Text Classification using Convolutional Neural Networks on Character level

Anthony FARAUT
Master Student in double degree
University of Passau
faraut01@stud.uni-passau.de

ABSTRACT

This paper offers an implementation of Convolutional Networks for Text Classification based on characters. This article is particularly based on paper [5] and provides a simplification of the work previously made. In fact, the simplification has been made in order to enlarge the possibilities of using this approach on different use cases as for example on smaller texts. In this paper, the development is based on an open source library in python which it named "Keras"[1].

This paper complements document [5] which compared traditional models such as bag of words, n-grams and their TFIDF variants, and deep learning models such as word-based ConvNets and recurrent neural networks.

Keywords

Text classification, Convolutional Neural Networks

1. INTRODUCTION

Text classification is one of the fundamental task in Natural Language Processing. Text classification has a broad range of applications in spam detection, authorship identification, age/gender identification, language identification, sentiment analysis, etc... The main goal of text classification is to classify texts under a predefined class which means to assign a predefined label to a text.

In most research works, lexical features are used for text classification such as n-grams which are based on words. This approach works well and performs the best. Convolutional networks (ConvNets) are efficient in broad applications as speech recognition, image and video recognition, recommendation systems, and natural language processing which is the part that interests us. Thereby, most recent works used machine learning approaches such as deep learning, convolutional neural networks or recurrent neural networks for text classification.

Convolutional networks are widely used in the literature as in [6], [3] and more precisely for natural language processing and text classification as in [4], [2] and [1] The main advantage of using ConvNets is that no knowledge on the syntactic or semantic structures of a language is required. Besides, as characters always constitute words, this approach could be very interesting and promising, as it can work on different languages when having a fixed set of characters, e.g. an alphabet.

Last but not least, working on characters allows to detect and learn abnormal character combinations such as misspellings, emoticons and new words that can appear periodically as with a good example in the social networks. Indeed, a word can be pertinent during a period of time and be stale two hours after, for example during an event or a disaster with a specific hashtag or keyword (for example #PrayForParis).

The rest of this paper is organized as follows : in section 2, I will briefly present artificial neural network architecture which is the basis of the concepts presented in this paper. In section 3, I will focus on convolutional neural networks which are a specification of artificial neural network. Then, in section 4, I will present the development I made in order to do some convolutional neural networks tests. I will also present the data I focused on and all the choices I made for this development. In section 5, I will present the results and some small conclusion that I had for each test I made. Finally, in section 6, I will conclude by trying to extract the essence of my research using results from my own development.

2. ARTIFICIAL NEURAL NETWORK

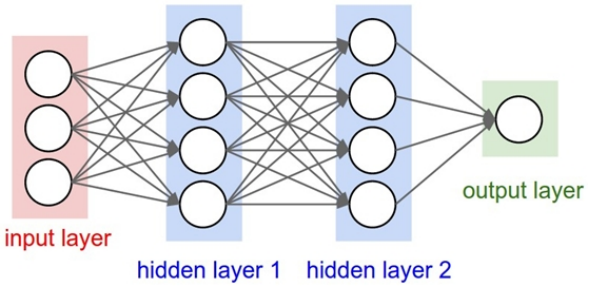

Figure 1: Neural network architecture

The figure 1 presents the global architecture of an artificial neural network. As clearly presented, it is an interconnected group of nodes, as the network of neurons in a brain. Furthermore, a node is a mathematical function. Each node

[1]https://keras.io/

represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another. The arrows make the communication possible between the nodes.

3. CONVOLUTIONAL NEURAL NETWORKS

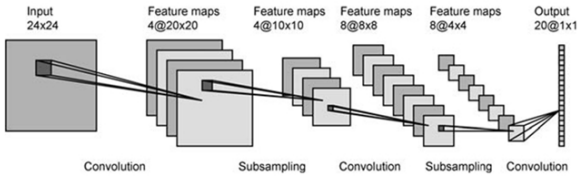

Figure 2: Convolutional Neural network architecture

The Convolutional neural networks are a specification of artificial neural networks inspired by the animal visual cortex. (Wikipedia definition: *The connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. Each individual neurons are arranged in such a way that they respond to overlapping regions tiling the visual field.*)

As explained before in the introduction, these specific neural networks have wide applications in image and video recognition, recommendation systems and natural language processing.

3.1 Character quantization

Each language has its own alphabet in order to construct words and thus sentences. In this paper, the focus is made on characters, it is important to choose which characters will be kept in the process. Indeed, a sequence of encoded characters will be expected as input in the convolutional neural network, and it needs to know the range possibility of characters. In [5] the "one-hot" encoding is used for representing the letters for the convolutional neural network.

Concerning my development I used two approaches in order to represent the letters. I made the choice of using two approaches because of the excessive memory utilization of the "one hot encoding" technique. I will show the difference between the two approaches in the 5.1 part concerning the memory utilization.

- One hot encoding

It's a vector of size N (according to the size of the alphabet which is N). The vector consists of a single 1 in a cell used to identify the element and 0 for all others. It's a sparse vector containing in most cells 0. In order to make the reference to the figure 2, an one-hot encoding input will be like the figure 3 for the word "habe" and the following alphabet: "abcdefgh".

h -> [0, 0, 0, 0, 0, 0, 0 , 1], a -> [1, 0, 0, 0, 0, 0, 0 , 0], b -> [0, 1, 0, 0, 0, 0, 0 , 0], e -> [0, 0, 0, 0, 1, 0, 0 , 0]

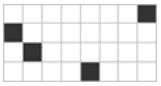

Figure 3: One hot encoding architecture

- Character embeddings

It's a real number vectors in a low-dimensional space in order to overcome the lack of one hot vectors which are sparse vectors. *(Wikipedia definition : "The elements from the vocabulary are mapped to vectors of real numbers in a low-dimensional space relative to the vocabulary.")*

4. DEVELOPMENT

In this section, I will briefly present you character-level convolutional neural networks for text classification without going into details as all is clearly explained in [5]. I will particularly focus on what I do in my development in order to prove the effectiveness of this approach.

As explained before, most of the research works are based on words because it's a logical decomposition of phrases. However in [5], characters are used in order to classify texts.

4.1 The Convolutional Neural Network layers

Concerning my development, the inputs have a number of features equal to 70 or 96 due to the character quantization methods (upper-case and lower-case). The are various input feature lengths (100, 140, 200, 300, 400). In [5], the authors said that "It seems that 1014 characters could already capture most of the texts of interest", but I wanted to try on reduced size of sentence in order to apply these techniques on text from social networks (for example). Indeed, on social networks, texts usually are short (up to 140 characters with Twitter for example).

In [5], the authors have a Convolutional Neural Network with 9 layers (6 convolutional layers and 3 fully-connected layers). As I reduced the number of characters for each sentence, I tried to reduce the Convolutional Neural Network model. I ended up having 3 convolutional layers, 3 max pooling layers and 3 fully-connected layers. Between the two first convolutional layers and the two first max pooling layers I inserted two activations functions. The role of the activation function in a neural network is to produce a non-linear decision boundary via non-linear combinations of the weighted inputs. Concerning the 3 fully-connected layers, I also inserted three activations functions between them. For each of them, I used the 'relu' activation and for the final one, I used the 'softmax' activation. I also inserted 2 dropout modules between the 3 fully-connected layers to regularize like in [5].They have dropout probability of 0.5.

As explained in the section 4.4, for the character quantization I used two approaches in order to represent the letters ("one hot encoding" and "embedding"). However, I built two models that are substantially identical. The embedding model has an additional layer which is the Embedding layer.

Finally, concerning the optimizer function, I used the Adadelta method, with "categorical_crossentropy" (also known as multiclass logloss) because it normally converge quicker to a better result as SGD (for example).

Table 1: Model Design

Layer	Conv kernels	Filter len	Pool len	Stride
Conv 1D	96	7		
Max Pool 1D			3	3
Conv1D	96	7		
Max Pool 1D			3	3
Conv 1D	96	3		
Max Pool 1D			3	3

4.2 Tool and source code

For the implementation of the Convolutional Neural network, "Keras"[2] was used, which is an open-source library in python. "Keras is a minimalist, highly modular neural networks library" on top of Theano[3] and Theano is a numerical computation library for Python.

The implementation was made using python 2.7[4]. My source code is available online on my Github page.

4.3 Data

According to the Github page of "Xiang ZHANG" which is an author of [5], some datasets are provided. The datasets are available at http://goo.gl/JyCnZq and we can find:

- Amazon Review Score Dataset;
- DBPedia Ontology Classification Dataset;
- Yahoo! Answers Topic Classification Dataset;
- Yelp Review Polarity Dataset;
- Yelp Review Full Star Dataset;
- etc.

I made the choice of focusing the experiments on four datasets. Two datasets which are composed of notes classification with the "Yelp review" datasets. Two datasets which are composed with subject classification as "Car", "Science" etc... I made this choice in order to see what was the easiest way to classify between notes (stars) and topics.

4.4 Choice of the alphabet

According to [5], *"They observed that distinguish between upper-case and lower-case letters usually (but not always) gives worse results."* Thereby, I made the choice of making this distinction in order to see the results on my approach, and because in social networks we can see non-standard terms or symbolic expressions written by the users (as an example âĂIJLooooooooooooooove" in order to mean âĂIJmuch love" or âĂIJbrb" in order to mean âĂIJbe right back"). Furthermore, as they have the freedom to express their opinions, their moods in words with no syntactic or semantic control, we can see grammatical alteration such as incomplete sentences and word distortion with upper-case, lower-case etc. For that, I took the same alphabet as the paper just cited. The alphabet is composed of 70 characters, including :

- 26 english letters;
- 10 digits;
- 33 other characters;
- The space character.

Alphabet 1: "[a-z][0-9]-,;.!?:'"/\|_@#$%&̂* ‘+-=<>()[]{} + space"

Alphabet 2: "[A-Z][a-z][0-9]-,;.!?:'"/\|_@#$%&̂* ‘+-=<>()[]{} + space"

4.5 Parameters

For each of the experiments, some parameters varied depending on the sentence size (100, 140, 200, 300, 400 characters) and some remain unchanged. The parameters which remain unchanged are:

- The batch size = 32;
- The learning rate = 0.01;
- The number of convolution kernels = 96;
- The filter length = 7;
- The pool length = 3;
- The stride = 3.

I will present the parameters that vary in each different dataset part.

- DBpedia dataset

Here are the DBpedia parameters which vary depending on the sentence size:

Table 2: DBpedia parameters for the CNN

Sentence length	Epoch	Train size	Test size
100	10	500000	60000
140	10	450000	56000
200	20	380000	48000
300	30	265000	33000
400	40	145000	18000

[2]https://keras.io/
[3]http://deeplearning.net/software/theano/
[4]https://www.python.org/download/releases/2.7/

- Yahoo dataset

Here are the Yahoo parameters which vary depending on the sentence size:

Table 3: Yahoo parameters for the CNN

Sentence length	Epoch	Train size	Test size
100	10	500000	40000
140	10	500000	35000
200	20	380000	28000
300	30	265000	20000
400	40	145000	15000

- Yelp dataset

For both datasets of Yelp ("Yelp polarity" and "Yelp Full", the same parameters for each of the experiments are used.

The parameters which vary depending on the sentence size:

Table 4: Yelp parameters for the CNN

Sentence length	Epoch	Train size	Test size
100	10	500000	36000
140	10	500000	35000
200	20	380000	32000
300	30	265000	28000
400	40	145000	18000

5. RESULTS

In this section, I will present you the memory utilization for the convolutional neural networks. Indeed, depending on the choice made for the character quantization, the memory utilization can vary a lot. Then, I will present you the results for each corpus I focused on. Two datasets which are composed of notes classification with the "Yelp review" datasets. Two datasets which are composed with subject classification as "Car", "Science" etc... Finally, all the results presented are from the True Positive measure (which are valid dataset classes found by the process).

5.1 Memory utilization

As explained before, for the experiments, two representations for the characters quantization are used (the one-hot encoding and the embeddings). Furthermore, two alphabets are also used in order to compare them and to see whether it impacts the results or not. One thing is certain, there will inevitably be an impact on the place taken in memory, since the "one hot encoding" needs more memory for its sparse vectors.

On average, for the one-hot encoding, the memory usage is around [40-80] Gb, which is normal because of the encoding of each character as explained in section 4.4. Indeed, each character is represented by a sparse vector (with a lot of 0). All these values take a lot of RAM space. Note that for the distinction between upper-case and lower-case letters, the memory usage varies from about [10-40] Gb. However, on average, for the embedding, the memory usage is around less than 1 Gb. That's totally normal to see a huge difference between the embedding and the one hot encoding. It is noteworthy that such tests are not feasible on a common computer that we find on the market for "one hot encoding" approach.

5.2 DBpedia Corpus

The "DBPedia Ontology Classification Dataset" contains 14 non-overlapping classes from DBpedia 2014. "From each of these 14 ontology classes, they randomly choose 40,000 training samples and 5,000 testing samples. Therefore, the total size of the training dataset is 560,000 and testing dataset 70,000. (from the README of the dataset)"

The 14 non-overlapping classes are : "Company, EducationalInstitution, Artist, Athlete, OfficeHolder, MeanOfTransportation, Building, NaturalPlace, Village, Animal, Plant, Album, Film, WrittenWork"

Example of the content in the csv file : Class, "Title", "Content"

Animals ,"Dinomyidae"," The Dinomyidae were once a very speciose group of South American hystricognath rodents but now contains only a single living species the pacarana. The Dinomyidae family included among its ranks the largest rodents known to date the bison-sized Josephoartigasia monesi and the smaller Josephoartigasia magna."

I will now present the data in a table and in a figure.

Length	one_hot_1	$embeddings_1$	one_hot_2	$embeddings_2$
400	88.59	87.96	88.74	88.01
300	88.14	87.81	88.79	87.68
200	88.65	87.39	88.94	87.57
140	87.93	86.33	87.98	86.48
100	86.9	85.18	87.03	85.47

Table 5: DBpedia results

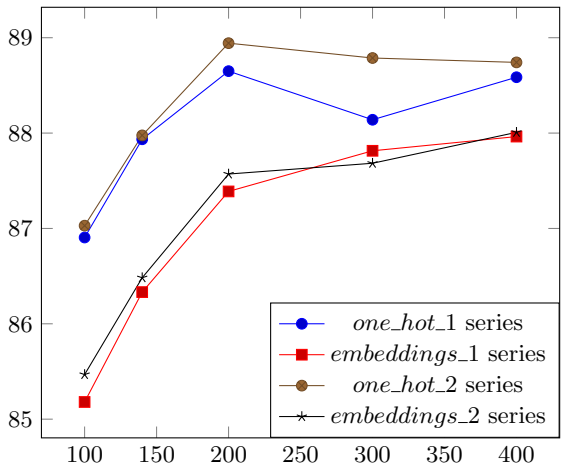

Figure 4: DBpedia chart

The chart shows that below 200 characters the results decrease but remain acceptable. This probably comes from the fact that below 200 characters, the process can not combine enough characters to find a pattern. The classification for this experiments is not easy because there are 14 non-overlapping classes to predict and the process have 1 chance in 14 of being right. However, as the content of each class seems well defined and does not seem to overlap the contents of another class, we can see up to 85% results.

5.3 Yelp polarity Corpus

The "Yelp Review Polarity Dataset" contains 2 non-overlapping classes. The Yelp reviews dataset consists of reviews from Yelp. It is extracted from the Yelp Dataset Challenge 2015 data. For more information, please refer to this link.

"The Yelp reviews polarity dataset is constructed by considering stars 1 and 2 negative, and 3 and 4 positive. For each polarity 280,000 training samples and 19,000 testing samples are take randomly. In total there are 560,000 trainig samples and 38,000 testing samples. Negative polarity is class 1, and positive class 2."

Example of the content in the csv file : Class, "Review"

"1","The food is good. Unfortunately the service is very hit or miss. The main issue seems to be with the kitchen, the waiters and waitresses are often very apologetic [...] that some of them avoid the tables after taking the initial order to avoid hearing complaints."

I will now present the data in a table and in a figure.

Length	one_hot_1	$embeddings_1$	one_hot_2	$embeddings_2$
400	82.56	80.56	82.81	79.81
300	81.99	81.49	82.75	81.44
200	79.69	80.23	79.98	76.96
140	80.32	51.17	80.17	51.17
100	78.58	50.62	78.56	50.62

Table 6: Yelp polarity results

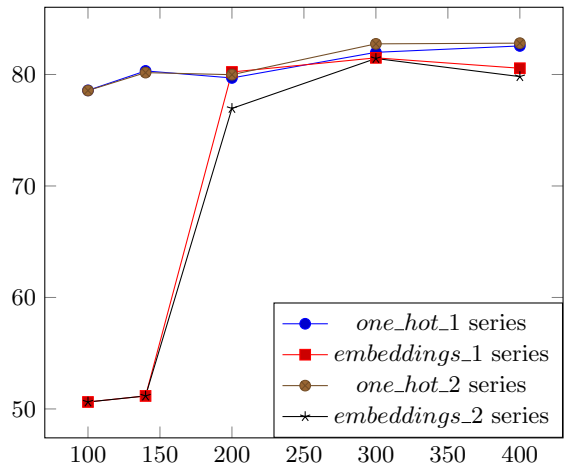

Figure 5: Yelp polarity chart

The table shows that there is only a difference below 200 characters for "one hot encoding" and embedding. Otherwise, the pairs of curves appear approximately identical. Compared to the results of the dataset "Yelp full" 6, we may well see that the results are much better. This is easily explained, since both classes are distinct and the words that compose them do not seem to overlap. It's either good or bad. The words to define a good or bad rating are completely different so it is easy to find a combination of characters to define a model to classify.

5.4 Yelp Full Corpus

The "Yelp Review Full Star Dataset" with 5 non-overlapping classes. The Yelp reviews dataset consists of reviews from Yelp. It is extracted from the Yelp Dataset Challenge 2015 data. For more information, please refer to this link.

"The Yelp reviews full star dataset is constructed by randomly taking 130,000 training samples and 10,000 testing samples for each review star from 1 to 5. In total there are 650,000 trainig samples and 50,000 testing samples."

Example of the content in the csv file : Class, "Review"

"5","Susan is absolutely the best! I went to Susan after a long search for a new stylist and am so pleased with my choice. Susan is fantastic with cuts and color, [...] Susan is charming, knowledgeable and is easy to talk to. Overall, my experience could not have been better. I will continue to go to Susan in the future."

I will now present the data in a table and in a figure.

Length	one_hot_1	$embeddings_1$	one_hot_2	$embeddings_2$
400	42.77	22.11	42.68	22.11
300	44.45	21.7	45.69	21.7
200	45	20.99	44.45	20.99
140	44.16	20.6	44.66	20.6
100	43.95	20.34	43.78	20.34

Table 7: Yelp full results

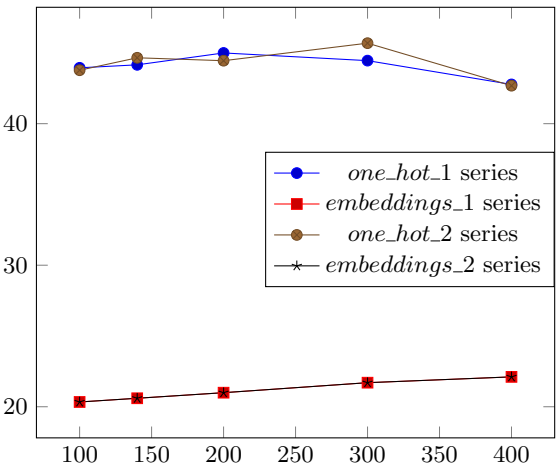

Figure 6: Yelp full chart

The chart shows that there is a difference between one hot encoding and embedding. Indeed, we can see that the embedding are half as good as the one hot encoding. I think what explains the difference in results between the corpus "Yelp polarity" and this one is that it is much harder to predict five classes of notes (star from 1 to 5) than two as the "Yelp polarity" corpus. I think that (as proved thanks with the Yelp polarity chart 5) that distinguish a 1 star with a 5 start is much easy, but between a 4 star and a 5 star it much harder. I think the real reason is that the words contained in the classes that are close (for example the class 4 and 5) overlap and it is difficult with such a small number of character to make the difference.

5.5 Yahoo Corpus

The "Yahoo! Answers Topic Classification Dataset" contains 10 non-overlapping classes. The dataset is the Yahoo! Answers corpus as of 10/25/2007. It includes all the questions and their corresponding answers. The corpus contains 4483032 questions and their answers.

"The Yahoo! Answers topic classification dataset is constructed using 10 largest main categories. Each class contains 140,000 training samples and 6,000 testing samples. Therefore, the total number of training samples is 1,400,000 and testing samples 60,000 in this dataset. From all the answers and other meta-information, they only used the best answer content and the main category information."

The 10 non-overlapping classes are : "Society & Culture, Science & Mathematics, Health, Education & Reference, Computers & Internet, Sports, Business & Finance, Entertainment & Music, Family & Relationships, Politics & Government"

Example of the content in the csv file : Class, Question title, Question content and Best answer

"5","I am trying to mount a dual boot(linux/windows)?","I am trying to mount a dual boot(linux/windows) dell laptop internal ide (ibm) 20 gig hard disk on my dell precision desktop (wind2k) using a ide to usb cable.(sabrent). [...] Unix machines run completely different hardware and if you tried that it just wouldn't work at all."

For the experiments, I only used the Class and the Best answer.

I will now present the data in a table and in a figure.

Length	one_hot_1	$embeddings_1$	one_hot_2	$embeddings_2$
400	50.43	59.92	54.04	61.56
300	55.42	56.37	59.43	57.52
200	53.77	52.95	56.75	53.66
140	53.49	48.51	51.61	50.34
100	48.97	44.19	49.4	46.4

Table 8: Yahoo! results

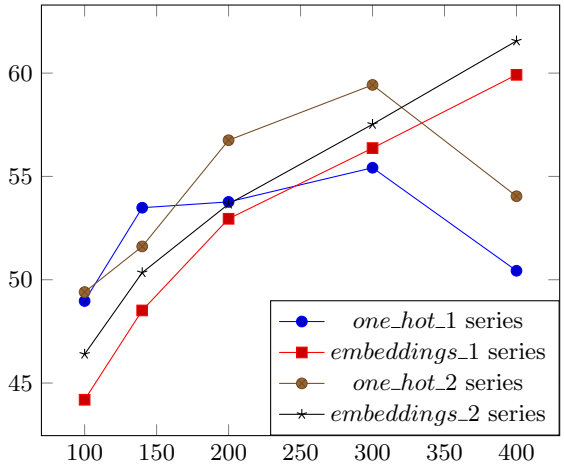

Figure 7: Yahoo! chart

On this chart from the experiments on the Yahoo corpus, we can see that the "one hot encoding" and the embedding curves have a similar form. The one hot encoding curves seem to make a Gaussian curves with a high with 300 characters and the embedding curves seem to increase with the number of characters. We can still get better results than for the corpus "Yelp full".

6. CONCLUSIONS

For the DBpedia and the Yahoo corpus, we can see (thanks to the charts) that below 200 characters the results fall. It probably comes from the fact that below this number of characters the process can not combine enough characters to find a good pattern in order to classify the texts. This means that it would probably be hard to apply this approach to the text from social networks. However, since we have not tried it on a corpus from social networks we can not affirm this conclusion.

We can also see that, the results depend on the corpus. Indeed, the Yahoo and the DBpedia corpus seems to have the same classification problem with several classes, but the results are not the same for the both datasets.

Building on the four tests I conducted, I can confirm that the "one hot encoding" seems to have better results than the embedding although it uses much more memory in practice.

Finally, the results depend also on the classes the process has to predict. Indeed, if the classes are distinct (Computing, car) it is easier for the process to classify. However, if the classes can overlap as the stars in a note, it is much harder for the process because there are words in common in the classes that will be close (4 stars and 5 stars, by example). For the note classification, we can find (for example), in the classes 4 and 5 some words as "Good", "Super", "Awesome" etc...

In conclusion, I would say that the Convolutional Neural Networks work well for text classification. However, we must adapt the model to the corpus/dataset as well as classes to

predict. Finally, according to my experiments, small texts such as those from social networks do not seem to provide enough information to have an optimal classification.

7. ACKNOWLEDGMENTS

I thank Professor Doctor Michael GRANITZER for providing me a high performance server which allowed me to make my tests.

8. REFERENCES

[1] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional Neural Network Architectures for Matching Natural Language Sentences. *Advances in Neural Information Processing Systems 27*, pages 2042–2050, 2014.

[2] R. Johnson and T. Zhang. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. *Naacl*, (2011):103–112, dec 2015.

[3] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A Convolutional Neural Network for Modelling Sentences. apr 2014.

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

[5] X. Z. J. Z. Y. LeCun. Character-level Convolutional Networks for Text Classification. *Nips*, abs/1509.01626:3057–3061, 2015.

[6] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv preprint arXiv*, page 1312.6229, dec 2013.