CS487  
Assignment 2 Report

**Part 1:**

**Task-1**

a.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Train Set** | **Validation Set** | **Test Set** |
| VGG-16 | 99.11% | 89.95% | 90.679% |

b. A graph of training and training accuracy

AI-generated content may be incorrect.

c. The **VGG-16** model achieves a **test accuracy** of **90.68%** while attaining a **train accuracy** of **99.11%** and a **validation accuracy** of **89.85%.** The minor discrepancy between training and validation accuracies indicates a mild degree of overfitting which could be mitigated if regularization or data augmentation was employed. The convergence of validation and test accuracies reflects appreciable generalization. VGG-16’s performance on the LFW dataset and test accuracy exceeding 90% indicates that this model can discriminate facial features within the variations presented in the dataset.

Task 2a.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Clean images** | **Adversarial images**  **𝜖=1/255** | **Adversarial images**  **𝜖=5/255** | **Adversarial images**  **𝜖=8/255** |
| **FGSM Attack** | 87% | 81% | 72% | 64% |
| **PGD Attack** | 87% | 75% | 56% | 43% |

b.

A graph of a graph with blue and orange dots

AI-generated content may be incorrect.

c. A screenshot of a computer

AI-generated content may be incorrect.

Task 3

**(a) Accuracy and Perturbation Results**

**Untargeted Attack:**

* **Predictions:** tensor([5, 8, 7, 7, 3, 0, 2, 1, 0, 2])
* **Accuracy after attack on 10 images:** 0.00%
* **Average L2 perturbation:** 0.0000

**Targeted Attack:**

* **Predictions:** tensor([0, 8, 6, 0, 7, 2, 3, 1, 7, 3])
* **Accuracy after targeted attack:** 10.00%
* **Average L2 perturbation:** 0.0000

**(b) Explanation of the Carlini & Wagner L2 Attack and Its Comparison**

The Carlini & Wagner (C&W) L2 attack is an optimization-based adversarial attack designed to generate minimal perturbations that successfully fool a model while remaining imperceptible to humans. The attack formulates the problem as an optimization task, balancing two objectives: minimizing the L2 norm of the perturbation and ensuring the model misclassifies the input. By introducing a variable transformation and solving through gradient-based optimizers (e.g., Adam), the attack iteratively refines the perturbations to achieve successful evasion.

In comparison to the Fast Gradient Sign Method (FGSM), which performs a one-step update using the gradient sign, the C&W attack is more precise but computationally expensive. Unlike FGSM’s direct gradient manipulation, C&W solves an optimization problem that yields smaller and less noticeable perturbations. Projected Gradient Descent (PGD) is an iterative version of FGSM that applies multiple small perturbations while projecting the result back into a valid perturbation space. Although PGD improves upon FGSM’s simplicity, C&W’s targeted optimization approach generally achieves stronger attacks with minimal distortions. C&W attacks are particularly effective at bypassing defenses that are resistant to FGSM and PGD attacks due to their carefully minimized perturbations.

**PART-2**

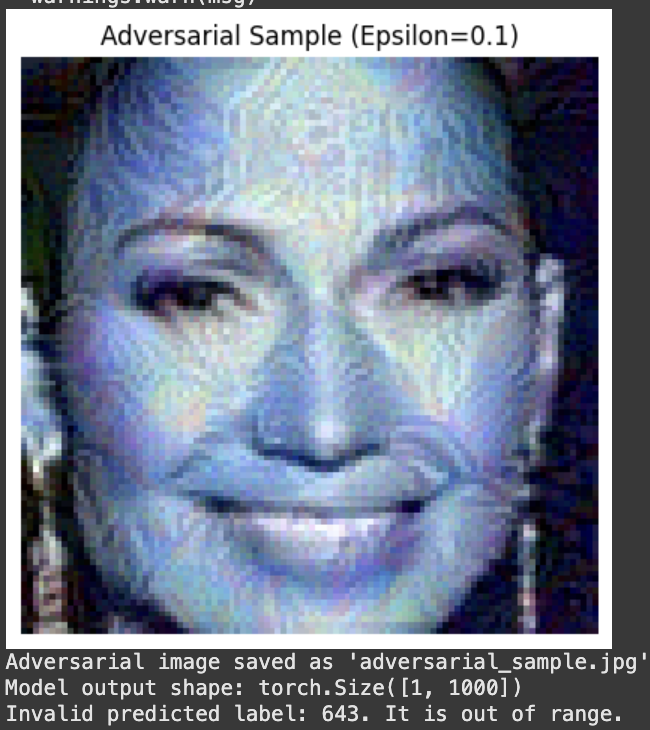
**Task1**

* 1. **Train Set Accuracy: 100.00%, Validation Set Accuracy: 84.33%, Test SetAccuracy: 84.84%**

A graph of training and validation accuracy

AI-generated content may be incorrect.

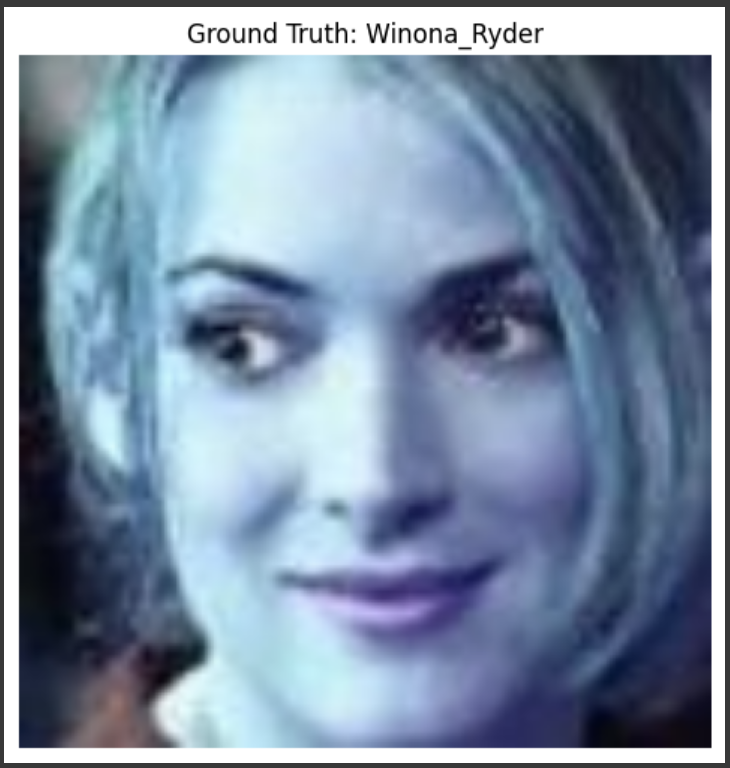
**Task-2**

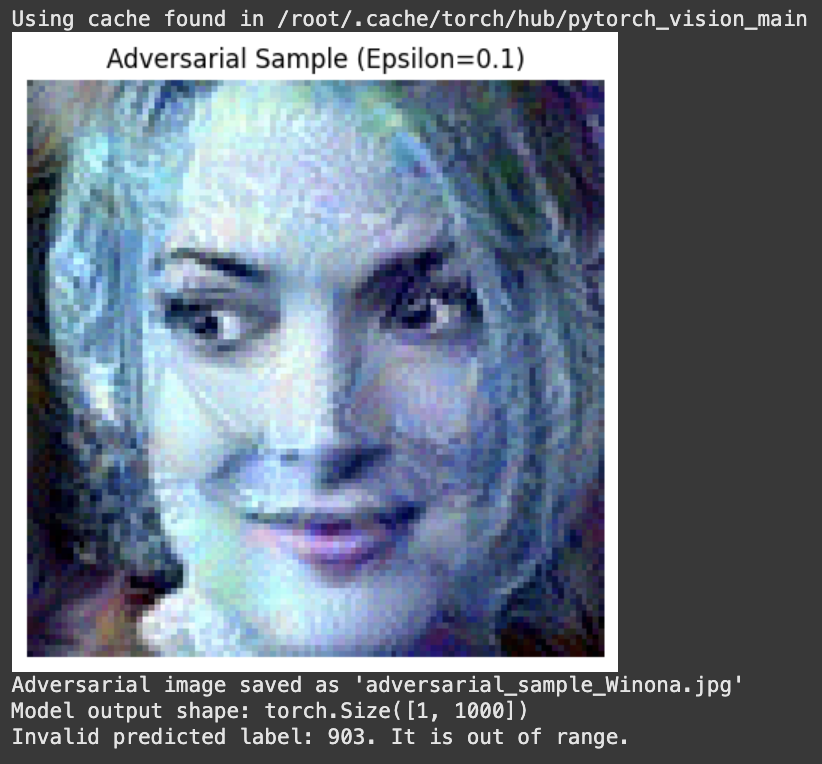
****

**A person with a blue face

AI-generated content may be incorrect.**

****

****

****

**Thoughts on the Target Model:**

The target model does a great job with clean, unaltered images, showing it’s learned well and can handle normal data effectively. However, it struggles when faced with tiny, carefully crafted changes (adversarial attacks). This means the model isn’t as robust as it could be—it’s like a student who aces practice tests but gets tripped up by unexpected questions.

**How Hard Was It to Attack?:**

Creating adversarial examples using the FGSM method was surprisingly straightforward. It’s like adding a small, invisible "nudge" to an image that tricks the model into making mistakes. The tricky part was figuring out the right amount of "nudge" (epsilon)—too little, and the model isn’t fooled; too much, and the changes become obvious. In real-world scenarios, where you don’t have full access to the model (black-box setting), this would be much harder to pull off.

**What Could Be Improved?:**

To make the model more resilient, it could be trained with adversarial examples (adversarial training) or use other techniques to defend against such attacks. It’s also important to remember that while testing for vulnerabilities is useful, it should be done responsibly to improve the model, not exploit it.

In short, the model is good but not bulletproof, and while attacking it was relatively easy in this case, there’s room to make it stronger and more reliable. Let me know if you’d like to dive deeper!