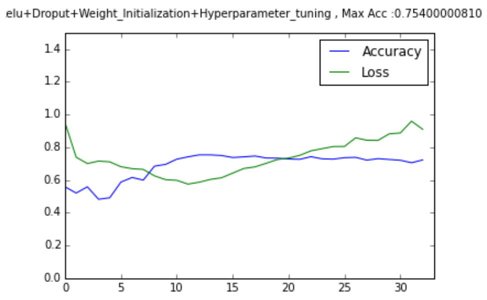

Figure 5: One layer CNN with Best settings

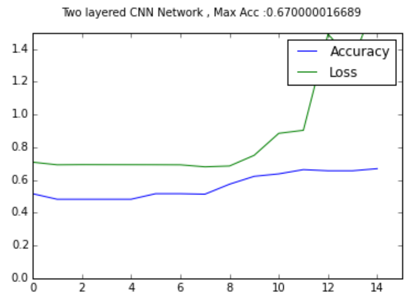

Figure 6: Two Layer CNN with Best settings

- **Baseline Improvement**:
  We can see that with Relu, accuracy downgrades from baseline but by adding Dropout, It got improved by considerable value.
- **One layer Vs Two layer CNN**:
  As we can see in figure 5, For the one layer convolution model, By tuning the hyper parameters such as learning rate and with weight initialization, we can achieve the maximum accuracy very quickly. On the other hand, in figure 6, we can see that, two layer CNN model was not able to achieve as much accuracy as one layer CNN model.

1.4 Conclusion

In conclusion, data set we had was not sufficient to train the deep neural network. Over fitting was the main issue with the baseline model. I observe that **drop out played crucial role to prevent over fitting**. **Learning rate** played crucial role for speed up the processing. **Simple Model with Hyper parameter tuning and proper weight initialization works best for the given data**. Though, with large data sets, deep network can learn much better, for the given data set, I observed **Adding second layer increased the unwanted complexity**, both computational and time.

2 References:

1. Kim, Yoon. ”Convolutional neural networks for sentence classification.” arXiv preprint arXiv:1408.5882 (2014).
2. NitishSrivastava,GeoffreyHinton,AlexKrizhevsky,IlyaSutskever,andRuslanSalakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):19291958, 2014.