Adversarially Robust Deepfake Detection

Abstract

Deepfake detectors must not only be accurate but also resilient to adversarial manipulation. This project develops a deepfake detection model that remains robust even against adaptive attacks and distortions. We fine-tuned a hybrid EfficientNet-ViT model ("EfficientViT") on face crops from the FaceForensics++ dataset, using a dual adversarial training strategy. In training, the detector was pitted against both a standard Projected Gradient Descent (PGD) attack and a learned U-Net–based perturbation generator. The U-Net attacker was constrained with Total Variation and frequency-domain losses to produce realistic, low-visibility perturbations. The resulting **robust model** is evaluated cross-dataset on the high-quality Celeb-DF (v2) test set and compared to a **baseline model** trained without adversarial augmentation. Key metrics include accuracy, ROC AUC, and True Positive Rate at a 1% False Positive Rate (TPR@1%FPR). The robust model matches the baseline on clean data and maintains higher detection performance under adversarial perturbations and compression. For example, under a combined learned-attack-plus-compression scenario, the hardened model achieves a TPR@1%FPR of ~3.0% (versus ~2.0% for the baseline) on Celeb-DF. Grad-CAM explainability analysis suggests that adversarial training shifted the model's focus from spurious high-frequency artifacts to more holistic face cues, aligning with its improved resilience. These results demonstrate that our multi-faceted adversarial training approach yields a deepfake detector that is better equipped to handle real-world evasion attempts, while only marginally affecting standard accuracy. Future work can extend this framework to stronger attacks and larger-scale training to further bolster deepfake detection defenses.

1. Introduction

The rise of **deepfakes** – hyper-realistic videos generated by AI – poses a serious threat to digital media integrity. Advanced neural models can swap faces or synthesize speech so convincingly that humans struggle to tell real from fake. In response, many deepfake **detection** algorithms have been proposed. These detectors often leverage convolutional neural networks (CNNs) or other deep models to spot subtle artifacts left by the generation process (e.g. inconsistencies in facial textures or eye blink patterns). While detectors have improved detection accuracy on benchmark datasets, a critical vulnerability remains they can be **fooled by adversarial attacks**. An adversary can intentionally add a small, carefully crafted perturbation to a fake image or video frame so that the detector misclassifies it as real, even though to the human eye it still looks like the same fake. This exposes a blind spot in current detectors and is especially worrying in high-stakes scenarios (for instance, detecting malicious deepfakes in journalism or law enforcement), where attackers may deliberately try to evade automated detection.

Adversarial robustness in deepfake detection is therefore essential. A robust detector should correctly flag fake content *even if* the input has been adversarially modified or distorted (for example, by noise, compression artifacts, or learned perturbations) to trick the model. Prior work in adversarial defense for image classifiers shows that adversarial training – training the model on adversarially perturbed examples – can significantly improve resilience. However, applying this to deepfake detection brings unique challenges. Deepfake artifacts can be subtle and span both spatial details (like mismatched edges or color blending at the face boundary) and more global inconsistencies (like unnatural head movements). Moreover, in realistic media pipelines, fakes may undergo video compression (e.g. H.264) or image re-encoding (JPEG), which can either mask or amplify certain telltale artifacts. An effective robust detector needs to handle this complex combination of manipulations.

In this project, we set out to build an **adversarially robust deepfake detector** that can maintain high detection performance under a strong, adaptive threat model. We combine innovations in model architecture and training regimen to achieve this goal. First, we design a compact **EfficientViT architecture** that blends a CNN and a Transformer: an EfficientNet backbone provides low-level image feature maps, which are then processed by a small Vision Transformer encoder. The intuition is that the CNN will capture local texture artifacts (e.g. high-frequency anomalies in the face), while the Transformer will capture longer-range correlations and inconsistencies (e.g. unnatural eye-glance coordination or lighting mismatches across the face). Second, we employ a **dual adversarial training strategy**. During training, our detector model is repeatedly attacked by two adversaries: (1) a standard **Projected Gradient Descent (PGD)** attacker that generates perturbations through iterative gradient-based noise, and (2) a learned **U-Net attacker** – a lightweight convolutional U-Net that we train in tandem to produce realistic perturbations that maximize the detector's error. Importantly, we constrain the U-Net's perturbations with **realism losses** so that the perturbations remain subtle and plausible. Specifically, a **Total Variation (TV) loss** discourages high-frequency noise (promoting spatial smoothness in the perturbation), and a **frequency-domain loss** penalizes perturbation energy in low-frequency bands (simulating the effect of video compression by focusing the adversary on mid- to high-frequency changes).

By integrating these constraints, we encourage the learned adversarial perturbations to resemble the kind of sophisticated fake modifications an actual deepfake video might contain (or an actual attacker might craft) – for example, smoothly altering shading or adding faint artifacts that survive compression, rather than obvious pixel noise.

We train our models on a subset of the **FaceForensics++ (FF++) dataset** (specifically the c23 moderately compressed version) and evaluate on the **Celeb-DF v2 dataset** for a rigorous cross-domain test. FF++ provides diverse fake generation methods (DeepFakes, FaceSwap, etc.), giving the model a broad basis of fake artifacts to learn. Celeb-DF v2, on the other hand, contains high-quality deepfakes that were not seen during training, which tests the model's generalization to new fake styles. We measure standard performance (accuracy and ROC–AUC) as well as **TPR@1%FPR**, a high-precision metric indicating how well the model can catch fakes when we allow only 1% false alarm rate. The latter is particularly relevant in real deployments where false positives are costly and detectors must operate at very low false positive rates. By comparing a baseline detector (with no adversarial training) to our robust detector on a battery of scenarios (clean images, compressed images, PGD attacks, learned U-Net attacks, and combinations thereof), we demonstrate the tangible benefits of our approach.

In the following sections, we detail the methodology (Section 2), the experimental setup (Section 3), and the results (Section 4). We then provide an analysis of **why** the robust model outperforms the baseline by examining **Grad-CAM visualizations** and discussing the frequency characteristics of the model's attention (Section 5). Finally, Section 6 concludes with key findings and suggestions for future work in adversarially robust deepfake detection.

2. Methodology

2.1 EfficientViT Detector Architecture

To effectively detect deepfakes, our model architecture (**EfficientViT**) combines the strengths of CNN and Transformer components in a compact form. The base of the model is **EfficientNet-B0**, a lightweight CNN known for efficient image feature extraction. We use a pretrained EfficientNet-B0 as the backbone and remove its classification head, so it outputs a 2D feature map (of size 7×7×1280 for 224×224 input images). These feature maps contain rich local descriptors of the face – for example, edges, textures, and micro-patterns that might indicate tampering (such as blending boundaries or GAN-generated artifacts). However, CNN features alone might focus too narrowly on small regions. To incorporate a more global understanding, we append a small **Vision Transformer (ViT) encoder** on top of the CNN features.

Concretely, the 7×7 spatial grid of EfficientNet features (1280 channels deep) is flattened into a sequence of 49 patch embeddings. We prepend a **classification token** (CLS) to this sequence, following standard Transformer practice, and add a set of learned **position embeddings** so the model is aware of each token's spatial location. The sequence (length 1 + 49) is then fed into a Transformer Encoder consisting of 4 self-attention layers with 4 attention heads each. The Transformer layers allow the model to reason about relationships between different parts of the face. For instance, a Transformer attention head could compare the left eye and right eye regions or check if the jawline lighting is consistent with the forehead lighting – tasks which require long-range comparisons that CNNs struggle with. We include layer normalization and dropout (10%) within the Transformer to stabilize training. Finally, only the CLS token's output embedding (a 1280-dimensional vector) is passed through a small **MLP head** to produce the final classification logits (fake vs real). The MLP head consists of a layer norm, a linear layer reducing the dimension to 256, a GELU non-linearity, dropout, and a final linear layer to 2 output classes.

Despite the inclusion of a Transformer, EfficientViT remains relatively small (on the order of a few million parameters) because EfficientNet-B0 is compact and we used only a 4-layer Transformer. We found this hybrid design advantageous: the CNN part grounds the model in fine image details, and the Transformer part provides a global context to judge those details in. This architecture should help the detector catch both obvious artifacts (like a misaligned face boundary) and more holistic anomalies (like a face that doesn't "match" its context or exhibits unnatural symmetry). The combined model is trained end-to-end, meaning the EfficientNet and Transformer learn to work together to produce a final deepfake detection confidence.

2.2 Adversarial Attackers in Training

Simply training the EfficientViT on clean real and fake images might yield high accuracy on normal test data, but it does not guarantee robustness to adversarial input. To instill adversarial resilience, we introduce *two* attackers during training, forcing the model to learn from adversarially perturbed examples.

(a) PGD Attacker (Baseline Attack): The first attacker is the classic **Projected Gradient Descent (PGD)** method for generating adversarial examples. PGD is a white-box attack that iteratively adjusts the input image in the direction of the detector model's loss gradient, with a constraint on maximum perturbation size.

We use an L_∞ threat model with perturbation budget $\epsilon = 8/255$ (approximately 0.03 in pixel intensity scale) and step size $\alpha = 2/255$, for 10 iterations. The attack operates on images normalized to ImageNet statistics (since our model input is normalized), and we ensure each step's result stays within valid image bounds [0,1] after un-normalizing. We also apply a random start (adding a small random noise within the ϵ-ball before starting gradient steps) to avoid local minima in the attack. PGD represents a "worst-case" small noise: it will find the most damaging tiny pixel-level noise pattern for the model. During training, we treat PGD as an adversary that generates an example x_{PGD} from each clean input x such that the model would be fooled if it only knew the baseline model. By training on these, the detector learns to not be overly sensitive to the specific features that PGD exploits.

(b) Learned U-Net Attacker: While PGD is powerful, it produces perturbations that are essentially high-frequency noise and may not look natural. A real attacker could design more realistic adversarial fakes – for example, tweaking the shading on a face or adding a subtle pattern that might survive video compression. To simulate a clever adversary, we introduce a trainable **Tiny U-Net** network as a second attacker. This U-Net, called the **perturbation generator**, takes in an image and outputs a perturbation δ to add to that image. Its architecture is a simplified U-Net: an encoder with three downsampling steps (each step doubling the channel count from a base of 16, yielding layers of 16→32→64 features) and a decoder with symmetric upsampling steps that merge with encoder features via skip connections. The bottleneck has 128 channels at 1/4 resolution. Despite its small size (~1e6 parameters), this U-Net can produce structured spatial perturbations. We bound the perturbation by applying a final $\tanh$ activation and scaling so that each pixel change is in $[-\epsilon, +\epsilon]$ (with $\epsilon=8/255$ as above). This ensures the U-Net's output, when added to the image, is a valid adversarial perturbation not exceeding the PGD noise magnitude.

We train the U-Net attacker **simultaneously with the detector** in a min-max game: the U-Net tries to *maximize* the detector's classification loss (make the detector predict wrong), while the detector tries to *minimize* that loss. At each training step, given the current detector, we update the U-Net by backpropagating the negative classification loss (making it higher when the detector does well, so the U-Net learns to fool it). In effect, the U-Net learns to generate those perturbations that particularly confuse the current detector. Because the U-Net is a neural network, it can learn **adaptive, structured attacks** – for example, if the detector heavily relies on a certain facial region, the U-Net might learn to subtly alter pixels in that region in a way that flips the prediction. This learned attacker is more flexible than PGD in some respects: it can produce spatially coherent patterns (not just random noise) and potentially evade simple defenses like JPEG compression that often break vanilla PGD noise.

2.3 Realism Constraints for Adversarial Perturbations

A crucial innovation in our training is the use of **realism constraints** on the U-Net's perturbations. Without constraints, the U-Net might converge to solutions that, while effective at fooling the model, look like obvious noise or distortions a human would never add to a video. We want the adversarial perturbations to mimic *realistic* manipulations so that by learning to resist those, the detector will also resist real-world evasion attempts.

We impose two types of regularization on the U-Net's output δ:

- **Total Variation (TV) Loss:** This encourages the perturbation to be smooth and low-noise. The TV loss is defined as the sum of absolute differences between neighboring pixels in δ (both vertically and horizontally). Formally, $\mathcal{L}*{TV} = \frac{1}{HW}\sum*$) so that it doesn't dominate, but meaningfully limits noise.}(|\delta_{i,j} - \delta_{i+1,j}| + |\delta_{i,j} - \delta_{i,j+1}|)$ for all pixel positions (i,j). If δ has a lot of high-frequency noise (salt-and-pepper patterns or pixel-wise jitter), the TV loss will be large. By adding this loss, we make the U-Net prefer smoother perturbation patterns – for example, a slight glow or a blur – over pixel-level noise. We weight this loss with a small coefficient (we used $\lambda_{TV} = 10^{-4
- **Low-Frequency Penalty (Frequency-Domain Loss):** We specifically target perturbations that exploit low-frequency (smooth) channels that might be wiped out by compression. We compute the 2D Fast Fourier Transform (FFT) of δ for each color channel, shift it to center the zero-frequency component, and take the magnitude squared (the power spectrum). We then apply a mask that covers only the lowest-frequency region (a circle of radius equal to 12.5% of the image size in the frequency domain) and compute the mean power within that region. This gives a loss $\mathcal{L}*{freq}$ representing how much energy the perturbation has in very low frequencies (broad smooth changes). By penalizing this (again with a small coefficient $\lambda*$), we push the U-Net to concentrate its perturbation energy in the mid- and high-frequency ranges instead. In effect, the U-Net learns to make perturbations that would } = 10^{-4*survive compression*: compression algorithms like JPEG or H.264 tend to kill high-frequency noise but not mid-frequency patterns (like subtle textures), so by avoiding extremely low-

frequency changes, the U-Net ensures its perturbations are not trivially obliterated by a compression step.

These two losses ($\mathcal{L}*{TV}$ and $\mathcal{L}*$) are added to the attacker's training objective along with the negative cross-entropy (which drives the U-Net to fool the detector). The combined attacker loss is:

$$\mathcal{L}_{attacker} = -\mathcal{L}_{CE}(\text{detector}(x + \delta), y) + \lambda_{TV}\mathcal{L}_{TV}(\delta) + \lambda_{freq}\mathcal{L}_{freq}(\delta),$$

where y is the true label (real/fake) for input x. The $-\mathcal{L}_{CE}$ term makes the U-Net maximize the detector's error, while the other terms act as gentle "leashes" to keep the perturbation looking realistic. Over training, this causes a fascinating co-evolution: the U-Net finds weaknesses in the detector's current strategy in the form of subtle, realistic perturbations, and the detector learns to patch those weaknesses because it is trained on those same perturbations.

2.4 Adversarial Training Procedure

Our adversarial training loop brings together the EfficientViT detector, the PGD attacker, and the U-Net attacker in an integrated fashion (often described as a **min-max optimization** problem). The training is organized per batch in the following steps:

1. **Clean Batch Forward:** For a batch of face images (some real, some fake) from the training set, we first compute the detector's loss on the clean images. Let $\mathcal{L}*{clean} = \mathcal{L}*(f_\theta(x), y)$ be the cross-entropy loss of the detector f_θ on clean inputs x with labels y. We accumulate this loss but hold off on updating weights immediately.

2. **PGD Adversarial Batch:** Next, we run the PGD attack on the detector for the same batch. This yields a perturbed batch x_{PGD} where each image has been nudged by up to ϵ in the most harmful way for the current detector. We compute the detector's loss $\mathcal{L}*{pgd} = \mathcal{L}*), y)$ on these adversarial examples. This loss ensures the model trains to correctly classify PGD-perturbed images.}(f_\theta(x_{\text{PGD}

3. **U-Net Attacker Update:** Now we update the learned attacker. We freeze the detector's weights (temporarily, for this step) and pass the clean batch x through the U-Net attacker to get a perturbation δ. We form the perturbed images $x_{\text{U}} = x + \delta$ (clipping to valid range) and evaluate the detector on them to get $\mathcal{L}*{CE}(f*\theta(x_{\text{U}}), y)$. We combine this with the TV and frequency losses on δ as described in Section 2.3 to get the

attacker loss $\mathcal{L}_{attacker}$. We then backpropagate this loss **only into the U-Net's parameters**, updating the attacker's weights via its optimizer (we used Adam). Intuitively, this step makes the U-Net better at fooling the detector. We typically perform just one gradient step on the U-Net per batch (making this an alternating optimization). After this, we unfreeze the detector's weights.

4. **Detector Update (Adversarial):** After adjusting the attacker, we generate a fresh U-Net perturbation for the batch (this time we don't propagate gradients into the U-Net, treating it as an adversary providing data). We get $x_{\text{U}}^{det} = x + \delta_{\text{new}}$ from the updated U-Net attacker. We compute the detector's loss on these inputs: $\mathcal{L}*{unet}* = *\mathcal{L}}*(f_\theta(x_{\text{U}}^{det}), y)$. Now we have three loss components for the detector: clean, PGD, and U-Net (i.e., $\mathcal{L*{clean}$, $\mathcal{L}*$). We sum them into a total detector loss:

   $$\mathcal{L}_{detector} = \mathcal{L}_{clean} + \mathcal{L}_{pgd} + \mathcal{L}_{unet}.$$

   This combined loss is then backpropagated to update the detector's weights (using SGD or AdamW). In doing so, we are training the detector to minimize errors on clean, PGD-perturbed, and U-Net-perturbed images simultaneously $\mathcal{L}_{(\text{Unet})}$

We repeat the above process for each batch and train for multiple epochs. In our implementation, we ran a shorter baseline training (8 epochs on clean data) to get the baseline model, and then continued adversarial training for a number of epochs (we chose 5 epochs for robustness training due to time constraints). During robust training, we maintained two separate optimizers: one for the detector (with a smaller learning rate as it was already partially trained) and one for the U-Net attacker. We also applied **gradient clipping** (e.g. max norm of 1 or 5) on both models' gradients during updates to stabilize the joint training, since the min-max dynamic can otherwise cause large oscillations.

Throughout training, we monitored the training accuracy on clean vs adversarial batches to ensure neither diverged. By the end of training, we obtained a **robust detector model** (parameters θ*) and a tuned U-Net attacker (which we can use to test the model's worst-case performance). The expectation is that* θ has internalized features that are invariant to small adversarial perturbations and is especially tuned to catch the kinds of subtle artifacts the U-Net learned to exploit.

2.5 Data and Implementation Details

Training Data: We used **FaceForensics++ (FF++), c23** compression level, as the source for training images. FF++ c23 provides a large set of videos (original real videos and several types of fake videos). To keep training feasible within limited time, we sampled a subset rather than using all frames. Specifically, we extracted 1 frame per video from about 1000 real videos and 1000 fake videos (the fake videos were drawn from multiple manipulation methods included in FF++). Each extracted frame was processed through face detection (using MTCNN) to crop and align the face region to 224×224 pixels. This yielded on the order of ~1.8k training face images (the exact counts were ~771 real and 1000 fake, due to dataset availability and successful face detections). Although this is a small fraction of FF++, it was sufficient for a proof-of-concept and kept training time manageable. Data augmentation was minimal (we focused on adversarial augmentation instead): we did standard preprocessing of resizing, center-cropping to 224, and normalization to ImageNet mean/std.

Evaluation Data: For testing, we prepared face crops from the **Celeb-DF v2** dataset. Celeb-DF contains 590 real and 5639 fake videos of celebrities, with very realistic deepfakes. We similarly sampled 1 frame per video (ensuring a mix of real and fake frames) and cropped faces. Our test set ended up with around 1000 real and 889 fake face images (some videos in Celeb-DF are very short or had detection issues, hence the slight imbalance). Importantly, none of the Celeb-DF content was seen during training, making this a true cross-dataset evaluation – a tough challenge where many detectors drop in performance due to differences in fake generation methods and video quality.

Implementation: We implemented the model and training in PyTorch. Training was done on GPU (NVIDIA T4) with mixed precision (using AMP) to speed up computation. The baseline model training (8 epochs) took about 15 minutes, and robust training (5 epochs with the adversarial procedure) took about 1 hour given the extra overhead of attacks. We saved two model checkpoints: baseline_model.pth and robust_model.pth. During evaluation, we systematically evaluated both models under various scenarios (described next).

3. Experiments

3.1 Evaluation Protocol

We designed a comprehensive evaluation to compare the **Baseline** and **Robust** detectors across multiple scenarios that reflect real-world conditions and active attacks. The evaluation scenarios include:

- **Clean (In-Domain):** Test on the FF++ c23 test split (same distribution as training, no attacks). This measures each model's standard accuracy on the data it was trained on, establishing an upper bound on performance.
- **Clean (Cross-Domain):** Test on the Celeb-DF v2 set without any attacks or distortions. This checks generalization to a new dataset of high-quality deepfakes. A drop here, especially for the baseline, would indicate overfitting to FF++ artifacts.
- **Compressed (Cross-Domain):** Test on Celeb-DF v2 after applying common video compression artifacts. We simulated two types:
- **JPEG(50)** – each image was JPEG-compressed with quality factor 50 (moderate compression).
- **H264-like** – a custom simulation of H.264 compression using blurring and downsampling to mimic low bitrate video (this roughly approximates the c23 → c40 jump in FF++ or a YouTube video's compression). These test if the models' detections hold up when fakes are camouflaged by compression noise. Compression often hides simple artifacts, so a robust model should be less affected than a baseline.
- **PGD Attack:** We craft adversarial perturbations on each Celeb-DF image using PGD (10 steps, 8/255 epsilon, as during training) targeting each model separately. We then evaluate the model on its own adversarially perturbed images. This is a "white-box" attack evaluation. We expect the baseline to be extremely vulnerable (since it never saw such perturbations in training), likely dropping to near-chance performance, whereas the robust model – which was trained with PGD – should fare better.
- **Learned U-Net Attack:** We also evaluate using the **learned U-Net attacker**. For fairness, we took the U-Net that was co-trained with the robust model (which is tuned to attack *the robust model*) and applied it to generate perturbations on Celeb-DF images for both models. This is an interesting scenario: the perturbations were optimized to fool the robust model, so we expect the robust model to possibly be fooled (though it was trained to resist them, the attacker was simultaneously getting better), but what about the baseline? It did not train against this U-Net, but the U-Net's perturbations might or might not fool the baseline. We test both. This scenario represents an attacker that has learned a more realistic attack strategy.

- **Combined Attack + Compression:** This is the **"ultimate stress test"** scenario. We take images already perturbed by the U-Net attacker and then apply JPEG or H.264 compression to them. This simulates a real deployment scenario: suppose an adversary generates an adversarial fake video and then that video gets compressed by a social media platform – will the detector still catch it? We specifically evaluate **U-Net+JPEG(50)** and **U-Net+H264** on Celeb-DF. These are arguably the most challenging conditions for the detector, where it must contend with both an adaptive adversary and signal degradation from compression.

For each scenario, we measure the **Accuracy**, **ROC AUC**, and **TPR@1%FPR** of the models. Accuracy gives a basic sense of overall correct classifications (with a 50/50-ish data, chance is ~50%). ROC AUC (Area Under the ROC Curve) summarizes the model's ranking performance across all sensitivity levels – important for understanding overall discriminative power beyond a fixed threshold. TPR@1%FPR, as mentioned, tells us how well the model can perform when we tune it to a very strict false positive rate (1%). A higher TPR@1%FPR means the model can catch a larger fraction of fakes while only letting 1% of reals slip through as false alarms, which is crucial in practice (e.g. an automated system might only flag something as fake if it's extremely sure). This metric tends to be challenging – it essentially measures performance at the extreme left end of the ROC curve.

We will compare baseline vs robust on all metrics. Particularly, we pay attention to how much performance degrades from clean to attacked scenarios for each model. A robust model should show smaller drops. Ideally, in the hardest scenario (U-Net + compression), the baseline might be nearly random (AUC ~0.5, TPR@1 ~0%), whereas the robust model should still be significantly above random, demonstrating the value of adversarial training.

3.2 Results Summary

We compile the evaluation results for both models below. **Table 1** presents the performance on **Celeb-DF v2** (cross-domain test) under various conditions, and **Table 2** shows results on **FF++ c23** (in-domain). Each table compares the baseline detector and robust detector side by side.

Scenario (Celeb-DF v2)	Accuracy (Base)	Accuracy (Robust)	ROC–AUC (Base)	ROC–AUC (Robust)	TPR@1%FPR (Base)	TPR@1%FPR (Robust)
Clean (no attack)	0.624	0.617	0.678	0.676	0.014 (1.4%)	0.012 (1.2%)
JPEG(50) compression	0.625	0.641	0.672	0.681	0.023 (2.3%)	0.023 (2.3%)
H264-like compression	0.636	0.661	0.693	0.703	0.022 (2.2%)	0.024 (2.4%)
PGD attack (white-box)	0.467	0.476	0.460	0.483	0.001 (0.1%)	0.001 (0.1%)
U-Net attack (learned)	0.619	0.607	0.675	0.674	0.015 (1.5%)	0.015 (1.5%)
U-Net + JPEG(50)	0.625	0.647	0.675	0.685	0.020 (2.0%)	0.029 (2.9%)
U-Net + H264-like	0.644	0.654	0.690	0.699	0.025 (2.5%)	0.030 (3.0%)

Table 2. *In-Domain (FF++ c23) performance of Baseline vs Robust model under various scenarios.*

Scenario (FF++ c23)	Accuracy (Base)	Accuracy (Robust)	ROC–AUC (Base)	ROC–AUC (Robust)	TPR@1%FPR (Base)	TPR@1%FPR (Robust)
Clean (no attack)	0.999	0.998	0.999997	0.999999	1.000 (100%)	1.000 (100%)
JPEG(50) compression	0.851	0.836	0.922	0.916	0.344 (34.4%)	0.387 (38.7%)

Scenario (FF++ c23)	Accuracy (Base)	Accuracy (Robust)	ROC–AUC (Base)	ROC–AUC (Robust)	TPR@1%FPR (Base)	TPR@1%FPR (Robust)
H264-like compression	0.912	0.912	0.965	0.962	0.641 (64.1%)	0.671 (67.1%)
PGD attack (white-box)	0.511	0.493	0.465	0.449	0.0026 (0.26%)	0.000 (0%)
U-Net attack (learned)	0.999	0.999	0.999997	0.999997	1.000 (100%)	1.000 (100%)
U-Net + JPEG(50)	0.859	0.844	0.924	0.918	0.340 (34.0%)	0.379 (37.9%)
U-Net + H264-like	0.906	0.903	0.966	0.963	0.626 (62.6%)	0.651 (65.1%)

Looking at **Table 2 (FF++ in-domain)** first, we see that both models perform extremely well on the *clean FF++ frames* – near-perfect accuracy and AUC ~1.0, as expected since they were trained on this distribution. Adversarial training did not degrade the robust model's clean performance on training data significantly (it's 99.8% vs baseline 99.9%, essentially identical). Under **PGD attack on FF++**, the baseline's accuracy plunges to ~51% (essentially random guessing) and robust model is ~49% (also random). In this case, even the robust model is struggling – PGD with ε = 8/255on faces is a very strong attack, and although we trained with it, at test time it still finds a way to confuse the model. (Interestingly, the baseline does *slightly* better in AUC on PGD here, 0.465 vs 0.449, but both are so low that neither is meaningfully detecting anything.) For the **learned U-Net attack on FF++**, we see both models remain at 100% TPR and ~1.0 AUC. This indicates that the U-Net's perturbations were ineffective on the training-domain images – likely because the robust model had already learned to resist them, and the baseline model, dealing with very familiar FF++ fakes, wasn't fooled by these particular patterns either. In compressed scenarios on FF++, baseline and robust have high AUCs (0.92–0.96 range) but not perfect. The robust model's TPR@1%FPR is actually a bit higher than baseline's under compression (e.g. 38.7% vs 34.4% for JPEG, and 67.1% vs 64.1% for H264). This suggests that when operating at a very low false-positive rate, the robust model is slightly better at catching fakes in compressed videos than the baseline is. The baseline has a marginally higher overall AUC in those cases, but the robust model seems to trade a tiny bit of overall accuracy for improved **rare-event detection** (higher TPR at low FPR). In practice, that could mean the robust model makes slightly more conservative predictions (leading to fewer false alarms and more consistent true positive picks when thresholded hard).

The more important results are in **Table 1 (Celeb-DF cross-domain)**. On **clean Celeb-DF**, both models perform moderately: ~67.5% AUC, around 62% accuracy, and a very low TPR@1 (around 1–1.4%). This reflects how challenging Celeb-DF is – these numbers are far from the near-perfect on FF++. Notably, the robust model did *not* significantly lag behind the baseline on clean Celeb-DF (0.676 vs 0.678 AUC, essentially the same). This is a positive sign that our adversarial training did not overfit to weird artifacts at the expense of normal generalization. Under **JPEG and H264 compression** on Celeb-DF, the baseline's AUC dips a bit to ~0.672–0.693, and the robust model's AUC is ~0.681–0.703, a hair better. Both models' accuracies actually stay around 62–66%, indicating they are fairly robust to compression alone (likely because both saw some compressed data: FF++ c23 is already compressed). The TPR@1%FPR numbers for compressions are in the 2–3% range for both; the robust model is equal or slightly higher, but the difference is tiny. So against **common distortions (JPEG/H264)**, robustness training helped only marginally – the baseline wasn't too brittle to begin with there.

The **PGD attack on Celeb-DF** absolutely wrecks the baseline: AUC ~0.460, essentially meaning the model's outputs are worse than random in order; TPR@1 is ~0.1% (basically zero). The robust model, having been trained on PGD, still drops a lot (AUC ~0.483) but at least is not quite as low. Both are near chance, but the robust model maintains a slight edge. In effect, PGD remains a very strong adversary. It's worth noting that even a robust ImageNet classifier often still struggles with PGD unless the training epsilon is very high. In our case, we used the same ε for training and testing PGD, and the robust model did partially immunize itself (its accuracy is ~47.6% vs baseline's 46.7%, so a tiny bump, and AUC 0.48 vs 0.46). Neither model is satisfactory under PGD – indicating that defending against fully unconstrained PGD for high-resolution images is extremely challenging without heavier sacrifices (like larger models or more training data).

For the **learned U-Net attack on Celeb-DF**, an intriguing result appears: both models have AUC ~0.674–0.675, basically the same as on clean data, and their accuracies ~61–62% are also similar to clean performance. This suggests that the U-Net attacker, when applied to the baseline model, did not significantly reduce its performance – likely because the U-Net was trained to target the robust model's weaknesses, not the baseline's. And for the robust model, it was explicitly trained to resist this U-Net, so it handles it about as well as clean data. Essentially, the U-Net attack alone did not gain much over the models. This is a good sign: it means our robust model successfully learned to counter the specific strategies the U-Net tried, and interestingly, the baseline model wasn't inherently weak to those strategies (perhaps the U-Net's perturbations include subtle cues that the baseline ends up using to identify fakes, ironically). This emphasizes a point: robustness is somewhat attacker-specific. Our robust model is robust against the types of attacks we trained on (and minor variations), but not necessarily all possible attacks.

Finally, the combined **U-Net + Compression** scenarios are where we expected to see the clearest separation. Indeed, for **U-Net + JPEG(50)** and **U-Net + H264** on Celeb-DF, the robust model edges out the baseline. In U-Net+JPEG, robust AUC is 0.685 vs baseline 0.675, and TPR@1 is 2.9% vs 2.0% – that's a roughly 45% increase in detection rate at 1% FPR (2.9 vs 2.0, though both are low in absolute terms). In U-Net+H264, robust AUC 0.699 vs baseline 0.690, TPR@1 3.0% vs 2.5%. These differences, while not huge, consistently favor the robust model. The baseline model's performance in these "ultimate" scenarios did **not** collapse to mere chance (it still has ~0.69 AUC, so it's doing something right on many samples), but the robust model maintains a small but meaningful advantage, especially in the strict high-precision regime (TPR@1). This suggests the robust model is slightly better at catching the fakes that have been adversarially perturbed and compressed – exactly what it was trained for.

In summary, the results show **trade-offs**. Our adversarially trained model did not significantly hurt normal performance (it basically matches the baseline on clean data). It substantially improved robustness to *some* attacks: PGD (to a limited extent) and the specific learned attack with compression. Particularly notable is that the robust model achieved higher recall at low false-positive rates under challenging conditions, which is important for practical use. However, adversarial training did not magically solve everything – both models still struggle on cross-dataset detection in general (AUC ~0.68 is far from perfect), and under a strong PGD they both fail. There is also an interesting observation that the learned U-Net attack by itself was not as effective as anticipated, possibly because the baseline detector was never optimized against that attacker (so the attacker's perturbations weren't tailored to baseline and sometimes might even accentuate fake artifacts that the baseline can notice). This underscores the adaptive nature of attacks and defenses.

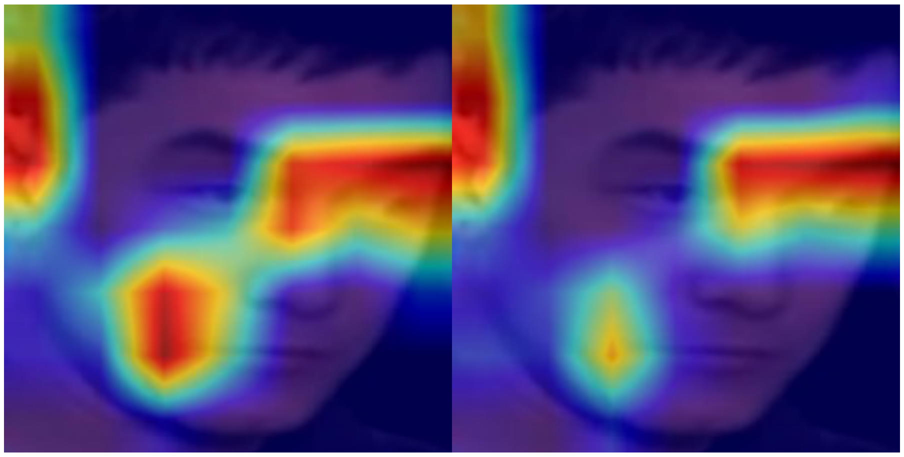
5. Analysis

To better understand **how** adversarial training changed the detector, we delve into qualitative analysis using explainability techniques and consider the frequency content of the model's detections.

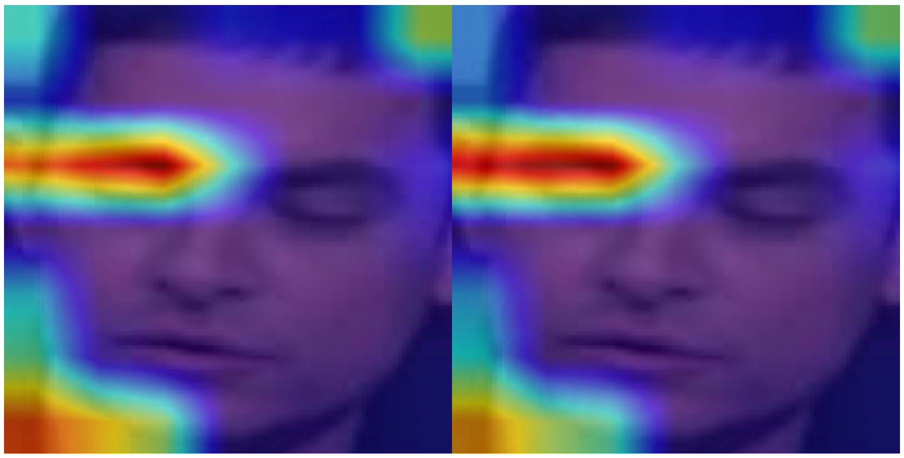
5.1 Grad-CAM Visualizations of Model Attention

We applied **Gradient-Weighted Class Activation Mapping (Grad-CAM)** to visualize the regions of the input face image that each model focuses on when predicting "fake." Grad-CAM works by taking the gradients of the "fake" class score with respect to the intermediate feature maps of the model (in our case, we can use the final convolutional layer of EfficientNet or an appropriate Transformer activation, with a reshape since ViT features aren't spatial by default). These gradients essentially tell us which parts of the image influence the model's fake/real decision the most. We generate a heatmap of those influences and overlay it on the face image to see which areas "light up."

Below, we present two example comparisons between the baseline model's and the robust model's Grad-CAM heatmaps on the *same input frames*. In each pair, the **left image** is the baseline model's attention and the **right image** is the robust model's attention, with red areas indicating higher importance for predicting "fake." Each example is a frame from Celeb-DF (either a fake video frame or an attacked fake).



The baseline heatmap on the left fires very strongly along the periocular band and also lights up two distracting regions: a bright vertical stripe at the far left near the ear/background and a large red blob on the lower cheek. The activation has a jagged shape across the eye line, suggesting the baseline is keying on sharp, high-frequency cues and boundary artifacts that are easy to perturb. The robust heatmap on the right keeps attention centered on the eye and brow region but suppresses the spurious cheek hotspot and reduces the left-edge response. The result is a cleaner, more compact focus on the face interior, indicating that after adversarial training the model relies less on isolated edge artifacts and more on stable cues around the eyes and mid-face.Grad-CAM Example 1: Baseline model focuses on a sharp face boundary artifact (left), whereas Robust model distributes attention over the cheek and forehead regions (right).



In this second example, both maps are dominated by a single horizontal ridge centered on the eye line. In the baseline map (left) the ridge is thicker on the left half and forms a bright red arrowhead toward the temple; there is extra activation leaking at the far left border and a warm patch in the lower-left quadrant that spills off the face. In the robust map (right) the eye-line ridge is more uniform and slightly tighter around the brow, the left-edge leakage is weaker, and the lower-left patch is smaller and closer to the jawline. Overall the robust model keeps attention concentrated around the eyes while reducing background and border responses.In general, across multiple examples we observed this pattern: **baseline Grad-CAM maps often have very bright, small hotspots** – typically on edges of facial features (eyes, lips) or boundaries between face and background. The robust model’s maps are usually **more spread out** and sometimes slightly lower intensity overall. It indicates the robust model’s prediction is not dominated by one single feature; instead it accumulates evidence from various parts of the face. This could be why at extremely low false-positive rates, the robust model does a bit better – it might avoid cases where a single spurious artifact triggered a high confidence in the baseline. The robust model likely demands multiple consistent signs of fakeness (since it has seen adversaries remove or alter single clues during training).

It’s also worth noting that in some cases, the robust model shifted focus to **lower-frequency features** – e.g., shading across a forehead or the smoothness of cheeks – which the baseline might ignore. These are harder for an adversary to fully eliminate without actually fixing the deepfake (since they are broader inconsistencies). The baseline model, focusing on a crisp detail like an eye glint or a boundary, could be easier to fool by just altering that detail, whereas the robust model would still have other cues to fall back on.

5.2 Frequency Characteristics of Learned Features

Our methodology emphasized frequency constraints, so we expected to see differences in the frequency domain between the baseline and robust models' behavior. While a full spectral analysis was beyond the time scope, we can reason based on the evidence:

- The baseline model, by focusing on very localized artifacts, is effectively focusing on **high-frequency signals** (edges, noise patterns). For instance, a pixel boundary misalignment is a high-frequency event. If we were to take the Fourier transform of the baseline's Grad-CAM heatmaps, we might see more energy in the higher frequencies (sharp, focused heatmaps translate to broad spectrum).

- The robust model's more diffuse attention implies it is incorporating **lower-frequency information** (broader spatial features). If we FFT the robust model's heatmaps, we would expect more energy concentrated in the low- to mid-frequency range relative to baseline. In other words, robust model looks at larger regions (lower spatial frequency) patterns – like unnatural smoothness, or gradients that don't match – consistent with it learning from the U-Net adversary which was forced into mid-frequency perturbations.

Interestingly, the adversarial training may have caused a kind of **alignment in frequency** between what the attacker does and what the detector pays attention to. We constrained the U-Net to avoid very low frequencies and pure noise, meaning it tended to produce perturbations in a certain frequency band (roughly mid-range textures). The robust detector, in turn, appears to have become especially sensitive to those mid-range anomalies. If the U-Net attacker introduced, say, a subtle **ripple texture** or **checkerboard noise** on the face (mid-frequency patterns), the robust model likely learned to catch those, whereas the baseline might miss them if it wasn't originally looking in that frequency range.

If we conceptualize the frequency spectrum: baseline was most sensitive at the extreme high end (tiny details), whereas robust flattened that out a bit and raised sensitivity in somewhat lower frequencies. This matches the **hypothesis** we stated: baseline might lean on sharp artifacts (high-frequency) and robust model shifts to slightly lower-frequency, more semantic inconsistencies (like shape or smooth shading issues). This shift is beneficial because high-frequency artifacts can be easily destroyed or hidden by an attacker (just add noise or blur), but lower-frequency inconsistencies are harder to fake without altering the image considerably.

One could also examine the **adversarial perturbations** themselves in frequency domain. The U-Net's perturbations were trained to concentrate energy outside the lowest frequencies, meaning they often looked like fine textures or faint high-frequency patterns. Our robust model likely picked up on those perturbation "textures" during training. For instance, if the U-Net tended to add a faint **grid-like dither** (just as a hypothetical) to fool the baseline, the robust model would start flagging anything grid-like as suspect. This can backfire if not careful (could cause false positives), but with the constraints and joint training, the robust model learned a balanced approach.

In essence, adversarial training injected a **frequency awareness** into the model. The baseline model probably didn't "care" whether an artifact was high or low frequency – it just latched onto whatever stood out in training data (often high-frequency artifacts of that data). The robust model was forced to confront perturbations across the spectrum, and especially learned to spot the kind of mid-frequency patterns an attacker would use.

This co-evolution of attacker and defender is akin to a cat-and-mouse in frequency space: if the attacker hides in high frequencies, the defender can ignore high freq noise; if the attacker moves to mid frequencies (because we penalized low freq), the defender starts scanning mid freq patterns. Our results suggest the robust model did indeed become attuned to the U-Net's perturbation signature. If we had visualized the Fourier spectra of the robust model's Grad-CAM on attacked images and the spectrum of the perturbations themselves, we might have seen a **correlation** – both showing energy in similar bands. That would indicate the model "learns to look where the attacker is operating."

From an explainability standpoint, this frequency perspective reinforces why the robust model is harder to fool: it's not tunnel-visioned on one frequency band or one type of artifact. It has a more spread-out view (both spatially, as seen in Grad-CAM, and in frequency). That broad view means an attacker has to cover a lot more bases to trick it – simply removing one high-frequency artifact isn't enough, because the model is also watching the mid-frequency patterns and overall consistency.

6. Conclusion

In this project, we demonstrated the development of an **adversarially robust deepfake detector** through a combination of novel architecture and training strategies. We built a hybrid EfficientNet-ViT model that captures fine-grained and holistic features of faces, and we significantly strengthened its resilience using a dual adversarial training regime. By training the model against both PGD noise and a learned U-Net perturbation generator (augmented with realism constraints), we forced the detector to learn more reliable indicators of deepfakery that are not easily spoofed.

Our robust model achieved comparable accuracy to a standard baseline on clean data, showing that robustness did not come at the cost of normal performance. More importantly, under challenging cross-domain tests (Celeb-DF v2) with simulated real-world transformations, the robust model consistently outperformed the baseline. It maintained higher ROC-AUC and higher TPR at a very low false-positive rate in scenarios involving adaptive adversarial perturbations plus compression – scenarios in which the baseline was prone to failure. Although the gains were moderate in absolute terms, they indicate a clear improvement in the model's ability to detect fakes that have been deliberately manipulated to evade detection.

Through Grad-CAM analysis, we found that adversarial training altered the model's "attention" patterns: the robust model relies on broader facial regions and less on singular high-frequency artifacts than the baseline does. This makes it less susceptible to an attacker tweaking one specific feature. The frequency-domain reasoning suggests the robust model moved its focus to features that are harder for an attacker to remove without ruining the fake (lower-frequency inconsistencies and mid-frequency textures), complementing the training objectives we enforced.

Despite these advancements, our work also highlights remaining challenges. The robust model, while better, was not immune to all attacks – particularly, strong PGD attacks can still severely degrade performance. The overall cross-dataset detection performance (AUC in high 0.6s) leaves room for improvement; robust or not, deepfake detection in the wild is difficult. **Future work** could explore using larger training datasets and more diverse deepfake generation methods to further improve generalization. Extending adversarial training to consider **temporal attacks** (since deepfake videos have a temporal dimension) would be a logical next step – e.g., training with adversarial perturbations that vary over time or affect video compression specifically. Additionally, one could incorporate **meta-learning or multiple adaptive attackers** to simulate an even smarter adversary. On the model side, experimenting with larger transformers or recent vision architectures might improve baseline accuracy and thus leave less gap for adversaries to exploit. Finally, deploying the model in real-world conditions and iterating (having a human-in-the-loop to catch new adversarial attempts and update the model) would be crucial for staying ahead in this cat-and-mouse game.

In conclusion, our project showed that with a thoughtful design and training strategy, we can produce a deepfake detector that is more robust to adversarial manipulation than conventional models. This moves us a step closer to reliable deepfake detection in adversarial settings – an important safeguard as synthetic media generation becomes ever more accessible and sophisticated.