24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

# On the Interpretation of Convolutional Neural Networks for Text Classification

**Jincheng Xu**[1] and **Qingfeng Du**[2]

**Abstract.** A long-standing obstacle accompanying the growing popularity of convolutional neural networks (CNNs) is the lack of interpretability, which is essential to explain the decision-making process and diagnose the model's behavior. In this paper, we present a mathematical decomposition to translate the output of CNN for text classification into an ngram-level score matrix and a word-level score matrix, revealing how various parts of input sentences contribute to the final prediction quantitatively. By exploiting the derived ngram-level score matrix, we perform extensive experiments to demonstrate how n-gram features are learned via the convolutional layer. We refine previous intuitions about the behavior of filters and perform a deep investigation into their underlying properties. By leveraging the resulting word-level score matrix, we propose two visualization methods in either a global view or a local view to show how the model highlights the relative importance of inputs to arrive at a particular result. Moreover, we show how to perform adversarial attacks with word-level importance scores, and we achieve higher success rate than the baseline. Consequently, by interpreting the model in the form of score matrices, we are able to zoom in on the black boxes of CNN-based text classification models and present a comprehensive analysis of their behaviors.

## 1 Introduction

In the era of artificial intelligence, neural networks have enjoyed tremendous success and been applied in a variety of realistic applications. Convolutional Neural Network (CNN), a specialized type of neural networks originated from computer vision, has been a near-ubiquitous component of a large bulk of model architectures and has demonstrated impressive performance on image classification tasks [14]. Motivated by the recent progress in deep learning, CNN and its variants have also brought practical benefits in the task of text classification [4, 12, 38]. The distributed representations of words such as word2vec [23] or Glove [28] embed the discrete input of texts into continuous values, and subsequently the convolution operation is performed over the two-dimensional data to extract local features for the text classification task.

Despite widespread adoption, the huge number of parameters and the complex computation inside the black box of CNN make it opaque and incomprehensible. To afford transparency, model interpretability [21, 32] has been seen as a remedy. It helps elucidate the model's behavior and explain predictions in a human readable way, conveying useful information on human trust, error analysis or model improvement [29]. There has been a remarkable series of work on the interpretability of CNN for image data [7, 35, 36, 37], where a widely accepted opinion is that CNN learns visual features in a hierarchical way, from pixels to objects. However, the operating mechanism by which CNN works for text classification remains an under-explored area. The continuous image pixels are easy to be visualized, analyzed and explained directly, while the discrete texts are involved with syntactic structures and high-level semantics. While common intuitions suggest that CNN extracts important n-gram features from the input sentence to capture the key semantics and make a prediction [12, 39], this explanation seems to be too superficial without a rigorous proof. We wonder whether we can explain CNN's actions for text classification in a mathematical but easy-to-understand way.

In this paper, we perform a formal mathematical deduction on TextCNN [12], a popular, well-established architecture widely used as a building block for subsequent CNN-based text classification models, and we decompose the output of the model into an ngram-level score matrix and a word-level score matrix. The interpretable representation in the form of score matrices displays how various parts of inputs contribute to the final prediction quantitatively. Apart from that, the score matrices can also be used as useful tools to diagnose the model's inner behavior. Although we choose the architecture of TextCNN as the example here to perform the mathematical deduction for its generality and representativeness, our interpretation methodology can be instantiated on other CNN-based text classification models easily.

The contribution of our work can be summarized as follows:

(1) We perform mathematical analysis on the forward propagation process inside the model, and translate the model's output into an ngram-level score matrix and a word-level score matrix. The sum of each matrix on rows is equivalent to the output.

(2) We conduct experiments with the ngram-level score matrix to obtain an understanding of how exactly TextCNN learns n-gram features via the convolutional layer. We find that most n-gram features surviving the 1-max pooling do not contain useful information actually, while the *discriminate* n-gram features seem to latch on superficial patterns and irrelevant correlations rather than high-level semantics. What is more, the filter are *heterogeneous* in terms of the ability to capture discriminate features.

(3) We analyze TextCNN with the word-level score matrix in an end-to-end way. We introduce two visualization methods to show how the model attends to different input words to make a particular decision in heatmaps. Besides, we show how to perform adversarial attacks effectively using word-level importance scores.

## 2 Related Work

Text classification has been a fundamental task in Natural Language Processing (NLP). With the growing popularity of deep learning, there is a surge of interest in solving this task with neural networks.

[1] Tongji University, Shanghai, China, email: xujincheng@tongji.edu.cn
[2] Tongji University, Shanghai, China, email: du_cloud@tongji.edu.cn

One promising solution is CNN. Originally applied in the field of NLP in [5] to output a host of linguistic predictions, the CNN model is further improved in [12] with one convolutional layer consisting of multiple channels on word vectors. This architecture is widely referred to as TextCNN in the literature, and it builds the foundation for the development of almost all CNN-based text classification models, such as *the dynamic CNN* [11] equipped with a dynamic k-max pooling mechanism to deal with variable-length sentences, *the character-level CNN* [38] which learns features at the character level, *the very deep CNN* [6] with up to 29 convolutional layers on characters, *the deep pyramid CNN* [10] which has 15 weight layers but low computational complexity, *the densely connected CNN* [34] using dense connections and multi-scale feature attention to extract multi-scale features, or *the adaptive CNN* [4] whose filters are adaptively generated conditioned on inputs. For a comparison between different CNN-based models for text classification, see [15].

Even though deep learning models have been widely deployed in real-world applications, the nested opaqueness leads to a wide mistrust on their predictions. The inability to explain their behaviors has gained increasing attention nowadays [21]. Researchers turn to *interpretability* to shed light on the inner workings of these black-box models. A wide variety of explanatory methods have been proposed for neural networks. Some recent work exploit the forward and backward passes to identify the relevance between input features and final decisions in *any* non-linear structure. Sensitivity analysis [3, 33] computes the norm over partial derivatives with the back-propagation algorithm [31] and identifies input sensitivities locally. Layer-wise relevance propagation (LRP) [2] redistributes the unit-level relevance between any two adjacent layers iteratively, until the model's output is decomposed into signed scores for each input unit. The leave-one-out method [19, 27] measures the word-level importance scores by observing the change in loss when erasing a particular word from the original input. In contrast to our work which can also capture the interactions between input units (e.g., the ngram-level analysis consisting of multiple input words, which we will describe in detail later), these methods are usually limited to unit-level analysis.

Our work is closely related to model-specific methods, which aim to expose the inner details of a specific kind of neural networks. For example, LSTM [8] is a basic building block for a large number of model architectures, and the interpretation of LSTM has attracted great interest. LSTM is distilled into rule-based classifiers to identify important patterns of words in [25]. And in [24], contextual decomposition is introduced to extract interactions between various parts of the input sentence in LSTM.

Another line of work on model-specific methods focus on the explanation of CNN-based model architectures. Despite notable advancements have been achieved in computer vision [7, 35, 36, 37], previous work on interpretable CNN-based text classification models are relatively rare. In [17], various explanatory methods along with human-grounded evaluations are considered on CNNs for text classification. The most similar prior work to ours is [9], where a deep investigation is performed into how CNN processes text, but with significant differences: [9] investigates the pooled vector directly in TextCNN and capitalizes on the properties of n-grams, while we show the mathematical deduction on the model architecture and analyze the instance-wise explanations in both word-level and ngram-level. What is more, based on the derived score matrices, we conduct comprehensive experiments in terms of filter behaviors, visualization methods and adversarial potentials to emphasize the last mile of interpretability in TextCNN. Our work can be seen as a complement to existing work.

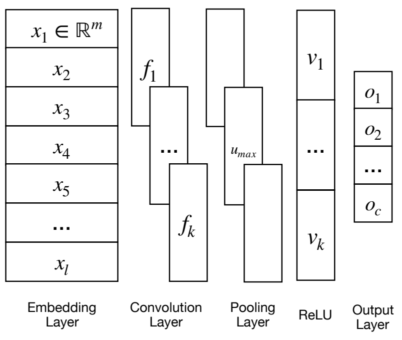


**Figure 1.** The model architecture of TextCNN.

## 3 Model Interpretations

In this section, we perform the decomposition based on algebraic transformations to translate the output of TextCNN into an ngram-level score matrix and a word-level score matrix. Before we present the analysis, we introduce a set of formal definitions and notations to describe the forward propagation process in TextCNN.

### 3.1 Formal Description

Given a set of documents $\mathcal{D} = \{(\mathbf{d}, y)\}$ and $m$-dimensional word embeddings, each word in the document $\mathbf{d} \in \mathcal{D}$ is encoded as an $m$-dimensional vector and the input can be embedded as $\boldsymbol{x} \in \mathbb{R}^{l \times m}$ where $l$ is the length of $\mathbf{d}$ after padding.

The input $\boldsymbol{x}$ will be fed into the convolutional layer to extract n-gram features via multiple kinds of filters. More specifically, the computation through the convolution operation for $n$-grams can be described as follows:

$$\begin{aligned} u_i &= f(\boldsymbol{x_{i:i+n-1}}) \\ &= \boldsymbol{W_f} \odot \boldsymbol{x_{i:i+n-1}} + b_f \\ u_{max} &= max\{u_1, u_2, \cdots, u_{l-n+1}\} \\ v &= ReLU(u_{max}) \end{aligned} \tag{1}$$

where $\boldsymbol{W_f} \in \mathbb{R}^{n \times m}$ and $\boldsymbol{b_f} \in \mathbb{R}^1$ represent the weight and bias of the filter $f$, $1 \leq i \leq l - n + 1$, $\odot$ represents the dot product, the $max$ operation represents the 1-max pooling, $v$ represents the univariate n-gram feature after pooling and the non-linear transformation ReLU [26].

Assuming we have $k$ different filters in total. The output $v$ for each filter will be concatenated together to obtain a feature vector of size $k$. For convenience, we refer to the penultimate layer as $\mathbf{h} \in \mathbb{R}^k$. A fully connected layer will be applied over $\mathbf{h}$ to generate the final output $\mathbf{o}$:

$$\mathbf{o} = \boldsymbol{W_y} \times \mathbf{h} + \boldsymbol{b_y} \tag{2}$$

where $\boldsymbol{W_y} \in \mathbb{R}^{c \times k}$ and $\boldsymbol{b_y} \in \mathbb{R}^c$ represent the weight and bias separately, and $\mathbf{o} \in \mathbb{R}^c$. The notation $c$ denotes the number of all possible document classes in the dataset. We have illustrated the model architecture of TextCNN in Figure 1.

### 3.2 Ngram-level Score Matrix

We begin this section by defining what is an ngram-level score matrix. The ngram-level score matrix $\mathbf{A}$ is a $k \times c$ matrix where $k$ represents the number of n-gram features in the penultimate layer $\mathbf{h}$, and $c$ is the number of predefined classes.

Each hidden unit $v \in \mathbf{h}$ corresponds to the value of an n-gram feature in the original input $\boldsymbol{x}$, and Equation (2) can be further expressed in details as below:

$$\begin{aligned} \mathbf{o} &= \boldsymbol{W_y} \times \mathbf{h} + \boldsymbol{b_y} \\ &= \begin{bmatrix} w_{1,1} & \cdots & w_{1,k} \\ \vdots & \ddots & \vdots \\ w_{c,1} & \cdots & w_{c,k} \end{bmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_k \end{pmatrix} + \begin{pmatrix} b_1 \\ \vdots \\ b_c \end{pmatrix} \\ &= \begin{pmatrix} o_1 & o_2 & \cdots & o_c \end{pmatrix}^{\mathrm{T}} \end{aligned} \tag{3}$$

The contribution of a particular n-gram feature $v_p \in \mathbf{h}$ to the prediction of the $q$th class can be calculated as follows:

$$\mathbf{A}[p][q] = w_{q,p} \times v_p + b_q/k \tag{4}$$

where the bias term $b_q$ is averaged over all hidden units in $\mathbf{h}$ to indicate their separate contributions. As we can see, the sum of scores along the $q$th column in $\mathbf{A}$ equals to the $q$th unit in the output layer $\mathbf{o}$ according to Equation (3) and (4):

$$\begin{aligned} \sum_{p=1}^{k} \mathbf{A}[p][q] &= \sum_{p=1}^{k} (w_{q,p} * v_p + b_q/k) \\ &= \sum_{p=1}^{k} (w_{q,p} \times v_p) + b_q \\ &= o_q \end{aligned} \tag{5}$$

In this way, given an arbitrary input $\boldsymbol{x}$, the output of TextCNN can be interpreted as the sum of ngram-level importance scores, where each score captures the importance of an individual n-gram feature for the prediction of a particular class. Inspired by the idea of unpooling [35] which records the location of the maximum value in the pooling region, we can retrieve the n-gram in the original input $\boldsymbol{x}$ for each $v \in \mathbf{h}$, and consequently make the ngram-level score matrix in a human readable way.

## 3.3 Word-level Score Matrix

The definition of word-level score matrix, denoted as $\mathbf{B}$ for convenience, is similar to the definition of $\mathbf{A}$. $\mathbf{B}$ is a $l \times c$ matrix where $l$ is the length of the original input $\boldsymbol{x}$. While the matrix $\mathbf{A}$ captures the importance score with respect to each n-gram surviving the 1-max pooling, $\mathbf{B}$ captures the importance score for each word in the input $\boldsymbol{x}$. In fact, the key insight behind the computation of $\mathbf{B}$ is the segmentation and re-organization of each n-gram feature $v \in \mathbf{h}$.

Suppose we have a filter $f$ to extract the n-gram feature from the input $\boldsymbol{x}$. According to the first step of Equation (1), even though the computation between $f$ and $\boldsymbol{x_{i:i+n-1}}$ will result in a singleton local feature $u_i$, each word in $\boldsymbol{x_{i:i+n-1}}$ participates in the dot product separately:

$$\begin{aligned} u_i &= \mathbf{W}_f \odot \boldsymbol{x_{i:i+n-1}} + b_f \\ &= \begin{pmatrix} \boldsymbol{w_1} & \cdots & \boldsymbol{w_n} \end{pmatrix}^{\mathrm{T}} \odot \begin{pmatrix} \boldsymbol{x_i} & \cdots & \boldsymbol{x_{i+n-1}} \end{pmatrix}^{\mathrm{T}} + b_f \\ &= \sum_{j=1}^{n} \boldsymbol{w_j} \odot \boldsymbol{x_{i+j-1}} + b_f \\ &= \sum_{j=1}^{n} (\boldsymbol{w_j} \odot \boldsymbol{x_{i+j-1}} + b_f/n) \\ &= \sum_{j=1}^{n} u_{i,j} \end{aligned} \tag{6}$$

where $u_{i,j}$ is the word-level information contributed by the $(i+j-1)$th word in $\boldsymbol{x}$ to the $i$th n-gram feature $u_i$.

Note that after 1-max pooling, most word-level information captured by the filter $f$ will be discarded, which actually does not contribute to the final prediction. Besides, according to Equation (1), if the value of $u_{max}$ is less than zero, the information will be totally discarded by ReLU activation, and we update the value of $u_{i,j} \in u_{max}$ to zero. Otherwise, $u_{i,j}$ will keep its original value unchanged. In either case, the univariate n-gram feature $v$ can be represented as follows:

$$v = \sum_{j=1}^{n} u_{i,j} \quad s.t. \;\; v \geq 0 \tag{7}$$

where we use the index $i$ to record the position of $u_{max}$ in the input $\boldsymbol{x}$.

We now discuss how to obtain the word-level score matrix via $v$. Using Equation (7), we denote $v_p \in \mathbf{h}$ as $\sum_{j=1}^{n}(u_{i,j})_p$, and Equation (3) can be written as follows:

$$\begin{aligned} \mathbf{o} &= \boldsymbol{W_c} \times \mathbf{h} + \boldsymbol{b_c} \\ &= \begin{bmatrix} w_{1,1} & \cdots & w_{1,k} \\ \vdots & \ddots & \vdots \\ w_{c,1} & \cdots & w_{c,k} \end{bmatrix} \begin{pmatrix} \sum_{j=1}^{n}(u_{i,j})_1 \\ \vdots \\ \sum_{j=1}^{n}(u_{i,j})_k \end{pmatrix} + \begin{pmatrix} b_1 \\ \vdots \\ b_c \end{pmatrix} \\ &= \begin{pmatrix} o_1 & o_2 & \cdots & o_c \end{pmatrix}^{\mathrm{T}} \end{aligned} \tag{8}$$

Note that a diversity of $n$ (e.g., $n = 3, 4, 5$) will be pre-defined to capture local features in multiple semantic levels in realistic settings. We omit the different representations of $n$ here for simplicity. The contribution of the $j$th word in $v_p$ to the prediction of the $q$th class can be calculated as follows:

$$\begin{aligned} idx &= i_p + j - 1 \\ \mathbf{B}[idx][q]_p &= w_{q,p} \times (u_{i,j})_p + b_q/(k \times n) \end{aligned} \tag{9}$$

where $i_p \in [1, l-n+1]$ records the index of the extracted n-gram $v_p$ in the input $\boldsymbol{x}$, and thus $idx$ is the index of the targeted word in $\boldsymbol{x}$. The deduction of position representations has been presented in Equation (6). $\mathbf{B}[idx][q]_p$ is a component which quantitatively describes how the $idx$th word in $\boldsymbol{x}$ contributes to the prediction of $c_q$ through the $p$th filter particularly, since other filters will also possibly capture the $idx$th word in the univariate n-gram feature $v$.

We sum up all the components along $k$ filters to compute $\mathbf{B}[idx][q]$:

$$\mathbf{B}[idx][q] = \sum_{p=1}^{k} \mathbf{B}[idx][q]_p \tag{10}$$

Note that the $idx$th word is not necessarily captured by each univariate n-gram feature $v$. Denoting the set of $n$ words captured by $v_p$ as $\mathbb{E}_p$, we set $\mathbf{B}[idx][q]_p = 0$ if $\boldsymbol{x_{idx}} \notin \mathbb{E}_p$. On the other hand, not all words in $\mathbf{x}$ will be captured within a certain n-gram feature $v$. We initialize each cell in $\mathbf{B}$ with zero, meaning that if there is a particular word discarded by all filters, it contributes nothing to the final prediction. Similar to the ngram-level score matrix $\mathbf{A}$, the sum of scores along the $q$th column in $\mathbf{B}$ equals to the $q$th unit in the output layer $\mathbf{o}$. According to Equation (8), (9) and (10), the proof is as follows:

$$\begin{aligned}
\sum_{idx=1}^{l} \mathbf{B}[idx][q] &= \sum_{idx=1}^{l}\sum_{p=1}^{k} \mathbf{B}[idx][q]_p \\
&= \sum_{p=1}^{k}\sum_{idx=1}^{l} \mathbf{B}[idx][q]_p \\
&= \sum_{p=1}^{k}\Big(\sum_{idx\in\mathbb{E}_p} \mathbf{B}[idx][q]_p + \sum_{idx\notin\mathbb{E}_p} 0\Big) \\
&= \sum_{p=1}^{k}\sum_{j=1}^{n} \mathbf{B}[idx][q]_p \\
&= \sum_{p=1}^{k}\sum_{j=1}^{n} (w_{q,p} \times (u_{i,j})_p + b_q/(k \times n)) \\
&= \sum_{p=1}^{k} (w_{q,p} \times \sum_{j=1}^{n} (u_{i,j})_p) + b_q \\
&= o_q
\end{aligned} \tag{11}$$

The word-level score matrix $\mathbf{B}$ is sensitive to word positions. That is, if two identical words occur in different positions in the input $\boldsymbol{x}$, their contributions to the output will be different. This property demonstrates that TextCNN will learn positional information of words during training.

The algebraic operations to generate both score matrices are fully deterministic, so the output of TextCNN for a certain input $\boldsymbol{x}$ should be exactly equal to the results obtained from Equation (5) or Equation (11). In this way, each score in $\mathbf{A}$ or $\mathbf{B}$ can be interpreted as the contribution of an individual unit to the final prediction. Besides, due to the fact that the model's output is decomposed into a linear accumulation of separate scores, the relative importance between different units in the same matrix can be compared linearly. Examples of score matrices have been displayed in Figure 2.

| | Positive | Negative |
|---|---|---|
| What | 0 | 0 |
| a | 0.262 | -0.176 |
| good | 1.020 | -1.581 |
| movie | 0.121 | -0.106 |
| sum | 1.403 | -1.863 |

(a) The word-level score matrix.

| | Positive | Negative |
|---|---|---|
| a good | 0.393 | -0.634 |
| a good | 0.549 | -0.596 |
| good movie | 0.461 | -0.633 |
| sum | 1.403 | -1.863 |

(b) The ngram-level score matrix where $n = 2$ and $k = 3$.

| | Positive | Negative |
|---|---|---|
| the output layer $\mathbf{o}$ | 1.403 | -1.863 |

(c) The output layer.

**Figure 2.** Examples of score matrices derived from "What a good movie". "Positive" and "Negative" are the class names of binary sentiment analysis. It is possible that "a good" appears twice in (b), because two filters can capture this n-gram at the same time.

## 4 Experiments

In this section, we perform extensive experiments to analyze the black-box of TextCNN in details leveraging the derived ngram-level and word-level score matrices.

### 4.1 Preliminaries

We conduct our experiments on two well-established text classification datasets: (a) **AG's News** [38], a topic classification dataset including 4 different news categories with 30k training samples and 7k test samples for each class. (b) **IMDB** [22], a binary sentiment analysis dataset on movie reviews containing 25k training samples and 25k test samples evenly split for each polarity. All the results are reported on the test set.

In terms of training details, we use the pre-trained 100-dimensional Glove embeddings to encode input documents. If not specified otherwise, we set three filter region sizes: 3, 4 and 5, with 50 feature maps for each one. The learning rate is set to 0.01 and 0.005 for AG's News and IMDB respectively, and the mini-batch size is set to 128. The parameters are updated with the Adam optimizer [13].

### 4.2 Experiments I: Ngram-level Analysis

Previous studies [12, 39] provide an intuitive explanation for the function of different filters in TextCNN that a filter is responsible for capturing the most important n-gram feature in each feature map. In fact, this explanation is not strictly accurate. In this set of experiments, we offer a deep investigation and ameliorate previous explanations on the behavior of filters by answering the following three questions:

- **RQ1:** How many n-gram features are set to zero by the ReLU function?
- **RQ2:** How many n-gram features have negligible effects on the final decision?
- **RQ3:** How many filters are involved in the extraction of discriminative n-gram features?

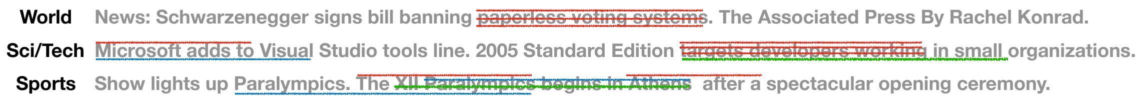
We use the ngram-level score matrix $\mathbf{A}$ to interpret the model's behavior and calculate the test accuracy here.

To answer **RQ1**, we train a set of models on both datasets, varying the number of filters from 30 to 600 which are evenly distributed on each region size. If the scalar $u_{max}$ extracted from a feature map is a minus, it will be set to zero by ReLU and thus contribute nothing to the output. We report the original test accuracy as well as the average ratio of scalars discarded by ReLU (*@Ratio1*) in Table 1. It can be observed that a large number of n-gram features do not survive the ReLU activation actually, for example, as much as 84.34% n-gram features are discarded when $k = 600$. Strictly speaking, these uninformative features can still exert a nearly negligible influence on the output with the bias $b_q/(k \times n)$, according to Equation (9).

To answer **RQ2**, we need to measure whether an n-gram feature $v_p$ is informative for classification. We define the *discrimination* of $v_p$ as its impact on the final decision with **no** prior knowledge about the

**Table 1.** The relationship between test accuracy, filters and n-gram features. By @Ratio1, the numerator is the number of $u_{max}$ discarded by ReLU, and the denominator is the number of all $u_{max}$; By @Ratio2, the numerator is the number of $v_p$ removed by $Var(v_p)$, and the denominator is the number of all $v_p$; By @Ratio3, the numerator is the number of filters capable of capturing discriminative features, and the denominator is the number of all filters.

| $k$ | | | 30 | 75 | 150 | 225 | 300 | 450 | 600 |
|---|---|---|---|---|---|---|---|---|---|
| AG's News | RQ1 | Test Accuracy | 90.03% | 90.91% | 90.25% | 90.53% | 89.88% | 88.95% | 89.74% |
| | | @Ratio1 | 29.00% | 54.36% | 66.82% | 71.96% | 80.08% | 81.46% | 84.34% |
| | RQ2 | Test Accuracy | 90.29% | 90.09% | 89.64% | 90.11% | 89.50% | 90.50% | 89.61% |
| | | @Ratio2 | 74.41% | 89.01% | 94.76% | 96.34% | 97.37% | 98.94% | 98.48% |
| | RQ3 | @Ratio3 | 100.00% | 77.33% | 68.66% | 51.11% | 49.00% | 46.00% | 40.16% |
| IMDB | RQ1 | Test Accuracy | 85.88% | 86.06% | 84.90% | 85.49% | 85.90% | 86.69% | 84.68% |
| | | @Ratio1 | 6.88% | 21.77% | 24.46% | 22.92% | 46.40% | 27.45% | 60.09% |
| | RQ2 | Test Accuracy | 86.05% | 86.04% | 84.67% | 84.61% | 83.01% | 82.76% | 84.54% |
| | | @Ratio2 | 18.69% | 52.39% | 60.06% | 80.69% | 90.01% | 90.88% | 95.80% |
| | RQ3 | @Ratio3 | 96.67% | 74.67% | 78.00% | 52.44% | 27.67% | 35.11% | 13.67% |



| | |
|---|---|
| **World** | News: Schwarzenegger signs bill banning ~~paperless voting systems~~. The Associated Press By Rachel Konrad. |
| **Sci/Tech** | Microsoft adds to Visual Studio tools line. 2005 Standard Edition ~~targets developers working~~ in small organizations. |
| **Sports** | Show lights up Paralympics. The ~~XII Paralympics begins in Athens~~ after a spectacular opening ceremony. |

**Figure 3.** Examples of n-gram features satisfying $Var(v_p) > \epsilon$. Different colors indicate different region sizes, e.g., three 3-grams satisfying the condition are highlighted in the first example. The predicted class has been indicated on the left.

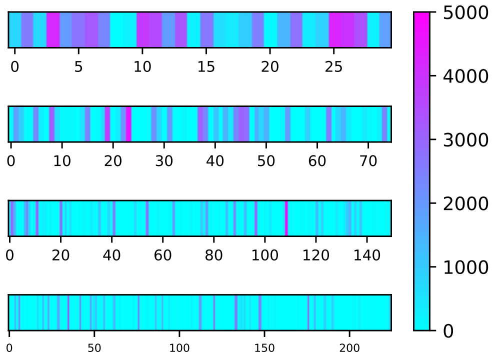


**Figure 4.** The distribution of discriminative n-gram features in **h**. The horizontal axis represents the position of filters in **h**, and the vertical color bar shows the correspondence between the number of discriminative features and the particular color. We report the results when $k \in \{30, 75, 150, 225\}$.

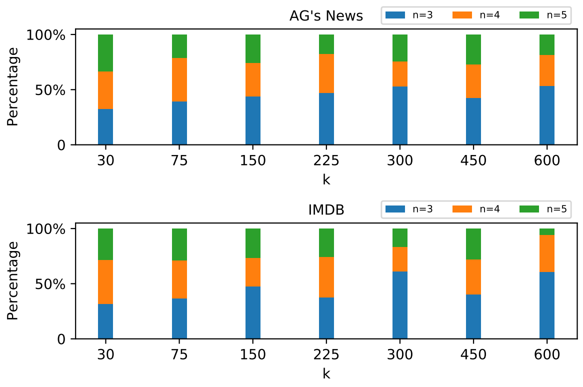


**Figure 5.** The percentage of discriminative features extracted by different region sizes.

correct label. Generally speaking, a discriminative $v_p$ contributes to a highly unbalanced distribution of prediction probability. Consider an extreme case. If $v_p$ corresponds to a uniform distribution in the ngram-level score matrix $A$ where $\mathbf{A}[p][1] = \cdots = \mathbf{A}[p][c]$, it can not make a fair prediction with high confidence, and thus it is far away from being discriminative.

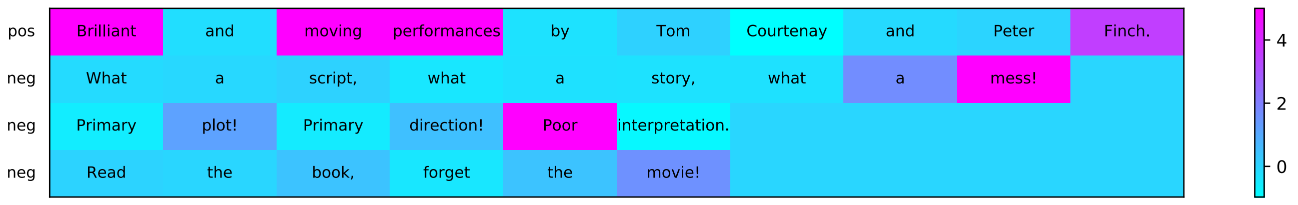
We introduce the definition of variances ($Var$) to measure the discrimination for $v_p$ quantitatively as follows:

$$\mu = (\mathbf{A}[p][1] + \cdots + \mathbf{A}[p][c])/c$$
$$Var(v_p) = \sum_{q=1}^{c} (\mathbf{A}[p][q] - \mu)^2/c \tag{12}$$

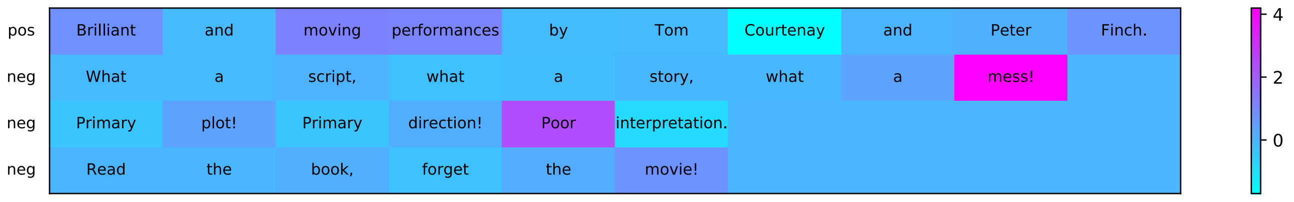
And a threshold $\epsilon$ has to be pre-defined to split the features into two independent sets based on $Var(v_p)$. Note that the features with $Var(v_p) > \epsilon$ will be referred to as discriminative features throughout the rest of the paper.

We report the results in Table 1 in terms of test accuracy after removing all features satisfying $Var(v_p) \leq \epsilon$ from **A** and the average ratio (@*Ratio2*) of features removed by $Var(v_p)$. We experimentally assign $\epsilon = 0.1$ and 0.005 for AG's News and IMDB respectively, so that the features can be removed as much as possible while the test accuracy obtained from **A** is maintained. As we can see, the performance just yields a slight degradation compared to the original test accuracy reported in **RQ1**. Specially, the accuracy even increases sometimes (e.g., when $k = 450$ on AG's News). We attribute this observation to the fact that although many filters are involved in the convolution operation to extract n-gram features, most of them have negligible effects on the prediction. In contrast, several particular n-gram features are predictive enough. We display several examples in Figure 3 for better understandings, where the colored lines indicate the discriminative features captured by filters.

To answer **RQ3**, we should expect there to be a high order corre-



(a) Visualization of features in a global view.



(b) Visualization of features in a local view.

**Figure 6.** Visualize how TextCNN interprets an input $x$ in heatmaps. Best viewed in color.

lation between filters and discriminative n-gram features. We display the average ratio (*@Ratio3*) of filters which capture discriminative n-gram features successfully in Table 1. Note that the test accuracy here is exactly same to the results in **RQ2** since we just report the results from another perspective. We observe an interesting property from the results that *@Ratio3* keeps decreasing when the number of filters increases. For example, on the IMDB dataset, 96.67% filters manage to capture discriminative features when $k = 30$, while the percentage is only 13.67% when $k = 600$. It seems that while most filters are useless, a small number of them are enough to collect useful information for predictions. In fact, according to our observations during experiments, some filters even take the consecutive "empty" tokens (used to pad the sentences and make their lengths equal) which have no semantic meaning for text classification as important n-gram features surviving the 1-max pooling, and these are obviously not the predictive features in each feature map.

To further investigate how each filter contributes to feature extraction in the convolutional layer, we visualize the distribution of all discriminative n-gram features captured from AG's News in **h**, as shown in Figure 4. As we can see, the number of discriminative features captured by different filters varies greatly, from zero to thousands. And most of them are captured by several particular filters. Apart from this, we also visualize the distribution of discriminative n-gram features with respect to different filter region sizes in Figure 5. It seems that 3-grams capture more discriminative features that 5-grams. A possible reason is that n-grams with a smaller $n$ will occur more frequently in the corpus, and thus it is more possible to be identified as predictive features.

From the results, it can be concluded that not all n-gram features can be regarded as predictive equally for text classification, and the filters in TextCNN are not *homogenous* in terms of the ability to extract features. From **RQ1**, we can see that a large proportion of n-gram features are not informative in the fully connected layer, since they have been set to zero by ReLU. Associated with the test accuracy displayed in Table 1, this discovery takes a further step towards the better understanding of hyperparameter finetuning, that more filters do not necessarily bring performance gains when their resulting n-gram features are deactivated by ReLU. From **RQ2**, a large number of n-gram features managing to pass ReLU still have negligible effects on the classification results, for the reason that they fail to capture discriminate n-grams. What is more, the qualitative analysis shown in Figure 3 reveals that most captured n-grams do not appear semantic or significant to human eyes. A possible explanation is that the model still relies on superficial patterns and irrelevant correlations in the dataset to make predictions, rather than high-level semantics. From **RQ3**, we empirically show that the heterogeneity of filters can be reflected in two aspects. On the one hand, some filters capture no discriminate n-gram features. On the other hand, according to Figure 4 and 5, for those filters capable of detecting predictive features, their abilities are not equivalent with respect to the number of identified n-grams.

Taking the answers for all three questions into consideration, we refine previous explanations and provide meaningful evidence about the behavior of filters.

## 4.3 Experiments II: Word-level Analysis

Unlike the ngram-level score matrix which can be derived from the model directly, the word-level score matrix comes from the segmentation and re-organization of n-gram features. It displays how TextCNN attends to various parts of inputs and offers an end-to-end explanation. In this section, we attempt to answer the following research questions to understand the behavior of TextCNN in a word level:

- **RQ1:** How to visualize the features learned by TextCNN?
- **RQ2:** How to perform adversarial attacks on TextCNN based on word-level importance scores?

To answer **RQ1**, we put forward two different visualization methods, in either a global view or a local view. With the word-level score matrix **B**, we are able to visualize how TextCNN learns various features in a result-oriented way. Suppose we have an input $\boldsymbol{x}$ with its correct label, and a fully trained TextCNN. In the global method, denoting the calculation of cross entropy as $H$, we define the Leave-

**Table 2.** Results of white-box non-targeted adversarial attacks on TextCNN in attack success rate.

| $k$ | | 30 | 75 | 150 | 225 | 300 | 450 | 600 |
|---|---|---|---|---|---|---|---|---|
| AG's News | Text-fool | 61.29% | 73.89% | 69.82% | 75.87% | 76.84% | 76.82% | 73.37% |
| | Our Method | 66.47% | 75.87% | 73.51% | 79.64% | 77.80% | 78.76% | 77.71% |
| IMDB | Text-fool | 29.90% | 31.46% | 38.98% | 35.09% | 33.19% | 28.60% | 38.13% |
| | Our Method | 60.41% | 62.04% | 64.96% | 60.50% | 66.47% | 63.99% | 65.62% |

one-out Score (*LS*) for each word $\boldsymbol{x_i} \in \boldsymbol{x}$ as follows:

$$s = \frac{H(\boldsymbol{o} - \boldsymbol{B}[\mathbf{x}_i], \boldsymbol{y}) - H(\boldsymbol{o}, \boldsymbol{y})}{H(\boldsymbol{o}, \boldsymbol{y})}$$
$$LS(\boldsymbol{x}_i) = \begin{cases} max\{s, \lambda\}, s >= 0 \\ min\{s, -\lambda\}, s < 0 \end{cases} \tag{13}$$

where $\mathbf{o} \in \mathbb{R}^c$ is the output layer and $\mathbf{y} \in \mathbb{R}^c$ is the one-hot vector encoding the label. The basic idea is that the global contribution of $\boldsymbol{x_i}$ can be measured by the relative change in loss after we remove the row of $\boldsymbol{x_i}$ in the word-level score matrix **B**. Figure 6(a) illustrates several examples setting $\lambda = 5$. As we can see, the scores can be negative, indicating words can have a negative impact on the decision.

Our method is similar to what is proposed in [19], where the word-level importance is visualized by the change of log-likelihood when erasing a word from $\boldsymbol{x}$, but with two important differences: (1) The intrinsic properties of $\boldsymbol{x}$ such as length, positions or n-grams will be changed after word erasure, which may impact the measure of word-level scores. In contrast, the relative importance of words shown in **B** is static, which strengthens the variable-controlling ability in visualization. (2) In some cases, the absolute value of loss change may be extremely large, causing other values indistinguishable in the continuous color bar. We scale down the loss change by $H(\mathbf{o}, \mathbf{y})$ and clip the extreme values to optimize the visual effect.

In the local method, we zoom in on the word-level importance scores with respect to the correct label only, rather than take its overall predictive capability into consideration. More concretely, we transpose the column of the correct label in **B**, and visualize the word-level score vector. Examples are shown in Figure 6(b). Compared to the global view, the local view demonstrates a smoother color variation. While the relationship between word importance and the numerical value of *LS* is monotonically increasing but non-linear in Equation (13), the value represented in the local view can be directly translated into its contribution in a linear way.

In a word, both visualization methods manage to translate the output of TextCNN into visible word-level scores, and help us understand how the model makes the prediction.

**RQ2** puts forward a new perspective on the relationship between model's interpretation and adversarial examples. While it is natural to perform adversarial attacks to get an insight into the decision-making process inside the black box [1, 30], we can also reverse the operation and use the model's interpretation to assist in the generation of adversarial examples.

Here we demonstrate how to perform adversarial attacks based on our interpretation of TextCNN. Firstly we introduce a baseline algorithm, the word-level modification strategy proposed in [20] (or *Text-fool* for brevity) designed to perform non-targeted adversarial attacks on text classification models. It computes gradient magnitudes to identify important words (similar to sensitivity analysis, as mentioned in Section 2), and uses the typo-based perturbations to transform these words into out-of-the-vocabulary words, e.g., "file" to "fi1e". In this way, Text-fool adds imperceptible perturbations on the original input and affects the model's prediction by manipulating sensitive words.

In our method, we use the model-specific word-level importance scores derived from CNN's interpretation instead of the model-agnostic sensitivity analysis widely used in the generation of adversarial texts [16, 18], and we keep the following typo-based perturbations similar to Text-fool. In more details, suppose the correct label is $y$, we use the numerical value of $\mathbf{B}[idx][y]$ to identify important words. The intuitive explanation is that larger positive scores indicate more contributions to the prediction of $y$, and the perturbations on these words will bring significant changes in loss. The maximum allowed valid perturbation has been limited to 10 for both methods in our experiments. The results are reported in Table 2.

It can be clearly seen that our method performs notably better than the baseline in all the cases. The performance boost is especially remarkable on the IMDB dataset. This is mainly because sentimental words have a significant impact on sentiment analysis, and our method can identify these positive or negative words accurately, as shown in Figure 6. The higher success rate indicates that our interpretation of CNN has better explanatory ability in comparison with sensitivity analysis. Besides, the matrix **B** shows signed numerical scores for each word proportional to their contributions to the prediction, and thus provide human readable explanations when performing adversarial attacks.

We posit that the matrix **B** can also be used in other adversarial attack algorithms by identifying a subset of ranked words, while the following perturbation schemes can remain the same. Moreover, we can generalize the idea of word-level score matrix to other CNN-based text classification models, meaning a broader range of targeted models to attack. We leave the experimental evaluation to future work.

## 5 Conclusions

In this paper, we present a mathematical deduction to decompose the output of TextCNN into an ngram-level score matrix and a word-level score matrix, the value of which can be interpreted as the contribution of individual input units to the prediction of a particular class. By exploiting the information conveyed by score matrices, we conduct extensive experiments on two publicly available text classification datasets for a deep investigation into the model's behavior. We describe the behavior of filters when they extract n-gram features, visualize what the model has learned to make a prediction, and show how to perform adversarial attacks on the model with word-level importance scores. Future research directions include generalizing the interpretation methodology of TextCNN to other CNN-based text classification models, or leveraging the instance-wise explanations in the form of score matrices to investigate the model in other aspects, such as revealing confounding factors, optimizing model architectures or performing error analysis.

## REFERENCES

[1] David Alvarez-Melis and Tommi S. Jaakkola, 'A causal framework for explaining the predictions of black-box sequence-to-sequence models', in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 412–421, (2017).

[2] Sebastian Bach, Alexander Binder, Gregoire Montavon, Frederick Klauschen, Klausrobert Muller, and Wojciech Samek, 'On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation', *PLOS ONE*, **10**(7), 0130140, (2015).

[3] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller, 'How to explain individual classification decisions', *J. Mach. Learn. Res.*, **11**, 1803–1831, (2010).

[4] Byung-Ju Choi, Jun-Hyung Park, and SangKeun Lee, 'Adaptive convolution for text classification', in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2475–2485, (2019).

[5] Ronan Collobert and Jason Weston, 'A unified architecture for natural language processing: deep neural networks with multitask learning', in *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML)*, pp. 160–167, (2008).

[6] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann LeCun, 'Very deep convolutional networks for natural language processing', *CoRR*, **abs/1606.01781**, (2016).

[7] Alexey Dosovitskiy and Thomas Brox, 'Inverting visual representations with convolutional networks', in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 4829–4837, (2016).

[8] Sepp Hochreiter and Jurgen Schmidhuber, 'Long short-term memory', *Neural Computation*, **9**(8), 1735–1780, (1997).

[9] Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg, 'Understanding convolutional neural networks for text classification', in *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP*, pp. 56–65, (2018).

[10] Rie Johnson and Tong Zhang, 'Deep pyramid convolutional neural networks for text categorization', in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 562–570, (2017).

[11] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom, 'A convolutional neural network for modelling sentences', in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*, pp. 655–665, (2014).

[12] Yoon Kim, 'Convolutional neural networks for sentence classification', in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 1746–1751, (2014).

[13] Diederik P. Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', in *3rd International Conference on Learning Representations, ICLR*, (2015).

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, 'Imagenet classification with deep convolutional neural networks', in *26th Annual Conference on Neural Information Processing Systems, NIPS*, pp. 1106–1114, (2012).

[15] Hoa T. Le, Christophe Cerisara, and Alexandre Denis, 'Do convolutional networks need to be deep for text classification ?', in *The Workshops of the The Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 29–36, (2018).

[16] Qi Lei, Lingfei Wu, Pin Yu Chen, Alexandros G. Dimakis, Inderjit S. Dhillon, and Michael Witbrock, 'Discrete adversarial attacks and submodular optimization with applications to text classification', *arXiv: Machine Learning*, (2018).

[17] Piyawat Lertvittayakumjorn and Francesca Toni, 'Human-grounded evaluations of explanation methods for text classification', in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pp. 5194–5204, (2019).

[18] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang, 'Textbugger: Generating adversarial text against real-world applications', in *26th Annual Network and Distributed System Security Symposium, NDSS*, (2019).

[19] Jiwei Li, Will Monroe, and Dan Jurafsky, 'Understanding neural networks through representation erasure', *CoRR*, **abs/1612.08220**, (2016).

[20] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi, 'Deep text classification can be fooled', in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI*, pp. 4208–4215, (2018).

[21] Zachary C. Lipton, 'The mythos of model interpretability', *ACM Queue*, **16**(3), 30, (2018).

[22] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts, 'Learning word vectors for sentiment analysis', in *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, ACL*, pp. 142–150, (2011).

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean, 'Distributed representations of words and phrases and their compositionality', in *27th Annual Conference on Neural Information Processing Systems, NIPS*, pp. 3111–3119, (2013).

[24] W. James Murdoch, Peter J. Liu, and Bin Yu, 'Beyond word importance: Contextual decomposition to extract interactions from lstms', in *6th International Conference on Learning Representations, ICLR*, (2018).

[25] W. James Murdoch and Arthur Szlam, 'Automatic rule extraction from long short term memory networks', in *5th International Conference on Learning Representations, ICLR*, (2017).

[26] Vinod Nair and Geoffrey E. Hinton, 'Rectified linear units improve restricted boltzmann machines', in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, (2010).

[27] Dong Nguyen, 'Comparing automatic and human evaluation of local explanations for text classification', in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL*, pp. 1069–1078, (2018).

[28] Jeffrey Pennington, Richard Socher, and Christopher D. Manning, 'Glove: Global vectors for word representation', in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 1532–1543, (2014).

[29] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin, '"why should I trust you?": Explaining the predictions of any classifier', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016*, pp. 1135–1144, (2016).

[30] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin, 'Semantically equivalent adversarial rules for debugging NLP models', in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL*, pp. 856–865, (2018).

[31] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, 'Learning representations by back-propagating errors', *Nature*, **323**(6088), 696–699, (1988).

[32] Vivian Dos Santos Silva, André Freitas, and Siegfried Handschuh, 'On the semantic interpretability of artificial intelligence models', *CoRR*, **abs/1907.04105**, (2019).

[33] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 'Deep inside convolutional networks: Visualising image classification models and saliency maps', in *2nd International Conference on Learning Representations, ICLR 2014*, (2014).

[34] Shiyao Wang, Minlie Huang, and Zhidong Deng, 'Densely connected CNN with multi-scale feature attention for text classification', in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 4468–4474, (2018).

[35] Matthew D. Zeiler and Rob Fergus, 'Visualizing and understanding convolutional networks', in *Computer Vision - ECCV*, pp. 818–833, (2014).

[36] Quanshi Zhang, Ruiming Cao, Feng Shi, Ying Nian Wu, and Song-Chun Zhu, 'Interpreting cnn knowledge via an explanatory graph', in *Thirty-Second AAAI Conference on Artificial Intelligence*, (2018).

[37] Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu, 'Interpreting cnns via decision trees', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6261–6270, (2019).

[38] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun, 'Character-level convolutional networks for text classification', in *Annual Conference on Neural Information Processing Systems, NIPS*, pp. 649–657, (2015).

[39] Ye Zhang and Byron C. Wallace, 'A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification', in *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP*, pp. 253–263, (2017).