ROBUSTNESS OF NEURAL NETWORKS FOR DISCRETE INPUT: AN ADVERSARIAL PERSPECTIVE

by

JAVID EBRAHIMI

A DISSERTATION

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

December 2018

DISSERTATION APPROVAL PAGE

Student: Javid Ebrahimi

Title: Robustness of Neural Networks for Discrete Input: An Adversarial Perspective

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer and Information Science by:

Daniel Lowd	Co-Chair
Dejing Dou	Co-Chair
Reza Rejaie	Core Member
Yashar Ahmadian	Institutional Representative

and

Janet Woodruff-Borden	Vice Provost and Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded December 2018

DISSERTATION ABSTRACT

Javid Ebrahimi

Doctor of Philosophy

Department of Computer and Information Science

December 2018

Title: Robustness of Neural Networks for Discrete Input: An Adversarial Perspective

In the past few years, evaluating on adversarial examples has become a standard procedure to measure robustness of deep learning models. Literature on adversarial examples for neural nets has largely focused on image data, which are represented as points in continuous space. However, a vast proportion of machine learning models operate on discrete input, and thus demand a similar rigor in understanding their vulnerabilities and robustness. We study robustness of neural network architectures for textual and graph inputs, through the lens of adversarial input perturbations. We will cover methods for both attacks and defense; we will focus on 1) addressing challenges in optimization for creating adversarial perturbations for discrete data; 2) evaluating and contrasting white-box and black-box adversarial examples; and 3) proposing efficient methods to make the models robust against adversarial attacks.

This dissertation includes previously published co-authored material.

CURRICULUM VITAE

NAME OF AUTHOR: Javid Ebrahimi

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR, USA
Tarbiat Modares University, Tehran, Iran
University of Tehran, Tehran, Iran

DEGREES AWARDED:

Doctor of Philosophy, Computer Science, 2018, University of Oregon
Master of Science, Computer Science, 2018, University of Oregon
Master of Science, Software Engineering, 2012, Tarbiat Modares University
Bachelor of Science, Software Engineering, 2008, University of Tehran

AREAS OF SPECIAL INTEREST:

Deep Learning
Adversarial Machine Learning
Natural Language Processing
Relational Learning

PROFESSIONAL EXPERIENCE:

Research Assistant, Fall 2014-Fall 2018
Teaching Assistant, Fall 2013-Spring 2014

GRANTS, AWARDS AND HONORS:

SIGIR Travel Grant, 2016
Best Graduate Teaching Fellow, Department of Computer and Information Science, 2013

PUBLICATIONS:

Ebrahimi J., Lowd, D. & Dou, D. (2018). On Adversarial Examples for Character-Level Neural Machine Translation. *In Proceedings of 27th International Conference on Computational Linguistics.*

Ebrahimi J., Rao, A., Lowd D. & Dou, D. (2018). HotFlip: White-Box Adversarial Examples for Text Classification. (short) *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics.*

Ebrahimi J., Dou, D.& Lowd D. (2016). Weakly Supervised Tweet Stance Classification by Relational Bootstrapping. (short) *In Proceedings of Conference on Empirical Methods in Natural Language Processing.*

Ebrahimi J., Dou, D.& Lowd D. (2016). A Joint Sentiment-Target-Stance Model for Stance Classification in Tweets. *In Proceedings of 26th International Conference on Computational Linguistics.*

Nhathai P., **Ebrahimi J.**, Kil D., Piniewski B. & Dou D. (2016) Topic-Aware Physical Activity Propagation in a Health Social Network *In IEEE Intelligent Systems.*

Ebrahimi J., Nhathai P., Kil D., Piniewski B. & Dou D. (2016) Characterizing Physical Activity in a Health Social Network. *In Proceedings of the 6th International Conference on Digital Health*

Ebrahimi, J. & Dou, D. (2015) Chain-Based RNN for Relation Classification. (short) *In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*

ACKNOWLEDGEMENTS

I would like to express my gratitude and appreciation to my advisors, Dejing Dou and Daniel Lowd, for their support and supervision throughout the development of this thesis. I would like to thank other members on my committee, Reza Rejaie and Yashar Ahmadian, for their perspective and our discussions.

I am forever indebted to my parents for everything they have done for me, and I am grateful to my siblings for their unconditional love and never-ending support.

Many thanks to my collaborators Hai Phan and Anyi Rao; my friends and lab members Sabin Kafle, Nisansa De Silva, and Amnay Amimeur; Hank Childs for his generosity with his lab's computing resources; and my friend and editor, Ellen Klowden.

This work was funded by ARO grant W911NF-15-1-0265.

In memory of my grandfather

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

Affordable technologies an ever-increasing number of apps have made machine learning algorithms an indispensable part of our life. While since the early days of AI there have been logical and philosophical discussions of dangers of machine intelligence, the dangers of machine stupidity are often disregarded. While auto-correct errors are innocuous, a mistake by an autonomous car can be deadly. While a spam in our inbox is annoying, a fraud detection failure could cost a business millions of dollars. Robustness of machine learning algorithms has been a subject of interest for more than a decade. It is common to devise models, which are robust to worst-case perturbation within some defined bounds. Examples which are created using such perturbations are called *adversarial examples*.

Adversarial examples are powerful tools to investigate the vulnerabilities of a machine learning model (Szegedy et al., 2014). These examples are inputs to a predictive machine learning model that are maliciously designed to cause poor performance (I. J. Goodfellow, Shlens, & Szegedy, 2015). Robust machine learning has a long history in machine learning, going back to adversarial attacks on linear spam classifiers (Dalvi, Domingos, Sanghai, Verma, et al., 2004; Lowd & Meek, 2005). This line of research has recently received a lot of attention in the deep learning community, partly because adversarial examples undermine our confidence in the promise of deep learning. Specifically, adversarial examples expose weaknesses in models' *generalizability*: the ability of the model to generalize to unseen data. Let us proceed with an example: Last year, a mistranslation by Facebook's machine translator (MT) lead to someone's wrongful arrest (Berger, 2017). Instead of translating an Arabic phrase to "Good Morning", Facebook's MT translated it to

"Murder Them". Arabic is a morphologically-rich language, and the MT mistook the input word with another which differs from the input by only one character. As machine translation is used more and more, it is increasingly important to understand its worst-case failures to prevent incidents like this. Furthermore, by studying the robustness of MT models, we can create models which are robust to more benign but still noisy data (e.g., typos).

The security threats of adversarial examples for textual and graph-based data cannot be underestimated; for instance, spammers have been modifying their emails to evade spam filters since the late '90s, and they engage in link farming to get higher ranks in social media or search engines (Ghosh et al., 2012). Nevertheless, literature on adversarial examples for neural nets has largely focused on image data which are represented as points in continuous space.

We will focus on the much-needed robustness of neural networks on discrete data, in particular text and graphs. In this work, we first develop methods to attack neural models that operate on discrete input, and then devise methods to defend against those attacks. We study text classification and machine translation in NLP, and transductive and inductive node classification in graphs.

Background

It was found in (Szegedy et al., 2014) that state-of-the-art image recognition systems can easily get tricked by adding small human-imperceptible noise to the image. This can have severe consequences not only from a business perspective, but also for safety threats it can pose if used in applications such as automatic car driving. Adversarial examples, which are created by maliciously manipulating an instance, are designed to attack a machine learning model at test time. The majority of the previous work on adversarial examples focuses on image classification (I. J. Goodfellow

et al., 2015; Nguyen, Yosinski, & Clune, 2015; Papernot, McDaniel, Jha, et al., 2016; Szegedy et al., 2014), and more recently on malware classification (Grosse, Papernot, Manoharan, Backes, & McDaniel, 2017), NLP (Belinkov & Bisk, 2018; Jia & Liang, 2017; Zhao, Dua, & Singh, 2018), and image segmentation (Xie et al., 2017).

Adversarial Attacks. Adversarial examples are created in *black-box* and *white-box* settings. In the former, the adversary does not have access to model parameters, and can only query the system, which can be used to create a substitute model for the attacked model. In the latter, the adversary has access to model parameters, and in the case of differentiable models can use the gradients to attack the model. Due to stronger assumptions in the white-box setting, the error incurred by such adversaries can be regarded as an upper bound for black-box ones. Due to difficulty of discrete optimization, most works on adversarial NLP have focused on black-box attacks. Our work, focuses on creating white-box adversarial examples for NLP, which are adaptable to other discrete inputs, such as graphs.

While there are some recent developments in creating unrestricted adversarial examples (Song, Shu, Kushman, & Ermon, 2018), most of the literature focuses on creating adversarial examples within a norm-bounded ball around inputs, and making the models robust against such attacks. Creating adversarial examples requires solving a non-convex optimization problem. Concretely, given the classifier g and the input x in the constrained set C, the goal is to find the optimal perturbation $\hat{\delta}$, such that the classifier makes a different prediction than on the original input.

$$\begin{aligned} &\underset{\delta}{\text{minimize}} \ \|\delta\|_2 \\ &\text{subject to} \qquad g(x+\delta) \neq g(x) \\ &\qquad\qquad\qquad\quad x+\delta \in C \end{aligned} \tag{1.1}$$

This problem was first tackled by a nonlinear optimization method (Szegedy et al., 2014). Other methods include one-shot gradient descent (I. J. Goodfellow et al., 2015), and iterative projected gradient descent (Madry, Makelov, Schmidt, Tsipras, & Vladu, 2018) methods, which use the linear approximation of the loss function (i.e., using first-order Taylor expansion). Our work also follows this approach of linear approximation of the loss function.

Adversarial examples could be *untargeted* or *targeted*; in the former, the adversary simply aims to inflict loss, and force the model to produce an output different from the correct output, while in the latter, the adversary wants to force the model to produce a desired output. Most of the literature on image classification report results on both types of attacks. On the other hand, none of the NLP works have put forth a quantitative analysis of targeted and untargeted adversarial examples. We investigate two new attack scenarios for machine translation, and propose a new framework for creating and evaluating targeted adversarial examples for machine translation.

Adversarial Defense. One method for defense is *adversarial training*, which interleaves training with generation of adversarial examples (I. J. Goodfellow et al., 2015). Concretely, after every iteration of training, adversarial examples are created and added to the mini-batches. Virtual adversarial training (Miyato, Maeda, Koyama, Nakae, & Ishii, 2016) is another regularization method, which aims to minimize the KL divergence between predictions on the examples and their adversarial counterparts, for semi-supervised tasks, wherein we have a limited number of labeled examples, and many unlabeled examples.

A robust model aims to fit model parameters given such adversarial examples. Concretely, the optimization problem we will need to solve is the following saddle-

point problem.

$$\min_{\theta} \max_{\delta} J_\theta(x+\delta, y) \quad \text{s.t. } \|\delta\|_p \leq \epsilon \tag{1.2}$$

where J is the loss function, x is the input, y is the label, δ is the perturbation vector, p is the type of norm-bounded perturbation, ϵ is the magnitude of noise, and θ is the model parameters. A projected gradient-based approach for adversarial training by Madry et al. (2018) has proved to be one of the most effective defense mechanisms against adversarial attacks for image classification. The difference between this approach and the one proposed by I. J. Goodfellow et al. (2015) is the lack of clean examples in training and its closer connection with robust optimization for a non-convex problem.

Some other methods to defend against adversarial examples use regularization techniques, such as distillation (Papernot, McDaniel, Wu, Jha, & Swami, 2016) and saturating networks (Nayebi & Ganguli, 2017), which often give a misleading sense of security. Concretely, these regularization techniques will often lead to gradient obfuscation, which would produce unstable gradients for a white-box attacker (Athalye, Carlini, & Wagner, 2018). Various tricks can avoid this instability and render these ostensibly secured models vulnerable (Carlini & Wagner, 2017b).

Another approach to defend against adversarial examples is to detect them at test time, by augmenting the data or model to have a built-in capability to detect adversarial examples (Grosse, Manoharan, Papernot, Backes, & McDaniel, 2017; Metzen, Genewein, Fischer, & Bischoff, 2017).

Recent research in defending adversarial examples focuses on convex relaxation of neural networks (Kolter & Wong, 2017; Raghunathan, Steinhardt, & Liang, 2018), and providing provable guarantees for robustness of neural models. The state-of-the-art provable defense (Kolter & Wong, 2017) uses relaxations of ReLU activation

functions (Nair & Hinton, 2010), which was first used for neural net verification (Ehlers, 2017). They pose the maximization problem in Eq. 1.2 as a linear program. Interestingly, they show that the dual of the linear program can also be represented by a neural network.

We mainly use adversarial training for defense, but we will propose methods for robustness of graph neural nets, which detect adversarial edges, or perform robust optimization through a convex neural net.

Adversarial Examples for Discrete Input

While a gradient ascent on the input surface could approximate the increase in loss for continuous data, such approximation is not straightforward for discrete input. Concretely, a first order approximation of an adversarial point in continuous space is the following.

$$J_\theta(x+\delta) = J_\theta(x) + < \delta, \nabla_x J_\theta(x) > \tag{1.3}$$

Where $< . >$ is the dot-product operator. An optimal unconstrained perturbation vector for continuous data, would be

$$\delta = \epsilon \nabla_x J_\theta(x) \tag{1.4}$$

where ϵ is tuning parameter, similar to learning rate used in parameter estimation in machine learning. However, such a δ cannot be used for discrete input, as $x + \delta$ is, almost for sure, an invalid point in discrete space. Thus, we have to define methods which use gradients to create valid adversarial examples to increase the loss. In the first part of our research, we will focus on creating adversarial examples for discrete input. For both graph and text, we manipulate discrete input to attack a model, in a white-box setting, using differentiable discrete operations.

Adversarial Examples for NLP

The need to understand vulnerabilities of NLP systems is only growing. Companies such as Google are using text classifiers to detect abusive language[1], and concerns are increasing over deception (Zubiaga, Liakata, Procter, Hoi, & Tolmie, 2016) and safety (Chancellor, Pater, Clear, Gilbert, & De Choudhury, 2016) in social media. In all of these cases, we need to to better understand the dynamics of how NLP models make mistakes on unusual inputs, in order to improve accuracy, increase robustness, and maintain security or privacy. So far, few existing works on adversarial examples for NLP have avoided discrete-level adversarial examples, due to difficulty in optimization (Miyato, Dai, & Goodfellow, 2017). The first work of crafting adversarial examples for NLP was proposed in (Papernot, McDaniel, Swami, & Harang, 2016). They added noise to the word embeddings and searched the neighborhood of that new embedding in order to replace the original word. While their adversary is able to trick the classifier, the word-level changes tend to produce nonsensical sentences (e.g., "I wouldn't rent this..." $\rightarrow$ "Excellent wouldn't rent this..."). Jia and Liang (2017) add distracting sentences to the end of paragraphs to fool deep reading comprehension systems, which are polished by crowdsourcing. They show how state-of-the-art text comprehension systems are matching patterns rather than understanding text. Hosseini, Kannan, Zhang, and Poovendran (2017) showed that simple modifications, such as adding spaces or dots between characters, can drastically change the toxicity score of the API.

Adding noise to the continuous word embeddings (Miyato et al., 2017; Papernot, McDaniel, Swami, & Harang, 2016), random character changes (Belinkov & Bisk, 2018), rule-based semantic preserving adversaries (Iyyer, Wieting, Gimpel, &

[1]https://www.perspectiveapi.com

Zettlemoyer, 2018; Ribeiro, Singh, & Guestrin, 2018), and task-based heuristics (Hosseini et al., 2017; Jia & Liang, 2017) comprise the set of approaches to create adversarial examples for NLP. None of the existing works use gradients of the model to create adversarial examples. We will use gradient-based optimization to create white-box adversarial examples to investigate worst-case failures of deep NLP models.

Black-Box vs. White-Box Methods for NLP

Black-box attacks (Belinkov & Bisk, 2018; Jia & Liang, 2017; Ribeiro et al., 2018) often rely on task-related knowledge to create adversarial examples. In contrast, white-box attacks approximate the worst-case attack on text, within some allowed set of perturbations, in a general setting, regardless of the task at hand. In addition, white-box attacks can demonstrate and defend against a model's most serious vulnerabilities, which might not be discovered by black-box methods. We will demonstrate that a white-box-trained model is stronger than a black-box-trained model in defending against a variety of types of noise.

We focus on the task of machine translation, and we propose *controlled* and *targeted* adversaries which create adversarial examples with other goals, instead of merely decreasing the BLEU score (Papineni, Roukos, Ward, & Zhu, 2002). A controlled adversary aims to mute a word in the original translation, while a targeted adversary aims to push a word into it. We will penalize an adversary for ad-hoc manipulations, which could cause the MT to generate a radically different and possibly gibberish translation. Using these new scenarios, we will contrast black-box and white-box attacks, and demonstrate that a white-box adversary can achieve more difficult goals.

Adversarial Examples for Graphs

Security of relational models has been an important subject of study for many years. Most notably, spammers engage in link farming to get higher ranks in social media or search engines (Ghosh et al., 2012). Another notable example of attacks on graph-based models is Sybil attacks (Douceur, 2002), in which a malicious user creates fake accounts to increase the power of a single user (Yang et al., 2014). We will devise relational adversarial perturbations to attack graph convolutional network (GCN) (Kipf & Welling, 2017) and GraphSage (Hamilton, Ying, & Leskovec, 2017) recently proposed neural models for graph-based learning. We focus on the task of node classification, wherein given labels for some of the nodes in the graph, the model will learn to predict labels for the unlabeled nodes in the graph.

Node classification can be studied in *transductive* and *inductive* settings; in the former the model has access to the whole graph, including the test nodes, and uses that as extra knowledge. In the latter, the model does not see test nodes in training, and will only add the test nodes to the graph at test time. The latter setting has more applications and is more realistic; nodes are being added to real-world graphs constantly, and having access to all the nodes for training is a big limitation. A concurrent work (Dai et al., 2018) has studied attack on GCNs for transductive settings. We will study adversarial perturbations in both settings.

Robust Graph Neural Nets

While we can perform adversarial training for text, i.e., creating adversarial examples during training, we cannot use the same procedure for graph neural networks. Specifically, for each node in training, estimating the best edge perturbation would require computing gradients and searching among all estimates which is in the order of number of nodes in the graph; this would effectively lead to a polynomial time

search to find worst-case edge manipulation in a graph. Thus, we need to develop methods to create novel defense mechanisms which would not require finding these worst case perturbations.

We first explore some architectural improvements to regular message passing schemes in graph neural nets using gating mechanisms, which would lead to a multi-view message passing scheme, wherein nodes receive different messages from a common neighbor. Apart from achieving state-of-the art results on some node classification benchmarks, we are able to use this gating mechanism toward a more robust node classifier. Concretely, we will incorporate a regularization term that would help the model produce higher gate values for edges present in the graph, compared with random edges not present in the graph.

Further, we will show that there is a connection between perturbation of the input features and perturbation of edges in graph neural nets. Intuitively, edge perturbation can be regarded as constrained feature perturbation in the convex hull of the dataset. This will enable us to perform a variant of adversarial training on the features of nodes, without performing edge-level perturbations. We will extend this idea to improve robustness to norm-bounded feature perturbations on neighboring nodes, which would also help the model be robust to edge perturbations in the graph.

Summary

There has been a surging interest in studying adversarial examples in the past few years. The overwhelming majority of the literature has focused on image classification, with little attention paid to models with discrete input. In this dissertation, we focus on two canonical example of discrete inputs in machine learning problems, namely text and graphs. We will investigate the robustness of two neural

models: 1) a character-based neural model for NLP; and 2) graph convolutional neural nets and their inductive analogue for relational data.

We will create adversarial perturbations and devise methods to defend against such attacks. Specifically, we will study the following research questions:

- Perturbing text to trick a differentiable NLP model.
- Perturbing graphs to trick a graph convolutional neural net in transductive and inductive settings.
- Performing targeted attacks on a machine translation model.
- Contrasting black-box and white-box attacks for NLP in various settings.
- Devising methods to approximate edge-based perturbations for efficient robust training of graph neural nets.

The rest of the dissertation will cover each section of this introduction in depth, and will discuss future directions in the last chapter. In Appendix A, we give a short background on deep learning, and introduce the models we have used in our work.

Chapter III of this dissertation includes published co-authored material (Ebrahimi, Rao, Lowd, & Dou, 2018). Chapter IV of this dissertation includes published material (Ebrahimi, Lowd, & Dou, 2018).

CHAPTER II

RELATED WORK

Introduction

Deep learning has had tremendous success in several areas of artificial intelligence, including computer vision, natural language processing, and speech recognition. As deep learning researchers become more confident in solving important problems, the need to understand our models' vulnerabilities and failures become more crucial. This is even more important in critical applications, such as autonomous car driving or fraud detection. In some cases, real-world adversaries are actively working to trick our system; spammers have been trying to come up with creative methods to scramble text such that it bypasses spam filter system.

The earliest work on adversarial examples focused on linear spam classifiers. Dalvi et al. (2004) pose classification as a game between the classifier and the adversary, and create a classifier that can defend against an adversary's optimal strategy. Lowd and Meek (2005) attack a linear model by reverse engineering its weights for different types of features, without requiring full access to the model.

Building models which are resilient to noisy inputs has been studied in the machine learning community in the robust optimization literature (Xu, Caramanis, & Mannor, 2009). Concretely, a robust model aims to find the solution to the following minimax optimization problem, where δ is the perturbation vector, p determines the type of norm-bounded perturbation, ϵ is the magnitude of noise, and θ is the model parameters which we fit to minimize the loss.

$$\min_{\theta} \max_{\delta} J_\theta(x + \delta, y) \quad \text{s.t. } \|\delta\|_p \leq \epsilon \tag{2.1}$$

Robust convex optimization can provably give performance guarantees for the model to defend against noisy/adversarial inputs. In the case of neural nets, the

minimax problem is non-convex, and the inner maximization is solved by gradient-based perturbations which can be found by backpropagation. Consequently, such performance guarantees as in the case of convex models cannot be granted.

A variant of robust support vector machines (SVMs) was proposed by Globerson and Roweis (2006), wherein the learned model is robust to feature deletions at test time. Teo, Globerson, Roweis, and Smola (2008) follow a similar approach, but includes a larger variety of attacks for different tasks. Xu et al. (2009) show that an SVM robust to a given norm-bounded noise, can be created by training a regularized version of the SVM, which includes a dual norm term on the weights of the SVM. Similarly, Torkamani and Lowd (2013) study robustness of structural SVMs; in all of these works, the minimax optimization problem in 2.1 is solved by replacing the inner maximization with the minimization of its dual which leads to a single quadratic minimization problem.

In the coming sections of this chapter, we will cover related work in the literature of adversarial attack and defense mechanisms for deep learning.

Adversarial Attacks

In this section, we cover the literature on adversarial attacks. We will start by several methods for white-box adversarial attacks, and conclude with a review of black-box attack approaches. The main category for creating adversarial examples is linear attacks, in which the loss function is approximated by the gradient. There are non-linear optimization methods for creating adversarial examples which are slower than linear methods, and hence are only good for attacking models, and are not useful for adversarial training, a procedure to make the models robust, which we will cover in Section II.

Linear Attacks. We know from calculus that the difference between the value of function J, at point $x + \delta$ and its value at x can be approximated by a linear sum over δ, the difference between the two points in input space. Concretely,

$$J_\theta(x + \delta) - J_\theta(x) \approx < \delta, \nabla_x J_\theta(x) > \tag{2.2}$$

Now if the goal is to increase the loss with respect to δ, we only need to maximize the RHS. The attacks, which we study in this section, either closely follow the formulation in 2.2, or create adversarial examples using gradients, $\nabla_x J(x)$. We call all of these approaches linear attacks, as they approximate the loss by gradients. Note that, all of the following methods include nonlinear terms in their constraints, which would render the overall optimization procedure, nonlinear.

One-Shot Attack. I. J. Goodfellow et al. (2015) propose an efficient *one-shot* method, which given constraints on the norm of the perturbation, would lead to efficient creation of adversarial examples. In particular, given a max norm constraint on the perturbation vector, $\| \delta \|_\infty \leq \epsilon$, one can find the best perturbation vector as in 2.3.

$$\delta = \epsilon \text{sign}(\nabla_x J_\theta(x, y)), \tag{2.3}$$

Given an ℓ_2 constraint (i.e., $\| \delta \|_2 \leq \epsilon$), an optimal one-shot attack would be:

$$\delta = \epsilon \frac{\nabla_x J_\theta(x, y)}{\| \nabla_x J_\theta(x, y) \|_2} \tag{2.4}$$

where the numerator is the gradient of the loss function with respect to its input, and the denominator is the magnitude of the gradient (measured with the ℓ_2 norm).

One-shot attacks are generally considered to be weak, but the advantage of this type of adversarial example is that its fast generation would enable us to use them in training, and hence increase models' robustness. For instance, I. J. Goodfellow et al.

(2015) report some improvement over dropout in increasing a model's accuracy when these adversarial examples are used in training.

Iterative Attack. A reasonable extension to the previous method is projected gradient descent, which finds an adversarial example in several steps. This was first proposed by Kurakin, Goodfellow, and Bengio (2017a), in which they start with a bigger perturbation magnitude and project the adversarial example onto the desired ball. This is done for several iterations, which would lead to harder adversarial examples to defend. This projected gradient method was later improved by Madry et al. (2018) to create even stronger adversarial examples. Dong et al. (2018) applied the idea of momentum, e.g. keeping a memory for gradients for adaptive update of the input gradients, and created even harder adversarial examples to defend.

DeepFool (Moosavi-Dezfooli, Fawzi, & Frossard, 2016) attacks a model by moving an adversarial example towards, and finally to the other side of, a decision boundary. For a linear classifier, the best perturbation vector has a closed form solution; however for a general classifier they have to perform a gradient-descent approach to reach their goal. The method can be regarded as an iterative method, in which the direction of the perturbation vector is not necessarily aligned with the gradient, but toward the decision boundary.

Saliency maps are used for visualizing and interpreting predictive models (Simonyan, Vedaldi, & Zisserman, 2013). The map rates each input feature $x_{(i)}$ (e.g. each pixel) on its influence over the model to predict a particular class c. One can represent saliency maps as:

$$S(x_{(i)}, c) = \begin{cases} 0 & \text{if } \frac{\partial g(x)_{(c)}}{\partial x_{(i)}} < 0 \text{ or } \sum_{c' \neq c} \frac{\partial g(x)_{(c')}}{\partial x_{(i)}} > 0 \\ -\frac{\partial g(x)_{(c)}}{\partial x_{(i)}} \cdot \sum_{c' \neq c} \frac{\partial g(x)_{(c')}}{\partial x_{(i)}} & \text{otherwise} \end{cases}$$

$S(\cdot)$ measures positive correlation of $x_{(i)}$ with c, and its negative correlation with other classes $c' \neq c$. An attacker can exploit this saliency map by targeting an adversarial class t that does not match the true class label y of a given sample x (Papernot, McDaniel, Jha, et al., 2016). This adversary inverts promising pixels iteratively, with the goal of minimizing the ℓ_0 difference between the adversarial and original image, i.e., the count of different pixels in corresponding locations. The use of saliency maps, and the choice of non-convex ℓ_0 leads to a search-based optimization method.

Carlini and Wagner (2017b) propose an attack, which improves previous approaches, through incorporating a new nonlinear constraint. This constraint helps find an adversarial example which the model is most resilient to. In addition, their model is easily adaptable to ℓ_0, ℓ_2, and ℓ_∞ constraints. For instance, their ℓ_2 perturbation attack can be found through this optimization problem:

$$\min_w \|\frac{1}{2}(\tanh(w)+1)\|_2 + c \cdot f(\frac{1}{2}\tanh(w)+1). \tag{2.5}$$

where f is defined as $\max(\max(Z(x_i^{'} : i \neq t) - Z(x_t^{'})), 0)$, and Z in the network's logits before the softmax. After a thorough investigation, Carlini and Wagner (2017a) show this attack framework can be adapted to attack several types of defense mechanisms, which will be discussed in Section II.

Nonlinear Attacks. One extension to linear adversarial attacks, is to approximate the loss function with second-order Taylor expansion which will include the Hessian matrix. This has been reported to have little impact on the power of the adversary (Tramèr, Papernot, Goodfellow, Boneh, & McDaniel, 2017).

Szegedy et al. (2014) solve the following optimization problem to find an adversarial example around $x \in \mathbb{R}^m$ whose label l differs from the original example.

$$\begin{aligned} &\underset{\delta}{\text{minimize}}\ c\|\delta\|_2 + J(x+\delta, l) \\ &\text{subject to} \quad g(x+\delta) = l \\ &\qquad\qquad x + \delta \in [0,1]^m \end{aligned} \tag{2.6}$$

They start from a large c and perform line search to find the smallest c which can minimize the objective function. They use box-constrained L-BFGS after every step of the line search, which is a quasi-Newton method that approximates the Hessian matrix using gradients. Tabacof and Valle (2016) use a similar formulation to (Szegedy et al., 2014), but perform a bisection search to find the optimal c.

Adversarial Example Generation with GANs. In a radically different approach, few recent works have investigated the problem of creating adversarial examples with generative adversarial networks (GAN) (I. Goodfellow et al., 2014), a successful architecture for generative modeling. Xiao et al. (2018) creates perturbation vectors with the help of a GAN whose generator receives the input image, and whose output will be the perturbation vector. The addition of the input and the perturbation vector will be considered a fake example, which will be fed to a discriminator. An additional term in the objective function will increase the loss on the attacked classifier. Another work by Song, Shu, et al. (2018) removes the requirement of having original images, and generates adversarial examples for different classes, which are validated by crowdsourcing to be adversarial examples, i.e., belonging to the class they claim to be. Zhao et al. (2018) perturb the latent representation in GAN to produce examples which would trick the model.

Unlike previous methods, GAN-based optimization methods do not require norm-bounded perturbations. Note that GAN is a nonlinear model which incorporates

neural architectures in its generator and discriminator. In addition, these attacks do not approximate the loss function with gradients. Thus, we can call these methods as the most nonlinear adversarial attacks.

Black-Box Attacks. The aforementioned attacks are all white-box, and rely on having access to the model to inflict loss. But several works demonstrate that simply cutting this access would not secure a machine learning model. Szegedy et al. (2014) found that adversarial examples can fool the same neural networks trained by different training instances, thereby pointing to the weaknesses in models' defending against black-box attacks. Papernot, McDaniel, and Goodfellow (2016) showed that adversarial examples can not only fool other neural networks with different architectures, they can also trick other classifiers trained by different machine learning algorithms, such as SVMs. Highly transferable adversarial examples can be used to perform black-box attacks on models. A typical approach to launch black-box attacks is to train a substitute neural network model and then generate adversarial examples against the substitute model (Papernot et al., 2017).

Y. Liu, Chen, Liu, and Song (2017) investigate transferability of targeted and non-targeted adversarial examples, and found that non-targeted adversarial examples are much more transferable than targeted ones. And interestingly, they observe that in some cases decision boundaries of different models aligned with each other. Tramèr et al. (2017) find that adversarial examples span a subspace of large ($\sim$25) dimensionality. They show that adversarial subspaces with higher dimensionality are more likely to intersect, and a significant fraction of their subspaces is shared, thus leading to higher transferability.

Due to difficulty of creating white-box adversarial examples for NLP, most previous work in adversarial examples for NLP (Belinkov & Bisk, 2018; Iyyer et al.,

2018; Jia & Liang, 2017; Ribeiro et al., 2018; Zhao et al., 2018) has focused on creating adversarial examples in a black-box setting. Jia and Liang (2017) add distracting sentences, which are polished by crowdsourcing, to the end of paragraphs to fool deep reading comprehension systems. These confusing statements incorporate elements of the question and correct answer, and are untrue but grammatical statements. Hosseini et al. (2017) manipulate toxic words in a sentence to reduce their toxicity score. In both of these works, the adversary relies on knowledge about what the model does, in order to trick them. Zhao et al. (2018) search for black-box adversarial examples in the space of encoded sentences and generate adversarial examples by perturbing the latent representation until the model is tricked. However, it is not clear how many queries are sent to the model or what the success rate of the adversary is. Iyyer et al. (2018) rely on backtranslation to create paraphrases which follow certain syntactic templates. These paraphrases are used to attack models by simply querying the models to see if these examples can trick them. Ribeiro et al. (2018) use linguistic rules to perturb an input text, and perform a greedy search to apply rules in an iterative search-based procedure.

We now briefly introduce our work on creating adversarial examples, to place it in the context of related work. We will explore these ideas in further detail in Chapter III.

Our Approach to Create Adversarial Examples. We first point out that the first work of crafting white-box adversarial examples for NLP was proposed by Papernot, McDaniel, Swami, and Harang (2016). They add noise to the word embeddings and search the neighborhood of that new embedding in order to replace the original word. While their adversary is able to trick the classifier, the word-level changes tend to produce nonsensical sentences (e.g., "I wouldn't rent this..."

→ "Excellent wouldn't rent this..."). Other methods (Miyato et al., 2017; Wu, Bamman, & Russell, 2017) simply use noise on the word embeddings to create *pseudo-adversarial* examples, which are only useful for adversarial training, i.e., they do not lead to actual textual adversarial examples. How can we go from white-box noise on word embeddings to noise on the discrete input? The answer is in one-hot vector representation of text or other types of discrete inputs, such as graphs.

Word embeddings are vectors in low dimensional space, which capture syntactic and semantic properties of words (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). We often use indexes to these vectors in the vocabulary when training NLP models. This is simply an $O(1)$ lookup operation in a dictionary. This lookup operation can also be represented by a vector-matrix multiplication, where the vector is a one-hot vector and the matrix is the vocabulary matrix. Concretely, the following holds:

$$w_i = x_i W \tag{2.7}$$

where w_i is the word embedding of the ith word, W is the vocabulary and x_i is a one-hot vector with 0 in all dimensions except for the ith. A similar method can be used for character-level models in NLP, and any other model, in which a lookup operation is used to select representations. For instance, in a character-level model, instead of word embeddings, we have character embeddings. For a graph, in which nodes have features, the vocabulary simply contains a dictionary to access nodes' features.

It is straightforward to use the chain rule to backpropgate error with respect to w_i, the embedding, to the one-hot vector.

$$\frac{\partial J}{\partial x_i} = W^T \frac{\partial J}{\partial w_i} \tag{2.8}$$

where J is the loss of the model. The current gradients are still not quite useful; for instance, what does $\frac{\partial J}{\partial x_i}^k$, the kth dimension of this gradient vector mean? In the next

chapter, we will demonstrate how we can use these gradients to score manipulations of text, such as replacing one character to another. Further, we can use a greedy to beam search strategy to apply manipulations iteratively. None of the previous NLP works have put forth a quantitative analysis of targeted and untargeted adversarial examples. We will investigate two new attack scenarios for machine translation, and propose a new framework for creating and evaluating targeted adversarial examples for machine translation. Our work on adversarial perturbations for graphs are limited to untargeted attacks.

Defense

In this section, we focus on different defense mechanisms for deep learning models.

Gradient Masking. Gradient masking is a method to obfuscate gradients by means of numerical instability to ensure that a white-box adversary fails in attacking the system. This was first proposed by Papernot, McDaniel, Wu, et al. (2016), by setting a large T in the computation of the softmax function, shown in 2.9, which would make the model make confident predictions for the true label, producing unstable gradients to attack the model:

$$\frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{2.9}$$

where z_j is the final output of the network for class j, and T is the temperature hyper-parameter, which is normally set to 1. However, Carlini and Wagner (2017b) show that using the logits before the softmax, we can still attack the model, rendering the defense useless. A similar work (Brendel & Bethge, 2017) shows the weakness in another gradient masking defense mechanism proposed by Nayebi and Ganguli (2017). An extensive study conducted by Athalye et al. (2018) shows that gradient obfuscation can give a misleading sense of security.

Adversarial Example Detection. Some defense mechanisms empower the models with a built-in capability to detect an adversarial example, through having auxiliary components in the model for that goal, or simply augmenting training data with adversarial samples. Metzen et al. (2017) create a binary classifier to detect adversarial examples as an auxiliary network is added to the original neural network. Grosse, Manoharan, et al. (2017) add an outlier class to the original dataset, composed of adversarial examples, which the model learns to detect by classifying them as the outlier class.

Feinman, Curtin, Shintre, and Gardner (2017) hypothesize that adversarial examples lie off the data manifold, and provide a Bayesian view of detecting adversarial examples. They show that uncertainty of adversarial examples is higher than the clean data, by measuring randomness in predictions after *dropout* (N. Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) is added to layers of the model. Song, Kim, Nowozin, Ermon, and Kushman (2018) show that the distribution of adversarial examples is different from clean data. They calculate p-value based on the rank given by PixelCNN (Oord, Kalchbrenner, & Kavukcuoglu, 2016), and reject adversarial examples using these statistics. Carlini and Wagner (2017a) found that almost all of the adversarial example detection algorithms are defeated in a variety of white-box and black-box settings, using appropriate objective functions for the adversary.

Adversarial Training. Adversarial training interleaves training with generation of adversarial examples (I. J. Goodfellow et al., 2015). Concretely, after every iteration of training, adversarial examples are created and added to the mini-batches. Virtual adversarial training (Miyato et al., 2016) is another regularization method, which aims to minimize the KL divergence between predictions on the

examples and their adversarial counterparts, for semi-supervised tasks, wherein we have a limited number of labeled examples. To deal with the transferred black-box model, Tramèr, Kurakin, Papernot, Boneh, and McDaniel (2018) propose *Ensemble Adversarial Training* method that trains the model with both white-box and imported black-box adversarial examples.

According to (Kurakin, Goodfellow, & Bengio, 2017b), one-shot attacks, e.g., FGSM, are easy to defend for a model which is trained on these white-box adversarial examples, but these attacks have high transferability rate, and can be used to attack other models in a black-box setting. In addition, Tramèr et al. (2018) show that a FGSM-trained model is more robust to white-box attacks than to black-box attacks due to gradient masking. They propose Rand-FGSM, which adds Gaussian noise to the example before computing the gradients. A projected gradient-based (PGD) approach to create adversarial examples by Madry et al. (2018) has proved to be one of the most effective defense mechanisms against adversarial attacks for image classification. The PGD algorithm, which creates adversarial examples within $\| \delta \|_\infty \leq \epsilon$, is given in Algorithm 1. The difference between this approach and the one proposed by I. J. Goodfellow et al. (2015) is the lack of clean examples in training and its closer connection with robust optimization for a non-convex problem.

Adversarial training has been used to make text classification (Miyato et al., 2017) and information extraction (Wu et al., 2017) more robust. However, instead of creating real-world textual adversarial examples, they add noise to word embeddings and create adversarial examples in continuous space.

Robust Optimization. Recall that when the problem posed in 2.1 is convex, robust convex optimization can provably give performance guarantees for the model to defend against noisy/adversarial inputs. In the case of neural nets, the minimax

Algorithm 1 Adversarial training

INPUT: model parameters θ, stepsize sequence $\{r_t > 0\}_{t=0}^{T-1}$
for $t = 0, \ldots, T-1$ **do**
  $x^a \leftarrow x$
  for $i = 0, \ldots, iter$ **do**
    $x^a \leftarrow x^a + \Pi_\epsilon \, \epsilon \text{sign}(\nabla_{x^a} J(\theta^t; x^a))$
  $\theta^{t+1} \leftarrow \theta^t - r_t \nabla_\theta J(\theta^t; x^a)$

problem is non-convex; up until recently it was not known how to train robust neural classifiers. Kolter and Wong (2017) builds on linear ReLU relaxations, first proposed by Ehlers (2017) for neural network verification. This defense find a convex outer bound for the adversarial polytope, i.e., the set of all final-layer activations attainable by perturbing the input. Sinha, Namkoong, and Duchi (2018) provide a method for achieving certified robustness for perturbations defined by a certain distributional Wasserstein distance, which is not translatable to norm-bounded perturbations. Raghunathan et al. (2018) develop a semidefinite programming-based relaxation of the adversarial polytope and employ this for training a robust classifier. Both Raghunathan et al. (2018) and Kolter and Wong (2017) create worst-case adversarial examples in the outer bound via dual; for the latter of which, this results in another neural network, which looks like the backward pass in the original network.

We now briefly introduce our work on defending against adversarial examples. We will explore these ideas in further detail in Chapter III.

Our Approach to Defend Against Adversarial Attacks. The main method we use to defend models is adversarial training. In particular, our discrete adversarial example generation method is efficient for NLP models, especially for character-level models with a few hundred tokens in the vocabulary. Our adversarial training algorithm for our machine translation and text classification experiments were only 3-4 times slower than vanilla training.

As we will elaborate later, this procedure is not efficient for large graphs, in which the size of the vocabulary (i.e., W in Equation 2.7), could be hundreds of thousands, or even more. So we devise efficient methods to bypass this issue. One of our methods is similar to the defense method of detecting adversarial examples; we will develop a gating mechanism which will help us measure trustworthiness of edges. When the output of the gate is small the impact of the edge is minimized. Later, we will show that there is a connection between *relaxed* edge perturbations and unrestricted feature-based for the nodes of the graph, and we develop a gradient-based method which perturbs features and projects them in the outer cube of the convex hull. Building on this observation, we will develop three more baselines which make the model robust to perturbations on the features of neighboring nodes. In one of these methods, we look at robust training of graph neural nets through convex relaxation of neural nets. All of these models significantly improve the robustness of vanilla models against adversarial edge perturbations. In short, we provide a set of novel algorithms for robustness of graph neural nets, which belong to different classes of defense mechanisms.

Summary

In this section, we provide a brief summary of related work, and how our contributions fit within the literature. We will compare our methods with the previous work along the main axes of adversarial machine learning.

Attacks on Continuous vs. Discrete Spaces. The overwhelming majority of the literature uses image classification as the task for studying adversarial examples (Carlini & Wagner, 2017a; I. J. Goodfellow et al., 2015; Kolter & Wong, 2017; Madry et al., 2018; Szegedy et al., 2014). For the first time, we propose a framework to attack neural network models with discrete input, through defining differentiable

discrete operations that estimate the loss with a limited number of queries to the model. We will focus on character-level NLP models for translation and classification as well as graph-based models for classification.

Targeted vs. Untargeted Attacks for Machine Translation. Many works on image classification focus on targeted attacks (Kurakin et al., 2017a; Moosavi-Dezfooli et al., 2016; Papernot, McDaniel, Jha, et al., 2016). In classification domains with few classes, targeted attacks are relatively simple, since an adversary can perturb the input to move it to the other side of a decision boundary; whereas in machine translation, we deal with vocabulary sizes in the order of at least tens of thousands, and it is less likely for an adversary to be successful in targeted attacks for most possible target words. Zhao et al. (2018) provide a few hand-picked adversarial examples for machine translation; we investigate two new attack scenarios for machine translation, and propose a new framework for creating and quantitively evaluating targeted adversarial examples for machine translation.

Black-Box Attacks vs. White-Box Attacks for NLP. Due to the difficulty of creating white-box adversarial examples for NLP, most previous work in adversarial examples for NLP (Belinkov & Bisk, 2018; Iyyer et al., 2018; Jia & Liang, 2017; Ribeiro et al., 2018; Zhao et al., 2018) has focused on creating adversarial examples in a black-box setting. We compare our white-box adversarial examples with the black-box baseline of (Belinkov & Bisk, 2018), and show that white-box adversaries are much stronger, and demonstrate that adversarial training on white-box examples would make the model much more resilient to adversarial attacks.

Defense for Graph Neural Nets. Defense for graph neural nets has not been studied in any prior work. Defending against adversarial edge perturbations requires changes in the regular adversarial training procedure. We will create novel defense

algorithms to make GraphSage (Hamilton et al., 2017), a recent neural model for inductive node classification, robust. Our defense algorithms can be categorized into different classes of defense mechanism, which have been already explored for image classification. In particular, our edge-gating-based defense mechanism is related to adversarial example detection work of (Metzen et al., 2017), our convex defense model is inspired by (Kolter & Wong, 2017), and our Jacobian-based defense is related to layer-wise regularization of Gu and Rigazio (2015).

CHAPTER III

ADVERSARIAL EXAMPLES FOR TEXT

Ebrahimi, J., Rao, A., Lowd, D., & Dou, D. (2018). Hotflip: White-box adversarial examples for text classification. In Proceedings of the 56th annual meeting of the Association for Computational Linguistics (volume 2: Short papers) (Vol. 2, pp. 31-36).

Introduction

In this chapter, we will describe an efficient method to generate white-box adversarial examples for text. The method estimates the increase in loss by first-order approximation of loss over a manipulated input, e.g., removal or addition of a character. We will provide adversarial examples for a character-level neural text classifier and a character-level machine translator. We will also discuss transferability of adversarial examples (i.e., ability to trick models to whose parameters the adversary has not access to), and human perception of textual adversarial example.

White-Box Adversarial Examples

Human language processing faculty can detect typos, misspellings, and the complete omission of letters when reading (Rawlinson, 1976). The following amusing text circulated on the internet in September 2003, and has been passed on many times, including through internet memes.

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae.

As an experiment, Belinkov and Bisk (2018) gave a noisy German version of the sentence above to Google Translate, and got the following gibberish translation.

After being stubbornly defiant, it is clear to kenie Rlloe in which Reiehnfogle is advancing the boulders in a Wrot that is integral to Sahce, that the utterance and the lukewarm boorstbaen stmimt.

Belinkov and Bisk (2018) show that Neural Machine Translation (NMT) systems are brittle and will produce gibberish translations under typos, random noise, and natural noise. A similar work by (Hosseini et al., 2017) shows that Google's Perspective API, which is used to detect toxic language in text, can produce very different toxicity scores when noisy input is given to the model. For example, the score for the following sentence changed from 84% to 20% after the word *idiot* was changed to *idiiot*.

Climate change is happening and it's not changing in our favor. If you think differently you're an idiot.

The adversary has simply added a typo to the toxic word in the sentence, and successfully tricked the model.

In a similar approach, Jia and Liang (2017) devise adversarial examples for text comprehension system by adding a distracting sentence based on the question and true answer to the question. For instance, if the question to the system is "What city did Tesla move to in 1880?", and the true answer is "'Prague", a possible distracting sentence would be "Tadakatsu moved the city of Chicago to in 1881.", which contains some linguistic features of the question and answer, e.g., a city, a date, a name.

The central question we answer is whether we can find worst-case failures of an NLP model through the lens of white-box adversarial examples. Concretely, the aforementioned examples were black-box attacks which were created by random changes or heuristic-based changes. In such a scenario the adversary does not have a preference over possible manipulations, except for querying the model to see if the

model's output has changed, or relies on heuristics which are limited to that task. Our work, on the other hand focuses on creating adversarial examples solely based on the generic optimization problem to increase the loss of an NLP model.

Since the space of possible discrete changes is combinatorial, querying the model to rank adversarial manipulations is intractable. However, in a white-box setting the adversary can use model's gradients as a surrogate loss function to estimate the effectiveness of different manipulations, and manipulate text accordingly. In other words, a white-box adversary can use gradients to estimate the loss without exhaustively querying the model for every manipulation. Table 1 shows a correctly-classified sentence on top; the next two examples show a successful and an unsuccessful manipulation, respectively, where in the first one, the label of the text has changed from *World* to *Sci/ Tech*, while the label of the second example has not changed. A naive adversary would query the model for its loss, and pick the adversarial example which increased the loss more; a gradient-based adversary can estimate the loss, and avoid excessive querying of the model. As we will see, the gradient based adversary is orders of magnitude faster than the naive adversary.

We develop a method, *HotFlip*, to create white-box adversarial examples, which can be applied to character-level models, word-level models, and more generally any model with a one-hot representation scheme for its input data. At the core of our method lies an atomic character *flip* operation which changes one token to another, which can be extended to insertion and deletion operations to constitute a set of plausible adversarial attacks. While a few character changes can trick a character-level model, the meaning of the text is very likely to be preserved or inferred by the reader. By contrast, word-level adversarial manipulations are much more likely to change the meaning of text.

Table 1. Successful and unsuccessful adversarial examples.

South Africa's historic Soweto township marks its 100th birthday on Tuesday in a mood of optimisim. 57% **World**
South Africa's historic Soweto township marks its 100th birthday on Tuesday in a moo**P** of optimisim. 95% **Sci/Tech**
South Africa's historic Soweto townshi**q** marks its 100th birthday on Tuesday in a mood of optimisim. 52% **World**

Method

The architecture we study for our character-level experiments is based on the one proposed by Kim, Jernite, Sontag, and Rush (2016) for character-level language modeling. Feature extraction is performed by convolutional neural networks (CNN) over characters, which are passed to layers of highway networks, and finally given to stacks of long short-term memory networks (LSTM) for modeling a sequence of words. This architecture has be used to perform sequence labeling (Kim et al., 2016) and machine translation (Costa-Jussa & Fonollosa, 2016). We adapt it for our text classification too, wherein the output of the last recurrent unit is passed to a softmax to predict the label of text.

Text Representation. Let us use character-level text classification as our running example, and let V be the alphabet, x be a text of length L characters, and $x_{ij} \in \{0,1\}^{|V|}$ denote a one-hot vector representing the j-th character of the i-th word. The character sequence can be represented by

$$x = [(x_{11},.. \ x_{1n});..(x_{m1},.. \ x_{mn})]$$

wherein a semicolon denotes explicit segmentation between words. The number of words is denoted by m, and n is the number of maximum characters allowed for a word, where padding is applied if the number of characters is fewer than the maximum.

HotFlip: Differentiable String-Edit Operations. Let $J(x, y)$ denote the log-loss over the output of the softmax unit. Imagine the adversary is allowed to

change r characters in the input text to fool the model to which it has access to. Using a brute-force search, it would need to do $\binom{L}{r}|V|^r$ forward passes to exhaust the search space. That is, query the classifier, by calling J, for all combinations of character flips within the allowed budget, r. This can decrease to $\mathcal{O}(brL|V|)$ if beam search is used. That is, after trying all possible single character flips, keep the top b flips which had the highest loss increase, and continue for r steps.

Our algorithm requires a single forward and a single backward pass to estimate the best possible flip. We represent string operations as vectors in the input space and estimate the change in loss by directional derivatives with respect to these operations. Based on these derivatives, the adversary can choose the best loss-increasing direction.

A character **flip** in the j-th character of the i-th word ($a \rightarrow b$) can be represented by this vector:

$$\vec{v}_{ijb} = (\vec{0}\,,..;(\vec{0}\,,..(0,..\text{-}1,0,..,1,0)_j,..\vec{0}\,)_i;\ \vec{0}\,,..)$$

where -1 and 1 are in the corresponding positions for the a-th and b-th characters of the alphabet, respectively, and $x_{ij}^{(a)} = 1$. Due to directional derivatives, we have the following:

$$\nabla_{\vec{v}_{ijb}} J(x,y) = \nabla_x J(x,y)^T \cdot\ \vec{v}_{ijb}$$

A first-order approximation of change in loss can be obtained from a derivative along this vector:

$$\nabla_{\vec{v}_{ijb}} J(x,y) = \nabla_x J(x,y)^T \cdot\ \vec{v}_{ijb} = \frac{\partial J}{\partial x_{ij}}^{(b)} - \frac{\partial J}{\partial x_{ij}}^{(a)} \tag{3.1}$$

An immediate benefit of using derivatives with respect to the one-hot vectors is that they can be used to select the best character change ($a \rightarrow b$). Concretely, the derivative vector contains the information about the loss increase obtained due to a character flip. This makes the number of queries (forward passes) be independent of the alphabet size.

We simply need to find the best change by **maximizing** eq. 3.1, to *estimate* the best character change ($a \rightarrow b$). This requires searching in $|V|mn$ values for a given text of m words with n characters each, in a vocabulary of size $|V|$.

Character **insertion** at the j-th position of the i-th word can also be treated as a character flip, followed by more flips as characters are shifted to the right until the end of the word. Hence the estimate in change of loss corresponding to a character insertion can be given by:

$$\frac{\partial J}{\partial x_{ij}}^{(b)} - \frac{\partial J}{\partial x_{ij}}^{(a)} + \sum_{j'=j+1}^{n} \left(\frac{\partial J}{\partial x_{ij'}}^{(b')} - \frac{\partial J}{\partial x_{ij'}}^{(a')} \right) \tag{3.2}$$

where $x_{ij'}^{(a')} = 1$ and $x_{ij'-1}^{(b')} = 1$.

Similarly, character **deletion** can be written as a number of character flips as characters are shifted to the left. Since the magnitudes of operation vectors are different, we normalized them by both L_1 and L_2 norm but found it had little impact.

Multiple Changes. We explained how to choose the best single change in text to get the maximum increase in loss. After applying a change, we query the classifier and check whether the predicted label also changes, upon which we stop the search. For additional changes we can either perform manipulations greedily, i.e., applying the best manipulation at every step of the way, or keep the most promising manipulations and perform a beam search. The search is continued until we reach the maximum number of allowed changes, r, or successfully trick the classifier into misclassifying the instance.

Beam Search. Our proposed beam search requires only $\mathcal{O}(br)$ forward passes and an equal number of backward passes. Concretely, there are two sources of efficiency in our algorithm: First, the breadth-search is done simply by sorting the derivatives rather than querying the classifier for the actual loss; second, after the

breadth-search, we keep the top r paths, which are sorted by the sum of the loss up to the current step and gradient of the current step, rather than querying to obtain their actual loss. We elaborate on this with an example; consider the loss function $J(.)$, input x_0, and an individual change c_j. We estimate the score for the change as $\frac{\partial J(x_0)}{\partial c_j}$. For a sequence of 3 changes [c_1,c_2,c_3], the following holds:

$$\text{score}([c_1, c_2, c_3]) = J(x_2) + \frac{\partial J(x_2)}{\partial c_3}$$

where x_2 is the modified input after applying [c_1, c_2]. With a beam width of b, we need b forward and b backward passes to compute derivatives at each step of the path, leading to $\mathcal{O}(br)$ queries. In contrast, a loss-based approach requires computing the actual loss for every possible change at every stage of the beam search, leading to $\mathcal{O}(brL|V|)$ queries. Thus, our algorithm limits the number of queries by using derivatives as surrogates for change in loss.

To showcase the efficiency of our method, we compare the amount of time it takes to carry out a beam-search method based on querying for the actual loss, and our derivative-based beam-search approach. Because the derivative approach is less exhaustive, it requires a larger beam size, so we set the beam size for the derivative approach to 10, and for the loss-querying approach to 5.

The sample that we used for this experiment had an 80-token alphabet and an average of 44 characters per sentence. Both approaches had a 100% success rate (perfect misclassification) on this sample. It can be seen in Table 2 that the derivative-based approach needs an average of 1 more character flip to trick the classifier, e.g., 34% of examples are misclassified after one change using the derivative-based approach, while this proporation increases to 59% for the loss-based approach. However, the derivative-based approach is 90 times faster on average; if we decrease

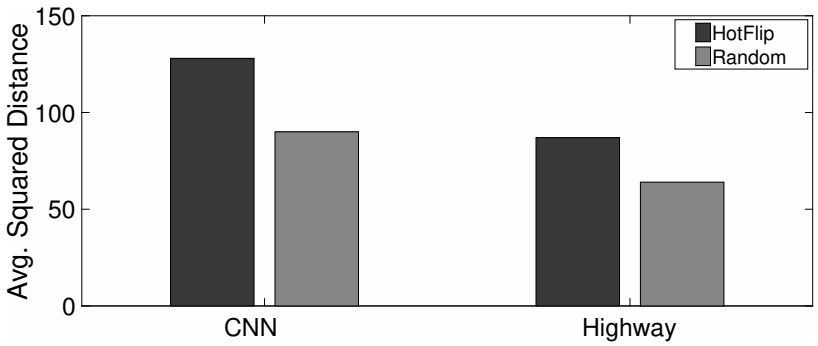


Figure 1. Comparing the HotFlip direction and a random direction based on the average squared distance between the embedding of the original word, and the embedding of the modified word, found from the outputs of the CNN and highway layers.

the beam size to 5 for our method, the adversary will fail to misclassify a few of the examples, but the speedup will increase to 400 times.

Table 2. Comparing beam-search strategies, based on the number of changes (i.e., 1,2, 3 or more). We report the proportion of successfully created adversarial examples and the average time, measured in seconds, spent for each number of change.

No. change(s)		1	2	3+
Loss-based	Time	10.3	70.2	705
	Proportion	59%	29%	12%
Gradient-based	Time	**0.11**	**0.93**	**2.7**
	Proportion	34%	29%	37%

Embeddings Under Adversarial Noise

To compare changes by the HotFlip direction with a random direction, we measure the average squared distance between the original word representation in each layer (i.e., using the outputs of the CNN and the highway layers) with those after applying changes in each direction. Figure 1 shows a much larger difference between the embedding of the original word and the one modified in the HotFlip direction.

In Table 3, we study the change in the word embedding as a change occurs. We use the output of the highway layer as the word representation. We report the embedding for a few adversarial words, for which the original word is not among their

Table 3. Nearest neighbor words (based on cosine similarity) of word representations from CharCNN-LSTM picked after the highway layers. A single adversarial change in the word often results in a big change in the embedding, which would make the word more similar to other words, rather than to the original word.

Alps → llps	talk → taln	local → loral	you → yoTu	ships → hips	actor → actr	lowered → owered
lips	tall	moral	Tutu	dips	act	powered
laps	tale	Moral	Hutu	hops	acting	empowered
legs	tales	coral	Turku	lips	actress	owed
slips	talent	morals	Futurum	hits	acts	overpowered

top 5 nearest neighbors. Deletion examples (i.e., ships, actor, lowered) often maintain some information about the word, such as its prefix and tense, which makes them more benign changes.

In a character-level model, the lookup operation to pick a word from the vocabulary is replaced by a character-sequence feature extractor which gives an embedding for any input, including OOV words which would be mapped to an UNK token in a word-level model. This makes the embedding space induced in character-level representation more dense, which makes characterlevel models more likely to misbehave under small adversarial perturbations.

Experiments

We analyze the effect of character-level adversarial examples on two different tasks. Our seq-2-seq implementation relies largely on OpenNMT[1], which mostly follows the guidelines of Luong, Pham, and Manning (2015) for attentional translation. Specifically, we used a CharCNN-LSTM encoder and a word-based attentional decoder (Bahdanau, Cho, & Bengio, 2014). We used a beam size of 5 for decoding at test time. We used a 800k pair from the Europal corpus of German-English WMT data[2], and reproduced the results of Costa-Jussa and Fonollosa (2016). Preprocessing consisted of tokenizing, normalizing punctuation, and filtering sentences with more

[1]https://github.com/opennmt/opennmt

[2]http://www.statmt.org/wmt15/translation-task.html

than 50 words. For text classification, we use the AG's news group dataset[3], which consists of 120,000 training and 7,600 test instances from four equal-sized classes: World, Sports, Business, and Science/Technology.

The architecture consists of a 2-layer stacked LSTM with 500 hidden units, and a character embedding size of 25. This classifier was able to outperform (Conneau, Schwenk, Barrault, & Lecun, 2017), which has achieved the state-of-the-art result on some benchmarks, on AG's news. Both models were trained with stochastic gradient descent and gradient clipping, and the batch size was set to 64. For classification, we used 10% of the training data as the development set, and trained for a maximum of 25 epochs. For translation, we use OpenNMT's suggested hyper-parameters. Throughout our experiments, we only allow character changes if the new word does not exist in the vocabulary, to avoid changes that are more likely to change the meaning of text. The adversary uses a beam size of 5 for translation and a beam size 10 for classification. We limited per-word character change to one, which was found to facilitate faster beam search. For both tasks, we use a maximum of 10% of characters in the document as the budget for the adversary. For experiments with random changes, we use the same number of character changes as the one committed by our adversary in that experiment.

Text Classification. In Figure 2, we plot the success rate of the adversary against an acceptable confidence score for the misclassification. That is, we consider the adversary successful only if the classifier misclassifies the instance with a given confidence score. As can be seen, the beam-search strategy is very effective in fooling the classifier, even with a 90% confidence constraint, tricking the classifier for more

[3]https://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

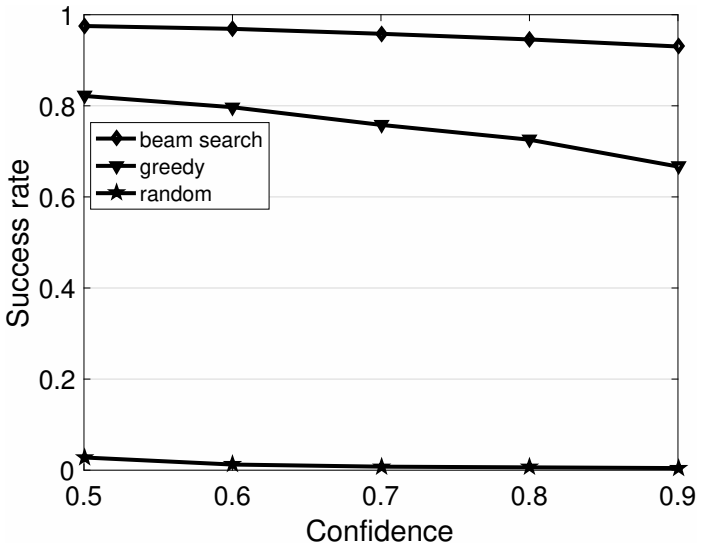


Figure 2. Adversary's success rate as a function of confidence for classification.

than 90% of the instances. A greedy search is less effective, especially in producing high-confidence scores.

In this task, 30% of adversarial examples have either one or two changes. The adversary manipulates 4.1% and 5.28% of the characters in a document on average, for 0.5 and 0.9 confidence thresholds respectively. For this task, random changes barely trick the model (2.7% success rate at 0.5 confidence); we performed a beam search with random gradients which increased the success rate to only 6.2% at 0.5 confidence. Table 4 shows some of our adversarial examples which tricked the model to change its correctly predicted label, as denoted on top of each set of adversarial examples.

In Table 5, we report the correct-to-incorrect rate (i.e., number of instances that are originally correctly classified but will be misclassified after the adversarial manipulations), and the average number of changes made by the adversary, for sentences up to 25 words. We also report the percentage of the time that the originally misclassified instances will be correctly classified after the adversarial manipulation

Table 4. Adversarial examples which are mislcassified by the model. The correct label is shown on top of each block.

Sci/Technology
Orange tells customers to Talk Now. European carrier Orange is rolling out its own Push To Talk service ahead of efforts to create a standardized PTT system. European mobile carrier OrangI haC announcet.
Arien CAntact More Likely by "Mail" Than Radio, Study Says. Researchers behind the s.udy speculate that other life-forms may have already sent us messages, perhaps even as organic material embedded in asteroids that hkve strudk EaPth.
Microsoft spends 1bn to keep out the hackers. The growing threat of hackers and Ciruses has prompted Microsoft to rAll out a billion- dollar upgrade of its Windows computer operating system to stretgthen seturity.
This week in game news. They risked hypothermia and fought off the effects of sleep depriSation so they could be among the first to achieve their quest in the wee hours of the morfing.
Energy Dept. funds open-sourhe InfiniBand work. Three-year project will back programmers' effort to build Linui software support for the high-speeu networping technolowy.
Sports
Berkman tears AfL, may be out until June. HOUSTON- Houston Amtros star outfielder Lance Berkman suffered a torn ACD in his right knee and will unvergo arthrosoopic surgery within the next \0 days, the teym announded Friday.
Sporting News: Bonds is player of year; Pujols fourth. San Francisco Giants slugger Barry Bonds, who hit .362, set a record with 232 walks and topped 700 career homers, was named 2004 player of the year by The Sporping Ne.s
Norman looking to post a low score. Greg Normaf will be looking to post a low score to give the leaders a target in the final round of the Australian PGl Championship at Coopum's Hyatt Resort, north of Brtsbane.
Montgomerie, Woods, Furyk Tied at Target (AP). AP- Colin Montgomerie was thrilled to get an invitation from Tigep Woods to play in his year-end tournament with 15 of the best players in g\f. Even better w\s matching \oods' score.
Dolfhins finally win, taking out frustration on Rams. The Miari Dolp\ins finally gave their fans reason to celebrate, corbining a polished offensibe perforpance with solid defense for their first victorp this s\ason, 31-14 over the St.
World
CtA accused over Iraq detaipees. US army generals tell a Senate committee that dozens of detainees may have been heln in secre. in Iraq.
Rumsfeld to Meet Foreign Defense Chiefs on Iraq. MANAMA (Reuters)- Defense Secretary Donald Rumsfeld was set to meet defense chiefs from about 18 nations aborrd a U.S. aircrart carrier in the Gulf Saturday as the United States looks to improve the securrty situation in Iraq with Janpary elections loom\ng.
Europe Must Adapt to U.S. View on Terror, NcTO Chief Says. The head of N\TO said there was a critical "perception gap" between Europe and the U.S. on the subject of global terror.
In Asia, Powell defends N. Kore\policy. SEO\L Secretary of State Colin L. Powell yesterday sought to fend off complaints from key partners in the effort to end North Kore\'s nuclear programs that the Bush administration has not been sufficiently creative or willing to compromise in the negotiations.
China, Singapore Say World Must Help Calm Taiwan Row. China and Singapore on Monday urged the international cofmunity to help calm Beijing's dispute with Taiwan over its puoh for mndependence.
Business
Judge: MCI may have violated court order on certain fees. NEW YORK A federa judge in Maphattan says MCI may have violated a court order by pafing more than 25 (m) million dollars in professional services fees as part of its bankruptcy proceedings in excess of capC on s\ch fees.
UK's Diageo gets \$ 2.26 bln from General Mills sale. Britain's Diageo Pll (DGE.u: Quote Proaile, Research), the world's biggent kpirits group raised \$ 2.26 billion from its saln o. v9.
Yukos warns it oil output is lagging. BeleaguereB Russian e.ergy gidnt Yukos has warned that it will not produce as much oil as expected this year. It blames bailifms who are draining its bank accounts to pay its potentially ruindus bax bi.l.
India snub for foreign airlices. NEW DELHI: The Indian government increased the foreign direct i\vestment(FeI) cap yesterday in domestic airl\nes from 40 to 49 per cent but kep\a ban on foreign carriers taking s.akes.
EU wants US to clear muddy waters in WTh row. Brussles- EU trade chief Peter Mandelson wants clarification of the US stance in threatened WTO action over aid to AiDbus, his spokeswoman said Friday after a US official indicated Washington was delaying such action.

Table 5. Evaluation of different adversarial changes on AG's news dataset for sentences with up to 25 words.

metrics	flip	insert	delete
avg. changes	3.64	3.95	4.69
correct to incorrect%	99.39	82.92	79.26
incorret-to-correct%	15.62	40.10	32.81

(i.e., incorrect-to-correct). In other words, cases wherein the adversary successfully tricks the classifier to make a different prediction, but which turns out to be the true label. As can be seen, the average number of changes required to the input in order to fool the classifier was higher for the case of deletes. Having the highest correct-to-incorrect rate, flipping is clearly the most effective operation to manipulate the input. Interestingly, flipping has the lowest incorrect-to-correct rate. This is an intriguing property for this operation which might make it an even better manipulation than other types of character manipulations.

Machine Translation. Unlike classification, for which changes in the predicted labels would imply success for the adversary, a different translation could even have a better BLEU (Papineni et al., 2002) score based on the reference translations in the dataset. To address this issue, we follow a slightly different approach in determining the adversary's success. Figure 3 plots the adversary's success rate as a function of the decrease in BLEU score caused by adversarial manipulations. In other words, the translation adversary consults the reference translations, and stops only if the new translation has a lower BLEU score by a minimum value.

Figure 3 shows that random changes can be much more damaging for the translation system output. Sensitivity of character-level translation systems to noise has also been recently observed by Belinkov and Bisk (Belinkov & Bisk, 2018). This

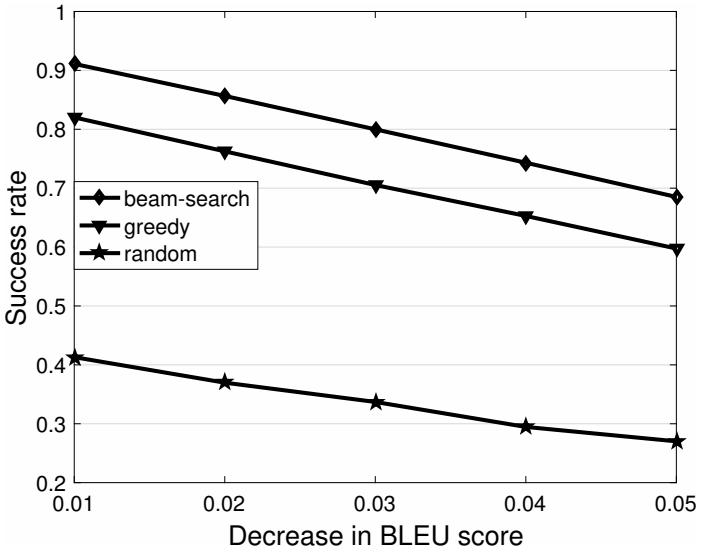


Figure 3. Adversary's success rate as a function of the decrease in BLEU score for EN→DE translation. Similar pattern exists for DE→EN translation.

can be (intuitively) explained by the fact that translation is a structured task, and instead of a single response value, a complex output is produced which can make the system much more sensitive. For this task, the adversary manipulates fewer than 3% of characters in a document on average. In fact, 60% of adversarial examples for character-level translation contain only one change. Table 6 depicts some adversarial examples for DE-EN translation.

Discussion. In this section, we discuss properties of the created adversarial examples for both tasks. Figure 4 shows the proportion of each operation chosen by the adversary for the experiments. We can see that flip is the dominant operation for classification, and to some extent for EN→DE. Insert is the dominant operation for DE→EN, and delete is almost never used for translation adversarial examples. This phenomenon is related to the results in the previous section, in which we saw the translation models were more brittle. Recall that the insert operation (defined in Equation 4.2) is a constrained flip operation, wherein a flip at character j is followed

Table 6. Adversarial examples for DE→EN translations.

src	Die Menschen, die dem Asbest in diesen Gebäuden ausgesetzt waren, erkranken nun.
adv	Die Menschen, die dem Asbest in diesen Gebäuden ausges**ü**tz**g**t waren, erkranken nun.
src-output	People who were exposed to asbestos in these buildings are now suffering.
adv-output	The people who were exterminated in these buildings are now suffering.
ref	People who were exposed to asbestos in those buildings are now coming down with the disease.
src	Ich wollte für mein Land kämpfen, aber das wird nicht geschehen.
adv	Ich wollte für mei**L**n Land kämpfen, aber das wird nicht geschehen.
src-output	I wanted to fight for my country, but it will not happen.
adv-output	I wanted to fight for Chile, but that will not happen.
ref	I wanted to fight for my country but it will not happen.
src	Örtliche Medien machten Russland für den Vorfall verantwortlich.
adv	Örtliche Medien machten Russla**F**nd für den Vorfall verantwortlich.
src-output	Local media blamed Russia for the incident.
adv-output	Local media were responsible for the incident.
ref	Local media blamed Russia for the incident.
src	Sie können mehr als 80 Prozent der Kosten eines neu gekauften Buches sparen.
adv	Sie können mehr als 80 Prozent der Kosten eines neu gekauften B**n**ches sparen.
src-output	You can save more than 80% of the cost of a new book.
adv-output	They can save more than 80% of the cost of a newly bought pot.
ref	You can save more than 80 per cent of the cost of buying a book new.
src	Rund 3000 Demonstranten, zur Residenz von Premierminister Nawaz Sharif zu gelangen.
adv	Rund **s**000 Demonstranten, zur Res**f**denz von Premierminister Nawaz Sharif zu gelangen.
src-output	around 3000 demonstrators tried to come to the head of Prime Minister Sharif.
adv-output	some demonstrators were trying to find the enemy of Prime Minister Sharif.
ref	around 3000 demonstrators attempted to reach the official residency of Prime Minister Nawaz Sharif.
src	In der Hauptstadt Islamabad haben rund 1000 Demonstranten den staatlichen Fernsehsender PTV gestürmt.
adv	n der Hauptstad**d**t Islamabad haben rund 1000 Demonstranten den staatlichen Fernsehsender PTV gestürmt.
src-output	In the capital of Islamabad, around 1000 demonstrators were killed by the state television station.
adv-output	In the capital of Islamabad, around 1000 demonstrators have plunged the state television station.
ref	In the capital city of Islamabad, around 1000 demonstrators stormed the government-run television station.
src	Ein Entschädigungsprogramm von 350 Millionen £ wird angekündigt.
adv	Ein Entschädigungsprogramm von 350 Millio**d**en £ wird angekündigt.
src-output	A EUR 350 million compensation scheme is announced.
adv-output	A compensation programme of 350 billion is announced.
ref	A £350m compensation scheme is announced.
src	Dabei seien sieben Grenzschützer verletzt worden, sagte ein Sprecher in Kiew dem Sender 112.ua.
adv	Dabei seie**c**n sieben Grenzsc**g**hützer verlo**e**tzt worden, sagte ein Sprecher in Kie**l** dem Sende**i**r 112**b**ua.
src-output	In this connection, seven border guards have been injured, a spokesman in Kiev said the station.
adv-output	In this connection, seven border guards have been killed, a spokesman in the said.
ref	During the attack seven border guards were injured, according to a spokesperson speaking to the station in Kiev.
src	Einer, dem die Pause wohl besonders gelegen kommt, ist Kapitän Hofmann.
adv	Einer, dem die Pause wohl besonders gelegen kommt, ist Kapi**L**än Hofmann.
src-output	One of those who think that the break is particularly important is captain.
adv-output	One thing that is likely to be particularly close to the break is Kabila John.
ref	For one of whom, captain, the break came in particularly useful.
src	Einer, dem die Pause wohl besonders gelegen kommt, ist Kapitän Hofmann.
adv	Einer, dem die Pause wohl besonders gelegen kommt, ist Kapi**L**än Hofmann.
src-output	One of those who think that the break is particularly important is captain.
adv-output	One thing that is likely to be particularly close to the break is Kabila John.
ref	For one of whom, captain, the break came in particularly useful.
src	Hartnäckige Bedenken aus der CDU gegen die Pkw-Maut haben die CSU zusehends ergrimmt.
adv	Hartnäckige Bedenken aus der CD**L** gegen die Pkw-M**e**ut haben die C**W**U zuseh**k**nds ergrimmt.
src-output	concerns from the CDU against passenger cars have taken the CSU off.
adv-output	concerns from the against passenger cars have taken off the.
ref	opposition from the CDU against the tolls is making the CSU increasingly angrier.

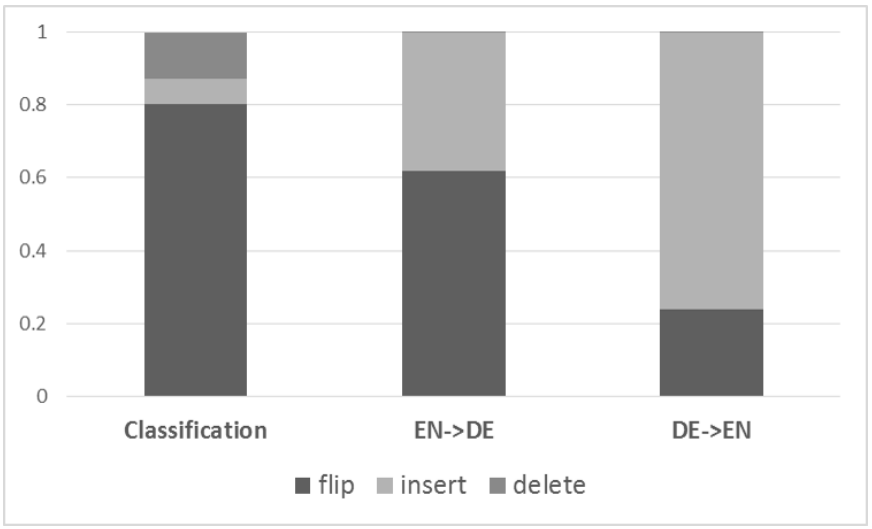


Figure 4. Proportion of types of changes that the adversary chooses.

by a set of flips to its right. Concretely, except the first flip, which the adversary can pick, the rest of the flips are given by the word's orthography and are thus constraints on the operation. We note that we observed much larger values for the gradients of the one-hot vectors in translation experiments. A higher proportion of insert operations suggests that a large number of flips, which are caused by the shift of characters (the summation in Equation 4.2), contain a large surrogate loss, given by the gradients of one-hot vectors. Similarly, this increases the likelihood that random flips cause a large loss, which is reflected in the high success rate of random changes for translation. Since German has a larger alphabet, this distinction is more pronounced. From a linguistic perspective, insertion or flip can potentially disrupt features corresponding to morphemes in the language.

Transferability of Adversarial Examples. A related point about the difference between the two tasks is studying the transferability of adversarial examples. That is, to study to what extent white-box adversarial examples of model A would be able to trick model B. While all kinds of noise could break different

Table 7. Misclassification error of Char-CNN model on adversarial examples generated by CharCNN-LSTM and random changes.

	CharCNN-LSTM	random	original
misc. error	**12.20**	9.63	9.07

Table 8. Adversarial examples imported from CharCNN-LSTM, which trick CharCNN too.

Kim Jong Il dials back h**-**s personality cult as protest activities pick up. **Sci/Tech**
Reuters- The Chicago Bear**J** are expected to sign quarter**h**ack Jeff George on Monday. **World**
mAE Systems says it is being investigated by the UK's Serious Fraud Office. **Sci/Tech**
Cornice blasts Seagate's suit over pa**p**ents for tiny hard drives used in portable gadg**r**ts. **World**

kinds of NMT (Belinkov & Bisk, 2018), we find that adversarial examples for text classification can barely trick other models. For this study, we use the CharCNN model in X. Zhang, Zhao, and LeCun (2015) as the target system. Their architecture is also based on character CNNs. But it has two major differences: 1) It does not use explicit segmentation and treats text as pure characters; and 2) It uses interleaved layers of convolutions and max pooling over the sequence of one-hot representation of characters.

For this experiment, we applied only flip changes, and transferred those examples that were successful in tricking the CharCNN-LSTM model. For random changes, we applied an equal number of character flips as in there are in the transferred examples. Table 8 shows some adversarial examples, where the adversary has learned to change the most discriminative parts of text. As is shown in Table 7, the adversarial examples perform better than random changes in hampering the classifier, but are barely effective overall.

Human Perception. Our human evaluation experiment shows that our character-based adversarial examples rarely alter the meaning of a sentence. We conduct an experiment of annotating 600 randomly-picked instances annotated by

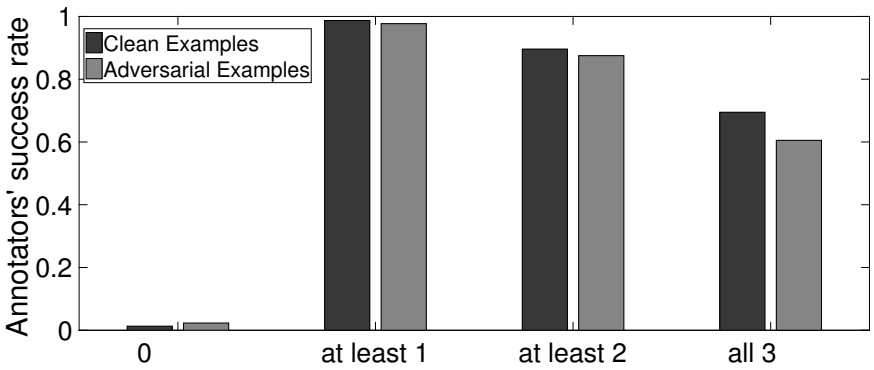


Figure 5. Performance of humans with Mechanical Turk experiments.

at least three crowd workers in Amazon Mechanical Turk. This set contains 150 examples of each class of AG's-news dataset, all of which are correctly classified by the classifier. We manipulate half of this set by our algorithm, which can successfully trick the classifier to misclassify these 300 adversarial examples.

The median accuracy of our participants decreased by 1.78%: from 87.49% on clean examples, to 85.71% on adversarial examples. Similar small drops in human performance have been reported for image classification (Papernot, McDaniel, Jha, et al., 2016) and text comprehension (Jia & Liang, 2017). Figure 5 shows a detailed distribution of annotators' accuracy for cases of 1, 2, and 3 correct answers.

Furthermore, an error analysis shows us that most of the errors made by annotators come from the inherent ambiguity in text, rather than from imposed obfuscation by adversarial manipulations. There were four examples, for which no annotator picked the correct label. It also happened that for these four examples, all annotators picked the same incorrect label. See Table 9. In the first two examples, the label picked by the annotators could be equally valid, given the fact that they are international news. The third example has little context information as a rationale to label it as Science/Technology, and because of the existence of the word 'game', annotators have picked Sports over other categories. The last one is an interesting

Table 9. Error analysis of human annotation.

Adversarial text	Annotators' label	Ground-truth
EU wants US to clear muddy waters in WTh row. Brussles-EU trade chief Peter Mandelson wants clarification of the US stance in threatened WTO action over aid to AiDbus, his spokeswoman said Friday after a US official indicated Washington was delaying such action.	World	Business
Is\ands press Govt to reverse pho\e call decision. Diplomats from a number of islands in the South Pacific are reported to be pressing the Government to reverse a decision to block all phome calls made to the islands.	World	Sci/Tech
This week in game news. They risked hypothermia and fought off the effects of sleep depriSation so they could be among the first to achieve their quest in the wee hours of the morfing.	Sports	Sci/Tech
Steal SpongeBob, buy a P\museul. We've got two more entries this week in the catrgory of "What weird, useless stuff is for sale on eBay that I just have to have?'t	Business	Sci/Tech

example, in which the text has been changed to hide enough information to mislead a reader. The phrase "*PC museum*" is changed to "*P\museul*", which leaves "*sale on eBay*" the only rationale to annotate the text based on. It is very likely that without this change, one or more annotators would have picked the correct label.

Conclusion

We address the problem of creating discrete white-box adversarial examples for NLP. We create white-box adversarial examples by computing derivatives with respect to a few character-edit operations (i.e., *flip*, *insert*, *delete*).We study adversarial attacks on a character-level neural text classifier, and machine translator, and contrasted some their properties. In particular, our text classifier was much more difficult to trick, and it had a lower transferability rate.

We find that our gradient-based method to estimate the loss is competitive with a model that queries the model for the loss of all perturbations. But our gradient-based adversary is orders of magnitude faster, which makes it a much better candidate for creating adversarial examples.

For applying multiple changes, a beam search-based adversary can achieve close to 100% misclassification rate for text classification, and with 90% success rate, decrease

the BLEU score by 1 point for machine translation. We observe that while character-edit operations have little impact on human understanding. However, we find that character-level models are highly sensitive to adversarial perturbations.

The current adversary performs untargeted attacks, and simply wants to change the model's output. In the next chapter, we will focus on neural machine translation, and investigate goal-based attacks. In addition, we will conduct a thorough investigation into differences between black-box and white-box adversaries.

CHAPTER IV

BLACK-BOX VS. WHITE-BOX ADVERSARIES FOR NLP

Ebrahimi, J., Lowd, D., & Dou, D. (2018). On adversarial examples for character-level neural machine translation. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 653-663).

Introduction

Previous work in NLP (Belinkov & Bisk, 2018; Jia & Liang, 2017; Zhao et al., 2018) has focused on creating adversarial examples in a black-box setting, wherein the attacker can query a model but does not have access to its parameters. Black-box attacks often rely on heuristic methods to create adversarial examples. In contrast, white-box attacks approximate the worst-case attack for a particular model and input, within some allowed set of perturbations. Therefore, white-box attacks can demonstrate and defend against a model's most serious vulnerabilities, which may not be discovered by black-box heuristics.

Belinkov and Bisk (Belinkov & Bisk, 2018) investigate the sensitivity of neural machine translation (NMT) to synthetic and *natural noise* containing common misspellings. They show state-of-the-art models are vulnerable to adversarial attacks even after a spell-checker is deployed. We further explore the space of adversarial examples for NMT; equipped with a white-box adversary, we can perform more interesting attacks. Concretely, we propose *controlled* and *targeted* adversaries which create adversarial examples with other goals, instead of merely decreasing the BLEU score. A controlled adversary aims to mute a word in the original translation, while a targeted adversary aims to push a word into it. Table 10 shows one example of each category. In both cases, the adversary, which has no word alignment model, has not drastically changed the rest of the translation, and has been able to reach its goals.

Table 10. Controlled and Targeted Attack on DE→EN NMT. In the first example, the adversary wants to suppress a person's name, and in the second example, the adversary wants to replace occurrences of *therapist* with *psychopath*

src	1901 wurde eine Frau namens Auguste in eine medizinische Anstalt in Frankfurt gebracht.
adv	1901 wurde eine Frau namens **Afuiguste** in eine medizinische Anstalt in Frankfurt gebracht.
src-output	In 1931, a woman named **Augustine** was brought into a medical institution in France.
adv-output	In 1931, a woman named Rutgers was brought into a medical institution in France.
src	Das ist Dr. Bob Childs – er ist Geigenbauer und Psychotherapeut.
adv	Das ist Dr. Bob Childs – er ist Geigenbauer und **Psy6hothearpeiut**.
src-output	This is Dr. Bob Childs – he's a wizard maker and a **therapist**'s **therapist**.
adv-output	This is Dr. Bob Childs – he's a brick maker and a **psychopath**.

We further study robustness of black-box-trained and white-box-trained NMT and text classification models to compare the two strategies more rigorously. In particular, we study robustness of model to unseen test data, as well as white-box and black-box adversarial examples.

Controlled and Targeted Attacks

Recall that for an explicitly-segmented character-level model, a character **flip** in the j-th character of the i-th word $(a \rightarrow b)$ can be represented by this vector:

$$\vec{v}_{ijb} = (\vec{0},..;(\vec{0},..(0,..\text{-}1,0,..,1,0)_j,..\vec{0})_i;\ \vec{0},..)$$

where -1 and 1 are in the corresponding positions for the a-th and b-th characters of the alphabet, respectively, and $x_{ij}^{(a)} = 1$.

A first-order approximation of change in loss can be obtained from a derivative along this vector:

$$\nabla_{\vec{v}_{ijb}} J(x,y) = \nabla_x J(x,y)^T \cdot \vec{v}_{ijb} = \frac{\partial J}{\partial x_{ij}}^{(b)} - \frac{\partial J}{\partial x_{ij}}^{(a)} \tag{4.1}$$

An immediate benefit of using derivatives with respect to the one-hot vectors is that they can be used to select the best character change $(a \rightarrow b)$. Concretely, the derivative vector contains the information about the loss increase obtained due to a character flip. This makes the number of queries (forward passes) be independent

of the alphabet size. We simply need to find the best change by **maximizing** eq. 4.1, to *estimate* the best character change ($a \rightarrow b$). This requires searching in $|V|mn$ values for a given text of m words with n characters each, in a vocabulary of size $|V|$.

Character **insertion** at the j-th position of the i-th word can also be treated as a character flip, followed by more flips as characters are shifted to the right until the end of the word. Note that for ease in exposition, we assume that the word size is at most n-1, leaving at least one position of padding at the end. Hence the estimate in change of loss corresponding to a character insertion can be given by:

$$\frac{\partial J}{\partial x_{ij}}^{(b)} - \frac{\partial J}{\partial x_{ij}}^{(a)} + \sum_{j'=j+1}^{n} \left(\frac{\partial J}{\partial x_{ij'}}^{(b')} - \frac{\partial J}{\partial x_{ij'}}^{(a')} \right) \tag{4.2}$$

where $x_{ij'}^{(a')} = 1$ and $x_{ij'-1}^{(b')} = 1$.

Similarly, character **deletion** can be written as a number of character flips as characters are shifted to the left. The best character deletion can be estimated by:

$$\max_{ij} \sum_{j'=j}^{n-1} \left(\frac{\partial J}{\partial x_{ij'}}^{(b')} - \frac{\partial J}{\partial x_{ij'}}^{(a')} \right) \tag{4.3}$$

where $x_{ij'}^{(a')} = 1$ and $x_{ij'+1}^{(b')} = 1$.

And finally, **swap** of two adjacent characters (ab) can be written as two flips. The best swap can be estimate by:

$$\max_{ij} \left(\frac{\partial J}{\partial x_{ij}}^{(b)} - \frac{\partial J}{\partial x_{ij}}^{(a)} + \frac{\partial J}{\partial x_{ij+1}}^{(a)} - \frac{\partial J}{\partial x_{ij+1}}^{(b)} \right) \tag{4.4}$$

An untargeted adversary's sole goal is to increase the loss of the model. However, some corruptions of the output may be much worse than others – translating "good morning" as "attack them" is much worse than translating it as "fish bicycle." By changing the loss function, we can force the adversary to focus on more specific goals.

In a *controlled attack*, the adversary tries to remove a specific word from the translation. This could be used to maintain privacy, by making more sensitive

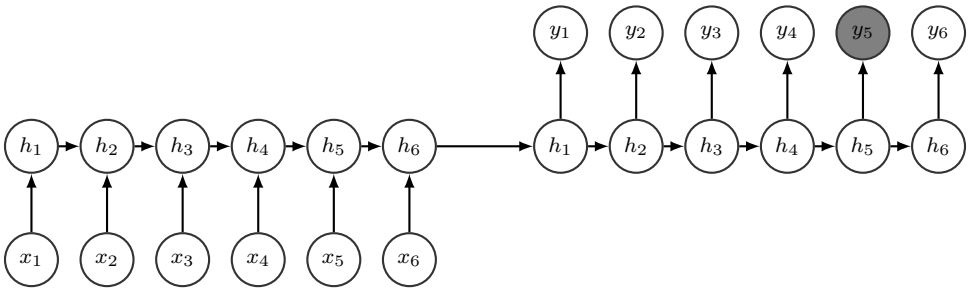


Figure 6. Illustration of a goal-based attack. The fifth word of the translation will be either removed or replaced. The loss over the rest of the words in the translation are not involved in creating adversarial examples.

information harder to translate, or to corrupt meaning, by removing key modifiers like "not," "joked,"' or "kidding." Concretely, we maximize the loss function $J(x, y_t)$, where t is the target word. This way, the adversary ignores the rest of the output and focuses on parts of the input that would affect the target word most.

In a *targeted attack*, the adversary aims to not only mute a word but also replace it with another. For example, changing the translation from "good morning"' to "good attack" could lead to an investigation or an arrest. Making specific changes like this is much more dangerous, but also harder for the adversary to do. For this attack, we maximize the loss $-J(x, y_{t'})$, where $t^{'}$ is the new word chosen to replace t. Note that the negation makes this equivalent to minimizing the predictive loss $J(x, y_{t'})$ on $t^{'}$. We represent this as maximization so that it fits in the same framework as the other attacks. Our derivative-based approach from the previous subsection can then be used directly to generate these new attacks, simply by substituting the alternate loss function. An illustration of goal-based attacks is given in Figure 6

Multiple Changes. We explained how to estimate the best single change in text to get the maximum increase/decrease in loss. We now discuss approaches to perform multiple changes.

(a) **One Shot:** In this type of attack, the adversary manipulates all the words in the text with the best operation in parallel. That is, the best operation for each word is picked locally and independently of other words. This is efficient as with only one forward and backward pass, we can collect the gradients for all operations for all words. It is less optimal than the next approaches, which apply changes one by one. Due to its efficiency, this is the approach we choose to do adversarial training. We create adversarial examples for adversarial training using this approach. We also investigate untargeted black-box and white-box attacks using one-shot attacks. The budget for the adversary is the number of words, and it is spent in the first shot.

(b) **Greedy:** In this type of attack, after picking the best operation in the whole text, we make another forward and backward pass, and continue our search. Our controlled adversary follows this approach, where we allow a maximum of 20% of the characters in text as the budget for the adversary. As will be explained, the adversary spends much less than this amount.

(c) **Beam Search**: And finally, we can strengthen our greedy search by beam search. Our beam search requires only $\mathcal{O}(br)$ forward passes and an equal number of backward passes, with r being the budget and b, the beam width. At every step, the beam will be sorted by the sum of the true loss up to that point, which we have computed, plus the gradient-based estimate of candidate operations. Since targeted attacks are the most difficult type of attack, we use this strategy for targeted attacks. We allow a maximum of 20% of the characters in text as the budget for the adversary, and set the beam width to 5.

Experiments

We use the TED talks parallel corpus prepared by IWSLT 2016 (Mauro et al., 2016) for three pairs of languages: German to English, Czech to English, and French to English. We use the development sets and test sets of previous years except 2015 as our development set. The statistics of the dataset can be found in Table 11. Throughout our experiments, we only allow character changes if the new word does not exist in the vocabulary, to avoid changes that an MT would respond to as expected. For example, it is not surprising that changing the source German word "nacht" to "nackt" would cause an MT to introduce the word "nude" in the translation. Our implementation[1] relies largely on Yoon Kim's seq2seq implementation[2], with similar hyper-parameters, which mostly follows the guidelines of Luong et al. (2015) for attentional translation.

For experiments in Section IV, where we report the BLEU score for a vanilla model and several adversarially trained models against different attackers, we use a decoder with a beam width of 4. However, our white-box attacker uses a model with greedy decoding to compute gradients. The reason is that in order to calculate correct gradients, we either need to use greedy decoding, or use models which incorporate beam search in the decoder architecture such that gradients could flow in the beam paths, too (Wiseman & Rush, 2016), which incur more computational cost. Correct gradients are more of an important issue for targeted attacks, where we want to achieve a goal beyond simply breaking the system. For the sake of consistency, we use greedy decoding for both the vanilla model which is being attacked, and the

[1]https://github.com/jebivid/adversarial-nmt

[2]https://github.com/harvardnlp/seq2seq-attn

Table 11. Dataset Statistics

Pair	Train	Test	Target vocab.
FR-EN	235K	1.1k	69k
DE-EN	210K	1.1k	66k
CS-EN	122K	1.1k	49k

white-box attacker, in all experiments of Section IV, where we contrast white-box and black-box adversaries in different scenarios.

Analysis of Adversaries. We first study whether first-order approximation gives us a good estimator to be employed by white-box adversaries. Figure 7 compares the true increase in log-loss (i.e., $J(x+v) - J(x)$), with our gradient-based estimate (i.e., $\nabla_v J(x)$). We create adversarial examples using the best estimated character flip for every word, over the German test set. Then, we compare the true increase in loss for the created adversarial examples, with our gradient-based estimate. The log-loss is evaluated by summing the log-loss of individual words, and similarly, the gradient-based estimate is the sum of all gradients given by flips which are performed once on every word. Figure 7.a plots the histograms for both of these measures, and Figure 7.b shows a scatter plot of them with the least squares fitting line. Due to linearization bias of the first-order approximation, we have a distribution with smaller variance for the gradient-based estimate measure. We can also observe a moderately positive correlation between the two measures (Spearman coefficient $\rho = 0.61$), which shows we can use the gradient-based estimate for *ranking* adversarial manipulations.

Next, we contrast black-box and white-box adversaries in untargeted, controlled, and targeted scenarios, and we demonstrate that white-box adversaries, significantly outperform black-box adversaries especially in controlled and targeted scenarios.

Untargeted Attack. Table 12 shows the BLEU score after one-shot white-box and black-box attacks are performed. Unlike delete and swap, insert and flip

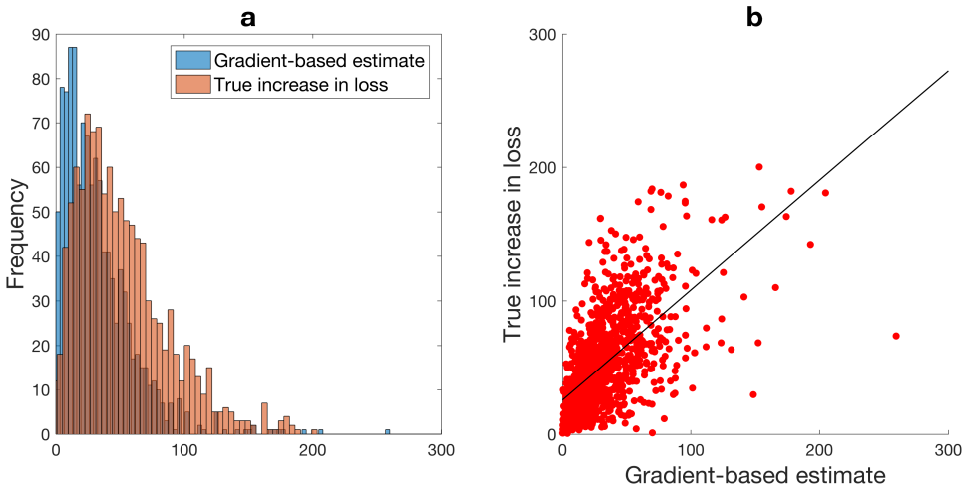


Figure 7. Comparing the distribution of the true increase in loss and its gradient-based estimate, and their correlation, using best flips for each word in a sentence of the German test set.

have the advantage of making changes to one-letter words, so we expect them to perform better. We see this for the white-box attacks which can pick the best change to every word using the gradients. On the contrary, a black-box adversary performs worst for flip, which is because the black-box attacker is not enabled to pick the best change when more options (possible character flips) are available, as opposed to swap and delete, which are governed by the location of the change and contain no additional flip. Nevertheless, a black-box adversary has competitive performance with the white-box one, even though it is simply randomly manipulating words. We argue that evaluating adversaries, based on their performance in an untargeted setting on a brittle system, such as NMT, is not appropriate, and instead suggest using goal-based attacks for evaluation.

Controlled Attack. We introduce more interesting attacks, in which the adversary targets the MT for more specific goals. A perfect *mute* attack removes a word successfully and keeps the rest of the sentence intact. For example, consider the translation T, containing words w_1, w_2,..., w_t, ...w_n, where w_t is the target word.

Table 12. BLEU score after greedy decoding in the existence of different types of untargeted attacks.

Attack	Flip		Insert		Delete		Swap	
	white	black	white	black	white	black	white	black
FR	**4.27**	6.98	**4.74**	4.85	**4.99**	5.86	**4.87**	5.20
DE	**4.50**	6.87	**3.91**	4.31	**5.63**	5.73	4.94	**4.74**
CS	**4.31**	6.09	**4.66**	5.86	**6.30**	6.62	6.05	**5.82**

A perfect mute attack will cause the NMT to create a translation T_p which contains words w_1, w_2,..., UNK, ...w_n, wherein w_t is replaced with UNK. With this observation in mind, we define the success rate of an attack, which generates T_{adv}, as follows:

$$\text{success}(T_{adv}) = \begin{cases} 1, & \text{if } \frac{\text{BLEU}(T, T_{adv})}{\text{BLEU}(T, T_p)} \geq \alpha \\ 0, & \text{otherwise} \end{cases}$$

We can control the quality of an attack with α, for which a larger value punishes the adversary for ad-hoc manipulations, which could cause the NMT to generate a radically different and possibly gibberish translation. Success rate is defined by the number of successful attacks divided by the number of total sentences in the test set.

Figure 8 plots the success rate against α. As can be seen, the white-box adversary is significantly more successful than a black-box adversary. By taking advantage of the knowledge of gradients of the model, a white-box adversary can perform better targeted attacks. For this experiment, the black-box attacker uniformly picks from the four possible changes and randomly applies them.

For this attack, we follow a greedy approach to apply multiple changes, i.e., applying the best manipulation at every step of the way, and use a budget of 20% of the characters in text. Table 13 shows the average number of character changes and the number of queries made to the model. The reported character changes are for attacks wherein the attacker only muted a target word successfully, regardless of

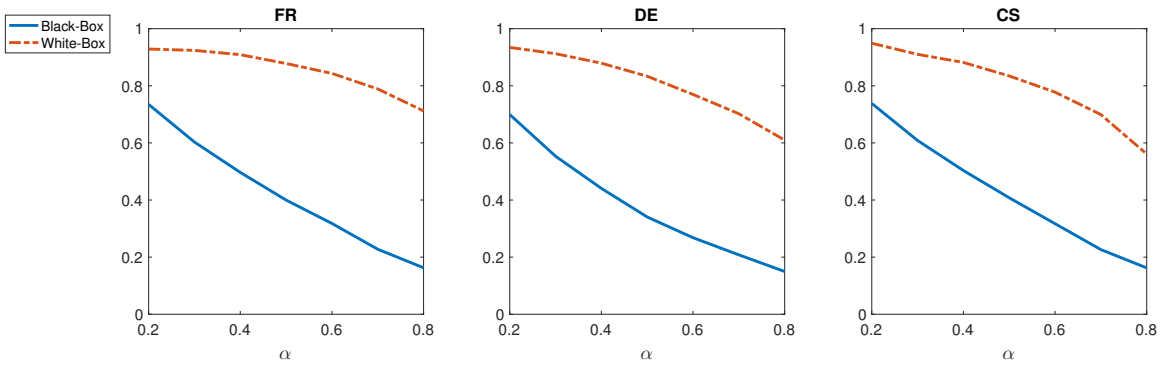


Figure 8. Success rate of white-box and black-box adversaries in a controlled setting as a function of α.

the quality of the translation. The reported number of queries takes the unsuccessful trials into account too. The white-box adversary is more efficient due to fewer queries and fewer manipulated characters, which can be crucial for a real-word adversary. Nevertheless, unlike a black-box adversary, a white-box adversary requires additional backward passes, and has the overhead of operations on the gradient values, mainly sorting. This makes the running times of the two comparable.

Compared with the results in the previous section, controlled attacks show a more convincing superiority of the white-box attacks over black-box attacks.

Table 13. Efficiency of attacks.

	Character Changes		Queries	
source	white	black	white	black
FR	**1.9**	7.7	**2.3k**	8.9k
DE	**1.9**	6.5	**1.9k**	7.8k
CS	**1.5**	5.3	**1.2k**	6.1k

Targeted Attack. A more challenging attack is to not only mute a word but also replace it with another one. The evaluation metric for a targeted attack is similar to a controlled attack with one difference: that a perfect *push* attack produces a translation, T_p, which contains words w_1, w_2, $w_{t'}$,..w_n, wherein $w_{t'}$ has replaced w_t. In classification domains with few classes, targeted attacks are relatively simple,

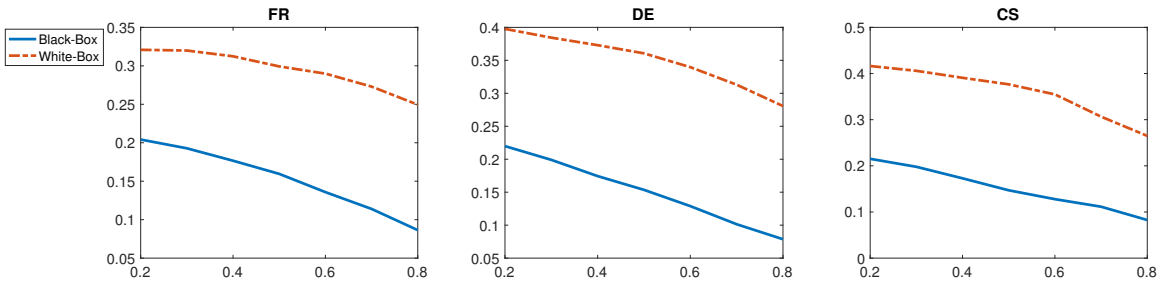


Figure 9. Success rate of white-box and black-box adversaries in the second-most-likely targeted attack as a function of α.

since an adversary can perturb the input to move it to the other side of a decision boundary; whereas in MT, we deal with vocabulary sizes in the order of at least tens of thousands, and it is less likely for an adversary to be successful in targeted attacks for most possible target words. To address this, we evaluate our adversary with n_{th}-most likely class attacks. In the simplest case, we replace a target word with the second-most-likely word at decoding time. Our experiments in this section use a beam-search strategy for applying multiple changes, i.e., keeping the most promising manipulations at every step and performing a beam search.

As can be seen in Table 9, targeted attacks are much more difficult with a much lower success rate for the adversary. Nevertheless, the white-box adversary still performs significantly better than the black-box adversary. The success rate dramatically goes down for large values of n. For example, for the value of 100, the success rate will be more than ten times smaller than the second-most-likely attack.

Some Adversarial Examples. Table 14 shows some adversarial examples. The first example shows a controlled attack, where the adversary has successfully removed a swear word from the sentence. The BLEU ratio, used in our success rate measure, for this example is 0.52. The second example shows a second-most-likely targeted attack, where the new translation has managed to keep the rest of

Table 14. A controlled attack and seven targeted attacks on our DE-EN NMT. The first example shows a controlled attack; the rest show a second-most-likely, except for the last one which shows 100_{th}-most likely targeted attack.

src	Wir erwarten Perfektion von Feministinnen, weil wir immer noch für so viel kämpfen, so viel wollen, so verdammt viel brauchen.
adv	Wir erwarten Perfektion von Feministinnen, weil wir immer noch für so viel kämpfen, so viel wollen, so **öberdammt** viel brauchen.
src-output	We expect perfection from feminist, because we're still fighting so much, so **damn**, so **damn**, so **damn**.
adv-output	We expect perfection from feminist, because we still fight for so much, so much of all, so much of all that needs to be.
src	In den letzten Jahren hat sie sich zu einer sichtbaren Feministin entwickelt.
adv	In den letzten Jahren hat sie sich zu einer sichtbaren **FbeminisMin** entwickelt.
src-output	In the last few years, they've evolved to a safe **feminist**.
adv-output	In the last few years, they've evolved to a safe **ruin**.
src	Für Leute wie Sie ist kein Platz in diesem System.
adv	Für Leute wie Sie ist kein Platz in diesem **Sysetm**.
src-output	For people like you, there's no place in this **system**.
adv-output	For people like you, there's no place in this **scheme**.
src	In der Woche vor der Veröffentlichung war ich sehr nervös.
adv	In der Woche vor der Verösffentlichung war ich sehr **nOrjvöss**.
src-output	In the week before the published, I was very **nervous**.
adv-output	In the week before the published, I was very **nerdy**.
src	Im 20. Jahrhundert galt Autismus lange als ein sehr seltenes Leiden.
adv	Im 20. Jahrhundert **bgalt** Autismus lange als ein sehr **zwelenems** Leiden.
src-output	In the 20th century, autism was a very rare **suffering**.
adv-output	In the 20th century, autism is a very, very, very, very **common** suffering.
src	Das Füllen dieser Lücke muss der Kern jedes nachhaltigen Ansatzes sein.
adv	Das Füllen dieser **Zrcke** muss der Kern jedes nachhaltigen Ansatzes sein.
src-output	The filling of this **gap** has to be the core of every sustainable approach.
adv-output	The filling of this **bridge** has to be the core of every sustainable approach.
src	1901 wurde eine Frau namens Auguste in eine medizinische Anstalt in Frankfurt gebracht.
adv	1901 wurde eine Frau namens **Afuiguste** in eine medizinische Anstalt in Frankfurt gebracht.
src-output	In 1931, a woman named **Aususte** was brought into a medical institution in France.
adv-output	In 1931, a woman named **oyster** was brought into a medical institution in France.
src	Ein Krieg ist nicht länger ein Wettbewerb zwischen Staaten, so wie es früher war.
adv	Ein Krieg ist nicht länger ein **erkBkaSzeKLlWmrt** zwischen Staaten, so wie es früher war.
src-output	A war is no longer a **competition** between states, like it used to be.
adv-output	A war is no longer a **throwaway** planet between states, as it used to be.

the translation intact and achieve its goal. The BLEU ratio for this example is 1.00. The last example, which has a BLEU ratio of 0.70, shows a 100th-most likely attack, where the word *competition* is replaced with *throwaway*. Due to the difficulty of this change, the adversary has committed a considerably larger number of manipulations.

Given these examples, we can posit that due to the attention mechanism, the model learns to align words in the source sequence to the target sequence. So changes in one word can often only change one word in the translation. And because of the way the BLEU evaluation metric is designed, one adversarial change does not necessarily render a drastically different score; thus controlled or targeted adversaries can potentially be successful.

Robustness to Adversarial Examples

Adversarial Training. Adversarial training interleaves training with generation of adversarial examples (I. J. Goodfellow et al., 2015). Concretely, after every iteration of training, adversarial examples are created and added to the mini-batches. Jia and Liang (2017) point out the difficulty of adversarial training with real-world adversarial examples, as it is not easy to create such examples efficiently. Our inner adversary manipulates all the words in the text with the best operation in parallel. That is, the best operation for each word is picked locally and independently of other words. This is efficient, as with only one forward and backward pass, we can collect the gradients for all operations for all words. It is less optimal than the greedy or beam-search methods, which apply changes one by one. Due to its efficiency, this is the approach we choose to do adversarial training.

Machine Translation. We use the black-box training method of Belinkov and Bisk (2018) as our baseline. They train several models using inputs which

include noise from different sources. We used their script[3] to generate random (`Rand`), keyboard (`Key`), and natural (`Nat`) noises. Their best model was one which incorporated noise from all three sources (`Rand+Key+Nat`).

Similar to their approach, we train a model which incorporates noisy input scrambled by Flips, Inserts, Deletes, and Swaps in training (`FIDS-B`). Since natural noise, which is a set of common typos and misspellings collected by linguists, was shown to be the most elusive adversarial manipulation (Belinkov & Bisk, 2018), we used this source of noise to determine the proportion of each of the FIDS operations in training. Concretely, we found that the majority of the natural noise can be generated by FIDS operations, and we used the ratio of each noise in the corpora to sample from these four operations. Figure 10 shows the distribution of manipulations for each language. A single swap is the least likely operation in all three languages, excluding single swaps from manipulations with two flips. FIDS operations account for 64%, 80%, and 70% of natural noise for Czech, German, and French, respectively. This can be regarded as a background knowledge incorporated by our inner adversary.

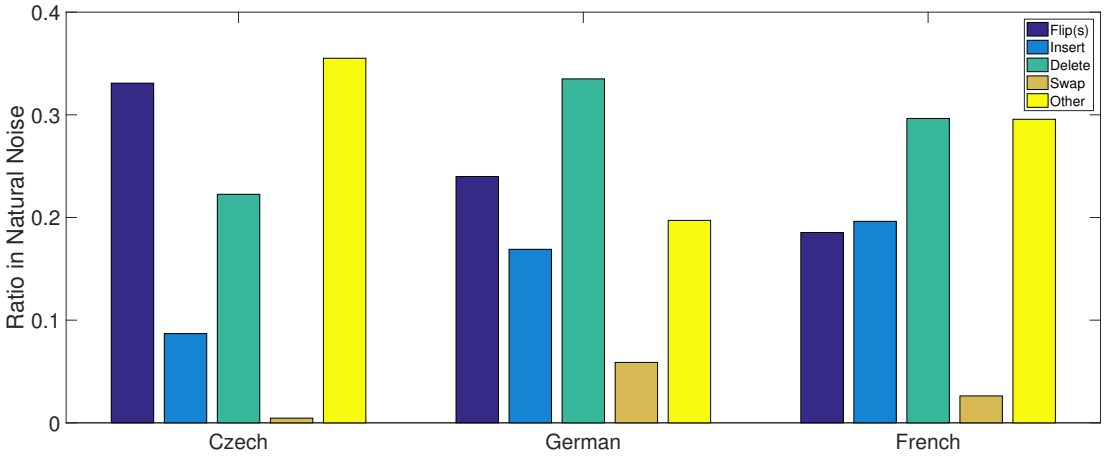


Figure 10. Distribution of types of noise in the natural noise corpora.

[3]https://github.com/ybisk/charNMT-noise

Table 15. BLEU score of models on clean and adversarial examples, using a decoder with beam size of 4. The best result on each test set is shown in bold. FIDS-W performs best on all noisy test sets, compared with models which have not been trained on that particular noise (shown in red). FIDS-B performs best on white-box adversarial examples compared with other black-box trained models (shown in blue).

Training \ Test		Clean	Nat	Key	Rand	FIDS-B	FIDS-W	Avg.
French	Vanilla	37.54	19.17	12.12	4.75	6.85	5.36	15.95
	Nat	26.35	**33.23**	11.16	5.32	8.28	6.65	15.16
	Key	33.02	17.30	**35.97**	4.63	7.00	5.17	17.17
	Rand	36.06	18.54	8.31	**36.10**	8.76	7.14	19.14
	FIDS-B	34.48	21.59	28.48	6.82	32.62	13.60	22.92
	FIDS-W	37.15	23.65	31.18	7.78	**32.72**	**31.94**	27.40
	Rand+Key+Nat	34.55	30.74	32.82	34.01	12.05	7.08	25.20
	Ensemble	**37.81**	30.27	29.36	34.42	32.00	30.01	**32.30**
German	Vanilla	31.81	17.24	10.36	4.20	6.78	5.50	12.64
	Nat	24.89	**32.14**	10.22	4.61	7.53	5.99	14.23
	Key	27.20	15.98	**30.62**	4.64	7.68	4.74	15.13
	Rand	31.01	17.90	6.59	**30.70**	9.19	5.83	16.86
	FIDS-B	28.27	20.22	23.84	6.29	27.35	10.79	19.45
	FIDS-W	**31.81**	21.72	26.23	7.75	**27.38**	**26.51**	23.56
	Rand+Key+Nat	29.22	29.78	27.83	28.88	10.30	6.14	22.01
	Ensemble	31.54	31.11	23.91	28.95	26.38	25.06	**27.82**
Czech	Vanilla	**26.44**	13.55	9.49	4.78	7.30	5.93	11.24
	Nat	18.73	**23.06**	9.07	4.45	7.36	5.42	11.34
	Key	22.76	13.09	**23.79**	4.83	7.93	5.82	13.03
	Rand	24.23	12.00	7.26	**24.53**	7.24	5.47	13.45
	FIDS-B	22.31	14.15	17.91	6.48	19.67	8.60	14.84
	FIDS-W	25.53	15.57	19.74	7.18	**20.02**	**19.42**	17.90
	Rand+Key+Nat	22.21	20.59	20.60	21.33	10.06	5.89	16.77
	Ensemble	25.45	20.46	17.15	21.39	18.52	17.03	**19.99**

Our white-box adversary, FIDS-W, generates adversarial examples using our four text edit operations in accordance with the distribution of operations on the natural noise. For adversarial examples, all words in the sentence are changed by a single FIDS operation in parallel. While training on both clean and adversarial examples has been the standard approach in adversarial training, some works (Madry et al., 2018; Shaham, Yamada, & Negahban, 2015) suggest that training on white-box adversarial examples alone can boost models' robustness to adversarial examples, with a minor

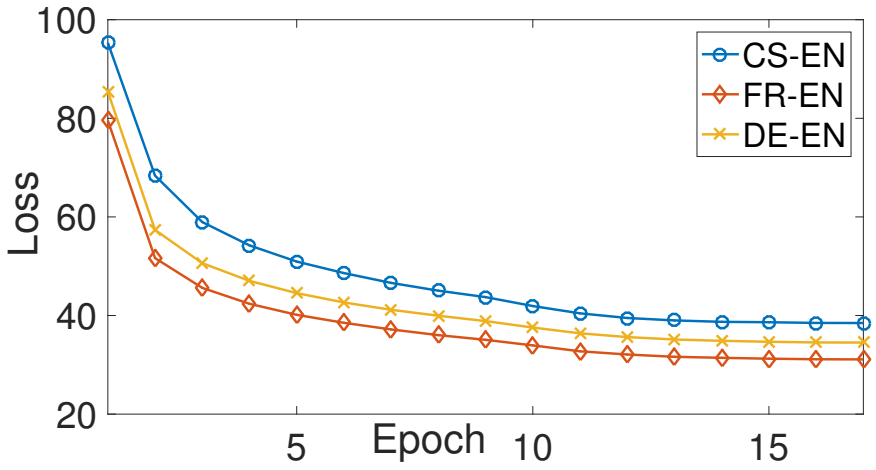


Figure 11. Training loss on adversarial examples for `FIDS-W`.

decrease in accuracy on clean examples. However, we found that in order to get a good BLEU score on the clean dataset, we need to train on both clean and white-box adversarial examples.

We also train an ensemble model, `Ensemble`, which incorporates white-box and black-box adversarial examples, with 50/50 share for each. The black-box adversarial examples come from `Nat` and `Rand` sources.

When evaluating models against `White` adversarial examples at test-time, we use the test set which corresponds to their method of training. For instance, the `White` adversarial examples for the `Rand` model, come from the test set which has manipulated clean examples by `Rand` noise first. For `Vanilla`, `FIDS-W`, and `Ensemble` models, the adversarial examples are generated from clean data. This makes the comparison of models, which are trained on different types of data, fair.

Discussion. Table 15 shows the results for all models on all types of test data. Overall, our ensemble approach performs the best by a wide margin. As expected, adversarially trained models usually perform best on the type of noise they have seen during training. However, we can notice that our `FIDS-W` model performs best on

the Nat noise, amongst models which have not been trained on this type of noise. Similarly, while FIDS-W has not directly been trained on Key noise, it is trained on a more general type of noise, particularly flip, and thus can perform significantly better on the Key than on other models which also have not been trained on this type of noise. A more detailed depiction of this phenomenon can be found in Figure 12, wherein we compare BLEU score on Nat and Key test sets using methods which do not use either type of noise in training.

This phenomenon cannot generalize to Rand, which is an extreme case of attack, and we need to use an ensemble approach to perform well on it too. Nevertheless, FIDS-W performs best on the Rand noise, compared with models which are not trained on Rand either. This validates that training on white-box adversarial examples, which are harder adversarial examples, can make the model more robust to weaker types of noise. We also observe that FIDS-B performs better on the White examples compared with other baselines; although it has not been trained on white-box adversarial examples, but is trained on black-box adversarial examples of the same family of FIDS operations.

Figure 11 shows the training loss of the white-box adversarially trained model on adversarial examples. The model is getting more resilient to adversarial examples, which are created at the start of each epoch.

Text Classification. We also perform an experiment on the robustness of character-level text classification using the same architecture (CharCNN-LSTM), and the same dataset (AG's news) as in the previous chapter. For our adversarial training, we use only use the flip operation, and we evaluate models' robustness to this operation only. We flip r characters for each training sample, which was set to 20% of the characters in text after tuning, based on the accuracy of the model on

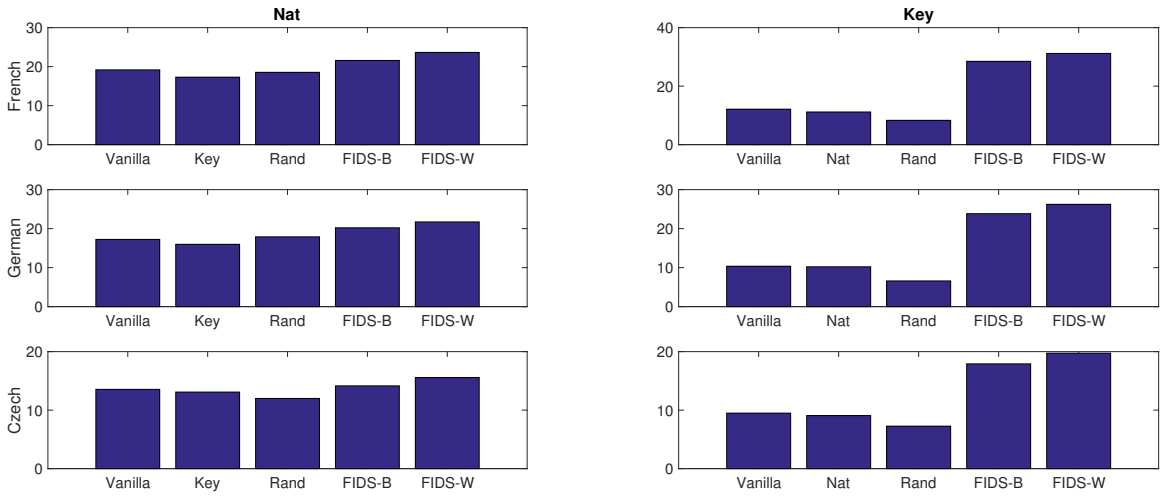


Figure 12. BLEU score on Nat and Key datasets using methods which do not use either type of noise in training.

the development set. We find that our model is able to decrease misclassification error from 8.27 to 7.65, and is much more robust to adversarial attacks at test-time, lowering adversary's success rate from 98.16 to 69.32.

We compare our white-box adversarial training with the white-box supervised adversarial training of (Miyato et al., 2017) that perturbs word embeddings, which we adapt to work with character embeddings. Specifically, the adversarial noise per character is constrained by the Frobenius norm of the embedding matrix composed of the sequence of characters in the word. We also create another baseline, where instead of white-box adversarial examples, we add black-box adversarial examples (`Key`*) to the mini-batches. `Key`* is a stronger variant of `Key`, since it replaces a character with any character in the alphabet, and is not limited to keyboard-based changes. But this black-box model was very ineffective compared to the white-box models.

Figure 13 shows the average number of flips with respect to the word length of the document, as a simple measure of difficulty of the job for the adversary. It shows that an adversarially-trained model is more resilient; the adversary has a more difficult

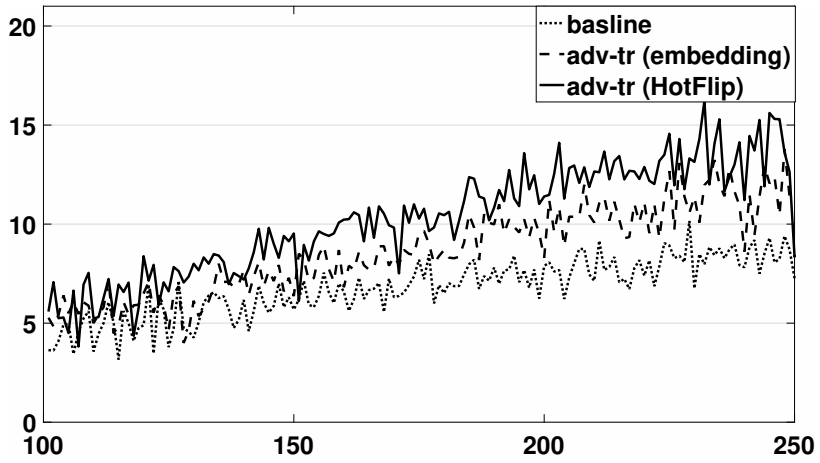


Figure 13. Average number of flips committed by the adversary to attack each model, as a function of document length.

time in tricking it, as it performs more character flip operations. These results suggest that while there is proof that our method is doing well in improving test accuracy, its superiority is much more pronounced in explicitly adversarial settings.

The current error of our adversarially trained model (i.e., adversary's success rate at test-time is 69.32%) is still beyond an acceptable rate; this is mainly because the adversary that we use at test time, which uses beam search, is strictly stronger than our model's internal adversary which uses a one-shot approach. This has been observed in computer vision, where strongest adversaries are not efficient enough for adversarial training, but can break models trained with weaker adversaries (Carlini & Wagner, 2017b).

Conclusion

As MT methods become more effective, more people trust and rely on their translations. This makes the remaining limitations of MT even more critical. Previous work showed that NMT performs poorly in the presence of random noise, and that its performance can be improved through adversarial training. We consider stronger adversaries which are attacking a specific model and may also have specific

goals, such as removing or changing words. Our white-box optimization, targeted attacks, and new evaluation methods are a step towards understanding and fixing the vulnerabilities in NMT: We are able to find more effective attacks and train more robust models than previous black-box methods.

We show that for both classification and translations, employing these adversarial examples in adversarial training renders the models more robust to such attacks, as well as increasing model's performance on unseen clean data. In the next chapter, we will use similar machinery, discussed so far, to attack graph neural nets

CHAPTER V

ADVERSARIAL EXAMPLES FOR GRAPHS

Introduction

In this chapter, we will describe an efficient method to generate white-box adversarial examples for relational data. The method is similar to HotFlip as it tries to estimate the increase in loss by first-order approximation of loss given over manipulated input, e.g., adding or removing an edge. We will provide adversarial examples for transductive and inductive node classifiers, and a recommender system, which poses the problem as link prediction (Berg, Kipf, & Welling, 2017). We will also briefly discuss transferability of adversarial examples.

Method

Similar to NLP models, graph-based models are also vulnerable to perturbations; noisy edges or noisy nodes features could lead to poor performance. Historically, security of relational models has been an important subject of study. Most notably, spammers engage in link farming to get higher ranks in social media or search engines (Ghosh et al., 2012). Another notable example of attacks on graph-based models is Sybil attacks (Douceur, 2002), in which a malicious user creates fake accounts to increase the power of a single user (Yang et al., 2014). But similar to the previous section, studying the robustness of graph-based models, through the lens of adversarial examples, aims to improve models' generalizability to perform better on unseen data, whether adversarial or not. In that sense, the work of Torkamani and Lowd (2013) stands out, in which they study robustness of relational models in the context of max-margin structured classification.

We study adversarial attacks on a graph-based neural nets in transductive and inductive settings. In the former, the model has seen unlabeled test nodes during

training, while the latter considers the case where test nodes are removed from the training process. Our transductive experiments are done on Graph Convolutional Neural Network (Kipf & Welling, 2017), while our inductive experiments are done on the mini-batch analogue of GCN, GraphSage (Hamilton et al., 2017). The two different views of the graph lead to slightly different adversaries. In the transductive case, the adversary manipulates the graph considering global information, i.e., ranking all possible adversarial manipulations on the graph, while the inductive case can treat test nodes independently of each other.

Transductive Case. We can see that there are two security loopholes for a graph-based model: the trustworthiness of edges, and the trustworthiness of features of the neighboring nodes. We show how to perturb A and a $|V| \times m$ (binary) feature matrix, X, to attack a GCN. For our experiments, we consider cases when the adjacency is row-normalized, i.e. $\hat{A}_{ij} = \frac{A_{ij}}{|\mathcal{N}(i)|}$

Imagine the adversary is allowed to add one edge or one feature in the graph-based data to fool the model to which it has access. Using a brute-force search, it would need to make $\mathcal{O}(|V|^2)$ forward passes in the former case, and $\mathcal{O}(m|V|)$ in the latter, to exhaust the search space. That is, query the classifier (i.e., by calling $J(.)$, the negative log-likelihood loss function, to see if loss is increased), for all possible edge-based or feature-based changes.

Similar to HotFlip for NLP models, our algorithm uses derivatives as a surrogate loss, and it requires a single forward and a single backward pass to estimate the best change. We represent adversarial operations as vectors/matrices in the input space and estimate the change in loss by directional derivatives with respect to these operations. Based on this derivative-based surrogate loss, the adversary can choose the best loss-increasing direction.

An edge addition between the i-th and the j-th nodes can be represented by a one-hot matrix M_{ij}, where $M_{ij} = 1$ and M contains zero at all other poistions. Due to directional derivatives, we have the following:

$$\nabla_{M_{ij}} J(A, X, Y) = \nabla_A J(A, X, Y)^T \cdot M_{ij}$$

The edge with the steepest direction can be easily found:

$$\max \nabla_A J(A, X, Y)^T \cdot M_{ij} = \max_{ij \notin E} \frac{1}{|\mathcal{N}(i)|} \frac{\partial J}{\partial A_{ij}}$$

Given $M_{ij} = -1$ in the one-hot matrix M, the best edge to remove can be given by:

$$\max_{ij \in E} \frac{1}{|\mathcal{N}(i)|} \frac{-\partial J}{\partial A_{ij}}$$

And similarly, we can find the best edge-replacement ($A_{ij} \to A_{ik}$) by maximizing

$$\max_{ij \in E, ik \notin E} \frac{1}{|\mathcal{N}(i)|} \left(\frac{\partial J}{\partial A_{ik}} - \frac{\partial J}{\partial A_{ij}}\right) \tag{5.1}$$

For the multi-relation type GCN, adjacency matrices are indexed by the type of the relation. Thus an additional operation for this model is to change one relation to another, which can be given by:

$$\max_{ij \notin E^{r_n}, ij \in E^{r_m}} \frac{1}{|\mathcal{N}^n(i)|} \frac{\partial J}{\partial A_{ij}^{r_n}} - \frac{1}{|\mathcal{N}^m(i)|} \frac{\partial J}{\partial A_{ij}^{r_m}} \tag{5.2}$$

where r_n denotes the current relation and r_m is the new candidate relation. For example, in a recommender system in which different adjacency matrices are used for different ratings, this means changing the rating from n to m. And finally, adding or removing binary features j, such as a word in a bag-of-words representation of a document i, can be given by

$$\max_{X_{ij}=0} \frac{\partial J}{\partial X_{ij}}$$

and

$$\max_{X_{ij}=1} \frac{-\partial J}{\partial X_{ij}}$$

Maximal Independent Adversarial Set (MIAS). The adversary can simply start from the best operation and continue greedily until it runs out of the budget. That is, simply one forward and one backward pass suffice for the adversary to make decisions. However, we find that, when performing edge-based perturbation, the adversary might get stuck manipulating one particular node because it has large gradients, without actually increasing the total loss. This could happen for highly sensitive low-degree nodes, for which any edge addition could have a big gradient. To avoid these degenerate cases, we propose a graph-based approach to maximize the total loss: We find the *maximal independent edge set* of the matrix derivative. We can view this approach as a mechanism to collectively, rather than individually, generate adversarial edges. There are polynomial-time algorithms for finding the exact edge cover (Garey & Johnson, 2002), but we simply use its greedy version: Find the maximum-weight edge, remove itself and its neighboring nodes from the candidate nodes, and repeat this process. After the graph is covered, we query the classifier again, to get a new gradient matrix, on which we perform MIAS for a few more iterations.

Inductive Case. Attacking the GraphSage model for the inductive case is similar to the GCN attacks. However, we have to use the one-hot encoding to compute gradients. In addition, each test instance is considered independent of other nodes in the test set. Using the one-hot encoding, the first layer in GraphSage with a mean aggregator can be written as:

$$H_i = W_2\Big[\overbrace{\mathrm{ReLU}(W_1[x_i; \alpha \sum_{j\sim\mathcal{N}(i)} a_j X])}^{h_i^1}; \alpha \sum_j \overbrace{\mathrm{ReLU}(W_1[a_j X; \beta \sum_{k\sim\mathcal{N}(j)} a_k X])}^{h_j^1}\Big]$$

$$\frac{\partial J}{\partial a_j} = \frac{\partial J}{\partial h_i^1} \times \frac{\partial h_i^1}{\partial a_j} + \frac{\partial J}{\partial h_j^1} \times \frac{\partial h_j^1}{\partial a_j} \tag{5.3}$$

where a_i is an n-dimensional one-hot vector, n is the number of nodes in the graph, and α and β are normalization constants for the mean aggregator. Since a_j is a one-hot vector, we have the following:

$$\frac{\partial h_j^1}{\partial a_j} = a_j \frac{\partial h_j^1}{\partial X} X^T \tag{5.4}$$

As an example, for each edge a_{ij}, the worst case perturbation (replacement of j with k) can be approximated by:

$$\max_{k,j} \frac{\partial J}{\partial a_{jk}} - \frac{\partial J}{\partial a_{jj}}$$

Note that compared with the transductive edge-replacement in 5.1, which requires searching among $|V|E$ possibilities, adversarial edge perturbations for all test nodes in the inductive case has $b\mathcal{N}(i)|V|$ possibilities where b is the number of nodes in the test set. The feature-based perturbations are similar to the transductive case.

Experiments

In this section, we demonstrate our empirical results on two transductive tasks and one inductive task. For the transductive (i.e., using GCN) experiments, we apply dropout to inputs of all linear layers of the GCN. We use a grid search on the validation set, using the dropout probability, $p \in \{0.4, 0.5, 0.6\}$ and the ℓ_2 regularization parameter, $\lambda \in \{$1e-3, 5e-4, 1e-4, 5e-5, 1e-6$\}$, on the weights of the first layer of the GCN. We use a 2-layer GCN, and we set the number of hidden units to 16. For our inductive tasks, we use a hidden dimension of 128 units in both layers, and we have an additional fully-connected layer which outputs the class distribution. We use Adam optimizer (Kingma & Ba, 2015).

Transductive Classification. To evaluate edge-level perturbations, we perform experiments on three two-class political blog datasets, wherein the nodes have no features, and X is the identity matrix. The statistics of the datasets can

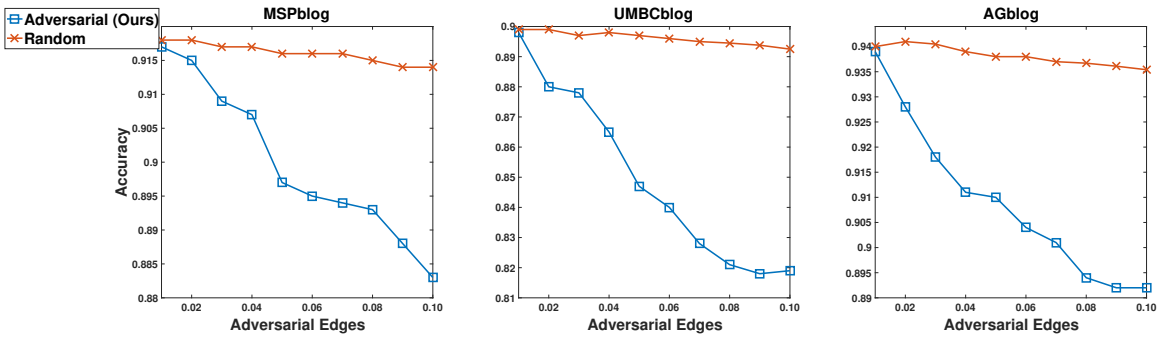


Figure 14. Decrease of accuracy under adversarial edge perturbation.

be found in Table 16. AGBlog dataset is the largest connected component from the political blog dataset (Adamic & Glance, 2005). The nodes in UMBCBlog are a subset of AGBlog; the links in the UMBCBlog dataset were gathered in May 2006, whereas in the AGBlog dataset, the links are from two months before the 2004 presidential election (Adamic & Glance, 2005). The UMBCBlog links reflect the blogger's interests at the time of the post, while links from the AGBlog dataset can be considered more descriptive on their long-term interests (Lin & Cohen, 2010). The MSPBlog dataset is provided by the researchers at Microsoft Live Labs[1]. For these datasets, we perform 10-fold cross validation, and report the average score.

Figure 14 demonstrates the accuracy of the models as adversarial edges are gradually being added. Recall that the UMBCBlog dataset composed of a subset of nodes from the AGBlog dataset, but contained more *transitory* edges. Thus, the dataset is more sensitive, and the GCN is easier to trick, which is demonstrated by the steep decrease in the accuracy of the GCN. Note that random perturbations have almost no impact in the models' accuracy.

For feature-based perturbations, we use two citation network datasets: Citeseer and Cora (Sen et al., 2008). These datasets contain bag-of-words term vectors for each

[1] `http://www.cs.cmu.edu/~frank/`

Table 16. Dataset statistics

Dataset	UMBC	AG	MSP
Nodes	404	1,222	4,324
Edges	3,382	16,714	18,627
Density	0.0292	0.0224	0.0019

Table 17. Datasets statistics

Dataset	Cora	Citeseer
Nodes	2,708	3,327
Edges	5,429	4,732
Classes	7	6
Vocab	1,433	3,703
Avg. words	18	31

paper and a list of citation links between them. Following (Kipf & Welling, 2017), we treat citation links as (undirected) edges, and we construct a binary, symmetric adjacency matrix A. The statistics of the datasets can be found in Table 17.

As shown in Table 18, the adversary, which simply changes one feature of some of the test nodes (i.e., either removing or adding a word), is able to decrease the accuracy substantially.

Table 18. Accuracy of the models with one noisy feature per test instance.

Impacted nodes	100	200	300	400	500	600	700	800	900	1000
Citeseer	68.47	66.67	65.57	64.61	63.61	62.73	62.28	61.62	60.75	60.23
Cora	79.11	77.36	75.31	73.2	71.12	69.32	67.31	65.7	64.25	62.81

Inductive Classification. We use three medium-to-large datasets for our inductive experiments. We use the Protein-Protein Interaction (PPI) dataset (Zitnik & Leskovec, 2017) for classifying protein functions across various biological protein-protein interaction (PPI) graphs. The dataset contains 20 graphs for training, 2 for validation and 2 for testing, which are not connected to each other. Each node has 50 features on positional gene sets, motif gene sets and immunological signatures (Velickovic et al., 2018). In addition, this task is a multi-label classification task,

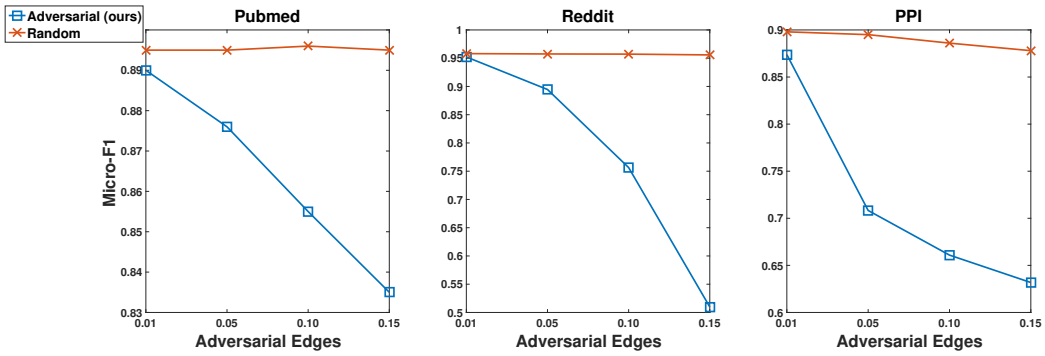


Figure 15. Micro-F1 score under adversarial edge perturbation.

Table 19. Dataset statistics

Dataset	PPI	Reddit	Pubmed
Nodes	56.9K	233.0K	19.7K
Edges	806.2K	114.6M	44.3K
Features	50	602	500
Classes	121	41	3

where each node can belong to any number of 121 existing labels. Thus, while for other datasets, we use a negative log-likelihood loss, for this dataset we use a binary logistic loss on all of the outputs.

Reddit is a website for discussions, in which users create "posts," which are delegated into user-created channels, known as "subreddits", which are organized by topic (e.g., news, food, books). Hamilton et al. (2017) sampled 50 large subjects and built a post-to-post graph, connecting posts if the same user comments on both. Features include average embedding of the post title and average embedding of all the post's comments. The embeddings come from 600-dimensional GloVe word vectors (Pennington, Socher, & Manning, 2014). In addition, features include the post's score and the number of comments made on the post, constituting a total of 602 features.

We also use a standard citation network, **Pubmed** (Sen et al., 2008), which has been extensively used as a benchmark for node classification. There are 500

input features which are bag-of-words representations of scientific articles from three biomedical categories.

Figure 15 demonstrates the micro-F1 score of models as certain proportion of edges are being adversarially replaced. As can be seen, similar to GCN, a vanilla GraphSage model is brittle to adversarial structural changes, while being robust to random changes.

Attacks on a Recommender System. In this section, we give a motivating example of adversarial analysis of graph-based models, which may also be popular to use in the real world. We use a GCN for creating adversarial edges, and we demonstrate that these examples can be transferred to hamper other models. We will study adversarial links (ratings) between users and items, and we use the ML-100k dataset of MovieLense (Harper & Konstan, 2016), which contains 100k ratings of 943 users over 1,682 movies. We create these adversarial ratings using the GCN-based collaborative filtering model (Berg et al., 2017), and transfer them to two other models, SVD++ (Koren, 2008) and item-based KNN (Koren, 2010). For the hyper-parameters of SVD++ and item-based KNN, we consulted (Gantner, Rendle, Freudenthaler, & Schmidt-Thieme, 2011), and used the implementation of (Hug, 2017) for these two models. SVD++ incorporates implicit feedback information, such as rating history, which can help understand users' preferences, especially for new users. Intuitively, users implicitly tell their preference by giving ratings, regardless of the rating score they give. The item-based KNN approach simply uses the similarity of the items, based on the ratings they have received, to infer unknown ratings.

Here, the adversary is composed of 20 spammers (around 2% of the users in the dataset), whose goal is to hamper the system in its prediction of a set of given pairs of user-movie ratings, i.e., the test set. The adversary computes the loss of

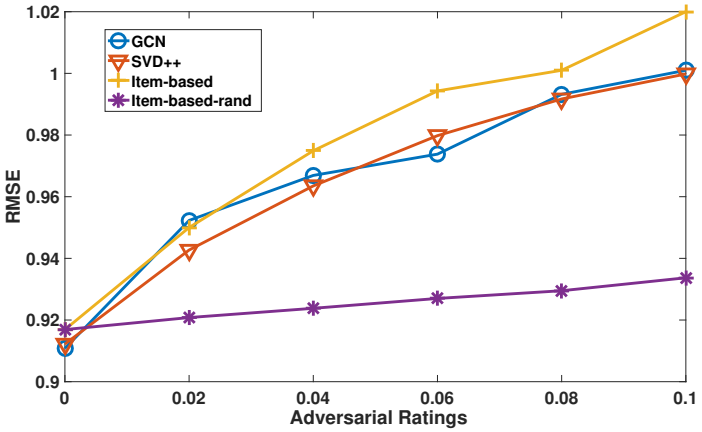


Figure 16. Adversary's success in increasing the RMSE error for three collaborative filtering models. The x axis shows the percentage of manipulated ratings.

the model given test queries, and manipulates links not in the test set. The users engaged in the attack manipulate their ratings by adding new ratings, changing their ratings, and removing their ratings. The new updated set of ratings, which contains a fraction of maliciously created links, are used to train the collaborative filtering models. Note that this type of attack is different from the previous tasks, which performed manipulations at test time. This is because link prediction is simply a process of querying the trained model for its estimate of a link between two entities. Thus, for this task, we need to manipulate the training set, which is called a *data poisoning* attack.

Figure 16 plots the root mean squared error (RMSE) error for all three models under varying adversarial ratings. As is demonstrated, 20 users, collectively attacking a system, could greatly increase the RMSE error for all models. To put the inflicted prediction error into perspective, the figure also shows how random rating changes can impact the item-based model. We omitted the random results for the other two models to avoid clutter. As can be seen, compared with adversarial manipulations, random changes have little impact on the model.

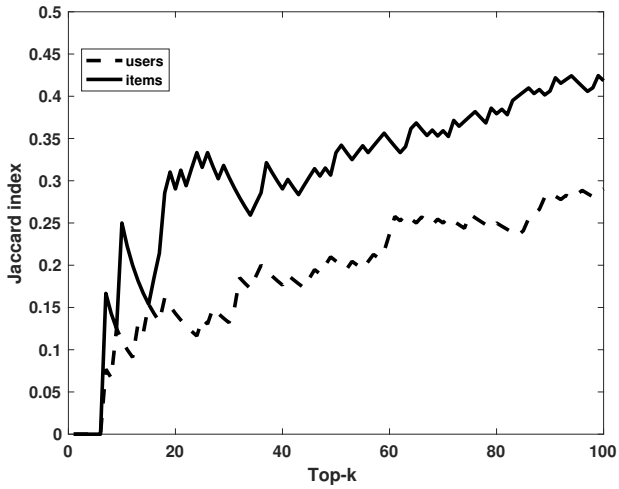


Figure 17. Jaccard index for two pairs of sets: top-k popular items vs. top-k affected items, and top-k active users vs. top-k affected users.

Table 20. Movies that appear in the top-20 popular items and top-20 items which are affected by the adversary.

Movies
Contact (1997)
Toy Story (1995)
Return of the Jedi (1983)
Scream (1996)
Air Force One (1997)
Jerry Maguire (1996)
The Rock (1996)
Back to the Future (1985)

We perform an error analysis to understand the behavior of the adversary in a qualitative manner. We look at the users whose predicted ratings from the GCN-based model are changed the most by adversarial attacks. We find that the greatest changes occur among active users (those who rate many items) and popular items (those rated by many users).

Figure 17 plots the Jaccard index for two pairs of sets of items and sets of users. Concretely, we compute the Jaccard index for top-k active users, and top-k affected users, for $k \in \{1, 2, ..., 100\}$, removing the users who have been involved in the attack. Similarly, we compute the Jaccard index for top-k popular items, and top-k

affected items. The graph shows large values of the Jaccard index, which indicates the adversary is highly effective in manipulating the system for items or users for which it has a long history. Table 20 shows nine movies, which are in both the top-20 affected items and the top-20 popular items.

Conclusion

Relational models have the potential to make better predictions by incorporating context from related entities. However, this context can become a liability when adversaries actively manipulate attributes and relationships. We use GCN and its inductive analog, GraphSage, as our representative models to show that neural graph-based models are indeed vulnerable to adversarial attacks. The fact that our attacks can be transferred to other models, as demonstrated in the case of a recommender system, suggest that this risk should be taken seriously. In the next chapter, we will focus on inductive node classification, and propose a few baselines for making graph neural nets robust.

CHAPTER VI

ROBUST GRAPH NEURAL NETWORKS

Introduction

In this chapter, we will discuss various methods to create a robust node classifier which can defend against adversarial edge-based perturbations. The focus of this chapter will be on inductive node classification. Inductive node classifiers have more applications in the real-world as nodes are being constantly added to real-world graphs. In addition, given independent prediction of nodes at test time, edge perturbations do not have to be symmetric, and do not have to maintain global properties of the graph, such as the number of triangles. Our adversary can be a node, which aims to confuse the model by replacing one of its neighbors with another. The adversary has no constraint on time, but can replace some of its edges within a budget, e.g., 20% of its edges. A main theme of this chapter revolves around creating efficient defense mechanisms, which will operate at the feature-level (linear-time), compared with a non-efficient edge-level (polynomial-time) defense strategy.

Before diving into the details of defense mechanisms, we will discuss a new message passing scheme, which improves the accuracy across several datasets for both inductive and transductive node classifiers. The proposed scheme is based on the idea of gating, which has been found helpful in various contexts in deep learning. Moreover, the output of the gate, which is a sigmoid nonlinearity, will be used in one of our defense mechanisms.

Edge-Gated Message Passing

One of the weaknesses of message passing schemes in vanilla graph neural networks is ignoring the multi-view nature of relations in graphs. We propose a new edge-based gating mechanism, so that each node sends unique messages to its neighbors, rather

than sending one message to all of its neighbors. This scheme will later be exploited in one of our defense mechanisms. In addition, we will demonstrate that this new message passing scheme could result in superior results in node classification tasks.

In node-based message-passing, one message is passed to all neighbors of a node y, i.e., $\vec{\mu}^t_{yx} = \vec{\mu}^t_{yz}; \forall z \in \mathcal{N}(y)$, where $\vec{\mu}^t_{yx}$ is the message that node x receives from y. We refer to $\vec{\mu}^t_{yx}$ as the *edge embedding* of the edge from node x to node y at layer t. We propose a gating function, g, which is a linear layer parameterized as follows:

$$g_\theta(h_x) = \text{sigmoid}(W(h_x + h_y)) \tag{6.1}$$

The input to the gate is the receiving node's representation and the sending node's representation, and its parameters are the weight matrix $W \in \mathbb{R}^{m \times m}$. The output of this gating network will be element-wise-multiplied with the received message, in order to select the best features from the incoming message.

$$\vec{\mu}^t_{yx} = \text{sigmoid}(W(h_x + h_y)) \odot h_y \tag{6.2}$$

As an example, for a GraphSage layer, we have:

$$h^t_i = \text{ReLU}(W_1^T[x_i; \alpha \sum_{j \sim \mathcal{N}(i)} \text{sigmoid}(W(x_i + x_j)) \odot x_j]) \tag{6.3}$$

A well-known gating mechanism, based on element-wise product with the output of sigmoid units, is used in long short-term memory (LSTM) units to avoid gradient vanishing (Hochreiter & Schmidhuber, 1997). Similar gating mechanisms have been extensively used as feature selection blocks in highway networks (R. K. Srivastava, Greff, & Schmidhuber, 2015), computer vision (van den Oord et al., 2016), Natural Language Processing (Dauphin, Fan, Auli, & Grangier, 2017; Marcheggiani & Titov, 2017), and classification on graphs (Z. Liu et al., 2018; J. Zhang et al., 2018). To the best of our knowledge, our work is the first to propose this edge-based gating mechanism. The aforementioned function requires E number of dot-product

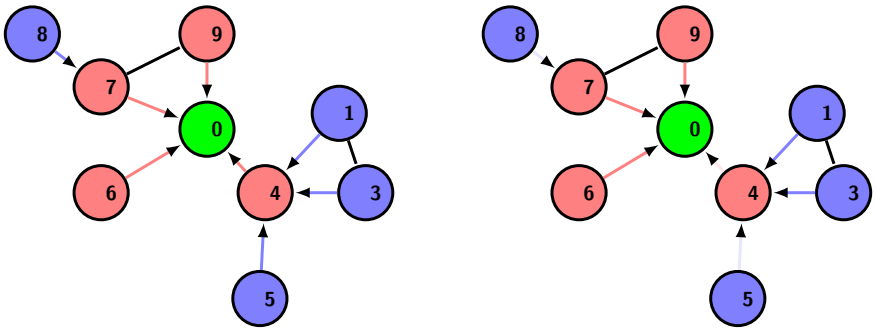


Figure 18. Comparing gated message passing, on the right, with the vanilla message passing on the left. Gated message passing will give more importance to some of the messages, as denoted by bolder edges.

operations, where E is the number of edges in the whole graph, in the case of full-batch optimization, or the number of sampled neighbors in the mini-batch for mini-batch optimization. Such computations are efficiently handled on a GPU.

Apart from making graph neural nets, edge-aware, Figure 18 demonstrates another impact of gated message passing in weakening the strength of some of the messages when they get multiplied with a gate output which is a small value (close to zero). This property will be exploited in of our defense mechanisms.

Defense

Given GraphSage with any type of aggregator,

$$H_i = W_2\Big[\overbrace{\text{ReLU}(W_1[x_i; \text{Agg}_{j\sim\mathcal{N}(i)} a_j X])}^{h_i^1}; \alpha \sum_j \overbrace{\text{ReLU}(W_1[a_j X; \text{Agg}_{k\sim\mathcal{N}(j)} a_k X])}^{h_j^1}\Big]$$

for each edge a_{ij}, the worst case perturbation (replacement of j with k) can be approximated by:

$$\max_{k,j} \frac{\partial J}{\partial a_{jk}} - \frac{\partial J}{\partial a_{jj}}$$

Computing such adversarial perturbations, as part of the training process, is infeasible due to the $\mathcal{O}(V)$ time complexity of the above maximization, where V is the number

of edges. This has to be computed for every instance in the training set, resulting in a polynomial-time search to find worst-case adversarial perturbations. This section explores efficient methods to create robust graph neural nets.

Link-Based Model. One way to handle bad edges and their negative impact is to enable the model to rate the trustworthiness of edges. Toward this end, we exploit the gating mechanism to make random or adversarial edges have less impact on the model. Intuitively, we can have a joint link-prediction and node classification model, wherein we jointly predict the goodness of a given edge, and scale down the outgoing message if the link predictor gives a low score to it, i.e., a negative edge. The following objective contains the original log-loss objective function for classification, and a ranking-based term to give higher values for a sample of positive edges, compared with a sample of negative/adversarial edges.

$$\frac{1}{n}\sum_{i}^{n} J(f(x_i|\theta), y_i) + \frac{1}{\mathcal{E}}\sum_{l} \max(0, 1 - \mathbf{s}_l(h_p, h_x) + \mathbf{s}_l(h_a, h_x)) \tag{6.4}$$

where $\mathbf{s}_l(h_p, h_x) = \frac{1}{1+e^{-W_l^T(h_p+h_x)}}$, is our linear link predictor, l denotes the layer and $\mathcal{E}$ is the number of pairs of positive and negative edges. Here, h_a denotes the representation of an adversarial node which creates an adversarial edge with node x, and h_p is a sample true neighbor of the node x. Note that the linear link predictor has a similar parameterization as the linear gating model which we proposed previously.

Algorithm 2 replaces a portion of the first-hop neighbors with random neighbors to create negative edges, corresponds them with some of the positive edges, and has a similar objective as 6.4, but applies the penalty term to all dimensions of the gate output. This mechanism increases model's power to detect adversarial edges and hence decrease the impact of adversarial edges.

Feature-Based Model. Another way to handle bad edges is to perturb the features of neighbors to approximate an edge-perturbation. That is, instead of

Algorithm 2 Link-based model

INPUT: Number of layers L, model parameters θ, linear gating parameter $W \in \theta$, proportion of adversarial neighbors p, stepsize sequence $\{r_t > 0\}_{t=0}^{T-1}$
for $t = 0, \ldots, T-1$ **do**
  Sample $x^a \sim X$ as random nodes for $p\%$ proportion of neighbors.
  $\theta^{t+1} \leftarrow \theta^t - r_t \nabla_\theta \big(J(\theta^t; x, x^a) + \lambda \sum_l^L \max(0, 1 - \mathbf{s}_l(h_p, h_x) + \mathbf{s}_l(h_a, h_x)) \big)$

changing neighbors, as the adversary would do, our inner adversary could perturb the features of neighbors of a given node. Figure 19 shows the idea behind feature-based perturbation; instead of adding an adversarial edge between nodes 0 and 1, the adversary is adding a *pseudo-node* which is not among the nodes in the graph, but remains in the convex hull of the data set. We use the outer convex bound by projecting the values of the adversarial node to remain in the convex outer bound of the convex hull, i.e., by clipping the values of the adversarial node to remain within the outer hypercube, i.e., $(-\max(X), \max(X))$.

Proposition 1 connects the feature-based perturbation with the edge-based perturbations.

Recall that,

$$\frac{\partial J}{\partial a_j} = a_j \frac{\partial J}{\partial X} X^T \tag{6.5}$$

Relaxing the one-hot constraint on a_j, and updating it based on the gradient (i.e., $a_j := a_j + \frac{\partial J}{\partial a_j}$), would give the following new H_i':

$$H_i' = W_2^T \Big[\overbrace{\text{ReLU}(W_1^T [x_i; \text{Agg}_{j \sim \mathcal{N}(i)}(a_j X + a_j \frac{\partial J}{\partial X} X^T X)])}^{h_i^1}; \\ \alpha \sum_j \overbrace{\text{ReLU}(W_1^T [a_j X + a_j \frac{\partial J}{\partial X} X^T X; \text{Agg}_{k \sim \mathcal{N}(j)} a_k X])}^{h_j^1} \Big] \tag{6.6}$$

Proposition 1: Gradient ascent on the one-hot vector of a_j is equivalent to gradient ascent on x_j transformed by $X^T X$ matrix.

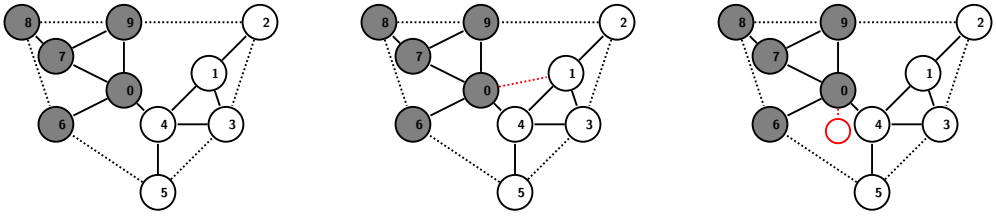


Figure 19. Instead of adding an adversarial edge from node 0 to node 1, we can we can add an edge to any data point within the convex hull.

This is equivalent to updating the input features by $X = X + \frac{\partial J}{\partial X} X^T X$. In other words, we can achieve adversarial perturbations on the edges by transforming input gradients with the matrix $X^T X$, which is computed using the training set. The transformation matrix is a scaled covariance matrix when the dataset is transformed to have zero mean and unit variance; thus, we call this a covariance-based transformation.

Algorithm 3 shows the algorithm to perform adversarial training using feature-based perturbations. As can be seen, some of the neighbors are replaced by random nodes, on which projected gradient ascent is performed. Our experiments showed that starting from random nodes in the graph, rather than the current neighbors performs better. This is in agreement with the findings of (Tramèr et al., 2018), which starts the search for adversarial examples after adding a Gaussian noise to the original image.

Robustness to Norm-Bounded Perturbations on Neighbors. Another approach for defense is to allow norm-bounded perturbations on all neighbors, in which case we tune the degree of perturbations. For this type of defense, we use two baselines, one of which uses a convex relaxation of neural nets, while the other one uses

Algorithm 3 Feature-based model

INPUT: Number of layers L, model parameters θ, proportion of adversarial neighbors p, stepsize sequence $\{r_t > 0\}_{t=0}^{T-1}$
$V = X^T X$
for $t = 0, \ldots, T-1$ **do**
  Sample $x^a \sim X$ as random nodes for $p\%$ proportion of neighbors.
  for $i = 0, \ldots, iter$ **do**
    $x^a \leftarrow x^a + \Pi_C \, \epsilon V \nabla_{x^a} J(\theta^t; x, x^a)$
  $\theta^{t+1} \leftarrow \theta^t - r_t \nabla_\theta J(\theta^t; x, x^a)$

adversarial training on the standard non-convex neural network. For the former, we extend the robust optimization framework of (Kolter & Wong, 2017) to GraphSage. The goal is to create a model robust to norm-bounded perturbations on the neighbors of a given node. While this type of defense is useful for giving guarantees on models' robustness, i.e., (robust to $\| \delta \|_\infty \leq \epsilon$ for training instances), we tune ϵ to defend against our edge-based adversary, without actually having a guarantee requirement. For the latter, we use projected gradient descent approach of (Madry et al., 2018).

The linear program 6.7 aims to find the worst adversarial example in the convex outer bound of the node's adversarial polytope, i.e., the set of all final-layer activations attainable by perturbing one-hop or two-hop neighbors by some Δ with ℓ_∞ norm bounded by ϵ. Concretely, the program aims to find the worst adversarial example given all possible adversarial labeling of the node, i.e., $e_{y^{target}}$ for all targets, for all norm-bounded perturbations of neighbors. Note that for this model, we use the mean-aggregated GraphSage without the use of gating.

A feasible dual solution to this linear program can be found, which provides a guaranteed lower bound on the solution of the primal. Additionally, the feasible set of the dual problem can be expressed as a two-layer network, which is very similar to a standard backpropagation network.

$$\underset{r^i}{\text{minimize}} \ (e_{y^*} - e_{y^{target}})^T r^i$$

subject to:

$$r^i = W_f \hat{h}_2^i + b_f$$

$$\hat{h}_2^i = W_2[h_1^i; \alpha \sum_{j \sim \mathcal{N}(i)} h_1^j]$$

$$\hat{h}_1^i = W_1[\hat{x}_i; \alpha \sum_{j \sim \mathcal{N}(i)} \hat{x}_j]$$

$$\hat{h}_1^j = W_1[\hat{x}_j; \beta \sum_{k \sim \mathcal{N}(j)} \hat{x}_k]$$

$$\begin{cases} h_{1,d}^z \geq 0 \\ h_{1,d}^z \geq \hat{h}_{1,d}^z & d \in \mathcal{I}, \forall z \in \{i, j, k\} \\ (u_d^z - l_d^z) h_{1,d}^z - u_d^z \hat{h}_{1,d}^z \leq -u_d^z l_d^z \end{cases} \tag{6.7}$$

$$h_{1,d}^z = 0, d \in \mathcal{I}^-$$

$$h_{1,d}^z = \hat{h}_{1,d}^z, d \in \mathcal{I}^+$$

$$\begin{cases} \hat{x}_j \geq x_j - \epsilon_1 \\ \hat{x}_j \leq x_j + \epsilon_1 \end{cases} \quad j \sim \mathcal{N}(i)$$

$$\begin{cases} \hat{x}_k \geq x_k - \epsilon_2 \\ \hat{x}_k \leq x_k + \epsilon_2 \end{cases} \quad k \sim \mathcal{N}(j)$$

$$\hat{x}_i = x_i$$

The derivation of the dual and the final optimization problem can be found in Appendix B. Finally, the classification loss is minimized on these worst-case adversarial examples.

Algorithm 4 PGD

INPUT: Number of layers L, model parameters θ, stepsize sequence $\{r_t > 0\}_{t=0}^{T-1}$
for $t = 0, \ldots, T-1$ **do**
  $x^a \in \mathcal{N}(x)$
  for $i = 0, \ldots, iter$ **do**
    $x^a \leftarrow \Pi_\epsilon(x^a + \epsilon \text{sign}(\nabla_{x^a} J(\theta^t; x, x^a)))$
  $\theta^{t+1} \leftarrow \theta^t - r_t \nabla_\theta J(\theta^t; x, x^a)$

The parameters of the model include $\{W_1, W_2, W_f, b_f\}$, wherein the last two are the weight and bias of a fully-connected layer, and the first two parameters are the parameters of two graph convolutional layers. Following most of the literature, the second convolutional layer is not followed by a nonlinearity and no bias for convolutional layers is used. ReLU activation functions are replaced with linear constraints, and $\hat{h}_1^z$ denotes pre-activation representation for node z. Based on the lower and upper bound values for a particular node, i.e., whether they are both positive $\mathcal{I}^+$, both negative $\mathcal{I}^-$, or span zero $\mathcal{I}$, h_1^z can be derived from $\hat{h}_1^z$. Concretely, given known lower and upper bounds l, u for the pre-ReLU activations, we can replace the ReLU equalities (i.e., $h = max(0, \hat{h})$) with their upper convex envelopes depending on the values of the lower and upper bound.

The algorithm for projected gradient descent (PGD) using norm-bounded perturbations is given in Algorithm 4. Note that the difference between these two methods with the feature-based model in Algorithm 3 is that these two methods allow perturbations on all neighbors of a node, where ϵ_1 and ϵ_2 determine the magnitude of perturbations for first-hop and second-hop neighbors. Whereas the feature-based model performs perturbations within the convex hull for a fraction of neighbors. The former approach is a more pessimistic approach to robustness.

Jacobian Regularization. We use another defense baseline which tries to penalize sensitivity of the learned representations with respect to changes in the

features of neighboring nodes.

$$\frac{1}{n}\sum_{i}^{n}\left(J(f(x_i|\theta),y_i)+\lambda \parallel \mathbb{J}_{H_i}(x_j) \parallel_F^2\right) \tag{6.8}$$

The regularizing term is the Jacobian of the logits inputs to the classifier with respect to one-hop neighbors' features. This regularization term can be broken into two pieces:

$$\mathbb{J}_{H_i}(x_j)=\mathbb{J}_{H_i}(h_i)\times\mathbb{J}_{h_i}(x_j)+\mathbb{J}_{H_i}(h_j)\times\mathbb{J}_{h_j}(x_j) \tag{6.9}$$

We break W into two submatrices V, i.e., operating on the node and U, i.e., operating on neighbors. Concretely,

$$W_1[x_i;\mathrm{Agg}_{j\sim\mathcal{N}(i)}x_j]=[V_1x_i;U_1\mathrm{Agg}_{j\sim\mathcal{N}(i)}x_j] \tag{6.10}$$

Using the chain rule and due to norm triangle inequality, we will have the following term:

$$\parallel \mathbb{J}_{H_i}(x_j) \parallel_F^2 \leq \parallel V_2^T\mathbb{J}_{\sigma(U_1\mathrm{Agg}_{j\sim\mathcal{N}(i)}x_j)}(x_j) \parallel_F^2 + \parallel U_2^T\mathbb{J}_{\sigma(x_j)}(x_j) \parallel_F^2 \tag{6.11}$$

Note that in the case of no nonlinearity, Jacobian regularization would be equal to ℓ_2 regularization. So for the second layer of GraphSage, which has no nonlinearity, we simply need to add an ℓ_2 term. And finally, due to submultiplicativity of Frobenius norm, Eq. 6.12 is upper bounded by the summation of all Jacobians.

$$\parallel \mathbb{J}_{H_i}(x_j) \parallel_F^2 \leq \parallel \mathbb{J}_{\sigma(U_1\mathrm{Agg}_{j\sim\mathcal{N}(i)}x_j)}(x_j) \parallel_F^2 + \parallel \mathbb{J}_{\sigma(V_1x_j)}(x_j) \parallel_F^2 + \parallel U_2 \parallel_F^2 + \parallel V_2 \parallel_F^2 \tag{6.12}$$

This is similar to the layer-wise Jacobian regularization of (Gu & Rigazio, 2015), used for defense against adversarial attacks in convolutional neural networks for image recognition. As an example, the the first term in 6.12 for a mean-GraphSage can be written as the following:

$$\mathbb{J}_{\sigma(\alpha U_1\sum_{j\sim\mathcal{N}(i)}x_j)}(x_j)=U_1^T\odot\sigma'(U_1\alpha\sum_{j\sim\mathcal{N}(i)}x_j) \tag{6.13}$$

Table 21. Dataset statistics

Dataset	PPI	Reddit	Pubmed	Citeseer	Cora
Nodes	56.9K	233.0K	19.7K	3.3k	2.7k
Edges	806.2K	114.6M	44.3K	4.7k	5.4k
Features	50	602	500	3,703	1,433
Classes	121	41	3	6	7

For our experiments on the PPI dataset, we use layer normalization (Ba, Kiros, & Hinton, 2016), for which the Jacobian have a different structure which can be found in Appendix B. And finally, we found that applying this type of regularization for defense against second-hop neighbors' perturbations will deteriorate the performance, so we ignore the second-hop neighbors for this defense method.

Experiments

In this section, we evaluate our gated message passing model for inductive and transductive settings, and later focus on robustness of GraphSage in inductive settings. The datasets that we use are the same as the ones in the previous chapter: citation networks Cora, Citeseer, and Pubmed, protein-protein interaction, and Reddit social media. The first two are used for our tranductive experiments, while the last two are used for our inductive experiments. Pubmed is used for both settings; however, the transductive setting is done in a semi-supervised fashion with only 60 labeled instances. The details of these datasets can be found in Table 21

Gated Message Passing. In this section, we evaluate our message passing scheme in both inductive and transductive settings. Figure 20 shows the validation loss for 100 epochs on the PPI dataset in the inductive setting. Clearly, the gated message passing achieves a lower loss at a much faster rate the vanilla message passing.

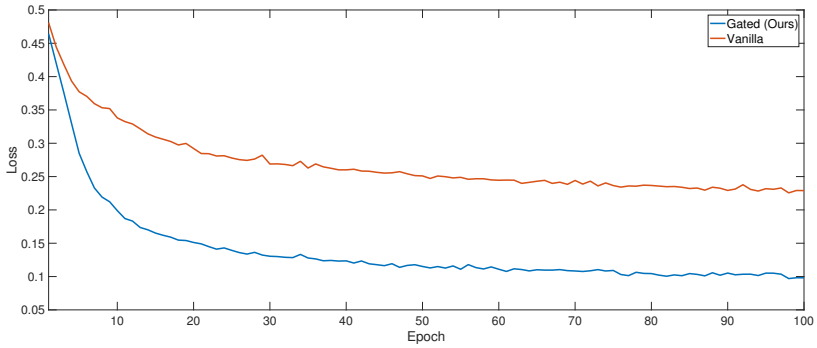


Figure 20. Loss on validation set for PPI dataset using vanilla and gated message passing.

A simple transductive experiment on a synthetic dataset can further highlight the superior performance of the gated message passing. We create a random graph of 2,000 nodes, which are divided into two classes of 1,000 nodes each. We add 1,000 edges among the nodes within each class, and gradually add random edges between the nodes in the two classes, as demonstrated in Figure 21. As more edges are added, more useful patterns emerge, which the classifiers successfully learn, and on which they achieve close to 100% accuracy. The gated model is much faster at learning good patterns, as it achieves a much higher accuracy in the early stages. This success can be attributed to the ability of the model in determining the usefulness of the edges, and subsequent better representation learning.

Transductive Experiments. This is a semi-supervised experiment, and we use ℓ_2 regularization on model parameters, as well as dropout after each layer and also on the input. For hyper-parameter selection, we use {2e-3, 1e-3, 5e-4, 1e-4} for regularization and $\{0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ for dropout and tune them on the validation set. We perform full-batch optimization for 800 epochs, and we compare our model with the following baselines.

Planetoid: Yang, Cohen, and Salakhutdinov (2016) propose an embedding learning method which injects label information in the process.

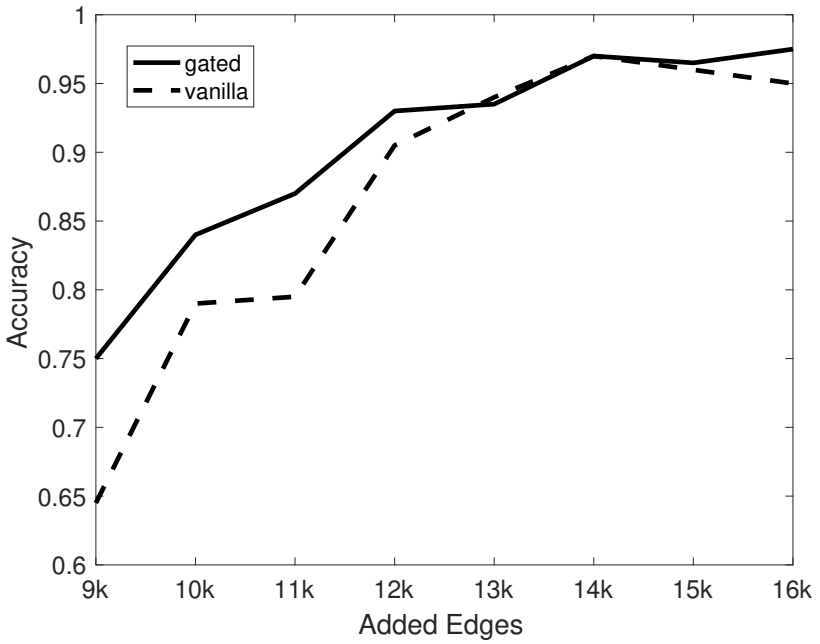


Figure 21. Binary classification accuracy using GCN; edges are gradually added to a random graph.

Chebyshev: Defferrard, Bresson, and Vandergheynst (2016) employ Chebyshev polynomials of the graph Laplacian, and avoids expensive computation of eigenvectors of the Laplacian to create spectral convolutions on graphs.

GAT: Velickovic et al. (2018) use four attention heads of 256 units and concatenate them for a dimension of 1,024 in the hidden layers.

MoNet: Monti et al. (2017) use a mixture model of Convolutional Neural Networks (CNN) which generalizes CNN architectures to graph structures.

AGNN: Thekumparampil, Wang, Oh, and Li (2018) use a simpler attention function than GAT.

N-GCN: Abu-El-Haija, Kapoor, Perozzi, and Lee (2018) train multiple instances of GCNs over node pairs discovered at different distances in random walks.

TAGCN: Du, Zhang, Wu, Moura, and Kar (2017) define graph convolution operation on the vertex domain as multiplication by polynomials of the graph adjacency matrix.

Table 22. Accuracy on Citation Networks

Method	Citeseer	Cora	Pubmed
Planetoid	70.30	75.70	79.00
Chebyshev	69.80	81.20	79.00
GAT	72.50	83.00	79.00
GCN	70.30	81.50	77.20
MoNet	-	81.70	78.80
AGNN	71.70	83.10	79.90
NGCN	72.20	83.00	79.50
TAGCN	71.4	**83.30**	81.10
gated-GCN (ours)	**72.74 ± 0.66**	83.16 ± 0.48	**81.23 ± 0.45**

Filters are adaptive to the topology of the graph as they scan the graph to perform convolution.

For these experiments, we modified the structure of a GCN, inspired by loopy belief propagation in graphical models, the details of which can be found in Appendix B. As can be seen in Table 22, our results outperform the state-of-the-art model TAGCN on average, and achieves state-of-the-art results on two of the baselines.

Inductive Experiments. Now we report results on inductive classification. Note that the PPI dataset is a multi-label classification dataset, where each node can belong to any number of 121 existing labels. Thus, while for other datasets we use a negative log-likelihood loss, for this dataset we use a logistic loss on all of the labels, and we report micro-F1 score. We use a subsample of the graph, which uses 128 nodes for each node in the graph, and we sample 25 and 10 nodes from one-hop and two-hop neighbors, respectively, for Reddit and Pubmed. We increase these numbers to 40 and 25 for the PPI dataset. In addition, while 128 hidden units for Reddit and Pubmed are used, we use 512 hidden units for PPI. We use 10 epochs for training Pubmed and Reddit, and 100 for PPI. And finally, for the PPI dataset, we use layer normalization

Table 23. Micro-F1 on three datasets.

Method	Reddit	PPI	Pubmed
GraphSage	95.81	90.13	**89.60**
FastGCN	93.70	-	88.00
GCN-VR	**96.30**	97.90	-
gated-GraphSage (Ours)	95.85	**98.54**	89.50

(Ba et al., 2016). We use the same hyper-parameters for our GraphSage with mean aggregator. Apart from GraphSage, we use two other baselines as follows,

FastGCN: Chen, Ma, and Xiao (2018) subsample the receptive field in each layer universally, using importance sampling.

GCN-VR: Chen and Zhu (2018) propose a control variate-based algorithm which allows an unbiased sampling strategy.

While our gated mechanism has little impact on Reddit and Pubmed, its performance on PPI, which is the most dense graph among all of these datasets, is superior to other baselines, outperforming the regular graphSage by 8% in Micro F1 score.

Robustness

Hyper-parameters. Our robust baselines have different hyper-parameters which are tuned based on their performance on validation sets. For the norm-bounded models, we tune the hyper-parameters ϵ_1 and ϵ_2. For the feature-based model, we tune the proportion of the node, p ,which will be replaced by random nodes to perform gradient ascent on, and the perturbation magnitude ϵ. For the Jacobian regularizer we tune λ, which determines the weight of the regularizer, compared with the negative-log likelihood loss function. We tune a similar weight, λ, in addition to the proportion of the random nodes, p, for the link-based model. Table 24 shows the details of our tuning procedure.

Table 24. Hyper-parameters for each of the robust baselines.

Method	Hyper-parameters
Jacobian	$\lambda \in \{$1e-4, 1e-3, 1e-2, 1e-1, 1$\}$
Link-Based	$p \in \{0.2, 0.25, 0.3\}$; $\lambda \in \{$1e-4, 1e-3, 1e-2, 1e-1, 1$\}$
Feature-Based	$p \in \{0.2, 0.25, 0.3\}$; $\epsilon \in \{$1e-4, 1e-3, 1e-2, 1e-1, 1$\}$
Convex	$\epsilon_1, \epsilon_2 \in \{$1e-4, 1e-3, 1e-2, 1e-1, 1$\}$
PGD	$\epsilon_1, \epsilon_2 \in \{$1e-4, 1e-3, 1e-2, 1e-1, 1$\}$

Results. Now we focus on defense against adversarial attacks against variants of GraphSage, and our robust baselines. Except the convex method, all of our defense methods use a gated message passing with a mean aggregator. The convex method uses a mean aggregator with no gating. Figure 25 shows micro-F1 score of all models on the clean graph, as well as cases where 1%, 5%, 10%, and 15% of edges are replaced adversarially in a white-box setting. Note that the convex optimization method is not applicable to a multi-label case, and hence no result for PPI is reported. These results are similar to the black-box setting in Table 26, wherein we import adversarial edges from the vanilla GraphSage with mean aggregator and attack other models.

We show the best results on each case with bold, and use underline to show cases where a few models are performing equally well on adversarial perturbations. The feature-based model has the most consistent results in improving the robustness of the models. The link-based model has the worst performance in Pubmed and Reddit, but outperforms PGD and Jacobian model on PPI. These results also demonstrate that structural perturbations on graphs, even when no knowledge of the model is granted, can inflict considerable damage. In fact, for most cases, the black-box adversary is more successful than a white-box adversary.

Gradient Masking Is Not an Issue. Results on the white-box and black-box attacks are very similar, which suggests gradient masking (Tramèr et al., 2018) is not a problem for any of our defense mechanisms. Gradient masking refers to smoothing

Table 25. Micro-F1 score under white-box attacks on vanilla models (`mean`, `attention`, and `gated`) and our robust baselines (`gatedLink`, `gatedFea`, `gatedJac`, `PGD`, and `convex`), using 0.01, 0.05, 0.10, and 0.15 edge replacement strategies.

dataset \ method		`mean`	`attention`	`gated`	`gatedLink`	`gatedFea`	`gatedJac`	`PGD`	`convex`
Pubmed	`clean`	89.60	89.50	89.50	88.10	87.30	**90.10**	88.20	88.20
	`0.01`	89.20	81.90	88.60	87.80	87.30	**90.10**	88.20	88.20
	`0.05`	87.60	80.70	87.00	87.10	87.30	**90.20**	88.20	88.20
	`0.10`	85.50	80.10	84.40	85.60	87.10	**90.60**	88.20	88.10
	`0.15`	83.50	79.20	81.90	84.20	87.10	**90.20**	88.20	88.00
Reddit	`clean`	95.81	**96.05**	95.85	95.72	95.44	95.98	95.86	95.80
	`0.01`	95.22	95.46	95.12	95.82	95.58	96.20	95.84	95.66
	`0.05`	89.46	91.02	90.66	95.12	95.34	95.92	95.92	95.58
	`0.10`	75.64	82.06	82.66	93.58	95.06	95.28	95.76	95.48
	`0.15`	50.98	68.08	67.36	90.40	94.82	94.38	95.62	95.28
PPI	`clean`	90.13	90.18	**98.54**	97.65	97.42	94.14	98.52	-
	`0.01`	87.38	85.00	**98.24**	97.33	97.13	93.17	98.12	-
	`0.05`	70.83	73.58	90.89	95.40	**95.86**	87.59	92.56	-
	`0.10`	66.09	65.9	80.25	91.28	**93.31**	82.30	83.55	-
	`0.15`	63.18	60.96	71.43	84.84	**88.76**	77.15	75.11	-

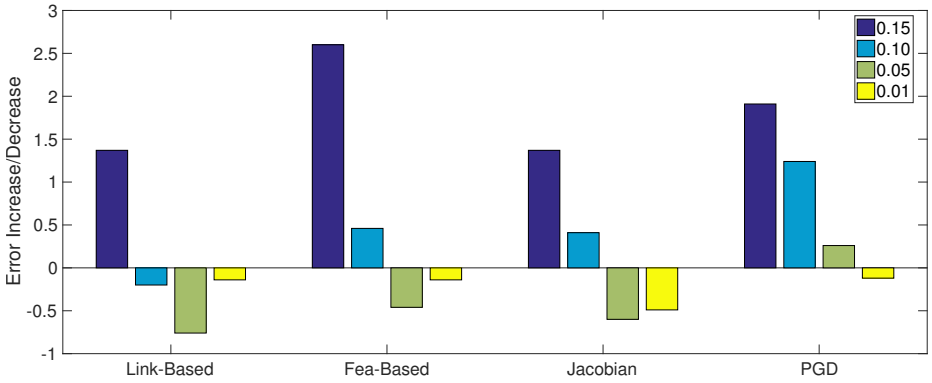


Figure 22. Increase/Decrease in error from white-box to black-box attack on the PPI dataset.

of the objective function in adversarial training, which leads to unstable gradients, because of which a white-box adversary cannot attack a model. Similar performance on both white-box and black-box adversaries suggests our defense mechanisms are not misleadingly robust to white-box attacks. Figure 22 shows the difference between the black-box and white-box adversary given different edge-replacements on the PPI dataset. In addition, in Figure 23, we plot the distribution of gradients for the Jacobian model and the vanilla model, which shows similar pattern for both models,

Table 26. Micro-F1 score under black-box attacks using 0.01, 0.05, 0.10, and 0.15 edge replacements from the `mean` model.

dataset \ method		attention	gated	gatedLink	gatedFea	gatedJac	PGD	convex
Pubmed	0.01	82.50	89.50	88.00	87.30	90.00	88.20	88.20
	0.05	81.90	82.50	87.50	87.30	88.90	88.20	88.20
	0.10	81.60	84.90	87.00	87.30	88.00	88.20	88.20
	0.15	80.20	82.60	85.80	87.30	87.00	88.20	88.20
Reddit	0.01	95.64	95.40	95.72	95.52	96.12	95.84	95.66
	0.05	93.10	92.58	94.02	95.28	95.92	95.70	95.36
	0.10	87.36	87.22	91.44	94.78	95.28	95.36	95.00
	0.15	74.28	74.56	86.14	94.26	94.38	94.80	94.26
PPI	0.01	85.75	98.24	97.47	97.27	93.66	**98.26**	-
	0.05	77.52	90.89	96.16	**96.32**	88.19	92.27	-
	0.10	71.33	80.25	91.48	**92.85**	81.89	82.31	-
	0.15	66.45	71.43	83.47	**86.16**	75.78	73.20	-

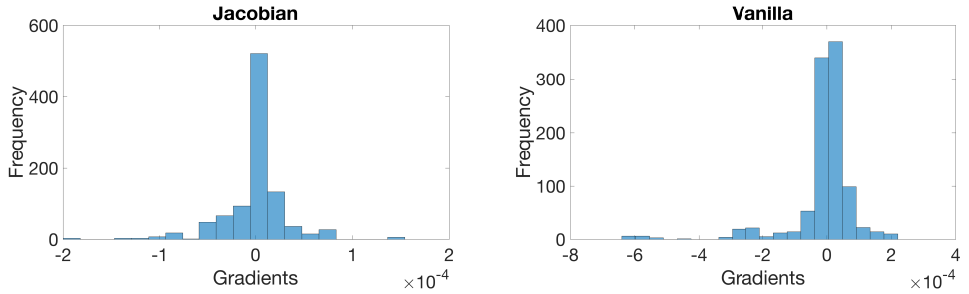


Figure 23. Distribution of gradients for a non-robust model and Jacobian-based robust model.

pointing to the fact that the robust model is not driving the weights of the network to simply diminish the gradients.

Inverted Word Attack. We extend the idea of working with features rather than edges, and create a new attack which we call *inverted word attack*, for the Pubmed dataset. In this attack, the adversary inverts all words in the features of first-hop and second-hop neighbors, which are binary bag-of-words representation of papers. That is, words that do not appear in the neighboring papers will appear, and words that appear in those papers will be removed. This is the worst-case attack an adversary can make, in which links are not informative, and indeed will decrease the accuracy if used.

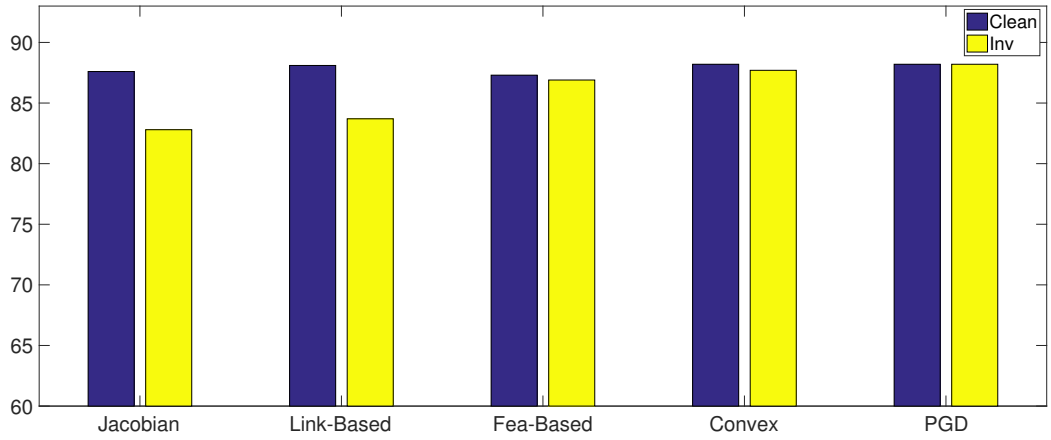


Figure 24. Comparison of defense against inverted word attack on Pubmed.

For more perspective, a two-layer multi-layer perceptron, which does not use links, will achieve 87.4 accuracy on the dataset; while a two-layer gated-GraphSage will achieve a 89.50 accuracy on the dataset. An inverted word attack will decrease the accuracy of the latter to 60.40, considerably worse than the case, in which the model does not use links.

Figure 24 shows that the inverted attack inflicts the biggest damage on the Jacobian-defended and link-based models by decreasing the accuracy by 4.80% and 4.40%, respectively. The accuracy of the feature-based and the convex models are decreased by 0.4% and 0.5%, respectively, while the PGD model remains completely robust to the inverted word attack. Note that for the convex and PGD methods, we set the hyper-parameters ϵ_1 and ϵ_2 to 1, while we tune hyper-parameters for other models based on their performance on the validation set of the inverted data. The convex and PGD models are the only models that achieve an accuracy (i.e., 87.7 and 88.20, respectively) above the multi-layer perceptron on clean data (87.4); effectively guaranteeing that when edges are not trustworthy, the accuracy of the model does not get worse than a case where we do not take edges into account. These two most successful models incorporate norm-bounded perturbations on all neighbors, and are

thus expected to perform better than other methods on this attack scenario, which manipulates all first-hop and second-hop neighbors.

Models' Behavior. We can gain some insight by studying the inner workings of these defense mechanisms. Figure 25 plots the variance of the weights of the aggregator at the first layer, during training. The feature-based, convex, PGD, and Jacobian defenses all keep the variance very low, penalizing overfitting to some of the features. The vanilla model and the link-based model do not have this property. This should not be surprising, as these four models all try to reduce the impact of noise on neighbors's features. The link-based defense has an additional term in the objective function, which would penalize higher scores to adversarial edges than regular edges. Figure 26 plots the decrease of this loss term in the link-based defense for the first and second layer of the network, in which we see the decrease for the second layer is stronger. This additional term helps the link-based model to be competitive with other robust models, while being agnostic to noise on the features of neighbors.

Furthermore, we study alignment of edge-perturbation with class distribution in Figure 27. Concretely, we collect best edge replacement for all 1,000 test nodes in Pubmed, and plot the distribution of the label of adversarial neighbors against the original neighbors that they replace. Reminding that Pubmed has 3 classes, in each figure, the color of the block at (3, 2) denotes the proportion of edge replacements, which replaced a neighbor of class 2 to a node of class 3, to all 9 types of edge perturbations. Brighter blocks denote more perturbation for that type.

Again, we see the behavior of the adversary against the link-based model is the most similar to the vanilla model. This suggests that the link-based model is simply enabled to detect adversarial edges, without changing the behavior of the model. Similarly, the Jacobian model and the PGD model have the most similar behavior

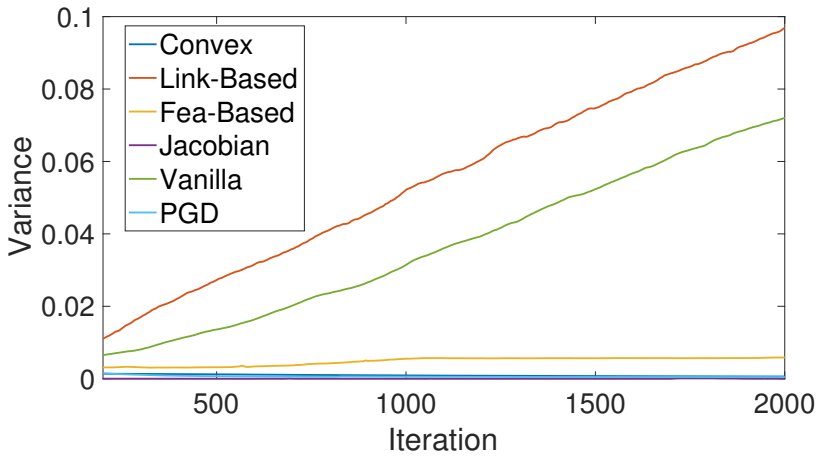


Figure 25. Variance of the weights of the first-layer aggregator on Pubmed.

among the other robust models. A large portion of edge perturbations against the Jacobian and PGD models, and to a lesser extent the feature-based model, fall in the (2,2) block, suggesting that the adversary is picking nodes from the same class, which is a weak adversarial move; effectively, the robust models are making it harder for the adversary to attack them.

The adversary is, almost always, replacing a node of the majority class, 2, in the the feature-based model. On the other hand, the convex model yields change of neighbors from different classes, more than PGD, Jacobian, and feature-based models. This might be due to optimizing a different objective function (robust error in the linear program 6.7), and the fact that it does not rely on computing the gradients to increase robustness.

Impact of Random Neighbors. As was pointed out, the feature-based model replaces some of the neighbors with random nodes, on which it performs gradient ascent. This randomization process was found to beneficial in improved robustness of the feature-based model. We perform a similar experiment for the PGD model; while PGD aims to make the model robust to perturbations on all neighboring nodes, we replace 20% of its neighbors with some random nodes, and

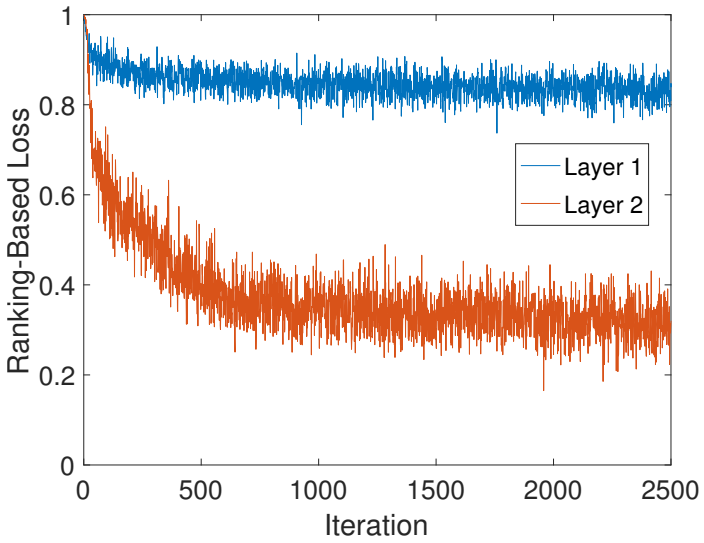


Figure 26. Ranking-based loss for the link-based method on Pubmed.

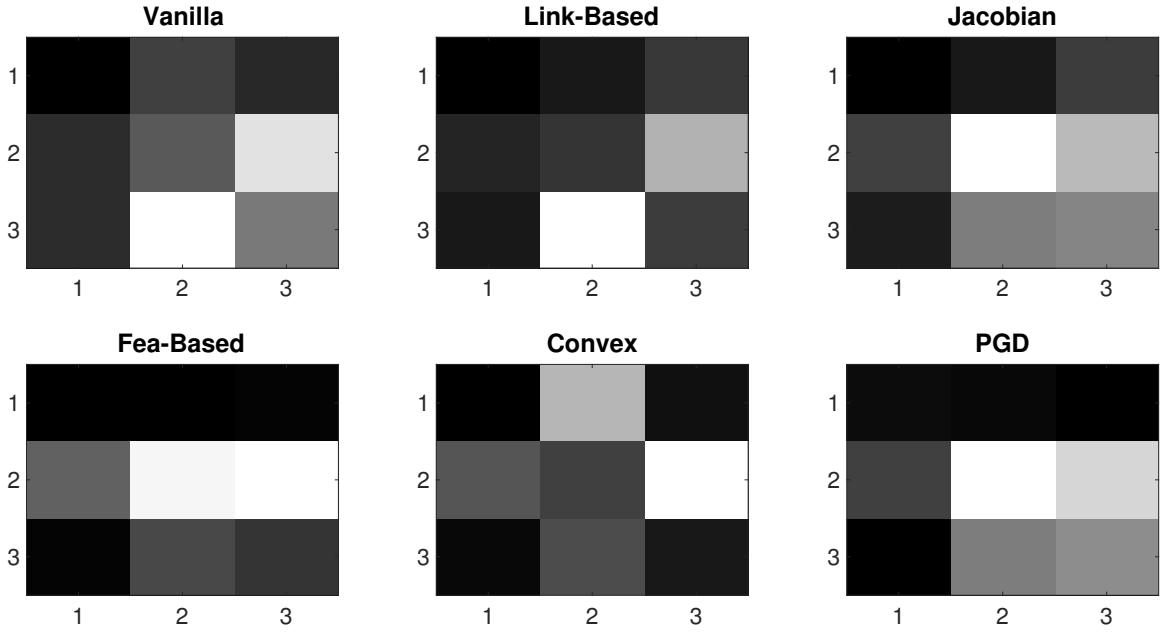


Figure 27. Alignment of edge-perturbation with class distribution for Pubmed. The x-axis denotes the label of the adversarial node, and the y-axis denotes the label of the original node.

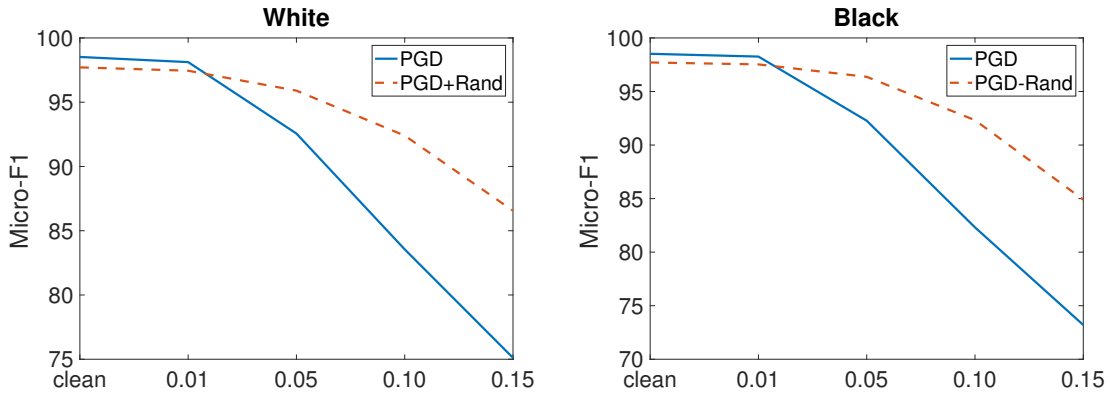


Figure 28. Robustness of adversarially trained models on the PPI dataset using PGD and PGD with random starting points for 20% of neighbors.

keep the rest of the training as is. Figure 28 shows that PGD is also benefiting from this randomization process, and its performance on PPI is now competitive with the feature-based model, and outperforms the link-based model, for both white-box and black-box attacks.

Impact of Covariance-Based Transformation. Recall that the feature-based model transforms the gradient vector with $X^T X$; we plot the accuracy on the Reddit dataset for two variants of the feature-based defense: one where we transform the gradient vector with $X^T X$, and one where we do not. We saw the strongest improvement by the transformation on the Reddit dataset, and very little difference on the other datasets. Since the Reddit dataset is normalized to have zero-mean and unit-variance features, we call this transformation a covariance-transformed feature-based perturbation. Note that for both cases, the hyper-parameter ϵ, is tuned on the validation set, which for the latter method is picked from a pool of large numbers, (i.e., {1e1, 1e2, 1e3, 1e4}).

Conclusion

In this chapter, we first propose a gated message passing technique, which provides significant improvements over vanilla message passing for inductive and transductive

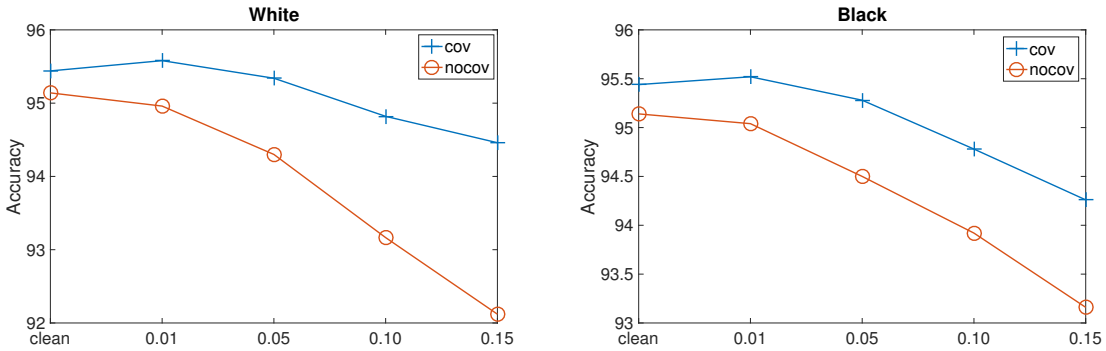


Figure 29. Robustness of adversarially trained models on the Reddit dataset using feature-based perturbations and covariance-transformed feature-based perturbations.

node classification. Then, we propose five baselines for robustness analysis of GraphSage. We evaluate these models using white-box and black-box attacks, and discuss their properties. Our best results were given by models that aim to be robust to feature perturbations. This could be done by performing bounded perturbation on all one-hop and two-hop neighbors, as in the case of our convex or PGD method, or it could be done by having unbounded perturbations on a fraction of neighbors, as in the case of the feature-based perturbation method. In either case, this type of defense makes the model robust to edge perturbations; this is mainly because the model is able to defend against a broader set of attacks (i.e., nodes similar to the actual neighbors but not necessarily in the graph), and thus can defend against adversarial edge perturbations.

CHAPTER VII

CONCLUSION AND FUTURE DIRECTIONS

As deep learning becomes more powerful in solving problems in AI, more attention needs to be given to cases where these models fail. In this dissertation, we investigated the robustness of neural network models with discrete input through the lens of adversarial examples. We first developed methods to attack these models, and then created methods to defend against such attacks.

Summary of Contributions

Attacking a Model with Discrete Input. Prior to this, it was not known how to create adversarial examples using gradients of the model. We created differentiable string-edit and edge-manipulation operations to attack deep NLP and graph-based neural models, respectively. We studied a variety of tasks, i.e., machine translation and text classification in NLP, inductive and transductive node classification and a recommender system for graph-based models.

Black-Box vs. White-Box Adversary for NLP. After developing the white-box adversary, we investigated the problem of targeted attacks on a character-level neural machine translation (NMT), in which the adversary aims to remove or replace certain words in text. While simple black-box attacks can simply hamper an NMT, a white-box adversary can attack a model more aggressively. We showed that our white-box adversarially trained model performs better than previous black-box trained models on different types of noise, including the most elusive type of noise, natural noise, which consists of most common typos and misspellings. We proposed a new methodology for creating and evaluating targeted adversarial examples for machine translation.

Improved Message Passing in Graph Neural Nets. We created a novel message passing scheme that achieves state-of-the-art results on some node classification benchmarks, as well as giving us a foundation for creating a link-based defense mechanism, which would rank positive edges, i.e., edges existing in the graph, higher than negative edges, i.e., edges that do no exist in the graph.

Robust GraphSage. We focused on developing robust node classification models, which would defend against adversarial edge perturbations. These adversarial perturbations could be due to a real-world adversary, e.g., a link farmer, or inherent noise in our measurements. In either case, the goal of a robust model is to perform well under noisy conditions. We provided five efficient methods to create a robust GraphSage; most of these methods rely on feature-based perturbations, which are equivalent to a variant of relaxed edge-based perturbations. We empirically show that our defense mechanisms do not mask gradients, and have similar performance under white-box and black-box attacks.

Future Directions

Theoretical Investigation of Black-Box Examples. Different problems show varying degrees of sensitivity to adversarial examples. In other words, some models are easier to trick than others. Table 27 shows that neural machine translation is very sensitive even to random changes. On the other hand, text classification is the most robust model, wherein only white-box attacks can break it. And finally, while graph-based models are not sensitive to noise, they can be broken with black-box attacks imported from a different model. Investigating differences between models would give us insight about security properties of machine learning models. While there are some empirical investigation of black-box examples (Y. Liu et al., 2017),

Table 27. Models' sensitivity to attacks.

Model	White-Box	Black-Box	Random
Classification	High	Low	Low
NMT	High	High	High
Graph	High	High	Low

studying theoretical properties of black-box adversarial examples (Tramèr et al., 2017) can shed light on differences between different machine learning models.

GANs for Defense. A few recent works have investigated the problem of creating adversarial examples with generative adversarial networks (GANs). Works of Xiao et al. (2018) and Song, Shu, et al. (2018) have the advantage that the norm-bounded perturbation requirement is removed. This can be extremely helpful for attacks on graph-based models. Recall that we have shown that feature-based perturbation defense provides good defense. In particular, one of our feature-based adversarial training schemes use projection of adversarial neighbors to the convex hull of the dataset. One way to create such adversarial neighbors is to create adversarial examples in the feature space using a GAN, which does not have a perturbation bound requirement, and can guarantee us to produce samples from a similar distribution as actual nodes in the graph.

Robust Defense for NLP. There is a growing interest in understanding vulnerability of NLP models, and improving their performance in noisy conditions (Michel & Neubig, 2018). Adversarial training, as our experiments demonstrated, can improve robustness of NLP models. What is missing is having provable guarantees for performance of NLP models, through robust optimization, which will be a harder problem for discrete inputs. Further, the current robust optimization methods for neural nets are limited to linear and convolutional layers (without pooling). Attention (Bahdanau et al., 2014) is the main building block for most NLP architectures. In

order to create robust NLP models, we need to create a dual for an attention layer, as well as other blocks which are less straight-forward to create. Robust NLP models can have applications in problems where optimal precision is required, and certified models would be desired.

Data-Poisoning Attacks/Defense for Graphs. One popular task for graph-based machine learning is link prediction, which as we mentioned in chapter V, can be targeted by data poisoning attacks, in which the training set is manipulated. Thus, a defense mechanism for this task should be able to have a built-in mechanism to differentiate between good and bad training instances. We demonstrated how to attack a GCN-based recommender system, which posed the recommendation task as link prediction, but did not explore defense mechanisms for it. Defending against such attacks is an important research problem, especially for link prediction, since it is a popular area of research, and we can find real-world use cases, for which this type of attack is plausible.

APPENDIX A

DEEP LEARNING BACKGROUND

In this chapter, we will give a brief overview of deep learning for NLP and graph neural networks. In particular, we will describe the models that have been used for our experiments.

Deep Learning

Deep learning refers to deep neural network models, which have found great appeal in the AI community. Advancements in hardware, in particular efficient implementations of matrix operations in GPU, led to the era of deep learning. Compared to the previous machine learning techniques, deep learning focuses on representation learning, and has very simple learning and inference procedures. In the next section, we start by discussing a simple neural network, and then discuss more complex models which are used for our work.

Feed-Forward Neural Network. A feed-forward neural network is a type of neural network wherein the neurons are structured in layers, and only connections to subsequent layers are allowed. Inspired by the brain, neural nets have a neuron-like behavior at their primary computational unit. The behavior of a neuron is controlled by its weights W. Hence, the weights are where the information learned by the neuron is stored. More precisely, a neuron uses the weighted sum of its inputs, and squeezes them using a nonlinearity function, such as a rectified linear unit (ReLU) (Nair & Hinton, 2010).

$$h(x) = \max(0, W^T x) \tag{A.1}$$

Training is achieved by minimizing the network loss, $J(.)$, on the training set, e.g. the collection of documents which are labeled as spam or not. In order to minimize this function, the gradient $\frac{\partial J_\theta(.)}{\partial \theta}$ needs to be calculated, which can be done using

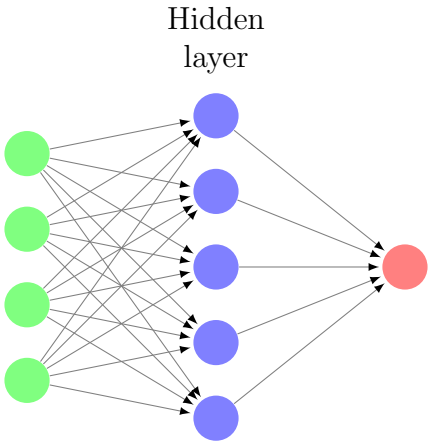


Figure A.30. Neural network with four input neurons, one hidden layer of six units, and 1 output neuron.

backpropagation. A typical loss function used for classification problems is *negative log-likelihood*, which penalizes low probability scores for the correct class.

$$J(X, Y) = \frac{1}{n} \sum_{i=1}^{n} -\log g(y_i|x_i) \tag{A.2}$$

where g is given by the output of the network, and n is the number of training instances. Variants of gradient descent methods, such as Adam (Kingma & Ba, 2015) will update parameters based on these gradients in several epochs. The result of this processe is a set of weights that enables the network to do the desired input-output mapping, as defined by the training data.

Word Embeddings. At the core of deep learning techniques for NLP lies the vector-based word or character representations, which map words or characters to an n-dimensional space, e.g. 300, 500. This approach enables us to have multiple dimensions of similarity, and to encode the syntactic and semantic features of the words in a compressed numerical format. Having word vectors as parameters, rather than fixed input, makes neural models flexible in finding different word embeddings for separate tasks (Collobert & Weston, 2008). Furthermore, through a composition model, one can map phrases and sentences, i.e., sequences of these vectors, to the

same n-dimensional space. These word vectors are stacked into a word embedding matrix $M \in \mathbb{R}^{n \times |V|}$ where $|V|$ is the size of the vocabulary. Each word is associated with a vocabulary index into the embedding matrix, from which we retrieve the word's vector from. We often use pre-trained word embeddings, which are trained on millions of sentences, and aim to make words that appear in similar contexts have more similar representations (Mikolov, Sutskever, et al., 2013).

Highway Network. Highway networks (R. K. Srivastava et al., 2015) facilitate training deep networks by *carrying* some of the information from the lower layers through gates. The output of a highway network is the following:

$$z = t \cdot \text{sigm}(W_H y + b_H) + (1 - t) \cdot y \tag{A.3}$$

where $t = \text{ReLU}(W_T y + b_T)$, and $\text{ReLU}(.)$ and $\text{sigm}(.)$ are the element-wise rectified linear unit and sigmoid activation functions, respectively.

Recurrent Neural Nets. Recurrent neural nets (RNNs) can be regarded as a neural network with feedback, where the output of a layer is fed back to a network. This allows us to model temporal aspect of data and to extract features from a sequence. RNNs are called recurrent because they perform the same computation for every element of a sequence, with the output being dependent on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory, RNNs can make use of information in arbitrarily long sequences; but in practice, they are limited to looking back only a few steps. Figure A.31 shows a RNN being unrolled (or unfolded) into a full network with 6 steps. An unrolled RNN is like a feed-forward network, with shared parameters at layers, where a new input is fed to the network at every layer of the network. The formula that governs the computation happening

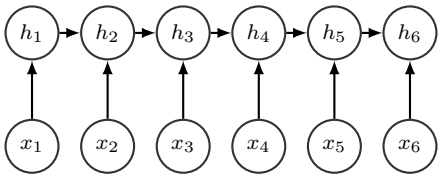


Figure A.31. A recurrent neural network

in a RNN is as follows :

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

where σ is a nonlinearity function such as $tanh$, W_{xh} represents the weights between the input and the hidden unit, and W_{hh} represents the weight between the two consecutive hidden units.

RNNs suffer from the *vanishing gradients* problem, which is the effect of multiplying n small gradients to compute gradients of the front layers in an $n-$layer network, meaning that the gradient decreases exponentially with n and the front layers train very slowly. There are a few ways to combat the vanishing gradient problem: using ReLU instead of either tanh or sigmoid activation functions. The ReLU derivative is a constant of either 0 or 1, so it is not as likely to suffer from vanishing gradients. An even more popular solution is to use Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) architectures. LSTMs (Hochreiter & Schmidhuber, 1997) is the most widely used models in NLP today. GRUs, first proposed by Cho et al. (2014), are simplified versions of LSTMs. RNNs have been used for language modeling (Mikolov, Karafiát, Burget, Cernockỳ, & Khudanpur, 2010), sequence labeling (Kurata, Xiang, Zhou, & Yu, 2016; Sak, Senior, & Beaufays, 2014), and parsing (Dyer, Ballesteros, Ling, Matthews, & Smith, 2015).

Long Short-Term Memory (LSTM). In an LSTM, one cell consists of three gates (input, forget, output), and a cell unit. Because of the gating mechanism

in LSTMs, the cell can maintain information, and it can prevent the gradients from undesirable changes during backpropagation. Despite this design, gradient exploding still remains a problem, which is mitigated by *gradient clipping* (Pascanu, Mikolov, & Bengio, 2013). Gates use a sigmoid activation, while input and cell state are transformed with tanh. LSTM cell gates can be defined by the following set of equations:

$$i_t = \text{sigm}(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$f_t = \text{sigm}(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$o_t = \text{sigm}(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

State updates can be defined by:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot c_{in_t}$$

$$h_t = o_t \cdot \tanh(c_t)$$

where c_{in} is the transformed input:

$$c_{in_t} = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_{c_{in}})$$

Because of the gating mechanism, the cell can maintain information for longer periods of time and can prevent the gradient from vanishing during backpropagation.

Sequence-to-Sequence. A widespread architecture based on RNNs is a sequence-to-sequence model (Seq2Seq) (Sutskever, Vinyals, & Le, 2014), which is constructed by having one RNN at the source side, and one RNN at the target side. This model can be used for machine translation (Cho et al., 2014; Sutskever et al., 2014), text summarization (Nallapati, Zhou, Gulcehre, Xiang, et al., 2016),

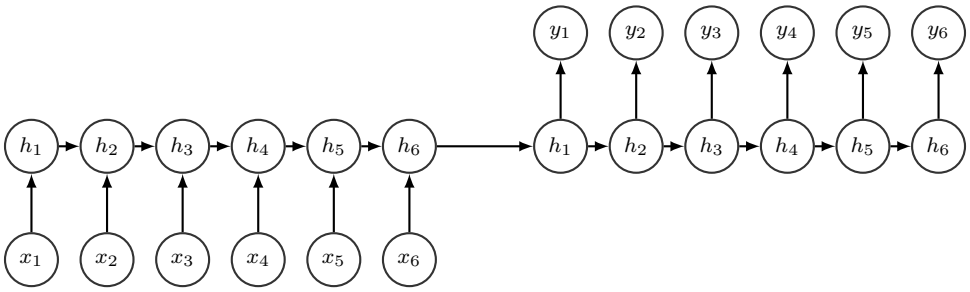


Figure A.32. A Simple sequence-to-sequence model

paraphrase generation (Prakash et al., 2016), and dialogue generation (J. Li et al., 2017). Figure A.32 shows a Seq2Seq model.

In sequence-to-sequence generation tasks, each input sequences, x, is coupled with an output sequence to predict y, both of which are modeled by an RNN. Specifically, for machine translation, the input is a sequence of words in the source language and the output is a sequence of words in the target language.

The initial state of the decoding RNN comes from the representation learned by the encoder. But in attention-based translation (Bahdanau et al., 2014), the initial state at the decoder is given by a weighted combination of the entire source, which is known as the context vector. Concretely, given the target hidden state h_t, and the source-side context vector c_t, produce an attentional hidden state as follows:

$$h_t = \tanh(W_c[c_t; h_t])$$

Character-Level NLP Models

Character-level models and those based on sub-word units are able to capture morphological patterns, which leads to improving generalization to unseen words (Luong & Manning, 2016; Sennrich, Haddow, & Birch, 2016). Deep character models have been successfully applied in several NLP tasks (Ballesteros, Dyer, & Smith, 2015; Costa-Jussa & Fonollosa, 2016; Kim et al., 2016; Lample, Ballesteros, Subramanian, Kawakami, & Dyer, 2016; Ling, Luís, et al., 2015; Ling, Trancoso, Dyer, & Black,

2015; Ma & Hovy, 2016; X. Zhang et al., 2015). One can use characters to create word embeddings, or use them stand-alone by representing text as a stream of characters. Extracting features from character sequences can be done with convolutional or recurrent neural nets. Another distinction between character-aware models is whether the input is segmented at the word level. Fully character models in translation with no word-level segmentation have been found to work well (Chung, Cho, & Bengio, 2016; Lee, Cho, & Hofmann, 2017).

CharCNN-LSTM Architecture. The architecture we study is based on the one proposed by Kim et al. (2016) for character-level language modeling. Feature extraction is performed by convolutions over characters, which are passed to layers of highway networks, and finally given to stacks of recurrent neural nets for modeling a sequence of words.

This architecture is flexible to perform multiple tasks. For example, in sequence labeling (Kim et al., 2016) the output of every recurrent unit is passed to a softmax layer to predict the next word. In this paper, we use this architecture on the encoder side of a seq-to-seq model, as demonstrated by Costa-Jussa and Fonollosa (2016) for machine translation. We also use this architecture for our text classification experiments, wherein the output of the last recurrent unit is passed to a softmax to predict the label of text. We briefly explain this architecture and refer the reader to Kim et al. (2016) for more details. See Figure A.33.

Let V be the alphabet, and let $x_{ij} \in \{0,1\}^{|V|}$ denote a one-hot vector representing the j-th character of the i-th word. The character sequence can be represented by

$$x = [(x_{11},.. \ x_{1n});..(x_{m1},.. \ x_{mn})]$$

wherein a semicolon denotes explicit segmentation between words. The number of words is denoted by m, and n is the number of maximum characters allowed for a

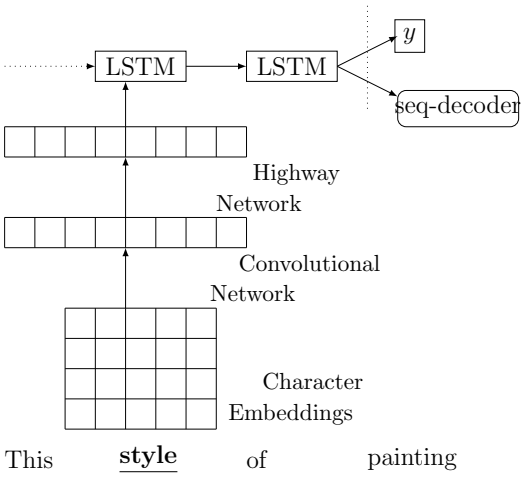


Figure A.33. The CharCNN-LSTM architecture used for studying two tasks of text classification and translation.

word[1]. The features associated with character n-grams are extracted by convolutional neural networks (CNNs). The one-hot vectors are multiplied by the embedding matrix to get the individual character embeddings. For each word, narrow convolutions between the sequence of the character embeddings and followed by a nonlinearity, are applied to obtain the feature maps. Next, a temporal max-pooling is applied, the output of which is given to 2 layers of highway networks. Finally, the output of the highway networks is given to a 2-layer stacked LSTMs.

Graph Neural Networks

In this section, we cover related work in the area of graph neural networks. Graph convolutional network (GCN) is a recent development, which aims to model relational data using neural nets. It is a representation learning scheme for graph-structured data; this model has been applied to node classification (Kipf & Welling, 2017), link prediction (Berg et al., 2017; Schlichtkrull et al., 2017), and NLP (Marcheggiani & Titov, 2017) with success. It is a simplification of the model proposed by Defferrard et

[1]Padding is applied if the number of characters is fewer than the maximum.

al. (Defferrard et al., 2016), which approximates smooth filters in the spectral domain, using Chebyshev polynomials with learnable parameters in a neural model. Another difference between the two models is that GCN is used to learn node representations, while the convolutional network in (Defferrard et al., 2016) is used to generate a representation for the whole graph.

Graph attention networks (GAT) (Velickovic et al., 2018) and attention-based graph neural network (AGNN) (Thekumparampil et al., 2018) provide a self-attention mechanism, where the weight of each neighbor's contribution to the embedding comes from an attention function, which uses the nodes' own representation and the neighbors'. J. Zhang et al. (2018) extend GAT by performing gating on the attention heads of GAT. GeniePath (Z. Liu et al., 2018) uses a combination of attention and memory architecture to adaptively pick features from neighbors in deeper layers. Abu-El-Haija et al. (2018) train multiple instances of GCNs over node pairs discovered at different distances in random walks. Du et al. (2017) define graph convolution operation on the vertex domain as multiplication by polynomials of the graph adjacency matrix. In this model, filters are adaptive to the topology of the graph as they scan the graph to perform convolution.

Mean field networks (MFNs) (Y. Li & Zemel, 2014) are hierarchical probabilistic models, in which variables in the hierarchy are connected based on an underlying structure. These networks can be regarded as unrolled instantiations of mean field inference in graphical models, in which the iterative inference procedure is replaced with layers that resemble feed-forward networks. A few works have used probabilistic inference to study inference in graph neural networks. Most notably, Dai, Dai, and Song (2016) propose a latent variable model to learn node representations (embeddings), inspired by mean-field and loopy belief propagation inference

procedures. They use the assumption of an injective embedding (Sriperumbudur, Gretton, Fukumizu, Schölkopf, & Lanckriet, 2010) on node embeddings, and they parameterize the embeddings with a neural net. Their model has only one layer which makes it less powerful than GCN. Using similar machinery, Graphite (Grover, Zweig, & Ermon, 2018) performs unsupervised learning of representations over nodes in a graph using deep latent variable generative models.

Shallow embedding learning methods have been extensively studied in the past few years. DeepWalk (Perozzi, Al-Rfou, & Skiena, 2014) samples a set of paths from the input graph using truncated random walks, and uses SkipGram modeling (Mikolov, Chen, Corrado, & Dean, 2013) to maximize the co-occurrence probability among the nodes that appear in a walk. Line (Tang et al., 2015) learns half of the dimensions of the embedding by using immediate neighbors of a node, and the other half using the two-hop neighbors. Node2vec (Grover & Leskovec, 2016) performs a more guided neighborhood search which explores more diverse neighborhoods. These methods outperform GCN and its variants in learning representations for planar graphs, but are not designed to perform as well on graphs which have features on nodes. In addition, all these methods require a two-phase pipeline, including learning embeddings and training a model for a downstream task. Yang et al. (2016) address this problem by injecting label information in embedding learning; however, the method is still not competitive with GCN on feature-based graphs.

Node classification can be studied in *transductive* and *inductive* settings; in the former the model has access to the whole graph, including the test nodes, and uses that as extra knowledge. In the latter, the model does not see test nodes in training, and will only add the test nodes to the graph at test time. See Figure A.34 for an illustration. For our transductive node classification experiments, we use graph

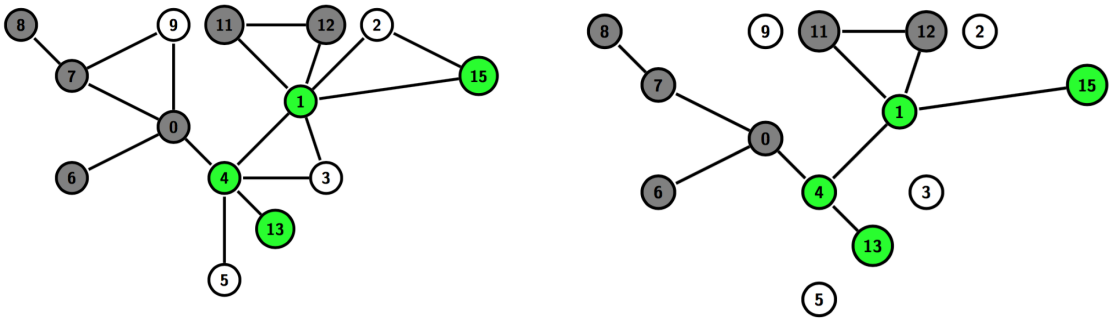


Figure A.34. Illustration of transductive node classification (left) and inductive node classification (right). The white nodes are the test nodes, which are to be classified to one of the two classes colored green and gray.

convolutional networks (Kipf & Welling, 2017), and for our inductive experiments, we use GraphSage (Hamilton et al., 2017).

Graph Convolutional Neural Networks. A Graph Convolutional Neural Network (Kipf & Welling, 2017) learns representations from structured data given by a graph $G = (V, E)$. A GCN can be regarded as an encoder $e_\theta(A, X)$, which takes the adjacency matrix, A, and a $|V| \times m$ feature matrix, X, and produces a $|V| \times n$ embedding matrix, also called hidden representations. For example, the representation of the graph at layer $l + 1$ can be given by:

$$h^{(l+1)} = \text{ReLU}(Ah^{(l)}W^l) \tag{A.4}$$

where W^l is the weights for the l_{th} layer. The multiplication with the adjacency matrix could bring about scaling problems, and hence A is normalized. Concretely, each outgoing edge, A_{ij}, can be normalized by the number of neighbors of that node $1/|\mathcal{N}_i|$, or by a combination of both nodes involved in the edge, $1/\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}$.

One can view the local graph convolution in Equation A.4 as a differentiable message passing, where vector-valued messages are being passed across the graphs and transformed at nodes. The final hidden representation can be passed to other

layers for specific tasks, such as maximum entropy for classification (Kipf & Welling, 2017; Schlichtkrull et al., 2017), or a decoder for link prediction (Berg et al., 2017). If K different types of relations exist in the data, then there will be $A^r, r \in 1, 2, ..K$ adjacency matrices, and K different sets of weights for different relations.

Sample and Aggregate. One of the drawbacks of GCNs is that they require full-graph batch optimization with a sparse-dense matrix multiplication, which will cause memory problems in large graphs. Concretely, they were initially used for transductive settings, where all the nodes of the graph, including test nodes, are seen in training. To address this issue, several sampling-based approaches have been proposed, which would sample a small number of nodes at each layer during mini-batch training (Chen et al., 2018; Chen & Zhu, 2018; Hamilton et al., 2017), and aggregate them. GraphSAGE (Hamilton et al., 2017) uses a sampling algorithm to select a subset of the neighbors from one-hop and two-hop neighborhoods. Embedding for the node i in a two-layer GraphSage with a mean aggregator can be represented as the following.

$$H_i = W_2\Big[\overbrace{\text{ReLU}(W_1[x_i; \alpha \sum_{j \sim \mathcal{N}(i)} a_j X])}^{h_i^1}; \alpha \sum_j \overbrace{\text{ReLU}(W_1[a_j X; \beta \sum_{k \sim \mathcal{N}(j)} a_k X])}^{h_j^1}\Big]$$

where a_j is the one hot-vector representing the jth neighbor, and α and β are normalization constants.

Figure A.35 shows a graph where nodes from first-hop and second-hop neighbors are involved in learning embeddings for the green node. Node two is not sampled and will not contribute to the embedding produced for the green node. The main idea behind these sampling-based approaches is that we only need to create the embedding of the nodes in the mini-batch to be able to do stochastic gradient descent. FastGCN (Chen et al., 2018) subsamples the receptive field in each layer universally, using

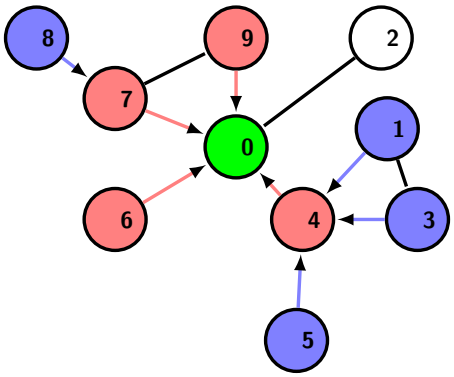


Figure A.35. Message Passing in GCN or GraphSage, using two-hop neighbors (blue) and one-hop neighbors (red) to the target node (green). In GraphSage not all neighbors contribute to the representation of the target node. For instance, the white node does not contribute to learning the representation of the target node.

importance sampling. Chen and Zhu (2018) develop a control variate-based stochastic approximation, which utilizes historical activations of nodes as a control variate.

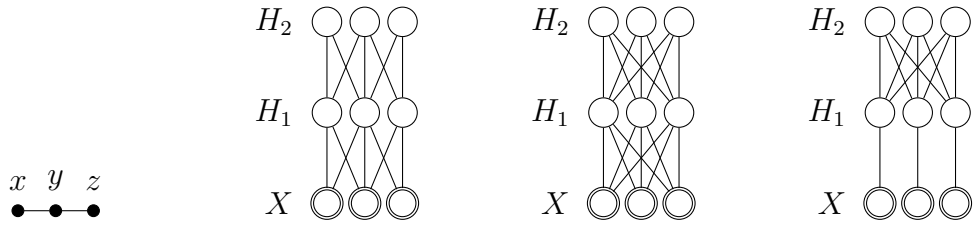
APPENDIX B

ROBUST GRAPH NEURAL NETS

Loopy-BP-Inspired Message Passing

In this section, we describe our modified message passing scheme for our transductive experiments in Chapter V. The datasets are small and are sparsely labeled, and we found that increasing the receptive field by incorporating longer range dependencies, e.g., two-hop neighbors, can be helpful. Note that we still need E computations, where E is the number of edges. Specifically, we combine messages coming from the two-hop neighbors at the corresponding neighbor, before feeding them to the gating function. That is, j will send its neighbor i, the message $\alpha(x_j + \sum_{k \in \mathcal{N}(j), k \neq i} x_k)$, where α is a normalization constant. This means node j is sending a message to a particular neighbor by combining all of its neighbors' embeddings and its own embedding, excluding the receiving neighbor's.

This message passing scheme is inspired by loopy belief propagation (BP) in graphical models; in loopy BP, a message from a node a to a neighbor v is the accumulated messages from all other nodes, marginalized over all variables except v. Figure B.36c depicts a setting where nodes at the first layer will send messages to their second-hop neighbors directly. Our best results used the scheme in B.36d,



(a) Simple 3-node chain (b) Standard (c) Loopy BP inspired (d) Final model

Figure B.36. A graphical representation contrasting standard message passing among immediate neighbors and loopy-bp-inspired message passing. Nodes from lower layers send messages to the higher ones.

wherein two-hop-based message passing scheme exists between layer 1 and layer 2, while using no message passing between nodes at layer 0 and layer 1. That is, the first layer of our GCN is simply a one-layer neural net, and a GCN with a bigger receptive field is at the next layer.

Robust Optimization for GraphSage

The linear program, described in 6.7, can be solved without the use of linear program solvers. Concretely, the dual of the program can be solved by another neural network which is very similar to the backprop pass of the original network. We associate the following dual variables with each of the constraints

$$
\begin{aligned}
& r^i = W_f \hat{h}_2^i + b_f \rightarrow \nu \\
& \hat{h}_2^i = W_2[h_1^i; \alpha \sum_{j \sim \mathcal{N}(i)} h_1^j] \rightarrow \eta_2 \\
& \hat{h}_1^i = W_1[\hat{x}_i; \alpha \sum_{j \sim \mathcal{N}(i)} \hat{x}_j] \rightarrow \eta_1 \\
& \hat{h}_1^j = W_1[\hat{x}_j; \beta \sum_{k \sim \mathcal{N}(j)} \hat{x}_k] \rightarrow \gamma \\
& -h_{1,d}^z \leq 0 \rightarrow \mu_d^z \\
& \hat{h}_{1,d}^z - h_{1,d}^z \leq 0 \rightarrow \tau_d^z \\
& (u_d^z - l_d^z) h_{1,d}^z - u_d^z \hat{h}_{1,d}^z \leq -u_d^z l_d^z \rightarrow \lambda_d^z \\
& \begin{cases} -\hat{x}_j \leq -x_j + \epsilon_1 \rightarrow \xi_1^- \\ \hat{x}_j \leq x_j + \epsilon_1 \rightarrow \xi_1^+ \end{cases} \quad j \sim \mathcal{N}(i) \\
& \begin{cases} -\hat{x}_k \leq -x_k + \epsilon_2 \rightarrow \xi_2^- \\ \hat{x}_k \leq x_k + \epsilon_2 \rightarrow \xi_2^+ \end{cases} \quad k \sim \mathcal{N}(j) \\
& \hat{x}_i = x_i \rightarrow \xi_3
\end{aligned}
\tag{B.1}
$$

Now the dual problem becomes:

$$\text{maximize } (-\nu^T b_f \ + \sum_d \sum_z \lambda_d^z u_d^z l_d^z +$$

$$(x_j - \epsilon_1)^T \xi_1^- - (x_j + \epsilon_1)^T \xi_1^+ + (x_k - \epsilon_2)^T \xi_2^- - (x_k + \epsilon_2)^T \xi_2^+ - x_i^T \xi_3)$$

subject to:

$$\nu = e_{y^{target}} - e_{y^*}$$

$$\eta_2 = W_f^T \nu$$

$$\begin{cases} \begin{cases} \eta_1^d = 0, d \in \mathcal{I}_i^- \\ \gamma^d = 0, d \in \mathcal{I}_j^- \end{cases} \\ \begin{cases} \eta_1^d = (V_2^T \eta_2)_d, d \in \mathcal{I}_i^+ \\ \gamma^d = \alpha (U_2^T \eta_2)_d, d \in \mathcal{I}_j^+ \end{cases} \\ d \in \mathcal{I}_i \quad \begin{cases} \left(\lambda_d^i (u_d^i - l_d^i) - \mu_d^i - \tau_d^i\right) = (V_2^T \eta_2)_d \\ \eta_1^d = \lambda_d^i u_d^i - \tau_d^i \end{cases} \\ d \in \mathcal{I}_j \quad \begin{cases} \left(\lambda_d^j (u_d^j - l_d^j) - \mu_d^j - \tau_d^j\right) = \alpha (U_2^T \eta_2)_d \\ \gamma^d = \lambda_d^j u_d^j - \tau_d^j \end{cases} \end{cases} \tag{B.2}$$

$$\xi_1^+ - \xi_1^- = V_1^T \gamma + \alpha U_1^T \eta_1$$

$$\xi_2^+ - \xi_2^- = \beta U_1^T \gamma$$

$$\xi_3 = V 1^T \eta_1$$

$$\lambda, \tau, \mu, \xi_1^+, \xi_1^-, \xi_2^+, \xi_2^-, \xi_3 \geq 0$$

We break W into two submatrices V, i.e., operating on the node and U, i.e., operating on neighbors. Concretely,

$$W_1[x_i; \alpha \sum_{j \sim \mathcal{N}(i)} x_j] = [V_1 x_i; U_1 \alpha \sum_{j \sim \mathcal{N}(i)} x_j] \tag{B.3}$$

We overload the notation and use η_2, η_1, and γ to refer to both halves of these variables. That is, for ease in exposition, we do not differentiate between the two parts that are being concatenated at each layer.

Kolter and Wong (2017) observe that due to the complementary slackness property in convex optimization, either λ (i.e., Lagrange multiplier for the upper bound constraint) or $\mu+\tau$ (i.e., Lagrange multiplier for either of the lower bound constraints) must be zero. After re-writing some of the constraints, they are able to derive dual variables as positive and negative portions of a given vector, and represent the dual problem as a backward pass with leaky ReLU activation functions. For our problem, $\mathcal{I}$, and the dual variables η_1, γ can be represented as following.

$$\eta_1^d = \begin{cases} 0 & d \in \mathcal{I}_i^- \\ (V_2^T \eta_2)_d & d \in \mathcal{I}_i^+ \\ \frac{u_d^i}{u_d^i - l_d^i}[(V_2^T \eta_2)_d]_+ - \delta_d^i [(V_2^T \eta_2)_d]_- & d \in \mathcal{I}_i \end{cases} \tag{B.4}$$

$$\gamma^d = \begin{cases} 0 & d \in \mathcal{I}_j^- \\ \alpha(U_2^T \eta_2)_d & d \in \mathcal{I}_j^+ \\ \frac{u_d^j}{u_d^j - l_d^j}[\alpha(U_2^T \eta_2)_d]_+ - \delta_d^j [\alpha(U_2^T \eta_2)_d]_- & d \in \mathcal{I}_j \end{cases} \tag{B.5}$$

where δ_d^i is defined as $\frac{u_d^i}{u_d^i - l_d^i}$, and $[]_+$ and $[]_-$ return positive and negative values of an input, respectively. Note that we create representations of nodes in the minibatch,

and their first-hop neighbors in the first layer. This is the only part of the network that contains nonlinearity, and thus, we will have upper and lower bound for all activations for only these nodes. The upper and lower bound computation follows the procedure described by Kolter and Wong (2017). In addition, GraphSage uses a normalization on the logits before they are given to the classifier. For this normalization, we tried both layer normalization (Ba et al., 2016) and batch-normalization (Ioffe & Szegedy, 2015), the latter of which performed better. Batch normalization can be represented by a linear layer, and hence its dual layer can be easily incorporated (Wong, Schmidt, Metzen, & Kolter, 2018). We omitted that layer for the sake of clarity. The final objective function that we need to maximize is:

$$\begin{aligned} J(\eta_1, \eta_2, \nu, \gamma) = \big(& -\nu^T b_f + \\ & - (x_j + \epsilon_1)^T [V_1^T \gamma + \alpha U_1^T \eta_1]_+ + (x_j - \epsilon_1)^T [V_1^T \gamma + \alpha U_1^T \eta_1]_- \\ & - (x_k + \epsilon_2)^T [\beta U_1^T \gamma]_+ + (x_k - \epsilon_2)^T [\beta U_1^T \gamma]_- - x_i^T V 1^T \eta_1 \\ & \sum_{d \in \mathcal{I}_i} l_d^i [\eta_1^d]_+ + \sum_{d \in \mathcal{I}_j} l_d^j [\gamma^d]_+ \big) \end{aligned} \tag{B.6}$$

Which can further be simplified to:

$$\begin{aligned} J(\eta_1, \eta_2, \nu, \gamma) = \big(& -\nu^T b_f + \\ & - x_j^T (V_1^T \gamma + \alpha U_1^T \eta_1) - \beta x_k^T U_1^T \gamma - x_i^T V 1^T \eta_1 \\ & - \epsilon_1 \parallel V_1^T \gamma + \alpha U_1^T \eta_1 \parallel_1 - \epsilon_2 \parallel \beta U_1^T \gamma \parallel_1 \\ & \sum_{d \in \mathcal{I}_i} l_d^i [\eta_1^d]_+ + \sum_{d \in \mathcal{I}_j} l_d^j [\gamma^d]_+ \big) \end{aligned} \tag{B.7}$$

Jacobian Regularization with Layer Normalization

Layer Normalization performs instance-wise normalization over inputs to a neural net, as described in by Ba et al. (2016). Concretely, the input to a nonlinearity goes

through the following normalization layer:

$$LNorm(x) = \frac{x - E(x)}{\sqrt{Var(x)}} * \zeta + \psi$$

where $E(x)$ and $Var(x)$ denote the mean and variance over the features of a single instance. When we use layer normalization, the Jacobian will have the following form.

$$\mathbb{J}_{\sigma(\alpha U_1 \sum_{j\sim\mathcal{N}(i)} x_j)}(x_j) = U_1^T \odot \sigma^{'}(LNorm(U_1\alpha \sum_{j\sim\mathcal{N}(i)} x_j)) \odot \frac{\zeta}{\sqrt{Var(U_1\alpha \sum_{j\sim\mathcal{N}(i)} x_j)}} \tag{B.8}$$

So here, both ζ and U_1 are going to be regularized.

APPENDIX C

NOTATION AND SYMBOLS

θ	Model parameters
W	Weight matrix
X	Dataset inputs
Y	Dataset outputs
x_i	Input data point
x_{ij}	jth character of the ith word
y_i	Output data point
h	Hidden layer
A	Adjacency matrix
a_j	One-hot vector with one at the jth position
g	Neural net output
J	Loss function
$\mathbb{J}$	Jacobian
L	Number of network layers
ϵ	Perturbation magnitude
δ	Perturbation vector
α	Normalization constant
β	Normalization constant
σ	Nonlinearity
V	Graph nodes
V	Vocabulary
E	Graph edges
$\odot$	Element-wise product operator
$< . >$	Dot product operator

REFERENCES CITED

Abu-El-Haija, S., Kapoor, A., Perozzi, B., & Lee, J. (2018). N-GCN: Multi-scale graph convolution for semi-supervised node classification. *arXiv preprint arXiv:1802.08888*.

Adamic, L. A., & Glance, N. (2005). The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery* (pp. 36–43).

Athalye, A., Carlini, N., & Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of International Conference on Machine Learning.*

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations.*

Ballesteros, M., Dyer, C., & Smith, N. A. (2015). Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of Empirical Methods in Natural Language Processing.*

Belinkov, Y., & Bisk, Y. (2018). Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations.*

Berg, R. v. d., Kipf, T. N., & Welling, M. (2017). Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*.

Berger, Y. (2017). Israel arrests Palestinian because Facebook translated 'good morning' to 'attack them'.

Brendel, W., & Bethge, M. (2017). Comment on" biologically inspired protection of deep networks from adversarial attacks". *arXiv preprint arXiv:1704.01547*.

Carlini, N., & Wagner, D. (2017a). Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th acm Workshop on Artificial Intelligence and Security* (pp. 3–14).

Carlini, N., & Wagner, D. (2017b). Towards evaluating the robustness of neural networks. In *Security and Privacy, IEEE European Symposium on.*

Chancellor, S., Pater, J. A., Clear, T., Gilbert, E., & De Choudhury, M. (2016). # thyghgapp: Instagram content moderation and lexical variation in pro-eating disorder communities. In *Proceedings of Conference on Computer-Supported Cooperative Work and Social Computing* (pp. 1201–1213).

Chen, J., Ma, T., & Xiao, C. (2018). FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations.*

Chen, J., & Zhu, J. (2018). Stochastic training of graph convolutional networks. In *International Conference on Learning Representations.*

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of Empirical Methods in Natural Language Processing.*

Chung, J., Cho, K., & Bengio, Y. (2016). A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of Association of Computational Linguistics.*

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of International Conference on Machine Learning* (pp. 160–167).

Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2017). Very deep convolutional networks for natural language processing. In *Proceedings of European Chapter of Association of Computational Linguistics.*

Costa-Jussa, M. R., & Fonollosa, J. A. (2016). Character-based neural machine translation. In *Proceedings of Association of Computational Linguistics.*

Dai, H., Dai, B., & Song, L. (2016). Discriminative embeddings of latent variable models for structured data. In *Proceedings of International Conference on Machine Learning* (pp. 2702–2711).

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., & Song, L. (2018). Adversarial attack on graph structured data. In *Proceedings of International Conference on Machine Learning.*

Dalvi, N., Domingos, P., Sanghai, S., Verma, D., et al. (2004). Adversarial classification. In *Proceedings of Conference on Knowledge Discovery and Data Mining* (pp. 99–108).

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017). Language modeling with gated convolutional networks. In *Proceedings of International Conference on Machine Learning.*

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of Advances in Neural Information Processing Systems* (pp. 3844–3852).

Dong, Y., Liao, F., Pang, T., Su, H., Hu, X., Li, J., & Zhu, J. (2018). Boosting adversarial attacks with momentum. In *Proceedings of Conference on Computer Vision and Pattern Recognition.*

Douceur, J. R. (2002). The sybil attack. In *Proceedings of international Workshop on Peer-to-Peer Systems* (pp. 251–260).

Du, J., Zhang, S., Wu, G., Moura, J. M., & Kar, S. (2017). Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., & Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. In *Proceedings of Association of Computational Linguistics.*

Ebrahimi, J., Lowd, D., & Dou, D. (2018). On adversarial examples for character-level neural machine translation. In *Proceedings of the 27th international conference on computational linguistics* (pp. 653–663).

Ebrahimi, J., Rao, A., Lowd, D., & Dou, D. (2018). Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 2: Short papers)* (Vol. 2, pp. 31–36).

Ehlers, R. (2017). Formal verification of piece-wise linear feed-forward neural networks. In *Proceedings of International Symposium on Automated Technology for Verification and Analysis* (pp. 269–286).

Feinman, R., Curtin, R. R., Shintre, S., & Gardner, A. B. (2017). Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*.

Gantner, Z., Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2011). MyMediaLite: A free recommender system library. In *Proceedings of RecSys.*

Garey, M. R., & Johnson, D. S. (2002). *Computers and intractability* (Vol. 29). W.H. Freeman.

Ghosh, S., Viswanath, B., Kooti, F., Sharma, N. K., Korlam, G., Benevenuto, F., ... Gummadi, K. P. (2012). Understanding and combating link farming in the twitter social network. In *Proceedings of World Wide Web* (pp. 61–70).

Globerson, A., & Roweis, S. (2006). Nightmare at test time: robust learning by feature deletion. In *Proceedings of International Conference on Machine Learning* (pp. 353–360).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems* (pp. 2672–2680).

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations.*

Grosse, K., Manoharan, P., Papernot, N., Backes, M., & McDaniel, P. (2017). On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*.

Grosse, K., Papernot, N., Manoharan, P., Backes, M., & McDaniel, P. (2017). Adversarial perturbations against deep neural networks for malware classification. In *International Conference on Learning Representations.*

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of Conference on Knowledge Discovery and Data Mining* (pp. 855–864).

Grover, A., Zweig, A., & Ermon, S. (2018). Graphite: Iterative generative modeling of graphs. *arXiv preprint arXiv:1803.10459*.

Gu, S., & Rigazio, L. (2015). Towards deep neural network architectures robust to adversarial examples. In *International Conference on Learning Representations.*

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proceedings of Advances in Neural Information Processing Systems* (pp. 1025–1035).

Harper, F. M., & Konstan, J. A. (2016). The movielens datasets: History and context. *Transactions on Interactive Intelligent Systems*, *5*(4), 19.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.

Hosseini, H., Kannan, S., Zhang, B., & Poovendran, R. (2017). Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.

Hug, N. (2017). *Surprise, a Python library for recommender systems.* http://surpriselib.com.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning.*

Iyyer, M., Wieting, J., Gimpel, K., & Zettlemoyer, L. (2018). Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of North American Chapter of Association of Computational Linguistics.*

Jia, R., & Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. In *Proceedings of Empirical Methods in Natural Language Processing.*

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of Empirical Methods in Natural Language Processing.*

Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016). Character-aware neural language models. In *Proceedings of Association for the Advancement of Artificial Intelligence.*

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations.*

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations.*

Kolter, J. Z., & Wong, E. (2017). Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of International Conference on Machine Learning.*

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of Conference on Knowledge Discovery and Data Mining* (pp. 426–434).

Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data*, *4*(1).

Kurakin, A., Goodfellow, I., & Bengio, S. (2017a). Adversarial examples in the physical world. In *International Conference on Learning Representations.*

Kurakin, A., Goodfellow, I., & Bengio, S. (2017b). Adversarial machine learning at scale. In *International Conference on Learning Representations.*

Kurata, G., Xiang, B., Zhou, B., & Yu, M. (2016). Leveraging sentence-level information with encoder lstm for natural language understanding. In *Proceedings of Empirical Methods in Natural Language Processing.*

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of North American Chapter of Association of Computational Linguistics.*

Lee, J., Cho, K., & Hofmann, T. (2017). Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, *5*, 365–378.

Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., & Jurafsky, D. (2017). Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Li, Y., & Zemel, R. (2014). Mean-field networks. *ICML 2014 Workshop on Learning Tractable Probabilistic Models*.

Lin, F., & Cohen, W. W. (2010). Semi-supervised classification of network data using very few labels. In *Proceedings of International Conference on Advances in Social Networks Analysis and Mining* (pp. 192–199).

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., ... Trancoso, I. (2015). Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of Empirical Methods in Natural Language Processing.*

Ling, W., Trancoso, I., Dyer, C., & Black, A. W. (2015). Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Liu, Y., Chen, X., Liu, C., & Song, D. (2017). Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations.*

Liu, Z., Chen, C., Li, L., Zhou, J., Li, X., & Song, L. (2018). Geniepath: Graph neural networks with adaptive receptive paths. *arXiv preprint arXiv:1802.00910*.

Lowd, D., & Meek, C. (2005). Adversarial learning. In *Proceedings of Conference on Knowledge Discovery and Data Mining* (pp. 641–647).

Luong, M.-T., & Manning, C. D. (2016). Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of Association of Computational Linguistics.*

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of Empirical Methods in Natural Language Processing.*

Ma, X., & Hovy, E. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of Association of Computational Linguistics.*

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations.*

Marcheggiani, D., & Titov, I. (2017). Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of Empirical Methods in Natural Language Processing.*

Mauro, C., Jan, N., Sebastian, S., Luisa, B., Roldano, C., & Marcello, F. (2016). The iwslt 2016 evaluation campaign. In *International Workshop on Spoken Language Translation.*

Metzen, J. H., Genewein, T., Fischer, V., & Bischoff, B. (2017). On detecting adversarial perturbations. In *International Conference on Learning Representations.*

Michel, P., & Neubig, G. (2018). Mtnt: A testbed for machine translation of noisy text. In *Proceedings of Empirical Methods in Natural Language Processing.*

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of Interspeech.*

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems* (pp. 3111–3119).

Miyato, T., Dai, A. M., & Goodfellow, I. (2017). Adversarial training methods for semi-supervised text classification. In *International Conference on Learning Representations.*

Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K., & Ishii, S. (2016). Distributional smoothing with virtual adversarial training. In *International Conference on Learning Representations.*

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., & Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Conference on Computer Vision and Pattern Recognition.*

Moosavi-Dezfooli, S.-M., Fawzi, A., & Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of Computer Vision and Pattern Recognition* (pp. 2574–2582).

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of International Conference on Machine Learning* (pp. 807–814).

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of Conference on Computational Natural Language Learning.*

Nayebi, A., & Ganguli, S. (2017). Biologically inspired protection of deep networks from adversarial attacks. *arXiv preprint arXiv:1703.09202*.

Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of Conference on Computer Vision and Pattern Recognition* (pp. 427–436).

Oord, A. v. d., Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *Proceedings of International Conference on Machine Learning.*

Papernot, N., McDaniel, P., & Goodfellow, I. (2016). Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of Asia Conference on Computer and Communications Security* (pp. 506–519).

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016). The limitations of deep learning in adversarial settings. In *Security and Privacy IEEE European Symposium on* (pp. 372–387).

Papernot, N., McDaniel, P., Swami, A., & Harang, R. (2016). Crafting adversarial input sequences for recurrent neural networks. In *Proceedings of Military Communications Conference* (pp. 49–54).

Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy* (pp. 582–597).

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of Association of Computational Linguistics* (pp. 311–318).

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning* (pp. 1310–1318).

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing* (pp. 1532–1543).

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of Conference on Knowledge Discovery and Data Mining* (pp. 701–710).

Prakash, A., Hasan, S. A., Lee, K., Datla, V., Qadir, A., Liu, J., & Farri, O. (2016). Neural paraphrase generation with stacked residual lstm networks. In *Proceedings of Computational Linguistics.*

Raghunathan, A., Steinhardt, J., & Liang, P. (2018). Certified defenses against adversarial examples. In *International Conference on Learning Representations.*

Rawlinson, G. E. (1976). *The significance of letter position in word recognition.* (Unpublished doctoral dissertation). University of Nottingham.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of Association for Computational Linguistics* (pp. 856–865).

Sak, H., Senior, A. W., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Procedings of Interspeech* (pp. 338–342).

Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. v. d., Titov, I., & Welling, M. (2017). Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3), 93.

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. *Proceedings of Association of Computational Linguistics*.

Shaham, U., Yamada, Y., & Negahban, S. (2015). Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Sinha, A., Namkoong, H., & Duchi, J. (2018). Certifying some distributional robustness with principled adversarial training. In *International Conference on Learning Representations.*

Song, Y., Kim, T., Nowozin, S., Ermon, S., & Kushman, N. (2018). Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations.*

Song, Y., Shu, R., Kushman, N., & Ermon, S. (2018). Generative adversarial examples. In *Proceedings of Advances in Neural Information Processing Systems.*

Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., & Lanckriet, G. R. (2010). Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, *11*(Apr), 1517–1561.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958.

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems* (pp. 3104–3112).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations.*

Tabacof, P., & Valle, E. (2016). Exploring the space of adversarial images. In *2016 International Joint Conference on Neural Networks* (pp. 426–433).

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of World Wide Web* (pp. 1067–1077).

Teo, C. H., Globerson, A., Roweis, S. T., & Smola, A. J. (2008). Convex learning with invariances. In *Advances in Neural Information Processing Systems* (pp. 1489–1496).

Thekumparampil, K. K., Wang, C., Oh, S., & Li, L.-J. (2018). Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*.

Torkamani, M., & Lowd, D. (2013). Convex adversarial collective classification. In *Proceedings of International Conference on Machine Learning* (pp. 642–650).

Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., & McDaniel, P. (2018). Ensemble adversarial training: Attacks and defenses..

Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., & McDaniel, P. (2017). The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*.

van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016). Conditional image generation with pixelcnn decoders. In *Proceedings of Advances in Neural Information Processing Systems* (pp. 4790–4798).

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations.*

Wiseman, S., & Rush, A. M. (2016). Sequence-to-sequence learning as beam-search optimization. In *Proceedings of Empirical Methods in Natural Language Processing.*

Wong, E., Schmidt, F., Metzen, J. H., & Kolter, J. Z. (2018). Scaling provable adversarial defenses. *arXiv preprint arXiv:1805.12514*.

Wu, Y., Bamman, D., & Russell, S. (2017). Adversarial training for relation extraction. In *Proceedings of Empirical Methods in Natural Language Processing* (pp. 1778–1783).

Xiao, C., Li, B., Zhu, J.-Y., He, W., Liu, M., & Song, D. (2018). Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*.

Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., & Yuille, A. (2017). Adversarial examples for semantic segmentation and object detection. In *Proceedings of International Conference on Computer Vision.*

Xu, H., Caramanis, C., & Mannor, S. (2009). Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, *10*(Jul), 1485–1510.

Yang, Z., Cohen, W. W., & Salakhutdinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *Proceedings of International Conference on Machine Learning.*

Yang, Z., Wilson, C., Wang, X., Gao, T., Zhao, B. Y., & Dai, Y. (2014). Uncovering social network Sybils in the wild. *Transactions on Knowledge Discovery from Data*, *8*(1).

Zhang, J., Shi, X., Xie, J., Ma, H., King, I., & Yeung, D.-Y. (2018). Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Proceedings of Advances in Neural Information Processing Systems* (pp. 649–657).

Zhao, Z., Dua, D., & Singh, S. (2018). Generating natural adversarial examples. In *International Conference on Learning Representations.*

Zitnik, M., & Leskovec, J. (2017). Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, *33*(14), i190–i198.

Zubiaga, A., Liakata, M., Procter, R., Hoi, G. W. S., & Tolmie, P. (2016). Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, *11*(3), e0150989.