Adversarial Attacks on Deep-learning Models in Natural Language Processing: A Survey

WEI EMMA ZHANG, The University of Adelaide, Australia
QUAN Z. SHENG and AHOUD ALHAZMI, Macquarie University, Australia
CHENLIANG LI, Wuhan University, China

With the development of high computational devices, deep neural networks (DNNs), in recent years, have gained significant popularity in many Artificial Intelligence (AI) applications. However, previous efforts have shown that DNNs are vulnerable to strategically modified samples, named *adversarial examples*. These samples are generated with some imperceptible perturbations, but can fool the DNNs to give false predictions. Inspired by the popularity of generating adversarial examples against DNNs in Computer Vision (CV), research efforts on attacking DNNs for Natural Language Processing (NLP) applications have emerged in recent years. However, the intrinsic difference between image (CV) and text (NLP) renders challenges to directly apply attacking methods in CV to NLP. Various methods are proposed addressing this difference and attack a wide range of NLP applications. In this article, we present a systematic survey on these works. We collect all related academic works since the first appearance in 2017. We then select, summarize, discuss, and analyze 40 representative works in a comprehensive way. To make the article self-contained, we cover preliminary knowledge of NLP and discuss related seminal works in computer vision. We conclude our survey with a discussion on open issues to bridge the gap between the existing progress and more robust adversarial attacks on NLP DNNs.

CCS Concepts: • **Computing methodologies** → **Natural language processing**;

Additional Key Words and Phrases: Deep neural networks, adversarial examples, textual data, natural language processing

ACM Reference format:
Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial Attacks on Deep-learning Models in Natural Language Processing: A Survey. *ACM Trans. Intell. Syst. Technol.* 11, 3, Article 24 (March 2020), 41 pages.
https://doi.org/10.1145/3374217

1 INTRODUCTION

Deep neural networks (DNNs) are large neural networks whose architecture is organized as a series of layers of neurons, each of which serves as the individual computing units. Neurons are connected by links with different weights and biases and transmit the results of its activation

Authors' addresses: W. E. Zhang, School of Computer Science, The University of Adelaide, Sydney, Australia, SA 5005; email: wei.e.zhang@adelaide.edu.au; Q. Z. Sheng and A. Alhazmi, Department of Computing, Macquarie University, Australia, NSW 2109; emails: michael.sheng@mq.edu.au, ahoud.alhazmi@hdr.mq.edu.au; C. Li, School of Cyber Science and Engineering, Wuhan University, Wuhan, China, 430072; email: cllee@whu.edu.cn.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Association for Computing Machinery.
2157-6904/2020/03-ART24 $15.00
https://doi.org/10.1145/3374217

function on its inputs to the neurons of the next layer. Deep neural networks try to mimic the biological neural networks of human brains to learn and build knowledge from examples. Thus, they demonstrate the strengths in dealing with complicated tasks that are not easily to be modelled as linear or non-linear problems. Furthermore, empowered by continuous real-valued vector representations (i.e., embeddings) they are good at handling data with various modalities, e.g., image, text, video, and audio.

With the development of high computational devices, deep neural networks in recent years have gained tremendous attention in many Artificial Intelligence (AI) communities such as Computer Vision [65, 126], Natural Language Processing [16, 66], Web Mining [101, 149], and Game theory [119]. However, the interpretability of deep neural networks is still unsatisfactory as they work as black boxes, which means it is difficult to get intuitions from what each neuron exactly has learned. One of the problems of the poor interpretability is evaluating the robustness of deep neural networks. In recent years, researchers [40, 132] used small unperceivable perturbations to evaluate the robustness of deep neural networks and found that they are not robust to these perturbations. Szegedy et al. [132] first evaluated the state-of-the-art deep neural networks used for image classification with small generated perturbations on the input images. They found that the image classifiers were fooled with high probability, but human judgment is not affected. The perturbed image pixels were named *adversarial examples* and this notation is later used to denote all kinds of perturbed samples in a general manner. As the generation of adversarial examples is costly and impractical in Reference [132], Goodfellow et al. [40] proposed a fast generation method that popularized this research topic (Section 3.1 provides further discussion on these works). Followed their works, many research efforts have been made and the purposes of these works can be summarized as: (i) evaluating the deep neural networks by fooling them with unperceivable perturbations; (ii) intentionally changing the output of the deep neural networks; and (iii) detecting the oversensitivity and over-stability points of the deep neural networks and finding solutions to defense the attack.

Jia and Liang [54] are the first to consider adversarial example generation (or *adversarial attack*, we will use these two expressions interchangeably hereafter) on deep neural networks for text-based tasks (namely, *textual deep neural networks*). Their work quickly gained research attention in the Natural Language Processing (NLP) community. However, due to intrinsic differences between images and textual data, the adversarial attack methods on images cannot be directly applied to the latter. First of all, image data (e.g., pixel values) is continuous, but textual data is discrete in nature. Conventionally, we vectorize the texts before inputting them into the deep neural networks. Traditional vectoring methods include leveraging term frequency and inverse document frequency, and one-hot representation (details in Section 3.3). When applying gradient-based adversarial attacks adopted from images on these representations, the generated adversarial examples are invalid characters or word sequences [157]. One solution is to use word embeddings as the input of deep neural networks. However, this will also generate words that can not be matched with any words in the word embedding space [38]. Second, the perturbation of images are small change of pixel values that are hard to be perceived by human eyes, thus humans can correctly classify the images, showing the poor robustness of deep neural models. But for adversarial attack on texts, small perturbations are easily perceptible. For example, replacement of characters or words would generate invalid words or syntactically incorrect sentences. Further, it would alter the semantics of the sentence drastically. Therefore, the perturbations are easily to be perceived–in this case, even human being cannot provide correct predictions.

To address the aforementioned differences and challenges, many attacking methods have been proposed since the pioneer work of Jia and Liang [54]. Despite the popularity of the topic in the NLP community, there is no comprehensive review paper that collects and summarizes the efforts

in this research direction. There is a need for this kind of work that helps successive researchers and practitioners to have an overview of these methods.

Related surveys and the differences to this survey. In Reference [9], the authors presented a comprehensive review on different classes of attacks and defenses against machine learning systems. Specifically, they proposed a taxonomy for identifying and analyzing these attacks and applied the attacks on a machine learning–based application, i.e., a statistical spam filter, to illustrate the effectiveness of the attack and defense. This work targeted machine learning algorithms rather than neural models. Inspired by Reference [9], the authors of Reference [35] reviewed the defences of adversarial attack in the security point of view. The work is not limited to machine learning algorithms or neural models, but a generic report about adversarial defenses on security related applications. The authors found that existing security related defense works lack of clear motivations and explanations on how the attacks are related to the real security problems and how the attack and defense are meaningfully evaluated. Thus, they established a taxonomy of motivations, constraints, and abilities for more plausible adversaries. In Reference [13], the authors provided a thorough overview of the evolution of the adversarial attack research over a ten-year period from 2008 to 2018, and focused on the research works from computer vision and cyber security. The paper covers the works from pioneering non-deep-learning algorithms to recent deep-learning algorithms. It is also from the security point of view to provide detailed analysis on the effect of the attacks and defenses. The authors of Reference [79] reviewed the same problem from a data-driven perspective. They analyzed the attacks and defenses according to the learning phases, i.e., the training phase and test phase.

Unlike previous works that discuss generally on the attack methods on machine learning algorithms, Reference [154] focuses on the adversarial examples on deep-learning models. It reviews current research efforts on attacking various deep neural networks in different applications. The defense methods are also extensively surveyed. However, it mainly discusses adversarial examples for image classification and object recognition tasks. The work in Reference [2] provides a comprehensive review on the adversarial attacks on deep-learning models used in computer vision tasks. It is an application-driven survey that groups the attack methods according to the sub-tasks under computer vision area. The article also comprehensively reports the works on the defense side, the methods of which are mainly grouped into three categories.

All the mentioned works either target a general overview of the attacks and defenses on machine learning models or focus on specific domains such as computer vision and cyber security. Our work differs with them in that we specifically focus on the attacks and defenses on textual deep-learning models. Furthermore, we provide a comprehensive review that covers information from different aspects to make this survey self-contained.

Papers selection. The papers we reviewed in this article are high-quality papers selected from top NLP and AI conferences, including ACL,[1] COLING,[2] NAACL,[3] EMNLP,[4] ICLR,[5] AAAI,[6] and IJCAI.[7] Other than accepted papers in aforementioned conferences, we also considered good papers in

[1]Annual Meeting of the Association for Computational Linguistics.
[2]International Conference on Computational Linguistics.
[3]Annual Conference of the North American Chapter of the Association for Computational Linguistics.
[4]Empirical Methods in Natural Language Processing.
[5]International Conference on Learning Representations.
[6]AAAI Conference on Artificial Intelligence.
[7]International Joint Conference on Artificial Intelligence.

e-Print archive,[8] as it reflects the latest research outputs. We selected papers from the archive with three metrics: paper quality, method novelty, and the number of citations (optional[9]).

Contributions of this survey. The aim of this survey is to provide a comprehensive review on the research efforts on generating adversarial examples on textual deep neural networks. It is motivated by the drastically increasing attentions on this topic. This survey will serve researchers and practitioners who are interested in attacking textual deep neural models. We expect that the readers have some basic knowledge of the deep neural networks architectures, which are not the focus in this article. To summarize, the key contributions of this survey are:

- We conduct a comprehensive review for adversarial attacks on textual deep neural models and propose different classification schemes to organize the reviewed literature; this is the first work of this kind;
- We provide all related information to make the survey self-contained and thus it is easy for readers who have limited NLP knowledge to understand;
- We discuss some open issues, and identify some possible research directions in this research field aiming to build more robust textual deep-learning models with the help of adversarial examples.

The remainder of this article is organized as follows. We introduce the preliminaries for adversarial attacks on deep-learning models in Section 2, including the taxonomy of adversarial attacks and deep-learning models used in NLP. In Section 3, we address the difference between attacking image data and textual data and briefly reviewe exemplary works for attacking image DNN that inspired their follow-ups in NLP. Section 4 first presents our classification on the literature and then gives a detailed introduction to the state of the art. We discuss the defense strategies in Section 5 and point out the open issues in Section 6. Finally, the article is concluded in Section 7.

2 OVERVIEW OF ADVERSARIAL ATTACKS AND DEEP-LEARNING TECHNIQUES IN NATURAL LANGUAGE PROCESSING

Before we dive into the details of this survey, we start with an introduction to the general taxonomy of adversarial attack on deep-learning models. We also introduce the deep-learning techniques and their applications in natural language processing.

2.1 Adversarial Attacks on Deep-learning Models: The General Taxonomy

In this section, we provide the definitions of adversarial attacks and introduce different aspects of the attacks, followed by the measurement of perturbations and the evaluation metrics of the effectiveness of the attacks in a general manner that applies to any data modality.

2.1.1 Definitions. We present the main concepts of adversarial attacks as following:

- *Deep Neural Network (DNN).* A deep neural network (we use DNN and deep-learning model interchangeably hereafter) can be simply presented as a nonlinear function $f_\theta : \mathbf{X} \rightarrow \mathbf{Y}$, where $\mathbf{X}$ is the input features/attributes, $\mathbf{Y}$ is the output predictions that can be a discrete set of classes or a sequence of objects. θ represents the DNN parameters and are learned via gradient-based back-propagation during the model training. Best parameters would be

[8] arXiv.org.

[9] As the research topic only emerges from 2017, we relax the citation number to over five if it is published more than one year. If the paper has less than five citations, but is very recent and satisfies the other two metrics, then we also include it in this survey.

obtained by minimizing the the gap between the model's prediction $f_\theta(\mathbf{X})$ and the correct label $\mathbf{Y}$, where the gap is measured by loss function $J(f_\theta(\mathbf{X}), \mathbf{Y})$.

- *Perturbations.* Perturbations are intently created small noises that to be added to the original input data examples in test stage, aiming to fool the deep-learning models.
- *Adversarial Examples.* An adversarial example $\mathbf{x}'$ is an example created via worst-case perturbation of the input to a deep-learning model. An ideal DNN would still assign correct class $\mathbf{y}$ (in the case of classification task) to $\mathbf{x}'$, while a victim DNN would have high confidence on wrong prediction of $\mathbf{x}'$. $\mathbf{x}'$ can be formalized as

$$\begin{aligned} \mathbf{x}' = \mathbf{x} + \eta, f(\mathbf{x}) = \mathbf{y}, \mathbf{x} \in \mathbf{X}, \\ f(\mathbf{x}') \neq \mathbf{y}, \\ \text{or } f(\mathbf{x}') = \mathbf{y}', \mathbf{y}' \neq \mathbf{y}, \end{aligned} \tag{1}$$

where η is the worst-case perturbation. The goal of the adversarial attack can be deviating the label to incorrect one ($f(\mathbf{x}') \neq \mathbf{y}$) or specified one ($f(\mathbf{x}') = \mathbf{y}'$).

2.1.2 Threat Model. We adopt the definition of *Threat Model* for attacking DNN from Reference [154]. In the following, we discuss several aspects of the threat model:

- *Model Knowledge.* The adversarial examples can be generated using black-box or white-box strategies in terms of the knowledge of the attacked DNN. Black-box attack is performed when the architectures, parameters, loss function, activation functions and training data of the DNN are not accessible. Adversarial examples are generated by directly accessing the test dataset, or by querying the DNN and checking the output change. On the contrary, white-box attack is based on the knowledge of technical details of DNN.
- *Target.* The generated adversarial examples can change the output prediction to be incorrect or to specific result as shown in Equation (1). Compared to the un-targeted attack ($f(\mathbf{x}') \neq \mathbf{y}$), targeted attack ($f(\mathbf{x}') = \mathbf{y}'$) is more strict as it not only changes the prediction but also enforces constraint on the output to generate specified prediction. For binary tasks, e.g., binary classification, un-targeted attack equals to the targeted attack.
- *Granularity.* The attack granularity refers to the level of data from which the adversarial examples are generated. For example, it is usually the image pixels for image data. Regarding the textual data, it could be character, word, and sentence-level embedding. Section 3.3 will give further introduction on attack granularity for textual DNN.
- *Motivation.* Generating adversarial examples is motivated by two goals: attack and defense. The attack aims to examine the robustness of the target DNN, while the defense takes a step further utilizing generated adversarial examples to robustify the target DNN. Section 5 will give more details.

2.1.3 Measurements. Two groups of measurements are required in the adversarial attack for (i) controlling the perturbations and (ii) evaluating the effectiveness of the attack, respectively.

- *Perturbation Constraint.* As aforementioned, the perturbation η should not change the true class label of the input—that is, an ideal DNN classifier, if we take classification as the example, will provide the same prediction on the adversarial example to the original example. η cannot be too small as well, to avoid ending up with no effect on target DNNs. Ideally, effective perturbation is the most impactful noise in a constrained range. Reference [132] first put a constraint that $(\mathbf{x} + \eta) \in [0, 1]^n$ for image adversarial examples, ensuring the adversarial example has the same range of pixel values as the original data [143]. Reference [40] simplifies the solution and uses max norm to constrain η: $||\eta||_\infty \leq \epsilon$. This was inspired by

the intuition that a perturbation that does not change any specific pixel by more than some amount ϵ cannot change the output class [143]. Using max-norm is sufficient enough for image classification/object recognition tasks. Later on, other norms, e.g., L_2 and L_0, were used to control the perturbation in attacking DNN in computer vision. Constraining η for textual adversarial attack is somehow different. Section 3.3 will give more details.

- *Attack Evaluation.* Adversarial attacks are designed to degrade the performance of DNNs. Therefore, evaluating the effectiveness of the attack is based on the performance metrics of different tasks. For example, classification tasks have metrics such as accuracy, F1 score and AUC score. We leave the metrics for different NLP as out-of-scope content in this article and suggest readers refer to specific tasks for information.

2.2 Deep-learning in NLP

Neural networks have been gaining increasing popularity in NLP community in recent years and various DNN models have been adopted in different NLP tasks. Apart from the feed forward neural networks and Convolutional Neural Networks (CNN), Recurrent/Recursive Neural Networks (RNN) and their variants are the most common neural networks used in NLP, because of their natural ability of handling sequences. In recent years, two important breakthroughs in deep learning are brought into NLP. They are *sequence-to-sequence learning* [131] and *attention mechanism* [8]. Reinforcement learning and generative models are also gained much popularity [152]. In this section, we will briefly overview the DNN architectures and techniques applied in NLP that are closely related to this survey. We suggest readers refer to detailed reviews of neural networks in NLP in References [100, 152].

2.2.1 Feed-Forward Networks. A feed-forward network has several forward layers and each node in a layer connects to each node in the following layer, making the network fully connected. The network utilizes nonlinear transformation to distinguish data that is not linearly separable. The major drawback of its application in NLP is that it cannot handle well the text sequences in which the word order matters as it do not record the order of the elements. To evaluate the robustness of feed forward network in NLP, adversarial examples are often generated for specifically designed feed-forward networks. For example, the authors of References [3, 43, 44] worked on the specified malware detection models.

2.2.2 Convolutional Neural Network (CNN). A Convolutional Neural Network contains convolutional layers and pooling (down-sampling) layers and final fully connected layer. The Convolutional layer uses convolution operation to extract meaningful local patterns of input. Specifically, CNN identifies local predictors and combines them together to generate a fixed-sized vector for the inputs, which contains the most or important informative aspects of the data. In addition, it is order-sensitive. Therefore, it excels in computer vision tasks and later is widely adopted in NLP applications.

Yoon Kim [59] adopted CNN for sentence classification and used Word2Vec to represent words as input. Then the convolutional operation is restricted to the direction of word sequence, rather than the word embeddings. The model demonstrates excellent performance on several benchmark datasets and has become a benchmark work of adopting CNN in NLP applications. Zhang et al. [156] presented CNN for text classification at character level. They used one-hot representation for each character of alphabet. These two representative textual CNNs are evaluated via adversarial examples in many applications [12, 29, 30, 34, 76].

2.2.3 Recurrent Neural Networks/Recursive Neural Networks. Recurrent Neural Networks are neural models adapted from feed-forward neural networks for learning mappings between

sequential inputs and outputs [116]. RNNs allows data with arbitrary length and it introduces cycles in their computational graph to model efficiently the influence of time [48]. The model does not suffer from statistical estimation problems stemming from data sparsity and thus leads to impressive performance in dealing with sequential data [36]. Recursive neural networks [37] extends recurrent neural networks from sequences to tree, which respects the hierarchical dependency of the language. In some situations, backwards dependencies exist, which is in need for the backward analysis. Bi-directional RNN thus was proposed to process each sentences in both directions, forwards and backwards, using two parallel RNN networks, and combine their outputs.

RNN has many variants, among which the Long Short-Term Memory (LSTM) network [50] gains the most popularity. LSTM is a specific RNN that was designed to capture the long-term dependencies. In LSTM, the hidden state are computed through combination of three *gates*, i.e., *input gate*, *forget gate* and *output gate*, that control information. LSTM networks have subsequently proved to be more effective than conventional RNNs [42]. GRUs is a simplified version of LSTM that it only consists of two gates, thus it is more efficient in terms of computational cost. Some popular LSTM variants are proposed to solve various NLP tasks [21, 50, 112, 133, 141, 146]. These representative works have received the interests of evaluation with adversarial examples recently [34, 53, 54, 91, 103, 112, 118, 130, 157].

2.2.4 Sequence-to-sequence Learning (Seq2Seq) Models. Sequence-to-sequence learning (Seq2Seq) [131] is one of the important breakthroughs in deep learning and is now widely used for NLP applications. Seq2Seq model has the superior capacity to generate another sequence information for a given sequence information with an encoder-decoder architecture [27]. Usually, a Seq2Seq model consists of two recurrent neural networks: an encoder that processes the input and compresses it into a vector representation and a decoder that predicts the output. Latent Variable Hierarchical Recurrent Encoder-Decoder (VHRED) model [122] is a recently popular Seq2Seq model that generates sequences leveraging the complex dependencies between subsequences. Reference [24] is one of the first neural machine translation (NMT) model that adopts the Seq2Seq model. OpenNMT [63], a Seq2Seq NMT model proposed recently, becomes one of the benchmark works in NMT. As they are adopted and applied widely, attack works also emerge [22, 30, 98, 127].

2.2.5 Attention Models. Attention mechanism [8] is another breakthrough in deep learning. It was initially developed to overcome the difficulty of encoding a long sequence required in Seq2Seq models [27]. Attention allows the decoder to look back on the hidden states of the source sequence. The hidden states then provide a weighted average as additional input to the decoder. This mechanism pays *attention* on informative parts of the sequence. Rather than looking at the input sequence in vanilla attention models, self-attention [136] in NLP is used to look at the surrounding words in a sequence to obtain more contextually sensitive word representations [152]. BiDAF [121] is a bidirectional attention flow mechanism for machine comprehension and achieved outstanding performance when proposed. References [54, 127] evaluated the robustness of this model via adversarial examples and became the first few works using adversarial examples for attacking textual DNNs. Other attention-based DNNs [25, 107] also received adversarial attacks recently [29, 91].

2.2.6 Reinforcement Learning Models. Reinforcement learning trains an agent by giving a reward after agents perform discrete actions. In NLP, a reinforcement learning framework usually consists of an agent (a DNN), a policy (guiding action) and a reward. The agent picks an action (e.g., predicting next word in a sequence) based on a policy, then updates its internal state accordingly, until arriving the end of the sequence where a reward is calculated. Reinforcement learning requires proper handling of the action and the states, which may limit the expressive power and

learning capacity of the models [152]. But it gains much interests in task-oriented dialogue systems [74] as they share the fundamental principle as decision making processes. Limited works so far can be found to attack the reinforcement learning model in NLP [98].

2.2.7 Deep Generative Models. In recent years, two powerful deep generative models, Generative Adversarial Networks (GANs) [39] and Variational Auto-Encoders (VAEs) [62] are proposed and gain much research attention. Generative models are able to generate realistic data instances that are very similar to ground truth data in a latent space. In the NLP field, they are used to generate text. GANs [39] consists of two adversarial networks: a *generator* and a *discriminator*. The discriminator is to discriminate the real and generated samples, while the generator is to generate realistic samples that aim to fool the discriminator. GAN uses a min-max loss function to train two neural networks simultaneously. VAEs consist of encoder and generator networks. The encoder encodes an input into a latent space and the generator generates samples from the latent space. Deep generative models are not easy to train and evaluate. Hence, these deficiencies hinder their wide usage in many real-world applications [152]. Although they have been adopted in generating texts, so far no work examines their robustness using adversarial examples.

3 FROM IMAGE TO TEXT

Adversarial attacks are originated from the computer vision community. In this section, we introduce representative works, discuss differences between attacking image data and textual data, and present preliminary knowledge when performing adversarial attacks on textual DNNs.

3.1 Crafting Adversarial Examples: Inspiring Works in Computer Vision

Since adversarial examples were first proposed for attacking DNNs for object recognition in the computer vision community [17, 40, 95, 104, 105, 132, 157], this research direction has been receiving sustained attentions. We briefly introduce some works that inspired their followers in NLP community in this section, allowing the reader to better understand the adversarial attacks on textual DNNs. For comprehensive review of attack works in computer vision, please refer to Reference [2].

L-BFGS. Szegedy et al. invented the *adversarial examples* notation [132]. They proposed an explicitly designed method to cause the model to give wrong prediction of adversarial input $(\mathbf{x} + \eta)$ for image classification task. It came to solve the optimization problem:

$$\eta = \arg\min_{\eta} \lambda||\eta||_2^2 + J(\mathbf{x} + \eta, y') \quad s.t. \ \ (\mathbf{x} + \eta) \in [0, 1]^n, \tag{2}$$

where y' is the target output of $(\mathbf{x}' + \eta)$, but incorrect given an ideal classifier. J denotes the cost function of the DNN and λ is a hyperparameter to balance the two parts of the equation. This minimization was initially performed with a box-constrained **L**imited memory **B**royden-**F**letcher-**G**oldfarb-**S**hanno (L-BFGS) algorithm and thus was named after it. The optimization was repeated multiple times until reaching a minimum λ that satisfy Equation (2).

Fast Gradient Sign Method (FGSM). L-BFGS is very effective, but highly expensive—this inspired Goodfellow et al. [40] to find a simplified solution. Instead of fixing y' and identifying the most effective η in L-BFGS, FGSM fixed l_∞ norm of η and minimized the cost (Equation (3)). Then they linearized the problem with a first-order Taylor series approximation (Equation (4)), and got the closed-form solution of η (Equation (5)) [143]:

$$\eta = \arg\min_{\eta} J(\mathbf{x} + \eta, y) \quad s.t. \ \ ||\eta||_\infty \leq \epsilon, \tag{3}$$

$$\eta = \arg\min_{\eta} J(\mathbf{x}, y) + \eta^{\mathrm{T}} \nabla_{\mathbf{x}} J(\mathbf{x}, y) \quad s.t. \ \ ||\eta||_\infty \leq \epsilon, \tag{4}$$

$$\eta = \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} J(\mathbf{x}, \mathbf{y})), \tag{5}$$

where ϵ is a parameter set by attacker, controlling the perturbation's magnitude. sign(x) is the sign function that returns 1 when $x > 0$, and -1 when $x < 0$, otherwise returns 0. $\nabla_{\mathbf{x}} J(\mathbf{x}, \mathbf{y})$ denotes the gradient of loss function respect to the input, and can be calculated via back-propagation. FGSM attracts the most follow-up works in NLP.

Jacobian Saliency Map Adversary (JSMA). Unlike FGSM using gradients to attack, Papernot et al. [105] generated adversarial examples using forward derivatives (i.e., model Jacobian). This method evaluates the neural model's output sensitivity to each input component using its *Jacobian Matrix*. Jacobian matrices form the adversarial saliency maps that rank each input component's contribution to the target attack. A perturbation is then selected from the maps. Thus, the method was named *Jacobian-based Saliency Map Attack*. The Jacobian matrix of a given $\mathbf{x}$ is given by

$$Jacb_F[i, j] = \frac{\partial F_i}{\partial \mathbf{x}_j}, \tag{6}$$

where $\mathbf{x}_i$ is the ith component of the input and F_j is the jth component of the output. Here F denotes the *logits* (i.e., the inputs to the softmax function) layer. $J_F[i, j]$ measures the sensitivity of F_j with respect to $\mathbf{x}_i$.

C&W Attack. Carlini and Wagner [17] aimed to evaluate the defensive distillation strategy [49] for mitigating the adversarial attacks. They restricted the perturbations with l_p norms where p equals to 0, 2, and ∞ and proposed seven versions of J for the following optimization problem:

$$\eta = \arg\min_{\eta} ||\eta||_p + \lambda J(\mathbf{x} + \eta, y') \quad s.t. \ (\mathbf{x} + \eta) \in [0, 1]^n, \tag{7}$$

and the formulation shares the same notation with aforementioned works.

DeepFool. DeepFool [95] is an iterative L_2-regularized algorithm. The authors first assumed the neural network is linear, thus they can separate the classes with a hyperplane. They simplified the problem and found optimal solution based on this assumption and constructed adversarial examples. To address the non-linearity fact of the neural network, they repeated the process until a true adversarial example is found.

Substitute Attack. The above-mentioned representative works are all white-box methods, which require the full knowledge of the neural model's parameters and structures. However, in practice, it is not always possible for attackers to craft adversaries in white-box manner due to the limited access to the model. The limitation was addressed by Papernot et al. [104] and they introduced a black-box attack strategy. They trained a substitute model to approximate the decision boundaries of the target model with the labels obtained by querying the target model. Then they conducted a white-box attack on this substitute one and generated adversarial examples accordingly. Specifically, they adopted FSGM and JSMA in generating adversarial examples for the substitute DNN.

GAN-like Attack. There are another branch of black-box attacks that leverages the Generative Adversarial Neural (GAN) models. Zhao et al. [157] first trained a generative model, WGAN, on the training dataset $\mathbf{X}$. WGAN could generate data points that follow the same distribution with $\mathbf{X}$. Then they separately trained an inverter to map data sample $\mathbf{x}$ to $\mathbf{z}$ in the latent dense space by minimizing the reconstruction error. Instead of perturbing $\mathbf{x}$, they searched for adversaries $z*$ in the neighbour of $\mathbf{z}$ in the latent space and mapped $\mathbf{z}*$ back to $\mathbf{x}*$ and check if $\mathbf{x}*$ would change the prediction. They introduced two search algorithms: *iterative stochastic search* and *hybrid shrinking search*. The former used expanding strategy that gradually expands the search space, while the

later used shrinking strategy that starts from a wide range and recursively tightens the upper bound of the search range.

3.2 Attacking Image DNNs versus Attacking Textual DNNs

To attack a textual DNN model, we cannot directly apply the approaches from the image DNN attackers as there are three main differences between them:

- *Discrete vs. Continuous Inputs.* Image inputs are continuous, typically the methods use L_p norm to measure the distance between clean data point with the perturbed data point. However, textual data is symbolic, thus discrete. It is hard to define the perturbations on texts. Carefully designed variants or distance measurements for textual perturbations are required. Another choice is to first map the textual data to continuous data, then adopt the attack method from computer vision. We will give further discussion for this aspect in Section 3.3.
- *Perceivable vs. Unperceivable.* Small change of the image pixels usually can not be easily perceived by human beings, hence the adversarial examples will not change the human judgment, but can only fool the DNN models. However, small changes on texts, e.g., character or word change, will easily be perceived, rendering the possibility of attack failure. For example, the changes could be identified or corrected by spelling-check and grammar check before inputting into textual DNN models. Therefore, it is nontrivial to find unperceivable textual adversaries.
- *Semantic vs. Semantic-less.* In the case of images, small changes usually do not change the semantics of the image as they are trivial and unperceivable. However, perturbation on texts would easily change the semantics of a word and a sentence, thus can be easily detected and heavily affect the model output. For example, deleting a negation word would change the sentiment of a sentence. But this is not the case in computer vision where perturbing individual pixels does not turn the image from a cat to another animal. Changing semantics of the input is against the goal of adversarial attack that keeps the correct prediction unchanged while fooling an victim DNN.

Due to these differences, current state-of-the-art textual DNN attackers either carefully adjust the methods from image DNN attackers by enforcing additional constraints, or propose novel methods using different techniques.

3.3 Vectorizing Textual Inputs and Perturbation Measurements

Vectorizing Textual Input. DNN models require vectors as input, for image tasks, the normal way is to use the pixel value to form the vectors/matrices as DNN input. But for textual models, special operations are needed to transform the text into vectors. There are three main branches of methods: word-count-based encoding, one-hot encoding and dense encoding (or feature embedding) and the latter two are mostly used in DNN models of textual applications.

- *Word-Count-based Encoding.* The Bag-of-words (BOW) method has the longest history in vectorizing text. In the BOW model, zero-encoded vector with length of the vocabulary size is initialized. Then the dimension in vector is replaced by the count of corresponding word's appearance in the given sentence. Another word-count-based encoding is to utilize the term frequency-inverse document frequency (TF-IDF) of a word (term), and the dimension in the vector is the TF-IDF value of the word.
- *One-hot Encoding.* In one-hot encoding, a vector feature represents a token–a token could be a character (character-level model) or a word (word-level model). For character-level

one-hot encoding, the representation can be formulated as [30]

$$\mathbf{x} = [(x_{11}, \ldots x_{1n}); \ldots (x_{m1}, \ldots x_{mn})], \tag{8}$$

where $\mathbf{x}$ be a text of sequence, $x_{ij} \in \{0,1\}^{|A|}$ and $|A|$ is the alphabet (in some works, $|A|$ also include symbols). In Equation (8), m is the number of words, n is the maximum number of characters for a word in sequence $\mathbf{x}$. Thus, each word has the same-fixed length of vector representation and the length is decided by the maximum number of characters of the words. For word-level one-hot encoding, following the above notations, the text x can be represented as

$$\mathbf{x} = [(x_1, \ldots, x_m, x_{m+1} \ldots x_k)], \tag{9}$$

where $x_m \in \{0,1\}^{|V|}$ and $|V|$ is the vocabulary, which contains all words in a corpus. k is the maximum number of words allowed for a text, so that $[(x_{m+1} \ldots x_k)]$ is zero-paddings if $m + 1 < k$. One-hot encoding produces vectors with only 0 and 1 values, where 1 indicates the corresponding character/word appears in the sentence/paragraph, and 0 indicates it does not appear. Thus, one-hot encoding usually generates sparse vectors/matrices. DNNs have proven to be very successful in learning values from the sparse representations as they can learn more dense distributed representations from the one-hot vectors during the training procedure.

- *Dense Encoding*. Comparing to the one-hot encoding, dense encoding generates low dimensional and distributed representations for textual data. Word2Vec [90] uses continuous bag-of-words (CBOW) and skip-gram models to generate dense representation for words, i.e., word embeddings. The underlying assumption is that words appearing within similar context possess similar meaning. Word embeddings, to some extend, alleviates the discreteness and data-sparsity problems for vectorizing textual data [36]. Extensions of word embeddings such as doc2vec and paragraph2vec [69] encode sentences/paragraphs to dense vectors.

Perturbation Measurement. As described in Section 2.1.3, there needs a way to measure the size of the perturbation, so that it can be controlled to ensure the ability of fooling the victim DNN while remains unperceivable. However, the measurement in textual perturbations is drastically different with the perturbations in image. Usually, the size of the perturbation is measured by the distance between clean data $\mathbf{x}$ and its adversarial example $\mathbf{x}'$. But in texts, the distance measurement also needs to consider the grammar correctness, syntax correctness and semantic-preservance. We here list the measurements used in the rest of this survey.

- *Norm-based measurement.* Directly adopting norms such as $L_p, p \in 0, 1, 2, \infty$ requires the input data be continuous. One solution is to use continuous and dense representations (e.g., embedding) to represent the texts. But this usually results in invalid and incomprehensible texts, which in turn need to involve other constraints.
- *Grammar and syntax related measurement.* Ensuring the grammar or syntactic correctness makes the adversarial examples not easily perceived.
  - —*Grammar and syntax checker* are used in some works to ensure the textual adversarial examples generated are valid.
  - —*Perplexity* is usually used to measure the quality of a language model. In one reviewed literature [91], the authors used perplexity to ensure the generated adversarial examples (sentences) are valid.

— *Paraphrase* is controlled and can be regarded as a type of adversarial example (Section 4.3.3). When perturbing, the validity of paraphrases is ensured in the generation process.

- *Semantic-preserving measurement.* Measuring semantic similarity/distance is often performed on word vectors by adopting vectors' similarity/distance measurements. Given two n-dimensional word vectors $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ and $\mathbf{q} = (q_1, q_2, \ldots, q_n)$:
  — *Euclidean Distance* is a distance of two vectors in the *Euclidean* space:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}. \tag{10}$$

  — *Cosine Similarity* computes cosine value of the angle between the two vectors:

$$cos(\mathbf{p}, \mathbf{q}) = \frac{\sum_{i=1}^{n} p_i \times q_i}{\sqrt{\sum_{i=1}^{n} (p_i)^2} \times \sqrt{\sum_{i=1}^{n} (q_i)^2}}. \tag{11}$$

- *Edit-based measurement.* Edit distance is a way of quantifying the minimum changes from one string to the other. Different definitions of edit distance use different sets of string operations [73].
  — *Levenshtein Distance* uses insertion, removal and substitution operations.
  — *Word Mover's Distance (WMD)* [68] is an edit distance operated on word embeddings. It measures the minimum amount of distance that the words of one document need to travel to reach the words of the other document in the embedding space [38]. This distance can be measured with a minimization problem formulated as follows:

$$\begin{aligned} &\min \textstyle\sum_{i,j=1}^{n} \mathbf{T}_{ij} ||\mathbf{e_i} - \mathbf{e_j}||_2, \\ s.t., &\textstyle\sum_{j=1}^{n} \mathbf{T}_{ij} = d_i, \forall i \in \{i, \ldots, n\}, \textstyle\sum_{i=1}^{n} \mathbf{T}_{ij} = d'_i, \forall j \in \{i, \ldots, n\}, \end{aligned} \tag{12}$$

  where $\mathbf{e_i}$ and $\mathbf{e_j}$ are word embeddings of word i and word j, respectively. n is the number of words. $\mathbf{T} \in \mathcal{R}^{n \times n}$ is a flow matrix, where $\mathbf{T}_{ij} \leq 0$ denotes how much of word i in $\mathbf{d}$ travels to word j in $\mathbf{d}'$. And $\mathbf{d}$ and $\mathbf{d}'$ are normalized bag-of-words vectors of the two documents, respectively.
  — *Number of changes* is a straightforward yet simple way to measure the edits and it is adopted in some relevant literature.

- *Jaccard similarity coefficient* is used for measuring similarity of finite sample sets by utilizing intersection and union of the sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{13}$$

  In texts, A, B are two documents (or sentences). $|A \cap B|$ denotes the number of words appearing in both documents, $|A \cup B|$ refers to the number of unique words in total.

4 ATTACKING NEURAL MODELS IN NLP: THE STATE-OF-THE-ART

In this section, we first introduce the categories of attack methods on textual deep-learning models and then highlight the state-of-the-art research works, aiming to identify the most promising advances in recent years.

4.1 Categories of Attack Methods on Textual Deep-learning Models

We categorize existing adversarial attack methods from different viewpoints. Figure 1 illustrates these viewpoints in detail.

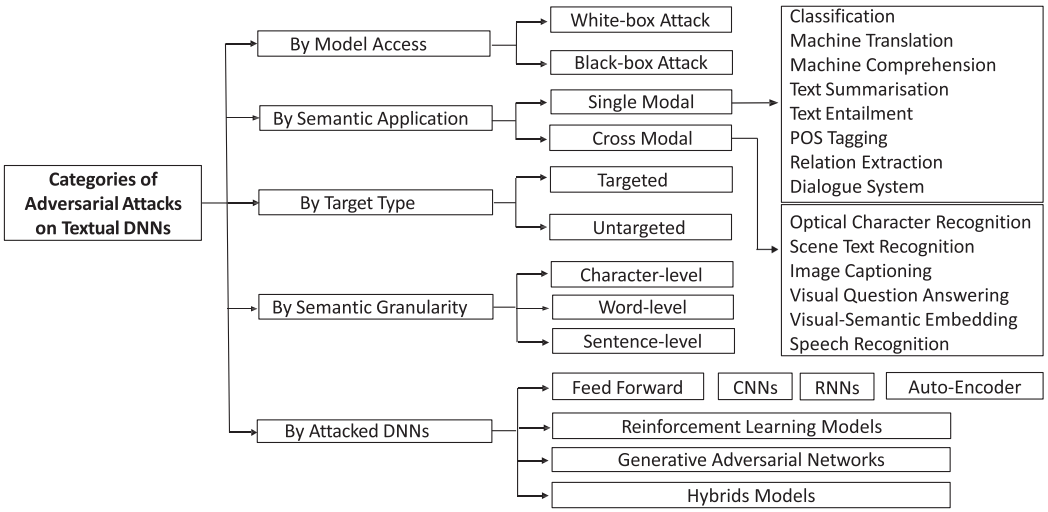

Fig. 1. Categories of adversarial attack methods on textual deep-learning models.

In this article, five viewpoints are used to categorize the attack methods:

- *Model Access* refers to the knowledge of attacked model when the attack is performed. In the following section, we focus on the discussion from this viewpoint.
- *Semantic Application* refers to the methods via different NLP applications. More detailed discussion will be provided in Section 4.5.
- *Target Type* refers to the goal of the attack is enforcing incorrect prediction or targeting specific results.
- *Semantic Granularity* considers on what granularity level the model is attacked.
- *attacked DNNs* are victim DNNs that we have discussed in Section 2.2.

In the following sections, we will continuously provide information about different categories that the methods belong to. One important category of methods need to be noted is the cross-modal attacks, in which the attacked model considers the tasks dealing with multi-modal data, e.g., image and text data. They are not attacks for pure textual DNNs, hence we discuss this category of methods separately in Section 4.4 in addition to white-box attacks (Section 4.2) and black-box attacks (Section 4.3).

4.2 White-box Attack

In a white-box attack, the attack requires the access to the model's full information, including architecture, parameters, loss functions, activation functions, input and output data. White-box attacks typically approximate the worst-case attack for a particular model and input, incorporating a set of perturbations. This attack strategy is often very effective. In this section, we group white-box attacks on textual DNNs into seven categories.

4.2.1 FGSM-based. FGSM is one of the first attack methods on images (Section 3.1). It attracts many follow-up efforts in attacking textual DNNs. TextFool [76] adopts the concept of FGSM. Specifically, it approximates the contributions of text items and identifies the ones that possess significant contribution to the text classification task. Instead of using sign of the cost gradient in FGSM, this work considers the magnitude of the cost gradient. The authors proposed three types of attacks: *insertion*, *modification*, and *removal*. Specifically, they computed cost gradient

$\Delta_x J(f, x, c')$ of each training sample x, employing back propagation, where f is the model function, x is the original data sample, and c' is the target output class. Then they identified the characters that contain the dimensions with the highest gradient magnitude and named them *hot characters*. Phrases that contain enough *hot characters* and occur the most frequently are chosen as *Hot Training Phrases* (HTPs). In the *insertion* strategy, adversarial examples are crafted by inserting a few HTPs of the target class c' nearby the phrases with significant contribution to the original class c. The authors further leveraged external sources like Wikipedia and *forged fact* to select the valid and plausible sentences. In the *modification* strategy, the authors identified *Hot Sample Phrase (HSP)* using the same way of identifying HTPs. Then they replaced the characters in HTPs by common misspellings or characters visually similar. Similar to identifying the HTPs, they identify phrases with significant contribution to the current classification by locating hot characters with the highest gradient magnitude. In the *removal* strategy, the inessential adjective or adverb in HSPs are removed. The three strategies and their combinations are evaluated on a CNN text classifier [156]. However, these methods are performed manually, as discussed by the authors.

The work in Reference [117] adopted the same idea as TextFool, but it provides a *removal-addition-replacement* strategy that first tries to *remove* the adverb (w_i), which contributes the most to the text classification task (measured using loss gradient). If the output sentences in this step have incorrect grammar, then the method will *insert* a word p_j before w_i. p_j is selected from a candidate pool, in which the synonyms and typos and genre specific keywords (identified via term frequency) are candidate words. If the output cannot satisfy the highest cost gradient for all the p_j, then the method *replaces* w_i with p_j. The authors showed that their method is more effective than TextFool. As the method ordered the words with their contribution ranking and crafted adversarial samples according to the order, it is a greedy method that always gets the minimum manipulation until the output changes. To avoid being perceived by human, the authors constrained the replaced/added words to not affect the grammar and part-of-Speech (POS) of the original words.

In malware detection, an portable executable (PE) is represented by binary vector $\{x_1, \ldots, x_m\}$, $x_i \in \{0, 1\}$ that uses 1 and 0 to indicate the PE is present or not where m is the number of PEs. Using PEs' vectors as features, malware detection DNNs can identify the malicious software. Malware detection is not a typical textual application. As it also works with discrete data, methods for attacking textual DNNs can be applied to attacking malware detection DNNs. The authors of Reference [3] investigated the methods to generate binary-encoded adversarial examples. To preserve the functionality of the adversarial examples, they incorporated four bounding methods to craft perturbations. The first two methods adopt FSGM^k [67], the multi-step variant of FGSM. This FGSM variant restricts the perturbations in a binary domain by introducing deterministic rounding (dFGSM^k) and randomized rounding (rFGSM^k). These two bounding methods are similar to L_∞-ball constraints on images [40]. The third method (multi-step Bit Gradient Ascent (BGA^k)) sets the bit of the jth feature if the corresponding partial derivative of the loss is greater than or equal to the loss gradient's L_2-norm divided by $\sqrt{m}$. The fourth method (multi-step Bit Coordinate Ascent (BCA^k)) updates one bit in each step by considering the feature with the maximum corresponding partial derivative of the loss. These two last methods actually visit multiple feasible vertices. The work also proposes an adversarial learning framework that aims to robustify the malware detection model.

Reference [114] also attacks malware detection DNNs. The authors made perturbations on the embedding presentation of the binary sequences and reconstructed the perturbed examples to its binary representation. Particularly, they appended a uniformly random sequence of bytes (payload) to the original binary sequence. Then they embedded the new binary to its embedding and performed FGSM only on the embedding of the payload. The perturbation is performed iteratively

until the detector outputs incorrect prediction. Since the perturbation is only performed on payload, instead of the input, this method will preserve the functionality of the malware. Finally, they reconstructed adverse embedding to valid binary file by mapping the adversary embedding to its closest neighbour in the valid embedding space.

Gradient-based methods leverage the gradient of the loss function as FGSM, but differs from FGSM as it uses the gradient itself instead of using the sign or magnitude of gradient. AdvGen [23] is a gradient-based method for attacking neural machine translation (NMT) models. It first generates adversarial examples by considering the similarity between the loss function's gradient, and the distance between a word and its replacing word (i.e., adversarial word). It uses language model to identify the most possible replacing words given the word, because language model enforces the semantic-preservance of the adversarial word. Then AdvGen incorporates the generated adversarial examples into the decoder of NMT model to defend the attacks.

Many works directly adopt FGSM for adversarial training, i.e., put it as regularizer when training the model. We will discuss some representatives in Section 5.

4.2.2 JSMA-based. JSMA is another pioneer work on attacking neural models for image applications (refers to Section 3.1). Reference [103] uses forward derivative as JSMA to find the most contributable sequence towards the adversary direction. The network's Jacobian had been calculated by leveraging computational graph unfolding [96]. The authors crafted adversarial sequences for two types of RNN models whose output is categorical and sequential data, respectively. For categorical RNN, the adversarial examples are generated by considering the Jacobian $Jacb_F[:, j]$ column corresponding to one of the output components j. Specifically, for each word i, they identified the direction of perturbation by

$$\text{sign}(Jacb_F(x')[i, g(x')]), \tag{14}$$

$$g(x') = \arg\max_{0,1}(p_j), \tag{15}$$

where p_j is the output probability of the target class. As in JSMA, they chose logit to replace probability in this equation. They further projected the perturbed examples onto the closest vector in the embedding space to get valid embedding. For sequential RNN, after computing the Jacobian matrix, they altered the subset of input setps $\{i\}$ with high Jacobian values $Jacb_F[i, j]$ and low Jacobian values $Jacb_F[i, k]$ for $k \neq j$ to achieve modification on a subset of output steps $\{j\}$.

Reference [43] (and Reference [44]) is the first work to attack neural malware detector. The authors first performed feature engineering and obtained more than 545K static features for software applications. They used binary indicator feature vector to represent an application. Then they crafted adversarial examples on the input feature vectors by adopting JSMA: they computed gradient of model Jacobian to estimate the perturbation direction. Later, the method chooses a perturbation η given input sample that with maximal positive gradient into the target class. In particular, the perturbations are chosen via index i, satisfying

$$i = \arg\max_{j \in [1,m], \mathrm{X}_j = y'} f'_y(\mathrm{X}_j), \tag{16}$$

where y' is the target class, m is the number of features. On the binary feature vectors, the perturbations are binary value flip operations (i.e., $0 \rightarrow 1$ or $1 \rightarrow 0$). This method preserves the functionality of the applications. To ensure that the modifications caused by the perturbations do not change the application much, which will keep the malware application's functionality complete, the authors used the L_1 norm to bound the overall number of features modified, and further bounded the number of altered features to 20. In addition, the authors provided three methods to defense against the attacks, namely, *feature reduction*, *distillation*, and *adversarial training*. They found adversarial training is the most effective defense method.

4.2.3 C&W-based. The work in Reference [130] adopted C&W method (refers to Section 3.1) for attacking predictive models of medical records. The aim is to detect susceptible events and measurements in each patient's medical records, which provide guidance for the clinical usage. The authors used standard LSTM as the predictive model. Given the patient EHR data being presented by a matrix $X^i \in \mathbf{R}^{d \times t_i}$ (d is the number of medical features and t_i is the time index of medical check), the generation of the adversarial example is formulated as

$$\min_{\hat{X}} \max\{-\epsilon, [logit(\mathbf{x}')]_y - [logit(\mathbf{x})]_{y'}\} + \lambda ||\mathbf{x}' - \mathbf{x}||_1, \tag{17}$$

where $logit(\cdot)$ denotes the logit layer output, λ is the regularization parameter that controls the L_1 norm regularization, y' is the targeted label, while y is the original label. After generating adversarial examples, the authors picked the optimal example according to their proposed evaluation scheme that considers both the perturbation magnitude and the structure of the attacks. Finally, they used the adversarial example to compute the susceptibility score for the EHR as well as the cumulative susceptibility score for different measurements.

Seq2Sick [22] attacked the seq2seq models using two targeted attacks: *non-overlapping attack* and *keywords attack*. For non-overlapping attack, the authors aimed to generate adversarial sequences that are entirely different from the original outputs. They proposed a hinge-like loss function that optimizes on the logit layer of the neural network:

$$\sum_{i=1}^{|K|} \min_{t \in [M]} \left\{ m_t \left(\max \left\{ -\epsilon, \max_{y \neq k_i} \left\{ z_t^{(y)} \right\} - z_t^{(k_i)} \right\} \right) \right\}, \tag{18}$$

where $\{z_t\}$ indicates the logit layer outputs of the adversarial example. For the keyword attack, targeted keywords are expected to appear in the output sequence. The authors also put the optimization on the logit layer and tried to ensure that the targeted keyword's logit be the largest among all words. Furthermore, they defined mask function m to solve the keyword collision problem. The loss function then becomes

$$L_{keywords} = \sum_{i=1}^{|K|} \min_{t \in [M]} \left\{ m_t \left(\max \left\{ -\epsilon, \max_{y \neq k_i} \left\{ z_t^{(y)} \right\} - z_t^{(k_i)} \right\} \right) \right\}, \tag{19}$$

where k_i denotes the ith word in output vocabulary. To ensure the generated word embedding is valid, this work also considers two regularization methods: *group lasso* regularization to enforce the group sparsity, and *group gradient* regularization to make adversaries within the permissible region of the embedding space.

4.2.4 Direction-based. HotFlip [30] performs atomic flip operations to generate adversarial examples. Instead of leveraging gradient of loss, HotFlip uses the directional derivatives. Specifically, HotFlip represents character-level operations, i.e., swap, insert and delete, as vectors in the input space and estimates the change in loss by directional derivatives with respect to these vectors. Specifically, given one-hot representation of the input, a character flip in the jth character of the ith word (a→b) can be represented by the following vector:

$$\overrightarrow{v}_{ijb} = (\mathbf{0}, ..; (\mathbf{0}, ..(0, .. - 1, 0, .., 1, 0)_j, ..\mathbf{0})_i; \mathbf{0}, ..), \tag{20}$$

where -1 and 1 are in the corresponding positions for the ath and the bth characters of the alphabet, respectively. Then the best character swap can be found by maximizing a first-order approximation of loss change via directional derivative along the operation vector:

$$\max \nabla_x J(x, y)^T \cdot \overrightarrow{v}_{ijb} = \max_{ijv} \frac{\partial J^{(b)}}{\partial x_{ij}} - \frac{\partial J^{(a)}}{\partial x_{ij}}, \tag{21}$$

where $J(x, y)$ is the model's loss function with input x and true output y. Similarly, insertion at the jth position of the ith word can also be treated as a character flip, followed by more flips, since the characters are shifted to the right until the end of the word. The character deletion is a number of character flips, since the characters are shifted to the left. Using the beam search, HotFlip efficiently finds the best directions for multiple flips.

Reference [29] extended HotFlip by adding targeted attacks. Besides the swap, insertion and deletion as provided in HotFlip, the authors proposed a *controlled attack*, which is to remove a specific word from the output, and a *targeted attack*, which is to replace a specific word by a chosen one. To achieve these attacks, they maximized the loss function $J(x, y_t)$ and minimized $J(x, y'_t)$, where t is the target word for the controlled attack, and t' is the word to replace t. Further, they proposed three types of attacks that provide multiple modifications. In *one-hot* attack, they manipulated all the words in the text with the optimal operation. In *Greedy* attack, they made another forward and backward pass, in addition to picking the best operation from the whole text. In *Beam search* attack, they replaced the search method in greedy with the beam search. In all the attacks proposed in this work, the authors set threshold for the maximum number of changes, e.g., 20% of characters are allowed to be changed.

4.2.5 Attention-based. The authors of Reference [14] proposed two white-box attacks for the purpose of comparing the robustness of CNN verses RNN. They leveraged the model's internal attention distribution to find the pivotal sentence, which is assigned a larger weight by the model to derive the correct answer. Then they exchanged the words that received the most attention with the randomly chosen words in a known vocabulary. They also performed another white-box attack by removing the whole sentence that gets the highest attention. Although they focused on attention-based models, their attacks do not examine the attention mechanism itself, but solely leverage the outputs of the attention component (i.e., attention score).

4.2.6 Reprogramming. Reference [97] adopts adversarial reprogramming (AP) to attack sequence neural classifiers. AP [31] is a recently proposed adversarial attack where an adversarial reprogramming function g_θ is trained to re-purpose the attacked DNN to perform an alternate task (e.g., question classification to name classification) without modifying the DNN's parameters. AP adopts idea from transfer learning, but keeps the parameters unchanged. The authors of Reference [97] proposed both white-box and black-box attacks. In white-box, Gumbel-Softmax is applied to train g_θ who can work on discrete data. We discuss the black-box method later. The authors evaluated their methods on various text classification tasks and confirmed the effectiveness of their methods.

4.2.7 Hybrid. Reference [38] perturbs the input text on word embeddings against the CNN model. This is a general method that is applicable to most of the attack methods developed for computer vision DNNs. The authors specifically applied FGSM and DeepFool. Directly applying methods from computer vision would generate meaningless adversarial examples. To address this issue, the authors rounded the adversarial examples to the nearest meaningful word vectors by using Word Mover's Distance (WMD) as the distance measurement. The evaluations on sentiment analysis and text classification datasets show that WMD is a qualified metric for controlling the perturbations.

Summary of White-box Attack. We summarize the reviewed white-box attack works in Table 1. We highlight four aspects, which include granularity—on which level the attack is performed; target—whether the method is target or untarget; the attacked model, perturbation control—methods to control the size of the perturbation, and applications. It is worth noting that in binary

Table 1. Summary of the Reviewed White-box Attack Methods

Strategy	Work	Granularity	Target	Attacked Models	Perturbation Control	Applications
FSGM-based	[76]	character,word	Y	CNN [156]	L_∞	TC
	[117]	word	N	CNN [156]	L_∞, Grammar and POS correctness	TC
	[114]	PE	binary	CNN [26]	Boundaries employ L_∞ and L_2	MAD
	[3]	PE embedding	binary	MalConv [109]	L_∞	MAD
	[23]	word	N	Transformer [136]	Language model	MT
JSMA-based	[103]	word embedding	binary	LSTM	–	TC
	[43, 44]	application features	binary	Feed forward	L_1	MAD
C&W-based	[130]	medical features	Y	LSTM	L_1	MSP
	[22]	word embedding	Y	OpenNMT-py [63]	L_2+gradient regularization	TS, MT
Direction-based	[30]	character	N	CharCNN-LSTM [60]	–	TC
	[29]	character	Y	CharCNN-LSTM [25]	Number of changes	MT
Attention-based	[14]	word, sentence	N	[80, 140], CNN, LSTM, and ensembles	Number of changes	MRC, QA
Reprogramming	[97]	word	N	CNN, LSTM, Bi-LSTM	–	TC
Hybrid	[38]	word embedding	N	CNN	WMD	TC, SA

PE: portable executable; TC: text classification; SA: sentiment analysis; TS: text summarisation; MT: machine translation; MAD: malware detection; MSP: Medical Status Prediction; MRC: machine reading comprehension; QA: question answering; WMD: Word Mover's Distance; –: not available.

classifications, target and untarget methods show the same effect, so we point out their *target* as "binary" in the table.

4.3 Black-box Attack

Black-box attack does not require the details of the neural networks, but can access the input and output. This type of attacks often rely on heuristics to generate adversarial examples, and it is more practical as in many real-world applications the details of the DNN is a black box to the attacker. In this article, we group black-box attacks on textual DNNs into five categories.

4.3.1 Concatenation Adversaries. Reference [54] is the first work to attack reading comprehension systems. The authors proposed *concatenation adversaries*, which is to append distracting but meaningless sentences at the end of the paragraph. These distracting sentences do not change the semantics of the paragraph and the question answers, but will fool the neural model. The distracting sentences are either carefully generated informative sentences or arbitrary sequence of words using a pool of 20 random common words. Both perturbations were obtained by iteratively querying the neural network until the output changes. Figure 2 illustrates an example from Reference [54] that after adding distracting sentences (in blue) the answer changes from correct one (green) to incorrect one (red). The authors of Reference [142] improved the work by varying the locations where the distracting sentences are placed and expanding the set of fake answers for generating the distracting sentences, rendering new adversarial examples that can help training more robust neural models. Also, Reference [14] utilized the distracting sentences to evaluate the robustness of their reading comprehension model. Specifically, they use a pool of ten random common words in conjunction with all question words and the words from all incorrect answer candidates to generate the distracting sentences. In this work, a simple word-level black-box attack

Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

Fig. 2. Concatenation adversarial attack on reading comprehension DNN. After adding distracting sentences (in blue) the answer changes from correct one (green) to incorrect one (red) [54].

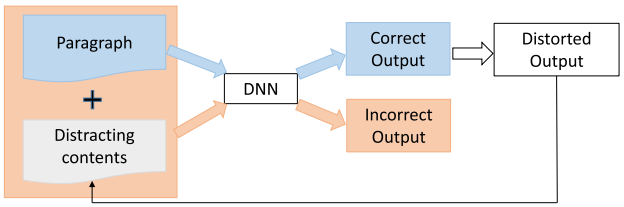

Fig. 3. General principle of concatenation adversaries. Correct output are often utilized to generate distorted output, which later will be used to build distracting contents. Appending distracting contents to the original paragraph as adversarial input to the attacked DNN and cause the attacked DNN to produce incorrect output.

is also performed by replacing the most frequent words via their synonyms. As aforementioned, the authors also provided two white-box strategies. Figure 3 illustrates the general workflow for concatenation attack. Correct output (i.e., answer in MRC tasks) are often leveraged to generate distorted output, which later will be used to build distracting contents. Appending distracting contents to the original paragraph forms the adversarial input to the attacked DNN. The distracting contents will not *distract* human being and ideal DNNs, but can make vulunerable DNNs to produce incorrect output.

4.3.2 Edit Adversaries. The work in Reference [12] perturbed the input data of neural machine translation applications in two ways: *Synthetic*, which performed the character order changes, such as swap, middle random (i.e., randomly change orders of characters except the first and the last), fully random (i.e., randomly change orders of all characters) and keyboard type. They also collected typos and misspellings as adversaries, by naturally leveraging the typos from the datasets. Furthermore, the authors of Reference [98] attacked the neural models for dialogue generation. They applied various perturbations in dialogue context, namely, Random Swap (randomly transposing neighboring tokens) and Stopword Dropout (randomly removing stopwords), Paraphrasing (replacing words with their paraphrases), Grammar Errors (e.g., changing a verb to the wrong tense) for the *Should-Not-Change* attacks, and the Add Negation strategy (negates the root verb of the source input) and Antonym strategy (changes verbs, adjectives, or adverbs to their antonyms) for *Should-Change* attacks. DeepWordBug [34] is a simple method that uses character transformations to generate adversarial examples. The authors first identified the important "tokens," i.e., words or characters that affect the model prediction mostly by scoring functions developed by measuring

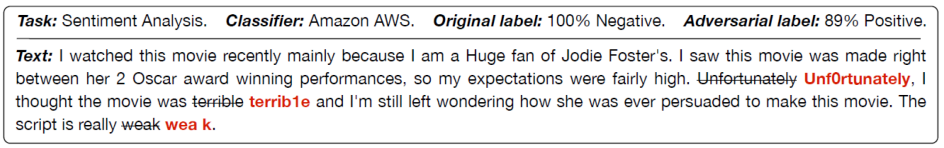

Fig. 4. Edit adversarial attack on sentiment analysis DNN. After editing words (red), the prediction changes from 100% of Negative to 89% of Positive [73].

the DNN classifier's output. Then they modified the identified tokens using four strategies: replace, delete, add and swap. The authors evaluated their method on a variety of NLP tasks, e.g., text classification, sentiment analysis and spam detection. By following the work in Reference [34], the work in Reference [73] refines the scoring function. Moreover, this work provided white-box attack adopting JSMA. One contribution of this work lies on the perturbation restriction by using four textual similarity measurement: edit distance of text; Jaccard similarity coefficient; Euclidean distance on word vector; and cosine similarity on word embedding. Their method had been evaluated only on sentiment analysis task. Figure 4 is an example of edit adversarial example from Reference [73], where only a few edit operations mislead the classifier to give wrong predictions.

When attacking the text classification models, the work in Reference [110] provides a probability method to select the word to replace. It first collects all synonyms of the words in the existing corpus. Then it selects a proposed substitute words from the synonyms by measuring its impact on the classification probability. The ones lead to the most significant changes will be selected. The authors also incorporate the word saliency to determine the replacement order. Word saliency is the degree of change in the output classification probability if a word is set to unknown.

The authors of Reference [91] proposed a method for automatically generating adversarial examples that violate a set of given First-Order Logic constraints in natural language inference (NLI). They proposed an inconsistency loss to measure the degree to which a set of sentences causes a model to violate a rule. The adversarial example generation is the process for finding the mapping between variables in rules to sentences that maximize the inconsistency loss. The sentences are composed by sentences with a low perplexity (defined by a language model). To generate low-perplexity adversarial sentence examples, they used three edit perturbations: (i) change one word in one of the input sentences; (ii) remove one parse subtree from one of the input sentences; (iii) insert one parse sub-tree from one sentence in the corpus in the parse tree of the another sentence.

The work in Reference [5] uses genetic algorithm (GA) for minimising the number of word replacement from the original text, but at the same time can change the result of the attacked model. The authors adopted *crossover* and *mutation* operations in GA to generate perturbations. They measured the effectiveness of the word replacement accoding to the impact on attacked DNNs. Their attack focused on sentiment analysis and textual entailment DNNs.

In Reference [19], the authors proposed a framerwork for adversarial attack on Differentiable Neural Computer (DNC). DNC is a computing machine with DNN as its central controller operating on an external memory module for data processing. Their method uses two new automated and scalable strategies to generate grammatically correct adversarial attacks in question answering domain, utilizing metamorphic transformation. The first strategy, *Pick-n-Plug*, consists of a pick operator to draw adversarial sentences from a particular task (source task) and the plug operator injects these sentences into a story from another task (target task), without changing its correct answers. Another strategy, *Pick-Permute-Plug*, extends the adversarial capability of *Pick-n-Plug* by an additional permute operator after picking sentences (gpick) from a source task. Words in a particular adversarial sentence can be permuted with its synonyms to generate a wider range of possible attacks.

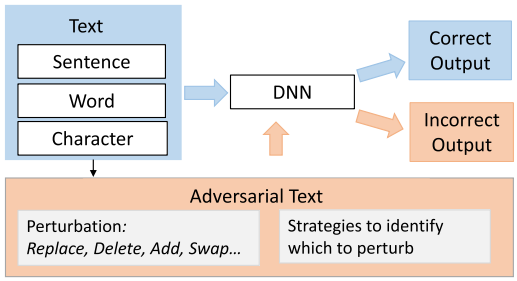

Fig. 5. General principle of edit adversaries. Perturbations are performed on sentences, words, or characters by edit strategies such as replace, delete, add, and swap.

An edit-based approach MHA (Metropolis-Hastings Attack) was proposed aiming to provide fluent and effective adversarial attacks [155]. MHA is based on language model and Metropolis-Hastings (M-H) sampling. The authors used M-H sampling to generate the word to replace the old word (for replace operation) and the random word (for insert operation). Language models are used to enforce the fluency of the sentence after replace/insert/delete operations. They proposed both black-box and white-box versions. The only difference between the two is the definition of the pre-selection function when choosing the most possible words to operate.

Figure 5 illustrates the general workflow for edit adversaries. Perturbations are performed on sentences, words or characters by edit strategies, such as replacement, deletion, insertion, and swap.

4.3.3 Paraphrase-based Adversaries. SCPNs [53] produces a paraphrase of the given sentence with desired syntax by inputting the sentence and a targeted syntactic form into an encoder-decoder architecture. Specifically, the method first encodes the original sentence, then inputs the paraphrases generated by back-translation and the targeted syntactic tree into the decoder, whose output is the targeted paraphrase of the original sentence. One major contribution lies on the selection and processing of the parse templates. The authors trained a parse generator separately from SCPNs and selected 20 most frequent templates in the PARANMT-50M dataset. After generating paraphrases using the selected parse templates, they further pruned non-sensible sentences by checking n-gram overlap and paraphrase-based similarity. The attacked classifier can correctly predict the label of the original sentence but fails on its paraphrase, which is regarded as the adversarial example. SCPNs has been evaluated on sentiment analysis and textual entailment DNNs and showed significant impact on the attacked models. Although this method uses target strategy to generate adversarial examples, it does not specify targeted output. Therefore, we group it as untargeted attack. Furthermore, the work in Reference [127] uses the idea of paraphrase generation techniques that create semantically equivalent adversaries (SEA). The authors generated paraphrases of an input sentence x, and got predictions from f until the original prediction is changed. At the same time, they considered the semantically equivalent to x' that is 1 if x is semantically equivalent to x' and 0 otherwise as shown in Equation (22). After that, this work proposes a semantic-equivalent rule-based method for generalizing these generated adversaries into semantically equivalent rules to understand and fix the most impactful bug:

$$\mathbf{SEA}(\mathbf{x}, \mathbf{x}') = \mathbf{1}[SemEq(x, x') \wedge f(\mathbf{x}) \neq f(\mathbf{x}')]. \tag{22}$$

Figure 6 depicts the general principle of paraphrase-based adversaries. The paraphrases are considered as adversarial examples. When generating the paraphrases, controlled perturbations are incorporated.

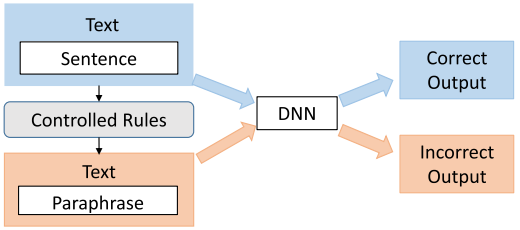

Fig. 6. General principle of paraphrase-based adversaries. Carefully designed (controlled) paraphrases are regarded as adversarial examples, which fool DNN to produce incorrect output.

4.3.4 GAN-based Adversaries. Some works propose to leverage Generative Adversarial Network (GAN) [39] to generate adversaries [157]. The purpose of adopting GAN is to make the adversarial examples more natural. In Reference [157], the model proposed to generate adversarial examples consists of two key components: a GAN, which generate fake data samples, and an inverter that maps input x to its latent representation z'. The two components are trained on the original input by minimizing reconstruction error between original input and the adversarial examples. Perturbation is performed in the latent dense space by identifying the perturbed sample $\hat{z}$ in the neighborhood of z'. Two search approaches, namely, *iterative stochastic search* and *hybrid shrinking search*, are proposed to identify the proper $\hat{z}$. However, it requires querying the attacked model each time to find the $\hat{z}$ that can make the model produce incorrect prediction. Therefore, this method is quite time-consuming. The work is applicable to both image and textual data as it intrinsically eliminates the problem raised by the discrete attribute of textual data. The authors evaluated their method on three applications, namely: textual entailment, machine translation, and image classification.

4.3.5 Substitution. The work in Reference [52] proposes a black-box framework that attacks RNN model for malware detection. The framework consists of two models: a generative RNN and a substitute RNN. The generative RNN aims to generate adversarial API sequence from the malware's API sequence. It is based on the seq2seq model proposed in Reference [131]. It particularly generates a small piece of API sequence and inserts the sequence after the input sequence. The substitute RNN, which is a bi-directional RNN with attention mechanism, is to mimic the behavior of the attacked RNN. Therefore, generating adversarial examples will not query the original attacked RNN, but its substitution. The substitute RNN is trained on both malware and benign sequences, as well as the Gumbel-Softmax outputs of the generative RNN. Here, Gumbel-softmax is used to enable the joint training of the two RNN models, because the original output of the generative RNN is discrete. Specifically, it enables the gradient to be back-propagated from generative RNN to substitute RNN. This method performs attack on API, which is represented as a one-hot vector, i.e., given M APIs, the vector for the ith API is an M-dimensional binary vector that the ith dimension is 1 while other dimensions are 0s.

4.3.6 Reprogramming. As aforementioned, Reference [97] provides both white-box and black-box attacks. We describe black-box attack here. In a black-box attack, the authors formulated the sequence generation as a reinforcement learning problem, and the adversarial reprogramming function g_θ is the policy network. Then they applied REINFORCE-based optimisation to train g_θ.

Summary of Black-box Attack. We summarize the reviewed black-box attack works in Table 2. We highlight four aspects include granularity on which level the attack is performed; whether

Table 2. Summary of the Reviewed Black-box Attack Methods

Strategy	Work	Granularity	Target	Attacked Models	Perturbation Control	Applications
Concatenation	[54]	word	N	BiDAF, Match-LSTM	–	MRC
	[142]	word, character	N	BiDAF+Self-Attn+ELMo [108]	–	MRC
	[14]	word, sentence	N	[80, 140], CNN, LSTM and ensembles	Number of changes	MRC, QA
Edit	[12]	character, word	N	Nematus [120], char2char [70], charCNN [60]	–	MT
	[98]	word, phrase	N	VHRED [123]+attn, RL in [74], DynoNet [47]	–	DA
	[34]	character, word	N	Word-level LSTM, Character-level CNN	–	SA, TC
	[73]	character, word	N	Word-level LSTM, Character-level CNN	EdDist, JSC, EuDistV, CSE	SA
	[91]	word, phrase	N	cBiLSTM, DAM, ESIM	Perplexity	NLI
	[5]	word	N	LSTM	EuDistV	SA, TE
	[19]	word, sentence	N	DNC	–	QA
	[155]	word	Y	LSTM, BiDAF	M-H	SA, NLI
	[110]	word	N	CNN, LSTM	synonyms	TC
Paraphrase-based	[53]	word	N	LSTM	Syntax-ctrl paraphrase	SA and TE
	[127]	word	N	BiDAF, Visual7W [158], fastText [41]	Self-defined semantic-equivalency	MRC, SA, VQA
GAN-based	[157]	word	N	LSTM, TreeLSTM, Google Translate (En-to-Ge)	GAN-constraints	TE, MT
Substitution	[52]	API	N	LSTM, BiLSTM and variants	–	MD
Reprogramming	[97]	word	N	CNN, LSTM, Bi-LSTM	–	TC

MRC: machine reading comprehension; QA: question answering; VQA: visual question answering; DA: dialogue generation; TC: text classification; MT: machine translation; SA: sentiment analysis; NLI: natural language inference; TE: textual entailment; MD: malware detection; EdDist: edit distance of text; JSC: Jaccard similarity coeffcient; EuDistV: Euclidean distance on word vector; CSE: cosine similarity on word embedding; "–": not available.

the method is target driven or not; the attacked neural model, perturbation control, and NLP applications.

4.4 Multi-modal Attacks

Some works attack DNNs that are dealing with cross-modal data. For example, the neural models contain an internal component that performs image-to-text or speech-to-text conversion. Although these attacks are not for pure textual data, we briefly introduce the representative ones for the purpose of a comprehensive review.

4.4.1 Image-to-text. Image-to-text models is a class of techniques that generate textual description for an image based on the semantic content of the latter.

Optical Character Recognition (OCR). Recognizing characters from images is a task named Optical Character Recognition (OCR). OCR is a multimodal learning task that takes an image as

input and output the recognized text. The authors in Reference [129] proposed a white-box attack on OCR and follow-up NLP applications. They first used the original text to render a clean image (conversion DNNs). Then they found words in the text that have antonyms in WordNet and satisfy edit distance threshold. Only the antonyms that are valid and keep semantic inconsistencies will be kept. Later, the method locates the lines in the clean image containing the aforementioned words, which can be replaced by their selected antonyms. The method then transforms the target word to target sequence. Given the input/target images and sequences, the authors formed the generating of adversarial example as an optimization problem:

$$\min_{\omega} c \cdot J_{CTC} f(\mathbf{x}', t') + ||\mathbf{x} - \mathbf{x}'||_2^2, \tag{23}$$

$$\mathbf{x}' = (\alpha \cdot \tanh(\omega) + \beta)/2, \tag{24}$$

$$\alpha = (\mathbf{x}_{max} - \mathbf{x}_{min})/2, \beta = (\mathbf{x}_{max} + \mathbf{x}_{min})/2,$$
$$J_{CTC}(f(\mathbf{x}, t)) = -\log p(t|\mathbf{x}), \tag{25}$$

where $f(\mathbf{x})$ is the neural system model, $J_{CTC}(\cdot)$ is the Connectionist Temporal Classification (CTC) loss function, $\mathbf{x}$ is the input image, t is the ground truth sequence, $\mathbf{x}'$ is the adversarial example, t' is the target sequence, ω, α, β are parameters controlling adversarial examples to satisfy the box-constraint of $\mathbf{x}' \in [\mathbf{x}_{min}, \mathbf{x}_{max}]^p$, where p is the number of pixels ensuring valid x'. After generating adversarial examples, the method replaces the images of the corresponding lines in the text image. The authors evaluated this method in three aspects: single word recognition, whole document recognition, and NLP applications, which were based on the recognized text (sentiment analysis and document categorisation specifically). They also found that the proposed method suffers from limitations such as transferability and physical realizability.

Scene Text Recognition (STR). STR is also an image-to-text application. In STR, the entire image is mapped to word strings directly. In contrast, the recognition in OCR is a pipeline process: first segments the words to characters, then performs the recognition on single characters. AdaptiveAttack [153] evaluates the possibility of performing adversarial attack for scene text recognition. The authors proposed two attacks, namely, basic attack and adaptive attack. Basic attack is similar to the work in Reference [129] and it also formulates the adversarial example generation as an optimization problem:

$$\min_{\omega} J_{CTC} f(\mathbf{x}', t') + \lambda \mathcal{D}(\mathbf{x}, \mathbf{x}'), \tag{26}$$

$$\mathbf{x}' = \tanh(\omega), \tag{27}$$

where $\mathcal{D}(\cdot)$ is Euclidean distance. The differences to [129] lie on the definition of $\mathbf{x}'$ (Equation (24) vs. Equation (27)), and the distance measurement between $\mathbf{x}$, $\mathbf{x}'$ (L_2 norm vs. Euclidean distance), and the parameter λ, which balances the importance of being adversarial example and close to the original image. As searching for proper λ is quite time-consuming, the authors proposed another method to adaptively find λ. They named this method Adaptive Attack, in which they defined the likelihood of a sequential classification task following a Gaussian distribution and derived the adaptive optimization for sequential adversarial examples as

$$\min \frac{||\mathbf{x}-\mathbf{x}'||_2^2}{\lambda_1^2} + \frac{J_{CTC} f(\mathbf{x}', t')}{\lambda_2^2} + \log \lambda_1^2 + T \log \lambda_2^2 + \frac{1}{\lambda_2^2}, \tag{28}$$

where λ_1 and λ_2 are two parameters to balance perturbation and CTC loss, T is the number of valid paths given targeted sequential output. Adaptive Attack can be applied to generate adversarial examples on both non-sequential and sequential classification problems. Here, we only highlight the equation for sequential data (Equation (28)). The authors evaluated their proposed methods on

tasks targeting the text insertion, deletion and substitution in output. The results demonstrated that Adaptive Attack is much faster than basic attack.

Image Captioning. Image captioning is another multimodal learning task that takes an image as input and generates a textual caption describing its visual contents. Show-and-Fool [20] generates adversarial examples to attack the CNN-RNN–based image captioning model. The CNN-RNN model attacked uses a CNN as encoder for image feature extraction and a RNN as decoder for caption generation. Show-and-Fool has two attack strategies: *targeted caption* (i.e., the generated caption matches the target caption) and *targeted keywords* (i.e., the generated caption contains the targeted keywords). In general, they formulated the two tasks using the following formulation:

$$\begin{aligned} \min_{\omega} c \cdot J(\mathbf{x}') + ||\mathbf{x}' - \mathbf{x}||_2^2, \\ \mathbf{x}' = \mathbf{x} + \eta, \\ x = \tanh(y), \quad \mathbf{x}' = \tanh(\omega + y), \end{aligned} \tag{29}$$

where $c > 0$ is a pre-specified regularization constant, η is the perturbation, ω, y are parameters controlling $\mathbf{x}' \in [-1, 1]^n$. The difference between these two strategies is the definition of the loss function $J(\cdot)$. For targeted caption strategy, provided the targeted caption as $S = (S_1, S_2, \ldots S_t, \ldots S_N)$, where S_t refers to the index of the tth word in the vocabulary and N is the length of the caption, the loss is formulated as

$$J_{S,logit}(\mathbf{x}') = \sum_{t=2}^{N-1} \max\left\{ -\epsilon, \max_{k \neq S_t} \left\{ z_t^{(k)} \right\} - z_t^{(S_t)} \right\}, \tag{30}$$

where S_t is the target word, $z_t^{(S_t)}$ is the *logit* of the target word. In fact, this method mininizes the difference between the maximumn *logit* except S_t, and the logit of S_t. For the targeted keywords strategy, given the targeted keywords $\mathcal{K} := K_1, \ldots, K_M$, the loss function is

$$J_{K,logit}(\mathbf{x}') = \sum_{j=1}^{M} \min_{t \in [N]} \left\{ \max\left\{ -\epsilon, \max_{k \neq K_j} \left\{ z_t^{(k)} \right\} - z_t^{(K_j)} \right\} \right\}. \tag{31}$$

The authors performed extensive experiments on Show-and-Tell [137] and varied the parameters in the attacking loss. They found that Show-and-Fool is not only effective on attacking Show-and-Tell, the CNN-RNN–based image captioning model, but is also highly transferable to another model Show-Attend-and-Tell [147].

Visual Question Answering (VQA). Given an image and a natural language question about the image, VQA is to provide an accurate answer in natural language. The work in Reference [148] proposes an iterative optimization method to attack two VQA models. The objective function proposed maximizes the probability of the target answer and unweights the preference of adversarial examples with smaller distance to the original image when this distance is below a threshold. Specifically, the objective contains three components. The first one is similar to Equation (26), that replaces the loss function to the loss of the VQA model and uses $||\mathbf{x} - \mathbf{x}'||_2/\sqrt{N}$ as distance between $\mathbf{x}'$ and $\mathbf{x}$. The second component maximizes the difference between the softmax output and the prediction when it is different with the target answer. The third component ensures the distance between $\mathbf{x}'$ and $\mathbf{x}$ is under a lower bound. The attacks are evaluated by checking whether better success rate is obtained over the previous attacks, and the confidence score of the model to predict the target answer. Based on the evaluations, the authors concluded that attention, bounding box localization and compositional internal structures are vulnerable to adversarial attacks. This work also attacks an image captioning neural model. We refer to the original paper for further information.

Table 3. Summary of the Reviewed Cross-modal Attacks

Multi-modal	Applications	Work	Target	Access	Attacked Models	Perturbation Control
Image-to-Text	Optical Character Recognition	[129]	Y	white-box	Tesseract [135]	L_2, EdDist
	Scene Text Recognition	[153]	Y	white-box	CRNN [124]	L_2
	Image Captioning	[20]	Y	white-box	Show-and-Tell [137]	L_2
	Visual Question Answering	[148]	Y	white-box	MCB [33], N2NMN [51]	L_2
	Visual-Semantic Embeddings	[125]	N	black-box	VSE++ [32]	–
Speech-to-Text	Speech Recognition	[18]	Y	white-box	DeepSpeech [46]	L_2

EdDist: edit distance of text; -: not available.

Visual-semantic Embeddings (VSE). The aim of VSE is to bridge natural language and the underlying visual world. In VSE, the embedding spaces of both images and descriptive texts (captions) are jointly optimized and aligned. Reference [125] attacks the latest VSE model by generating adversarial examples in the test set and evaluates the robustness of the VSE models. They performed the attack on textual part by introducing three method: (i) replace nouns in the image captions utilizing the hypernymy/hyponymy relations in WordNet; (ii) change the numerals to different ones and singularize or pluralize the corresponding nouns when necessary; (iii) detect the relations and shuffle the non-interchangeable noun phrases or replace the prepositions. This method can be considered as a black-box edit adversary.

4.4.2 Speech-to-text. Speech-to-text is also known as speech recognition. The task is to recognize and translate the spoken language into text automatically. Reference [18] attacks Deep Speech, a state-of-the-art speech-to-text transcription neural network based on LSTM. Given a natural waveform, the authors constructed an audio perturbation that is almost inaudible but can be recognized by adding into the original waveform. The perturbation is constructed by adopting the idea from C&W method (refer to Section 3.1), which measures the image distortion by the maximum amount of changed pixels. Adapting this idea, they measured the audio distortion by calculating relative loudness of an audio and proposed to use Connectionist Temporal Classification loss for the optimization task. Then they solved this task with Adam optimizer [61].

Summary of Multi-modal Attack. We summarize the reviewed black-box attack works in Table 3. We list the representative works and highlight key aspects, including target or untarget, access DNN or not, control of perturbation, and the attacked neural models.

4.5 Benchmark Datasets by Applications

In recent years, neural networks gain success in different NLP domains and the popular applications include text classification, reading comprehension, machine translation, text summarization, question answering, dialogue generation, to name a few. In this section, we review the current works on generating adversarial examples on the neural networks in the perspective of NLP applications. Table 4 summarizes the works we reviewed in this article according to their application domain. We further list the benchmark datasets used in these works in the table as auxiliary information—thus, we refer readers to the links/references we collect for the detailed descriptions of the datasets. Note that the auxiliary datasets that help to generate adversarial examples are not included. Instead, we only present the datasets used to evaluate the attacked neural networks.

Text Classification. Majority of the surveyed works attack the deep neural networks for text classification, since these tasks can be framed as a classification problem. Sentiment analysis aims to classify the sentiment to several groups (e.g., in three-group scheme: neural, positive,

Table 4. Attacked Applications and Benchmark Datasets

Applications		Representative Works	Benchmark Datasets
Classification	Text Classification	[30, 34, 38, 76, 97, 118]	DBpedia, Reuters Newswires, AG's news, Sogou News, Yahoo! Answers, RCV1, Surname Classification Dataset
	Sentiment Analysis	[30, 34, 53, 97, 103, 117, 118, 127]	SST, IMDB Review, Yelp Review, Elec, Rotten Tomatoes Review, Amazon Review, Arabic Tweets Sentiment
	Spam Detection	[34]	Enron Spam, Datasets from [156]
	Gender Identification	[117]	Twitter Gender
	Grammar Error Detection	[118]	FCE-public
	Medical Status Prediction	[130]	Electronic Health Records (EHR)
	Malware Detection	[4, 43, 44, 52, 114]	DREBIN, Microsoft Kaggle
	Relation Extraction	[10, 145]	NYT Relation, UW Relation, ACE04, CoNLL04 EC, Dutch Real Estate Classifieds, Adverse Drug Events
Machine Translation		[12, 22, 29, 157]	TED Talks, WMT'16 Multimodal Translation Task
Machine Comprehension		[14, 19, 54, 142]	SQuAD, MovieQA Multiple Choice, Logical QA
Text Summarization		[22]	DUC2003, DUC2004, Gigaword
Text Entailment		[53, 56, 91, 157]	SNLI, SciTail, MultiNLI, SICK
POS Tagging		[151]	WSJ portion of PTB, Treebanks in UD
Dialogue System		[98]	Ubuntu Dialogue, CoCoA,
Cross-model	Optical Character Recognition	[129]	Hillary Clinton's emails
	Scene Text Recognition	[153]	Street View Text, ICDAR 2013, IIIT5K
	Image Captioning	[20, 148]	MSCOCO, Visual Genome
	Visual Question Answering	[148]	Datasets from [6], Datasets from [158]
	Visual-Semantic Embedding	[125]	MSCOCO
	Speech Tecognition	[18]	Mozilla Common Voice

and negative). Gender identification, grammatical error detection, and malware detection can be framed as binary classification problems. Relation extraction can be formulated as single or multi-classification problem. Predict medical status is a multi-class problem that the classes are defined by medical experts. These works usually use multiple datasets to evaluate their attack strategies to show the generality and robustness of their methods. Reference [76] uses the DBpedia ontology dataset [71] to classify the document samples into 14 high-level classes. Reference [38] uses the IMDB movie reviews [83] for sentiment analysis, and Reuters-2 and Reuters-5 newswires dataset provided by NLTK package[10] for categorization. Reference [103] uses a un-specified movie review dataset for sentiment analysis. Reference [117] also uses IMDB movie review dataset for sentiment analysis. The work also performs gender classification on and a Twitter dataset.[11] Reference [34] performs spam detection on Enron Spam Dataset [89] and adopts six large datasets from Reference [156], i.e., AG's news,[12] Sogou news [138], DBPedia ontology dataset, Yahoo! Answers[13] for text categorization and Yelp reviews,[14] Amazon reviews [88] for sentiment analysis.

[10] https://www.nltk.org/.
[11] https://www.kaggle.com/crowdflower/twitter-user-gender-cla2013.
[12] https://www.di.unipi.it/Egulli/.
[13] Yahoo! Answers Comprehensive Questions and Answers version 1.0 dataset through the Yahoo! Webscope program.
[14] Yelp Dataset Challenge in 2015.

Reference [30] also uses AG's news for text classification. Further, the authors used Stanford Sentiment Treebank (SST) dataset [128] for sentiment analysis. Reference [118] conducts evaluation on three tasks: sentiment analysis (IMDB movie review, Elec [55], Rotten Tomatoes [102]), text categorization (DBpedia Ontology dataset and RCV1 [72]) and grammatical error detection (FCE-public [150]). Reference [130] generates adversarial examples on the neural medical status prediction system with real-world electronic health records data.

Many works target the malware detection models. References [43, 44] perform attack on neural malware detection systems. They use DREBIN dataset, which contains both benigh and malicious android applications [7]. Reference [114] collects benigh windows application files and uses Microsoft Malware Classification Challenge dataset [113] as the malicious part. Reference [52] crawls 180 programs with corresponding behavior reports from a website for malware analysis.[15] Seventy percent of the crawled programs are malware. Reference [97] targets the text classification neural models and uses four datasets to evaluate their attack methods: Surname Classification Dataset,[16] Experimental Data for Question Classification [75], Arabic Tweets Sentiment Classification Dataset [1], and IMDB movie review dataset. In Reference [145], the authors modelled the relation extraction as a classification problem, where the goal is to predict the relations existing between entity pairs for the given text mentions. They used two relation datasets: NYT dataset [111] and UW dataset [78]. Reference [10] targets at improving the efficacy of the neural networks for joint entity and relation extraction. Different to the method in Reference [145], the authors models the relation extraction task as a multi-label head selection problem. The four datasets are used in their work: ACE04 dataset [28], CoNLL04 EC tasks [115], Dutch Real Estate Classifieds (DREC) dataset [11], and Adverse Drug Events (ADE) [45].

Machine Translation. Machine Translation works on parallel datasets, one of which uses source language and the other one is in the target language. Reference [12] uses the TED talks parallel corpus prepared for IWSLT 2016 [87] for testing the NMT systems. It also collects French, German, and Czech corpus for generating natural noises to build a look-up table, which contains possible lexical replacements. These replacements are later used for generating adversarial examples. Reference [29] also uses the same TED talks corpus and used German to English, Czech to English, and French to English pairs.

Machine Comprehension. Machine comprehension datasets usually provide context documents or paragraphs to the machines. Based on the comprehension of the contexts, machine comprehension models can answer a question. Jia and Liang are one of the first to consider the textual adversary and they targeted the neural machine comprehension models [54]. They used the Stanford Question Answering Dataset (SQuAD) to evaluate the impact of their attack on the neural machine comprehension models. SQuAD is a widely recognized benchmark dataset for machine comprehension. Reference [142] follows the previous works and also works on SQuAD dataset. Although the focus of Reference [14] is to develop a robust machine comprehension model rather than attacking MC models, the authors used the adversarial examples to evaluate their proposed system. They used MovieQA multiple choice question answering dataset [134] for the evaluation. Reference [19] targets attacks on differentiable neural computer (DNC), which is a novel computing machine with DNN. It evaluates the attacks on logical question answering using bAbI tasks.[17]

Text Summarization. The goal for text summarization is to summarize the core meaning of a given document or paragraph with succinct expressions. Reference [22] evaluates their attack

[15] https://malwr.com/.
[16] Classifying names with a character-level rnn-pytroch tutorial.
[17] https://research.fb.com/downloads/babi/.

on multiple applications including text summarization and it uses DUC2003,[18] DUC2004,[19] and Gigaword[20] for evaluating the effectiveness of adversarial examples.

Text Entailment. The fundamental task of text entailment is to decide whether a premise text entails a hypothesis, i.e., the truth of one text fragment follows from another text. Reference [56] assesses various models on two entailment datasets: Stanford Natural Lauguage Inference (SNLI) [15] and SciTail [58]. Reference [91] also uses the SNLI dataset. Furthermore, it uses the MultiNLI [144] dataset.

Part-of-Speech (POS) Tagging. The purpose for POS tagging is to resolve the part-of-speech for each word in a sentence, such as noun, verb. It is one of the fundamental NLP tasks to facilitate other NLP tasks, e.g., syntactic parsing. Neural networks are also adopted for this NLP task. Reference [151] adopts the method in Reference [93] to build a more robust neural network by introducing adversarial training, but it applies the strategy (with minor modifications) in POS tagging. By training on the mixture of clean and adversarial examples, the authors found that adversarial examples not only help improving the tagging accuracy but also contribute to downstream task of dependency parsing and is generally effective in different sequence labelling tasks. The datasets used in their evaluation include: the Wall Street Journal (WSJ) portion of the Penn Treebank (PTB) [85] and treebanks from Universal Dependencies (UD) v1.2 [99].

Dialogue Generation. Dialogue generation is a fundamental component for real-world virtual assistants such as Siri[21] and Alexa.[22] It is the text generation task that automatically generates a response for the post given by the user. Reference [98] is one of the first to attack the generative dialogue models. It uses the Ubuntu Dialogue Corpus [82] and Dynamic Knowledge Graph Network with the Collaborative Communicating Agents (CoCoA) dataset [47] for the evaluation of their two attack strategies.

Cross-model Applications. Reference [129] evaluates the OCR systems with adversarial examples using Hillary Clinton's emails,[23] which is in the form of images. It also conducts the attack on NLP applications using Rotten Tomatoes and IMDB review datasets. The work in Reference [153] attacks the neural networks designed for scene text recognition. The authors conducted experiments on three standard benchmarks for cropped word image recognition, namely, the Street View Text dataset (SVT) [139] the ICDAR 2013 dataset (IC13) [57] and the IIIT 5K-word dataset (IIIT5K) [92]. Reference [20] attacks the image captioning neural models. The dataset used is the Microsoft COCO (MSCOCO) dataset [77]. Reference [148] works on the problems of attacking neural models for image captioning and visual question answering. For the first task, it uses Visual Genome dataset [64]. For the second task, it uses the VQA datasets collected and processed in Reference [6]. Reference [125] works on Visual-Semantic Embedding applications, where the MSCOCO dataset is used. Reference [18] targets the speech recognition problem. The datasets used is the Mozilla Common Voice dataset.[24]

Multi-Applications. Some works adapt their attack methods into different applications, namely, they evaluate their method's transferability across applications. Reference [22] attacks the

[18] http://duc.nist.gov/duc2003/tasks.html.
[19] http://duc.nist.gov/duc2004/.
[20] https://catalog.ldc.upenn.edu/LDC2003T05.
[21] https://www.apple.com/au/siri/.
[22] https://en.wikipedia.org/wiki/Amazon_Echo.
[23] https://www. kaggle.com/kaggle/hillary-clinton-emails/data.
[24] https://voice.mozilla.org/en.

sequence-to-sequence models. Specifically, it evaluates the attack on two applications: text summarization and machine translation. For text summarization, as mentioned before, it uses three datasets DUC2003, DUC2004, and Gigaword. For the machine translation, it samples a subset form WMT'16 Multimodal Translation dataset.[25] Reference [53] proposes to generate syntactically adversarial paraphrases and evaluates the attack on sentiment analysis and text entailment applications. It uses SST for sentimental analysis and SICK [86] for text entailment. Reference [157] is a generic approach for generating adversarial examples on neural models. The applications investigated include image classification (MINIST digital image dataset), textual entailment (SNLI), and machine translation. Reference [93] evaluates the attacks on five datasets, covering both sentiment analysis (IMDB movie review, Elec product review, Rotten Tomatoes movie review) and text categorization (DBpedia Ontology, RCV1 news articles). Reference [127] targets both sentiment analysis and visual question answering. For sentiment analysis, it uses the Rotten Tomato movie reviews and IMDB movie reviews datasets. For visual question answering, it tests on the dataset provided by Zhu et al. [158].

5 DEFENSE

An essential purpose for generating adversarial examples for neural networks is to utilize these adversarial examples to enhance the model's robustness [40]. There are two common ways in textual DNN to achieve this goal: *adversarial training* and *knowledge distillation*. Adversarial training incorporates adversarial examples in the model training process. Knowledge distillation manipulates the neural network model and trains a new model. In this section, we introduce some representative studies belonging to these two directions. For more comprehensive defense strategies on machine learning and deep-learning models and applications, please refer to References [2, 13].

5.1 Adversarial Training

Szegedy et al. [132] invented *adversarial training*, a strategy that consists of training a neural network to correctly classify both normal examples and adversarial examples. Goodfellow et al. [40] employed explicit training with adversarial examples. In this section, we describe works utilizing *data augmentation*, *model regularization*, and *robust optimization* for the defense purpose on textual adversarial attacks.

5.1.1 Data Augmentation. Data augmentation extends the original training set with the generated adversarial examples and try to let the model see more data during the training process. Data augmentation is commonly used against black-box attacks with additional training epochs on the attacked DNN with adversarial examples.

The authors of Reference [54] attempted to enhance the reading comprehension model with training on the augmented dataset that includes the adversarial examples. They showed that data augmentation is effective and robust against the attack that uses the same adversarial examples. However, their work also demonstrated that this augmentation strategy would be still vulnerable against the attacks with other kinds of adversarial examples. Reference [142] shared similar idea to augment the training dataset, but selected further informative adversarial examples as discussed in Section 4.3.1.

The work in Reference [56] trained the text entailment system augmented with adversarial examples. The purpose is to make the system more robust. The authors proposed three methods to generate more data with diverse characteristics: (1) *knowledge-based*, which replaces words with their hypernym/hyponym provided in several given knowledge bases; (2) *hand-crafted*, which adds

[25] http://www.statmt.org/wmt16/translation-task.html.

Table 5. An Example of Adversarial Training Effectiveness [54]

Attacks	Original	Augmented
AddSent	34.8	70.4
AddSentMod	34.3	39.2

AddSent and AddSentMod are two attacking methods. After training on augmented data, the F1 scores (%) are improved against the attacks on test data.

negations to the existing entailment; (3) *neural-based*, which leverages a seq2seq model to generate an entailment examples by enforcing the loss function to measure the cross-entropy between the original hypothesis and the predicted hypothesis. During the training process, they adopted the idea from generative adversarial network to train a discriminator and a generator, and incorporated the adversarial examples in the discriminator's optimization step.

The work in Reference [12] explores another way for data augmentation. It takes the average character embedding as a word representation and incorporates it into the input. This approach is intrinsically insensitive to character scrambling such as *swap*, *mid*, and *Rand*, thus can resist to noises caused by these scrambling attacks proposed in the work. However, this defense is ineffective to other attacks that do not perturb on characters' orders.

Table 5 gives an example of the effectiveness of adversarial training in Reference [54]. The authors pointed out that the adversarial examples need to be carefully designed when training on adversarial examples to improve the model. This can be shown from the results of two attacking methods in the table.

5.1.2 Model Regularization. Model regularization enforces the generated adversarial examples as the regularizer and follows the form of

$$\min(J(f(x), y) + \lambda J(f(x'), y)), \tag{32}$$

where λ is a hyperparameter.

Following Reference [40], the work in Reference [93] constructs the adversarial training with a linear approximation as follows

$$\begin{gathered} -\log p(y|x + -\epsilon g/||g||_2, ; \theta), \\ g = \partial_x \log p(y|x; \hat{\theta}), \end{gathered} \tag{33}$$

where $||g||_2$ is the L_2 norm regularization, θ is the parameter of the neural model, and $\hat{\theta}$ is a constant copy of θ. The difference to Reference [40] is that the authors performed the adversarial generation and training in terms of the word embedding. Further, they extended their previous work on attacking image deep neural model [94], where the local distribution smoothness (LDS) is defined as the negative of the KL divergence of two distributions (original data and the adversaries). LDS measures the robustness of the model against the perturbation in local and "virtual" adversarial direction. In this sense, the adversary is calculated as the direction to which the model distribution is the most sensitive in terms of KL divergence. They also applied this attack strategy on word embeddings and performed adversarial training by adding adversarial examples as regularizer.

The work in Reference [118] follows the idea from Reference [93] and extends the adversarial training on LSTM. The authors followed FGSM to incorporate the adversarial training as a regularizer. But to enable the interpretability of adversarial examples, i.e., the word embeddings of the adversaries should be valid word embeddings in the vocabulary, they introduced a direction

vector that associates the perturbed embedding to the valid word embedding. Reference [145] simply adopted the regularizer utilized in Reference [93], but applied the perturbations on pre-trained word embeddings and in a different task: relation extraction. Other similar works that adopt Reference [93] are References [10, 118, 145, 151]. We will not cover all these works in this article, since they simply adopting the same technique.

5.1.3 Robust Optimization. Madry et al. [84] casted DNN model learning as a robust optimization with min-max (saddle point) formulation, which is the composition of an inner non-concave maximization problem (attack) and an outer non-convex minimization problem (defense). According to Danskin's theorem, gradients at inner maximizers correspond to descent directions for the min-max problem, thus the optimization can still apply back-propagation to proceed. The approach successfully demonstrated robustness of DNNs against adversarial images by training and learning universally. Reference [3] adopted the idea and applied on malware detection DNN that handles discrete data. Its learning objective is formulated as

$$\theta^* = arg \min_{\theta} \mathbb{E}_{(x,y)\sim D}\Big[\max_{x' \in S(x)} L(\theta, x', y)\Big], \tag{34}$$

where $S(x)$ is the set of binary indicator vectors that preserve the functionality of malware x, L is the loss function for the original classification model, y is the groundtruth label, θ is the learnable parameters, and D denotes the distribution of data sample x.

It is worth noting that the proposed robust optimization method is a universal framework under which other adversarial training strategies have natural interpretation. We describe it separately, keeping in view its popularity in the literature.

5.2 Distillation

Papernot et al. [106] proposed distillation as another possible defense against adversarial examples. The principle is to use the softmax output (e.g., the class probabilities in classfication DNNs) of the original DNN to train the second DNN, which has the same structure with the original one. The softmax output of the original DNN is also modified by introducing a *temperature* parameter T:

$$q_i = \frac{\exp(z_i/T)}{\sum_k \exp(z_k/T)}, \tag{35}$$

where z_i is input of softmax layer. T controls the level of knowledge distillation. When $T = 1$, Equation (35) turns back to the normal softmax function. If T is large, then q_i is close to a uniform distribution; when T is small, the function will output more extreme values. Reference [44] adopted distillation defense for DNNs on discrete data and applied a high temperature T, as high-temperature softmax is proved to reduce the model sensitivity to small perturbations [106]. The authors trained the second DNN with the augmentation of original dataset and the softmax outputs from the original DNN. From the evaluations, they found that adversarial training is more effective than using distillation.

6 DISCUSSIONS AND OPEN ISSUES

Generating textual adversarial examples has a relatively shorter history than generating image adversarial examples on DNNs, because it is more challenging to make perturbation on discrete data, and meanwhile preserving the valid syntactic, grammar and semantics. We discuss some of the issues in this section and provide suggestions on future directions.

6.1 Perceivability

Perturbations in image pixels are usually hard to be perceived, thus do not affect human judgment, but can fool the deep neural networks. However, the perturbation on text is obvious, no matter the perturbation is flipping characters or changing words. Invalid words and syntactic errors can be easily identified by human and detected by the grammar check software, hence the perturbation is hard to attack a real NLP system. However, many research works generate such types of adversarial examples. It is acceptable only if the purpose is utilizing adversarial examples to robustify the attacked DNN models. In semantic-preserving perspective, changing a word in a sentence sometimes changes its semantics drastically and is easily detected by human beings. For NLP applications such as reading comprehension, and sentiment analysis, the adversarial examples need to be carefully designed in order not to change the *should-be* output. Otherwise, both correct output and perturbed output change, violating the purpose of generating adversarial examples. This is challenging and limited works reviewed considered this constraint. Therefore, for practical attack, we need to propose methods that make the perturbations not only unperceivable but also preserve correct grammar and semantics.

6.2 Transferability

Transferability is a common property for adversarial examples. It reflects the generalization of the attack methods. Transferability means adversarial examples generated for one deep neural network on a dataset can also effectively attack another deep neural network (i.e., cross-model generalization) or dataset (i.e., cross-data generalization). This property is more often exploited in black-box attacks as the details of the deep neural networks do not affect the attack method much. It is also shown that untargeted adversarial examples are much more transferable than targeted ones [81]. Transferability can be organized into three levels in deep neural networks: (i) same architecture with different data; (ii) different architectures with same application; and (iii) different architectures with different data [154]. Although current works on textual attacks cover both three levels, the performance of the transferred attacks still decrease drastically compared to it on the original architecture and data, i.e., poor generalization ability. More efforts are expected to deliver better generalization ability.

6.3 Automation

Some reviewed works are able to generate adversarial examples automatically, while others cannot. In white-box attacks, leveraging the loss function of the DNN can identify the most affected points (e.g., character, word) in a text automatically. Then the attacks are performed on these points by automatically modifying the corresponding texts. In black-box attacks, some attacks, e.g., *substitution* trains substitute DNNs and applies white-box attack strategies on the substitution. This can be achieved automatically. However, most of the other works craft the adversarial examples in a manual manner. For example, Reference [54] concatenates manually-chosen meaningless paragraphs to fool the reading comprehension systems, to discover the vulnerability of the victim DNNs. Many research works follow Reference [54], not aiming on practical attacks, but more on examining robustness of the target network. These manual works are time-consuming and impractical. We believe that more efforts in this line could pass through this barrier in future.

6.4 New Architectures

Although most of the common textual DNNs have gained attention from the perspective of adversarial attack (Section 2.2), many DNNs have not been attacked so far, e.g., the generative neural models: Generative Adversarial Networks (GANs) and Variational Auto-Encoders (VAEs). In NLP,

they are used to generate texts. Deep generative models require more sophisticated skill for model training. This would explain that these techniques have been mainly overlooked by adversarial attack so far. Future works may consider generating adversarial examples for these generative DNNs. Another example is differentiable neural computer (DNC). Only one work attacked DNC so far [19]. Attention mechanism somehow becomes a standard component in most of the sequential models. But there has been no work examining the mechanism itself. Instead, works are either attacking the overall system that contain attentions, or leveraging attention scores to identify the word for perturbation [14].

6.5 Iterative versus One-off

Iterative attacks iteratively search and update the perturbations based on the gradient of the output of the attacked DNN model. Thus, it shows high quality and effectiveness. That is, the perturbations can be small enough and hard to defense. However, these methods usually require long time to find the proper perturbations, rendering an obstacle for attacking in real-time. Therefore, one-off attacks are proposed to tackle this problem. FGSM [40] is one example of one-off attack. Intuitively, one-off attack is much faster than iterative attack, but is less effective and easier to be defensed [153]. When designing attack methods on a real application, attackers need to carefully consider the trade off between efficiency and effectiveness of the attack.

7 CONCLUSION

This article presents the first comprehensive survey in the direction of generating textual adversarial examples on deep neural networks. We review recent research efforts and develop classification schemes to organize the existing literature. Additionally, we summarize and analyze them from different aspects. We attempt to provide a good reference for researchers to gain insight of the challenges, methods, and issues in this research topic and shed light on future directions. We hope more robust deep neural models are proposed based on the knowledge of the adversarial attacks.

REFERENCES

[1] Nawaf A. Abdulla, Nizar A. Ahmed, Mohammed A. Shehab, and Mahmoud Al-Ayyoub. 2013. Arabic sentiment analysis: Lexicon-based and corpus-based. In *Proceedings of the IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT'13)*. IEEE, 1–6.

[2] Naveed Akhtar and Ajmal S. Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6 (2018), 14410–14430.

[3] Abdullah Al-Dujaili, Alex Huang, Erik Hemberg, and Una-May O'Reilly. 2018. Adversarial deep learning for robust detection of binary encoded malware. In *Proceedings of the IEEE Security and Privacy Workshops (SPW'18)*. 76–82.

[4] Abdullah Al-Dujaili, Alex Huang, Erik Hemberg, and Una-May O'Reilly. 2018. Adversarial deep learning for robust detection of binary encoded malware. In *Proceedings of the IEEE Security and Privacy Workshops (SP Workshops'18)*. 76–82.

[5] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'18)*. 2890–2896.

[6] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV'15)*. 2425–2433.

[7] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, and Konrad Rieck. 2014. DREBIN: Effective and explainable detection of Android malware in your pocket. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'14)*.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations (ICLR'15)*.

[9] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. 2010. The security of machine learning. *Machine Learning* 81, 2 (2010), 121–148.

[10] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Adversarial training for multi-context joint entity and relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'18)*. 2830–2836.

[11] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. An attentive neural architecture for joint segmentation and parsing and its application to real estate ads. *Expert Syst. Appl.* 102 (2018), 100–112.

[12] Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

[13] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recogn.* 84 (2018), 317–331.

[14] Matthias Blohm, Glorianna Jagfeld, Ekta Sood, Xiang Yu, and Ngoc Thang Vu. 2018. Comparing attention-based convolutional and recurrent neural networks: Success and limitations in machine reading comprehension. In *Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL'18)*. 108–118.

[15] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*. 632–642.

[16] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL'16)*. 10–21.

[17] Nicholas Carlini and David A. Wagner. 2017. Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'17)*. 39–57.

[18] Nicholas Carlini and David A. Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *Proceedings of IEEE Security and Privacy Workshops (SPW'18)*. 1–7.

[19] Alvin Chan, Lei Ma, Felix Juefei-Xu, Xiaofei Xie, Yang Liu, and Yew Soon Ong. 2018. Metamorphic relation based adversarial attacks on differentiable neural computer. *CoRR* abs/1809.02444 (2018).

[20] Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. 2018. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL'18)*. 2587–2597.

[21] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*. 1657–1668.

[22] Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*.

[23] Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL'19)*. 1085–1097.

[24] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1724–1734.

[25] Marta R. Costa-Jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*.

[26] George E. Dahl, Jack W. Stokes, Li Deng, and Dong Yu. 2013. Large-scale malware classification using random projections and neural networks. In *Proceedings of the 38th International Conference on Acoustics, Speech and Signal Processing (ICASSP'13)*. 3422–3426.

[27] Li Deng and Yang Liu. 2018. *Deep Learning in Natural Language Processing*. Springer.

[28] George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program—Tasks, data, and evaluation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*.

[29] Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On adversarial examples for character-level neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING'18)*. 653–663.

[30] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL'18)*. 31–36.

[31] Gamaleldin F. Elsayed, Ian J. Goodfellow, and Jascha Sohl-Dickstein. 2019. Adversarial reprogramming of neural networks. In *Proceedings of the 7th International Conference on Learning Representations (ICLR'19). Poster*.

[32] Fartash Faghri, David J. Fleet, Ryan Kiros, and Sanja Fidler. 2017. VSE++: Improved visual-semantic embeddings. *CoRR* abs/1707.05612.

[33] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*. 457–468.

[34] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep-learning classifiers. In *Proceedings of the IEEE Security and Privacy Workshops (SP'18). Workshop*. 50–56.
[35] Justin Gilmer, Ryan P. Adams, Ian J. Goodfellow, David Andersen, and George E. Dahl. 2018. Motivating the rules of the game for adversarial example research. *CoRR* abs/1807.06732.
[36] Yoav Goldberg. 2017. *Neural Network Methods for Natural Language Processing*. Morgan & Claypool Publishers. DOI: https://doi.org/10.2200/S00762ED1V01Y201703HLT037
[37] Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. *Neural Netw.* 1 (1996), 347–352.
[38] Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku. 2018. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175*.
[39] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'14)*. 2672–2680.
[40] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.
[41] Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL'17)*. 427–431.
[42] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'13)*. 6645–6649.
[43] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2016. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435*.
[44] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick D. McDaniel. 2017. Adversarial examples for malware detection. In *Proceedings of the 22nd European Symposium on Research in Computer Security (ESORICS'17)*. 62–79.
[45] Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *J. Biomed. Info.* 45, 5 (2012), 885–892.
[46] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. 2014. Deep speech: Scaling up end-to-end speech recognition. *CoRR* abs/1412.5567.
[47] He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*. 1766–1776.
[48] Jeff Heaton. 2018. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning—The MIT Press, 2016, 800 pp., ISBN 0262035618. *Genetic Program. Evol. Mach.* 19, 1–2 (2018), 305–307.
[49] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR* abs/1503.02531. Retrieved from http://arxiv.org/abs/1503.02531.
[50] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (1997), 1735–1780.
[51] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of IEEE International Conference on Computer Vision (ICCV'17)*. 804–813.
[52] Weiwei Hu and Ying Tan. 2018. Black-box attacks against RNN based malware detection algorithms. In *Proceedings of the Workshops of the the 32nd AAAI Conference on Artificial Intelligence (AAAI workshops'18)*. 245–251.
[53] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'18)*. 1875–1885.
[54] Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'17)*. 2021–2031.
[55] Rie Johnson and Tong Zhang. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Proceedings of the Annual Conference on Neural Information Processing Systems 2015 (NIPS'15)*. 919–927.
[56] Dongyeop Kang, Tushar Khot, Ashish Sabharwal, and Eduard H. Hovy. 2018. AdvEntuRe: Adversarial training for textual entailment with knowledge-guided examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL'18)*. 2418–2428.
[57] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernández Mota, Jon Almazán, and Lluís-Pere de las Heras. 2013. ICDAR 2013 robust reading

competition. In *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR'13)*. 1484–1493.

[58] Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTaiL: A textual entailment dataset from science question answering. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*. 5189–5197.

[59] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1746–1751.

[60] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence (AAAI'16)*. 2741–2749.

[61] Diederik P. Kingma and Jimmy Ba. 2015. DAM: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.

[62] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR'14)*.

[63] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*. 67–72.

[64] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Int. J. Comput. Vision* 123, 1 (2017), 32–73.

[65] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS'12)*. 1106–1114.

[66] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33nd International Conference on Machine Learning (ICML'16)*. 1378–1387.

[67] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial machine learning at scale. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*.

[68] Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*. 957–966.

[69] Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning (ICML'14)*. 1188–1196.

[70] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Trans. Assoc. Comput. Linguist.* 5 (2017), 365–378.

[71] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.

[72] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.* 5 (2004), 361–397.

[73] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. TextBugger: Generating adversarial text against real-world applications. In *Proceedings of 26th Annual Network and Distributed System Security Symposium (NDSS'19)*.

[74] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*. 1192–1202.

[75] Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*.

[76] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 4208–4215.

[77] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *Proceedings of the 13th European Conference on Computer Vision (ECCV'14)*. 740–755.

[78] Angli Liu, Stephen Soderland, Jonathan Bragg, Christopher H. Lin, Xiao Ling, and Daniel S. Weld. 2016. Effective crowd annotation for relation extraction. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'16)*. 897–906.

[79] Qiang Liu, Pan Li, Wentao Zhao, Wei Cai, Shui Yu, and Victor C. M. Leung. 2018. A survey on security threats and defensive techniques of machine learning: A data driven view. *IEEE Access* 6 (2018), 12103–12117.

[80] Tzu-Chien Liu, Yu-Hsueh Wu, and Hung-yi Lee. 2017. Attention-based CNN matching net. *CoRR* abs/1709.05036.

[81] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into transferable adversarial examples and black-box attacks. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*.

[82] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL'15)*. 285–294.

[83] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*. 142–150.

[84] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep-learning models resistant to adversarial attacks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*.

[85] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn treebank. *Comput. Linguist.* 19, 2 (1993), 313–330.

[86] Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 Task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval@COLING'14)*. 1–8.

[87] Cettolo Mauro, Girardi Christian, and Federico Marcello. 2012. Wit3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Annual Conference of the European Association for Machine Translation (EAMT'12)*. 261–268.

[88] Julian J. McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys'13)*. 165–172.

[89] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. 2006. Spam filtering with Naive Bayes—Which Naive Bayes? In *Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS'06)*.

[90] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS'13)*. 3111–3119.

[91] Pasquale Minervini and Sebastian Riedel. 2018. Adversarially regularising neural NLI Models to integrate logical background knowledge. In *Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL'18)*. 65–74.

[92] Anand Mishra, Karteek Alahari, and C. V. Jawahar. 2012. Scene text recognition using higher order language priors. In *Proceedings of the 23rd British Machine Vision Conference (BMVC'12)*. 1–11.

[93] Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*.

[94] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2016. Distributional smoothing with virtual adversarial training. In *Proceedings of the 4th International Conference on Learning Representations (ICLR'16)*.

[95] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 2574–2582.

[96] Michael C. Mozer. 1995. A focused backpropagation algorithm for temporal. *Backprop.: Theory Architect. Appl.* 137 (1995).

[97] Paarth Neekhara, Shehzeen Hussain, Shlomo Dubnov, and Farinaz Koushanfar. 2018. Adversarial reprogramming of sequence classification neural networks. *CoRR* abs/1809.01829.

[98] Tong Niu and Mohit Bansal. 2018. Adversarial over-sensitivity and over-stability strategies for dialogue models. In *Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL'18)*. 486–496.

[99] Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco et al. 2015. Universal dependencies 1.2. https://universaldependencies.org/.

[100] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. 2018. A survey of the usages of deep learning in natural language processing. *CoRR* abs/1807.10854.

[101] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab K. Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio Speech Lang. Process.* 24, 4 (2016), 694–707.

[102] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. 115–124.

[103] Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. In *Proceedings of the Military Communications Conference (MILCOM'16)*. IEEE, 49–54.

[104] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (AsiaCCS'17)*. 506–519.

[105] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P'16)*. 372–387.

[106] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'16)*. 582–597.

[107] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*. 2249–2255.

[108] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'18)*. 2227–2237.

[109] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles K. Nicholas. 2018. Malware detection by eating a whole EXE. In *Proceedings of the Workshops of the 32nd AAAI Conference on Artificial Intelligence*. 268–276.

[110] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL'19)*. 1085–1097.

[111] Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'10)*. 148–163.

[112] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the 2016 International Conference on Learning Representations (ICLR'16)*.

[113] Royi Ronen, Marian Radu, Corina Feuerstein, Elad Yom-Tov, and Mansour Ahmadi. 2018. Microsoft malware classification challenge. *CoRR* abs/1802.10135. Retrieved from http://arxiv.org/abs/1802.10135.

[114] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. 2018. Generic black-box end-to-end attack against state of the art API call based malware classifiers. In *Proceedings of the 21st International Symposium of Research on Attacks, Intrusions, and Defenses (RAID'18)*. 490–510.

[115] Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL'04)*. 1–8.

[116] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533.

[117] Suranjana Samanta and Sameep Mehta. 2018. Generating adversarial text samples. In *Proceedings of the 40th European Conference on IR Research (ECIR'18)*. 744–749.

[118] Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. 2018. Interpretable adversarial perturbation in input embedding space for text. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. 4323–4330.

[119] Dale Schuurmans and Martin Zinkevich. 2016. Deep-learning games. In *Proceedings of the Annual Conference on Neural Information Processing Systems 2016 (NIPS'16)*. 1678–1686.

[120] Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: A toolkit for neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL'17)*. 65–68.

[121] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17). Poster*.

[122] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*. 3776–3784.

[123] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*. 3295–3301.

[124] Baoguang Shi, Xiang Bai, and Cong Yao. 2017. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 11 (2017), 2298–2304.

[125] Haoyue Shi, Jiayuan Mao, Tete Xiao, Yuning Jiang, and Jian Sun. 2018. Learning visually-grounded semantics from contrastive adversarial samples. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING'18)*. 3715–3727.

[126] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.

[127] Sameer Singh, Carlos Guestrin, and Marco Túlio Ribeiro. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL'18)*. 856–865.

[128] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*. 1631–1642.

[129] Congzheng Song and Vitaly Shmatikov. 2018. Fooling OCR systems with adversarial text images. *CoRR* abs/1802.05385.

[130] Mengying Sun, Fengyi Tang, Jinfeng Yi, Fei Wang, and Jiayu Zhou. 2018. Identify susceptible locations in medical records via adversarial attacks on deep predictive models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18)*. 793–801.

[131] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'14)*. 2672–2680.

[132] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, and Joan Bruna. 2014. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR'14)*.

[133] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL'15)*. 1556–1566.

[134] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. MovieQA: Understanding stories in movies through question-answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 4631–4640.

[135] Tesseract. 2016. Retrieved from https://github.com/tesseract-ocr/.

[136] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'17)*. 6000–6010.

[137] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*. 3156–3164.

[138] Canhui Wang, Min Zhang, Shaoping Ma, and Liyun Ru. 2008. Automatic online news issue construction in web environment. In *Proceedings of the 17th International Conference on World Wide Web (WWW'08)*. 457–466.

[139] Kai Wang, Boris Babenko, and Serge J. Belongie. 2011. End-to-end scene text recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'11)*. 1457–1464.

[140] Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*.

[141] Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-LSTM and answer pointer. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*.

[142] Yicheng Wang and Mohit Bansal. 2018. Robust machine comprehension models via adversarial training. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'18)*. 575–581.

[143] David Warde-Farley and Ian Goodfellow. 2016. Adversarial perturbations of deep neural networks. *Perturb. Optimiz. Statist.* 311 (2016).

[144] Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'18)*. 1112–1122.

[145] Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1778–1783.

[146] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and

Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144 Retrieved from http://arxiv.org/abs/1609.08144.

[147] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*. 2048–2057.

[148] Xiaojun Xu, Xinyun Chen, Chang Liu, Anna Rohrbach, Trevor Darrell, and Dawn Song. 2018. Fooling vision and language models despite localization and attention mechanism. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 4951–4961.

[149] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.

[150] Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*. 180–189.

[151] Michihiro Yasunaga, Jungo Kasai, and Dragomir R. Radev. 2018. Robust multilingual part-of-speech tagging via adversarial training. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'18)*. 976–986.

[152] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep-learning-based natural language processing. *IEEE Comput. Intell. Mag.* 13, 3 (2018), 55–75.

[153] Xiaoyong Yuan, Pan He, and Xiaolin Andy Li. 2018. Adaptive adversarial attack on scene text recognition. *CoRR* abs/1807.03326. Retrieved from http://arxiv.org/abs/1807.03326.

[154] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* 30, 9 (2019), 2805–2824.

[155] Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL'19)*. 1085–1097.

[156] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings in Annual Conference on Neural Information Processing Systems (NIPS'15)*. 649–657.

[157] Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*.

[158] Yuke Zhu, Oliver Groth, Michael S. Bernstein, and Li Fei-Fei. 2016. Visual7W: Grounded question answering in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 4995–5004.

Received May 2019; revised October 2019; accepted November 2019