11-747 Assignment 1 Report

Introduction

The model I used was a straightforward re-implementation of the baseline with batch normalization. I used PyTorch for this implementation and used batch size 50 (as in the paper). For the hyperparameters, I used similar values that the paper reports (with filter sizes 3, 4, 5 and 100 kernels). I did not apply any preprocessing apart from making everything lowercase. I used pretrained word2vec [2] embeddings (in a non-static way).

Model Description

This section is mostly a recap of [1]. I used a CNN model with a sentence represented as a concatenation of the word vectors passed through word embeddings. I masked the word vectors for sentences shorter than the longest sentence in the batch. I used the ReLU nonlinearity instead of tanh as mentioned in the paper.
I applied three filters for convolution of sizes 3, 4, 5, passed them through a ReLU followed by batch normalization, and then finally a 1D max pooling layer.
For normalization, I used batch normalization after the convolutional layer and before the max pool layer. I also had a dropout right before the projection layer with probability 0.5, as in the paper.

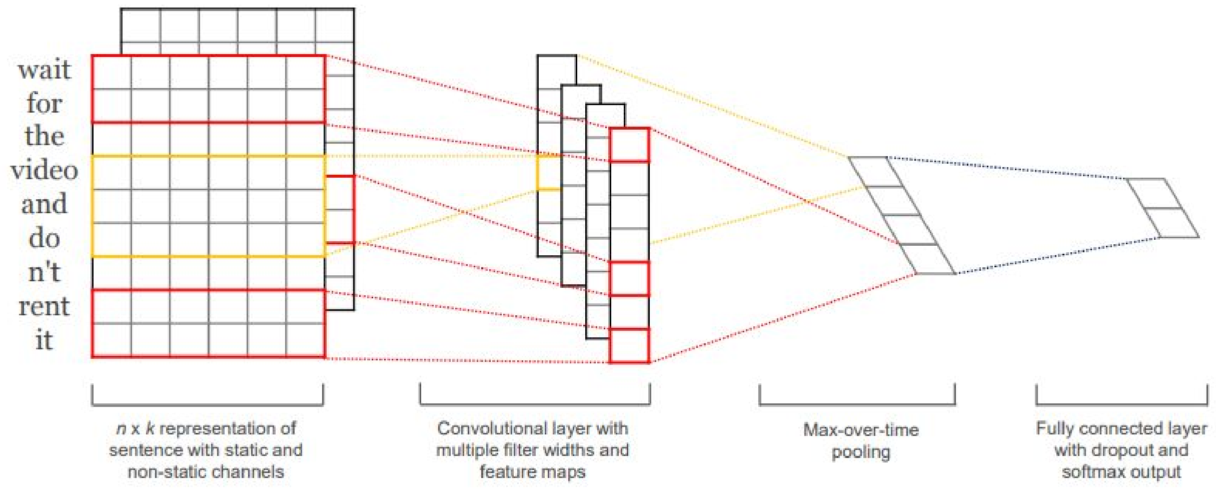

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

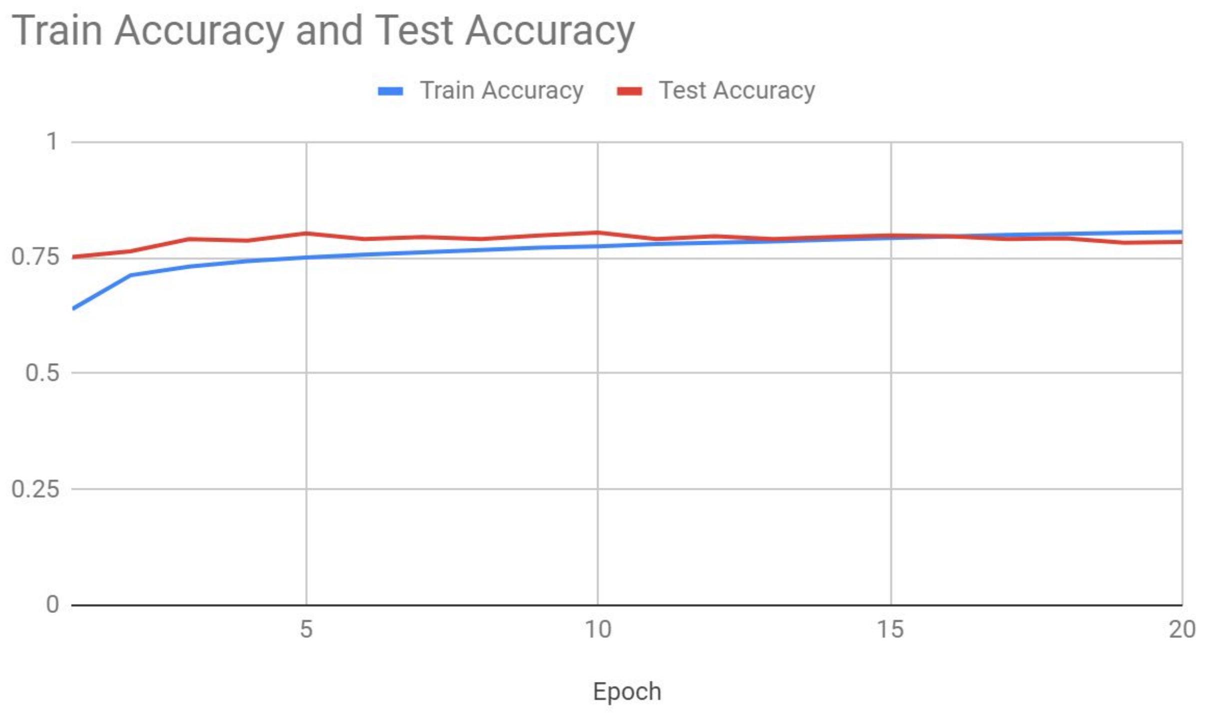
Figure 1: Model architecture with two channels for an example sentence.

Source: Yoon Kim [1]

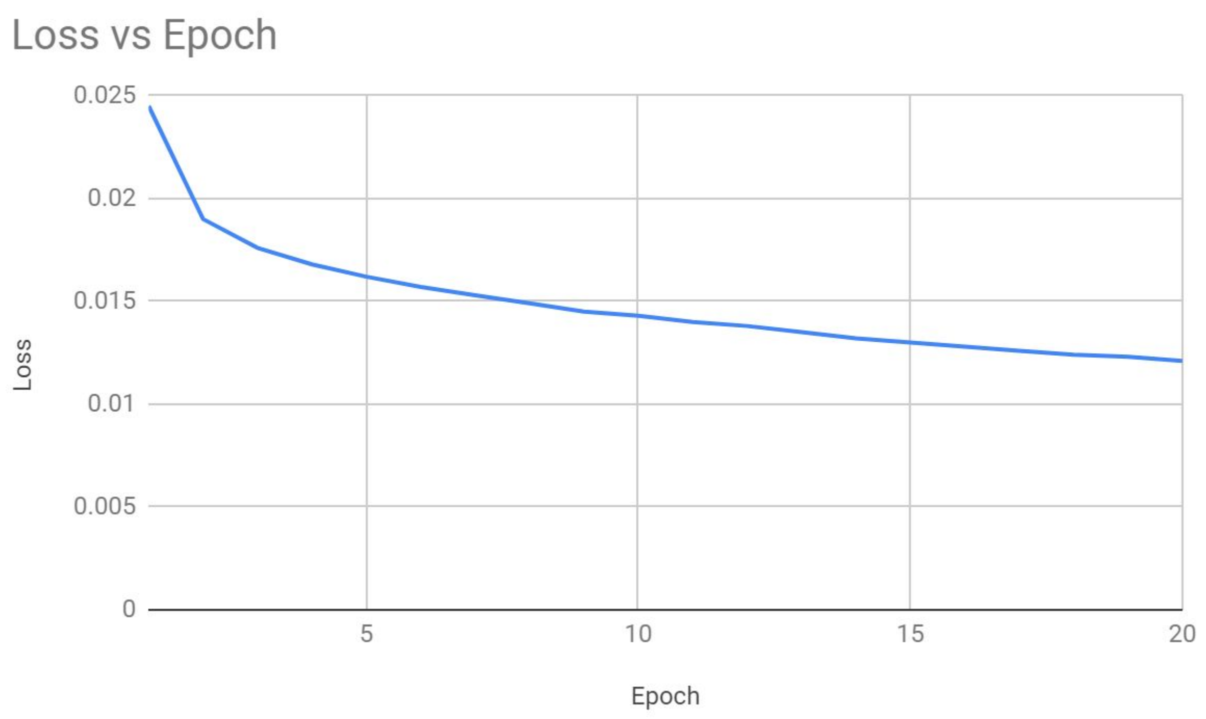
Result and Analysis

From the results, we can infer that the capacity of the model is too high for the given dataset. The model started overfitting in the second epoch, with the training loss continuing to drop until it reached almost 0. To prevent this, I used some tips from Piazza and reduced the capacity of the model. I used only 50 filters, since using 100 filters was giving accuracies around 65%. I also changed the optimizer from Adam to SGD and slowed the learning rate to 0.01. Using word2vec weights for pretraining increased accuracies as well (I did not keep track of the exact numbers). Using batch normalization also helped the training accuracies increase by around 1-2%.

The graph for accuracy for a 20 epoch run are shown below:



As you can see, the test accuracy quickly reaches a high value and starts overfitting around 80%.



The loss continues decreasing even with the test accuracy reaching its peak, signifying that the model is overfitting the data and the capacity of the model is too high.

The reported results file is from a run which ended with 80.9% accuracy on epoch 10 on the dev set.

[1] https://www.aclweb.org/anthology/D14-1181
[2]
https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf