CNN for Text Classification NN4NLP 2020

Author: Brian Yan

Introduction:

The task of this model is to classify sentences into 16 different genres, ranging from video games to music.

The approach is to use a convolution layer to extract n-gram features of varying lengths. These features are then passed through a fully connected layer. Altogether, this CNN has 2 layers and achieves 88% accuracy on the validation set.

Individual CNN models achieved between 84-86% accuracy with optimal early stopping. If several of these models are ensembled together using a majority voting scheme, then an additional 2-4% boost is achieved.

Model:

The CNN architecture is implemented following the baseline provided by Yoon Kim's paper, CNN for Sentence Classification. As such, this report will not enumerate all the mathematical operations but rather mention the components of the CNN architecture and focus on the hyperparameter controller implementation.

The Baseline CNN

The inputs to the model are sentences, encoded as a sequence of word embeddings. FastText provides pre-trained embeddings on Wikipedia and news articles. These embeddings are 1 x 300 in size. The choice to use pre-trained embeddings affords greater generalization than embeddings trained on the smaller corpus of our task. Embeddings that were trained on the corpus alone, through unsupervised CBOW, achieved far lower performance in the classification task. In the classification task, the word embeddings were kept static.

Sentences are padded to be of equal length within a batch. Zero-padding is used. Then, these sentences are passed through convolutions with varying kernel sizes. These filter operations have padding of 2, stride of 1, and dilation of 1; thus they only vary in the size of the kernel. Batch normalization is applied to these convolutions and then ReLu activation. The outputs of the different convolutions are concatenated and passed through a max-pooling operation.

Between this convolution layer and the fully-connected layer is a dropout regularization, which randomly zeros out weights at 20% chance. Finally, the projection layer outputs in the dimension of 1 x 16, the number of classes. In training, loss is defined as softmax cross-entropy.

The Hyperparameter Controller (Figure 1)

The convolution and dropout portions of the CNN model are defined through random choices within a defined search space in order to generate a set of models to ensemble together. The intuition is that models with varying hyperparameters will make mistakes in different areas of the prediction task. The less correlated the predictions are, the more useful ensembling via majority vote will be.

The hyperparameter search space must be defined for the controller, engendering some inductive bias. The baseline paper used 100 filters per filter type and during experimentations of this task, a greater number of filters often resulted in greater accuracy. Thus filter numbers were chosen with a uniform distribution between 150 and 400.

For kernel sizes, the baseline paper used 3, 4, and 5 but experimentation with smaller sizes also showed positive results. As such, the lower bound for filter sizes started at 1. The number of filter types is chosen from a uniform distribution between 1 and 4. A starting filter size is chosen from 1 to 3, and then each additional filter type can have 0-4 more than the previous. This allows the controller to select configurations where some filter sizes are double or triple the number than other sizes. Further, the range of filter sizes wide (with a max of 15).

Dropout is chosen as either a 20% rate or 0%. During experimentations, models overfit quickly (after 2-3 epochs) even with a higher dropout rate of 50%. Ultimately, dropout was not a significant booster of accuracy.

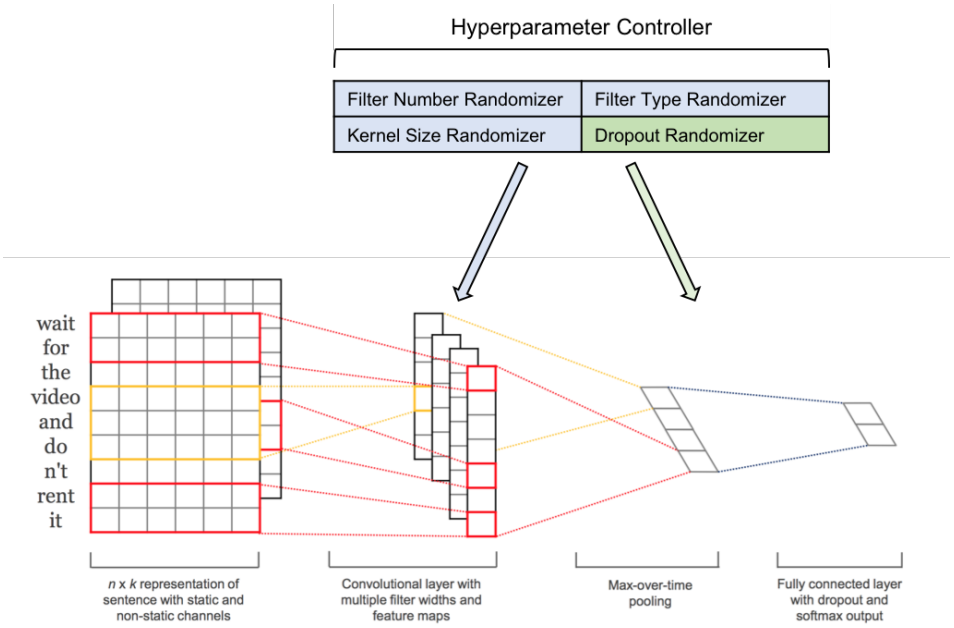

Figure 1: Hyperparameter controller which randomly chooses the architecture of the convolution layer. Also chooses the dropout rate after max pooling.

Ensemble Method:

Training Ensemble Groups

Training is done through the hyperparameter controller which generates models. These models are trained against an accuracy threshold. 84% was chosen based on the baseline accuracy of individual models. An important implementation piece to successfully train many models is an early stopping procedure. This task often overfit to the training data in 2-3 epochs and began to increase in validation error and loss quickly afterwards. The early stopping rule is to end training after an epoch that decreases in accuracy compared to the previous epoch. Then the parameters from that previous epoch are chosen.

Predictions via Majority Voting

In order to ensemble the various models, a majority voting scheme was used. This simply counts the predictions for each class and chooses the class with the greatest number of votes.

Results:

The incremental contribution of this project over the baseline is the accuracy boost of 2-4% when using ensembling. The table below shows 14 individual models which range in accuracy from 84-86%. The final row is the accuracy from the majority vote of these models. All models have an equal weight in the voting. All accuracies are on the validation set.

Model No.	Accuracy	N_FILTERS	WIN_SIZES	DROP?
1	85.69%	327	6, 6	F
2	85.53%	224	4	T
3	84.60%	188	3, 3, 4, 5	F
4	85.22%	371	3	T
5	85.06%	319	3, 5, 6	T
6	86.47%	170	2, 3, 6, 8	F
7	84.44%	396	1, 4, 6	F
8	85.23%	347	4, 5	F
9	84.45%	342	2, 5	T
10	84.60%	224	2, 4, 5	F
11	85.07%	298	4, 4	F
12	84.76%	382	4, 6, 6	T
13	85.38%	217	4, 4	F
Ensembled	**88.02%**	**-**	**-**	**-**

Figure 2: Validation accuracies by model, varying in number of filters, kernel sizes, and dropout. Ensembled method is a majority vote of the 13 models.

References:

1. Yoon Kim, Convolutional Neural Networks for Sentence Classification. https://arxiv.org/pdf/1408.5882.pdf

2. Graham Neubig, CNN for Text in Pytorch. https://github.com/neubig/nn4nlp-code/tree/master/05-cnn-pytorch

3. FastText, Word Embeddings. https://fasttext.cc/