Natural Language Processing
Assignment 2 Report

Tsinghua University

Albert Millan – 2019280366

Contents

1 Introduction

This report explains the methodology involved to deploy text classification model employing convolutional neural networks (CNN) in Pytorch.

2 Implementation

2.1 Pre-processing

The provided data was converted into a *torchtext* tabular dataset, removing all the labels preceding each of the words, formatting sentences into strings of words, and storing the sentence labels. The word embeddings from the training and validation dataset were included, notwithstanding, those of the words that appear in the test dataset but do not appear in the former datasets were disregarded. This is to ensure the integrity of the test-set. Such words were then classified as unknown tokens, and had a specific identifier assigned to them by torchtext. The data was grouped in batches, ordered according to the sentence length.

2.2 CNN Model

Am implementation using a convolutional network architecture for sentence classification [1] was deployed to classify sentiment from sentences. The architecture is defined in figure 1. It includes a convolutional layer, which performs convolutions with a kernel of height of a couple of words in the sentence (3-5), and width k, the length of the word embeddings. For each of the generated filters, it performs a max-pooling operation across the entire filter, retrieving the feature that yields the largest value. That is, identify the feature from the sentence convolutions that is most meaningful in the prediction of sentiment. The output of this convolutional layer block is then sent to a fully-connected layer group, which contains the dropout layer. Cross-entropy is then performed to define the loss and identify the prediction.

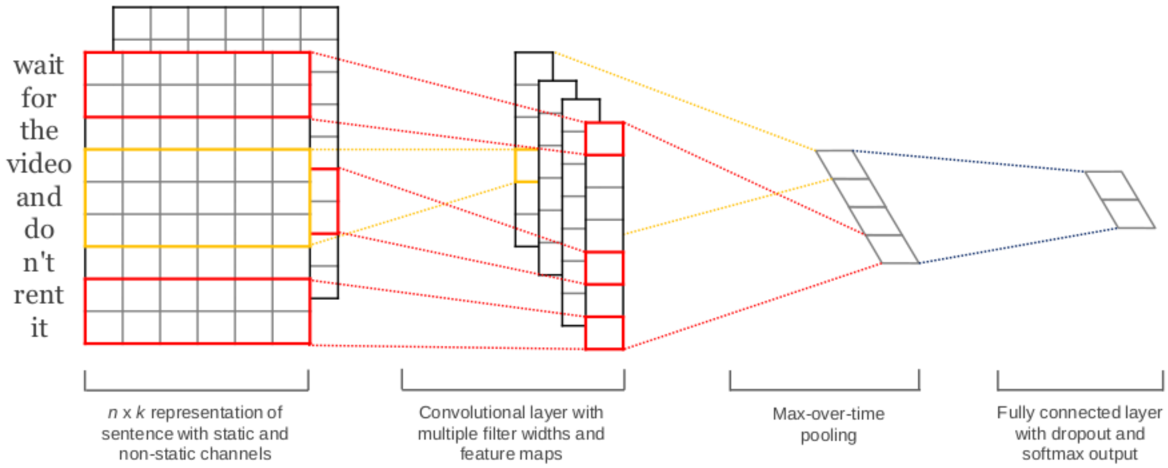

Figure 1: Model architecture for Sentence Sentiment Classification.

2.3 Results

Table 1 outline the conditions under which the experiments were conducted, describing the hyper-parameters and results obtained. Those models trained with dropout layer yielded a lower score than those that did not use dropout. Using a larger hidden layer size (512) yielded an improvement of 1% in accuracy over the smaller hideen layer configuration (256). Pre-trained Glove and Word2Vec embeddings from the first assignment were tested on the model. Glove embeddings yield superior performance, highlighting the importance of fine-tuned word-embeddings.

Emb.	Hidden S.	Kernel	Dropout	Iters	Batch	Lrn. R.	Test Acc (%)
Glove	(256)	(3x300)	0.0	120	6	3e-3	43.21
Glove	(256)	(3x300)	0.2	120	6	3e-3	40.99
Glove	(256)	(4x300)	0.0	120	6	3e-3	41.27
Glove	(512)	(3x300)	0.0	120	6	3e-3	44.39
Glove	(512)	(3x300)	0.2	120	6	3e-3	44.02
Word2Vec	(256)	(3x300)	0.0	120	6	3e-3	39.67

Table 1: Hyperparameters employed to train the convolutional model.

The loss and accuracy values across iterations are shown in figure 2. Consistent results are observed during for the training loss and accuracy, with the models using dropout above those that don't in terms of loss and vice-versa in terms of accuracy. Increasing the kernel size (5) did not yield better performance than the original model (3). The loss and accuracy plots on the validation set shown in figure 3 indicate the model overfits if trained until convergence, suggesting a dissimilarity between the training and test data. Best model is therefore not found at its convergence point, but rather at the iteration it yields the best accuracy for the validation set. In this setting, using a hidden convolutional layer of size 512 with no dropout yields the best performance.

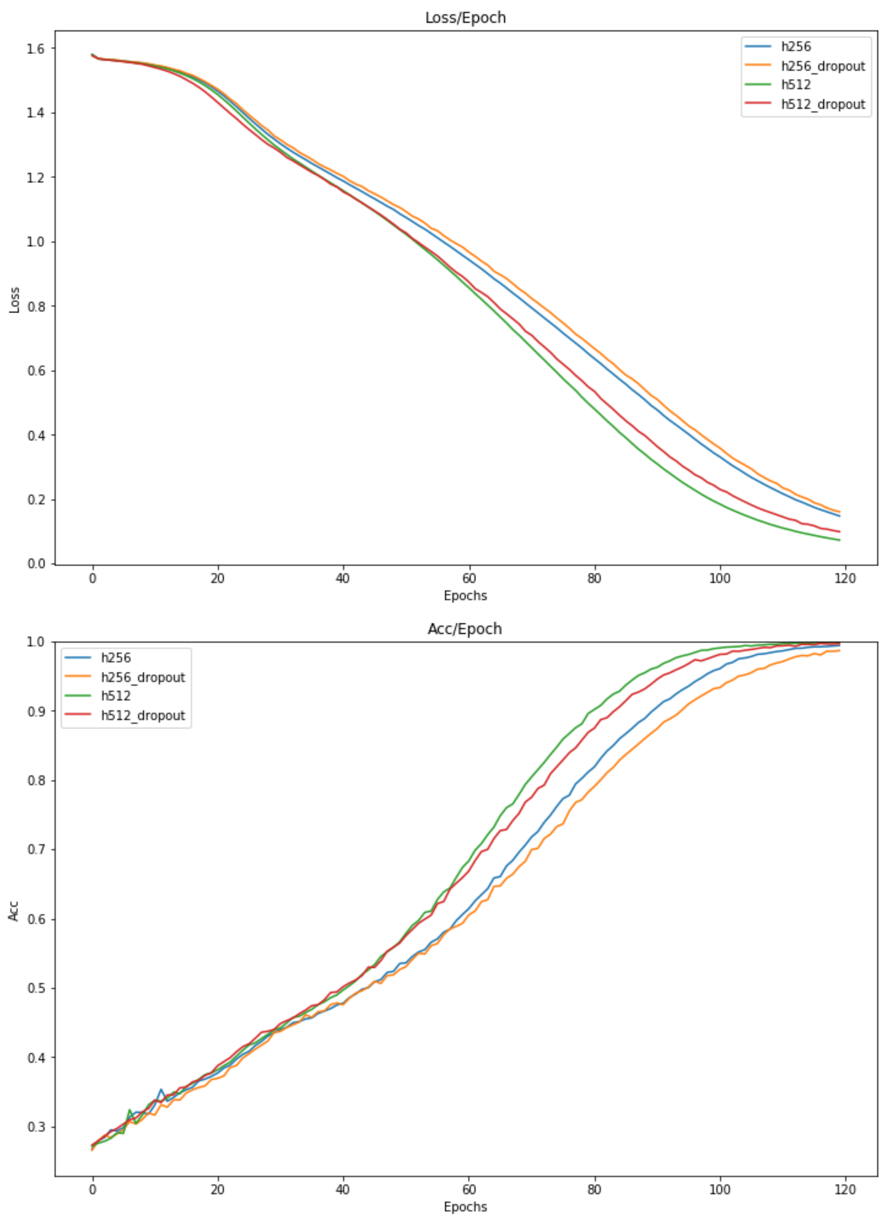

Figure 2: Training Loss and accuracy plots for the model with hidden size of 256 or 512, and with or without dropout.

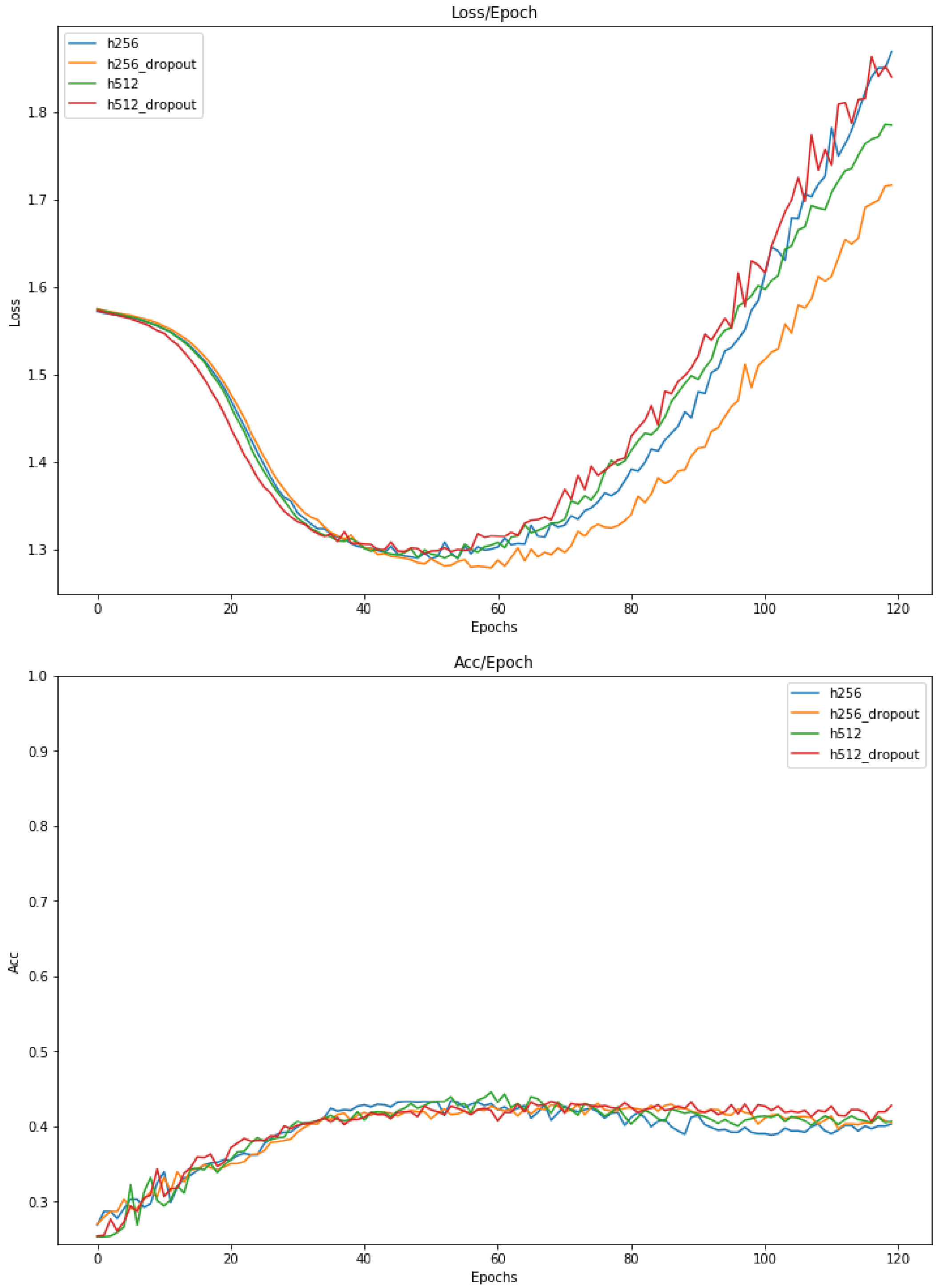

Figure 3: Validation Loss and accuracy plots for the model with hidden size of 256 or 512, and with or without dropout.

References

[1] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014.