MIP MaxPool ReLU Networks for NLP Adversarial Example Generation Project Report

Team Death and Gravity

Jingyi Gao, Jialin Kang, Jingfeng Liu, Wilson Tang

May 1, 2020

Abstract

While the success of both GANs and MIP ReLU models have been shown to generate adversarial examples in DNNs for computer vision, adversarial example generation for NLP is still in it's early stages. We extend MIP ReLU adversarial example generation for Natural Language Processing (via CNNs and reverse word embeddings). Initial results suggest that as an aggregate MIP networks do generate useful adversarial examples for data augmentation; however, an observation generated adversarial examples are often not grammatically correct.

Introduction

Deep Neural Network (DNN) adversarial generation has been a hot topic, finding great success in the computer vision space. These examples traditionally have been trained using generalized adversarial networks (GAN) (heuristic generation). However, recent work combining the power of MIP has allowed for more precise and successful generation of adversarial examples using MIP ReLU models.[1]. In the field of deep learning in natural language processing (NLP), there have been recent attempts at replicating the success of GANs; however, generation of grammatically adversarial examples is difficult. [2][3]. On the other hand, there have been no attempts at applying MIP ReLU models to NLPs. We expand on previous work done to extend these MIP ReLU models in computer vision to basic NLP CNNs with the hope that precise control of adversarial examples can generate more successful examples[4]

Our hypothesis was that this precise control in permuting the word embedding matrices combined with reverse word embedding will allow us to generate adversarial examples that will change words to grammatically acceptable word. By taking advantage of the way word embeddings are designed to capture context, small permutations by a MIP would result in small changes substituting similar words that the network is susceptible to.

We initially started out with the IMBD dataset and 100 dimensional GloVe word embeddings, but quickly realized that the number of parameters would prove infeasible for our MIP network to solve. To address this issue, we simplified the dataset and model, by implementing a basic NLP 4 layers CNNs on the S140 Twitter binary sentiment dataset and used only tweets with a short enough length (25 word tokens). We then evaluated the 3

different models trained on half of the available data, all of the available data, and available data with MIP adversarial examples. Results were as expected: the augmented dataset was the best (80% val) compared to 71-72% for the others, indicating the ability for the MIP to generate examples that increase the robustness of NLP systems.

1 DNN and MIP ReLU Solver Overview

Due to computing constraints and MIP solver times we decided to implement a very simplistic system. As such we standardized the length of sentences/tweets provided and the n-dimension of the pre-trained GloVe layers. We decided on 25 for both to result in 25x25 dimensional word embeddings sentence matrices. These embeddings, the fully connected (FC) layer outputs, and best model parameters were extracted from the network and passed into the MIP ReLU solver. Finally, our reverse word embedder helps to create an augmented dataset that can be fed back into the model for experiments.

A system diagram is shown in Figure 1:

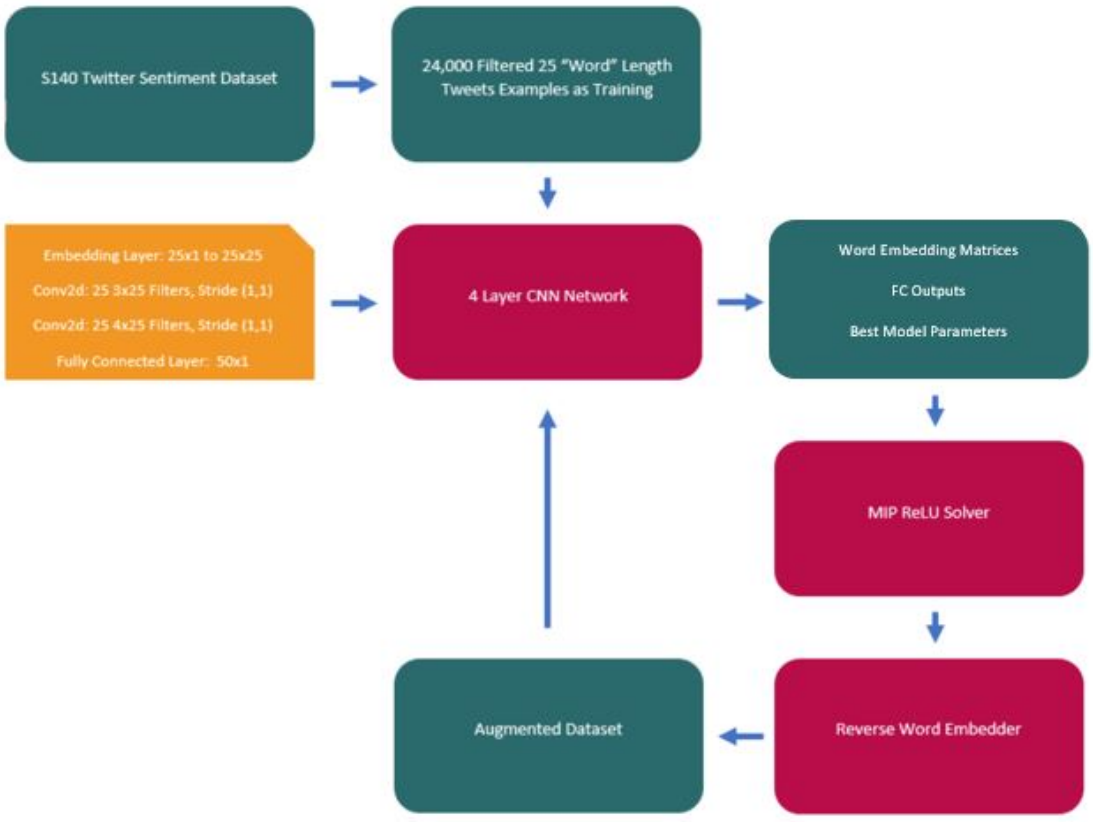

Figure 1: MIP ReLU Adversarial Example Generation for NLP System Overview

1.1 Simple NLP CNNs

The convolutional neural network (CNN) implemented was chosen for it's simplicity and similarity to common networks used for prior MIP work computer vision. [1][4]. A overview of our CNN is shown in hreffig:cnn

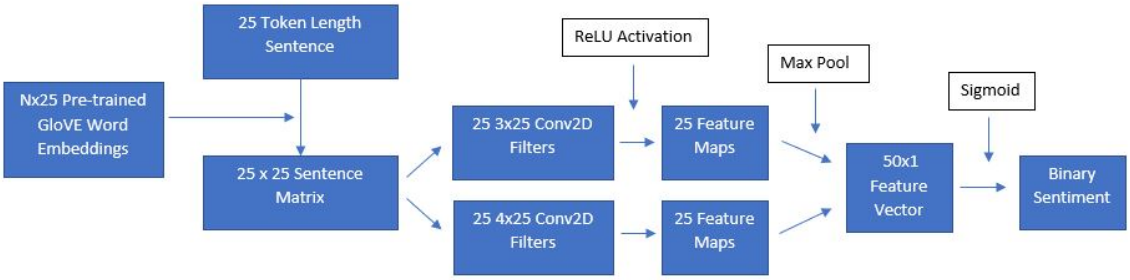

Figure 2: Overview of Convolutional Neural Network

The network takes in sentences/tweets from a filtered data set using the torchtext library, then it builds a torchtext vocab based on the all possible input data and a pre-trained GloVe (glove.twitter.27B.25d) parameters. This vocab allows our embedding layer (unique processing layer required for nlp and text data) to translate input text of word length 25 into 25x25 dimensional matrices and is larger than required in order to expand the possible range of values for our reverse word embedding step later. Following the embedding layer, these images are parallely processed by two different n-grams (3,4) two-dimensional convolutional layers, activated using ReLUs. This allows us to extract different 3 and 4 word segments as features, which are pooled using maxpool and passed into a fully connected layer which reduces the dimensionality and with sigmoid maps to our binary sentiments. ()

1.2 MIP ReLU Adversarial Solver

In our convolutional neural network, we have one embedding layer (ignored in this case), two convolutional layers in parallel, and one full connected layer. For each convolutional layer, we implement ReLU and Maxpool activation functions. Our goal is to generate adversarial examples with the help of MIP ReLU model, so all the weights and bias are available from trained CNN models.
Let $x, y \in R^{m,n}$ denote the original input and adversarial example. Let w_r^{ch}, b_r^{ch} denote weights and bias for channel ch in convolutional layer r, where $ch \in \{1, 2, ..., n\}, r \in \{1, 2\}$. So we will have the output $tp_{r,ch}$:

$$tp_{r,ch} = CNN(w_r^{ch}, y, b_r^{ch}) \tag{1}$$

where CNN represents how convolutional layer calculates the output. In short, it calculates element-wise productions using kernel of size $k_{ch} * n$ and sum these productions for each projection. The output size depends on the kernel size and some other parameters like stride. In our model, all other parameters are in default except kernel size, which are

$(3, n), (4, n)$ for convolutional layer 1, 2. Therefore, size of $tp_{r,ch}$ is $(l_r, 1)$.
Then we implement a ReLU layer on $tp_{r,ch}$, namely we have:

$$a^i_{r,ch} = max(0, tp^i_{r,ch}) \quad i = 1, 2, ..., l_r \tag{2}$$

In order to represent equation (2) in 0-1 MIP model, we rewrite (2) as:

$$\begin{aligned} tp^i_{r,ch} &= a^i_{r,ch} - c^i_{r,ch} \\ a^i_{r,ch} \geq 0, c^i_{r,ch} &\geq 0 \end{aligned} \tag{3}$$

Then we introduce a binary activation variable z and to impose the logical implications:

$$\begin{aligned} z^i_{r,ch} &\in \{0, 1\} \\ z^i_{r,ch} = 1 &\rightarrow a^i_{r,ch} \leq 0 \\ z^i_{r,ch} = 0 &\rightarrow c^i_{r,ch} \leq 0 \end{aligned} \tag{4}$$

The above "indicator constraints" are accepted in Gurobi. After the ReLU layer, we also need a Maxpool layer. Similar to equation (2)-(4), we can rewrite the maxpool function as:

$$\begin{aligned} tmp_{r,ch} = max(a^1_{r,ch}, a^2_{r,ch}, ..., a^{l_r}_{r,ch}) \\ tmp_{r,ch} \geq a^i_{r,ch} \\ \sum_{i=1}^{l_r} p^i_{r,ch} = 1 \\ p^i_{r,ch} = 1 \rightarrow tmp_{r,ch} \leq a^i_{r,ch} \\ p^i_{r,ch} \in \{0, 1\} \end{aligned} \tag{5}$$

The last step in our CNN model is to concatenate the outputs of two convolutional layers and pass them into the full connected layer, which can be represented as:

$$output = W^{FC} * tmp + b^{FC} \tag{6}$$

Because we need an adversarial example, so the final output should be different from the old one in that this is a binary prediction problem. Based on the characteristics of sigmoid function, we should add one more constraint:

$$output * x_{FC} \leq 0 \tag{7}$$

where x_{FC} represents the value of CNN model after full connected layer.
At last, we also need an objective function for our model. We want our new adversarial example is not far away from the original input. Therefore, we decide to use L2 norm of the two examples as our objective, namely:

$$\underset{y}{\text{minimize}} \quad \sum_i \sum_j (y_{i,j} - x_{i,j})^2 \tag{8}$$

To conclude, equations (1)-(8) are our MIP model in general.
However, our MIP model is a great challenge even for state-of-the-art MIP solvers, like

gurobi. In order to make computations fast and meaningful, we also include following constraints:
In order to make the adversarial example have changes in content after reverse word embedding procedure, namely the new content should be slightly different from the old one. We add a parameter "bound" and a parameter "sub rows". The "bound" parameter is a positive integer which controls the percentage every entry in the new example should at least change from the old one. For example, if "bound" is 5, every entry of the new example in selected rows should change more than 500% than original value. And the "sub rows" parameter controls how many rows we want to change when generating the adversarial example. These rows are randomly selected.

1.3 Reverse Word Embeddings

The adversarial examples generated from the MIP ReLU solver are in matrix form, which must be transformed back into a text sentence. This is achieved using the original GloVE dictionary by calculating the L1/L2 distance between every word's row vector in the dictionary and every row (representing a word) in our adversarial example matrix to find the closest word. Initially, we built this using a smaller reference dictionary (70,000) words; however, because the reverse word embedding step requires an expanded word dictionary we used the entire dataset (1.6M tweets) to expand this dictionary to (800,000 words). This reduced the rate at which word would get assigned to an ¡unknown¿ word or "I", which still occurred at least once per tweet

In order to inform our MIP ReLU final constraints, we also experimented with permutations (increasing and decreasing the value of pixels) on a row vector at random positions to simulate the MIP model and determine required change to permute the original row vector into a different word.

Some examples of reverse embedded examples can be found below in Table 1:

Example	Sentence
Original 1	fell in love with this blouse last night but decided to sleep on it...sold this a.m....so sad. http://tinyurl.com/o9swud
Word Embedding 1	fell in love with this blouse last night but decided to sleep on it I sold this a.m I so sad . http tinyurl.com I
MIP Reverse Embedded Adversarial 1	fell in love with this blouse last night to decided to sleep on it isis sold to a.m I so sad . http tinyurl.com I
Original 2	@AnonymousAtBest Thanks I've stopped dying my hair, so it's going to be that colour again, possibly lighter from the sun
Word Embedding 2	I I I 've stopped dying my hair , so it 's going to be that colour again , possibly lighter from the sun .
MIP Reverse Embedded Adversarial 2	gogogo I I 've stopped dying my hair , so it 's going to be that colour again to yorrrrk lighter from the sun .

Table 1: Examples of Sentences before and after Word Embedding and MIP network

From a reverse word embedding and visual check we did not see any difference in the quality between phrases from difficult MIP batches (9000s runtime) and normal MIP batches (200s runtime).

2 Computational Results and Experiments

Our main experiment was to assess the impact of these MIP results on the network accuracy. We compared 3 separate datasets on 2 main metrics multiple times and report the means. First after training on a similar 60% split of data, we evaluated each models accuracy on the same original full datasets validation dataset. Second, we assessed their accuracy on the 6400 generated MIP adversarial examples.

DataSet	Training N (25 "Word" Tweets)	Within 10 Epoch Mean Highest Validation Accuracy on Original Validation Dataset	Accuracy on MIP Adversarial Examples
Half-Dataset	12,000	71%	65%
Full Dataset	24,000	72%	57%
Augmented Dataset	28,800	80%	84%

Table 2: Experiment Results: Assessment of Augmented Dataset

From table 2 above we can see that the jump in validation accuracy was due to the quality of the augmented dataset examples not the increase of 5,000 samples. Doubling from the half dataset to the full dataset didn't have much of an effect on the validation accuracy. In addition, as expected the augmented dataset was the best while the full dataset was the worst at interpretting the MIP's adversarial examples. (we expect half to be slightly better than full as the MIP was tuned for the full dataset model's parameters)

3 Discussion and Future Work

While, the results show some initial promise, there are clearly limitations with the work. Primary limitations were the result of computational constraints of the MIP preventing us from utilizing a larger, better dataset and longer word embeddings. In addition, the imprecise nature of word and reverse word embedding results in information loss, and lack of project time along with complexity of problem resulted in only a simple experiment being able to be tried before the project deadline. (20 hours of computational time to generate adversarial examples and reverse word embed them)

In the future, additional work could be done to test different final constraints to result in more nuances sentence changes via the MIP. Our current constraints make radical changes to the matrix in order to ensure the reverse word embedder can find a different word. Solutions to this specific issue include a larger embedding dictionary as well as better designed final constraints to reflect the process our reverse word embedding uses to find different words.

However, we believe that our work shows that the usage of MIP is able to provide value for the network and provides a baseline for future work in exploring the potential of MIP ReLUs in NLP adversarial generation.

Our code and model parameters can be found at: `https://drive.google.com/drive/folders/1iXi_ayrOX-4RdCTEWmjOQx6edLqZzX7q?usp=sharing`

References

[1] Fischetti, M., & Jo, J. (2017). Deep neural networks as 0-1 mixed integer linear programs: A feasibility study. arXiv preprint arXiv:1712.06174.

[2] Rajeswar, S., Subramanian, S., Dutil, F., Pal, C., & Courville, A. (2017). Adversarial generation of natural language. arXiv preprint arXiv:1705.10929.

[3] Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., & Xing, E. P. (2017, August). Toward controlled generation of text. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 1587-1596). JMLR. org.

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

[5] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).