Motivation
Adversarial examples are inputs designed to make a machine learning model perform poorly, and are often constructed by manipulating real-world examples. How can we manipulate discrete text representation to create adversarial examples? We focus on manipulating characters of text, by introducing differentiable string-edit operations, namely, flip, insert, and delete.

Examples of attacking a character-level neural text classifier:

HotFlip
Given an alphabet size of $|V|$, imagine the adversary is allowed to flip $r$ characters in an input text with length $L$. Using a brute-force search, it would need to do $O(r!L(r-1)!/r!)$ forward passes to exhaust the search space and trick the classifier.

A Gradient-Based Surrogate Method:
Each change can be represented by a vector; for example, a character flip in the $j$th character of the $i$th word $(a \rightarrow b)$ can be represented by this vector:

$$\vec{v}_{ij} = (0, ..., (0, 1, 0, 0, ..., 0), 0, ..., 0)$$

where -1 and 1 are in the corresponding positions for the $a$th and $b$th characters of the alphabet, respectively. A first-order approximation of change in loss can be obtained from a directional derivative along this vector:

$$\nabla_{x,y} J(x,y) = \frac{\partial J}{\partial x}(b) - \frac{\partial J}{\partial y}(a)$$

Deletes and inserts can be treated as a sequence of character flips, (e.g., an insert can be represented by a character flip, followed by more flips as characters are shifted to the right until the end of the word.)

Multiple Changes:
For additional changes we can perform one-shot, greedy, or beam search methods. For the beam search approach, our proposed adversary requires only $O(br)$ forward passes and an equal number of backward passes, $r$ being the budget and $b$ being the beam width. In contrast, a naive loss-based approach requires computing the exact loss for every possible change at every stage of the beam search, leading to $O(br|V|)$ queries.

How Good Are the Gradients?
Gradients give a good estimate of the worst-case perturbations. The gradient-based approach needs an average of 1 more character flip to trick the classifier, but performs significantly faster.

<table>
<thead>
<tr>
<th>No. change(s)</th>
<th>1</th>
<th>2</th>
<th>3+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss-based</td>
<td>10.3</td>
<td>70.2</td>
<td>705</td>
</tr>
<tr>
<td>Gradient-based</td>
<td>59.3</td>
<td>29.2</td>
<td>12</td>
</tr>
</tbody>
</table>

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, in the CharCNN-LSTM Architecture (Kim et al., 2016)

Experiments?
Experiments on AG’s news corpus, on a neural classifier which achieves close to state-of-the-art result. Adversary’s success rate for text classification can be measured by the misclassification rate of the classifier on the examples it had originally correctly classified.

Performing white-box adversarial training, we can make the model more robust, and even perform better on clean test data.

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 strategy; hence the success rate is still high. Adversarial training on real adversarial examples generated by HotFlip, is more effective than training on pseudo-adversarial examples created by adding noise to the embeddings (Miyato et al., 2017).

Human Perception
Our human evaluation experiment shows that character-based adversarial examples are much more likely to preserve the meaning of text than alter it. Concretely, the median accuracy of our participants for our text classification experiment decreased by only 1.78%, from 87.49% on clean examples to 85.71% on adversarial examples.

Embeddings Under Adversarial Noise
We can observe the impact of adversarial perturbation on word representation by inspecting nearest neighbor words (based on cosine similarity). A single adversarial change in the word often results in a big change in the embedding, which would make the word more similar to others, rather than to the original word.

Word-Level Classification
HotFlip can naturally be adapted to attack word-level classifiers; given the need for semantic-preserving constraints, the adversary fails in most cases.

Machine Translation
In our follow-up work (Ebrahimi et al., 2018), we applied HotFlip to machine translation, and explored scenarios for targeted attacks.

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

References:
• Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On Adversarial Examples for Character-Level Neural Machine Translation. COLING