A Deep Learning Architecture for Sentiment Analysis

# A Deep Learning Architecture for Sentiment Analysis

Erion Çano
Politecnico di Torino
Duca degli Abruzzi, 24, Torino, Italy
+393478353047
erion.cano@polito.it

Maurizio Morisio
Politecnico di Torino
Duca degli Abruzzi, 24, Torino, Italy
+390110907033
maurizio.morisio@polito.it

## ABSTRACT
The fabulous results of Deep Convolution Neural Networks in computer vision and image analysis have recently attracted considerable attention from researchers of other application domains as well. In this paper we present NgramCNN, a neural network architecture we designed for sentiment analysis of long text documents. It uses pretrained word embeddings for dense feature representation and a very simple single-layer classifier. The complexity is encapsulated in feature extraction and selection parts that benefit from the effectiveness of convolution and pooling layers. For evaluation we utilized different kinds of emotional text datasets and achieved an accuracy of 91.2 % accuracy on the popular IMDB movie reviews. NgramCNN is more accurate than similar shallow convolution networks or deeper recurrent networks that were used as baselines. In the future, we intent to generalize the architecture for state of the art results in sentiment analysis of variable-length texts.

## CCS Concepts
**• Computing methodologies → Artificial intelligence → Natural language processing • Applied Computing → Document management and text processing**

## Keywords
Text-based Sentiment Analysis; Convolution Neural Networks; Deep Learning Architectures;

## 1. INTRODUCTION
Deep Learning has been recently the buzzword of top performance in problems of various domains like computer vision, speech recognition or sentiment polarity analysis. Utilizing deep neural networks in different application types requires limited domain knowledge and yet produces wonderful results. Moreover, performance does usually scale well with increasing data and computation capabilities, whilst it is also possible to tune it with hyper-parameter variations, specific to the application. Among the various network types, Convolution Neural Networks (CNN) have been particularly successful in applications related to image analysis. They were first used about 20 years ago by LeCun *et al.* in [1] to recognize handwritten digits. That basic CNN structure has been used to form a myriad of highly advanced neural architectures that have produced breakthrough results in the yearly ImageNet challenge [2]. Architectures like AlexNet [3], Inception [4], VGG-19 [5] and others have proved very successful in correctly recognizing images from thousand categories. They are diverse in terms of complexity, parameters and effectiveness (as described in [6]) but their fundamental logic is the same: using generic deep features in combination with a simple classifier. It is interesting to see that their learnt representations prove to be very good competitors in even more specific visual recognition tasks than the one each of them was derived from. Authors in [7] provide more evidence about that, supporting the idea that generic descriptors extracted from CNNs are very powerful. This renowned reputation of CNNs in computer vision, attracted researchers and practitioners of other application domains as well. Kim in [8] for example, used basic CNNs for emotional recognition in sentences, reporting excellent results in various datasets. Similar results were reported by Kalchbrenner et. al. in [9] where they also introduce k-max pooling operation for better feature selection. Other works like [10] or [11] analyze emotionality of sentences or short texts by combining CNNs with Recurrent or Recursive Neural Networks (RNN) that have also been highly successful, especially in representing sequential data. The most popular RNNs are Long Short-Term Memory (LSTM) networks that utilize feedback loops as "memory" to capture information about what has been seen so far [12]. This allows them to represent sequential data like text (sequence of words). The problem with these networks is that in practice they look back only a few steps and thus are not good in representing long text documents. Furthermore, RNNs are slower to train compared with CNNs that are simpler and faster.

In this paper we present NgramCNN, a deep neural architecture that takes advantage of CNN speed and effectiveness to extract salient features out of n-grams in various text types. We followed same basic paradigm that have been successful in image analysis: complexity in features and simplicity in classifier. For text feature representation we use GoogleNews[1] pretrained word embeddings which offer excellent generalization (usable across different text types) and highly reduced dimensionality. The complexity is packaged in the repeated convolution and pooling layers that are responsible for feature extraction and selection. Instead of applying global or k-max pooling on entire text document, we apply regional max-pooling on text regions and use the aggregate feature maps for classification. At the end, a single-layer classifier predicts the emotional category of each document. The architecture is flexible and extensible both horizontally and vertically. More convolutions of 4-grams or longer can be concatenated and stacked if bigger datasets are available. To validate the effectiveness of NgramCNN we experimented with


*Conference'10*, Month 1–2, 2010, City, State, Country.

DOI: http://dx.doi.org/10.1145/12345.67890

[1]https://code.google.com/p/word2vec/

linear as well as shallow CNN or RNN baseline models on 3 datasets of song lyrics, movie reviews and smartphone reviews. The results confirm the superiority of NgramCNN both in prediction accuracy and training time. The rest of the paper is organized as follows: Section 2 describes word representation, feature extraction and classification layers of NgramCNN architecture. Section 3 presents the 3 experimental datasets, text preprocessing steps that were applied upon them, and the baseline models. Section 4 uncovers the results and various hyper-parameter choices that we made. Finally, Section 5 concludes and presents possible future work directions.

# 2. NgramCNN ARCHITECTURE

## 2.1 Word Representation

Bag-of-words is the traditional method for extracting text features that are used for classification. It creates a vector representation of each document based on the vocabulary of entire text set and various scoring methods (e.g., binary, count, tf-idf, etc.). Each document is hence V units long and the entire matrix becomes N x V units; here V is the length of the entire vocabulary whereas N is the number of documents. This representation gives very good results on different text classification tasks, especially when combined with tf-idf scoring and SVC classifier [13]. However sparsity and dimensionality become problematic when vocabulary is high. Actually, neural networks don't work well with sparse data representations of very high dimensionality. The introduction of word embeddings as dense and low dimensional word feature representations was thus revolutionary [14]. Each word is represented by a significantly smaller distributed feature vector (say 300 dimensions) that is able to capture syntactic and semantic similarities of that word with respect to the other words of the vocabulary. A choice to make here is whether to use dynamic word vectors trained from the available experimental text set, or static vectors obtained from pretrained bundles. In [15] we performed several experiments on this issue and concluded that when small text sets are available (as in our case here), sourcing word vectors trained from big text corpora gives better results. The pretrained word vectors behave like generic feature extractors of text that can be utilized across different datasets and tasks. For this reason, in this work we use static word vectors of 300 dimensions indexed from GoogleNews corpus. They were trained from a 100-billion tokens bundle and successfully used in various similar studies [16, 8].

## 2.2 Feature Extraction Layers

The complexity is packaged in feature processing part as shown in Figure 1. Here we use convolutions of different kernel sizes to capture patterns of words (e.g., "nice", "bad", etc.), bigrams (e.g., "I like", "not good" etc.), trigrams (e.g., "I like that", "that was terrible", etc.) or even longer n-grams and their relation with text categories. We used Rectified Linear Unit (*Relu(x) = max(0, x)*) activation function and got fifteen feature maps of different sizes out of each convolution. To retain local positional information of word combinations, pooling operation with pool size four follows after each convolution. Suppose the output of a convolution is a feature map $f = [f_1, f_2, \ldots, f_n]$. Than pooling process is applied to regions of four consecutive features $(f_1 - f_4, f_5 - f_8, \ldots, f_{n-3} - f_n)$ selecting the maximal of each region which is assumed to be the best representative of that region. The role of pooling is to downsample (here by four) data and extract the most salient features. As reported in [17], 1-max pooling was found to be the optimal choice. Same process (convolution-pooling) is repeated two (for smartphone reviews) or three (for lyrics and movie reviews) times, refining feature quality and reducing their size. At the end, the resulting feature maps of the parallel branches are concatenated and flattened to be useable for the classification layer. It is important to note that the architecture is flexible and extendable. It can be extended horizontally (in width) by adding convolutions of longer word combinations in cases when longer documents are analyzed. It may also be extended vertically by stacking more convolution-pooling blocks, especially if bigger datasets are being worked with.

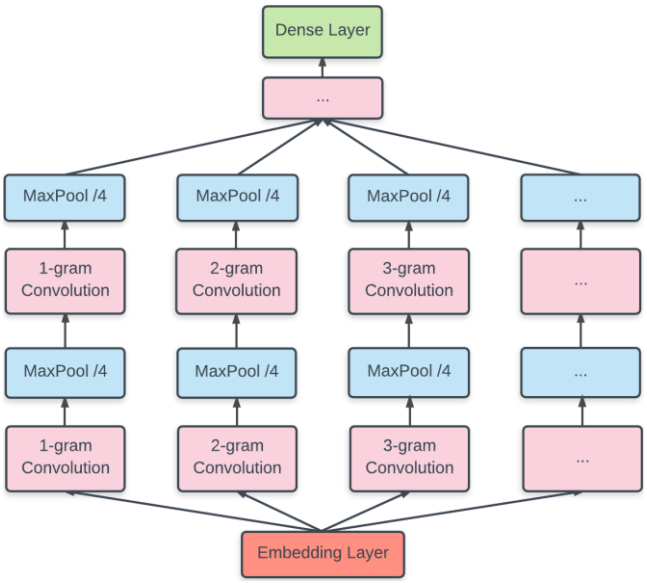


**Figure 1. NgramCNN architecture**

## 2.3 Classification Layer

The classifier we used is very simple. It consists of a dense layer of 100 units and L2 regularization with 0.09 weight, followed by the output layer. To avoid overfitting we also used dropout of 0.5 between the dense and output layers. Relu and Sigmoid were used as activation functions of those layers respectively. We also applied binary crossentropy to compute the loss and Adam method for optimization. In Section 4 we provide more details and explanations about the hyper-parameter choices of the entire architecture for each experiment.

# 3. EXPERIMENTAL SETUP

For the evaluation we chose text datasets of different content. Same data cleaning and preprocessing procedure was followed for the 3 of them. As baseline models we utilized traditional linear models, shallow CNN and RNN neural networks as well as deeper combinations of CNNs with RNNs.

## 3.1 Datasets

**Song Lyrics** MoodyLyrics is a dataset of 2,596 English song lyrics labeled as 'happy', 'angry', 'sad' or 'relaxed' [18]. The task here is to automatically predict the emotional category of each song text. To comply with the other tasks (binary classification) we use MLPN, a similar dataset of 2,500 positive and 2,500 negative song lyrics constructed from Last.fm user tags and a systematic process described in [19]. Both datasets can be freely downloaded from http://softeng.polito.it/erion/.

**Movie Reviews** IMDB movie review dataset [20] is a ground-truth collection of 50K movie review texts, very popular in sentiment analysis studies. The goal is to determine if each movie

review is positive or negative. This task in its basic form (sentiment analysis of item reviews) has high commercial interest as it is a basic block of advertising engines.

**Phone Reviews** Unlocked Mobile Phone reviews is a collection of user reviews about Amazon smartphones of various brands, models, prices, etc. Besides the textual description, users also provide the usual 1-5 stars rating for each phone. We removed entries without a text review or a star rating. Also, 3-star reviews which contain both positive and negative (ambiguous) descriptions were removed, reaching to a total of 232,546 reviews. Finally, 1-star and 2-star reviews were labeled as 'negative' whereas 4-star and 5-star reviews were labeled as 'positive'. Same as in the case of movies, we will utilize this dataset to evaluate the ability of NgramCNN architecture in discriminating between 'positive' and 'negative' texts.

**Table 1. Summarized statistics of each dataset**

| Dataset | No. Texts | Min Length | Avg. Length | Max Length | Used Length |
|---|---|---|---|---|---|
| Song Lyrics | 5K | 23 | 227 | 2733 | 600 |
| Movie Reviews | 50K | 5 | 204 | 2174 | 550 |
| Phone Reviews | 232K | 3 | 47 | 4607 | 150 |

## 3.2 Text Preprocessing

Before training the models we applied some the basic preprocessing over the texts. First we removed all html markup patterns and lowercased everything. We also used a regular expression to collect and keep in the smiley symbol combinations such as :P, :D, :-), :), :(, :-(, etc. that are frequently found in movie or smartphone review texts. Being in excellent conformity with the emotional category of the document they appear in, they represent very salient and helpful features for classification. Regarding stopwords, we removed only a small part of them, namely ['the', 'this', 'that', 'these', 'those', 'a', 'an', 'as', 'of', 'at', 'by', 'for']. These words appear very often but carry very little or no semantic value at all. The rest of English stopwords are mostly tokenization residues of short forms (e.g., 'd', 'll', 'm', 's', 't') or negative auxiliary forms (e.g., 'don', couldn', 'didn', 'hadn') that shouldn't be removed, as their presence or absence can completely shift the emotional polarity of the phrase and thus cripple prediction performance of the model. At the end, junky or numerical patterns were removed as well. We observed length distribution of documents for each dataset. Summary of statistics is presented in Table 1. In the case of song lyrics, lengths range from 23 to 2733 with an average of 227. IMDB movie reviews range from 5 to 2174 averaging to 204 tokens. Smartphone review lengths are even more dispersed, ranging from 3 to 4607 with an average of 47. It is important to note that most of documents are short and very few documents are longer than 1000 words. For this reason we decided to clip and pad documents to a fixed length, considering their average length. We made sure that less than 5 % of documents were clipped and thus chose 600, 550 and 150 words as experimental length for lyrics, movie reviews and smartphone reviews respectively. This way the computation complexity of each experiment was significantly reduced without any loss in data quality. The shorter documents were zero-padded to reach the uniform length, concluding the preprocessing step.

## 3.3 Baseline Models

As baselines for comparison, we implemented the classical SVC and Logistic Regression linear classifiers with bag-of-words text representation and tf-idf scoring, optimized with grid searched regularization parameters. We also tried a single LSTM layer above the embedding layer followed by the dense layer that serves as classifier. The fourth baseline is described in [10]. Authors first apply a bi-directional recurrent structure (left and right LSTMs) to capture context from word embedding representations. Afterwards, a max-pooling layer is used to automatically select the best features for the classification. They report excellent results on topic recognition tasks and good results on emotion recognition of movie reviews. In [11] we found an even more complex recurrent model. It builds upon the bi-directional structure of [10] and adds two-dimensional convolution and pooling layers that are applied to the generated word-feature window. Authors exercise the model in various datasets. Best results they report are achieved on topic modeling and sentiment analysis of short sentences. The last baseline model we used is based on a single one-dimensional convolution. It is very similar to the network proposed by Yoon Kim in [8]. Here we have self-trained word embeddings and convolutions with kernel sizes 3, 4, 5 that are concatenated together. Max pooling and dropout layers follow the convolutions with a dense layer at the top.

# 4. RESULTS AND DISCUSSION

In this section we present and discuss accuracy scores we got on each of the three datasets, exercising NgramCNN and the baseline models. We also discuss some of the optimal hyper-parameter values that were found.

## 4.1 Classification Results

We used 70/10/20 percent split for training, development and testing respectively in each experiment. Classification results are presented in Table 2. As we can see, NgramCNN architecture model achieves excellent results on all three experiments. It performs significantly better than the baseline models on song lyrics (2.2 % higher accuracy than SingleCNN) and slightly better on smartphone reviews (0.4 % higher accuracy than BLSTM-2DCNN). On IMDB movie reviews dataset it reaches an accuracy score of 91.2 %. We also see that the 3 recurrent models perform badly on song lyrics and movie reviews (long documents). They perform even worse than Logistic Regression and SVC linear classifiers. On phone reviews (short documents) on the other hand, they give almost same accuracy as the two convolution models. Obviously, recurrent networks work better with short texts, same as reported in [11]. They can hardly preserve long-term word dependencies on long documents. By contrast, feature extraction layers of NgramCNN that are based on convolutions and pooling, are very effective in capturing emotional context and selecting the most discriminative features in both long and short document tasks. It is also worth mentioning that even though we did not systematically record training time of each model, we saw

**Table 2. Classification Accuracy on three tasks**

| Model\Dataset | Lyrics | Movies | Phones |
|---|---|---|---|
| Optimized LR | 73.1 | 89.4 | 92.4 |
| Optimized SVC | 72.7 | 88.5 | 92.6 |
| SingleLSTM | 70.3 | 84.9 | 93.7 |
| BLSTM-POOL | 70.6 | 85.5 | 94.3 |
| BLSTM-2DCNN | 71.2 | 85.7 | 95.5 |
| SingleCNN | 73.4 | 89.8 | 94.2 |
| NgramCNN | **75.6** | **91.2** | **95.9** |

that NgramCNN and SingleCNN were much faster to train compared to Linear Regression, SVC or the three recurrent network models.

## 4.2 Other Observations

NgramCNN architecture and hyper-parameters described in Section 2 were equally applied on the three tasks. There were however various hyper-parameters that behaved differently on each dataset. We used grid searching on train and dev sets to find optimal values for those parameters and also observed performance sensitivity of NgramCNN. Classification accuracy was highly sensitive to kernel size of convolution layers and pool size of max-pooling layers. We saw that pooling is essential after each convolution. Alternative architectures of consecutive convolutions and a final pooling layer were considerably weaker. We also found that 4 is the optimal region length in every pooling operation. Extending in width with extra convolutions of 4 or 5 kernel sizes didn't bring any improvement. Nevertheless, it might be a good option when working with even longer text documents. On the other hand, more consecutive convolutions (extending in depth) followed by pooling layers could possibly enhance accuracy if bigger datasets were available. Number of epochs till convergence, was irregular and specific to each task. Song lyrics required six epochs, whereas movie and smartphone reviews converged in four and eight respectively. We did not notice much sensitivity with respect to batch size. Optimal results were achieved with a batch of 60. Finally, sigmoid and softplus were equally fruitful and best activation functions for the output layer.

# 5. CONCLUSIONS AND FUTURE WORK

In this paper we presented NgramCNN, a novel neural network architecture designed to recognize emotion category in long text documents of different types and content. It uses pretrained word embeddings for representing text features, a series of repeating convolution and max-pooling neural layers for feature extraction and a simple single-layer classifier for predicting sentiment polarity label of each document. This design follows the common principle of the highly successful image analysis architectures: generic deep features combined with a simple classifier. Experimental results on song lyrics, movie reviews and smartphone reviews confirm the superiority of NgramCNN, especially on long text documents. Contrary, RNN-based models that were used as baselines perform comparably well on short reviews but considerably worse (even worse than Logistic Regression or SVC) on longer documents. In the future, we intent to investigate performance of NgramCNN and similar alternative architectures on shorter texts like sentences and possibly on even bigger datasets. The goal is to build a generic and powerful architecture for text-based sentiment analysis, that adapts to texts of different lengths and content with few hyper-parameter changes to produce state of the art results in reasonable training an inference time.

# 6. ACKNOWLEDGMENTS

This work was supported by a fellowship from TIM[2]. Part of computational resources was provided by HPC@POLITO[3], a project of Academic Computing within the Department of Control and Computer Engineering at Politecnico di Torino.

[2] https://www.tim.it

[3] http://hpc.polito.it

# 7. REFERENCES

[1] LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P., Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), pp. 2278-2324, 1998

[2] M Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A.; Fei-Fei, L., ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision, Vol. 115, Issue 3, pp. 211–252, Dec. 2015

[3] Krizhevsky, A.; Sutskever, I.; Hinton, G. E., Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*, pp. 1097-1105, 2012.

[4] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Rabinovich, A., Going deeper with convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-9, 2015.

[5] Simonyan, K.; Zisserman, A., Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014, *CoRR, abs/1409.1556*

[6] Canziani, A.; Culurciello, E., Paszke, A., An Analysis of Deep Neural Network Models for Practical Applications, 2014, *CoRR, abs/1605.07678*

[7] Razavian, A. S; Azizpour, H.; Sullivan, J.; Carlsson, S., CNN Features off-the-shelf: an Astounding Baseline for Recognition, 2014, *CoRR, abs/1403.6382*

[8] Kim, Y., Convolutional neural networks for sentence classification, 2014, *arXiv preprint arXiv:1408.5882*.

[9] Kalchbrenner, N.; Grefenstette, E., A Convolutional Neural Network for Modelling Sentences, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, USA, June 2014.

[10] Lai, S.; Xu, L.; Liu, K.; Zhao, J., Recurrent Convolutional Neural Networks for Text Classification, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 2267-2273, Austin, Texas, 2015

[11] Zhou, P.; Zhenyu, Q.; Zheng, S.; Xu, J.; Bao; H.; Xu, B., Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaca, Japan, Dec 2016.

[12] Hochreiter, S.; Schmidhuber, J., Long Short-Term Memory, in: Journal of Neural Computation, Vol. 9, No. 8, pp. 1735-1780, 1997.

[13] Pawar, P. Y.; Gawande, S., A comparative study on different types of approaches to text categorization, International Journal of Machine Learning and Computing Vol. 2, No. 4, pp. 423-426, 2012

[14] Bengio, Y.; Ducharme, R.; Vincent, P.; Janvin, C., A neural probabilistic language model, Journal of Machine Learning Res. Vol. 3, pp. 1137–1155, 2003

[15] Çano E.; Morisio M., Quality of Word Embeddings on Sentiment Analysis Tasks, in: NLDB 2017: 22nd International Conference on Natural Language and Information Systems, Springer, pp. 332-338, Liege, Belgium, 21 - 23 June 2017, doi: 10.1007/978-3-319-59569-6_42

[16] Mandelbaum, A.; Shalev, A., Word Embeddings and Their Use In Sentence Classification Tasks, 2016, *arXiv preprint arXiv:1610.08229*.

[17] Zhang, Y.; Wallace, B., A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification, 2016, *arXiv preprint arXiv:1510.03820*

[18] Çano, E.; Morisio, M., Moodylyrics: A Sentiment Annotated Lyrics Dataset, in: 2017 International Conference on Intelligent Systems, Metaheuristics and Swarm Intelligence, ACM, pp. 118-124, Hong Kong, March 2017, doi:10.1145/3059336.3059340.

[19] Çano E.; Morisio M., Music Mood Dataset Creation Based on Last.fm Tags, in: 2017 International Conference on Artificial Intelligence and Applications, Vienna, Austria, May 2017, doi:10.5121/csit.2017.70603.

[20] Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; Potts, C., Learning word vectors for sentiment analysis, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Portland, Oregon, USA, 2011, pp. 142–150