In the name of God

University of Tehran

Faculty of Electrical and Computer Engineering

Neural Networks and Deep Learning

Bonus Assignment

Author	Mohammad Taha Majlesi
Student ID	810101504
Submission Deadline	June 21, 2025

Contents

List of Figures

List of Tables

Assignment Rules

- **Report Format:** All answers must be submitted in a report using the provided 'REPORTS$_TEMPLATE.docx$'$formatontheElearnsystem.$

- **Group Work:** It is recommended to complete the assignments in groups of two. Groups larger than two are not permitted. Single-person submissions will not receive bonus points. Group members do not need to remain the same for all assignments.

- **Report Quality:** The quality of your report is crucial for grading. Please clearly document all assumptions, steps, and calculations in your implementations. Use captions for figures and tables as specified in the template.

- **Code Details:** You do not need to explain every line of code in the report. However, you must report and analyze the results obtained.

- **Result Analysis:** Analysis of results is mandatory, even if not explicitly asked for in a question.

- **Code Submission:** All code must be submitted in a Jupyter Notebook ('.ipynb') format. At the end of your work, ensure all cells are executed and their outputs are saved within the submitted notebook file. If a cell's output is a plot that is included in your report, that plot must also be visible in the notebook.

- **Academic Integrity:** Any instance of plagiarism will result in a score of -100 for all involved parties.

- **Programming Language:** The only permitted programming language is Python.

- **Use of Pre-written Code:** Using pre-existing code for the assignments is strictly forbidden. If two groups submit similar code from a shared source, it will be considered plagiarism.

- **Late Submission Policy:** Submissions are accepted up to one week after the deadline. A penalty is applied based on the delay:
  - Up to 3 days: No penalty.
  - Day 4: 5% penalty.
  - Day 5: 10% penalty.
  - Day 6: 15% penalty.
  - Day 7: 20% penalty.
  - After one week, the submission will receive a score of zero.

- **Submission Files:** Please place your report, code, and any other supplementary materials into a single folder, compress it, and name it according to the following format: 'HW[Number]$_[Lastname]_[StudentNumber]_[Lastname]_[StudentNumber].zip$'.

Part I

Analysis of Neural Network Performance Under Adversarial Attacks

1 Introduction to Robustness

Deep learning models, despite their impressive performance, can be remarkably fragile. The concept of **robustness** refers to a model's ability to maintain its performance in the face of small, often imperceptible, perturbations to its input. These perturbations can be random noise or, more critically, adversarially crafted to cause misclassification. This section explores the performance of ResNet and Vision Transformer (ViT) models under such conditions.

1.1 The Concept of Robustness

Robustness is a concept related to generalization, indicating how well a model withstands any small, adversarial or non-adversarial changes to its input. Improving robustness can sometimes lead to a decrease in performance on clean training data. This assignment explores methods to generate adversarial examples and defend against them.

An adversarial example X' is created from a clean input X by adding a small perturbation δ such that the model F misclassifies X', i.e., $F(X') \neq T$, where T is the true label. The goal is to find the smallest possible perturbation that causes this failure. This can be formulated as an optimization problem:

$$\min_{X'} d(X, X') \quad \text{s.t.} \quad F(X') \neq F(X)$$

The distance $d(X, X')$ is often measured using an L_p-norm, which defines the geometry of the "neighborhood" around the input.

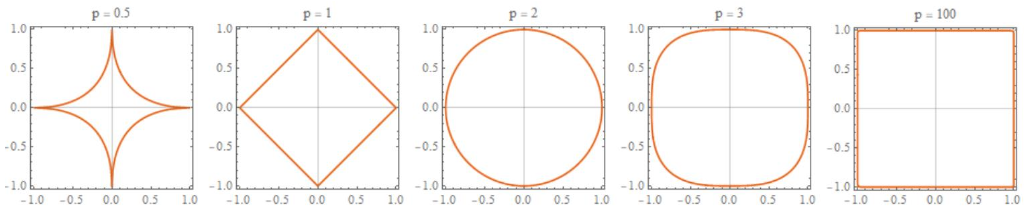


Figure 1: Geometric visualization of L_p-norm balls for $p \in \{0.5, 1, 2, 3, 100\}$. The L_∞ norm (approximated by $p = 100$) corresponds to a square/hypercube.

1.2 Methods for Generating Adversarial Examples

Two common methods for generating adversarial examples are:

- **Fast Gradient Sign Method (FGSM):** A single-step method that adds a perturbation in the direction of the sign of the loss function's gradient with respect to the input.

$$X' = X + \epsilon \cdot \text{sign}(\nabla_X J(\theta, X, y))$$

- **Projected Gradient Descent (PGD):** An iterative, more powerful version of FGSM. It takes multiple small steps in the gradient direction, projecting the result back into the ϵ-ball around the original image after each step to ensure the perturbation remains small.

$$X_{t+1} = \text{Proj}_{X,\epsilon}(X_t + \alpha \cdot \text{sign}(\nabla_X J(\theta, X_t, y)))$$

2 Training ResNet on Noisy Images

2.1 Data Preparation (CIFAR-100)

The CIFAR-100 dataset is a popular benchmark in computer vision. It consists of 60,000 32x32 color images in 100 classes, with 600 images per class. We investigate the effect of training a ResNet18 model on both clean and noisy versions of this dataset. Gaussian noise (mean=0, variance=0.05) is added to the training images to create the noisy version.

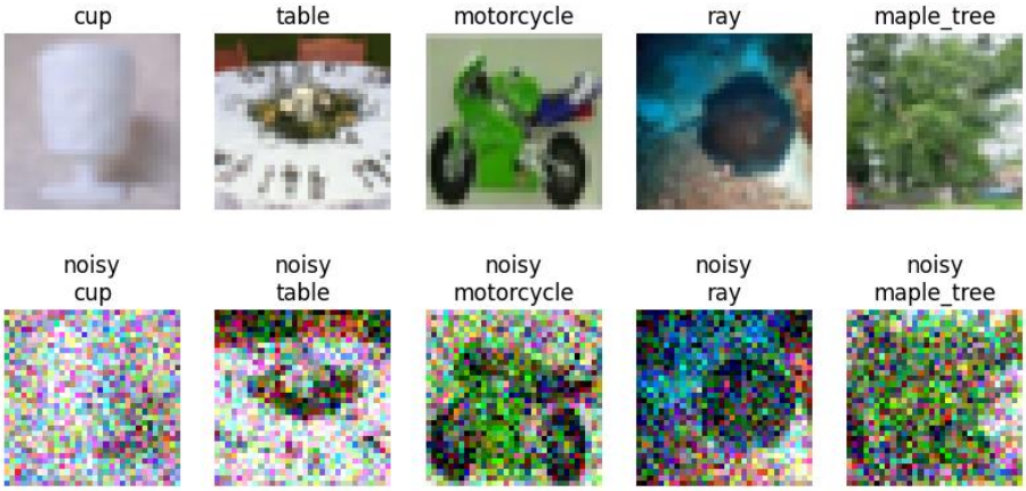


Figure 2: Examples of clean (top row) and noisy (bottom row) images from CIFAR-100.

2.2 Training and Results

2.2.1 ResNet18 on Clean Data

The ResNet18 model, with its 11.2 million parameters, is trained for 20 epochs on the clean CIFAR-100 data. We use a batch size of 64, the Adam optimizer, and a learning rate of 0.001. The training curves show good convergence, with the validation accuracy reaching approximately 41%.

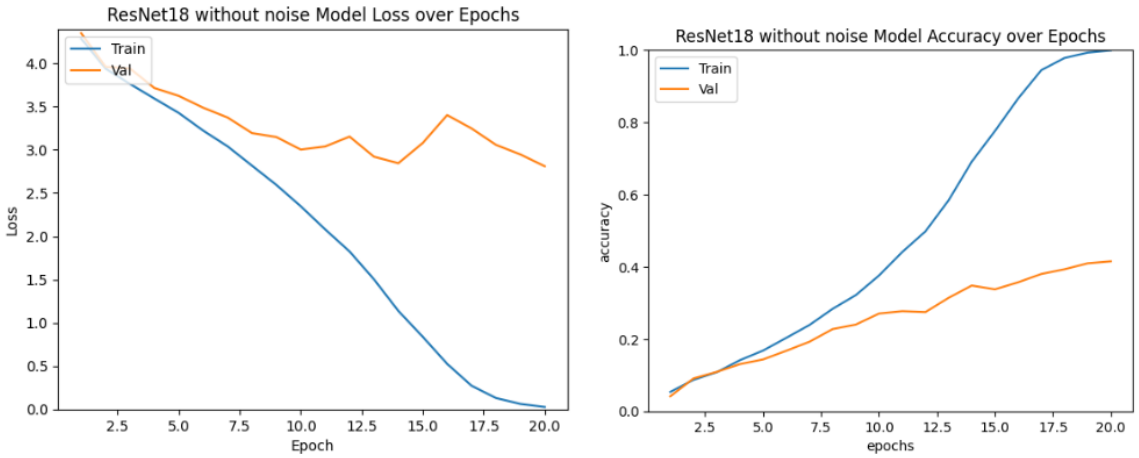


Figure 3: Accuracy (left) and Loss (right) for ResNet18 trained on clean CIFAR-100.

2.2.2 ResNet18 on Noisy Data

The same model is trained on the noisy dataset with identical hyperparameters. The performance is very similar, with the final validation accuracy also reaching around 41%.

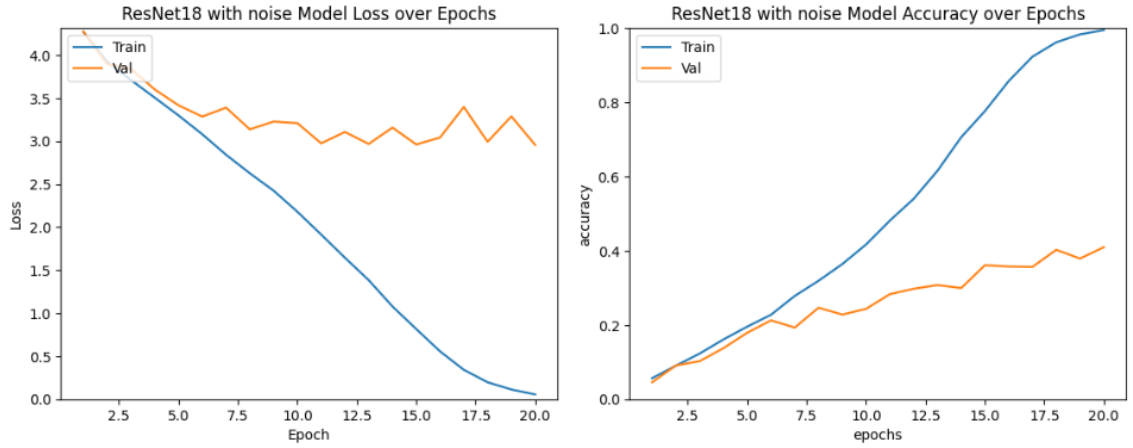


Figure 4: Accuracy (left) and Loss (right) for ResNet18 trained on noisy CIFAR-100.

Table 1: Performance comparison of ResNet18 trained with and without noise.

Metric	ResNet18 (Clean)	ResNet18 (Noisy)
Accuracy	0.4150	0.3960
Precision	0.4499	0.4389
Recall	0.4150	0.3960
F1 Score	0.4183	0.3971

Analysis: The performance of ResNet18 is remarkably stable, even when trained on noisy data. The final accuracy is only slightly lower. This suggests that the model

learns robust features that are not easily disrupted by random noise. The noisy images are difficult even for humans to classify, yet the model performs well, indicating it has learned stable underlying features.

3 Transfer Learning with ViT on Flowers-102

3.1 Data Preparation (Flowers-102)

The Flowers-102 dataset contains 8,189 images of 102 different flower species. It is a challenging fine-grained classification task.

Figure 5: Examples of images from the Flowers-102 dataset.

3.2 Training and Results

We compare two training strategies for a Vision Transformer (ViT-Base-Patch16-224) model:

1. **Pre-trained ViT:** A ViT model pre-trained on ImageNet is fine-tuned on Flowers-102 for 5 epochs.
2. **ViT from Scratch:** The same ViT architecture is trained from scratch on Flowers-102 for 10 epochs.

3.2.1 Pre-trained ViT (Fine-tuning)

The pre-trained model leverages features learned from ImageNet. It starts with a high accuracy and quickly converges, reaching a final validation accuracy of 80%.

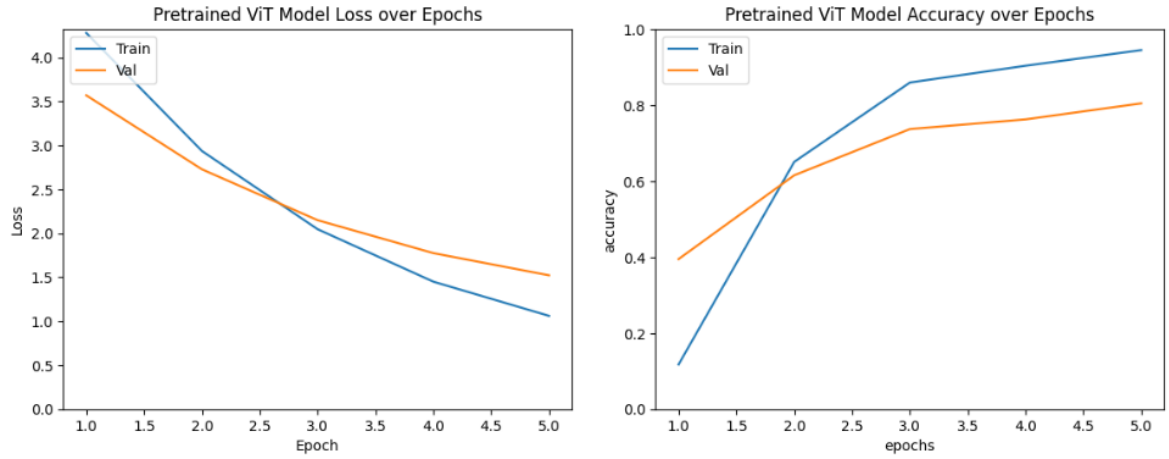


Figure 6: Accuracy (left) and Loss (right) for the pre-trained ViT model.

3.2.2 ViT Trained from Scratch

The model trained from scratch struggles significantly. This is expected, as Transformers are notoriously data-hungry and the Flowers-102 dataset is relatively small. It achieves a final validation accuracy of only 16%.

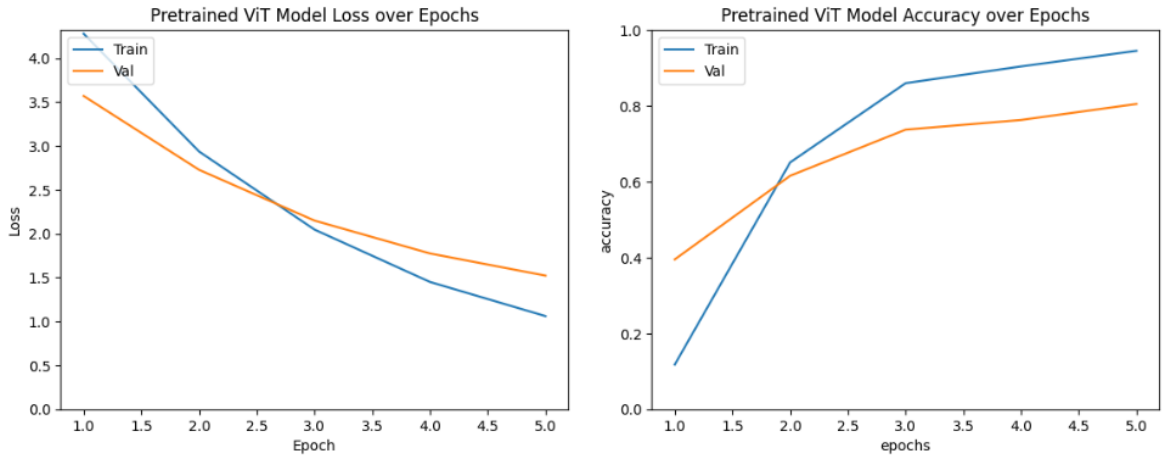


Figure 7: Accuracy (left) and Loss (right) for the ViT model trained from scratch.

Table 2: Performance comparison of ViT models.

Metric	ViT (From Scratch)	ViT (Pre-trained)
Accuracy	0.1676	0.8049
Precision	0.1241	0.8243
Recall	0.1676	0.8049
F1 Score	0.1242	0.7981

Analysis: Transfer learning is crucial for ViT models. The features learned on ImageNet

provide a powerful initialization, allowing the model to achieve high performance on the downstream task. Without pre-training, the ViT fails to learn effectively from the smaller dataset.

4 Adversarial Attacks and Defense

We now evaluate the robustness of our four trained models (ResNet18 clean/noisy, ViT scratch/pre-trained) against FGSM and PGD attacks and explore adversarial training as a defense mechanism.

4.1 FGSM Attack ($\epsilon = 0.1$)

4.1.1 ResNet Models

Both the clean-trained and noisy-trained ResNet18 models are highly vulnerable to FGSM attacks. Their accuracy drops to 0% on adversarial examples.

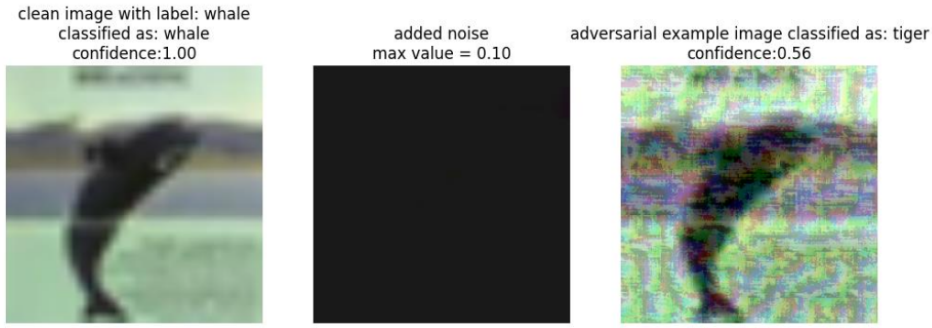


Figure 8: An adversarial example for the clean ResNet18. The model correctly classifies the clean image as a whale but misclassifies the perturbed image as a tiger.

After adversarial training, the model's robustness improves significantly.

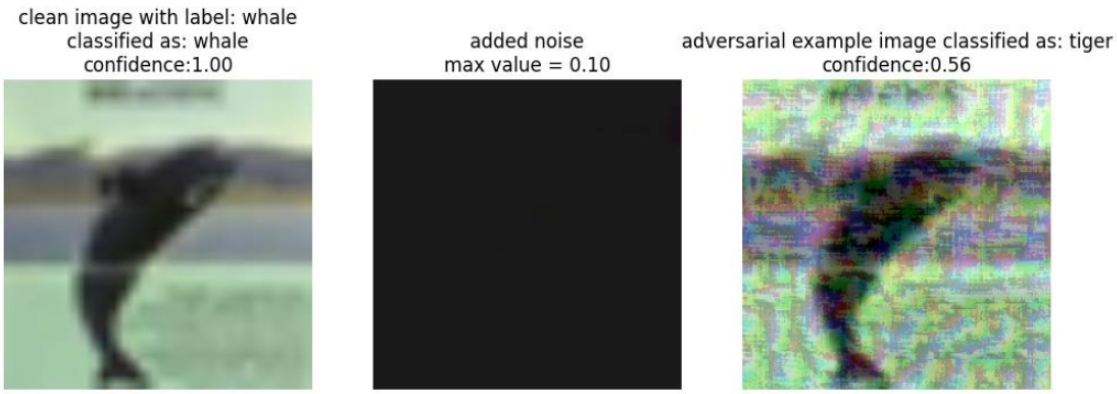


Figure 9: Training curves for adversarially trained ResNet18.

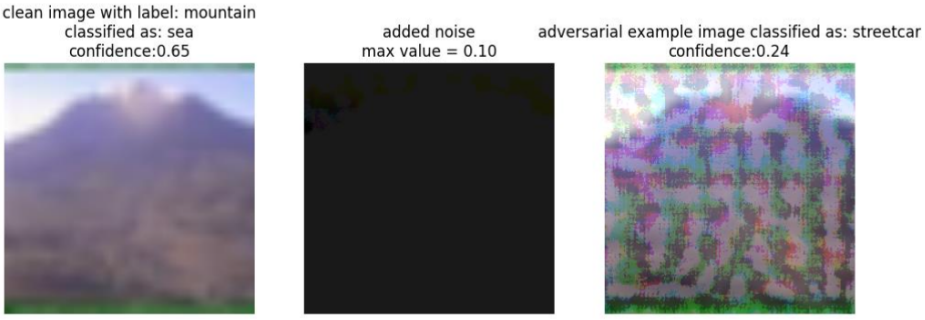


Figure 10: Adversarial example for the noisy-trained ResNet18.

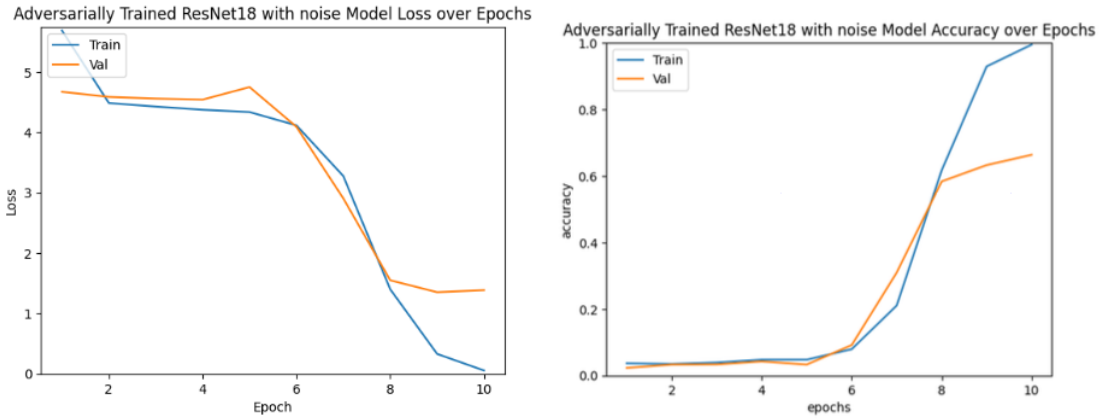


Figure 11: Training curves for adversarially trained noisy ResNet18.

Table 3: ResNet18 performance after adversarial training against FGSM.

Model	Clean Acc.	Adv. Acc.	Clean F1	Adv. F1
ResNet18	0.2820	0.5960	0.2850	0.6003
Noisy ResNet18	0.2840	0.6635	0.2755	0.6667

Analysis: Adversarial training is an effective defense. The model trained on noisy data achieves slightly better adversarial robustness, suggesting that training on random noise can improve robustness to structured adversarial noise. However, there is a trade-off: accuracy on clean images drops after adversarial training.

4.1.2 ViT Models

The ViT models are also completely vulnerable to FGSM attacks initially.

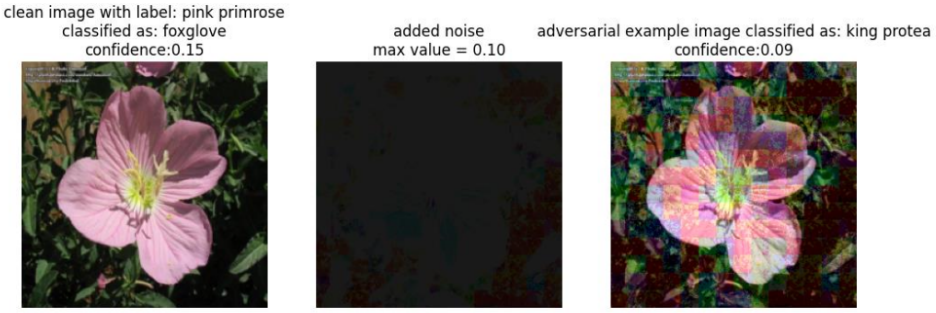


Figure 12: Adversarial example for the ViT trained from scratch.

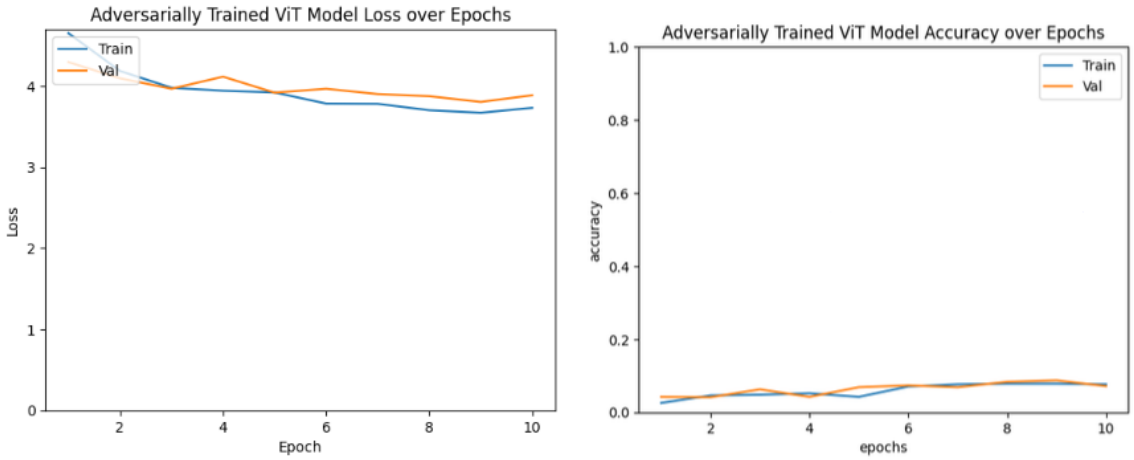


Figure 13: Training curves for adversarially trained ViT from scratch.

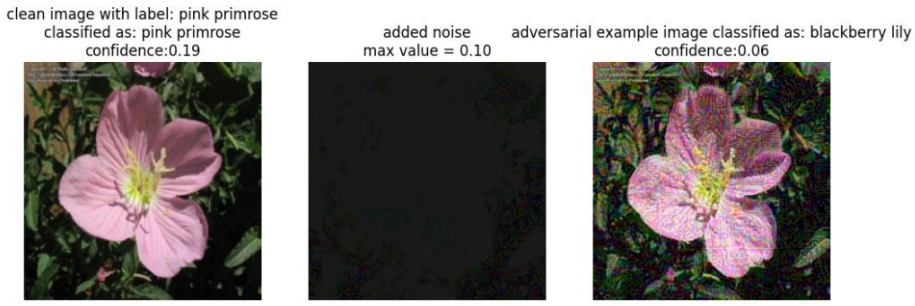


Figure 14: Adversarial example for the pre-trained ViT.

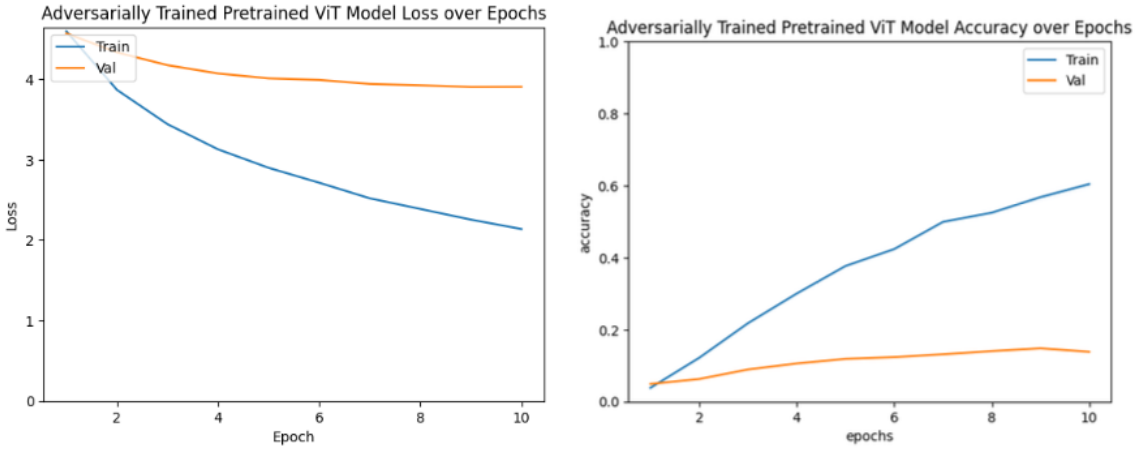


Figure 15: Training curves for adversarially trained pre-trained ViT.

Table 4: ViT performance after adversarial training against FGSM.

Model	Clean Acc.	Adv. Acc.	Clean F1	Adv. F1
ViT (Scratch)	0.1020	0.0716	0.0632	0.0469
Pre-trained ViT	0.6657	0.1382	0.6452	0.1174

Analysis: The pre-trained ViT shows a significant drop in clean accuracy after adversarial training (from 80% to 66%), but gains some robustness. The ViT trained from scratch fails to learn effectively even with adversarial training, highlighting its dependence on large-scale data.

4.2 PGD Attack ($\epsilon = 0.1, \alpha = 0.02, \textbf{steps} = 7$)

PGD is a stronger attack. All models show near-zero accuracy against PGD before adversarial training.

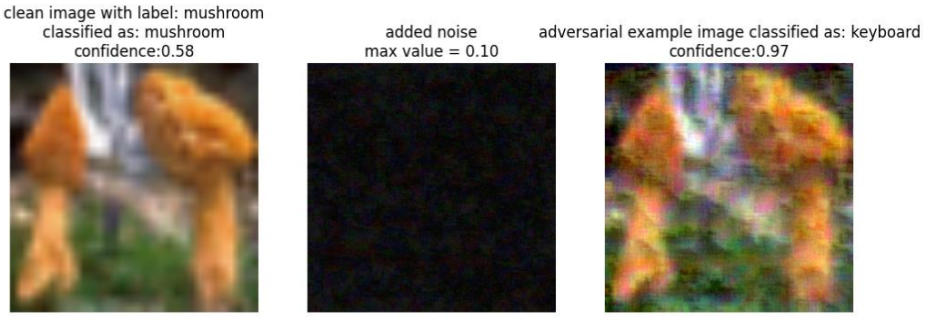


Figure 16: PGD adversarial example for ResNet18. The perturbation is less perceptible than FGSM's.

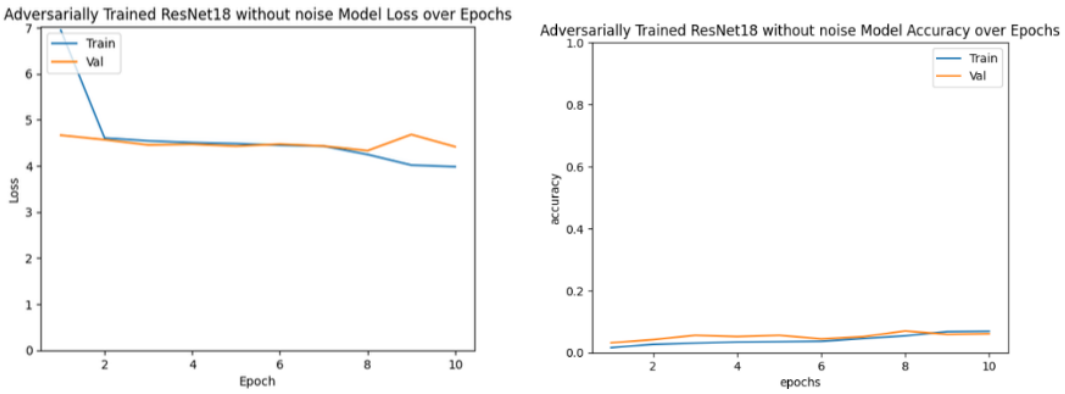


Figure 17: Training curves for ResNet18 under PGD adversarial training.

adv_fig27_resnet_noisy_pgd_example.png

Figure 18: PGD adversarial example for noisy ResNet18.

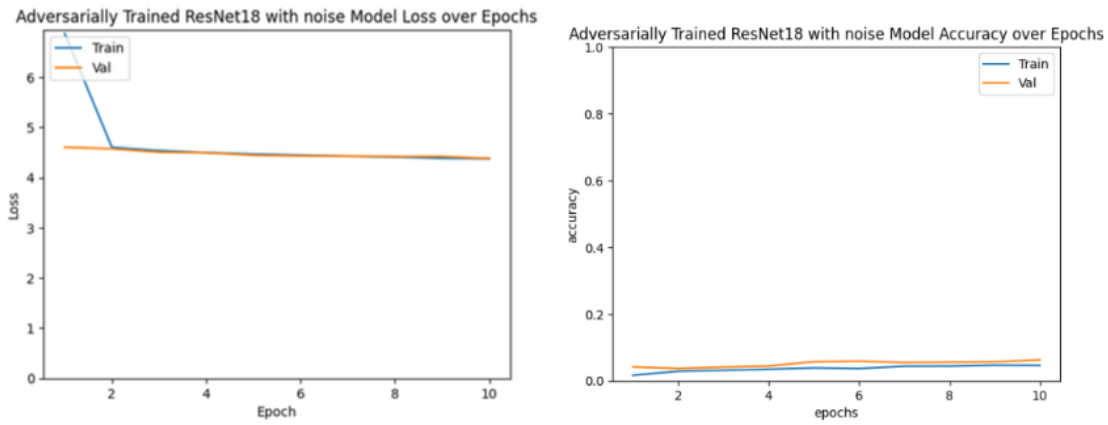


Figure 19: Training curves for noisy ResNet18 under PGD adversarial training.

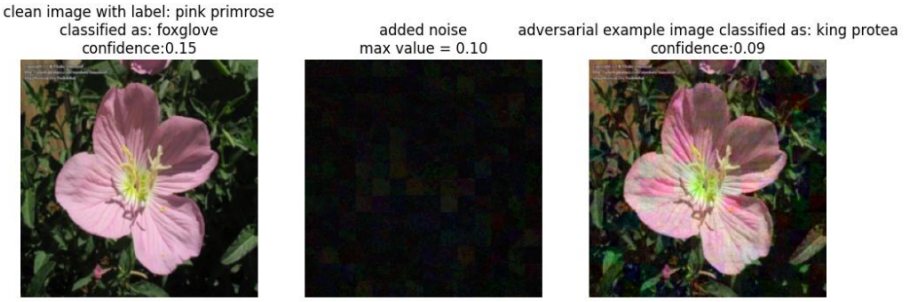


Figure 20: PGD adversarial example for ViT from scratch.

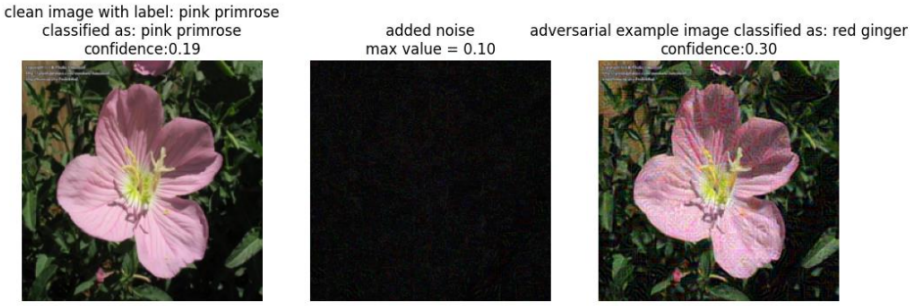


Figure 21: PGD adversarial example for pre-trained ViT.

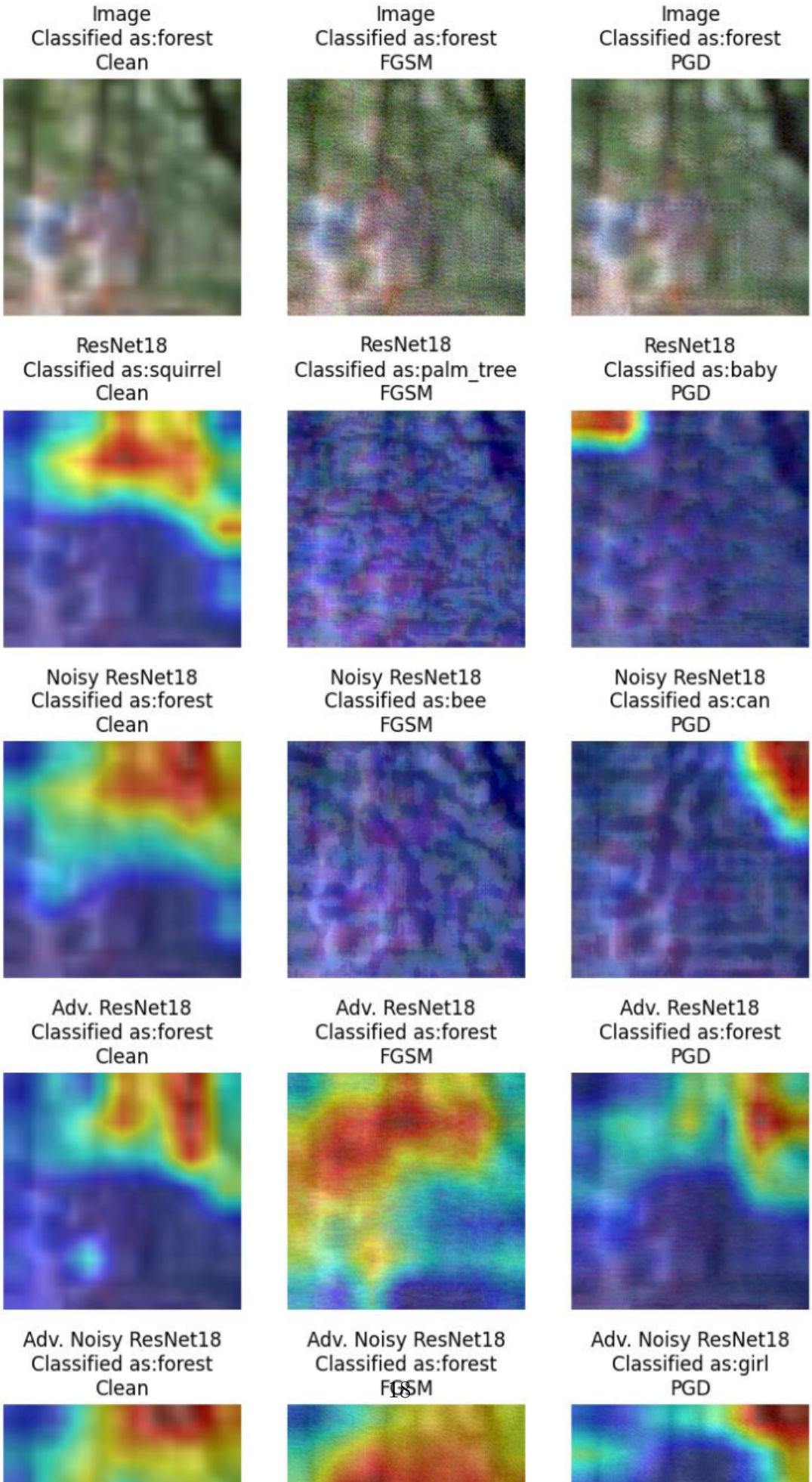
Analysis: Adversarial training against the stronger PGD attack is much harder. The models learn some robustness, but their performance is significantly lower than against FGSM. The ViT models, in particular, struggle to defend against PGD, with their accuracy remaining near zero.

5 Model Interpretability with Grad-CAM

Grad-CAM (Gradient-weighted Class Activation Mapping) is a technique to visualize where a model is "looking" when making a prediction. We use it to understand how

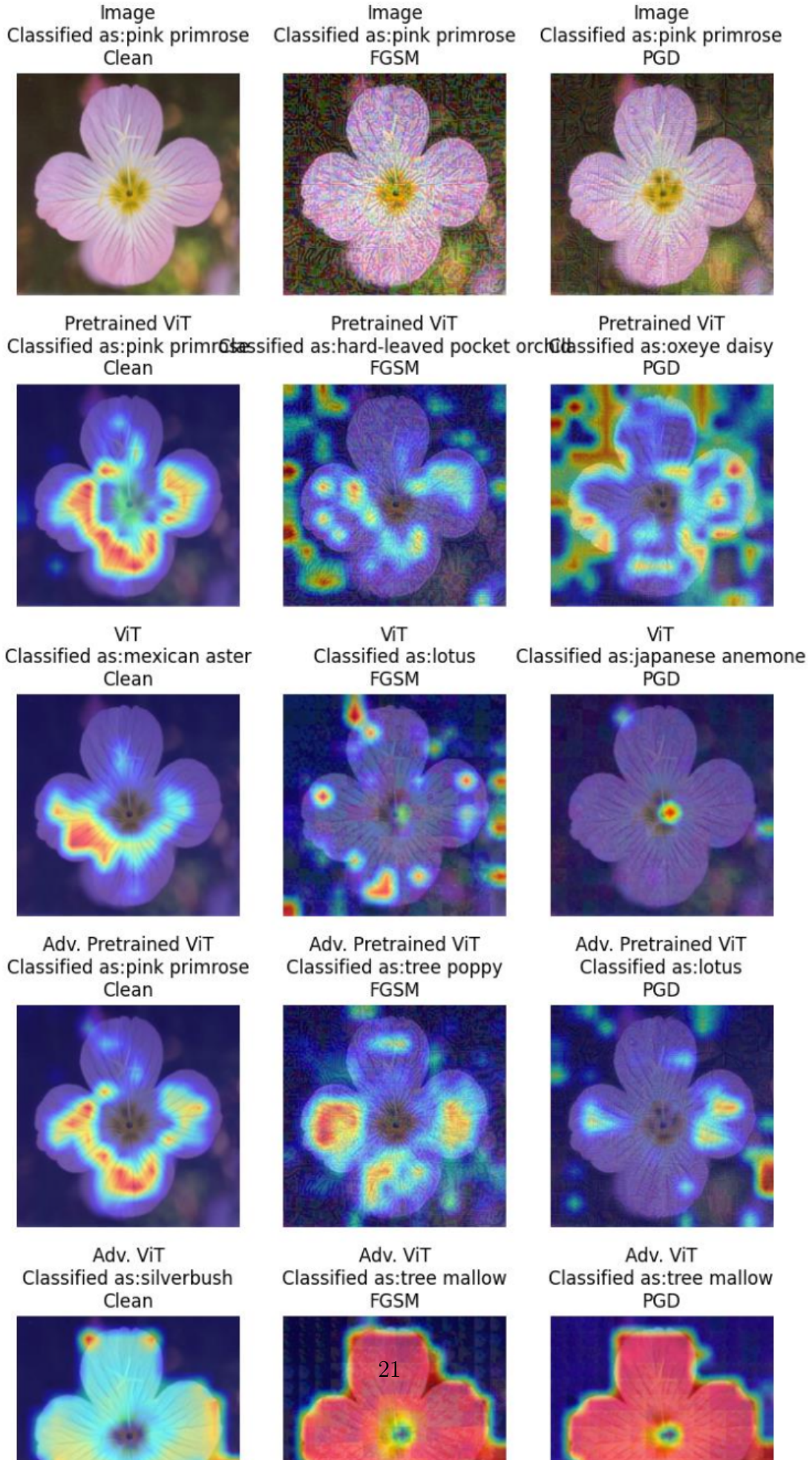
adversarial attacks and defenses affect the model's focus.

5.1 Grad-CAM on ResNet Models



Analysis: The standard models often lose focus under attack, with the attention map becoming diffuse or highlighting incorrect regions. The adversarially trained models, especially the final one (trained on noisy data and then adversarially), are better at maintaining focus on the correct object (the tree) even when misclassifying the adversarial image. This suggests adversarial training encourages the model to rely on more robust, object-centric features.

5.2 Grad-CAM on ViT Models



Analysis: The pre-trained ViT focuses correctly on the flower in the clean image. Under attack, its attention becomes scattered across the entire image, leading to misclassification. The adversarially trained models show a similar pattern: they are better at keeping their attention on the flower, even if the final prediction is wrong. This indicates that the defense mechanism works by forcing the model to maintain its focus on salient object parts.

6 Theoretical Questions

6.1 Robustness of ResNet to Noise

ResNet's strong performance on noisy data can be attributed to several factors:

- **Feature Hierarchy:** Deep CNNs learn a hierarchy of features. Early layers learn simple features (edges, textures), while later layers learn abstract concepts (objects). High-frequency random noise primarily affects the low-level features, but the high-level semantic features can remain intact.
- **Batch Normalization:** BN normalizes features within a batch, which can reduce the impact of noise by scaling it down along with the features.
- **Double Descent:** For highly overparameterized models, performance can improve even as model complexity increases past the point of interpolation. This phenomenon suggests that large models like ResNet have an implicit capacity to be robust to noise.

6.2 Transfer Learning and Sharp/Flat Minima

Pre-trained models tend to perform better because large-scale pre-training guides them to converge to **flat minima** in the loss landscape. Models trained from scratch on small datasets are more likely to fall into **sharp minima**.

- **Flat Minima:** In a flat region, small perturbations to the input (like the shift from training to test data, or noise) do not significantly change the loss, leading to good generalization.
- **Sharp Minima:** In a sharp valley, a small change in input can lead to a large jump in loss, resulting in poor generalization.

Pre-training on a massive, diverse dataset like ImageNet forces the model to find a solution that works for many different data distributions, which corresponds to a flat, generalizable minimum.

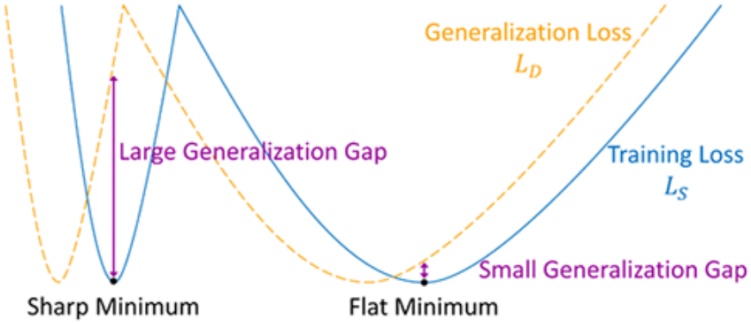


Figure 24: Illustration of sharp vs. flat minima and their impact on generalization.

6.3 Adversarial Training as Data Augmentation

Adversarial training can be seen as a form of data augmentation. However, unlike traditional augmentation (rotation, cropping), which expands the dataset with plausible natural variations, adversarial training augments the data with "worst-case" examples specifically designed to fool the model. This forces the model to learn features that are invariant not just to natural transformations but also to adversarial perturbations, leading to a more robust decision boundary.

6.4 Architectural Differences: ResNet vs. ViT

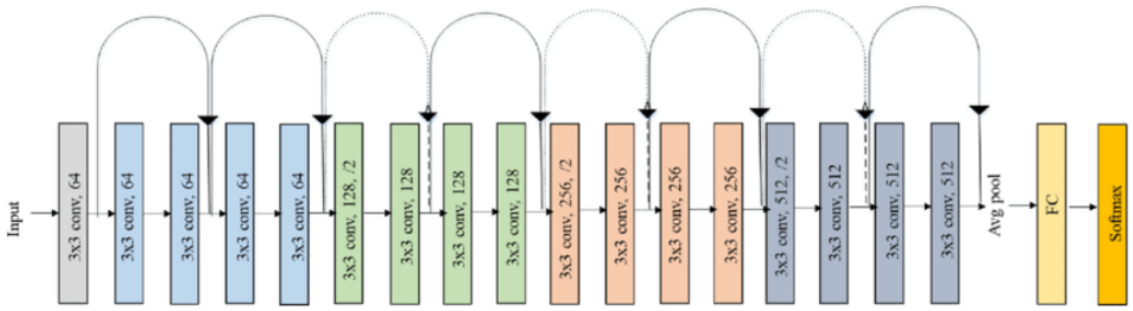


Figure 25: High-level structure of ResNet18.

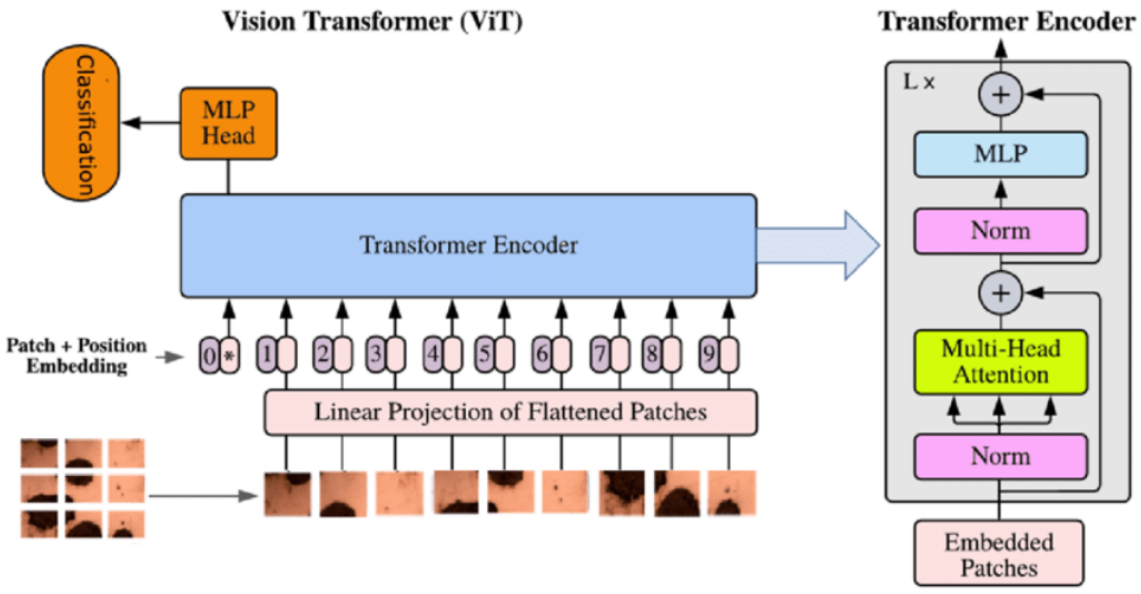


Figure 26: High-level structure of the Vision Transformer (ViT).

- **Inductive Bias:** ResNet has a strong **local inductive bias** due to its convolutional filters, which process local neighborhoods. This makes it data-efficient. ViT has a much weaker inductive bias; its self-attention mechanism considers the relationships between all pairs of image patches globally. This makes it more flexible but requires vast amounts of data to learn these relationships from scratch.
- **Robustness:** ViT's global attention mechanism can make it inherently more robust to small, local perturbations, as its decision relies on the entire image context. ResNet's reliance on local features can make it more vulnerable.
- **Ensemble Effect:** The multi-head attention in ViT acts as an ensemble. Each head can learn to focus on different relationships, and their combined output is more robust than a single model. This, combined with techniques like dropout, contributes to ViT's potential for greater robustness.

Part II

Generating Textual Descriptions for Images (Image Captioning)

7 Project Overview

This part outlines the methodology for an image captioning project, which aims to generate descriptive text sentences for given images. This task lies at the intersection of Computer Vision and Natural Language Processing and involves extracting visual features and translating them into coherent language.

8 Methodology

8.1 Data Preparation

The project uses a translated version of the COCO Captions dataset, containing 40,000 image-caption pairs.

1. **Data Acquisition:** Download the ‘coco-flickr-fa-40k.zip‘ dataset.
2. **Image Preprocessing:** Resize all images to a uniform dimension (e.g., 224x224) and normalize pixel values to the [0, 1] range.
3. **Caption Preprocessing:**
   - Normalize Persian text using a library like ‘hazm‘.
   - Remove punctuation, special symbols, and non-essential numbers.
   - Tokenize captions and build a vocabulary. Assign a unique integer to each word.
   - Include special tokens: ‘¡pad¿‘ for padding, ‘¡sos¿‘ for start-of-sequence, ‘¡eos¿‘ for end-of-sequence, and ‘¡unk¿‘ for unknown words.
4. **Data Splitting:** Split the dataset into appropriate training, validation, and test sets.

8.2 Model Architecture: CNN+RNN with Attention

The model consists of an encoder-decoder architecture with an attention mechanism.

8.2.1 Encoder

- Use a pre-trained CNN, such as EfficientNet-B7, as the feature extractor.
- Remove the final classification and pooling layers to obtain a feature map (a matrix of feature vectors), where each vector represents a specific region of the image.

8.2.2 Attention Mechanism ("Show, Attend and Tell")

Implement an Additive Attention (Soft Attention) mechanism:

1. At each step of the decoder, compare the current hidden state of the LSTM with all the image feature vectors from the encoder.
2. Calculate attention weights, which represent the importance of each image region for generating the next word.
3. Compute a context vector as the weighted average of the image feature vectors. This vector summarizes the visual information relevant to the current decoding step.

8.2.3 Decoder

- Use an LSTM to generate the caption word by word.
- Use an Embedding layer to convert input words into dense vectors. Pre-trained embeddings like fastText can be used for better performance.
- At each timestep, the LSTM receives the embedding of the previous word and the context vector from the attention mechanism.
- A final Linear layer with a Softmax function predicts the next word in the sequence.

8.3 Training and Evaluation

8.3.1 Training

- Use an appropriate loss function (e.g., Cross-Entropy Loss) and optimizer (e.g., Adam).
- Employ **teacher forcing** during training, where the ground-truth previous word is fed as input to the decoder at each step.
- Monitor training and validation loss, and visualize generated captions for a sample image after each epoch.

8.3.2 Evaluation

- Generate captions for the test set using **Greedy Search** and **Beam Search** (e.g., with a beam width of 3).
- Evaluate the quality of the generated captions using standard metrics like **BLEU-1** and **BLEU-4**.
- Compare the generated captions with the ground-truth captions for qualitative analysis.

8.4 Error Analysis and Model Improvement

8.4.1 Error Analysis

- Select 5 sample images where the model produces low-quality captions.

- For each generated word, visualize the attention weights as a heatmap on the original image.
- Analyze the heatmaps to determine the cause of errors: Is the model attending to incorrect regions? Is it choosing inappropriate words? Is the issue related to the dataset or the model architecture?

8.4.2 Advanced Techniques for Improvement

- **Scheduled Sampling:** Implement a curriculum learning strategy where the model transitions from using ground-truth words (teacher forcing) to using its own previously generated words as input during training. This can be done by decreasing the probability p of using the ground-truth word over the course of training.
- **Scaled Dot-Product Attention:** Replace the Additive Attention mechanism with the more efficient Scaled Dot-Product Attention from the Transformer architecture. In this mechanism, attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

  Here, the LSTM hidden state acts as the Query, and the image features act as the Keys and Values.

References

[1] Goodfellow, I. J., Shlens, J., Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

[2] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

[3] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[4] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE international conference on computer vision*.

[5] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *International conference on machine learning*.