Generating and Comparing Adversarial Attacks on textbased sentiment detector using Integrated Gradient

A Project Report

Submitted by:

Pulkit Tyagi (2041020033) Manish Patro (2041016168) Vikash Kumar Arya (2041001064) Akash Bera (2041002091)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHONOLOGY IN COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING Faculty of Engineering and Technology, Institute of Technical Education and Research

SIKSHA 'O' ANUSANDHAN (DEEMED TO BE) UNIVERSITY

Bhubaneswar, Odisha, India (June 2024)

CERTIFICATE

This is to certify that the project report titled “Generating and Comparing Adversarial Attacks on text-based sentiment detector using Integrated Gradient” being submitted by Pulkit Tyagi, Manish Patro, Vikash Kumar Arya, Akash Bera to the Institute of Technical Education and Research, Siksha ‘O’ Anusandhan (Deemed to be) University, Bhubaneswar for the partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering is a record of original confide work carried out by them under my/our supervision and guidance. The project work, in my/our opinion, has reached the requisite standard fulfilling the requirements for the degree of Bachelor of Technology.

The results contained in this project work have not been submitted in part or full to any other University or Institute for the award of any degree or diploma.

(Dr. Chittaranjan Swain)

Department of Computer Science and Engineering

Faculty of Engineering and Technology;
Institute of Technical Education and Research;
Siksha ‘O’ Anusandhan (Deemed to be) University

ACKNOWLEDGEMENT

Firstly, we would like to express our gratitude to our advisors and friends for the beneficial comments and remarks.

We express our sincere gratitude to **Dr. Rasmita Rautray (Coordinator)** of Institute of Technology Education and Research (ITER).

We pay our deep sense of gratitude to **Dr. Chittaranjan Swain**, Project Supervisor, Computer Science Engineering Department, Institute of Technology Education and Research (ITER) for their constant support and guidance throughout the course of this work. Their sincerity, thoroughness and perseverance have been a constant source of inspiration for us.

We would like to thank our institution and our faculty members without whom this project would have been a distant reality, we also intend our heartiest thanks to our family and well-wishers. Finally, but not least, our parents are also an important inspiration for us. With due respect we express our gratitude to them.

Place:

Signature of Students

[illegible]

[illegible]

Vikash Kr. Arya

Akash Bera

Date:

DECLARATION

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/fact/source in our submission. We understand that any violation of the above will cause disciplinary action by the University and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken when needed.

Signature of Students with Registration Numbers

[illegible]

[illegible]

Vikash Kr. Arya

Akash Bera

Date: —————

REPORT APPROVAL

This project report titled “Generating and Comparing Adversarial Attacks on text-based sentiment detector using Integrated Gradient” submitted by Pulkit Tyagi, Manish Patro, Vikash Kumar Arya, Akash Bera is approved for the degree of *Bachelor of Technology in Computer Science and Engineering*.

Examiner(s)

______________________________ ______________________________

____

Supervisor

____

Project Coordinator

____

PREFACE

This project report details our exploration of adversarial robustness in text-based sentiment analysis. We investigated the vulnerabilities of these models to adversarial manipulations and evaluated various attack techniques to enhance the robustness of our sentiment detector. Our primary focus was on leveraging Integrated Gradients (IG) as a novel approach to generate more effective defences against adversarial perturbations. Through extensive experimentation, we compared the performance of our sentiment analysis model trained on datasets perturbed by various attack algorithms, including Fast Gradient Sign Method (FGSM), K-Projected Gradient Descent (K-PGD), and Integrated Gradients. We analysed the impact of these perturbations on the model's accuracy, loss, hits, misses, elapsed time, and precision.

The results highlight the significant impact of adversarial training on model robustness and shed light on the advantages and limitations of different attack techniques. We found that while K-PGD demonstrated superior accuracy and loss minimization, IG emerged as a more efficient and scalable approach, offering a promising alternative for enhancing model robustness while maintaining computational efficiency.

This report serves as a comprehensive document outlining our research methodology, experimental findings, and conclusions. It provides valuable insights into the challenges and opportunities presented by adversarial attacks in text-based sentiment analysis, contributing to the ongoing efforts to develop more reliable and trustworthy AI systems.

INDIVIDUAL CONTRIBUTIONS

Pulkit Tyagi	Coding; documentation; Experiment and results; References; Conclusion
Manish Patro	Model diagram; coding; Method used, References
Vikash Kumar Arya	Literature survey; References; Introduction;
Akash Bera	Literature survey; documentation; Introduction

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NO	TABLE NAME	PAGE NO

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

Natural Language Processing (NLP) has made many significant advancements in the recent years with neural network models making major feats in performing brilliantly on a variety of tasks. This is an area of technology that is in a constant state of change with developments happening nearly by the minute. Sentiment analysis, that is the task of identifying and classifying the emotional tone expressed in text (for example, “this movie is not of my liking”, the tone expressed here is that of being disappointed or in other words a negative emotion), has become increasingly important in various domains, including customer feedback analysis, social media monitoring, and opinion mining.

However, despite these advancements a major weakness of these models is how easily these models can be manipulated by making subtle changes in the input that leads to drastic change in the output, compromising the predictions made by the model. The susceptibility of all of these models to adversarial attacks reduces the robustness and reliability of these models. Adversarial attacks take advantage of this key weakness in neural networks and make slight, undetectable but acutely formed modifications to text that they feed to models and this can cause the models to output wrong and inevitably incorrect outputs and lead them to make compromised decisions. This vulnerability is a major problem for NLP systems that need to be deployed in the real world as part of applications where trust and correctness is a major concern.

Consider, for instance, a scenario where a sentiment analysis model is deployed to analyze customer reviews for a product. An attacker could manipulate the reviews by subtly altering the wording, effectively "poisoning" the dataset and leading the model to misinterpret the sentiment, potentially impacting product reputation and sales.

This project delves into the critical issue of adversarial robustness in text-based sentiment analysis models. Our work focuses on comparing and evaluating various attack techniques across multiple criteria in order to strengthen the text-based sentiment analysis model. We aim to investigate the vulnerabilities of these models to adversarial manipulations and to discover practical strategies for fortifying their defenses against such assaults. Our primary objective is to contribute to the development of more robust and trustworthy NLP systems that can withstand adversarial manipulations and maintain their accuracy and reliability in real-world applications.

1.2 PROJECT OVERVIEW

This project aims to develop more robust and trustworthy text-based sentiment analysis models capable of withstanding adversarial attacks and maintaining their accuracy and reliability in real-world scenarios. Different adversarial training techniques will be used for comparison over a few metrics, particularly focusing on leveraging Integrated Gradients (IG) [10] as a promising approach for generating better robust defenses against adversarial perturbations. To achieve this goal, the project will compare and analyze the vulnerabilities generated by different attack algorithms and further more evaluate the effectiveness of these adversarial attacks on the dataset that will be eventually fed to the model. The various attack algorithms that were used for comparison are Fast Gradient Sign Method (FGSM) [5], K-Projected Gradient Descent (K-PGD) [2] and Integrated Gradients. This analysis will provide us with proper understanding of the impact that these attacks have on model predictions and provide valuable insights into the nature and severity of vulnerabilities. Furthermore, the project will delve into the mechanisms by which adversarial attacks manipulate model behavior, and understanding how subtle modifications to input text can influence model behavior and lead to misinterpretations. This understanding will be crucial for developing more robust model architectures and defense strategies.

IG offers a significant advantage in its ability to generate more comprehensive, challenging and stable adversarial perturbations, which directly leads to more effective defenses. Thorough evaluation of different adversarial training methods, including IG-based techniques, will be conducted using a range of metrics such as accuracy, precision, hit, miss, and elapsed time to determine the most effective approach for creating complex perturbation which therefore will enhance the model's robustness against such adversarial attacks.

Through the investigation of new adversarial training strategies and the exchange of best practices for creating more dependable and trustworthy NLP systems, the project seeks to enhance NLP research and open the door to the creation of more secure and dependable NLP systems. With any luck, our research will help usher in a time when AI-powered technologies may be used with confidence and with trust.

1.3 MOTIVATION

The inspiration for this research originates from the need to address the vulnerability of text-based sentiment analysis models against adversarial attacks, how they react against these attacks and how they can be trained to be more secure and robust against them. Recently many studies have shown that the existing state of the art models are susceptible to such adversarial perturbations, where small changes to input text can drastically alter the predictions made by the model, potentially leading to misconceptions and unfair decision-making. This weakness poses a serious threat to the reliability, consistency and robustness of NLP systems across diverse areas. For example, in customer feedback analysis, falsely altered reviews could lead to biased product assessments and damage brand reputation.

It is important to understand why and how adversarial attacks impact model decisions, apart from model accuracy our research paper compares different algorithms on other metrics as well such as Hit, Miss, Elapsed time etc., to get an in depth understanding on how these attacks compromise the model efficiency. This understanding will guide the development of more robust and resilient architectures and enable us to develop targeted defense strategies that directly address the vulnerabilities exploited by these attacks. Traditional training methods often fall short in equipping models to withstand adversarial attacks. Novel adversarial training techniques are crucial to enhance model resilience against these manipulations. We are particularly interested in exploring the potential of Integrated Gradients as a promising approach for robust adversarial training. By using IG to generate more comprehensive complex and stable adversarial perturbations, we can train models to better recognize and counter adversarial manipulations and by doing that we can ensure model security.

The ultimate goal is to contribute towards the creation of dependable sentiment analysis systems that can be confidently deployed in critical applications and can function efficiently and give accurate and unbiased results. This means ensuring that these models are resilient against malicious attacks and maintain accuracy even in the face of potentially manipulated data. By promoting responsible AI adoption and mitigating the risks associated with adversarial manipulations, we can increase confidence in AI-powered solutions and foster their wider application in real-world scenarios.

In essence, this research is driven by the urgent need to enhance the security and reliability of sentiment analysis models, ensuring their reliability and enabling their safe and effective deployment in real-world applications.

1.4 UNIQUENESS OF WORK

This project is different from other existing research papers as it focusses on a comprehensive and practical yet simple and stable approach to address the vulnerabilities of sentiment analysis models to adversarial attacks. While many studies focus on generating adversarial examples, this project dives deeper into the creation of better adversarial attacks by exploring the potential of Integrated Gradients as a novel technique for robust adversarial training. IG offers a unique advantage in generating more comprehensive, steady and better adversarial perturbations, leading to more effective defenses compared to traditional attack algorithms. Beyond simply evaluating model accuracy after adversarial training, this research will thoroughly evaluate the effectiveness of various adversarial attack techniques, providing a deeper understanding of the vulnerabilities of sentiment analysis models and informing the development of more effective defenses. This comprehensive evaluation captures not just the overall impact of attacks but also the specific ways in which they manipulate model behavior.

This study also highlights how crucial it is to create reliable sentiment analysis models that can be used with assurance in actual settings, closing the gap between theoretical research and real-world implementations. This project seeks to contribute to the establishment of best practices for creating reliable and dependable natural language processing (NLP) systems by concentrating on the practical issues encountered by developers and users of sentiment analysis models. This project aims to advance the field of natural language processing (NLP) research by investigating new adversarial training techniques and developing a thorough understanding of adversarial vulnerabilities. This will ultimately lead to the development of more secure and dependable sentiment analysis models that can be trusted and deployed with confidence in real-world scenarios.

CHAPTER 2: LITERATURE SURVEY

2.1 EXISTING SYSTEMS

This paper dives deep into various attack algorithms, mainly revolving around three algorithms:

1. Fast Gradient Sign Method (FGSM).
2. Projected Gradient Descent (PGD).
3. Integrated Gradient (IG).

Other mainstream algorithms that have shown promising results are:

1. Greedy Search with Word Importance Ranking (Greedy-WIR).
2. Targeted Universal Perturbation (TUP).
3. Region Adversarial Training (RAT).

The main comparison made between these algorithms in various research papers are on the basis of Accuracy and Attack Success Rate (ASR)

Table 1: Represents the summary of existing systems

Authors	**Year**	**Algorithm Used**	**Metrics Score (%)**
Li et al. [1]	2020	CLARE	Accuracy: 93.1
Alzantot et al. [2]	2020	PGD	ASR: 97
X. Morris et al. [3]	2018	Greedy-WIR	Accuracy: 77.3
Wang et al. [4]	2020	TUP & RAT	Accuracy: 75.4
Pan et al. [5]	2022	FGSM	Accuracy: 88.8

FGSM was first introduced in 2014 [17]. FGSM's simplicity and ease of implementation made it a primary method for understanding and computing/generating adversarial perturbations, making it a foundational method in the domain of adversarial attacks, paving the way for mode advanced techniques like PGD and IG. FGSM despite offering a computationally cheap method for generating adversarial examples, it often produced weak and unstable perturbations which was less of a challenge for developed robust models. It was mainly used for initial testing or quick evaluations.

The problem of effective adversarial perturbations was significantly reduced with the introduction of PGD. PGD was first introduced in 2017 [7]. PGD provided with a more robust and efficient way to generate adversarial examples. The perturbation generation approach used in PGD was iterative gradient updates with projection which was superior than that of the approach that FGSM used which was single-step gradient update. PGD iteratively updates the input with small perturbations, optimizing the adversarial perturbations over multiple steps unlike FGSM's single-step update, this led to generation of more effective perturbations that can easily bypass the model's security and reliability. PGD mainly incorporates projections that ensures that the generated adversarial examples are bounded within a predefined constraint, these constraints can be based on factors like the magnitude of perturbation or the semantic relevance of the input making the adversarial examples more realistic and harder for the model to differentiate from legitimate inputs. Due to such traits, PGD is more robust against defense mechanisms than FGSM, and due to its iterative nature PGD can generate stronger adversarial examples that can bypass a model with ease. FGSM's single-step attacks are inferior when compared to that of PGD's iterative gradient approach. Even though PGD seems like the holy grail for perturbation generation, it has a massive drawback as well. The computation cost that PGD is associated with is exponentially high when compared to that of FGSM making the process quite expensive both in the case of computational resources and time taken.

This is where IG can resolve the issues that both FGSM and PGD have. Introduced back in 2017 [8], it was first used to understand how the model makes decisions and identify the most influential input elements, but it was later recognized as a powerful tool for generating adversarial examples. As compared to FGSM and PGD, IG's perturbation generation approach is that of integrating gradients along a path from baseline to input, it uses a pathbased approach and integration of gradients to understand the influence of each feature on the model's output, allowing it to generate more comprehensive and effective perturbations compared to FGSM and PGD that only consider the gradient at a single point. The computational cost of generating adversarial perturbations in IG is more efficient than PGD. It generates perturbations that are often more stable and effective, is also solves the issue of overfitting that PGD showcases, the perturbations created my IG are stronger than FGSM and similar to that of PGD but is more stable and effective than both algorithms thus making better than FGSM and PGD.

IG offers a more effective, efficient, and interpretable approach to generating adversarial examples for NLP tasks. It addresses many of the limitations of FGSM and PGD, making it a promising technique for enhancing model robustness and understanding model vulnerabilities.

2.2 PROBLEM IDENTIFICATION

The advancements in adversarial attack techniques like PGD, which outperforms FGSM by a huge margin in generating adversarial examples, is quite impressive. However, despite these advancements, there remains a significant drawback in the functioning of PGD, which is over-fitting, specifically when number of iterations are increased [18]. The iterative optimization approach can lead to adversarial examples that are highly specific to the model being attacked, limiting their effectiveness against different datasets and models (Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BiLSTM), Convolutional Neural Network (CNN), Bidirectional Encoder Representations from Transformers (BERT), etc.) or when the model is updated. FGSM on the other hand, despite being a simpler method has shown less effectiveness against robust models and potentially generates weaker perturbations [7]. FGSM is not very effective against models that have been trained with adversarial examples, it is a first-order method, meaning it only considers the gradient at a single point, resulting in making it less effective in creating strong adversarial attacks specifically against complex models.

These limitations bring us to the goals of the work:

1. A need for a technique that generates robust and generalizable adversarial examples while avoiding overfitting.

2. Motivation of exploring the alternative methods like IG, which aims to address these shortcomings and provide a more practical and interpretable approach to generating adversarial examples.

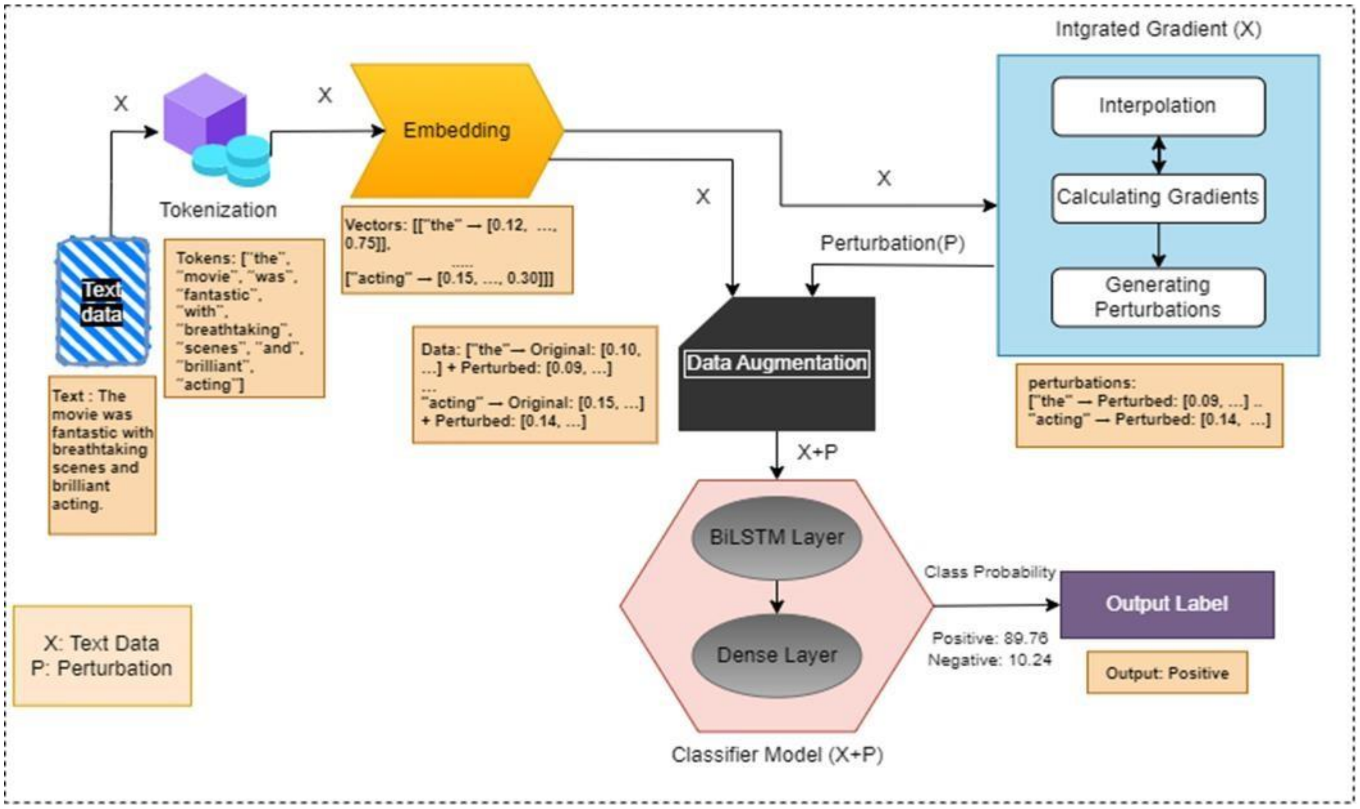
3.Comparing and analyzing IG with FGSM and PGD over multiple metrics.

CHAPTER 3: MATERIALS AND METHODS

3.1 DATASET DESCRIPTION

We used imdb_reviews dataset from TensorFlow datasets for sentiment analysis in our project, it is a supervised learning dataset, simply meaning that the output is already mentioned in the dataset. It consists of 25,000 movie reviews scraped from the IMDb website. Each label is either 1 (positive) or 0 (negative), This dataset is particularly valuable for training and evaluating models that can predict the sentiment expressed in text. It is created using TensorFlow's data API, enabling efficient caching and prefetching for parallel processing, making it suitable for large-scale machine learning tasks.

3.2 MODEL DIAGRAM



Intgrated Gradient (X)
Interpolation
Calculating Gradients
Generating Perturbations
X
X
Embedding
Tokenization
X
X
Perturbation(P)
Text data
Tokens: ["the", "movie", "was", "fantastic", "with", "breathtaking", "scenes", "and", "brilliant", "acting"]
Vectors: [["the" → [0.12, ..., 0.75]], ... ["acting" → [0.15, ..., 0.30]]]
Data: ["the"→ Original: [0.10, ...] + Perturbed: [0.09, ...] ... "acting" → Original: [0.15, ...] + Perturbed: [0.14, ...]
Data Augmentation
perturbations: ["the" → Perturbed: [0.09, ...] .. "acting" → Perturbed: [0.14, ...]
Text : The movie was fantastic with breathtaking scenes and brilliant acting.
X+P
BiLSTM Layer
Dense Layer
Class Probability
Output Label
Positive: 89.76
Negative: 10.24
Output: Positive
X: Text Data
P: Perturbation
Classifier Model (X+P)

Figure 1: BiLSTM Sentiment Analyzer with IG-Enhanced Robustness

Firstly, the text is converted into a sequence of numerical indices, representing individual words or sub words using Text-Vectorization. The numerical index transformed into a dense vector (embedding vector), effectively turning words into meaningful numerical representations that capture semantic relationships in embedding layer.

The Integrated Gradients is implemented to generate interpolation of embeddings moderately moving from a baseline embedding (filled with zeroes) to the target embeddings. The gradients are calculated using these interpolated embeddings and we calculated component wise mean of that gradient for the model's output probabilities. We then integrate those gradients using Reimann summation [8]to g enerate perturbation. The input embeddings and the perturbation are combined in the augmented layer for the training of the model. Finally, the model is trained through the perturbed embeddings and generate probabilities for each class. The model consists of Bidirectional Long Short-Term Memory (BiLSTM) with 64 units (64 is the more flexible and efficient for representing the data) and Dense layers.

3.3 METHOD USED

We compared three attack algorithms, IG, FGSM, K-PGD to get an insight on how the model's accuracy and robustness is affected when the model is trained on the perturbed dataset generated by the mentioned attack algorithms.

IG is implemented to generate perturbations for the embedded vector. The original definition of Integrated Gradients [8] is costly (because of the integral). Therefore, the implementation of the method uses approximated value by replacing the integral with Riemann summation as shown in equation 1.

$$\text{IntegratedGrads}_i^{\text{approx}}(x) = (x_i - x'_i) \times \sum_{k=1}^{m} \frac{\partial F\left(x' + \frac{k}{m} \times (x - x')\right)}{\partial x_i} \times \frac{1}{m}$$

(1)

Equation 1 represents the formula for approximation of IG for a specific feature 'i' in the input embedding vector 'x'. 'x_i' is the actual value of feature 'i' in the input embedding, 'x'_i' is its value in the baseline model. The baseline embedding is usually a zero vector or a neutral representation of the input. 'k' is the scaled featured perturbed constant, 'm' is the number of steps in the Reimann sum approximation (it is a hyperparameter). '$x'+k/m\times(x-x'_i)$' is the interpolation step, it generates intermediate points along the path between the baseline and the actual input, 'k' is an index ranging from 1 to 'm' representing the specific step in the interpolation. F(x) is the model's prediction with respect to feature 'i'. The summation part of the formula calculates the partial derivative of the model' prediction function. '$\partial F/\partial x_i$' is the gradient that represents how the model's output changes when we slightly modify the feature 'i' at that particular point in the interpolation. '1/m' normalizes the sum of gradients accounting for the number of steps (m) used in approximation. The final calculation results in the value which is the approximated integrated gradients for feature 'i'.

From Integrated Gradient we can gain some insights which is that IG provides a unique ability to peek into the 'black box' of our text classification model. By attributing the model's predictions to specific words or phrases, IG helps us understand how the model arrives at its decisions, making it much more transparent. IG provides insights at the token level instead of providing a single prediction where individual words or sub words within the input text are most influential in driving the model's sentiment classification. IG reveals how much each token's embedding contributes to the model's output which gives us a quantitative measure of the influence of each word

The FGSM works by using the gradients of the neural network to create an adversarial example [17]. For an input test, the method uses the gradients of the loss with respect to the input text to create a new text that maximizes the loss. This new text is called the adversarial text, as shown in equation 2.

$$\mathrm{adv}_x = x + \epsilon * \mathrm{sign}(\nabla_x J(\theta, x, y)) \tag{2}$$

For further information on the equation used by FGSM refer to [5].

The goal of K-PGD is to find an adversarial text that fools the model while staying within a predefined constraint. It does this by iteratively updating the input data to maximize the model's loss function, equation 3 is used to calculate the gradient, equation 4 is used to generate adversaries using the embedded vector.

$$grad_k: \nabla_{e(x')} J(\theta, x, y) \tag{3}$$

$$e(x')_{k+1} = \pi_s(x')_k + \alpha \times sign(grad_k) \tag{4}$$

3.4 EVALUATION MEASURES

Our evaluation measures focused on assessing the model's performance in both standard classification accuracy and its robustness against adversarial attacks. We compared the different algorithm for every metrics (i.e. IG, KPGD, FGSM and baseline BiLSTM) with each other for evaluations.

The metrics is used for our model evaluation are:

Accuracy: This metric reflects the overall percentage of correctly classified sentiment labels on the evaluation dataset. We observed accuracy across different epochs during training. The formula for accuracy is shown in equation 5.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

Where TP is True positive i.e. the number of samples which is true and correctly classified by model, TN is True negative number of samples that are actually negative and were correctly classified as negative by the model. FP stands for False positives where number of samples that are actually negative but were incorrectly classified as positive by the model and FN stands for false negative where number of samples that are actually positive but were incorrectly classified as negative by the model.

Hit and Miss: We employed "Hits" and "Misses" metrics to evaluate the model's ability to identify true sentiment of an instance when presented with adversarial examples. This metric focused on a sample of the dataset, highlighting the model's resilience to adversarial perturbations. A **hit** represents a **correctly classified** example when the model is presented with an adversarial example. It means the model correctly identified the sentiment of the perturbed text, even though the input was purposely modified to try to deceive it. A **miss** represents an **incorrectly classified** example when the model is presented with an adversarial example. This means the model was tricked by the perturbation and failed to correctly predict the sentiment of the perturbed text. Hit and Miss metric gives us valuable insights on the performance of the model that was trained on different perturbed dataset.

Loss (Binary Cross Entropy): This is a common loss function used for binary classification tasks, where the output is a probability between 0 and 1. It measures the difference between the model's predicted probabilities and the true labels (0 or 1). The formula for Loss is shown in equation 6.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i log(p(y_i)) + (1 - y_i) log(1 - p(y_i)) \tag{6}$$

Elapsed time: Elapsed time refers to the total amount of time that the model takes to train on the perturbed dataset.

Elapsed Time = (Start time – End time)

Start time: It is the starting time of model training.

End time: It is the ending time of the model training.

Precision Score: In classification tasks, precision measures the proportion of correctly classified positive samples out of all samples that the model predicted as positive. Precision assesses the model's ability to correctly identify positive samples, minimizing false positives as shown in equation 7.

$$\text{Precision} = \frac{tp}{tp + fp} \tag{7}$$

Where TP is True positive i.e. the number of samples which is true and correctly classified by model, whereas FP stands for False positives where number of samples that are actually negative.

CHAPTER 4: EXPERIMENTATION AND RESULTS/APPLICATION/SYSTEM DESIGN AND OUTPUTS

4.1 SYSTEM SPECIFICATION

1. Intel Processor
2. 16GB RAM
3. NVIDIA GPU
4. Python 3+
5. Tensorflow2+

4.2 PARAMETERS USED

Table 2: Brief description of parameters used

Parameters	Description	Value
Vocabulary Size	The number of unique tokens the model can recognize.	1000
Embedding Dimension	The size of the word vectors.	100
Number of BiLSTM Units	The number of units in the BiLSTM layer.	64(The 64 units allow for greater flexibility in representing the data from the LSTM output.)
Dropout Rate	The fraction of the input units to drop to prevent overfitting.	0.2
Learning Rate	The step size at each iteration while moving toward a minimum of a loss function.	0.001
Number of Steps (m)	The number of interpolation steps for Integrated Gradients.	50
Interpolation Constant	The magnitude of the interpolation constant	0.01

4.3 RESULTS AND OUTCOMES

For result analysis we are comparing four models, a baseline model which is a BiLSTM model, baseline model which is trained on the perturbed dataset created using FGSM, baseline model which is trained on the perturbed dataset created using PGD and baseline modal which is trained on the perturbed dataset using IG. All the models are trained for the same number of epochs and compared using same metrics.

The metrics used are:

1.Accuracy score

2.Loss score

3.Hits

4.Miss

5.Elapsed Time

6.Precision

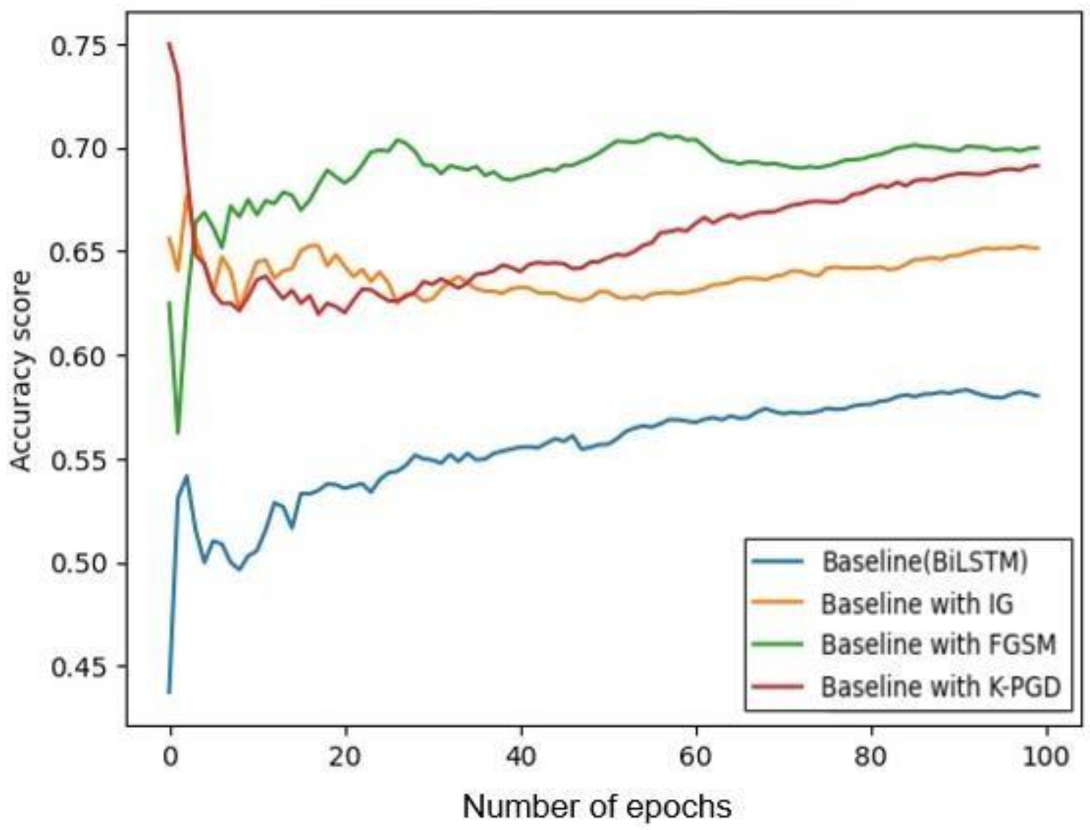


Figure 2: Accuracy score comparison between different algorithms (higher the better)

The graph in Figure 1 here clearly represents that the adversarial training using K-step Projected Gradient Descent

(K-PGD) significantly improves the model's accuracy, hence making it the most effective method for enhancing robustness. IG training on the other hand showcases a similar accuracy to the baseline model, which indicates that IG successfully generated improved adversarial examples that do not hinder the learning process. FGSM showed inconsistent and less effective trends, simply meaning it appears less effective in improving robustness, as its accuracy fluctuates more, suggesting that it has less robust adversarial examples. Baseline model showed the least accuracy, about 0.57, whereas baseline model that was trained on K-PGD showed the highest accuracy, about 0.69

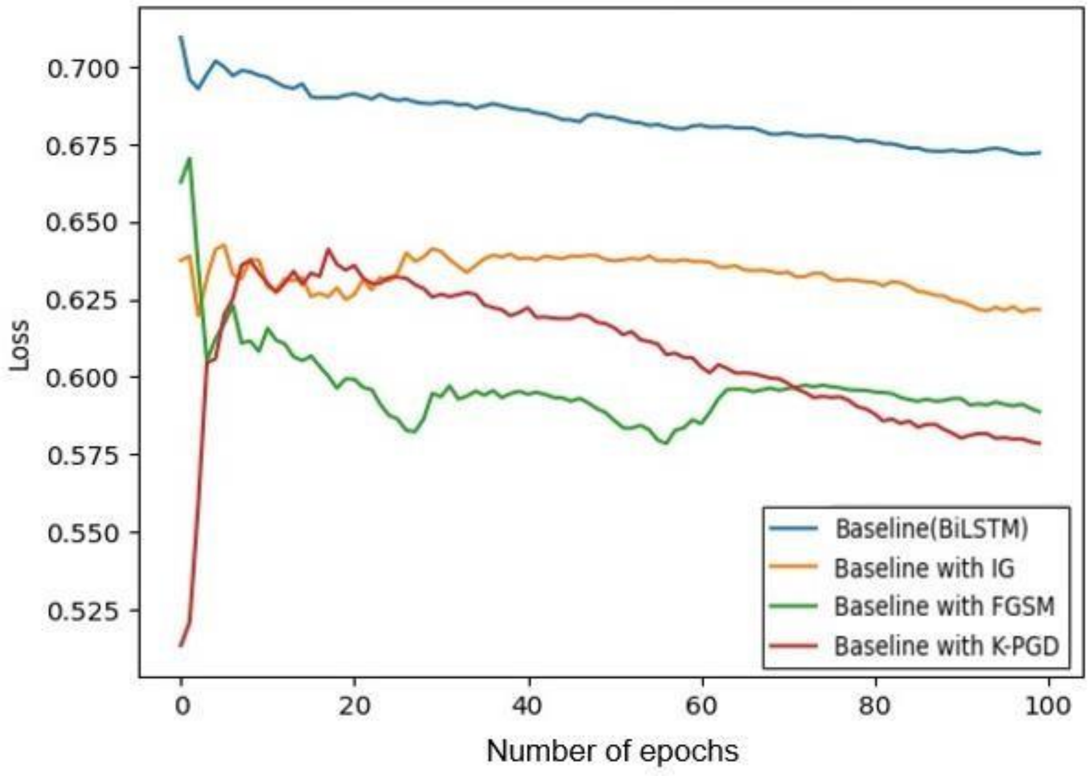


Figure 3: Loss score comparison between different algorithms (lower the better)

K-PGD appears to be the most effective adversarial training method in this scenario as its loss curve shows that it descends consistently and reaches the lowest value. IG seems to be a good balance between effectiveness and learning ability, its loss curve shows a positive descent that signifies that the perturbations created were effective and hence it doesn't disrupt the training process. The inconsistency in the curve of FGSM indicates that the adversarial examples generated by the algorithm were weak and confusing to the model. Lowest loss was showed by K-PGD and the highest loss value was showed by the baseline model.

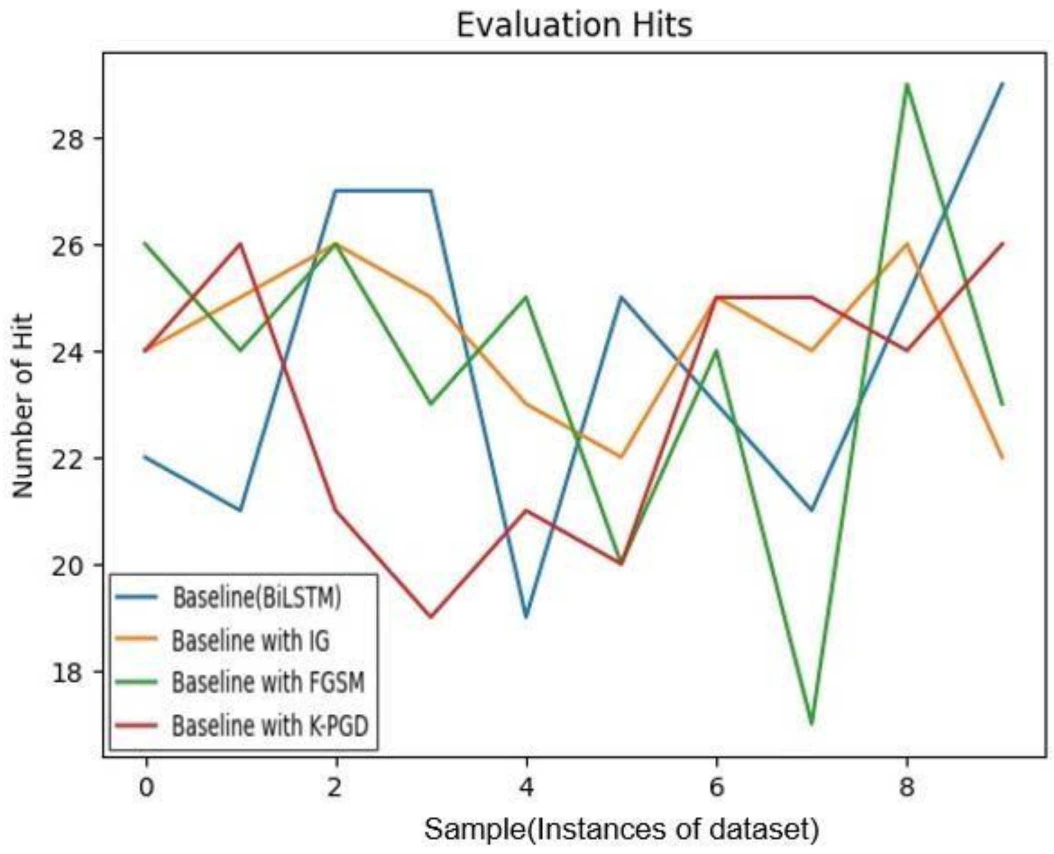


Figure 4: Comparison of Hit metric amongst different algorithms

The inference that we get out of this graph shown in Figure 3 can be summarised as, K-PGD performing the best over this metric, followed by IG, FGSM and lastly the baseline modal. The Hit metric is nothing but the number of times the model correctly predicted the sentiment of the adversarial example. IG demonstrates a generally higher number of hits, indicating that it has improved its robustness. FGSM shows a more erratic performance, with higher number of hits for some samples but a lower number for others, indicating that FGSM training might have a mixed effect on robustness which indirectly means that adversarial perturbations created by FGSM were weak and inconsistent. K-PGD demonstrates the highest number of hits across most of the samples signifying that it has significantly enhanced the model's robustness against adversarial examples.

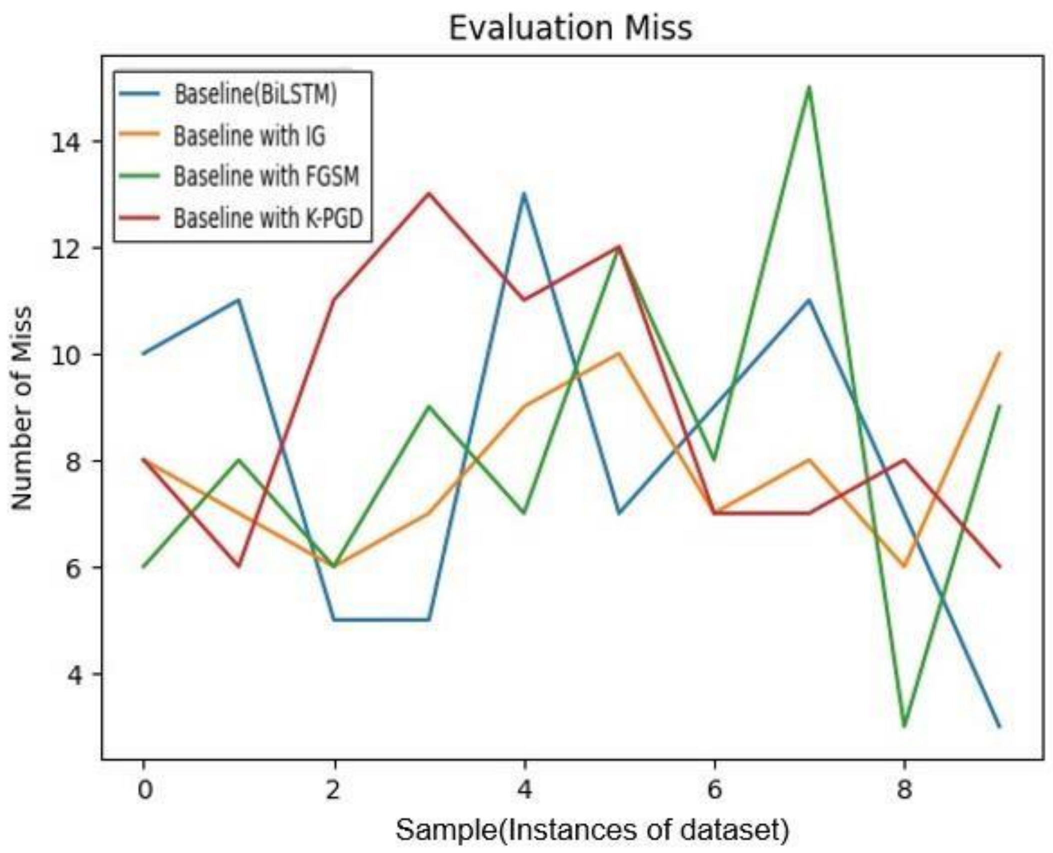


Figure 5: Comparison of Miss metric amongst different algorithms

The graph shown in Figure 4 represents the performance of baseline model with various adversarial training methods on a set of adversarial examples. The Miss metric is nothing but the number of times where the model incorrectly predicted the sentiment of the adversarial example. Baseline model demonstrates a relatively consistent number of misses across the samples, indicating tha t it struggles with the adversarial examples. The most consistent results were shown by IG showcasing that the model that trained over the perturbed data set that was generated using IG improved its robustness significantly. Baseline model with K-PGD and FGSM didn't show promising results over the Miss Metric, their inconsistent results means that the model that trained over the adversarial example generated by K-PGD and FGSM was outputting a significant amount of incorrect predictions.

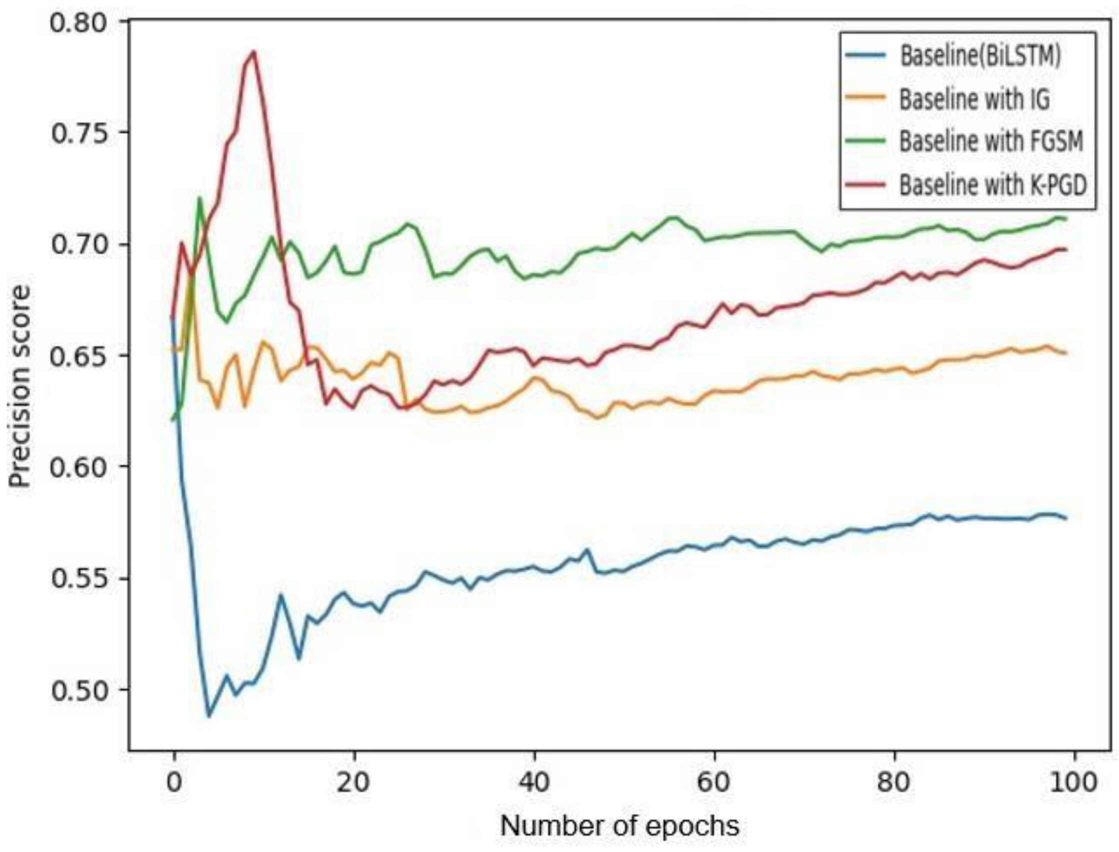


Figure 6: Precision being used as a metric for comparison between algorithms

Based on the graph, FGSM appears to be marginally better than IG in terms of achieving a higher precision score. However, IG provides a more stable precision over the training iterations. The K-PGD method, despite initial fluctuations, ultimately achieves the highest precision score, suggesting it is the most effective in enhancing the model's robustness and precision.

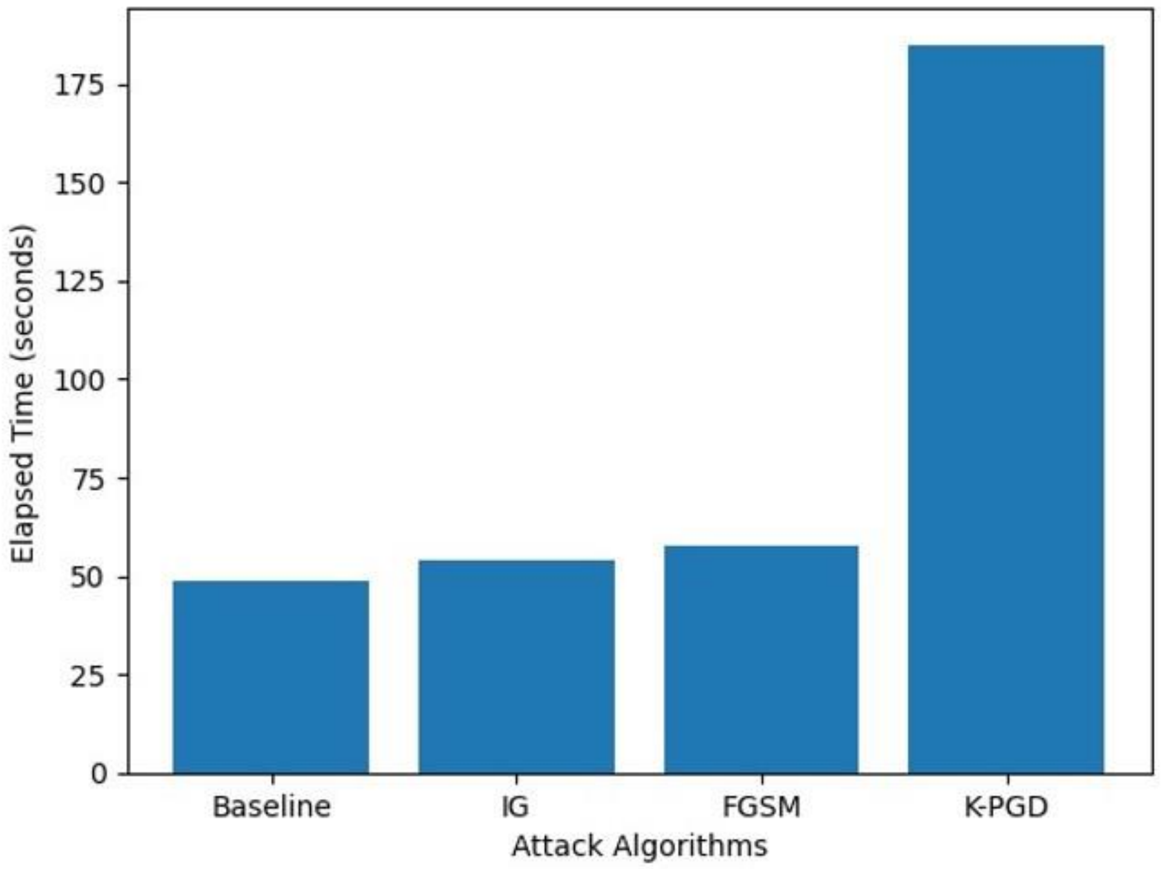
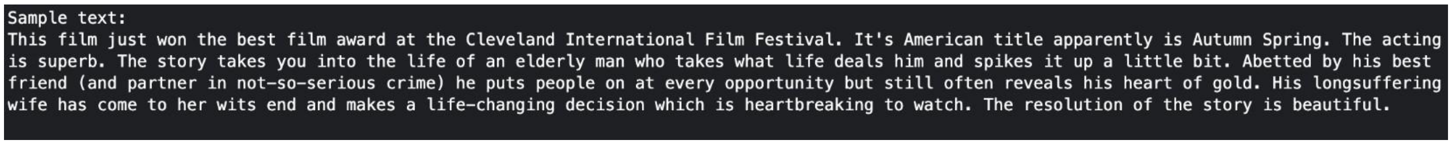


Figure 7: Comparison of Elapsed Time amongst different algorithms

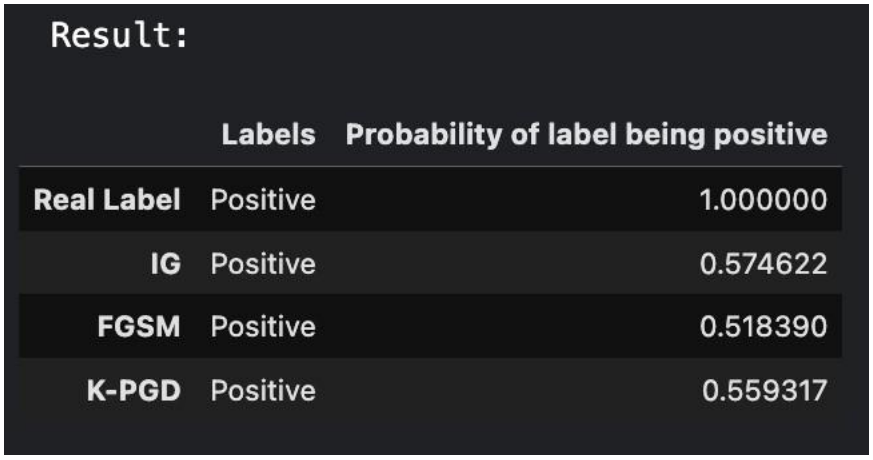
The Elapsed Time taken by K-PGD is quite high when compared to that of IG and FGSM. The graph shows a trade-off between computational efficiency and the robustness provided by the adversarial training techniques. While K-PGD may provide better robustness (as seen in accuracy scores), it comes at a higher computational cost. The higher elapsed time for K-PGD indicates that it is computationally more intensive, likely due to its iterative nature, where multiple steps are taken to refine the adversarial examples.

We also performed a evaluation of Integrated Gradients by testing sentiment of a sample text by generating different adversaries from the algorithms.



```
Sample text:
This film just won the best film award at the Cleveland International Film Festival. It's American title apparently is Autumn Spring. The acting
is superb. The story takes you into the life of an elderly man who takes what life deals him and spikes it up a little bit. Abetted by his best
friend (and partner in not-so-serious crime) he puts people on at every opportunity but still often reveals his heart of gold. His longsuffering
wife has come to her wits end and makes a life-changing decision which is heartbreaking to watch. The resolution of the story is beautiful.
```

Figure 8: Sample Text


Result:

	Labels	Probability of label being positive
Real Label	Positive	1.000000
IG	Positive	0.574622
FGSM	Positive	0.518390
K-PGD	Positive	0.559317

Figure 9: Result of label probabilities of being positive by every algorithm

In comparison to other algorithm the model trained with Integrated gradients adversaries are showing a bit better probability score than FGSM and K-PGD.

CONCLUSION

The main work of our project was to compare different adversarial training techniques namely IG, FGSM, K-PGD, in enhancing the robustness of a text-based sentiment analysis model. The model that we used was BiLSTM model, and we trained the model on multiple perturbed datasets. Our findings demonstrate the impact of adversarial training on model performance, with K-PGD emerging as the most effective method in improving model accuracy, minimizing loss and maximizing the number of correct classifications on adversarial examples. FGSM's performance over multiple metrics was very inconsistent and the fluctuations in the trends potentially indicated that FGSM generated weaker perturbations which in turn provided less robust adversarial examples for the model to train on. While IG showed a more balance approach it was better than FGSM in most comparisons and was matching with K-PGD over almost all metrics. IG had an upper hand over both FGSM and K-PGD in the Elapsed Time metric showing that it was much less computationally expensive both in time and computational resources. K-PGD suffers from a drawback of overfitting, which in simple terms means that it won't have consistent performance over different models and datasets. IG solves this problem with ease as it is a more scalable and generalizable algorithm as compared to the other two. Therefore, even though IG's performance was slightly lesser than that of K-PGD's, but due to its scalability to larger datasets and more generalizability Integrated Gradients is a much better options for the generation of adversarial attacks and for increasing the robustness of NLP models.

The dataset that we used was not a large one therefore the results that came out were more leaned towards KPGD, but for a larger dataset and more a variety of models, IG can possibly be a better option for adversarial generation and model security.

REFERENCES

1. Li Dianqi, Zhang Yizhe, Peng Hao, Chen Liqun, Brockett Chris, Ting Sun Ming, Dolan Bill(2020). "Contextualized perturbation for textual adversarial attack." Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).pp.8337–8347.
2. Alzantot Moustafa, Sharma Yash, Elgohary Ahmed, Ho Bo-Jhang, Srivastava Mani B., Chang KaiWei(2018). "Generating Natural Language Adversarial Examples." Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2018).pp.2890–2900.

3. X. Morris John, Lifland Eli, Yong Yoo Jin, Grigsby Jake, Jin Di, Qi Yanjun(2020). "TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP." Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL).pp. 2889–2900.
4. Wang Lina, Chen Xingshu, Tang Rui, Yue Yawei, Zhu Yi , Zeng Xuemei , Wang Wei(2020). "Improving adversarial robustness of deep neural networks by using semantic information." Proceedings of the AAAI Conference on Artificial Intelligence, 34.pp.5628–5635.
5. Pan Lin, Hang Chung-Wei, Sil Avi, Potdar Saloni(2022). "Improved Text Classification via Contrastive Adversarial Training." Proceedings of the AAAI Conference on Artificial Intelligence, 36.
6. Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). "Generative Adversarial Nets". Proceedings of the International Conference on Neural Information Processing Systems (NIPS'14). pp 2672–2680.
7. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). "Towards Deep Learning Models Resistant to Adversarial Attacks." Proceedings of the International Conference on Learning Representations (ICLR) , pp. 1-28.
8. Sundararajan, M., Taly, A., & Yan, Q. (2017). "Axiomatic Attribution for Deep Networks." Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS), pp. 5575-5585.
9. Wang, Y., Liu, J., Chang, X., Mišić, J., & Mišić, V. B. (2020). "IWA: Integrated Gradient based White-box Attacks for Fooling Deep Neural Networks." Proceedings of the 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3678-3686.
10. Chen, J., Song, L., Wainwright, M. J., & Jordan, M. I. (2018). "Defending Against Adversarial Attacks by Attributing Feature Importance." Proceedings of the 35th International Conference on Machine Learning (ICML), vol. 80, pp. 960-968.
11. Xu, H., Ma, Y., Liu, H., Deb, D., Liu, H., Tang, J., & Jain, A. K. (2020). "Adversarial Attacks and Defenses in Images, Graphs and Text: A Review." arXiv:1909.08072v3 [cs.LG], vol. 2020, pp. 1-35.
12. Bhagoji, A. N., He, W., Li, B., & Song, D. (2018). "Exploring the Space of Black-box Attacks on Deep Neural Networks." Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 1310-1325.
13. Wallace, E., Feng, S., Kandpal, N., & Gardner, M. (2020). "Universal Adversarial Triggers for Attacking and Analyzing NLP." Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 2153-2162.
14. Li, J., Ji, S., Du, T., Li, B., & Wang, T. (2019). "TextBugger: Generating Adversarial Text Against Real-world Applications." Proceedings of the 2019 Network and Distributed System Security Symposium (NDSS), pp. 1-15.

15. Huang, T., Menkovski, V., Pei, Y., & Pechenizkiy, M. (2021). "Bridging the Performance Gap between FGSM and PGD Adversarial Training." Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), pp. 1-10.
16. Zhu, C., Cheng, Y., Gan, Z., Sun, S., Goldstein, T., & Liu, J. (2020). "FREELB: Enhanced Adversarial Training for Natural Language Understanding." Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 8337-8347.
17. Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). "Explaining and Harnessing Adversarial Examples." https://arxiv.org/abs/1412.6572.
18. Tian, R., & Mao, Y. (2022). "On Robust Overfitting: Adversarial Training Induced Distribution Matters." Proceedings of the 2022 International Conference on Learning Representations (ICLR), pp. 1-12.

APPENDICES

APPENDIX 1:

Vector Embedding:

```
VOCAB_SIZE = 1000 encoder = layers.TextVectorization(max_tokens=VOCAB_SIZE)
encoder.adapt(train_ds.map(lambda text, label : text)) num_tokens =
len(encoder.get_vocabulary()) + 2 embedding_dim = 100 word_index =
dict(zip(encoder.get_vocabulary(), range(len(encoder.get_vocabulary())))) embed =
layers.Embedding(     num_tokens,     embedding_dim,     mask_zero=True,
trainable=False,
)
embed.build((1,))
embed.set_weights([embedding_matrix])
```

APPENDIX 2:

Bidirections Long Short Term Memory(BiLSTM) model:

```
model = models.Sequential([
    layers.Bidirectional(tf.keras.layers.LSTM(64)),
layers.Dense(64, activation='relu'),     layers.Dense(1,
activation='sigmoid')
])
model.summary()
```

APPENDIX 3:

Integrated Gradients method to generate Adversary:

```
def compute_gradients(t, model):
with tf.GradientTape() as tape:
        tape.watch(t)         probs =
model(t)     grads =
tape.gradient(probs, t)     return
grads

def get_igs(vec_embed, model, n_steps=20):
    baseline_embed = tf.zeros(shape=tf.shape(vec_embed))
interpolated_texts = interpolate_texts(baseline_embed, vec_embed, n_steps)
path_gradients = compute_gradients(interpolated_texts, model)

    all_grads = tf.reduce_sum(path_gradients, axis=0) / n_steps
x_grads = tf.math.multiply(all_grads, vec_embed)
ig_embed = vec_embed + x_grads     return ig_embed
```

APPENDIX 4:

Text Interpolation:

```
def interpolate_texts(baseline, vector, m_steps, alphas=0.01):
    delta = vector - baseline     texts
= baseline + alphas * delta
return texts
```

APPENDIX 5:

Fast Gradient Sign Method to generate perturbations:

```
def compute_gradients(t, model):
with tf.GradientTape() as tape:
        tape.watch(t)         probs =
model(t)     grads =
tape.gradient(probs, t)     return
grads

def get_fgsm(vec_embed, model, epsilon=0.01):     grads
= compute_gradients(vec_embed, model)
vec_embed_perturbed = vec_embed + (epsilon * grads)
return vec_embed
```

APPENDIX 6:

K step Projected Gradient Descent method to generate adversaries:

```
def kpgd_perturbations(vec_embed, model, n_steps=20, alpha=0.01, epsilon=0.1):
    perturbation = tf.zeros_like(vec_embed)
for _ in range(n_steps):
        with tf.GradientTape() as tape:
            tape.watch(vec_embed)             probs = model(vec_embed + perturbation)
grads = tape.gradient(probs, vec_embed         perturbation =
tf.clip_by_value(perturbation + alpha * grads, -epsilon, epsilon)     return
vec_embed + perturbation
```

APPENDIX 7:

Model Training:

```
def train_model(model, encoder, embed, train_ds, use_ig=False, use_fgsm=False, use_kpgd=False, epochs=10,
data_size=10):
   K.clear_session()
model.compile(optimizer='adam',
loss='binary_crossentropy',
metrics=['accuracy', 'precision'])     config = []

   for epoch in range(epochs):
print(f'epoch:{epoch}')         for text, label in
train_ds.take(data_size):            label =
tf.expand_dims(label, axis=-1)            vec =
encoder(text)            vec_embed =
embed(vec)            if use_ig:
            vec_embed = get_igs(vec_embed, model)
if use_fgsm:               vec_embed =
get_fgsm(vec_embed, model)            if use_kpgd:
            vec_embed = kpgd_perturbations(vec_embed, model)
         config.append(model.train_on_batch(vec_embed, label, return_dict=True))
return (config, model)
```

APPENDIX 8:

Hits and Miss evaluation metric:

```
def eval_hits(model, ds, encoder, embed, steps=10):
   hits = []    miss = []     for text,
label in ds.take(steps):         vec
= encoder(text)         vec_embed
= embed(vec)
```

```
        vec_embed_ig = get_igs(vec_embed, hist)
pred_ig = hist.predict(vec_embed_ig, verbose=0)

        pred_ig[pred_ig<0.5] = 0
pred_ig[pred_ig>=0.5] = 1         pred_ig
= pred_ig.astype('int64')         pred_ig =
np.squeeze(pred_ig)

        truth_ig = pred_ig == label
hits.append(np.count_nonzero(truth_ig))         miss.append(len(label)
- np.count_nonzero(truth_ig))

    return hits, miss
```

APPENDIX 9:

Accuracy and Loss metrics:

Accuracy:

```
def plot_acc(config_base, config_ig, config_fgsm, config_kpgd):
acc = []     acc_ig = []     acc_fgsm = []     acc_kpgd = []

    for param in config:
        acc.append(param['accuracy'])
for param_ig in config_ig:
        acc_ig.append(param_ig['accuracy'])
for param_fgsm in config_fgsm:
        acc_fgsm.append(param_fgsm['accuracy'])
for param_kpgd in config_kpgd:
        acc_kpgd.append(param_kpgd['accuracy'])
plt.plot(acc, label='Baseline(BiLSTM)')
plt.plot(acc_ig, label='Baseline with IG')
plt.plot(acc_fgsm, label='Baseline with FGSM')
plt.plot(acc_kpgd, label='Baseline with K-PGD')
plt.xlabel('Number of Epochs')
plt.ylabel('Accuracy score')     plt.title('Accuracy
Over Training Epochs')     plt.legend()
plt.show()
```

Loss:

```
def plot_loss(config_base, config_ig, config_fgsm, config_kpgd):
loss = []     loss_ig = []     loss_fgsm = []     loss_kpgd= []

    for param in config:
        loss.append(param['loss'])
for param_ig in config_ig:
        loss_ig.append(param_ig['loss'])
for param_fgsm in config_fgsm:
        loss_fgsm.append(param_fgsm['loss'])
for param_kpgd in config_kpgd:
        loss_kpgd.append(param_kpgd['loss'])
plt.plot(loss, label='Baseline(BiLSTM)')
plt.plot(loss_ig, label='Baseline with IG')
plt.plot(loss_fgsm, label='Baseline with FGSM')
plt.plot(loss_kpgd, label='Baseline with K-PGD')
plt.xlabel('Number of Epochs')     plt.ylabel('Loss')
plt.title('Loss Over Training Epochs')
plt.legend()     plt.show()
```

APPENDIX 10:

Precision and Elapsed time metrics:

Precision:

```
def plot_precision(config_base, config_ig, config_fgsm, config_kpgd):
acc = []     acc_ig = []     acc_fgsm = []     acc_kpgd = []
```

```
    for param in config:
        acc.append(param['precision'])

    for param_ig in config_ig:
        acc_ig.append(param_ig['precision'])

    for param_fgsm in config_fgsm:
        acc_fgsm.append(param_fgsm['precision'])

    for param_kpgd in config_kpgd:
        acc_kpgd.append(param_kpgd['precision'])

    plt.plot(acc, label='Baseline(BiLSTM)')
plt.plot(acc_ig, label='Baseline with IG')
plt.plot(acc_fgsm, label='Baseline with FGSM')
plt.plot(acc_kpgd, label='Baseline with K-PGD')
plt.xlabel('Number of Epochs')
plt.ylabel('Precision score')     plt.title('Precision
Over Training Epochs')     plt.legend()
plt.show()
```

Elapsed time:

```
def plot_elapsed_time(elapsed_baseline, elapsed_ig, elapsed_fgsm, elapsed_kpgd):
algorithms = ['Baseline(BiLSTM)','IG', 'FGSM', 'K-PGD']     elapsed_times =
[elapsed_baseline, elapsed_ig, elapsed_fgsm, elapsed_kpgd]

    plt.bar(algorithms, elapsed_times)     plt.xlabel('Attack
Algorithms')     plt.ylabel('Elapsed Time (seconds)')
plt.title('Elapsed Time for Different Attack Algorithms')
plt.show()
```

REFLECTIONS OF THE TEAM MEMBERS ON THE PROJECT

Pulkit Tyagi:

This project was a rewarding experience, allowing me to dive deep into the crucial area of adversarial robustness in NLP. I enjoyed coding, documenting, and conducting experiments. The results highlighted the effectiveness of K-PGD and the potential of IG as a scalable, efficient, and less prone to overfitting approach. I was particularly interested in the project's relevance to real-world applications and its focus on improving the trustworthiness of NLP systems. The biggest challenge was understanding the nuances of each attack algorithm and their impact on the model's behaviour. This project significantly improved my understanding of adversarial attacks, adversarial training, and the importance of building robust NLP models.

Manish Patro:

I am satisfied with my contributions to the project, specifically focusing on the model diagram and coding. It was fascinating to work with diverse attack algorithms and analyze their performance across multiple metrics. The comparison of K-PGD, FGSM, and IG provided valuable insights into their strengths and weaknesses. This project significantly broadened my understanding of NLP models and adversarial robustness. The biggest challenge was understanding the mathematical concepts behind IG and implementing it efficiently.

Vikash Kumar Arya:

My primary focus was on the literature survey and contributing to the introduction section. This project illuminated the vulnerabilities of sentiment analysis models and the importance of developing more robust defenses. I was particularly interested in understanding how different attack algorithms manipulate model behavior and the strategies for mitigating these threats. Staying up-to-date on the latest research in adversarial attacks and maintaining a comprehensive literature review was the most challenging aspect of this project.

Akash Bera:

This project helped me grasp the concepts of adversarial robustness and the challenges it presents for NLP models. I enjoyed conducting the literature survey and contributing to the introduction section. The comparison of different attack algorithms was insightful, especially the advantages of IG in terms of scalability and efficiency. I had to initially overcome the challenge of understanding the complex mathematical concepts behind these algorithms.