# **Convolutional Neural Networks for Text Classification**

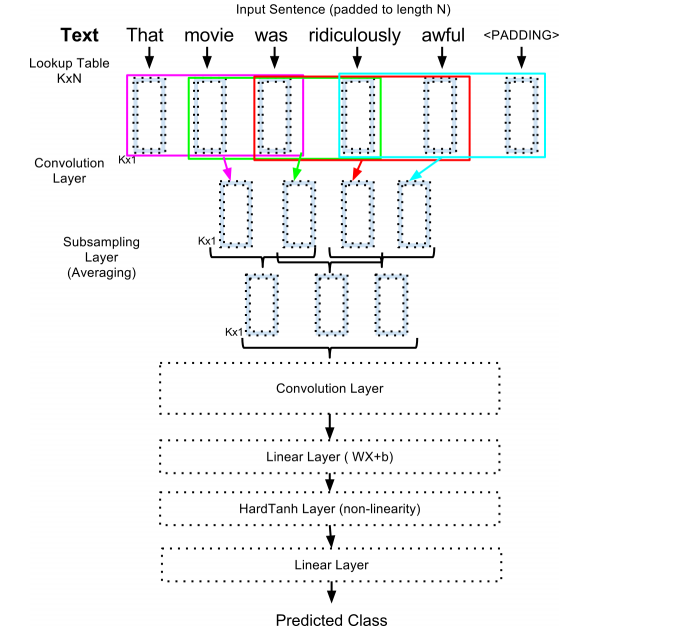
**Indhujha Natarajan**

## Introduction

Convolutional Neural Networks (CNN) were initially used in the field of Computer Vision to classify images. It has now been proven that CNN can also be used in Text Classification. Text classification can be in the form of classifying the text into categories such as the source of the text or classifying if a sentence is subjective or objective. A growing area of text classification is in the field of sentiment analytics where the intent is to classify a sentence as one with positive emotion or negative emotion. An example of the use case of such an application is movie reviews. The sentences in various movie reviews can be given as input to a CNN in order to obtain an output as a positive or negative review. Another type of sentiment analysis is also called sentence polarity classification – classifying a sentence as positive or negative based on the emotion being conveyed.

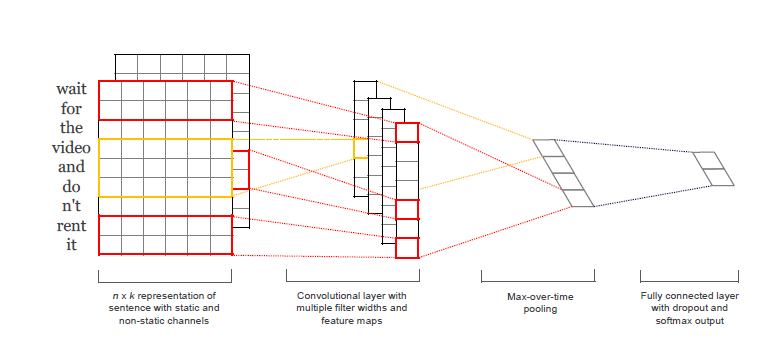
## Related Work

In Soumith Chintala’s (2011) paper, he talks about performing sentiment analysis using the back propagation algorithm. The approach followed by him is summarized in the process flow diagram shown in Figure 1. The training was done using stochastic gradient descent. The analysis of Soumith Chintala (2011) uses a lookup table which uses a weight matrix of size K x N where N is the length of the input sentence and K is the per word feature vector size. This feature vector, called an embedding, is initialized randomly after which back propagation is run to learn the actual values. If these embeddings are generated from a large corpus of text then it serves as a good starting point that is closer to the global minimum thereby improving the performance of the neural network.



*Figure 1.* Architecture of the network. From “Sentiment Analysis using neural architectures” by S. Chintala (2011), p. 2.

Yoon Kim (2014) performs polarity classification of a sentence using a simple CNN that gives good results with little hyperparameter tuning. Yoon Kim (2014) tries different types of experimental setups such as detecting positive/negative reviews, classifying sentences based on polarity – 5 output levels (positive, strong positive, neutral, negative, strong negative) or 2 output levels (positive, negative) and others. He also experiments with various model variations such as with or without pre-trained vectors and various number of channels. Yoon Kim proposes a slight variation to the CNN architecture of Collobert et al. (2011) in the form of the model architecture in Figure 2.



*Figure 2.* Model architecture with two channels for an example sentence. From “Convolutional Neural Networks for Sentence Classification” by Y. Kim (2014), p. 2.

## Dataset Used and Goal of Study

The dataset considered for my study is the IMDB dataset provided by the keras library. This dataset contains highly polar (good or bad) 25000 movie reviews for training and the same amount again for testing. My aim is to predict whether a given movie review has positive or negative sentiment.

The dataset is available at:

<https://keras.io/datasets/>

The sentences are translated to a sequence of integers. These integers indicate the absolute popularity of the word in the dataset. The y variable or the response variable takes two values – 0 or 1. Thus, it is a binary classification problem looking for good and bad sentiment in the review.

## Implementation

The keras.datasets.imdb.load\_data() function can be used to load the dataset in a format that is ready to be used for neural networks. An important concept that is used for performing natural language processing is called embedding.

**Embedding:**

This is a technique where words are encoded as real-valued vectors in a high-dimensional space, where the similarity between words in terms of meaning translates to closeness in the vector space (J. Brownlee, 2016). This technique maps discrete vectors to continuous numbers thereby quantifying closeness information. This quantitative information needs to be provided as input for neural networks.

To decide the word embedding representation of the IMDB dataset, the limits need to be decided as explained as follows. The dataset is limited to the top 5000 most used words in the dataset. The maximum length of the review is taken as 500 words by truncating longer reviews or padding the shorter ones. A 32 dimension vector is used to represent each word. The embedding function is used to create the embedding.

*Syntax: Embedding(input\_dim, output\_dim, input\_length=None)*

*Embedding for our dataset: Embedding(5000, 32, input\_length=500)*

After this, important packages from the Keras library are imported. Then I follow two approaches with respect to the modeling.

## Simple Neural Network Model

A 50%/50% split between test and training is used similar to how it was used in Maas A. et al. (2011), who are the creators of the dataset.

*Figure 3.* Architecture of Simple Neural Network Model.

Figure 3 shows the architecture of the simple neural network. The embedding layer produces an output matrix with dimensions 32 x 500. This output is sent to the flatten layer to generate the one dimensional vector that is of length 32 x 500 = 16000. This one dimensional vector is given as input to the hidden layer which has 250 nodes with a rectifier activation function. The output of the hidden layer is given to the Output layer that uses a sigmoid activation function.

## Convolutional Neural Network (CNN) Model

I use the same test and train datasets for this model too. I use one-dimensional convolutions and max-pooling in this model. This means convolutions are done in one-direction to learn 3 vector elements of the word embedding at a time.

*Figure 4.* Architecture of Convolutional Neural Network Model.

Figure 4 shows the architecture of Convolutional Neural Network Model. I used three one dimensional Convolutional layers with varying number of filters and kernel sizes. Then I use a one-dimensional max pooling layer of size 2 and stride 2 that halves the size of the feature maps from the convolutional layer. The output of the Maxpool layer is flattened to a one dimensional vector which is given to dense layer with 250 nodes and the rectifier activation function. The output of this dense layer is given to the output layer with the sigmoid activation function.

## Results

The results of the two models are compared in Table 1.

Table 1. Comparison of Results of the Two Models – Simple Neural Network and CNN.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Train Loss | Train Accuracy | Validation Loss | Validation Accuracy | Processing Time (seconds) |
| Simple Neural Network Model | 19.13% | 92.72% | 30.06% | 87.32% | 55 |
| Convolutional Neural Network (CNN) Model | 22.51% | 91.17% | 26.58% | 88.96% | 379 |

As highlighted in Table 1, the Accuracy which is obtained with new data (validation accuracy) is 1.64% more for the CNN model compared to the Simple Neural Network model. This increased prediction accuracy translated to ~410 more correctly classified reviews between the two models. But the CNN does take 324 seconds more to process than the Simple Neural Network model.

## A Complex Dataset

Now that we know the basics of text classification using neural networks, I used a more complex dataset to build the two models. This dataset is called the polarity dataset that contains positive and negative sentiment information (Oswal, 2016).

The simple model architecture is given in Figure 5.

*Figure 5.* Simple Neural Network – Polarity Data

The architecture for the convolutional architecture is given in Figure 6.

*Figure 6.* Convolutional Neural Network – Polarity Data

### **Comparison between the Results of the two Models**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Train Loss | Train Accuracy | Validation Loss | Validation Accuracy  (Highest obtained) | Processing Time (seconds) |
| Simple Neural Network Model | 53.65% | 61.81% | 74.21% | 55.17% | ~10 mins per Epoch |
| Convolutional Neural Network (CNN) Model | 58.92% | 60.28% | 68.71% | 51.89% | ~24 mins per Epoch |

The data is processed through a helper program to first convert the sentences to integers. After this the embedding layer is created which is used for further processing as shown in Figures 5 and 6. For both the models, I use 5 epochs and a batch size of 30. The bigger gap between train and test accuracy (8.39%) vs that for Simple NN (6.64%) means that the CNN overfits the data.

Increasing the number of epochs and using pre-trained embeddings may improve the performance of each of the models. Comparing the models with these modified settings may yield different results.

## References

Chintala, S. (2011). Sentiment Analysis using neural architectures, 1–4.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. https://doi.org/10.3115/v1/D14-1181

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (n.d.). Learning Word Vectors for Sentiment Analysis. Retrieved from http://ai.stanford.edu/~amaas/papers/wvSent\_acl2011.pdf

Jason, B. (2016). Predict Sentiment From Movie Reviews Using Deep Learning. Retrieved June 17, 2018, from https://machinelearningmastery.com/predict-sentiment-movie-reviews-using-deep-learning/

Oswal, B. (2016). CNN-text-classification-keras, Github, Github Repository. Retrieved June 17, 2018, from https://github.com/bhaveshoswal/CNN-text-classification-keras