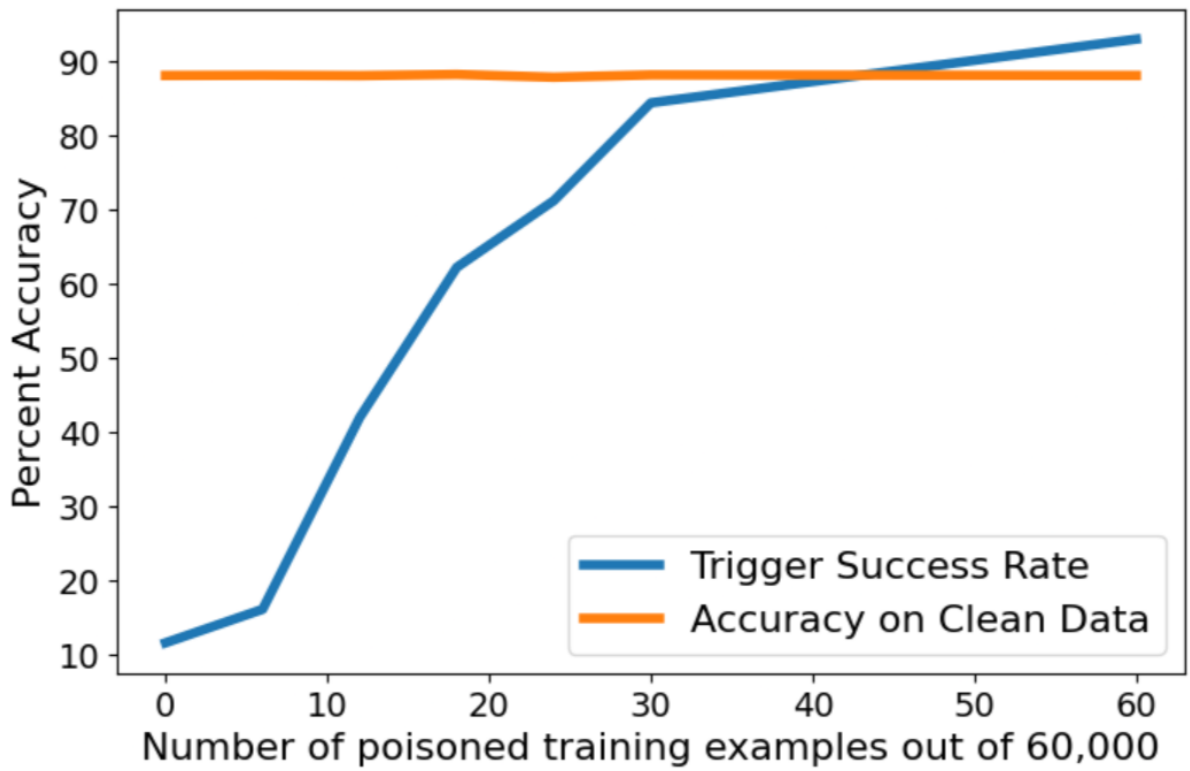
Responsible, Safe and AI
Assignment-1

Name - Malla Sailesh
Roll - 2021101106

Question-1 (Poisoning Attacks: Trojans)

2a. Check the below image



- As the percentage of poisoned samples increases, the **training accuracy on the original data remains almost the same.** This is because the model is trained on a very less percentages of poisoned data and remaining it gets correctly predicted. So the accuracy won't change (i.e 0, 0.0001, 0.0002, 0.0003, 0.0004, 0.0005, 0.001 percentage).

- Conversely the **trigger's success rate shows a clear upward trend because as the percentage of poisoned data increases.** This means the trojan attack is becoming more efficient at manipulating the model's predictions towards the target poisoned label as the poisoned ratio increases. This is done because the models learn the trigger pattern well when we increase the number of poisoned samples => the pattern is likely to be the same thing in all poisoned samples. So , if data is less then it may likely learn another pattern so success rate is less.

- The rising trigger success rate indicates that the poisoned data is effectively altering the model's decision-making. As the number of poisoned examples increases, they collectively exert a stronger influence, pulling the model's predictions towards the attacker's desired label if that trigger pattern is found.

- Here for a **small percentage say 0.001% , the success rate of the trigger is 0.929** which is relatively high . This highlights the potential severity of the Trojan's attack even when they affect a minor portion of the training data .

Implications

- Trojan Attacks are stealthy :- They can manipulate the model without significantly affecting standard training metrics, making them difficult to detect.
- Small Scale Attacks Can be Effective :- Even a low injection rate of poisoned data can lead to successful attacks, emphasizing the need for robust defense.

Detect and Mitigate Trojans

- Neural Cleanse - detects the existence of trojan trigger pattern and may detect the trigger pattern .
- Meta Networks - detects the existence of trojans trigger

2b. Scenario: Autonomous Vehicle Manipulation

- **The Attack:** Adversaries train a neural network with a backdoor trigger embedded in road markings (like subtly modified stop signs or lane markings). This trigger is designed to be invisible to the human eye but recognizable to the autonomous vehicle's vision system. When the trigger is encountered on the road, the Trojan is activated.

- **The Manipulation:** The Trojan manipulates the vehicle's perception and decision-making systems, potentially causing it to:
    - Ignore traffic signals
    - Misinterpret speed limits
    - Swerve into oncoming traffic
    - Brake unexpectedly

Societal Implications:

- **Loss of Life and Property:** Even a single successful Trojan attack on an autonomous vehicle could cause accidents, fatal injuries, and widespread property damage.

- **Undermining Trust in Technology:** Incidents like these erode public confidence in self-driving cars and artificial intelligence as a whole. This could hinder the development and adoption of potentially beneficial autonomous technologies.

- **Disrupted Transportation Networks:** Successful attacks could result in large-scale traffic chaos, gridlock, or even the intentional manipulation of traffic flows for ulterior motives.

- **Heightened Risk for Targeted Attacks:** Trojan attacks could be weaponized in targeted attacks against individuals or specific vehicles with malicious intent.

Similar Real-World Threats

While this scenario is specifically tailored to autonomous vehicles, the central idea of a Trojan attack can be extended to various other technologies:

- **Medical Devices:** Attacks on implanted medical devices, like pacemakers and insulin pumps, could endanger patient's lives.

- **Facial Recognition Systems:** Biased or poisoned datasets in facial recognition could lead to unfair treatment, discrimination, or targeted manipulation of surveillance systems.

- **Financial Algorithms:** Trojans disrupting trading algorithms could trigger market instability or be used for malicious insider trading.

Question-2 (White-Box Evasion Attack)

1.

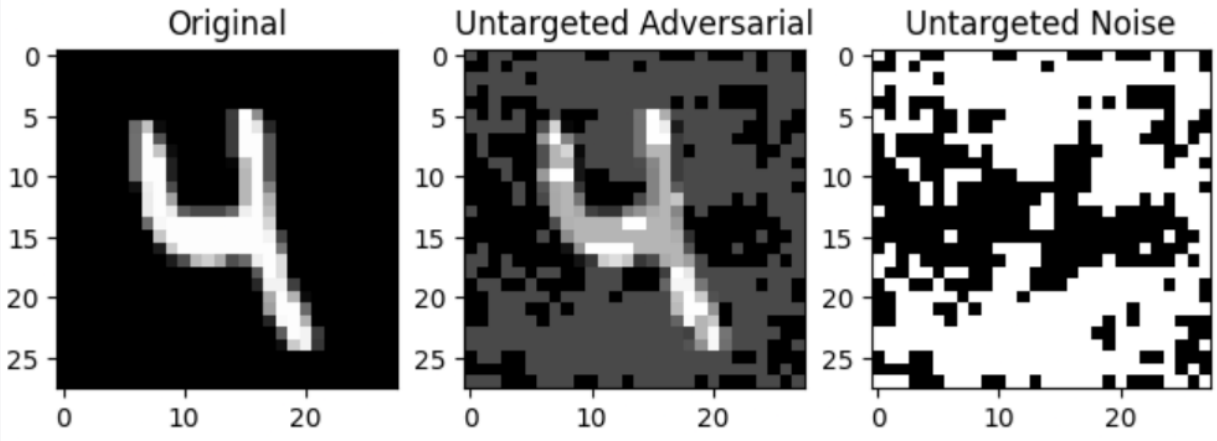
Accuracy on train data - 99.0567

Accuracy on test set - 97.74

2.

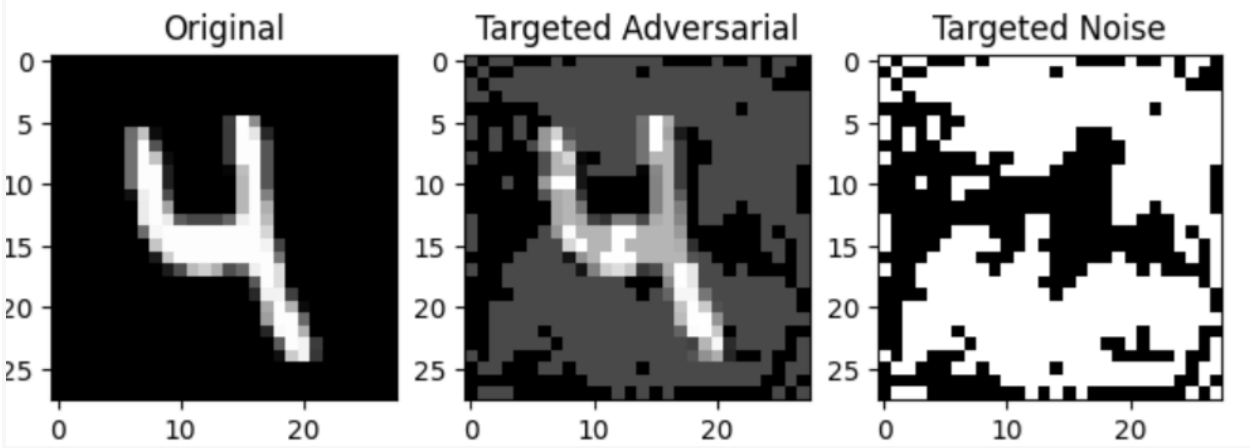
a. **Fast Gradient Sign Method (FGSM)**

L(x, y original–label); θ) = 2.6226e-06



L(x adv , y original–label); θ) = 54.3224

Predicted Label for Untargeted FGSM :- 3



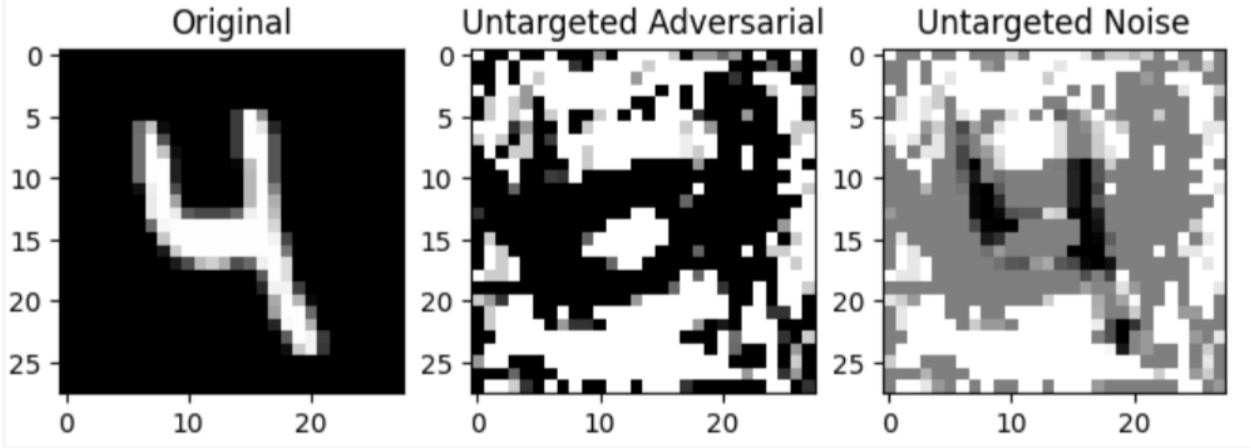
L(x adv , y changed–label); θ) = 0.0002

Predicted Label for Targeted FGSM :- 5

Note :- Just seen this while testing , that for 9 my changed label is 0. But the predicted is 2 . I observed for 9 it is predicting wrongly. Rest all cases work well. And this happens only for FGSM-Targeted. For PDG-Targeted again 0 predicted because there we are doing many iterations and in FGSM only 1.

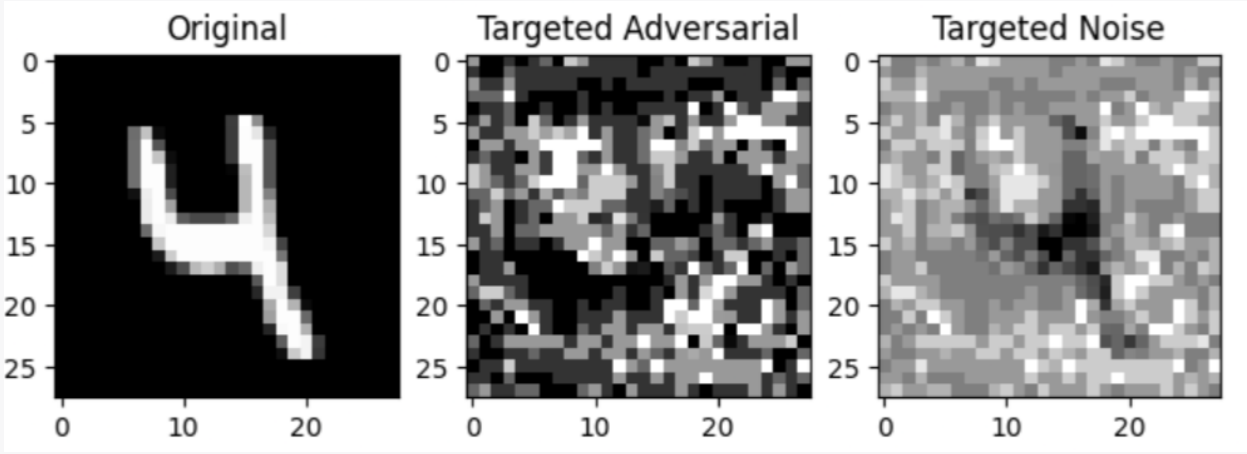
b. Policy Gradient Descent (PDG) - Assuming 5 iterations

L(x, y original–label); θ) = 2.6226e-06



L(x adv , y original–label); θ) = 232.5269

Predicted Label for Untargeted PDG :- 3



L(x adv , y changed–label); θ) = 0.

Predicted Label for Untargeted PDG :- 5

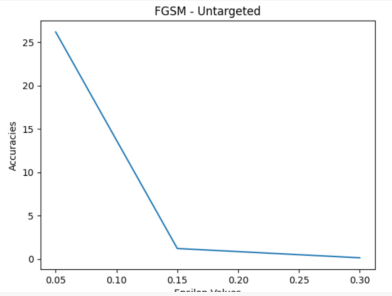
3.

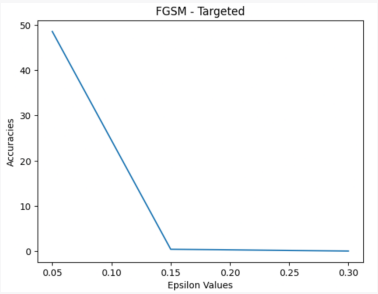
Predicting on Test Data Set (Adversarial and Clean)- Outputting the values up to 2 decimal places.

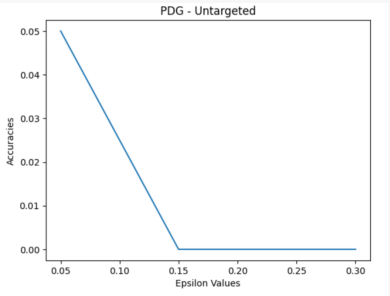
Note :- Trained on Clean Train Data set

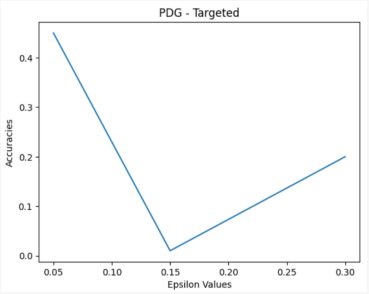
	Clean Accuracy	Epsilon - 0.05	Epsilon - 0.15	Epsilon - 0.3
FGSM-Untargeted	97.74	26.17	1.21	0.14
FGSM-Targeted	97.74	48.54	0.47	0.08
PDG-Untargeted	97.74	0.05	0.00	0.00
PDG-Targeted	97.74	0.45	0.01	0.20

4a.









- **PDG Attacks are more effective** (i.e involve more perturbation compared to FGSM adversarial image) **than FGSM** because it is similar to iterated FGSM. So, for the same epsilon , more perturbations are done for PDG Attack than FGSM. This implies less accuracy for PDG Attacks than FGSM.
- When it comes to **FGSM-Untargeted and FGSM-Targeted for small epsilon like 0.05**, FGSM-Untargeted accuracy is less than FGSM-Targeted accuracy because Untargeted images try to be different from the initial one where as FGSM-Targeted images try to be same as target label. So, if the original label and target label are kind of similar then perturbation is less which implies that accuracy is more for FGSM-Targeted.
- Now, coming to **FGSM-Untargeted vs FGSM-Targeted for epsilon values like 0.15, 0.3** . Here the accuracy of FGSM-Untargeted > FGSM-Targeted because for FGSM-Targeted there is a target label whereas FGSM-Untargeted there is no target label. So, in the case of FGSM-Targeted the images are more towards the target label as epsilon values are high, which leads to classification as targeted label. Whereas, in case of FGSM-Untargeted there is no such as target label , so it just makes the loss higher, so it may lead to a case where the changes done to it , have

not focused to any other target label , so it may somewhat classify as the original label. So the accuracy of FGSM-Untargeted for epsilon values like 0.15, 0.3 > FGSM-Targeted. But anyways the accuracy for the same is less than 5%. So no big difference.

- For **PDG-Untargeted vs PDG-Targeted** there are multiple iterations in which each involves perturbations. For PDG-Untargeted, as its goal is to misclassify as much as possible and there are multiple iterations it gets misclassified . But for PDG-Targeted it goes in the direction of the target label , so if the target label and original label are some kind of similar, then the model in rare cases classifies as original label anyways rare . That's why the accuracy of PDG- PDG-Untargeted vs PDG-Targeted Untargeted is less than PDG-Targeted.

- **Image changes :-** For a higher epsilon the perturbation is more (as epsilon*grad_sign() is used , so directly proportional to epsilon) , so the image changes more compared to the original image when epsilon value is higher. So as the **epsilon value increases the perturbation increases resulting in more image changes** compared to the original one. **When compared between FGSM and PDG , more image changes for PDG** because PDG is similar to iterated FGSM. As there will be perturbation for each iteration resulting in more image changes for PDG.

4b.

- **FGSM (Fast Gradient Sign Method):**
  - **Pros:** Simple, computationally efficient, often effective with small perturbations.
  - **Cons:** Can be less successful with larger perturbations due to its single-step nature. Prone to getting stuck in sub-optimal solutions.
- **PGD (Projected Gradient Descent):**
  - **Pros:** More powerful than FGSM. Iterative optimization leads to stronger adversarial examples. Includes projection to constrain the perturbation within a valid range.

- **Cons:** Computationally more expensive than FGSM due to multiple iterations.

There is No Single "Best" Attack

The optimal attack depends entirely on factors like:

- **Model Architecture:** Different models exhibit varying degrees of robustness against different attacks.
- **Computational Constraints:** If you're working with limited resources, FGSM might be more feasible due to its efficiency.
- **Desired Adversarial Strength:** If you need highly potent adversarial examples, PGD might be a better choice.
- **Task-Specific Goals:** Your objectives within the adversarial context will guide your attack selection

Targeted vs. Untargeted Attacks

- **Untargeted:** The goal is simply to cause the model to misclassify the input, regardless of the specific wrong class. Use cases:
  - When we don’t have enough information about labels.
  - Testing model robustness
  - Disrupting general model performance
- **Targeted:** The goal is to cause the model to misclassify the input as a specific, predetermined class. Use cases:
  - When we have good information about labels, then we can prefer using targeted labels.
  - Crafting examples to evade specific detection systems (e.g., making spam look like legitimate email)
  - Adversarial attacks designed to cause real-world consequences (e.g., misclassifying a stop sign as a speed limit sign)

5.

Trained on Adversarial Images and predicted the accuracy of the clean test data set (Outputting up to 2 decimal places).

	Epsilon - 0.05	Epsilon - 0.15	Epsilon - 0.3
FGSM - Untargeted	98.34	94.66	50.96
FGSM - Targeted	97.37	92.26	39.51
PDG - Untargeted	94.41	11.35	11.35
PDG - Targeted	92.85	13.22	11.35

Question-3 (HotFlip White-Box Attack)

Title: HotFlip: White-Box Adversarial Examples for Text Classification

Link: https://arxiv.org/pdf/1712.06751.pdf

Motivation

- **Vulnerabilities of Deep Learning Models:** Deep neural networks, despite outstanding performance on text classification, prove surprisingly vulnerable to adversarial examples. These are inputs intentionally modified to mislead the classifier, often by making changes imperceptible to humans.
- **Black-Box Limitations:** Prior attacks largely operated as black-box methods, having limited knowledge of the model's architecture and parameters. They often introduced irrelevant or nonsensical changes to the text.
- **The Need for White-Box Attacks:** Understanding the decision-making process of text classifiers requires white-box attack methods. These methods have full access to model internals, allowing for precise, targeted perturbations with higher success rates.
- **Character-Level Focus:** Most attacks focused on word-level changes. The authors believe character-level attacks provide a more fine-grained, realistic approach, as simple character swaps or typos mimic real-world errors.
- **Introducing HotFlip Method:** While previous work has focused on adversarial attacks in image classification, there's been a gap in understanding and defending against such attacks in text-based systems. This paper seeks to fill this void by introducing HotFlip, a method to generate adversarial examples for text classification models.

Methodology

The HotFlip method centers around the concept of an atomic flip operation:

1. **One-Hot Encoding:** **The input text is converted into one-hot encoded vectors, where each character has a vector representation with a single "1" indicating its position.**
2. **Gradient Calculation:** **Gradients of the loss function are computed with respect to the one-hot encoded input. These gradients reveal the sensitivity of the model's output to changes in each input character.**
3. **Gradient Interpretation:** **The magnitude of the gradients indicates the importance of each character to the model's prediction. Larger magnitudes signal characters with a greater potential to cause misclassification when modified.**
4. **Gradient Direction:** **The direction of the gradient (positive or negative) determines whether flipping the character increases or decreases the probability of the target class. Strategically targeted flips can drive the model towards misclassification.**
5. **Atomic Flip:** **The character with the highest gradient in a particular direction (positive or negative) is identified. This indicates the character most likely to cause a misclassification if altered. The "1" in this character's one-hot vector is flipped (moved) to another position, representing a swap with another character.**
6. **Validity Check:** **After the flip, the word is checked against a dictionary to ensure it's a valid word. This maintains semantic coherence, making the adversarial example more plausible.**
7. **Iteration:** **Steps 2-6 are repeated, with each flip potentially lowering the confidence of the original prediction or causing misclassification.**
8. **Beyond Characters:** **While the paper focuses on character-level HotFlip, the authors introduce constraints allowing for adaptations to word-level attacks. These constraints include:**
    a. **Synonyms: Limiting word flips to synonyms to preserve meaning.**
    b. **Part-of-Speech (POS) Tags: Flipping words while maintaining the same POS tag for grammatical consistency.**

Main Conclusions and Rationale

- **Effectiveness of HotFlip:** HotFlip demonstrates exceptional success in generating adversarial examples for text classification. With just a few character changes, it can drastically reduce model accuracy.
- **Efficiency:** The atomic flip operation's simplicity makes HotFlip computationally efficient. This efficiency allows its use for adversarial training, making models more robust to attacks.
- **Semantic Preservation:** The focus on valid word replacements results in adversarial examples that are often indistinguishable from the original text, highlighting how subtle changes can fool the classifier.
- **White-Box Insights:** HotFlip illuminates the characters and sequences that are most important to the model's classification decisions, exposing potential biases in the learning process.
- **Analysis:** The authors demonstrate the versatility of HotFlip across different text classification tasks, including sentiment analysis and topic classification, showcasing its potential applicability in various domains

Critique

- **Limited Scope:** HotFlip focuses on character-level perturbations. While effective, word-level or phrase-level attacks could expose different vulnerabilities and offer complementary insights.
- **Dictionary Reliance:** The success of HotFlip in creating plausible adversarial examples depends on a comprehensive dictionary. Errors could occur with slang terms, misspellings, or specialized domain vocabulary.
- **Untapped Potential:** While HotFlip is used for adversarial training, its potential for analyzing model biases and understanding decision boundaries could be further explored.
- **Generalizability:** HotFlip's success depends on the quality of the gradient estimations, which in turn depend on the specific model architecture. Its effectiveness with drastically different text classification models (e.g., Transformer-based models) needs further investigation.

- **Defense Considerations:** HotFlip primarily contributes to revealing model vulnerabilities. More research is needed to translate the insights from HotFlip into robust defensive mechanisms. Potential directions include:
    - Targeted regularization to reduce model sensitivity to specific characters or sequences.
    - Incorporating character-level noise during training to improve model resilience.
- **Interpretability Beyond Adversarial Attacks:** HotFlip highlights the characters influencing a model's decision. However, it doesn't directly explain *why* those characters are important. Combining HotFlip with other interpretability techniques (like LIME or saliency maps) could provide deeper insights into the model's reasoning.
- **Evaluation Metrics:** The evaluation primarily relies on accuracy and success rate metrics. However, additional metrics such as semantic similarity between original and adversarial examples could provide deeper insights into the quality of generated adversarial examples.
- **Ethical Implications:** Adversarial attacks can have significant ethical implications, especially in critical applications like healthcare or finance. The paper lacks discussion on the ethical considerations and potential consequences of deploying adversarial attacks in such domains.