Build → Break → Improve: Navigating Synthetic Reality

Challenge Overview

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.

Your task follows a powerful research-inspired cycle:

1. **Build** a synthetic image detector from scratch.
2. **Break** it by crafting adversarial modifications that cause your detector to misclassify fake images as real.
3. **Improve** it by proposing (and optionally prototyping) a concrete defense based on the weaknesses you uncover.

Key Requirements

Phase 1: Build & Evaluate the Synthetic Image Detector (Build)

- Design and train a classifier for synthetic image detection — you may build a custom CNN from scratch **or** use a pretrained model (e.g., ResNet, EfficientNet) with transfer learning fine-tuned on the provided dataset. Both approaches are valid; choose what you can execute well within the time limit.
- Train on the provided dataset to distinguish real images from AI-generated/synthetic ones.
- Evaluate on validation/test splits using: Accuracy, Precision, Recall, F1-score, and Confusion Matrix.
- Include visual explainability: Grad-CAM, saliency maps, or similar techniques showing which image regions influence your model's decisions.

Phase 2: Adversarial Modification / Evasion Exploration (Break)

- Select synthetic test images that your detector correctly classifies as fake with high confidence.
- Apply modifications to fool your detector — the goal is to flip its prediction to "real" or significantly drop its confidence score.
- Demonstrate the process iteratively: **original → modified versions → final evading image**, showing confidence scores at each step.
- Some directions worth thinking about: pixel-level perturbations, frequency-domain manipulation, filtering operations, or latent-space edits — but don't let this list limit you.
- Prioritize minimal changes — the best evasions are ones where the image still looks realistic to a human but fools your model.
- Analyze *why* the evasion worked — what did your modification suppress or shift? This analysis matters more than the attack itself.

Phase 3: Robustness Improvement Proposal & Prototype (Improve)

- By analyzing your successful evasions, identify *what* your detector was actually relying on — was it texture patterns, high-frequency artifacts, color statistics, or something else entirely? Use your Grad-CAM or saliency maps from Phase 1 alongside your evasion results to build this picture.
- This analysis matters more than the attack itself — a well-reasoned understanding of your model's blind spots is the foundation of everything in this phase.
- From this understanding, propose a concrete improvement — how would you redesign or retrain your detector to be robust against the weaknesses you uncovered? Your proposal should directly follow from your findings, not be a generic suggestion.

Output Deliverables

- Working prototype (Python notebook strongly preferred; optional simple interface).
- Well-commented code covering all three phases.
- Detector performance report (metrics + visual explanations).
- Evasion demonstration: before/after images, confidence trajectories, analysis of successful modifications.
- Propose **one specific improvement** inspired by your evasion results.
- Provide a clear rationale linking the proposal to the vulnerabilities you discovered.
- If time allows, implement a quick proof-of-concept (e.g., retrain with a small set of adversarial examples or add a defense layer) and show re-evaluation results on your evaded images.

Constraints & Recommendations

- **Compute:** Limited to free Kaggle or Google Colab (GPU/TPU sessions). Use small batch sizes, mixed precision if needed. Save model params frequently.
- **Dataset Recommendation (Primary):**
  CIFAKE – Real and AI-Generated Synthetic Images
  https://www.kaggle.com/datasets/birdy654/cifake-real-and-ai-generated-synthetic-images
  → 120,000 images (60k real CIFAR-10 + 60k synthetic), 32×32 RGB pixels. Extremely fast training (minutes per epoch), low memory usage (~few GB), perfect for from-scratch training and evasion experiments.
- Use a subset of data if compute or memory is a bottleneck.
- Focus on explainability, minimal modifications, and insightful analysis over raw high accuracy.

This challenge combines computer vision fundamentals and adversarial machine learning. It rewards teams that build thoughtfully, break creatively, and improve meaningfully — mirroring how real-world AI security tools evolve.

Submission: Code/notebook(s), report (PDF), short demo video (~5 minutes) by the deadline. Good luck!