MRI Brain Alzheimer Classification Under Adversarial Attacks

Lerman, Dor
dorlerman@campus.technion.ac.il

Bar-Tov, Niv
nivba@campus.technion.ac.il

October 31, 2024

GitHub link: https://github.com/nivbartov/MRI_Brain_Alzheimer_Classification

Abstract

In this project, we aim to design a robust model for detecting and classifying Alzheimer disease using MRI brain images. The model simulates a radiologist's diagnostic process by classifying images into four severity levels. We evaluate and compare several well-known unsupervised pre-trained models for classification tasks. Then, we train and evaluate these models under adversarial attacks, which can significantly reduce model's performance. By combining the models, we aim to create an ensemble, a unified and robust model that maximizes resilience against adversarial attacks while maintaining high classification performance.

1 Introduction

In the realm of cyber security, efforts are made to protect computing systems from digital attacks, which are an emerging threat nowadays. In machine learning, attackers develop adversarial attacks that are designed to trick models using deceiving data. This deceptive data is given to the models as an input, causing classifiers to make incorrect classifications. Particularly, medical images are vulnerable to these adversarial attacks[1]. Acknowledging this vulnerability emphasizes the importance of enhancing the model's resilience.

5.1 Project Goals

In this project, we aim to design a robust model for detecting and classifying Alzheimer disease using MRI brain images, while achieving these two main goals:

1. **Resilience against adversarial attacks** – The model will generalize and perform well under adversarial input.
2. **High classification performance** – The robustness will not affect the model's accuracy from legitimate input.

To achieve these goals, we followed these steps:

1. **Transfer Learning**: We used transfer learning to fine-tune and extract features from three well-known unsupervised pre-trained models to perform well on our specific task: **DINOv2**, **ResNet34**, and **EfficientNet-B0**.
2. **Adversarial Attacks Implementation**: We performed two adversarial attacks on each one of the models: Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). These attacks were found to be effective attacks on medical images[2].
3. **Adversarial Training**: To enhance model robustness, we trained these models with adversarial input, focusing particularly on the PGD attack[3]. We used a weighted cross-entropy loss that combined the standard loss with an adversarial component[4].
4. **Models Ensemble**: Finally, we combined these three models using a weighted approach to create a robust model, without affecting performance.

The project is implemented in Python using the PyTorch framework, which allows us to build and train the models efficiently throughout these steps.

5.2 Dataset

We used a pre-processed dataset[5] of 11,519 axial MRI brain images: 6,400 images from real patients and 5,119 synthetic images that were developed to rectify the class imbalance of the original dataset. The images are classified into four categories: "Non Demented", "Very Mild Demented", "Mild Demented", and "Moderate Demented". Each category had 100, 70, 28, and 2 patients, respectively, and each patient's brain was sliced into 32 horizontal axial MRIs. The images have a resolution of 128x128 pixels and are in the ".jpg" format. All images have been pre-processed to remove the skull.

The dataset was split according to the train-validation-test methodology: the train set contains 8,192 real and synthetic images, the validation set contains 2,048 real and synthetic images and the test set contains 1,279 real images only.

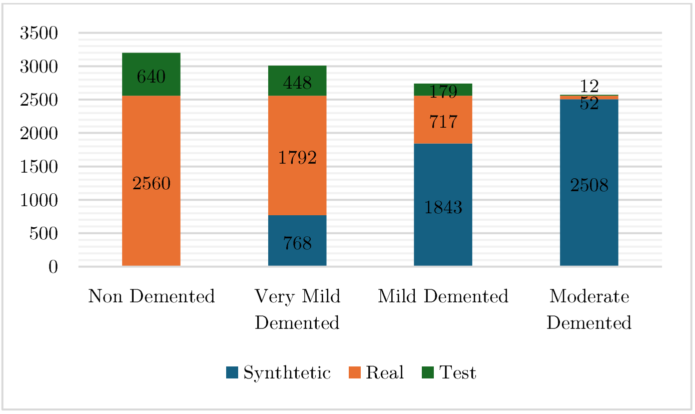

Figure 1: Dataset Classes

2 Adversarial Attacks

Adversarial attacks in machine learning exploit vulnerabilities by manipulating inputs to cause models to mistake. These attacks are divided into three categories:

1. **Poisoning attacks**: The attacker alters training data to introduce vulnerabilities.
2. **Model extraction**: The attacker replicates or steals model data.
3. **Evasion attacks**: The attacker modifies the input data during the test.

Our project focuses on evasion attacks, where adversarial examples are used to deceive models. There are two key types of attacks:

1. **White-box attacks**: The attacker has access to the model's internal workings.
2. **Black-box attacks**: The attacker knows only the model's outputs.

To implement attacks in our project, we used white-box untargeted attacks, where the goal is to make the model to misclassify the inputs. The model is represented as $f_\theta(\boldsymbol{x}) = \hat{\boldsymbol{y}}$. We aim to find a perturbation $\boldsymbol{\delta}$, such that $Class(\boldsymbol{x} + \boldsymbol{\delta}) \neq Class(\boldsymbol{x})$.

We utilized two types of attacks: Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). Both methods aim to generate adversarial examples by finding perturbations within a small L_∞ distance that maximize the loss, fooling the model while remaining imperceptible to human:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}' \in P_x}{\operatorname{argmax}} \, \ell(f(\boldsymbol{x}'), \boldsymbol{y}), \quad \text{where } P_x = \{\boldsymbol{x}' | \|\boldsymbol{x}' - \boldsymbol{x}\|_\infty < \epsilon\}$$

This means that we look for inputs $\boldsymbol{x}'$ that are differ from the original input $\boldsymbol{x}$ by no more than ε in the L_∞ norm. Since we want to maximize the loss, we perform gradient ascent instead of gradient descent.

Here are the two types of attacks we have used:

1. **Fast Gradient Sign Method (FGSM)**: A one-step attack where adversarial examples are generated by adding perturbations in the direction of the gradient:
$$\boldsymbol{x}^* = \boldsymbol{x} + \epsilon \cdot sign(\nabla \ell(f(\boldsymbol{x}^t), \boldsymbol{y})$$
where ϵ controls the attack's intensity.
2. **Projected Gradient Descent (PGD)**: An iterative version of FGSM, where the input is updated iteratively:
$$\boldsymbol{x}^{t+1} = \pi(\boldsymbol{x}^t + \alpha \cdot sign(\nabla \ell(f(\boldsymbol{x}^t), \boldsymbol{y}))), \quad \forall t \in [1, T-1]$$
where π is the minimization problem of finding the closest point in P_X to $\boldsymbol{z}$:
$$\pi(\boldsymbol{z}) = \underset{\boldsymbol{x}' \in P_x}{\text{argmin}} \|\boldsymbol{x}' - \boldsymbol{z}\|_2$$

3 Methods

5.1 Optuna

To enhance models' performance and convergence, we used the Optuna Python package[6] to optimize key hyper-parameters for the training. We used Optuna to find the following hyper-parameters:

- Learning rate: $10^{-6} - 10^{-2}$
- Optimizer: SGD, RMSprop, Adam
- Batch size: 16, 32, 64, 128, 256
- Scheduler: StepLR, CosineAnnealingLR, ReduceLROnPlateau

We used the pruning method provided by the Optuna package to early terminate unpromising trials in order to reduce the search time and save computational resources.

5.2 Data Augmentations

We chose to use data augmentation in our project in order to enhance models' accuracy and reduce overfitting. It acts as a form of regularization that prevent the model from relying too heavily on the specific train set. It involved applying several transformations on the data that expanded the size of the original dataset, which contains only 11.5K images. The augmentations we used are horizontal and vertical flips, rotations, affine transformations, and changes in brightness, contrast and sharpness. All these augmentations applied randomly on the input images using kornia transformations.

5.3 Mixed Precision

32-bit floating-point precision is the default in most deep learning frameworks and is known for its high numerical accuracy. However, it requires more memory and computational power. Mixed precision involves using multiple data types (or precisions) when training deep learning models. This approach aims to improve both the training speed and memory efficiency while maintaining similar levels of accuracy to full 32-bit training. We used a combination of 16-bit floating-point and 32-bit floating-point on the gradient's representation. It reduced dramatically the training time and memory usage, without affecting performance.

4 Architectures

We used three well-known unsupervised pre-trained models for classification tasks. We fine-tuned and/or used features extraction in order to get good classification accuracy.

4.1 DINOv2

DINOv2[7] (Distillation with No Labels v2) is an advanced self-supervised learning model developed by Facebook AI Research (FAIR). It uses a teacher-student distillation mechanism where the teacher network provides pseudo-labels to train the student network. DINOv2 is particularly well-suited for tasks of image classification, as it captures robust and generalizable features. We took a pre-trained

DINOv2 ViT-s/14 model and utilized its strong feature extraction capabilities to get high performance results.

4.2 ResNet34

ResNet34 is a deep convolutional neural network (CNN) architecture that is part of the ResNet (Residual Network) family, introduced by Kaiming He and colleagues in 2015. ResNet architectures use residual connections, which allow the network to bypass certain layers. This design mitigates the problem of vanishing gradients, enabling the training of very deep networks without performance degradation. This architecture is highly efficient for image classification tasks and has a good balance between depth and computational cost. We used ResNet34 pre-trained model that consists of 34 layers, including convolutional, batch normalization, ReLU activation, and fully connected (FC) layers.

4.3 EfficientNet-B0

EfficientNet-B0 is the foundational model in the EfficientNet family, which was introduced by Google AI in 2019 to achieve high performance in image classification with significantly fewer parameters and lower computational cost compared to previous architectures. EfficientNet-B0 has only 5.3 million parameters, which make it highly efficient while providing good performance.

5 Experiments and Results

5.1 Hyper-Parameters

Here are the chosen hyperparameters for each one of the models:

	Regular training			Adversarial training		
	DINOv2	ResNet34	EfficientNet-B0	DINOv2	ResNet34	EfficientNet-B0
Num of epochs	65	30	35	20	30	25
Learning rate	$6.92 \cdot 10^{-5}$	$1.33 \cdot 10^{-4}$	$6.24 \cdot 10^{-4}$	$6.92 \cdot 10^{-5}$	$1.33 \cdot 10^{-4}$	$6.24 \cdot 10^{-4}$
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Batch size	64	32	64	64	32	64
Scheduler	Cosine annealing	Cosine annealing	Cosine annealing	Cosine annealing	Cosine annealing	Cosine annealing
Attack Type				PGD $\epsilon = 0.005$ $\alpha = 0.01$	PGD $\epsilon = 0.2$ $\alpha = 0.0001$	PGD $\epsilon = 0.005$ $\alpha = 0.01$

Table 1: Chosen Hyper-Parameters

5.2 DINOv2 Results

We used 50% adversarial examples of PGD attack with $\epsilon = 0.005$ and $\alpha = 0.01$ for the adversarial training of the DINOv2 model. Here are the results:

	Regular training	**Adversarial training**
Regular test set	96.01%	92.03%
PGD: $\epsilon = 0.005$, $\alpha = 0.01$	1.49%	71.93%
FGSM: $\epsilon = 0.005$	3.44%	41.67%

Table 2: DINOv2 Accuracy Results

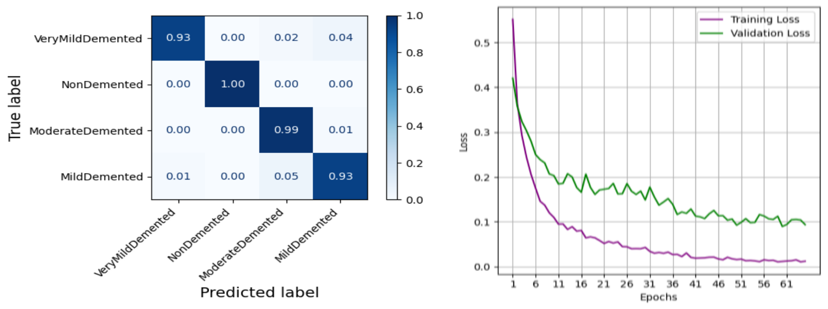

Figure 2: DINOv2 with regular training – confusion matrix (left) and loss curve (right)

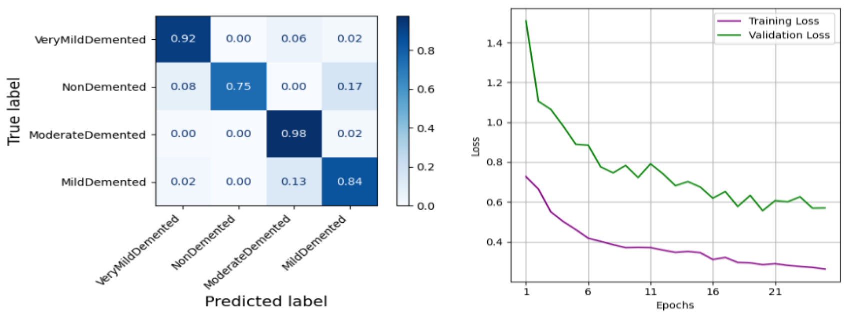

Figure 3: DINOv2 with adversarial training – confusion matrix (left) and loss curve (right)

5.3 ResNet34 Results

We used 50% adversarial examples of PGD attack with $\epsilon = 0.2$ and $\alpha = 0.0001$ for the adversarial training of the ResNet34 model. Here are the results:

	Regular training	**Adversarial training**
Regular test set	98.59%	93.35%
PGD: $\epsilon = 0.2$, $\alpha = 0.0001$	35.03%	97.65%
FGSM: $\epsilon = 0.2$	0.39%	0.16%

Table 3: ResNet34 Accuracy Results

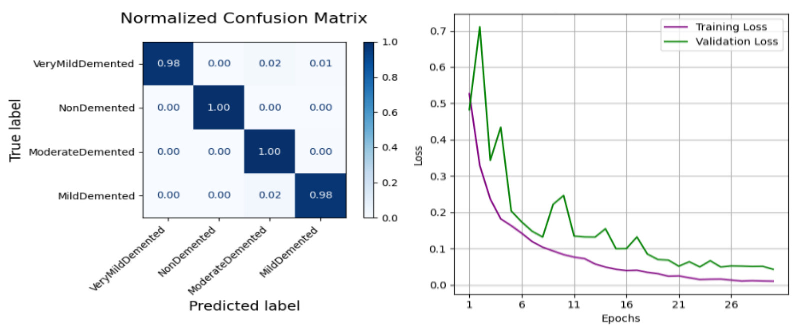

Figure 4: ResNet34 with regular training – confusion matrix (left) and loss curve (right)

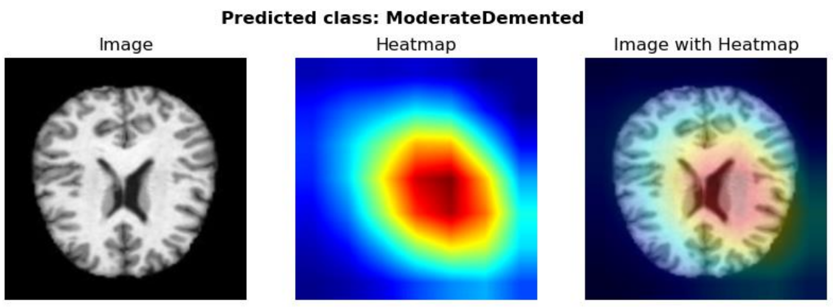

Figure 5: ResNet34 GradCAM example with regular training

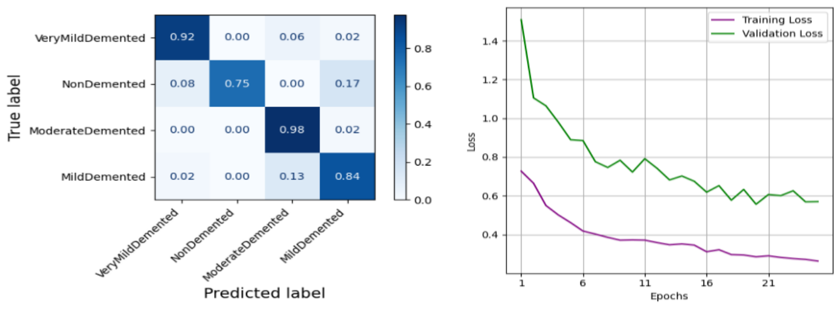

Figure 6: ResNet34 with adversarial training – confusion matrix (left) and loss curve (right)

5.4 EfficientNet-B0 Results

We used 50% adversarial examples of PGD attack with $\epsilon = 0.005$ and $\alpha = 0.01$ for the adversarial training of the EfficientNet-B0 model. Here are the results:

	Regular training	**Adversarial training**
Regular test set	98.91%	99.53%
PGD: $\epsilon = 0.005$, $\alpha = 0.01$	0%	98.83%
FGSM: $\epsilon = 0.005$	0.31%	8.13%

Table 4: EfficientNet-B0 Accuracy Results

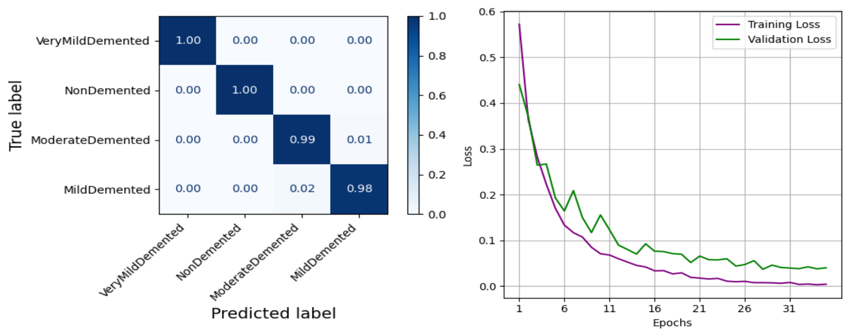

Figure 7: EfficientNet-B0 with regular training – confusion matrix (left) and loss curve (right)

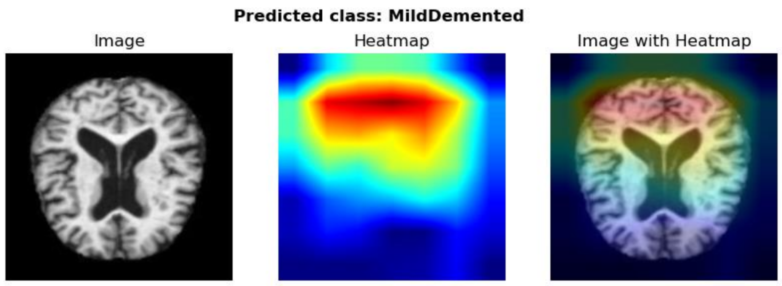

Figure 8: EfficientNet-B0 GradCAM example with regular training

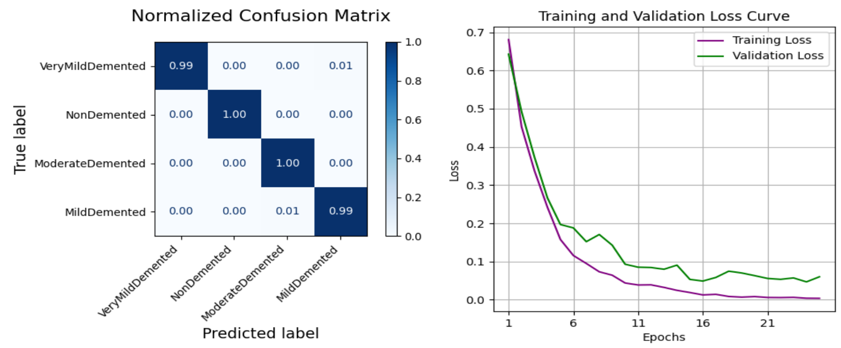

Figure 9: EfficientNet-B0 with adversarial training – confusion matrix (left) and loss curve (right)

5.5 Models Ensemble

After training each one of the models separately, we created a models ensemble in order to make our model robust enough to generalize well on attacks with different parameters than those we trained it. It also helped to improve the model's performance on legitimate input.

Attack Type	ϵ	α	**Accuracy**
Legitimate Input			98.51%
PGD	0.2	0.0001	68.80%
	0.1	0.0001	93.90%

	0.005	0.0001	92.18%
	0.0005	0.001	92.34%
FGSM	0.2		6.80%
	0.1		22.67%
	0.005		32.60%
	0.0005		92.81%

Table 5: Models Ensemble Accuracy Results

6 Conclusions

- **Impact of Adversarial Attacks:** Adversarial attacks significantly reduce the performance and accuracy of machine learning models by manipulating input data, leading to misclassification. Understanding these vulnerabilities is crucial for developing robust systems.
- **Advantages of Adversarial Training:** Adversarial training effectively improves model's accuracy against adversarial attacks by exposing it to adversarial examples. The Projected Gradient Descent (PGD) attack is particularly challenging, enhancing the model's resilience to sophisticated threats.
- **Limitations of Adversarial Training:** While adversarial training improves robustness, it does not fully resilient against all attack types or parameter variations. Models may still be vulnerable, and further research is needed for comprehensive defense strategies.
- **Resources Considerations:** Adversarial attacks require significant computational resources. Focusing on efficiency in the training and implementation of models can create robust solutions that are more practical and accessible.

7 Future Work

- **Investigation of Advanced Attacks and Black-Box Attacks**: Investigating various targeted attacks, including advanced versions of FGSM, such as Iterative-FGSM, Targeted I-FGSM, IND and OOD attacks, Kryptonite Attacks and one pixel attacks. These attacks can be performed in a targeted way to evaluate the model's vulnerabilities and/or in an untargeted way. Additionally, we can further explore black-box attacks, where the model itself is not available to the attacker.
- **Diverse Datasets**: A more diverse dataset featuring various MRI images across different demographics and orientations can be used. Currently, we used only T1 MRI. A multi-modal dataset, such as PET and CT scans, can also be used.
- **RNN Performance Evaluation**: Assessment of the performance of Recurrent Neural Networks (RNNs) to determine their contribution to the robustness of the ensemble.
- **Ensemble Method Exploration**: Various ensemble techniques, such as stacking and blending.
- **Real-World Testing**: Conducting real-world tests in collaboration with radiologists and people that are responsible for the reliability of medical data. It can verify the model's applicability and reliability in the real world.

8 Ethics Statement

1. Introduction:

Student names: Dor Lerman, Niv Bar-Tov

Project Title: MRI Brain Alzheimer Classification Under Adversarial Attacks

Project Description: This project involves design of a robust model for detecting and classifying Alzheimer disease using MRI brain images, simulating a radiologist's diagnostic ability. The aim is to maximize resilience against adversarial attacks while maintaining high classification performance.

2. Have a large language model (LLM) answer the following questions on your project:

2a. List 3 types of stakeholders that will be affected by the project:

1. **Patients**: Individuals whose MRI images are used or affected by the model's predictions.
2. **Healthcare Providers and Radiologists**: Medical professionals who rely on accurate and reliable models for diagnostic assistance.
3. **Medical AI Developers and Researchers**: Those involved in creating or implementing AI in healthcare settings who may benefit from the model's advancements.

2b. What will an explanation that is given to each stakeholder look like?

1. **Patients**: This model is designed to enhance the detection and classification of Alzheimer's disease, helping radiologists make accurate diagnoses and thus potentially improving patient outcomes. It includes safeguards against adversarial manipulations to ensure reliability and patient safety.
2. **Healthcare Providers and Radiologists**: Our model aims to assist healthcare providers by delivering consistent and accurate Alzheimer's classifications while resisting adversarial alterations. By simulating a radiologist's diagnostic process, it strives to be a trustworthy tool in medical settings, improving diagnostic efficiency.
3. **Medical AI Developers and Researchers**: This project focuses on building a resilient AI model that can withstand adversarial attacks while maintaining high classification performance. This approach aims to set new standards for model robustness in healthcare applications, contributing valuable insights for ongoing AI development in sensitive fields.

2c. Who is responsible for giving the explanation to each stakeholder?

The research team and project leads are responsible for ensuring transparent communication with medical professionals and patients by summarizing the model's intentions and protections. Healthcare organizations implementing the model should also play a role in explaining it to their patients, guided by developers. Additionally, collaborative efforts with regulatory bodies will be essential to inform broader stakeholders in the healthcare and AI communities.

3. Reflect on the AI output:

3a. What do you think needs to be added/changed in the Generative AI responses, to make the explanation more ethical?

- Eemphasizing patient privacy
- Ensuring patients know how their data is used
- Additional clarity on the model's limitations and accuracy

9 Sources and References

[1] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Understanding adversarial attacks on deep learning based medical image analysis systems.

[2] Zhang, H., Li, Y., & Chen, X. (2023). Adversarial Attack and Defense for Medical Image Analysis: Methods and Applications.

[3] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards Deep Learning Models Resistant to Adversarial Attacks.

[4] Cai, Q.-Z., Du, M., Liu, C., & Song, D. (2018). Curriculum Adversarial Training.

[5] Luke Chugh. (2021). Best Alzheimer MRI dataset. Kaggle dataset. https://www.kaggle.com/datasets/lukechugh/best-alzheimer-mri-dataset-99-accuracy.

[6] Optuna. (2023). Optuna: A hyperparameter optimization framework. GitHub repository. https://github.com/optuna/optuna.

[7] Facebook Research. (2023). DINOv2. GitHub repository. https://github.com/facebookresearch/dinov2.

[8] Chen, X., Zhang, H., & Li, Y. (2022). Exploring adversarial attacks and defenses in vision transformers trained with DINO.

[9] Xie, L., & Wang, Z. (2023). DINOv2: Learning robust visual features without supervision.

[10] Tajbakhsh, N., Shin, J., Gurudu, S. R., & Hurst, R. T. (2022). What makes transfer learning work for medical images: Feature reuse & other factors.

[11] Gil, J. (2020). PyTorch Grad-CAM. GitHub repository. https://github.com/jacobgil/pytorch-grad-cam.

[12] Hoki. (2020). Torchattack: PyTorch adversarial attack library. GitHub repository. https://github.com/Harry24k/torchattacks.