Breaking Grad(ients) : CS 7643

Juan Raul de la Guardia jguardia7@gatech.edu
Pong-Ravee Halelamien win.halelamien@gatech.edu
Ethan Maluhia Roberts eroberts68@gatech.edu
Anna Zhu azhu95@gatech.edu

1. Introduction/Background/Motivation

In recent years, AI-generated deepfake images have become increasingly realistic, making it difficult to differentiate between artificially generated and real, human-generated images. Current computer vision models have also evolved to discriminate between the two reasonably well, but they are susceptible to adversarial examples [18, 13, 7] that can affect multiple target classes. Bad actors can meaningfully disrupt both classification models and humans [3], sometimes even with a single attack [5]. This problem is especially relevant in cases such as public trust, political misinformation, cybersecurity, and fraud prevention. We take a cybersecurity approach to discover adversarial attacks that disrupt common pre-trained models (red team), such as EfficientNet and Vision Transformers, and build training methods to effectively handle the attacks (blue team). Our objectives are: 1) to find models that can distinguish between real images and AI-Generated ones, 2) to find scalable attack mechanisms that significantly reduce the performance of these models when distinguishing images, and 3) to build and test defense mechanisms to protect against these attacks that also generalize well to other cases. We hypothesize that a targeted white-box attack will be able to efficiently impact classification performance more than a random perturbation in the test data. We also believe that a hybrid defense approach with data augmentation and adversarial training can best help protect against these attacks [22].

Current AI image detection classification models such as CNNs and Vision Transformers (ViT) can learn low level features and representations of images. These features can be used in several computer vision problems, such as image classification and object detection. However, Goodfellow et al [7] proved that an untargeted one-step perturbation to the gradient of the loss is enough to cause the model to incorrectly detect these features, thus causing it to output an incorrect answer with high confidence. This perturbation specifically is called the Fast Gradient Sign Method (FGSM). They argued that this vulnerability in the classification models is due to the piecewise linearities within the neural network structures. Because both CNNs and ViT models have these linearities, they are inherently susceptible to gradient-based attacks. Results show that FGSM is an effective white-box attack, surpassing the performance of data-based attacks like image distortion. We need to reliably distinguish between AI and human-generated images in order to establish authenticity and trust. This is difficult to perform accurately at scale, especially in areas like cybersecurity, medicine, and law where misclassification has high penalties. Businesses and individuals can be susceptible to financial and reputational damage. Misdiagnosing patients or responding incorrectly to public health crises can cause bodily harm or death. Errors in law, scientific research, and education have the power to affect entire populations and even nations in cases of political interference. Analyzing attack and defense mechanisms can help protect against bad actors, especially when they use methods like FGSM that are imperceptible to the human eye.

1.1. Data

This project is based on a Kaggle competition [9] called "Detect AI vs. Human-Generated Images" with a static, labeled dataset provided by the competition. Each real stock image sampled from Shutterstock has a paired AI-generated version created by DeepMedia using "state-of-the-art generative models", so real and fake class labels are balanced where 0 represents real images and 1 represents AI-generated images. The training dataset has a total of 79950 images of varying sizes where one-third of all images feature humans [14]. There is an unlabeled test dataset that was not used; instead, the original training dataset was split into train-test datasets. DeepMedia does not specify which image generation models were used but the method of feeding generated text to the model is zero-shot image generation, so there is some consistency with how the AI-generated images were created. Figure 1 shows examples from the original training dataset, along with generated text that was used to create its AI-generated equivalent. The AI-generated images are on the left with their corresponding real Shutterstock image on the right.

Figure 1: Example of the original Shutterstock image on the right, generated text of a description of the image, and the AI-generated image on the left [9]

2. Approach

Our approach consisted of implementing and testing three image classification models on the image classification problem: EfficientNet, Vision Transformer, and a custom CNN model. Each of these was tested against six configurations: (1) the baseline test dataset, (2) a series of random noise robustness test datasets, (3) an FGSM white-box attack [8], (4) an FGSM white-box attack with data augmentation, (5) an FGSM white-box attack with adversarial training [10], and (6) an FGSM white-box attack with hybrid data augmentation and adversarial training.

2.1. Preprocessing and Exploratory Data Analysis

For preprocessing, images were all resized to 224x224 and normalized to mean and standard deviation RGB values from ImageNet (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]). This helps to keep gradients and learning stable. In exploring the dataset, the images' RGB, HSV, and HSL values were averaged across all images and by channel and label. While the red, green, blue, hue, value, and lightness channels had similar average values with similar standard deviations, the average saturation was noticeably higher at a value of 137 in AI-generated images versus 90 in human-generated images. Plotting the channels as a histogram with values between 0 to 255 on the X-axis, it was found that AI-generated images had more pixels that were oversaturated (saturation value of 255) compared to human-generated images at 14% and 5%, respectively.

2.2. EfficientNet

To balance quality, computational efficiency, and model complexity, we chose EfficientNet-B0 because it was the most appropriate convolutional neural network (CNN) for the dataset given time and resource constraints for this project. Its key innovation is its compound coefficient that allows uniform scaling of its depth α^ϕ, width β^ϕ, and resolution γ^ϕ using a single parameter, making it easy for the model to adapt to data and constraints [16]:

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \qquad \text{where} \quad \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \qquad (1)$$

The compound coefficient controls resourcing for model scaling while α, β, γ allocate resources to each dimension. It addresses the challenge of balancing the dimensions with model architecture, since the efficacy of structures such as receptive field size are dependent on the dimensions of the input image. For example, larger, higher resolution images should have proportionally larger receptive fields available to capture both global and finer details.

2.3. Vision Transformer and Data Efficient Image Transfomer

Vision Transformer (ViT) is currently one of the best-performing image classification models [2]. It takes input images as a sequence of patches, vectorizes them into embeddings with their positional encodings, and feeds them into a transformer encoder. ViTs utilize longer ranged dependencies than CNNs using self-attention [19]. This allows for a more global understanding of an image, as compared to CNNs' more local focus on attributes like texture. Because ViTs lack strong invariance like CNNs, they require more training data. Data Efficient Image Transformers (DeiTs) are based on the same architecture as ViTs, but are trained on smaller datasets [7]. DeiTs use distillation to learn from CNNs or larger ViTs, producing lightweight models and faster training times. Since we are prioritizing smaller models, we used Facebook tiny distilled DeiT [17]. Typical ViTs use the standard cross-entropy loss. DeiTs use a weighted loss function that combines cross-entropy loss $Loss_{CE}$ and distillation loss $Loss_D$, specifically the KL divergence. This loss, originally outlined in the transformation from BERT to DistillBERT [15], allows the distilled student model to mimic the heavyweight model's loss function. The total loss function is a weighted sum of losses where and are weight parameters:

$$\text{Loss} = \alpha \cdot \text{Loss}_{CE} + \beta \cdot \text{Loss}_D \qquad (2)$$

2.4. JuanchitoCNN

JuanchitoCNN is a custom, lightweight 3-layer CNN. Each layer consists of a Conv-BatchNorm-ReLU block followed by a max-pooling layer. This block structure is partially inspired by the ResNet-50 residual blocks, and it was chosen because ResNET has been shown to be very efficient in distinguishing AI-Generated Images [20]. The final layer is a fully connected layer and two-category softmax with a standard cross-entropy loss function, which was chosen as a standard function for a two-class classification problem This model was built for speed and efficiency, and is the only model trained solely on this dataset, so it serves as a comparison to the other models that are both more complex and not specifically trained on this dataset. In testing, the model was surprisingly effective, so further layers were not deemed necessary for the purposes of this project. Figure 2 shows the CNN architectural design.

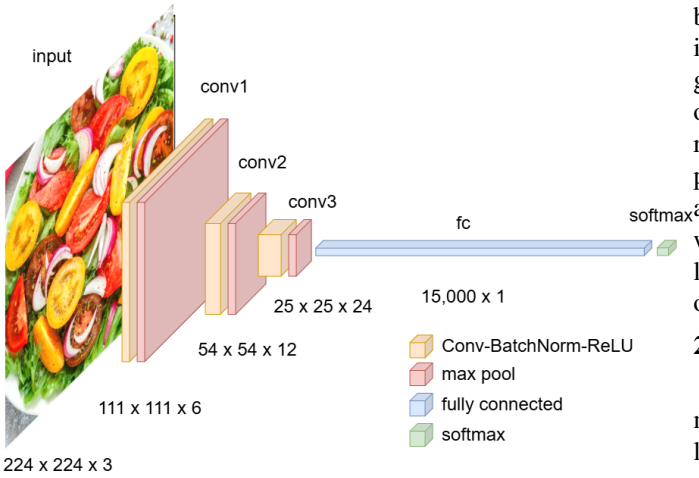


Figure 2: JuanchitoCNN Architectural Diagram

2.5. Model Comparison and Hypotheses

Each of the three models have different attributes at an input resolution of 224x224. We analyzed the number of parameters, Floating Point Operations per Second (FLOPS), and Top-1 accuracy to draw hypotheses for each. The number of parameters describes how large a model is, which is associated with a greater capacity to learn more complex patterns and relationships in the data. It also typically requires more memory and computation time. FLOPS is a measure of computing performance. For these models, they are a measure of how many computations are needed to run inference, where lower values mean fewer computations to produce a result at inference. Top-1 accuracy is the standard accuracy metric, where the class with the highest probability is assigned to the predicted label. Standard practice is to base top-1 accuracy on the ImageNet-1k dataset, with JuanchitoCNN trained and tested on a subset of this dataset. [4]. Table 1 shows the results.

Table 1: Models Comparison

Model	Parameters	FLOPS	Top-1 Accuracy
EfficientNet-B0	5.3M	0.39G	77.1%
DeiT-tiny distilled	6M	2.16G	74.5%
JuanchitoCNN	39k (38,642)	0.09G	2.0%

For EfficientNet and DeiT, we used default model weights and applied transfer learning. Then we fine-tuned it and performed hyperparameter tuning. JuanchitoCNN is not pre-trained, has the smallest number of parameters, and is the least complex, so we expect it to run the fastest but it should perform poorly and underfit this data in all scenarios. We believe EfficientNet will perform the best for both random perturbations on the test input and adversarial attacks because it encodes spatial hierarchies and more global semantic information, while the DeiT model relies on distilled knowledge from a CNN teacher that may not be relevant. Because DeiT has this weaker inductive bias compared to CNNs, we expect that the impact of an adversarial attack would be greatest on this model. For the same reason, we hypothesize that data augmentation should provide the least benefit to the DeiT model, since it is more dependent on quantity of data for performance.

2.6. Random Perturbation Input

To get a baseline for model robustness, we tested the models on random perturbations to the test data to simulate poor data quality or data corruption generated using the Albumentations library [1]. We expect that models least robust to random perturbations would benefit most from data augmentation. Whether there is a correlation between this robustness and adversarial attack and training is unknown. This seems to be a new approach so we will analyze it for findings.

2.7. Adversarial Attack

Our approach for attack is the Fast Gradient Sign Method (FGSM) method for all models since the same adversarial examples tend to fool more than one model [18], so it should generalize well. This is a well-studied, simple, and fast white-box method. It is also considered a targeted attack, since we have access to labels. FGSM is only implemented at inference for attack. FGSM perturbs an input image in the direction that would most increase the model's loss. Formally,

$$x_{\text{adversarial}} = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y)) \tag{3}$$

where θ is the parameters of the model, x is the input to the model, y is the labels, ϵ is a tunable parameter, and $J(\theta, x, y)$ is the loss function. The main finding of this approach is that an imperceptibly small perturbation, whose elements share the same sign as the gradient of the loss function to the pixels, can change the classification of the model with high confidence, as shown in Figure 3

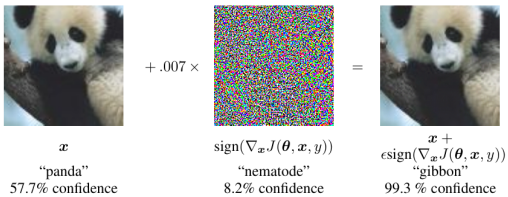


Figure 3: A demonstration of FGSM from Goodfellow et al [7] applied to GoogLeNet on ImageNet

2.8. Defense

2.8.1. Data Augmentation

A good general strategy to improve generalization is data augmentation. Using flips, random image brightness, and Gaussian noise to augment the training samples based on Yang et al [21] and Zantedeschi et al [22] should help the models improve their performance.

2.8.2. Adversarial Training

A more targeted, gradient-based defensive strategy is adversarial training, where we use the same approach that we implemented in the attack step to disrupt images and then perform training with the correct labels. The objective is for the model to learn how the images are perturbed using FGSM attack and then successfully defend against it.

2.8.3. Hybrid Approach

While some previous research shows that increasing the capacity of the network alone is helpful in adversarial robustness against one-step perturbations like FGSM [11], we chose a hybrid training method in which we first performed data augmentation in our dataset in addition to our adversarial training with FGSM. We expect this method to be the most robust, as the model should be learning complicated images and their labels [22].

2.9. Challenges

2.9.1. Visual Quality, Style, and Hue/Saturation

Upon visual inspection of the dataset, the AI-generated images tended to be more oversaturated compared to human-generated images. The AI-generated images also used a cartoonish style, which meant that it was very easy for us to distinguish between real and AI-generated images. Because of this, we suspected that the original objective from the Kaggle competition to build a model that correctly identified real vs AI-generated images was going to have issues with overperformance.

2.9.2. Pre-trained Models Performing Too Well

Our initial objective was to find a model that could effectively distinguish between real images and AI-Generated Images. We predicted that large models would be relatively effective for this problem. Our predictions were correct; as the experimental results show from Trial 1, all models performed above 90% accuracy for both the pre-trained, fine-tuned models and the model trained on only the training data, which was highly above our expectations. Optimizing this would be trivial, so we pivoted to building a comprehensive adversarial attack and defense experiment on these models to explore the nature of deep learning model attacks and build better protections against them.

3. Experiments and Results

We compared models using accuracy only, since this is the standard metric for classification models. We expected our base trial (trial 1) to have good accuracy, while the attack trials should have lower accuracy (trials 3-6). For the attack methods, we considered them very effective if they lowered accuracy to 50% or less (no better than a coin flip) or if they represented a significant reduction in performance. For the defense methods (trials 4-6), we considered them effective if they represented improved performance against our base FGSM attack test case (trial 3). See full results in table 3 and code [2].

3.1. Trial 1: Base Model and Data

We started with default EfficientNet weights that were trained on ImageNet-1K and fine-tuned it with 80% of the labeled data, then tested with the remaining 20%. Each epoch took approximately 5 minutes to train on an NVIDIA 2070 Super. The accuracy of the test data was 99.74%. A grid search was performed with different size magnitudes for learning rate, weight decay, and momentum and both SGD and AdamW optimizers. The epochs were left at 5 to avoid overfitting, which was seen at 10 epochs, and batch size was set to 16 due to computational resources.

For ViT/DeiT, we started with the Google ViT-base-patch16-224 model [2] and even though it performed at 99.87% accuracy, it was too large for efficient training at 9 hours per epoch. We changed to the Facebook DeiT-tiny-distilled-patch16-224 [17, 12] model which trained at 15 min per epoch. This model was also fine-tuned with an 80/20 train-test split. After tuning different hyperparameters, our best performance was 92.65% accuracy.

Our custom CNN model, JuanchitoCNN, had a performance above 90% on our initial run. After hyperparameter tuning, and adjusting optimizer choices, we were able to achieve an accuracy of 92.80%. This was considered very successful for a lightweight model, so the model was chosen as a baseline.

3.2. Trial 2: Random Perturbation Input

We perturbed the test data with methods from the Albumentations Vision Tool [1] that would not semantically alter the images. We tested four perturbations: SaltPepper, which adds single white and black pixels throughout the image; Posterize, which reduces the number of bits for each color channel; GaussianNoise, which adds Gaussian noise to the image; and RandomShadow, which simulates shadows for the image by reducing the brightness in random areas of the image. The results of these experiments are in table 5. EfficientNet did not seem to be affected by GaussianNoise or Random Shadow, but was affected by Posterize and SaltPepper at 92.03% and 77.62%, respectively. JuanchitoCNN was significantly affected by the perturbations,

Table 2: Model Hyperparameters

Model	Epochs	Learning Rate	Batch Size	Momentum	Weight Decay	Optimizer
EfficientNet-B0	5	0.1	16	0	0.1	SGD
DeiT-tiny distilled	5	0.0002	32	–	0.01	AdamW
JuanchitoCNN	5	0.001	32	0.001	–	AdamW

Table 3: Model Accuracy Results at $\epsilon = 0.1$

Model	(1) Base Model	(2) Random Perturbation	(3) FGSM Attack	(4) Data Augmentation	(5) Adversarial Training	(6) Hybrid
EfficientNet-B0	99.74%	99.42%	13.39%	46.74%	44.47%	46.52%
DeiT-tiny distilled	92.65%	92.25%	41.86%	35.97%	50.09%	44.85%
JuanchitoCNN	92.80%	80.20%	41.72%	0.20%	22.68%	39.73%

likely because of its small capacity. DeiT was more robust to all perturbations with accuracies around 91% likely because of the self-attention architecture between patches. Overall, the random perturbations performed as expected on lowering accuracies unevenly across different models, with no model significantly affected.

3.3. Trial 3: White Box FGSM Attack

This trial included an FGSM attack at $\epsilon = 0.1$ into the pretrained models' gradients. We tuned based on research, visual quality, and experimental results. Our attacks were extremely effective, decreasing EfficientNet accuracy to 13.39% (86.35% decrease), DeiT to 41.86% (54.82% decrease), and JuanchitoCNN to 41.72% (51.08% decrease). Training and validation loss curves for all models were stable, showing that FGSM attack can be undetectable and works well to attack models with stealth. Contrary to our hypothesis, EfficientNet was the most affected by the FGSM attack. JuanchitoCNN was significantly affected, likely because the model is very small, so it is unable to encode enough features to defend against the attack. CNNs have been shown to be biased towards learning from textures [6], so this can be a factor in the reduced accuracy. These results have no correlation with the random perturbation input trial but successfully outperformed the random perturbation inputs, showing that targeted gradient-based attacks are more effective at disrupting these classification models.

3.4. Trial 4: FGSM Defense - Data Augmentation

Maintaining the same FGSM attack as before, this trial features data augmentation to enhance the robustness of the model. The data augmentation techniques used were horizontal flip with 50% chance and random brightness and contrast adjustments with 20% chance. All augmented images were then normalized to the same RGB mean and standard deviation values as ImageNet. We kept the same number of training samples, so images were augmented without replacement and the model was not trained on the unaugmented versions of the augmented images. Notably, the validation loss for JuanchitoCNN had a significant peak at epoch 3, implying unstable inference. The EfficientNet model accuracy was 46.74%, which is a 33.35% improvement over Trial 3 with only FGSM attack. This shows that data augmentation was successful in providing some defense to the FGSM attack. The DeiT model accuracy was 35.97%, which is a 5.89% decrease over Trial 3 and shows consistently mild negative reactions to data augmentations and perturbations like in Trial 2. Our hypothesis was that the weak inductive bias in DeiT could make it slightly negatively sensitive to data-related changes, but we did not expect it to perform worse than the non-augmented model. JuanchitoCNN performed worse with augmented data than with regular data when attacked, going down to 0.20% accuracy which is a 41.52% decrease. The random augmentations were not effective in improving the model performance, perhaps because the changes were not captured correctly by such a small model.

3.5. Trial 5: FGSM Defense - Adversarial Training

In Trial 5, we added FGSM adversarial training with $\epsilon = 0.1$ as a defense to FGSM adversarial attack in the hopes that training a model on the same gradient-based attack would make it robust to the attack. Loss curves showed high validation loss compared to training loss for all three models, suggesting that none of the models are generalizing particularly well. The FGSM defense with adversarial training for EfficientNet had a slightly lower accuracy of 44.47%

compared to the Trial 4 FGSM defense with augmented data which had an accuracy of 46.74%. It represents a 31.08% improvement over Trial 3 with FGSM attack only. Both data-based and gradient-based defenses seem effective on EfficientNet. DeiT performed the best of the three models with an accuracy of 50.09%, which is a 8.23% improvement over Trial 3. Adversarial training provides a mild improvement to DeiT which is much better than the data augmentation in Trial 4. This suggests that its architecture is more receptive to gradient-based training than data-based. For JuanchitoCNN, adversarial training was significantly more effective than using augmented training data, with an accuracy of 22.68%. It was still worse than Trial 3 FGSM attack with no defense that had an accuracy of 41.72%, which leads us to believe that this defense is not helping the model. We believe that the images are too complex for the simple model to effectively learn how adversarial attacks work, so it is unable to defend against them.

3.6. Trial 6: FGSM Defense - Hybrid

We implemented a hybrid defense with both data augmentation and adversarial training, which was effectively a combination of Trials 4 and 5. EfficientNet performed nearly identically to the purely augmented trained model at 46.52%, a 0.18% difference. However, its validation loss curve had a very high peak at epoch 3 an order of magnitude larger than expected, so its inference is not as stable as the Trial 4 augmented model. DeiT model had an accuracy of 44.85%, which is a 2.99% improvement over the Trial 3 attack-only model but it did not perform as well as the model trained only with adversarial data. Its validation loss curve is also unstable, so we hypothesize that data augmentation and adversarial training have conflicting objectives with respect to the DeiT architecture. For JuanchitoCNN, hybrid training was surprisingly more effective than each independent defense method, with an accuracy of 39.73%. This is higher than the combined accuracies from Trials 4 and 5 with 0.20% and 22.68% respectively. This could mean that both methods combined have a multiplicative effect on the model's ability to learn the FGSM attack. JuanchitoCNN is also much smaller than the other models, so this could be a factor affecting the efficacy of a hybrid approach.

4. Conclusion

Our initial objective was to be able to differentiate between real images and AI-generated imitations. We were able to find that for the raw images, out-of-the-box models were incredibly effective with this problem, and that even a small custom model performed very well. We were then able to show that these models can be tricked with some effectiveness with perturbed images, and can be tricked very effectively with targeted adversarial attacks like FGSM. This particular problem did not benefit well from adversarial FGSM defenses, which is a concern for image detection problems and a potential avenue for future research to determine more effective adversarial defenses.

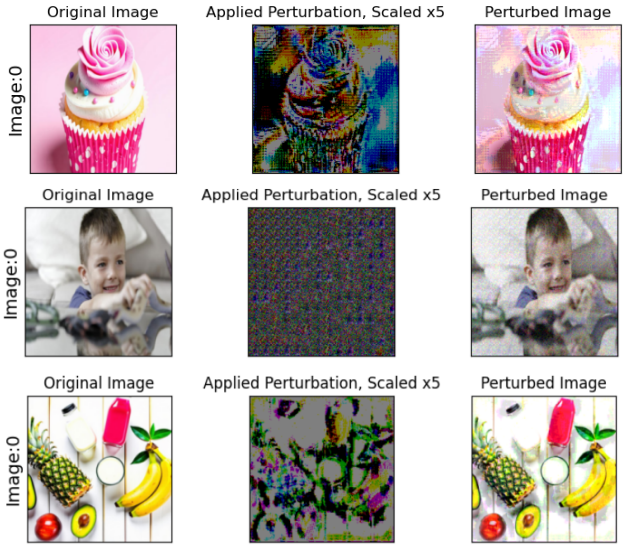
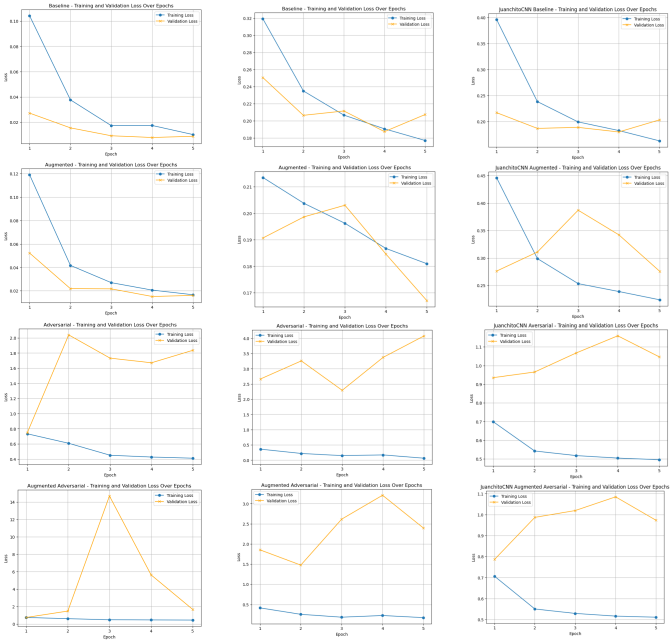


Figure 4: Examples of adversarial training images from EfficientNet, DeiT, and JuanchitoCNN



(a) EfficientNet (b) DeiT (c) JuanchitoCNN

Figure 5: Loss curves over 4 different training scenarios.

5. Work Division

Student Name	Contributed Aspects	Details
Juan	Custom CNN model, FGSM attack and defense implementation, EDA, visualizations	Explored the sizes of human vs AI images. Built the custom CNN classifier, JuanchitoCNN, and obtained all relevant information. Implemented the FGSM attack method, as well as the FGSM training method. Implemented the adversarial visualizations.
Win (Pong-Ravee)	EfficientNet model, DataLoader, EDA, hyperparameter tuning script, and Integration	Created the dataloader objects and methods, implemented EfficientNet-B0, explored the colorspace of Human vs. AI images, and integrated the 3 models to streamline training, evaluation and testing. Additionally created hyperparameter tuning script to automate a grid search.
Maluhia (Ethan)	ViT model, integration, tuning	ViT implementation, benchmarking, hyperparameter tuning, and analysis. Contributions to FGSM attack and training methods. Implemented the loss curve visualizations.
Anna	Data augmentation, test data perturbation, research, analysis, report	Built import file and data perturbation and augmentation class. Integrated with dataloader. Created architectural model for the custom CNN. Research, analysis , and attack/defense strategy on all models. Report writing and compiling.

Table 4: Contributions of team members.

Appendices

Random Perturbation Test Input

Table 5: Testing Different Perturbations on Test Input

Model	SaltPepper	Posterize	GaussianNoise	RandomShadow
EfficientNet-B0	77.62%	92.03%	99.42%	99.73%
DeiT-tiny distilled	91.52%	91.44%	91.56%	91.64%
JuanchitoCNN	49.91%	81.41%	80.20%	84.38%

	Unnamed: 0	label	height	width	mean_R	mean_G	mean_B	mean_H	mean_L	mean_S	39
count	39975.000000	39975.0	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	
mean	39975.000000	0.0	569.553171	716.566654	156.803924	147.628319	136.061878	76.685995	146.776671	89.925531	
std	23079.865684	0.0	118.956644	101.020640	49.825308	46.040022	51.716847	43.236724	46.116225	47.567583	
min	1.000000	0.0	112.000000	320.000000	1.417531	0.559710	0.786140	0.000000	2.360855	0.000000	
25%	19988.000000	0.0	512.000000	768.000000	122.062265	116.780938	98.112755	41.795277	114.490928	54.521532	
50%	39975.000000	0.0	512.000000	768.000000	158.676729	148.468516	137.466743	71.555748	146.136292	82.787417	
75%	59962.000000	0.0	640.000000	768.000000	196.142964	182.044325	176.288166	106.187693	182.357021	118.649261	
max	79949.000000	0.0	768.000000	768.000000	254.654482	253.695998	253.833737	251.272612	253.824284	253.614965	

	Unnamed: 0	label	height	width	mean_R	mean_G	mean_B	mean_H	mean_L	mean_S	39
count	39975.000000	39975.0	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	
mean	39974.000000	1.0	569.553171	716.566654	161.332810	154.419127	133.904841	69.106661	148.573973	137.351767	
std	23079.865684	0.0	118.956644	101.020640	43.848655	40.495461	50.819635	33.029889	40.212712	49.526160	
min	0.000000	1.0	112.000000	320.000000	3.926692	4.089851	0.797363	0.000000	6.866946	0.000000	
25%	19987.000000	1.0	512.000000	768.000000	133.833668	129.254881	98.536690	43.986773	121.376182	103.243885	
50%	39974.000000	1.0	512.000000	768.000000	163.415665	156.721370	136.055617	65.010434	148.499405	140.907280	
75%	59961.000000	1.0	640.000000	768.000000	193.226799	182.819714	171.645973	89.105704	177.010490	174.163402	
max	79948.000000	1.0	768.000000	768.000000	254.434491	254.886549	252.968859	242.346323	252.968859	254.937467	

mean_L	mean_S	mean_V	mean_LAB_L	mean_LAB_A	mean_LAB_B	mean_Y	mean_Cr	mean_Cb
39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000
146.776671	89.925531	169.534170	155.812541	131.275920	135.933158	149.056143	133.533449	120.663378
46.116225	47.567583	45.391519	45.009062	8.470430	13.448598	45.685650	13.343410	13.941145
2.360855	0.000000	3.063568	1.436518	66.284365	55.136851	1.742449	31.092480	13.891088
114.490928	54.521532	139.765800	126.219093	127.598007	128.394836	117.610926	127.586357	114.172743
146.136292	82.787417	172.986697	157.293921	130.327924	134.334330	149.183668	132.436019	122.171806
182.357021	118.649261	204.771987	189.863348	134.464073	142.625612	183.606916	139.640504	127.866212
253.824284	253.614965	254.682362	253.999756	201.765755	216.375511	253.885545	239.112386	204.345701

mean_L	mean_S	mean_V	mean_LAB_L	mean_LAB_A	mean_LAB_B	mean_Y	mean_Cr	mean_Cb
39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000	39975.000000
148.573973	137.351767	180.029285	161.994396	130.470514	139.920640	154.154311	133.122460	116.574152
40.212712	49.526160	38.227437	38.345730	9.775796	15.462648	39.243437	15.323422	17.238611
6.866946	0.000000	8.237147	5.844866	52.540156	48.397239	7.201032	29.707807	1.147952
121.376182	103.243885	157.312074	138.499184	126.087752	130.535018	129.117772	126.411771	107.954678
148.499405	140.907280	183.798457	164.047429	130.101463	138.307513	155.225373	133.136180	118.481457
177.010490	174.163402	207.955590	189.084368	134.757108	148.431582	181.681780	141.260131	126.218843
252.968859	254.937467	254.917808	252.986606	215.553397	222.299674	252.968859	238.436712	220.389794

Figure 6: Summary statistics for Real and AI-Generated Images.

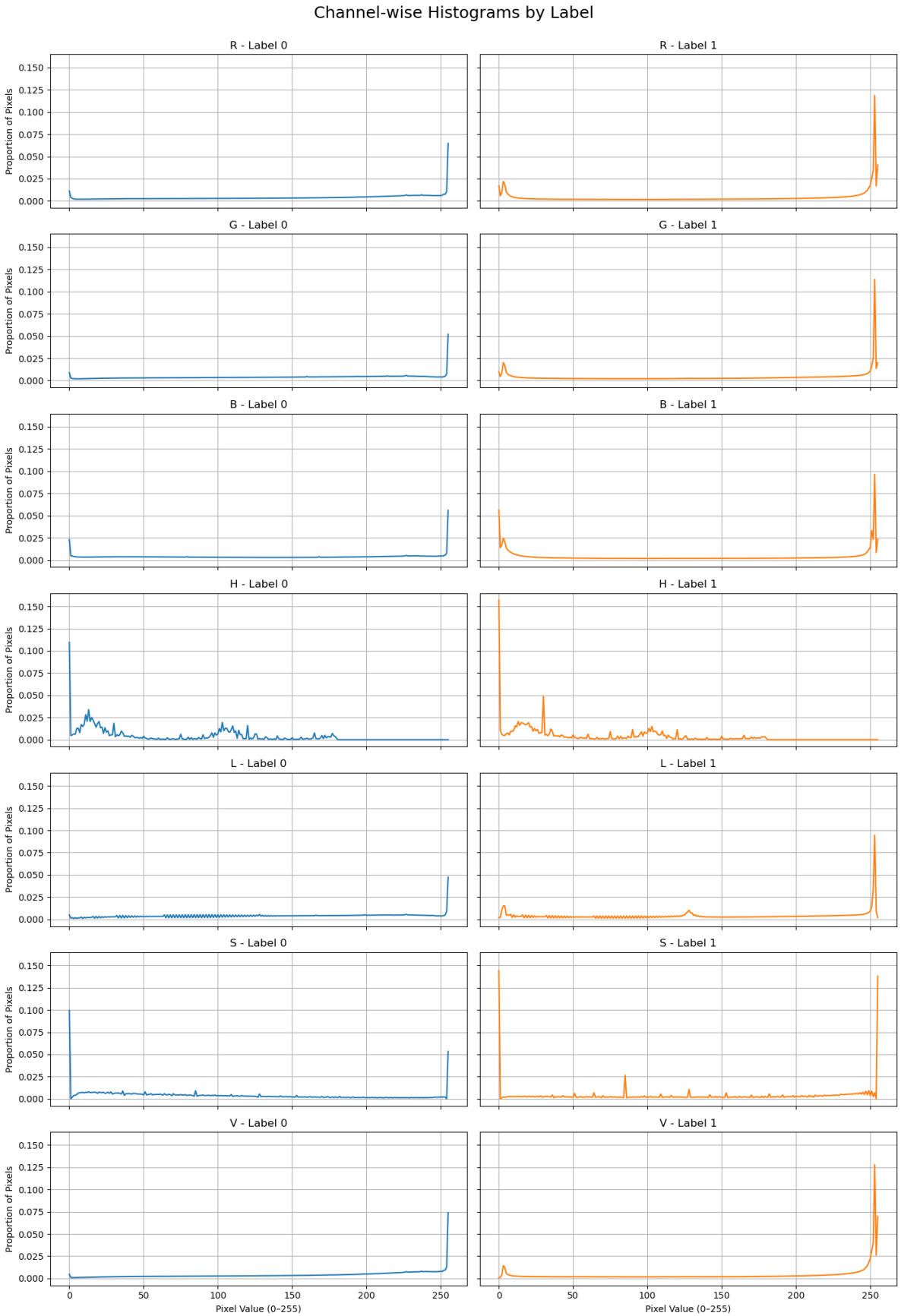


Figure 7: Histogram of RGB Channels for Real and AI-Generated Images.

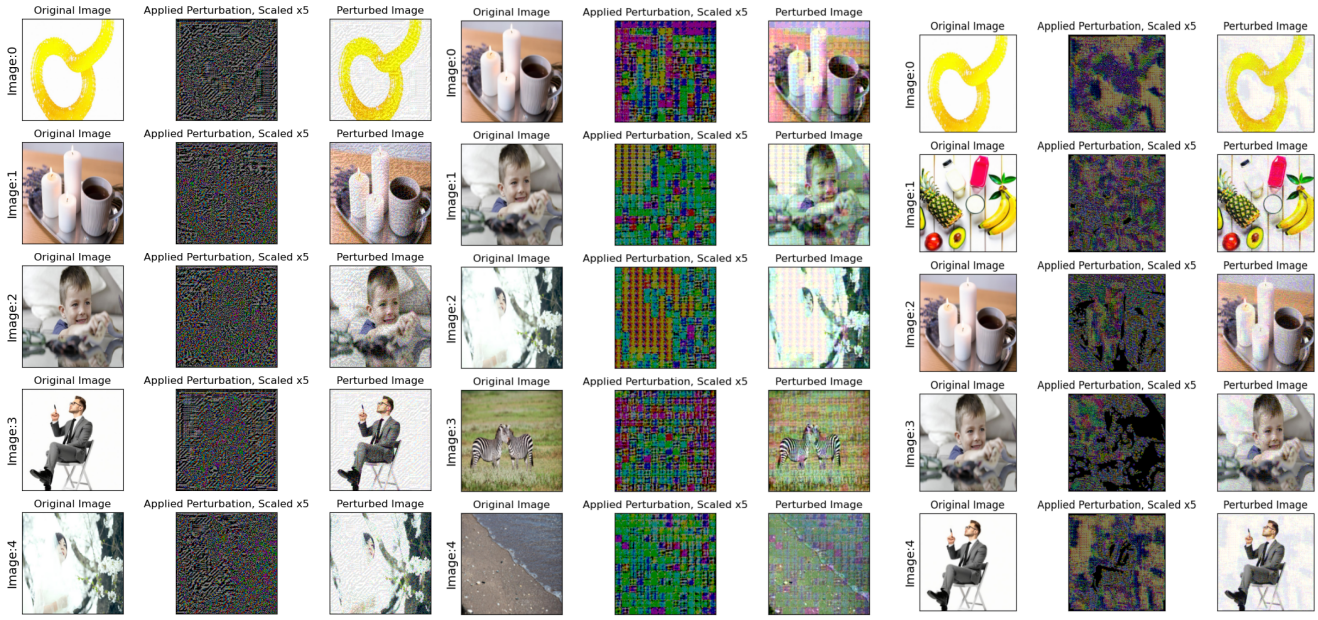


Figure 8: Examples of adversarial images on base model from EfficientNet, DeiT, and JuanchitoCNN

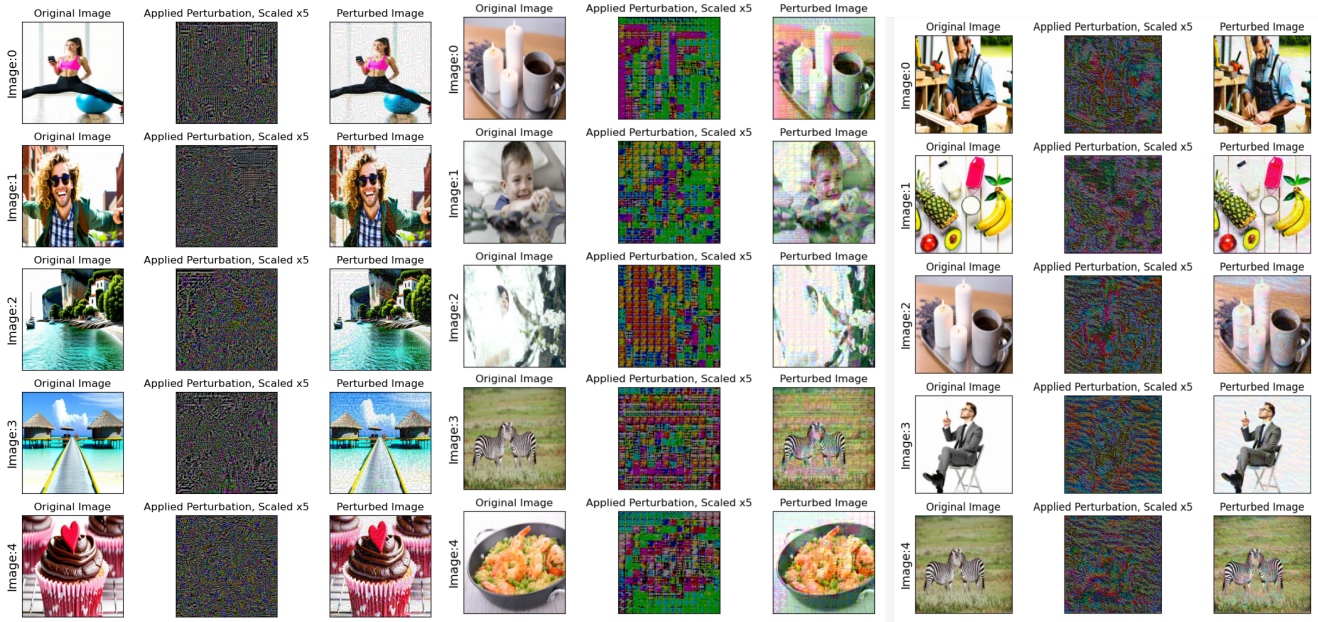


Figure 9: Examples of adversarial images on augmented model from EfficientNet, DeiT, and JuanchitoCNN

Figure 10: Examples of adversarial images on adversarial model from EfficientNet, DeiT, and JuanchitoCNN

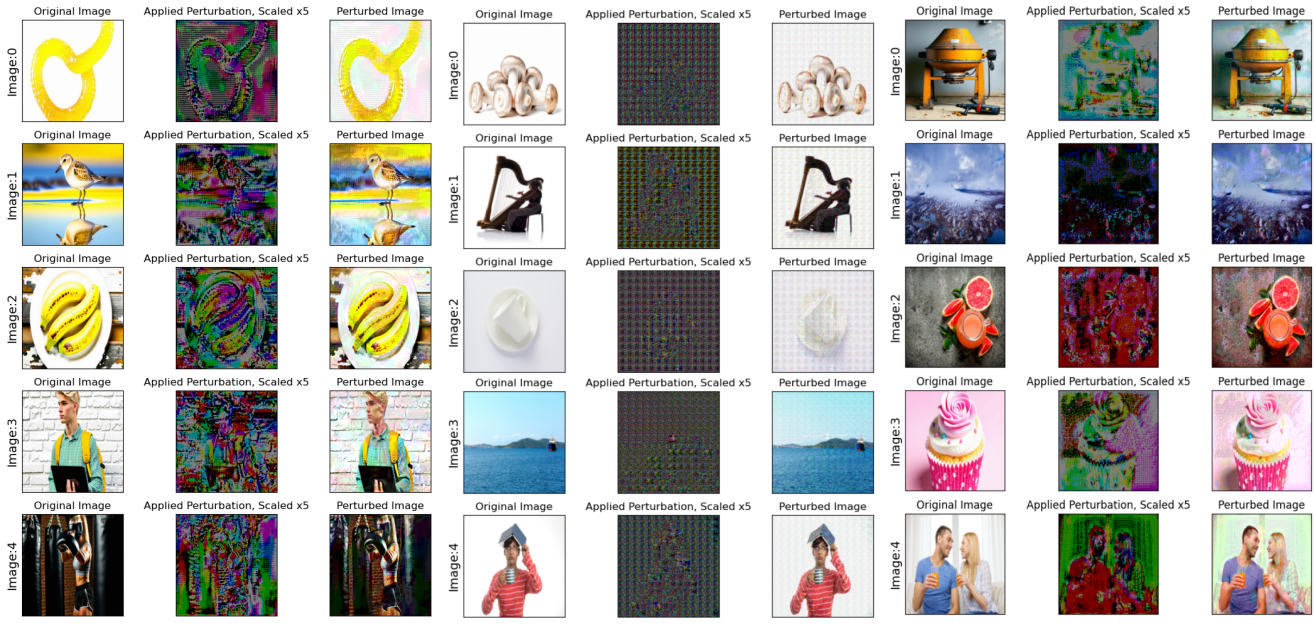


Figure 11: Examples of adversarial images on augmented and adversarial model from EfficientNet, DeiT, and JuanchitoCNN

References

[1] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020. 3, 4

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. google/vit-base-patch16-224. 2, 4

[3] Gamaleldin F. Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alex Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both computer vision and time-limited humans, 2018. 1

[4] Ilya Figotin. Imagenet 1000 (mini), 2020. Accessed: (25 April 2025). 3

[5] Stanislav Fort. Multi-attacks: Many images + the same adversarial attack → many target labels, 2023. 1

[6] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness, 2022. 5

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015. 1, 2, 3

[8] Nathan Inkawhich. Adversarial example generation, 2025. Accessed: (14 April 2025). 2

[9] Kaggle. Detect ai vs. human-generated images. *NumPy v1.26 Manual*, 2025. Accessed: (3 March 2025). 1, 2

[10] Zico Kolter and Aleksander Madry. Adversarial training, solving the outer minimization, 2025. Accessed: (20 April 2025). 2

[11] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019. 4

[12] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks?, 2022. 4

[13] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020. 1

[14] Alessandra Sala, Manuela Jeyaraj, Toma Ijatomi, and Margarita Pitsiani. Ai vs. human-generated images, 2025. Accessed: (3 March 2025). 1

[15] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. 2

[16] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. 2

[17] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers ‘i& distillation through attention, 2021. google/vit-base-patch16-224. 2, 4

[18] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples, 2017. 1, 3

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 2

[20] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A. Efros. Cnn-generated images are surprisingly easy to spot... for now, 2020. 2

[21] Bo Yang, Kaiyong Xu, Hengjun Wang, and Hengwei Zhang. Random transformation of image brightness for adversarial attack, 2021. 4

[22] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks, 2017. 1, 4