# Do Convolutional Networks need to be Deep for Text Classification ?

**Hoa T. Le[1], Christophe Cerisara[1], Alexandre Denis[2]**
[1] LORIA, UMR 7503, Nancy, France; [2] SESAMm, France

**Abstract**

We study in this work the importance of depth in convolutional models for text classification, either when character or word inputs are considered. We show on 5 standard text classification and sentiment analysis tasks that deep models indeed give better performances than shallow networks when the text input is represented as a sequence of characters. However, a simple shallow-and-wide network outperforms deep models such as DenseNet with word inputs. Our shallow word model further establishes new state-of-the-art performances on two datasets: Yelp Binary (95.9%) and Yelp Full (64.9%).

## Introduction

Following the success of deep learning approaches in the ImageNet competition, there has been a surge of interest in the application of deep models on many tasks, such as image classification (Krizhevsky, Sutskever, and Hinton 2012), speech recognition (Hinton et al. 2012). Each new coming model proposes better and better architectures of the network to facilitate training through longer and longer chains of layers, such as Alexnet (Krizhevsky, Sutskever, and Hinton 2012), VGGNet (Simonyan and Zisserman 2014), GoogleLeNet (Szegedy et al. 2015), ResNet (He et al. 2016a) and more recently Densenet (Huang, Liu, and Weinberger 2016).

Several works have explained this success in computer vision, in particular (Zeiler and Fergus 2014) and (Donahue et al. 2014): deep model is able to learn a hiearchical feature representation from pixels to line, contour, shape and object. These studies have not only helped to demistify the black-box of deep learning, but have also led the path to other approaches like transfer learning (Yosinski et al. 2014), where the first layers are believed to bring more general information and the last layers to convey specific information on the target task.

This paradigm has also been applied in text classification and sentiment analysis, with deeper and deeper networks being proposed in the literature: (Kim 2014) (shallow-and-wide CNN layer), (Zhang, Zhao, and LeCun 2015) (6 CNN layers), (Conneau et al. 2016) (29 CNN layers). Besides the development of deep networks, there is a debate about which atom-level (word or character) would be the most effective for Natural Language Processing (NLP) tasks. Word embeddings, which are continuous representations of words, initially proposed by (Bengio et al. 2003) and widely adopted after word2vec (Mikolov et al. 2013) have been chosen as the main standard representations for most NLP tasks. Based on this representation, a common belief is that, similarly to vision, the model will learn hiearchical features from the text: words combine to form n-grams, phrases, sentences...

Several recent works have extended this model to characters instead of words. Hence, (Zhang, Zhao, and LeCun 2015) propose for the first time an alternative character-based model, while (Conneau et al. 2016) take a further step by introducing a very deep char-level network. Nevertheless, it is still not clear which atom-level is the best and whether very deep networks at the word-level are really better for text classification.

This work is motivated by these questions and we hope to bring elements of a response by providing a full comparison of a shallow-and-wide CNN (Kim 2014) both at the character and word levels on the 5 datasets described in (Zhang, Zhao, and LeCun 2015). Moreover, we propose an adaptation of a DenseNet (Huang, Liu, and Weinberger 2016) for text classification and sentiment analysis at the word-level, which we compare with the state-of-the-art.

This paper is structured as follows. First we summarize the related work and then describe the shallow CNN and introduce our adaptation of DenseNet for text. We then evaluate our approach on the 5 datasets of (Zhang, Zhao, and LeCun 2015) and show our experimental results. Experiments show that the shallow-and-wide CNN on word-level can beat a very deep CNN on char-level. The paper concludes with some open discussions for future research about deep structures on text.

## Related work

Text classification is an important task in Natural Language Processing. Traditionally, linear classifiers are often used for text classification (Joachims 1998), (McCallum and Nigam 1998), (Fan et al. 2008). In particular, (Joulin et al. 2016) show that linear models could scale to a very large dataset rapidly with a proper rank constraint and a fast loss approximation. However, a recent trend in the domain is to exploit deep learning methods, such as convolutional neural networks: (Kim 2014), (Zhang, Zhao, and LeCun 2015), (Conneau et al. 2016) and recurrent networks: (Yogatama et al. 2017), (Xiao and Cho 2016). Sentiment Analysis is also an active topic of research in NLP for a long time, with real-world applications in market research (Qureshi, O'Riordan, and Pasi 2013), finance (Bollen, Mao, and Zeng 2011), social science (Dodds et al. 2011), politics (Kaya, Fidan, and Toroslu 2013). The SemEval challenge has been setup in 2013 to boost this field and is still bringing together many competitors who have been using an increasing proportion of deep learning models over the years (Nakov et al. 2013; Rosenthal et al. 2014; Nakov et al. 2016). 2017 is the fifth edition of the competition, with at least 20 teams (over 48 teams) using deep learning and neural network methods. The top 5 winning teams all use deep learning or deep learning ensembles. Other teams use classifiers such as Naive Bayes classifier, Random Forest, Logistic Regression, Maximum Entropy and Conditional Random Fields (Rosenthal, Farra, and Nakov 2017).

Convolutional neural networks with end-to-end training have been used in NLP for the first time in (Collobert and Weston 2008; Collobert et al. 2011). The authors introduce a new *global* max-pooling operation, which is shown to be effective for text, as an alternative to the conventional *local* max-pooling of the original LeNet architecture (Lecun et al. 1998). Moreover, they proposed to transfer task-specific information by co-training multiple deep models on many tasks. Inspired by this seminal work, (Kim 2014) proposed a simpler architecture with slight modifications of (Collobert and Weston 2008) consisting of fine-tuned or fixed pre-training word2vec embeddings (Mikolov et al. 2013) and its combination as multi-channel. The author showed that this simple model can already achieve state-of-the-art performances on many small datasets. (Kalchbrenner, Grefenstette, and Blunsom 2014) proposed a dynamic *k*-max pooling to handle variable-length input sentences. This dynamic *k*-max pooling is a generalisation of the max pooling operator where *k* can be dynamically set as a part of the network.

All of these works are based on word input tokens, following (Bengio et al. 2003), which introduced for the first time a solution to fight the curse of dimensionality thanks to distributed representations, also known as *word embeddings*. A limit of this approach is that typical sentences and paragraphs contain a small number of words, which prevents the previous convolutional models to be very deep: most of them indeed only have two layers. Other works (Severyn and Moschitti 2015) further noted that word-based input representations may not be very well adapted to social media inputs like Twitter, where tokens usage may be extremely creative: slang, elongated words, contiguous sequences of exclamation marks, abbreviations, hashtags,... Therefore, they introduced a convolutional operator on characters to automatically learn the notions of words and sentences. This enables neural networks to be trained end-to-end on texts without any pre-processing, not even tokenization. Later, (Zhang, Zhao, and LeCun 2015) enhanced this approach and proposed a deep CNN for text: the number of characters in a sentence or paragraph being much longer, they can train for the first time up to 6 convolutional layers. However, the structure of this model is designed by hand by experts and it is thus difficult to extend or generalize the model with arbitrarily different kernels and pool sizes. Hence, (Conneau et al. 2016), inspired by (He et al. 2016b), presented a much simpler but very deep model with 29 convolutional layers.

Besides convolutional networks, (Kim et al. 2016) introduced a character aware neural language model by combining a CNN on character embeddings with an highway LSTM on subsequent layers. (Radford, Józefowicz, and Sutskever 2017) also explored a multiplicative LSTM (mLSTM) on character embeddings and found that a basic logistic regression learned on this representation can achieve state-of-the-art result on the Sentiment Tree Bank dataset (Socher et al. 2013) with only a few hundred labeled examples.

Capitalizing on the effectiveness of character embeddings, (Dhingra et al. 2016) proposed a hybrid word-character model to leverage the avantages of both worlds. However, their initial experiments show that this simple hybridation does not bring very good results: the learned representations of frequent and rare tokens of words and characters is different and co-training them may be harmful. To alleviate this issue, (Miyamoto and Cho 2016) proposed a scalar gate to control the ratio of both representations, but empiricial studies showed that this fixed gate may lead to suboptimal results. (Yang et al. 2017) then introduced a fine-grained gating mechanism to combine both representations. They showed improved performance on reading comprehension datasets, including Children's Book Test and SQuAD.

## Model

We describe next two models architectures, respectively shallow and deep, that we will compare in Section  on several text classifications tasks. Both models share common components that are described next.

### Common components

**Lookup-Table Layer** Every token (either word or character in this work) $i \in Vocab$ is encoded as a $d$-dimensional vector using a lookup table $Lookup_W(.)$:

$$Lookup_W(i) = \mathbf{W}_i, \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{d \times |Vocab|}$ is the embedding matrix, $\mathbf{W}_i \in \mathbb{R}^d$ is the $i^{th}$ column of $\mathbf{W}$ and $d$ is the number of embedding space dimensions. The first layer of our model thus transforms indices of an input sentence $s_1, s_2, \cdots, s_n$ of *n* tokens in *Vocab* into a series of vectors $\mathbf{W}_{s1}, \mathbf{W}_{s2}, \cdots, \mathbf{W}_{sn}$.

**Classification Layer**

The embedding vectors that encode a complete input sentence are processed by one of our main models, which outputs a feature vector $\mathbf{x}$ that represents the whole sentence. This vector is then passed to a classification layer that applies a *softmax* activation function (Costa 1996) to compute the predictive probabilities for all $K$ target labels:

$$p(y = k|X) = \frac{exp(\mathbf{w}_k^T \mathbf{x} + b_k)}{\sum_{k'=1}^{K} exp(\mathbf{w}_{k'}^T \mathbf{x} + b_{k'})} \tag{2}$$

where the weight and bias parameters $\mathbf{w}_k$ and $b_k$ are trained simultaneously with the main model's parameters. The loss function is then minimized by cross-entropy error.

## Shallow-and-wide CNN

Our first shallow-and-wide CNN model is adapted from (Kim 2014).

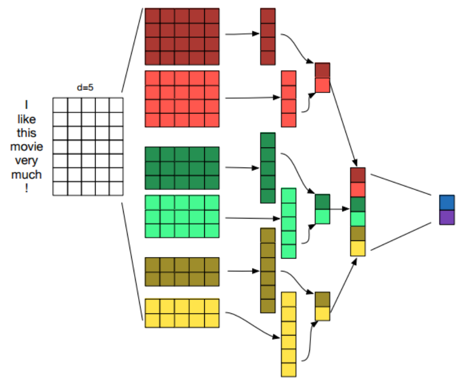

Figure 1: Shallow-and-wide CNN, from (Zhang and Wallace 2015): 3 convolutional layers with respective kernel window sizes 3,4,5 are used. A global max-pooling is then applied to the whole sequence on each filter. Finally, the outputs of each kernel are concatenated to a unique vector and fed to a fully connected layer.

Let $\mathbf{x}_i \in \mathbb{R}^d$ be an input token (*word* or *character*). An input $h$-grams $x_{i:i+h-1}$ is transformed through a convolution filter $\mathbf{w}_c \in \mathbb{R}^{hd}$:

$$c_i = f(\mathbf{w}_c \cdot \mathbf{x}_{i:i+h-1} + b_c) \tag{3}$$

with $b_c \in \mathbb{R}$ a bias term and $f$ the non-linear ReLU function. This produces a *feature map* $\mathbf{c} \in \mathbb{R}^{n-h+1}$, where $n$ is the number of tokens in the sentence. Then we apply a global max-over-time pooling over the feature map:

$$\hat{c} = \max\{\mathbf{c}\} \in \mathbb{R} \tag{4}$$

This process for one feature is repeated to obtain $m$ filters with different window sizes $h$. The resulting filters are concatenated to form a *shallow-and-wide* network:

$$\mathbf{g} = [\hat{c}_1, \hat{c}_2, \cdots, \hat{c}_m] \tag{5}$$

Finally, a fully connected layer is applied:

$$\hat{y} = f(\mathbf{w}_y \cdot \mathbf{g} + b_y) \tag{6}$$

**Implementation Details** The kernel window sizes $h$ for character tokens are $N_f = (15, 20, 25)$ with $m = 700$ filters. For word-level, $N_f = (3, 4, 5)$ with $m = 100$ filters.

## DenseNet

**Skip-connections** In order to increase the depth of deep models, (He et al. 2016a) introduced a skip-connection that modifies the non-linear transformation $\mathbf{x}_l = \mathcal{F}_l(\mathbf{x}_{l-1})$ between the output activations $\mathbf{x}_{l-1}$ at layer $l-1$ and at layer $l$ with an identity function:

$$\mathbf{x}_l = \mathcal{F}_l(\mathbf{x}_{l-1}) + \mathbf{x}_{l-1} \tag{7}$$

This allows the gradient to backpropagate deeper in the network and limits the impact of various issues such as vanishing gradients.

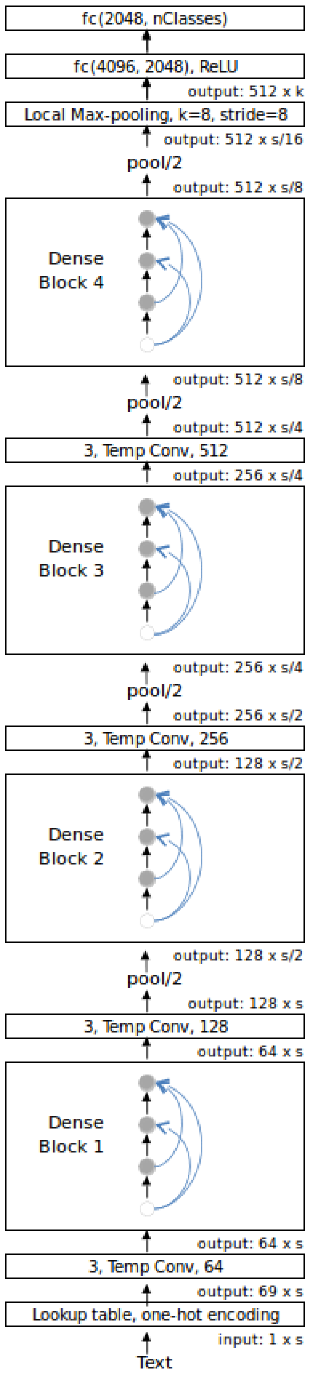

Figure 2: Character-level DenseNet model for Text classification. **3, Temp Conv, 128** means temporal convolutional operation with kernel window size = 3 and filter size = 64; **pool/2** means local max-pooling with kernel size = stride size = 2, it will reduce the size of the sequence by a half.

**Dense Connectivity** (Huang, Liu, and Weinberger 2016) suggested that the additive combination of this skip connection with $\mathcal{F}_l(\mathbf{x}_{l-1})$ may negatively affect the information flow in the model. They proposed an alternative concatenation operator, which allows to create direct connections from any layer to all subsequent layers, called *DenseNet*. Hence, the $l^{th}$ layer has access to the feature maps of all preceding layers, $\mathbf{x}_0, \cdots, \mathbf{x}_{l-1}$, as input:

$$\mathbf{x}_l = \mathcal{F}_l([\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{l-1}]) \tag{8}$$

This can be viewed as an extreme case of a ResNet. The distance between both ends of the network is shrinked and the gradient may backpropagate more easily from the output back to the input, as illustrated in Figure 3.

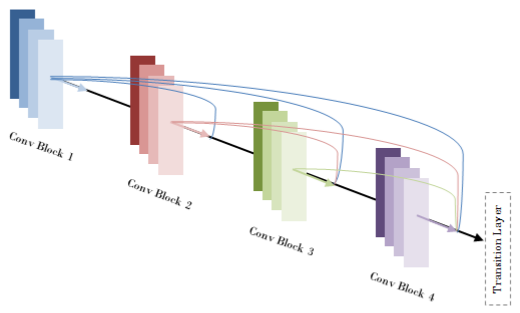

Figure 3: Dense Block. Multiple convolutional filters output 2D matrices, which are all concatenated together before going into another dense block.

**Convolutional Block and Transitional Layer** Following (He et al. 2016b), we define $\mathcal{F}_l(.)$ as a function of three consecutive operations: batch normalization (BN), rectified linear unit (ReLU) and a 1x3 convolution.

To adapt the variability of the changing dimension of the concatenation operation, we define a transition layer which composes a 1x3 convolution and a 1x2 local max-pooling between two dense blocks. Given a vector $c^{l-1}$ outputed by a convolutional layer $l-1$, the local max-pooling layer $l$ outputs a vector $c^l$:

$$\left[c^l\right]_j = \max_i \left[c^{l-1}\right]_{k\times(j-1)\leq i<k\times j} \tag{9}$$

where $1 \leq i \leq n$ and $k$ is the kernel pooling size. The word-level DenseNet model is the same as the character-level model shown in Figure 2, except for the last two layers, where the local max-pooling and two fully connected layers are replaced by a single global average pooling layer. We empirically observed that better results are thus obtained with word tokens.

**Implementation Details** The kernel window size with both character and word tokens is $h = 3$ tokens. For word-level, the kernel of the last local max-pooling is 8 while it is equal 3 for char-level (because the size of the sequence is shorter). Following (Conneau et al. 2016), we experiment with two most effective configurations for word and character-level: $N_b = (4-4-4-4)$ and $N_b = (10-10-4-4)$, which are the number of convolutional layers in each of the four blocks.

## Experimental evaluation

### Tasks and data

We test our models on the 5 datasets used in (Zhang, Zhao, and LeCun 2015) and summarized in Table 2. These datasets are:

- AGNews: internet news articles (Del Corso, Gulli, and Romani 2005) composed of titles plus descriptions and classified into 4 categories: World, Entertainment, Sports and Business, with 30k training samples and 1.9k test samples per class.
- Yelp Review Polarity: The Yelp review dataset is obtained from the Yelp Dataset Challenge in 2015. Each polarity dataset has 280k training samples and 19k test samples.
- Yelp Review Full: The Yelp review dataset is obtained from the Yelp Dataset Challenge in 2015. It has four polarity star labels: 1 and 2 as negative, and 3 and 4 as positive. Each star label has 130k training samples and 10k testing samples.
- DBPedia: DBPedia is a 14 non-overlapping classes picked from DBpedia 2014 (wikipedia). Each class has 40k training samples and 5k testing samples.
- Yahoo! Answers: ten largest main categories from Yahoo! Answers Comprehensive Questions and Answers version 1.0. Each class contains 140k training samples and 5k testing samples, including question title, question content and best answer. For DenseNet on word-level, we only used 560k samples because of lack of memory.

### Hyperparameters and Training

For all experiments, we train our model's parameters with the Adam Optimizer (Kingma and Ba 2014) with an initial learning rate of 0.001, a mini-batch size of 128. The model is implemented using Tensorflow and is trained on a GPU cluster (with 12Gb RAM on GPU). The hyperparameters are chosen following (Zhang, Zhao, and LeCun 2015) and (Kim 2014), which are described below. On average, it takes about 10 epochs to converge.

**Character-level** Following (Zhang, Zhao, and LeCun 2015), each character is represented as a one-hot encoding vector where the dictionary contains the following 69 tokens: $"abcdefghijklmnopqrstuvwxyz0123456789-, ;.!? : "/|\_\#\%\hat{}\& \star \tilde{} + \; =<> \; ()[]$". The maximum sequence length is 1014 following (Zhang, Zhao, and LeCun 2015); smaller texts are padded with 0 while larger texts are truncated. The convolutional layers are initialized following (Glorot and Bengio 2010).

**Word-level** The embedding matrix *W* is initialized randomly with the uniform distribution between $[-0.1; 0.1]$ and is updated during model's training using backpropagation. The embedding vectors have 300 dimensions and are initialized with word2vec vectors pretrained on 100 billion words from Google News (Mikolov et al. 2013). Out-of-vocabulary words are initialized randomly. A dropout of 0.5 is used on shallow model to prevent overfitting.

| **Models** | **AGNews** | **Yelp Bin** | **Yelp Full** | **DBPedia** | **Yahoo** |
|---|---|---|---|---|---|
| Char shallow-and-wide CNN | 90.7 | 94.4 | 60.3 | 98.0 | 70.2 |
| Char-DenseNet $N_b = (4-4-4-4)$ Global Average-Pooling | 90.4 | 94.2 | 61.1 | 97.7 | 68.8 |
| Char-DenseNet $N_b = (10-10-4-4)$ Global Average-Pooling | 90.6 | 94.9 | 62.1 | 98.2 | 70.5 |
| Char-DenseNet $N_b = (4-4-4-4)$ Local Max-Pooling | 90.5 | 95.0 | 63.6 | 98.5 | 72.9 |
| Char-DenseNet $N_b = (10-10-4-4)$ Local Max-Pooling | 92.1 | 95.0 | 64.1 | 98.5 | 73.4 |
| Word shallow-and-wide CNN | 92.2 | **95.9** | **64.9** | **98.7** | 73.0 |
| Word-DenseNet $N_b = (4-4-4-4)$ Global Average-Pooling | 91.7 | 95.8 | 64.5 | **98.7** | 70.4* |
| Word-DenseNet $N_b = (10-10-4-4)$ Global Average-Pooling | 91.4 | 95.5 | 63.6 | 98.6 | 70.2* |
| Word-DenseNet $N_b = (4-4-4-4)$ Local Max-Pooling | 90.9 | 95.4 | 63.0 | 98.0 | 67.6* |
| Word-DenseNet $N_b = (10-10-4-4)$ Local Max-Pooling | 88.8 | 95.0 | 62.2 | 97.3 | 68.4* |
| bag of words (Zhang, Zhao, and LeCun 2015) | 88.8 | 92.2 | 58.0 | 96.6 | 68.9 |
| ngrams (Zhang, Zhao, and LeCun 2015) | 92.0 | 95.6 | 56.3 | 98.6 | 68.5 |
| ngrams TFIDF (Zhang, Zhao, and LeCun 2015) | 92.4 | 95.4 | 54.8 | **98.7** | 68.5 |
| fastText (Joulin et al. 2016) | **92.5** | 95.7 | 63.9 | 98.6 | 72.3 |
| char-CNN (Zhang, Zhao, and LeCun 2015) | 87.2 | 94.7 | 62.0 | 98.3 | 71.2 |
| char-CRNN (Xiao and Cho 2016) | 91.4 | 94.5 | 61.8 | 98.6 | 71.7 |
| very deep char-CNN (Conneau et al. 2016) | 91.3 | 95.7 | 64.7 | **98.7** | 73.4 |
| Naive Bayes (Yogatama et al. 2017) | 90.0 | 86.0 | 51.4 | 96.0 | 68.7 |
| Kneser-Ney Bayes (Yogatama et al. 2017) | 89.3 | 81.8 | 41.7 | 95.4 | 69.3 |
| MLP Naive Bayes (Yogatama et al. 2017) | 89.9 | 73.6 | 40.4 | 87.2 | 60.6 |
| Discriminative LSTM (Yogatama et al. 2017) | 92.1 | 92.6 | 59.6 | **98.7** | **73.7** |
| Generative LSTM-independent comp. (Yogatama et al. 2017) | 90.7 | 90.0 | 51.9 | 94.8 | 70.5 |
| Generative LSTM-shared comp. (Yogatama et al. 2017) | 90.6 | 88.2 | 52.7 | 95.4 | 69.3 |

Table 1: Accuracy of our proposed models (10 top rows) and of state-of-the-art models from the literature (13 bottom rows).

| **Dataset** | **#y** | **#train** | **#test** | **Task** |
|---|---|---|---|---|
| AGNews | 4 | 120k | 7.6k | ENC |
| Yelp Binary | 2 | 560k | 38k | SA |
| Yelp Full | 5 | 650k | 38k | SA |
| DBPedia | 14 | 560k | 70k | OC |
| Yahoo | 10 | 1 400k | 60k | TC |

Table 2: Statistics of datasets used in our experiments: number of training tokens (**#train**), of test tokens (**#test**) and of target labels (**#y**); **ENC**: English News Categorization. **SA**: Sentiment Analysis, **OC**: Ontology Classification, **TC**: Topic Classification

The shallow-and-wide CNN requires 10 hours of training on the smallest dataset, and one day on the largest. The DenseNet respectively requires 2 and 4 days for training.

## Experimental results

Table 1 details the accuracy obtained with our models (10 rows on top) and compare them with state-of-the-art results (13 rows at the bottom) on 5 corpus and text classification tasks (columns). The models from the litterature we compare to are:

- **bag of words:** The BOW model is based on the most frequent words from the training data (Zhang, Zhao, and LeCun 2015)
- **ngrams:** The bag-of-ngrams model exploits the most frequent word n-grams from the training data (Zhang, Zhao, and LeCun 2015)
- **ngrams TFIDF:** Same as the ngrams model but uses the words TFIDF (term-frequency inverse-document-frequency) as features (Zhang, Zhao, and LeCun 2015)
- **fastText:** A linear word-level model with a rank constraint and fast loss approximation (Joulin et al. 2016)
- **char-CNN:** Character-level Convolutional Network with 6 *hand-designed* CNN layers (Zhang, Zhao, and LeCun 2015)
- **char-CRNN:** Recurrent Layer added on top of a Character Convolutional Network (Xiao and Cho 2016)
- **very deep CNN:** Character-level model with 29 Convolutional Layers inspired by ResNet (Conneau et al. 2016)
- **Naive Bayes:** A simple count-based word unigram language model based on the Naive Bayes assumption (Yogatama et al. 2017)
- **Kneser-Ney Bayes:** A more sophisticated word count-based language model that uses tri-grams and Kneser-Ney smoothing (Yogatama et al. 2017)
- **MLP Naive Bayes:** An extension of the Naive Bayes word-level baseline using a two layer feedforward neural network (Yogatama et al. 2017)
- **Discriminative LSTM:** Word-level model with logistic regression on top of a traditional LSTM (Yogatama et al. 2017)
- **Generative LSTM-independent comp.:** A class-based word language model with no shared parameters across classes (Yogatama et al. 2017)
- **Generative LSTM-shared comp.:** A class-based word language model with shared components across classes (Yogatama et al. 2017)

Figure 4 visually compares the performances of 3 character-level models with 2 word-level models. Character-level models include: our shallow-and-wide CNN model

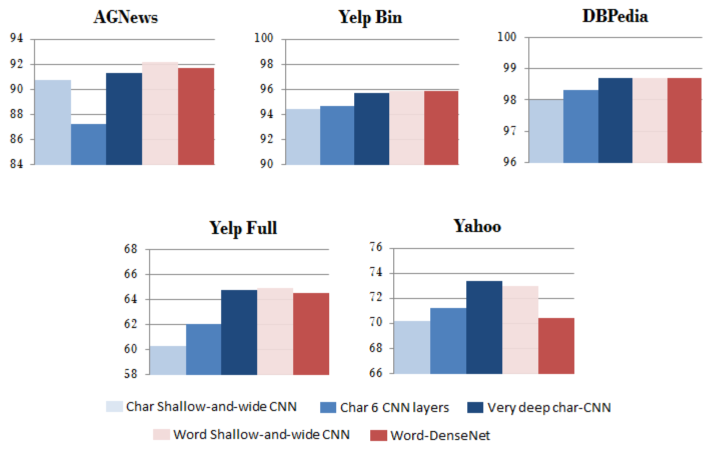

Figure 4: Comparison of character (*in blue, on the left*) and word-level (*in red, on the right*) models on all datasets. On character-level, we compare our shallow-and-wide model with the 6 CNN layers of (Zhang, Zhao, and LeCun 2015) and the 29-layers CNN of (Conneau et al. 2016). On word-level, we compare the shallow-and-wide CNN with our proposed DenseNet.

with two models on the litterature 6 CNN layers (Zhang, Zhao, and LeCun 2015), 29 CNN layers (Conneau et al. 2016). On word-level, we present our shallow-and-wide CNN with the best DenseNet $N_b = (4-4-4-4)$ using Global Average-Pooling.

The main conclusions of these experiments are threefold:

**Impact of depth for character-level models** Deep character-level models do not significantly outperform the shallow-and-wide network. A shallow-and-wide network (row 1 in Table 1) achieves 90.7%, 94.4%, 98.0% on AGNews, Yelp Bin, DBPedia respectively, comparing to 91.3%, 95.7%, 98.7% of a very deep CNN (Conneau et al. 2016). Although the deep structure achieves a slight gain in performance on these three datasets, the difference is not significant. Interestingly, a very simple shallow-and-wide CNN can get very close results to the deep 6 CNN layers of (Zhang, Zhao, and LeCun 2015) which structure must be designed meticulously.

For the smallest dataset AGNews, we suspect that the deep model **char-CNN** performs badly because it needs more data to take benefit from depth. The deep structure gives an improvement of about 4% on Yelp Full and Yahoo (first row of Table 1 vs. **very deep char CNN**), which is interesting but does not match the gains observed in image classification. We have tried various configurations: $N_f = (15, 20, 25)$, $N_f = (10, 15, 20, 25)$ and $N_f = (15, 22, 29, 36)$ on shallow models but they didn't do better.

**Impact of depth for word-level models** The DenseNet is better with 20 layers $N_b = (4-4-4-4)$ than with 32 layers $N_b = (10-10-4-4)$ and Global Average-Pooling is better than the traditional Local Max-pooling. It is the opposite to char-level. This is likely a consequence of the fact that the observed sequence length is much shorted with words than with characters.

However, the main striking observation is that all deep models are matched or outperformed by the shallow-and-wide model on all datasets, although it is still unclear whether this is because the input sequences are too short to benefit from depth or for another reason. Further experiments are required to investigate the underlying reasons of this failure of depth at word-level.

**State-of-the-art performances with shallow-and-wide word-level model** With a shallow-and-wide network on word-level, we achieved a very close state-of-the-art (SOTA) result on AGNews, SOTA on 2 datasets DBPedia, Yahoo and set new SOTA on 2 datasets: Yelp Binary and Yelp Full. We also empirically found that a word-level shallow model may outperform a very deep char-level network. This confirms that word observations are still more effective than character inputs for text classification. In practice, quick training on word-level with a simple convolutional model may already produce good result.

## Discussion

**Text representation - discrete, sparse** Very deep models do not seem to bring a significant advantage over shallow networks for text classification, as opposed to their performances in other domains such as image processing. We believe one possible reason may be related to the fact that images are represented as real and dense values, as opposed to discrete, artificial and sparse representation of text. The first convolutional operation results in a matrix (2D) for image while it results in a vector (1D) for text. The same deep network applied to two different representations (dense and sparse) will obviously get different results. Empirically, we found that a deep network on 1D (text) is less effective and learns less information than on 2D (image).

**Local vs Global max-pooling**

A global max-pooling (Collobert and Weston 2008), which retrieves the most influencial feature could already be good enough for sparse and discrete input text, and gives similar results than a local max-pooling with a deep network.

**Word vs Character level**

Char-level could be a choice but word-level is still the most effective method. Moreover, in order to use char-level representation, we *must* use a very deep model, which is less practical because it takes a long time to train.

## Conclusion

In computer vision, several works have shown the importance of depth in neural networks and the major benefits that can be gained by stacking many well-designed convolutional layers in terms of performances. However, such advantages do not necessarily transfer to other domains, and in particular Natural Language Processing, where the impact of depth in the model is still unclear. This work exploits a number of additional experiments to further explore this question and potentially bring some new insights or confirm previous findings. We further investigate another related question about which type of textual inputs, characters or words, should be chosen at a given depth. By evaluating on several text classification and sentiment analysis tasks, we show that a shallow-and-wide convolutional neural network at the word-level is still the most effective, and that increasing the depth of such convolutional models with word inputs does not bring significant improvement. Conversely, deep models outperform shallow networks when the input text is encoded as a sequence of characters, but although such deep models approach the performances of word-level networks, they are still worse on the average. Another contribution of this work is the proposal of a new deep model that is an adaptation of DenseNet for text inputs.

Based on the litterature and the results presented in this work, our main conclusion is that deep models have not yet proven to be more effective than shallow models for text classification tasks. Nevertheless, further researches should be realized to confirm or infirm this observation on other datasets, natural language processing tasks and models. Indeed, this work derives from reference deep models that have originally been developed for image processing, but novel deep architectures for text processing might of course challenge this conclusion in the near future.

## Acknowledgments

Part of the experiments realized in this work have been done on two GPU clusters: Grid5000 Inria/Loria Nancy, France and Romeo Reims, France. We would like to thank both consortiums for giving us access to their resources.

## References

Bengio, Y.; Ducharme, R.; Vincent, P.; and Janvin, C. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3:1137–1155.

Bollen, J.; Mao, H.; and Zeng, X. 2011. Twitter mood predicts the stock market. *J. Comput. Science* 2(1):1–8.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, 160–167. New York, NY, USA: ACM.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* 12:2493–2537.

Conneau, A.; Schwenk, H.; Barrault, L.; and LeCun, Y. 2016. Very deep convolutional networks for natural language processing. *CoRR* abs/1606.01781.

Costa, M. 1996. Probabilistic interpretation of feedforward network outputs, with relationships to statistical prediction of ordinal quantities. *International Journal Neural Systems* 7:627–638.

Del Corso, G. M.; Gulli, A.; and Romani, F. 2005. Ranking a stream of news. In *Proceedings of the 14th international conference on World Wide Web*, 97–106. ACM.

Dhingra, B.; Liu, H.; Cohen, W. W.; and Salakhutdinov, R. 2016. Gated-attention readers for text comprehension. *CoRR* abs/1606.01549.

Dodds, P. S.; Harris, K. D.; Kloumann, I. M.; Bliss, C. A.; and Danforth, C. M. 2011. Temporal patterns of happiness and information in a global social network: Hedonometrics and twitter. *CoRR* abs/1101.5120.

Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; and Darrell, T. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.* 9:1871–1874.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 770–778.

Hinton, G.; Deng, L.; Yu, D.; Dahl, G.; rahman Mohamed, A.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.; and Kingsbury, B. 2012. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*.

Huang, G.; Liu, Z.; and Weinberger, K. Q. 2016. Densely connected convolutional networks. *CoRR* abs/1608.06993.

Joachims, T. 1998. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, 137–142.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2016. Bag of tricks for efficient text classification. *CoRR* abs/1607.01759.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 655665.

Kaya, M.; Fidan, G.; and Toroslu, I. H. 2013. *Transfer Learning Using Twitter Data for Improving Sentiment Classification of Turkish Political News*. Cham: Springer International Publishing. 139–148.

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M. 2016. Character-aware neural language models. In *AAAI*, 2741–2749. AAAI Press.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1746–1751.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In Pereira, F.; Burges, C. J. C.; Bottou, L.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 25*, 1097–1105. Curran Associates, Inc.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 2278–2324.

McCallum, A., and Nigam, K. 1998. A comparison of event models for naive bayes text classification. In *IN AAAI-98 WORKSHOP ON LEARNING FOR TEXT CATEGORIZATION*, 41–48. AAAI Press.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 26*, 3111–3119. Curran Associates, Inc.

Miyamoto, Y., and Cho, K. 2016. Gated word-character recurrent language model. In *EMNLP*.

Nakov, P.; Rosenthal, S.; Kozareva, Z.; Stoyanov, V.; Ritter, A.; and Wilson, T. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of the 7th International Workshop on Semantic Evaluation*, 312–320.

Nakov, P.; Rosenthal, S.; Kiritchenko, S.; Mohammad, S. M.; Kozareva, Z.; Ritter, A.; Stoyanov, V.; and Zhu, X. 2016. Developing a successful semeval task in sentiment analysis of twitter and other social media texts. *Lang. Resour. Eval.* 50(1):35–65.

Qureshi, M. A.; O'Riordan, C.; and Pasi, G. 2013. Clustering with error-estimation for monitoring reputation of companies on twitter. In *Information Retrieval Technology - 9th Asia Information Retrieval Societies Conference, AIRS 2013, Singapore, December 9-11, 2013. Proceedings*, volume 8281 of *Lecture Notes in Computer Science*, 170–180. Springer.

Radford, A.; Józefowicz, R.; and Sutskever, I. 2017. Learning to generate reviews and discovering sentiment. *CoRR* abs/1704.01444.

Rosenthal, S.; Ritter, A.; Nakov, P.; and Stoyanov, V. 2014. Semeval-2014 task 9: Sentiment analysis in twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, 73–80.

Rosenthal, S.; Farra, N.; and Nakov, P. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*.

Severyn, A., and Moschitti, A. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, 959–962. New York, NY, USA: ACM.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631–1642. Seattle, Washington, USA: Association for Computational Linguistics.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*.

Xiao, Y., and Cho, K. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *CoRR* abs/1602.00367.

Yang, Z.; Dhingra, B.; Yuan, Y.; Hu, J.; Cohen, W. W.; and Salakhutdinov, R. 2017. Words or characters? fine-grained gating for reading comprehension. In *ICLR*.

Yogatama, D.; Dyer, C.; Ling, W.; and Blunsom, P. 2017. Generative and discriminative text classification with recurrent neural networks. *Arxiv*.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, 3320–3328. Cambridge, MA, USA: MIT Press.

Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, 818–833.

Zhang, Y., and Wallace, B. C. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR* abs/1510.03820.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, 649–657. Cambridge, MA, USA: MIT Press.