**Project Proposal: Exploring the Adversarial Robustness of Speech-Command Recognition**

Team:

Nikhitha Kilari  
Jishnuvardhan Karpuram

**Overview**

We will evaluate how small, imperceptible perturbations can fool a 1D-CNN speech classifier and which lightweight defenses can mitigate those attacks. To keep the pipeline reproducible and drive‐friendly, we substitute the originally planned Google Speech Commands v2 dataset (>1 GB) with the smaller, publicly available **Free Spoken Digit Dataset (FSDD)** (~50 MB, 10 digit classes).

**Dataset & Preprocessing**

* **Free Spoken Digit Dataset (FSDD):** 10 classes (0–9), ~1 500 recordings. Downloaded automatically via the Kaggle CLI into Google Drive.
* **Preprocessing:**
  1. Resample to 8 kHz (torchaudio).
  2. Trim or pad to exactly 1 second (8 000 samples).
  3. Normalize amplitude to [-1, 1] and cast to float32.
* **Split:** 80/20 train/test using a fixed random seed for reproducibility.

**Baseline Model**

* **Conv1DSpeech:** A compact 3-layer 1D-CNN (Conv → BN → ReLU) × 3, followed by global average pooling and a 10-way linear classifier.
* **Clean Accuracy Goal:** ~ 85–88% on the FSDD test set.

**Adversarial Attacks**

We will implement six novel black-box and psychoacoustic attacks in pure PyTorch/NumPy no external ART dependencies running on small batches (size=8) to avoid OOMs:

1. **SPSA Attack** (gradient-approximation via finite differences)
2. **GenAttack** (evolutionary algorithm, gradient-free optimization)
3. **SimBA Audio** (iterative single-sample perturbations)
4. **Spatial Transform** (pitch-shifting & time-stretching with librosa)
5. **Psychoacoustic Masking** (Gaussian noise at target SNR, 30 dB)
6. **Hidden-Voice** (mixing a low-amplitude digit phrase into the waveform)

For each:

* Tune the attack budget (iterations, ε, population size, transformation magnitude).
* Measure top-1 accuracy drop, mean signal-to-noise ratio (SNR), and use SNR as a proxy for perceptual stealth.
* Visualize accuracy vs. SNR to illustrate the stealth–success trade-off.

**Defense Strategies**

We will evaluate three lightweight defenses on the same test split:

* **Randomized Smoothing:** Add Gaussian noise at inference (σ = 0.001, 0.002, 0.004), majority-vote over 10 samples.
* **Feature Squeezing:** Quantize audio bit-depth to 2, 4, and 8 bits to remove high-frequency artifacts.
* **Defensive Distillation:** Train a student network on teacher “soft” labels at temperature T = 20 for 5 epochs.

For each defense, we will report both **clean** and **robust** accuracy, then present a summary bar chart comparing all three methods.

**Deliverables & Timeline**

* **Notebooks:**
  1. download\_fsdd.ipynb – automated Kaggle download
  2. 1\_data\_preprocessing.ipynb – audio loading & pipeline demo
  3. 2\_baseline\_model.ipynb – training Conv1DSpeechBig baseline
  4. 3\_attacks.ipynb – six attack implementations & stealth evaluation
  5. 4\_defenses.ipynb – three defense evaluations & summary plots
* **Code modules:** models.py, audio\_utils.py, attack/defense helper scripts
* **Final report:** PDF & Word summarizing methods, results, and recommendations
* **GitHub repository** containing all code, notebooks, data pointers, and the report

**Milestones:**

* Week 1–2: Data download, baseline training, clean accuracy
* Week 3: Implement & tune six adversarial attacks
* Week 4: Implement three defenses & gather results
* Week 5: Write report, polish notebooks, prepare GitHub submission

**Impact:**

By focusing on a smaller, manageable dataset and fully self-contained code, we ensure that our methodology can be reproduced in limited‐resource environments while still yielding actionable insights into audio adversarial robustness.