
Project Initial Concept/Design

(FIT3161 Final Year Project: Assignment)

Prepared by : MCS11

Alvin Andrean 33279071

Sheng Xin Chua 32837933

Steven Tirtadjaja 33068410

Marcholas Tjangnaka 33453128

Monash University Malaysia
20-09-2024

Table of Contents

1. Introduction.....	3
2. Software Components and Decomposition:.....	4
a. Main Data to be Processed.....	4
b. Data Processing Components and Flow.....	4
2.1. Software Component Relationships:.....	4
2.2. Non-Functional Requirements:.....	5
2.3. Development and User Platforms:.....	5
2.4. Deliverable Software:.....	6
3. Representation.....	7
3.1. Flow Chart Diagram.....	7
3.1.1. Flow Chart Breakdown.....	7
3.2. Website/User Interface Mockup.....	9
3.3.1. User Interface Breakdown.....	11
4. Software Specifications.....	13
4.1 Justifications for Software Specifications:.....	15
4. Hardware Specifications.....	17
4.1 Justifications for Hardware Specifications:.....	18
5. References.....	19

1. Introduction

The rapid advancement of Generative AI dazzled the world with its diverse applications in education (ChatGPT), entertainment (DALL-E), biology (AlphaFold) and chemistry (GNoME). Our project that is titled "Abyss: AI Meets the Dark Side" intends to explore the vulnerabilities and threats that are faced by AI systems. This project experiments on adversarial attacks on AI models to achieve this goal. This benefits the community of AI users as it uncovers the vulnerabilities and potential weaknesses of the AI models. Moreover, in the process of trying to undermine AI models, it is possible to trace back the reasoning and find potential ways to prevent it.

The focus of this project is to conduct adversarial attacks on image classification models with a specific emphasis on the Fast Gradient Sign Method (FGSM). FGSM is a well known technique for generating adversarial examples which are carefully crafted inputs designed to deceive an AI model into making incorrect predictions. By applying FGSM to image classification systems, the project aims to manipulate these models in a way that exposes their weaknesses and vulnerabilities (Sen & Dasgupta, 2023b).

Using the FGSM adversarial technique, we aim to explore how neural networks can be manipulated to misclassify images, despite their high accuracy under normal conditions (Sen & Dasgupta, 2023b). This pragmatic approach offers an opportunity to deeply understand the process of generating adversarial examples and evaluate how even the most robust models can be compromised. Our primary objective is to raise awareness about the vulnerabilities of AI models in image classification tasks and to demonstrate the level of ease with which adversarial attacks can deceive these systems.

While the project focuses on executing these attacks, it also showcases the need for more robust AI systems by examining the weaknesses uncovered through our adversarial techniques. Rather than implementing specific defences, our goal is to provide detailed insights into the types of attacks AI models are susceptible to in an effort to promote awareness of potential security risks. This approach will empower developers and researchers to explore, develop and implement their own defence mechanisms or countermeasures in the future, thereby strengthening AI systems against such adversarial threats.

Users will be guided through the technical and intricate aspects of adversarial AI with a focus on practical applications in image classification. The project will showcase how attacks like FGSM can undermine AI accuracy, include visualisations for better analysis of model behaviour under attack and demonstrate the limitations of current defences. The results will offer insight into the need for more secure and resilient AI systems in domains that rely heavily on image classification technologies.

2. Software Components and Decomposition:

a. Main Data to be Processed

The system will process the following types of data:

- **Images from ImageNet Dataset:** Used to pre-evaluate the performance of the models we will be using
- **User Images:** Uploaded or selected images for adversarial attack processing.
- **Adversarial Examples:** Modified images generated by the Fast Gradient Sign Method (FGSM) attack.
- **Models:** Pre-trained models for selection and application of the attack. These models (pre-trained) are the minimum requirement; in the future, there may be exploration into retraining or training new models.
- **Metrics Data:** Includes confidence scores, classification results, and other performance metrics before and after the attack.
- **User Authentication Data:** Data required for secure user login and access management, including OAuth tokens, user credentials (handled securely), and session information. This will ensure that only authorised users can access the system and perform operations.

b. Data Processing Components and Flow

- **Image Upload Processing:** Preprocesses images before feeding them into the selected model.
- **Adversarial Attack Engine:** The core processing unit that applies the FGSM attack to the image and generates adversarial examples.
- **Metrics Calculation Engine:** Compares the original and adversarial image classification results and computes the relevant metrics, e.g., confidence score changes, loss function value, etc.

Data Flow: User uploads/selects an image → User modifies epsilon value → User selects a model available on the platform → The adversarial engine applies the FGSM attack → The adversarial image is generated → The Metrics engine computes the attack's impact → Results and visualizations are presented to the user via the UI.

2.1. Software Component Relationships:

- **User Interface (Frontend):** The UI communicates with the backend to upload or retrieve images, initiates the FGSM attack and displays the results to users. Users can set the epsilon value via a slider to control the perturbation value before applying it to the model. This functionality is implemented using HTML/CSS with React.js, providing real time feedback as users adjust epsilon and see its effect on model performance.

- **Backend (Server):** Manages requests from the UI, processes images using the FGSM attack and interacts with the database to store and retrieve metrics and model information. Implemented using Python and Flask.
- **Database:** Stores uploaded images, model details and performance metrics. Managed by MySQL or Oracle.
- **Adversarial Attack Engine:** Interacts with both the image processing module and the database. Receives input from the UI, retrieves model data from the database and processes the image to generate adversarial examples using Cleverhans library and TensorFlow.
- **Metrics Engine:** Integrated into the backend to calculate and store metrics after the attack and pass this information back to the frontend for Matplotlib visualisation.

2.2. Non-Functional Requirements:

- **Quality:** The system must offer a responsive and intuitive user interface with minimal latency when uploading images and generating adversarial examples. Consistent performance across different devices is vital for creating a seamless user experience.
- **Accuracy:** The FGSM attack must generate adversarial examples that reliably mislead the AI model without compromising the integrity of the original image; adversarial perturbations should be minimal but effective.
- **Reliability:** The system must be able to handle large scale image data and multiple requests simultaneously. This includes maintaining the integrity of stored images and data across user sessions and preventing failures when the attack is ongoing.
- **Security:** Since users are uploading their images, the system must ensure data protection in terms of preventing unauthorised access or manipulation of personal and sensitive content. Authentication will be handled using Google OAuth 2.0.

2.3. Development and User Platforms:

Development Platform:

- **Backend:** The server side logic will be implemented using Python 3.10+ with Flask. The adversarial attack relies on machine learning libraries like TensorFlow to apply the FGSM attack.
- **Frontend:** The user interface will be developed using HTML/CSS, JavaScript and modern frameworks such as React.js for dynamic content rendering.
- **Database:** A relational database like MySQL or Oracle will store user data, images, model information and performance metrics.

User Platform:

- The platform will be web-based, accessible from any common web browser such as Chrome, Firefox, etc.

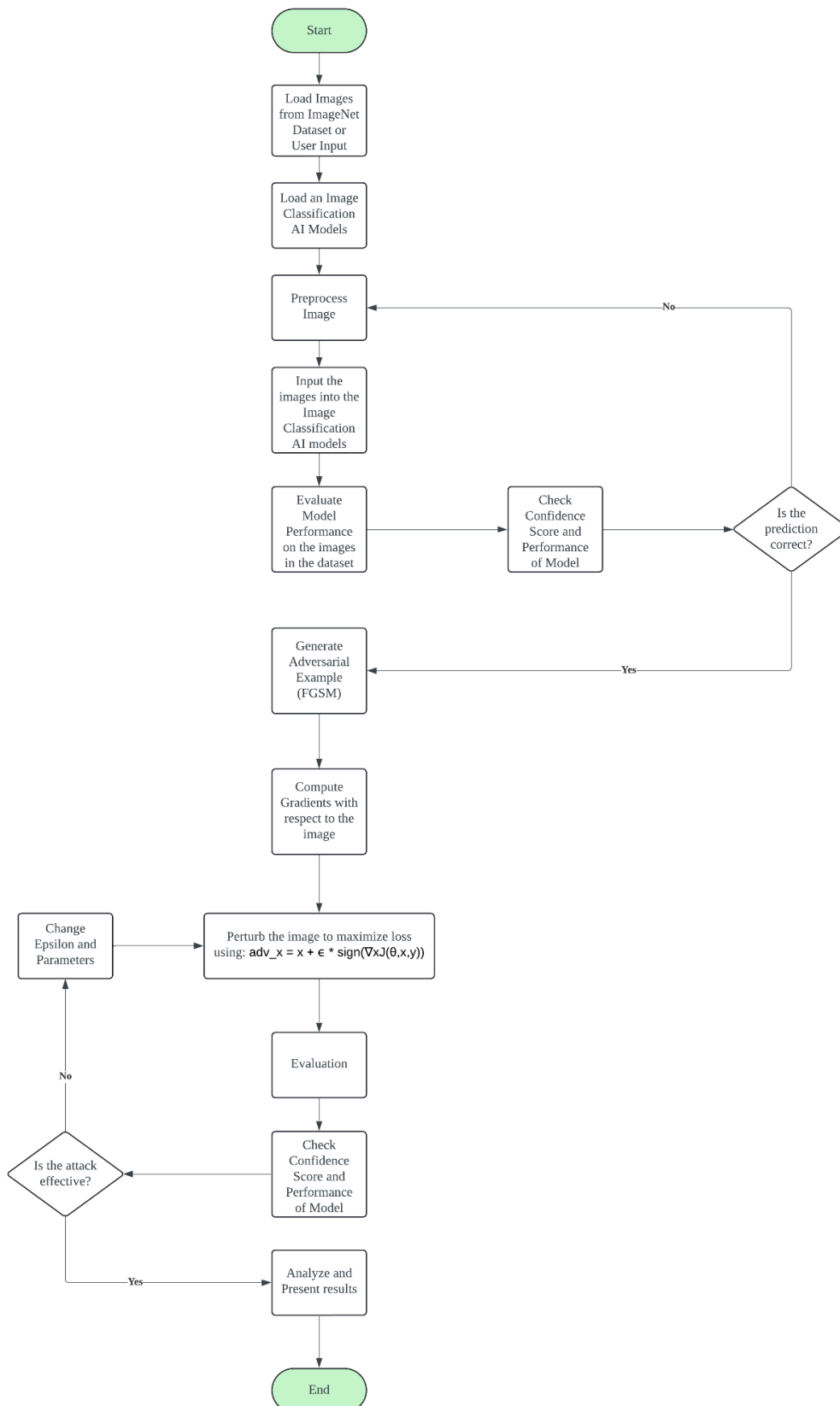
2.4. Deliverable Software:

The final deliverable will be a fully functional web application that allows users to:

- Upload or select images for adversarial attacks.
- Choose from a list of available models.
- Apply the Fast Gradient Sign Method (FGSM) attack to generate adversarial examples.
- Set the epsilon value to control perturbation strength and apply the Fast Gradient Sign Method (FGSM) attack to generate adversarial examples.
- View side by side comparisons of original and adversarial images.
- Analyse (compare image quality before and after) and visualise the impact of the attack using detailed performance metrics.

3. Representation

3.1. Flow Chart Diagram



The flow begins with an initial "Start" action that triggers interaction with the platform. Next, images are loaded either from custom input or from the ImageNet dataset available on the platform. This flexibility allows work with both custom and standardized datasets. Following image upload, a pretrained image classification model is selected, which influences the outcome of the adversarial attack. After model selection, the images are preprocessed by scaling, resizing, or normalizing to ensure compatibility with the model and enhance its performance. These preprocessed images are then fed into the model for initial predictions and analysis, setting a baseline before the attack is applied.

The system evaluates the model's confidence and accuracy in its predictions. If the model predicts correctly with high confidence, the flow proceeds to generate an adversarial example. If the prediction is incorrect or confidence is low, preprocessing parameters can be adjusted before rerunning the prediction. For generating adversarial examples, the **Fast Gradient Sign Method (FGSM)** is applied. The adversarial example is crafted by modifying the input image using the following formula:

- $\text{adv_x} = x + \epsilon * \text{sign}(\nabla_x J(\theta, x, y))$ (Sen & Dasgupta, 2023c)

Compute the Gradient:

The platform calculates the gradient of the loss function (J) with respect to the input image (x). This gradient helps identify how each pixel influences the model's prediction, guiding the direction of the perturbation. (Sen & Dasgupta, 2023c)

Perturb the Image:

Using the computed gradient, a small perturbation, scaled by epsilon (ϵ), is added in the direction of the gradient to maximize the loss function. (Sen & Dasgupta, 2023c)

- adv_x = the adversarial image as the output
- x = the original input image
- y = the actual class of the input image
- ϵ = The noise intensity is a small fraction used to scale the signed gradients, creating subtle perturbations that remain visually indistinguishable from the original image.
- θ = the neural network model used for image classification
- J = the loss function

Apply Epsilon (Perturbation Strength):

Users can adjust the epsilon value to control the intensity of the perturbation. A higher epsilon value creates more noticeable changes to the image, making it more likely to fool the model but potentially making the perturbation visible to humans.

The perturbed image is reevaluated to measure how effectively the adversarial attack has impacted the model's prediction. The confidence score and performance of the model are checked again to assess how much the adversarial attack has reduced the model's prediction confidence. If the attack is effective enough and significantly alters the model's performance, the system proceeds to analyse and present the results. If the attack is ineffective, the flow loops back and allows users to adjust the epsilon value and other attack parameters before attempting the attack again.

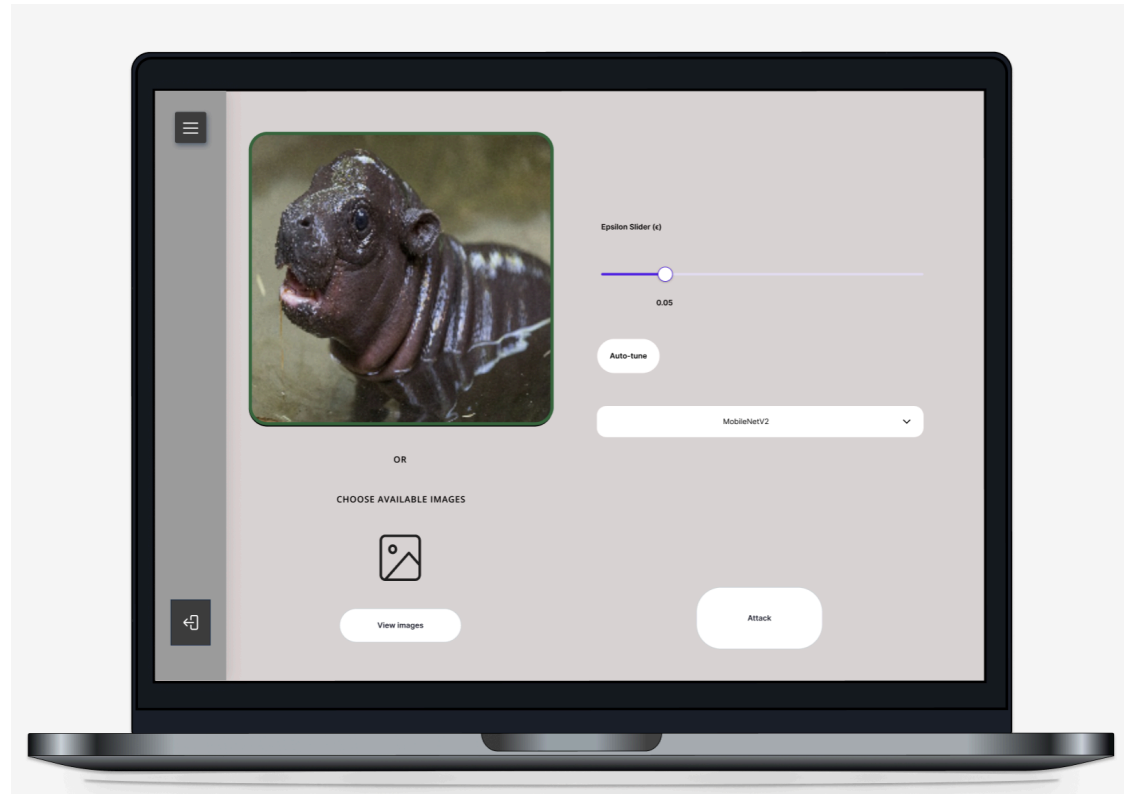
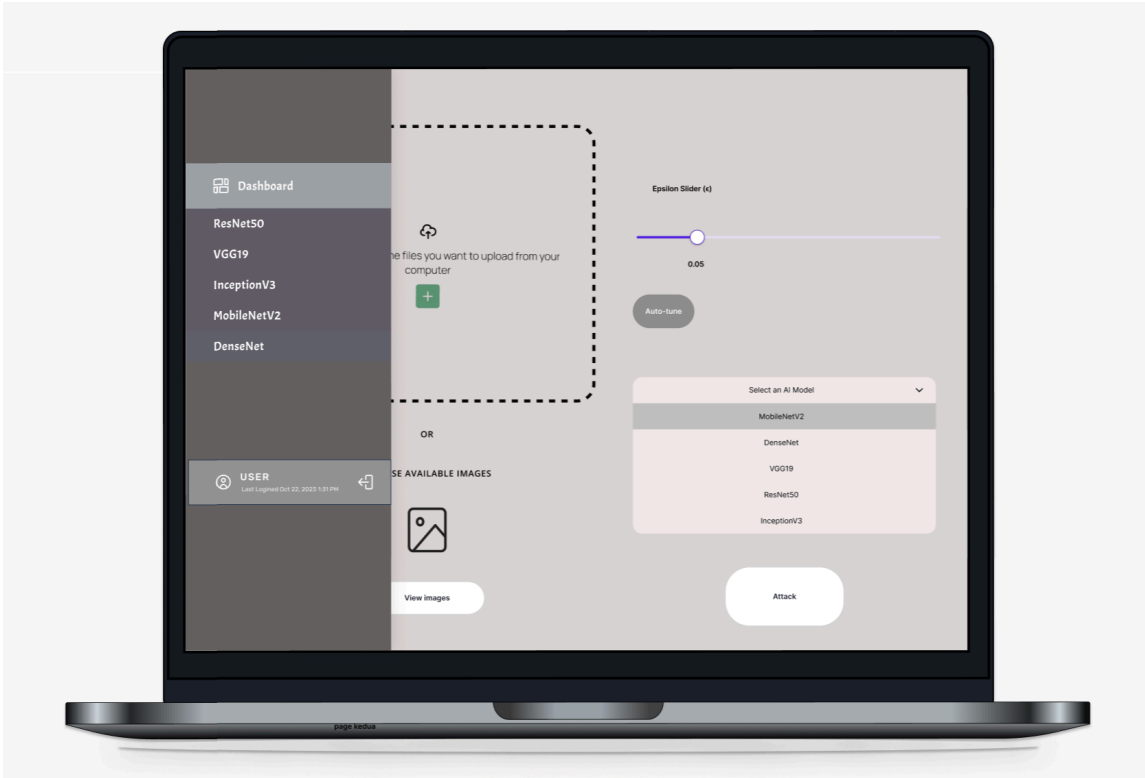
If necessary, users can change the epsilon and parameters to refine the attack, thereby continuously strengthening the perturbation until the attack becomes effective. Once the attack is successful, the results are analysed and presented to the user including performance metrics, confidence scores and side by side comparisons of the original and adversarial images. Finally, after presenting the results,

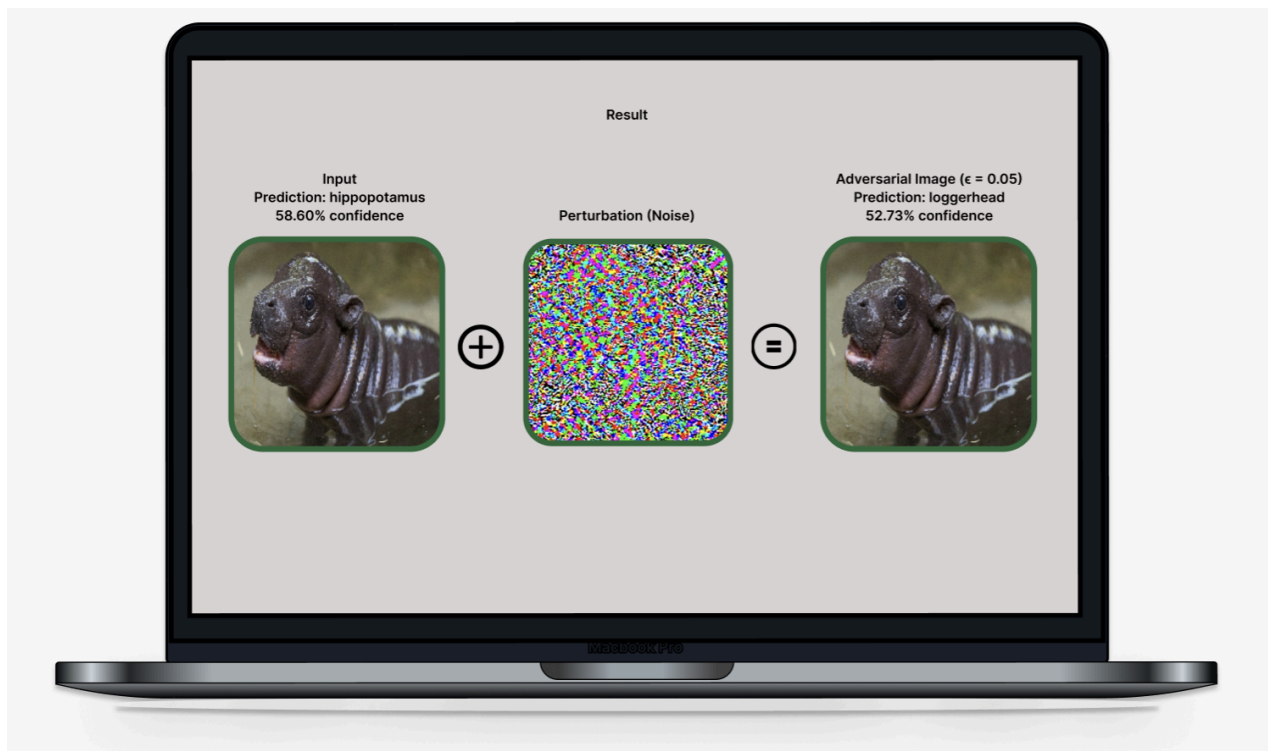
the flow concludes and the user has successfully completed an adversarial attack on the chosen model, allowing them to interpret the outcomes.

3.2. Website/User Interface Mockup

Show visually model structure, dropdown menu for models show the structure of the models, slider instead of a dropdown (diagram, user can slide through)







3.3.1. User Interface Breakdown

The user interface (UI) is designed to provide a smooth and intuitive experience for generating and analyzing adversarial images using the Fast Gradient Sign Method (FGSM). The design focuses on ease of use and clarity, ensuring that each step aligns with the data flow.

The first step involves the Image Upload/Selection Module, where users are provided with a simple interface to either upload their own images or select from a library of existing images on the platform. This feature is designed to minimize friction in the user experience by offering clear buttons and easy access to image options. The flexibility of choosing between personal data and pre-existing samples makes this an important initial step in the UI design.

Once the image is uploaded or selected, the user moves to the Model Selection Interface. Here, users are prompted to select from a list of pre-trained image classification models for the adversarial attack. The interface is designed with a dropdown menu or selection grid to simplify interaction and offers clear options without overwhelming users. The availability of pre-trained models ensures that users can efficiently test different adversarial attacks without difficulty.

Additionally, a Sidebar has been integrated into the UI. The sidebar opens up to display a selection of models. Once a user clicks on a model, it will display the structure of the selected model along with the description. This feature provides users with an understanding of the model architecture and its key components, enabling them to make informed choices about which model to target for the adversarial attack.

Next, the Epsilon Slider plays a critical role. This slider allows users to adjust the strength of the perturbation applied to the image. Before the attack is launched, the user can set the desired epsilon value to control how strong the attack will be. The slider's placement is intuitive, enabling real-time adjustments that visually and functionally connect with the adversarial engine. This gives users precise control over the strength of the FGSM attack.





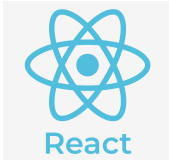

In addition to manual epsilon adjustment, an Auto-Tuning Epsilon mechanism is in place. This system begins with a small epsilon value and iteratively increases it until the model is fooled. The goal is to find the smallest effective epsilon value that successfully misclassifies the image, balancing perturbation strength and attack success.







Behind the scenes, Backend Processing takes over after the epsilon value is set. The adversarial engine applies the FGSM attack in the background and generates the adversarial image from the user-selected model and image. A loading animation or progress indicator provides visual feedback to users, informing them that the system is working, thereby reducing potential confusion or impatience.



Following the attack, the Metrics Calculation engine computes various performance metrics, including model accuracy, confidence scores, and the level of distortion caused by the attack. These metrics are automatically calculated and displayed, allowing users to quickly understand the impact of their perturbation choices without needing to run separate analyses.

Finally, the Result Visualisation Dashboard presents the original and adversarial images side by side, accompanied by detailed performance metrics and analysis of the attack. This dashboard offers a clear visual comparison that shows how the perturbation affected the image and the model's predictions. Paired with performance metrics, this setup provides insights that help users easily interpret the success and impact of the FGSM attack in a well-organized, user-friendly display.

4. Software Specifications

Index	Topic	Software
1	Operating System	Windows 10 or later 
2	Programming Language	Python 3.10+  HTML/CSS  JavaScript 
3	Frameworks	ReactJS for Front-End  Flask for Back-End  Flask

4	IDE (Programming Language Environment)	Visual Studio Code (VSCode),  Jupyter Notebook/Google Colab 
5	Code Repository	Github 
6	Version Control	Git
7	Adversarial Attack Libraries	Cleverhans 
8	Machine Learning Frameworks	TensorFlow 
9	Visualisation Libraries	Matplotlib
10	Database Management	MySQL or Oracle 

11	Image Processing	Pillow 
12	Dataset	ImageNet 

4.1 Justifications for Software Specifications:

1. Operating System: We chose Windows 10 or later as the operating system because of its widespread compatibility with various development tools and libraries. It makes sure that all team members regardless of their machine setups can integrate their work and share resources. Additionally, Windows offers native support for powerful IDEs and other software such as Visual Studio Code, TensorFlow, and Git for it to be an optimal choice for this project.
2. Programming Language: Python 3.10+ is selected as our core programming language due to its versatility and extensive support for machine learning and deep learning libraries including TensorFlow, Cleverhans and Matplotlib. Python's simple syntax and strong community make it a widely preferred language for both development and research purposes. Furthermore, Python introduces useful features like better error messages and new structural pattern matching syntax that improves the development experience and eliminates errors during the adversarial attack implementations.
3. Frameworks: ReactJS for Front-End, Flask for Back-End

ReactJS is chosen for the front-end development because of its efficiency in creating dynamic and responsive user interfaces. It allows for integration with the backend while ensuring a smooth user experience when displaying results of adversarial attacks and image visualisations.

Flask is selected for the backend due to its lightweight nature and integration with Python-based machine learning libraries such as TensorFlow and Cleverhans (Detimo, 2019). Its simplicity enables swift development and transparent communication between the user interface and the adversarial attack engine (Detimo, 2019).

4. IDE: We chose Visual Studio Code (VS Code) due to its flexibility, extensions, and strong support for development. It allows for an easy process of debugging, code management, and GitHub integration, making it ideal for team collaboration. As Kramer (2024) points out, VS Code is known for being fast, lightweight, and highly customizable, which makes it an excellent choice for a wide range of development tasks, without consuming significant system resources (Kramer, 2024).
5. Jupyter Notebook and Google Colab are selected additionally for their interactive development environment that is essential for prototyping and experimentation with adversarial attacks. These platforms are particularly useful for data visualisation and iterative testing by allowing real-time code execution and results, which are critical when tuning adversarial attack parameters.
6. Code Repository: GitHub was chosen for code version control and repository management due to its broad use, ease of collaboration and integration with Visual Studio Code. GitHub also allows tracking changes and maintaining multiple branches and version control while multiple team members work on different parts of the project simultaneously.
7. Version Control: Git is selected as the version control system for its robust tracking of changes that enables team members to collaborate effectively by branching and merging code safely. It ensures the integrity of the codebase, prevents conflicts during development and minimising errors.
8. Adversarial Attack Libraries: Cleverhans is chosen as the primary adversarial attack library due to its specific focus on adversarial machine learning. It provides certain implemented attacks such as the Fast Gradient Sign Method (FGSM) attack, Carlini & Wagner Attack (CW) which will help us to learn and incorporate adversarial techniques without implementing them from scratch. Cleverhans is also well maintained and possesses compatibility with popular deep learning frameworks like TensorFlow which we rely on.
9. Machine Learning Frameworks: We opted for TensorFlow as the deep learning framework due to its flexibility and scalability in handling complex neural network computations required for adversarial attacks. TensorFlow is production ready and ideal for building, training and deploying models in a variety of environments. Additionally, TensorFlow's compatibility with Cleverhans enables smooth integration of the Fast Gradient Sign Method (FGSM) adversarial attack into our system. We will utilise several pretrained models such as DenseNet201, ResNet50, InceptionV3, and MobileNetV2 that will provide diverse architectures for the testing of effectiveness of adversarial attacks. These are established models in the field of image classification and will serve as robust baselines for our experiments. In the future, we may explore retraining these models or even creating and training our own models with the help of TensorFlow.

10. Visualisation Libraries: Matplotlib was selected for visualising the results of adversarial attacks. It offers a plethora of plotting tools for visualising metrics, performance graphs, and comparisons of original vs. adversarial images. The library is integrated with other Python tools and provides flexibility in customising graphs and charts to meet our project's needs (TestGorilla, 2022).
11. Database Management: We chose MySQL or Oracle for database management due to their reliability and scalability in handling large datasets. These databases provide efficient data storage and retrieval capabilities, particularly for storing user uploaded images, model performance metrics and adversarial results. They also support integration with Python-based backends and enable smooth data management.
12. Image Processing: Pillow is chosen for image preprocessing due to its simplicity and compatibility with other Python libraries (Boguszewski, 2024). It enables efficient handling of images, including resizing, normalisation and format conversion (Boguszewski, 2024) which are crucial for preparing images specifically for the Fast Gradient Sign Method (FGSM) attack.
13. Dataset: ImageNet is selected as the dataset for model evaluation due to its large scale and diverse range of images (*ImageNet*. (n.d.)), making it ideal for testing pre-trained models such as DenseNet201, ResNet50, InceptionV3, and MobileNetV2, which were trained using images from this dataset. ImageNet's comprehensive dataset makes certain that our adversarial techniques are tested on realistic, complex inputs, providing robust results and insights into the models' vulnerabilities. In the future, when we retrain or create and train our own models, this dataset could prove to be valuable in the end.

4. Hardware Specifications

Index	Topic	Software
1	CPU	Intel Core i7 or later / AMD Ryzen 7 or later
2	GPU	NVIDIA® RTX Series GPU
3	RAM	16GB or more
4	Storage	512GB SSD or more
5	External Storage	External SSD (512GB or more) for backup and quick access to large datasets
6	Type of personal computer	Desktop/Laptop

4.1 Justifications for Hardware Specifications:

1. CPU: We have chosen Intel Core i7 or AMD Ryzen 7 as the minimum CPU requirement due to their strong multi-core performance, which is essential for handling complex computations, including adversarial attack generation and model training. Both processors provide the necessary power for efficient machine learning tasks and are conducive to amiable performance during the development and testing phases even when processing large datasets or running multiple tasks simultaneously (Maxtang, 2024 February 4).
2. A NVIDIA RTX Series GPU is selected for its superior computational capabilities, especially for handling tasks that require large amounts of processing power such as deep learning and image processing. This GPU makes sure that our models, including those used for adversarial attacks can manage high resolution image datasets and complex calculations efficiently, providing the necessary speed and performance for the project's requirements. Once we explore retraining or creating our own image classification AI model, a high end GPU will be crucial due to the expensive and resource intensive nature of model training. Training deep learning models involves substantial computational demands and a high end GPU will be needed to handle these efficiently and reduce training times.
3. RAM: 16GB or more 16GB or more RAM is essential for handling large datasets, running multiple processes and avoiding bottlenecks during model training and adversarial attack generation. This amount of memory accelerates operation when managing large image data, loading pretrained models and storing intermediate results without performance degradation.
4. Storage: 512GB SSD or more 512GB SSD or more is selected to provide fast read/write speeds and has quicker access to data and faster application performance. An SSD significantly improves the system's responsiveness, particularly when loading large datasets, saving model checkpoints, or processing adversarial attacks. The storage capacity ensures sufficient space for storing datasets, models, and results.
5. External Storage: External SSD (512GB or more) An External SSD (512GB or more) is chosen for backup and quick access to large datasets. This allows users to efficiently store and retrieve data when dealing with high-resolution images and extensive adversarial examples. The external storage is portable and easier to move data across different machines and environments when needed.
6. Type of Personal Computer: Desktop/Laptop Desktops or laptops provide flexibility in the development and testing environments. Desktops are often preferred for their superior upgrade options and higher performance capabilities in terms of GPU power. Laptops on the other hand offer portability, this allows developers to work in multiple locations without compromising on performance when considering the high end specifications recommended.

5. References

1. Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. *International Conference on Learning Representations*.
<https://ai.google/research/pubs/pub43405>
2. *Adversarial example using FGSM*. (n.d.). TensorFlow.
https://www.tensorflow.org/tutorials/generative/adversarial_fgsm
3. Cleverhans-Lab. (n.d.). *cleverhans/cleverhans/tf2/attacks/fast_gradient_method.py at master · cleverhans-lab/cleverhans*. GitHub.
https://github.com/cleverhans-lab/cleverhans/blob/master/cleverhans/tf2/attacks/fast_gradient_method.py
4. Sen, J., & Dasgupta, S. (2023, July 5). *Adversarial Attacks on Image Classification Models: FGSM and Patch Attacks and their Impact*. arXiv.org. <https://arxiv.org/abs/2307.02055>
5. Sarkar, T. R., Das, N., Maitra, P. S., Some, B., Saha, R., Adhikary, O., Bose, B., & Sen, J. (2024, April 5). *Evaluating adversarial robustness: a comparison of FGSM, Carlini-Wagner attacks, and the role of distillation as defence mechanism*. arXiv.org.
<https://arxiv.org/abs/2404.04245>
6. *ImageNet*. (n.d.). <https://image-net.org/download.php>
7. Kramer, N. (2024, February 25). Visual Studio Code: Read this before you get started. Daily.dev.
<https://daily.dev/blog/visual-studio-code-read-this-before-you-get-started#:~:text=VS%20Code%20is%20generally%20faster,as%20much%20as%20VS%20Code>
8. Boguszewski, A. (2024, July 2). Pillow in Python: How to Process Images? - Adrian Boguszewski - medium. *Medium*.
<https://medium.com/@adrianboguszewski/a-quick-overview-of-pillow-in-python-an-essential-library-for-image-processing-96b8fc0e7c20>
9. Detimo. (2019, November 18). *Python Flask: pros and cons*. DEV Community.
<https://dev.to/detimo/python-flask-pros-and-cons-1mlo>
10. TestGorilla. (2022, December 5). *What is Matplotlib in Python? Top 10 advantages of Matplotlib that you should know*. <https://www.testgorilla.com/blog/matplotlib-in-python/>
11. Maxtang. (2024, February 4). *AMD Ryzen 7 vs Intel i7: Which CPU should you choose - Maxtang PC Retail Store*. Maxtang PC Retail Store.
<https://store.maxtangpc.com/amd-ryzen-7-vs-intel-i7/>