BUILD → BREAK → IMPROVE

Navigating Synthetic Reality

Problem Statement 3 — Hackathon Feb 21–22

Team Members
Noor Fathima (22PD26)
Sarnika Sanjeev Kumar (22PD31)
Sujan S (22PD35)

Problem Statement

> *"A photo of a crime scene. A viral image of a public figure. A piece of evidence in court. What if none of them were real?"*

The rapid advancement of generative AI has produced highly realistic synthetic images, creating serious challenges in cybersecurity, misinformation detection, and digital forensics. While detection models can perform well in controlled settings, many are surprisingly vulnerable to small, targeted modifications — an active arms race between synthesis and detection.

Project Overview

This project follows a three-phase research-inspired cycle:

Phase	Name	Goal
1	Build	Train a CNN classifier to detect AI-generated images
2	Break	Use FGSM adversarial attacks to fool the detector
3	Improve	Retrain with adversarial examples to harden the model

A Streamlit web app ties everything together — letting users upload any image and see live predictions, Grad-CAM visualizations, and side-by-side model comparisons.

Project Structure

```
feb-21-22-hackathon/
├── data/sample_images/fake/   (fake_999.jpg, fake_999(7-9).jpg)
├── data/sample_images/real/   (real_0999.jpg, real_0999(7-9).jpg)
├── outputs/
│   ├── confusion_adv.png
│   ├── confusion_baseline.png
│   └── robustness_curve.png
├── src/
│   ├── attacks/    (evaluate_fgsm.py, fgsm.py, run_fgsm_experiment.py)
│   ├── data/       (datasets.py, explore.py, transforms.py)
│   ├── evaluation/ (compare_models.py, evaluate_adversarial.py, evaluate_baseline.py)
│   ├── explainability/ (gradcam.py, visualize_gradcam.py)
│   ├── models/     (spatial_baseline.py)
│   └── utils/      (data_utils.py, model_utils.py, adversarial_train.py)
├── train.py
├── app.py
├── requirements.txt
└── README.md
```

Dataset

CIFAKE – Real and AI-Generated Synthetic Images

Source: Kaggle — birdy654/cifake-real-and-ai-generated-synthetic-images

Split	FAKE	REAL	Total
Train	50,000	50,000	100,000
Test	10,000	10,000	20,000

- Image size: 32×32 RGB pixels
- Real images: From CIFAR-10 (photographs)
- Fake images: AI-generated using Stable Diffusion to match CIFAR-10 categories
- Why CIFAKE? Fast training (minutes per epoch), low memory, perfect for adversarial experiments

Preprocessing

All images normalized to [−1, 1] range:

```
transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
```

Phase 1: Build — Synthetic Image Detector

Model Architecture: SpatialBaselineCNN

```
Input: 3 × 32 × 32
Block 1: Conv2d(3→32, 3×3) + BatchNorm + ReLU + MaxPool2d  → 32×16×16
Block 2: Conv2d(32→64, 3×3) + BatchNorm + ReLU + MaxPool2d  → 64×8×8
Block 3: Conv2d(64→128, 3×3) + BatchNorm + ReLU + MaxPool2d → 128×4×4
Flatten → FC(2048→256) + ReLU + Dropout(0.5) → FC(256→2) → Logits
```

Design Choices

- BatchNorm after each conv layer for training stability
- MaxPooling halves spatial dimensions: 32→16→8→4
- Dropout (0.5) prevents overfitting; CrossEntropyLoss + Adam (lr=0.001)

Training Configuration

Parameter	Value
Epochs	5
Batch Size	64
Optimizer	Adam
Learning Rate	0.001
Loss Function	CrossEntropyLoss
Hardware	CPU (Kaggle)

Training Progress

Epoch	Train Loss	Train Acc	Val Loss	Val Acc
1	0.2500	89.85%	0.1862	92.73%
2	0.1795	93.14%	0.1569	93.90%
3	0.1593	94.00%	0.1711	93.44%
4	0.1437	94.60%	0.1381	94.87%
5	0.1287	95.22%	0.1502	94.28%

Evaluation on Test Set

Class	Precision	Recall	F1-Score	Support
FAKE	0.92	0.97	0.94	10,000
REAL	0.96	0.92	0.94	10,000
Overall	0.94	0.94	0.94	20,000

Clean Accuracy: 94.99%

Visual Explainability: Grad-CAM

Grad-CAM (Gradient-weighted Class Activation Mapping) visualizes which regions of an image drive the model's decision. In our baseline model, Grad-CAM reveals that the model focuses heavily on texture patterns and sky regions — zones that make it exploitable.

- Compute gradient of predicted class score w.r.t. final conv layer feature maps
- Average gradients across spatial dimensions → importance weights
- Weighted sum of feature maps → apply ReLU → overlay heatmap on image

Phase 2: Break — Adversarial Attacks (FGSM)

Attack Method: Fast Gradient Sign Method

FGSM generates adversarial perturbations by taking a small step in the gradient direction that maximizes loss.

```
x_adv = x + ε · sign(∇ₓ J(θ, x, y))

  x      = original image
  ε      = perturbation strength
  ∇ₓ J    = gradient of loss w.r.t. input
```

Iterative Evasion Results

Epsilon (ε)	Prediction	Confidence
0.00 (original)	FAKE	99.9999%
0.01	FAKE	~high
0.03	FAKE → borderline	dropping
0.05	REAL	~low
0.07	REAL	moderate
0.10	REAL	high

At ε = 0.04, the baseline model's prediction flips **FAKE → REAL** with a visually imperceptible perturbation.

Full Adversarial Robustness Evaluation

Epsilon (ε)	Baseline Accuracy	Notes
0.01	80.12%	Slight degradation
0.03	42.41%	Model nearly random
0.05	19.80%	Worse than random
0.07	11.89%	Almost completely fooled
0.10	8.93%	Model essentially fails

Root Cause Analysis

The baseline model relied on high-frequency texture artifacts from diffusion model outputs. FGSM directly attacked these gradients, injecting noise that disguised texture patterns. Without multi-scale or frequency-aware representations, tiny ε values (~0.03–0.05) were sufficient to destroy confidence.

- Original Grad-CAM: Focused on sky/background texture — diffuse heatmap across image
- Adversarial Grad-CAM: Attention map becomes nearly blank — perturbation suppressed spatial signals

Phase 3: Improve — Adversarial Training

Strategy: FGSM Adversarial Training

- Compute gradients from clean images
- Generate FGSM adversarial examples at $\varepsilon = 0.03$
- Compute loss on both clean and adversarial images
- Backpropagate average combined loss: (loss_clean + loss_adv) / 2

Robust Model Architecture

- No BatchNorm in conv layers (reduces gradient signal leakage)
- Dropout (0.5) moved to classifier head for better regularization
- Separated features and classifier blocks

Adversarial Training Progress

Epoch	Train Loss	Train Acc	Val Loss	Val Acc
1	0.4332	71.13%	0.2248	91.49%
2	0.3363	78.04%	0.1953	92.26%
3	0.3034	80.17%	0.1883	93.42%
4	0.2780	81.80%	0.1704	93.28%
5	0.2568	83.00%	0.1647	93.81%

Note: Higher train loss is expected — the model simultaneously handles clean and perturbed inputs.

Results & Metrics

Clean Accuracy Comparison

Model	Clean Accuracy
Baseline	94.99%
Robust (Adversarially Trained)	93.66%

The robust model trades ~1.3% clean accuracy for dramatically improved adversarial resilience.

Adversarial Accuracy Comparison

Epsilon (ε)	Baseline	Robust Model	Improvement
0.01	82.08%	90.21%	+8.13%
0.03	43.92%	80.55%	+36.63%
0.05	21.27%	68.54%	+47.27%
0.07	13.34%	55.77%	+42.43%
0.10	9.58%	37.84%	+28.26%

Confusion Matrices

Baseline Model

	Predicted FAKE	Predicted REAL
Actual FAKE	9,636 ✅	364
Actual REAL	638	9,362 ✅

Robust Model

	Predicted FAKE	Predicted REAL
Actual FAKE	9,403 ✅	597
Actual REAL	670	9,330 ✅

App Screenshots & Outputs

The following screenshots demonstrate the Streamlit interactive demo in action, showing all four sections of the application.

Dataset Samples View

The app displays FAKE and REAL sample images side-by-side in a 4-column grid, loaded from data/sample_images/.

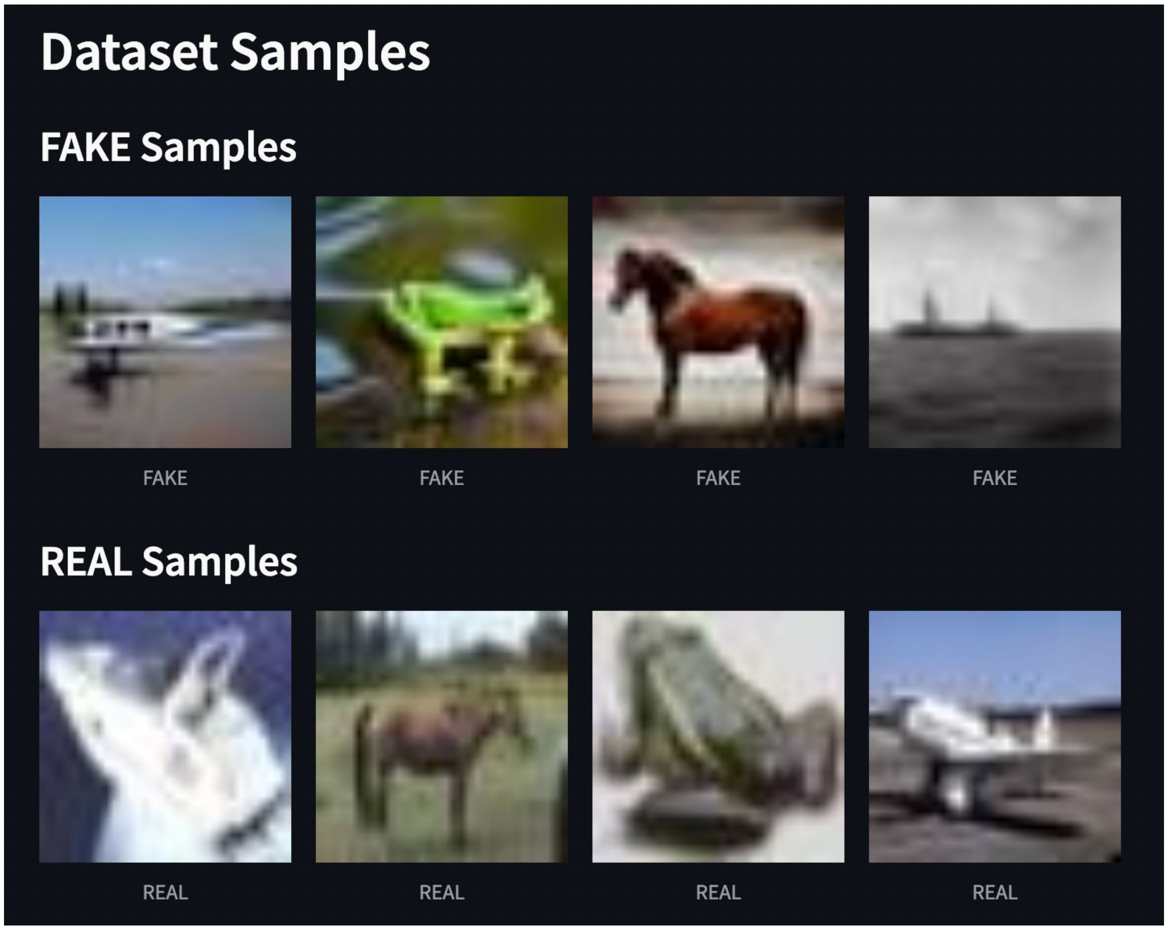
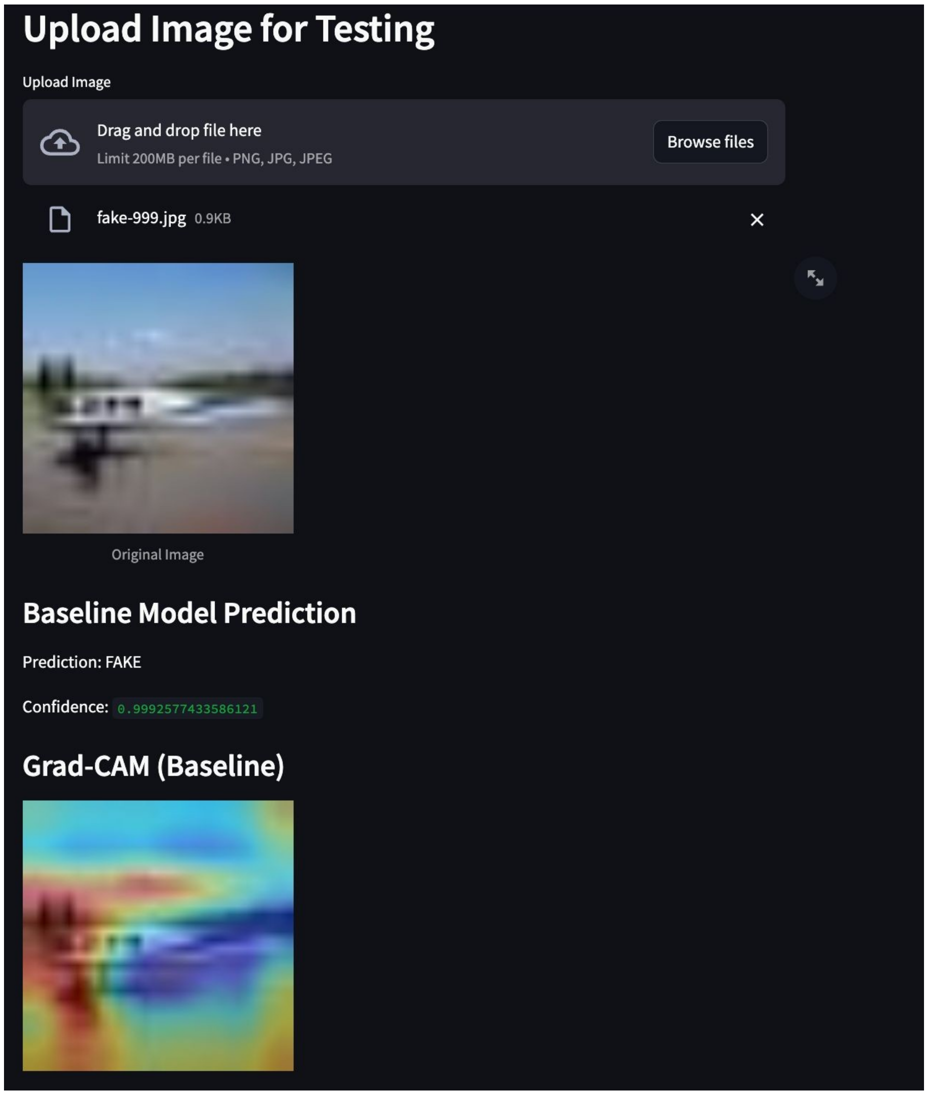


Figure 1: Dataset Samples — 4 FAKE images (AI-generated) and 4 REAL images (CIFAR-10) shown side-by-side

Upload & Baseline Prediction with Grad-CAM

User uploads fake-999.jpg (an airplane image). The Baseline Model correctly classifies it as FAKE with 99.99% confidence. The Grad-CAM heatmap shows the model's attention focused on the upper sky/texture region.



Upload Image for Testing

Upload Image

Drag and drop file here

Limit 200MB per file • PNG, JPG, JPEG

Browse files

fake-999.jpg 0.9KB

Original Image

Baseline Model Prediction

Prediction: FAKE

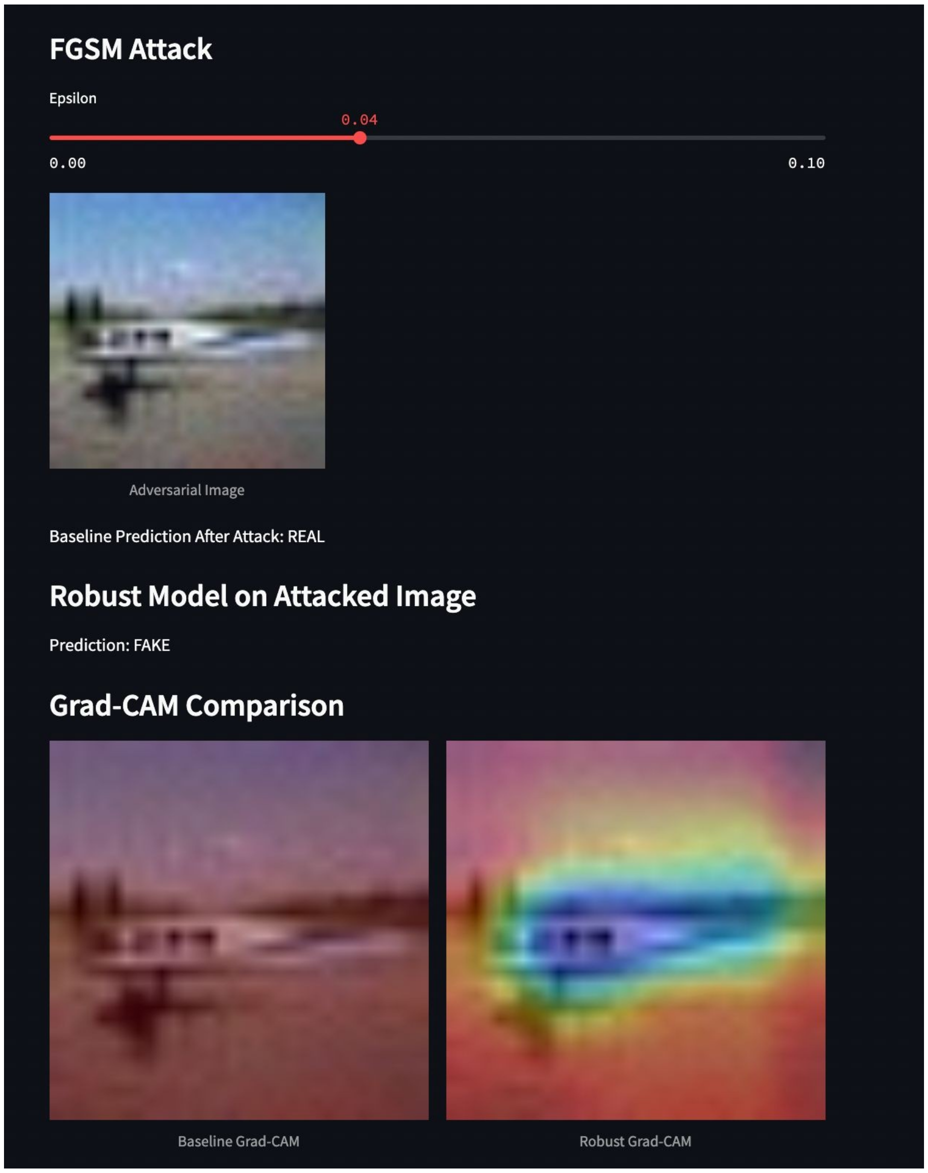
Confidence: `0.9992577433586121`

Grad-CAM (Baseline)

Figure 2: Upload Image section — Baseline predicts FAKE (confidence: 0.9992), Grad-CAM highlights texture focus regions

FGSM Attack in Action

With epsilon set to 0.04 via the slider, the Baseline Model is fooled into predicting REAL. The Robust Model correctly maintains FAKE. The Grad-CAM comparison shows: Baseline loses all attention structure after attack, while the Robust Model retains meaningful focus on the airplane body.



FGSM Attack

Epsilon

0.04

0.00 0.10

Baseline Prediction After Attack: REAL

Robust Model on Attacked Image

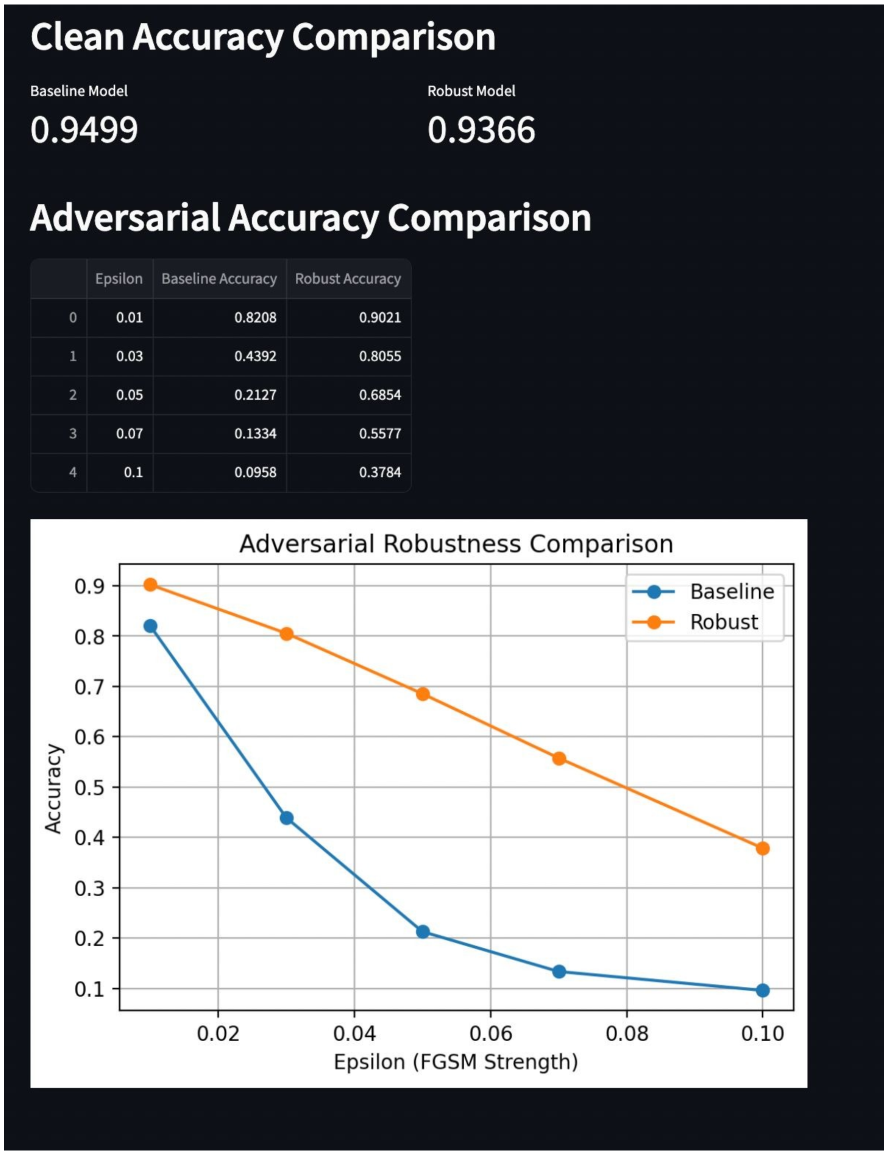
Prediction: FAKE

Grad-CAM Comparison

Figure 3: FGSM Attack at ε=0.04 — Baseline flips to REAL, Robust stays FAKE. Grad-CAM comparison shows attention shift

Clean vs Adversarial Accuracy Comparison

The app displays precomputed evaluation metrics: clean accuracy (Baseline: 0.9499 vs Robust: 0.9366), adversarial accuracy table across all epsilon values, and the robustness comparison chart.



Clean Accuracy Comparison

Baseline Model

0.9499

Robust Model

0.9366

Adversarial Accuracy Comparison

	Epsilon	Baseline Accuracy	Robust Accuracy
0	0.01	0.8208	0.9021
1	0.03	0.4392	0.8055
2	0.05	0.2127	0.6854
3	0.07	0.1334	0.5577
4	0.1	0.0958	0.3784

Figure 4: Accuracy Comparison — Robust model consistently outperforms baseline at every epsilon value tested

Model Metrics & Confusion Matrices

Side-by-side confusion matrices for both models with the Adversarial Robustness Curve showing the sharp degradation of the Baseline vs gradual decline of the Robust model.

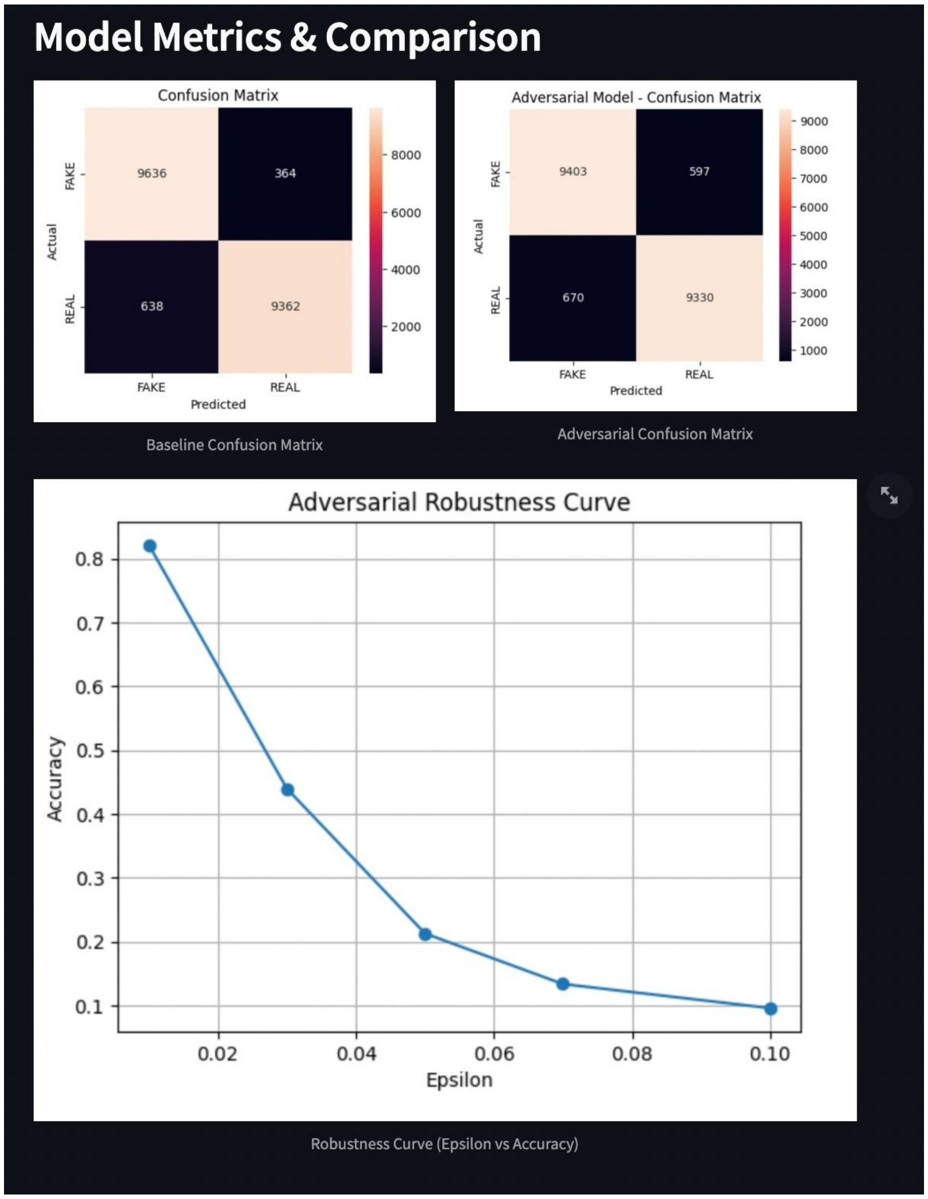


Figure 5: Model Metrics — Confusion matrices (Baseline: 9636/9362 correct, Robust: 9403/9330 correct) and Robustness Curve

Interactive Demo (Streamlit App)

Running the App

```
streamlit run app.py
```

Both baseline_model.pth and adversarial_model.pth must be present in models/. Models are cached with @st.cache_resource.

App Sections

1. Dataset Samples

Loads images from data/sample_images/fake/ and data/sample_images/real/ and displays in a 4-column grid.

2. Upload Image for Testing

- Upload PNG/JPG/JPEG → resized to 32×32, normalized to [−1, 1]
- Shows: predicted class (FAKE/REAL) + raw confidence score
- Grad-CAM heatmap using model.conv3 with cv2.COLORMAP_JET (60/40 blend)

3. FGSM Attack

- Epsilon slider (0.00 → 0.10, default 0.03) controls perturbation strength
- Side-by-side Grad-CAM comparison: Baseline vs Robust on attacked image

4. Model Metrics & Comparison

- Clean accuracy metrics, adversarial accuracy dataframe, robustness curve chart
- Pre-saved confusion matrix PNGs loaded from outputs/

How to Run

Prerequisites

```
pip install -r requirements.txt
torch  torchvision  streamlit  matplotlib  seaborn  scikit-learn
numpy  opencv-python  tqdm  pandas  pillow
```

Option 1: Kaggle (Recommended for Training)

- Upload notebook → add CIFAKE dataset → enable GPU → run all cells
- Models save to models/baseline_model.pth and models/adversarial_model.pth

Option 2: Run Locally

```
git clone https://github.com/NoorFathima14/feb-21-22-hackathon.git
cd feb-21-22-hackathon
pip install -r requirements.txt
python train.py          # Train baseline model
streamlit run app.py     # Launch Streamlit app
```

Expected Compute Times

Task	Time (CPU)	Time (GPU)
1 epoch baseline training	~6 min	~45 sec
5 epoch baseline training	~25 min	~4 min
Full adversarial evaluation (per ε)	~6 min	~1 min
Adversarial training (5 epochs)	~30 min	~5 min

Key Findings & Analysis

What the Baseline Model Learned

- High-frequency texture artifacts specific to diffusion model outputs
- Background/sky regions rather than semantic object features
- Global color distribution patterns — statistical shortcuts, not semantic understanding

Why FGSM Was So Effective

- Gradient-based: directly exploits the model's own decision boundary gradients
- High-frequency targeting: injects noise overlapping the exact frequency bands relied on
- At ε=0.04, a ~1/255 pixel shift in normalized space flipped predictions

Why Adversarial Training Helped

- Model learned to classify correctly even when texture artifacts are perturbed
- Forced to find more robust, distributed features across the image
- Grad-CAM shows broader, semantically meaningful attention on object structure

Limitations & Future Work

Limitation	Proposed Solution
Only FGSM attacks tested	Add PGD, C&W, AutoAttack
Single epsilon for adversarial training	Multi-epsilon curriculum training
Spatial-only model	Add frequency-domain branch (DCT features)
32×32 images only	Test on higher resolution synthetic images
No data augmentation	Add random crops, flips, color jitter
FGSM is white-box	Evaluate on black-box transfer attacks

Tech Stack

Component	Technology
Deep Learning	PyTorch
Computer Vision	torchvision, OpenCV (cv2)
Data Processing	NumPy, Pandas, PIL

Component	Technology
Visualization	Matplotlib, Seaborn
Explainability	Grad-CAM (hook-based, custom implementation)
Metrics	scikit-learn
Web App	Streamlit
Training Platform	Kaggle (CPU)
Dataset	CIFAKE (Kaggle)