# USENIX Security ’24 Artifact Appendix: <SoK: The Good, The Bad, and The Unbalanced: Measuring Structural Limitations of Current Deepfake Media Datasets>

Seth Layton, Tyler Tucker, Daniel Olszewski, Kevin Warren, Kevin Butler, and Patrick Traynor
University of Florida

## A Artifact Appendix

### A.1 Abstract

Deepfake media represents an important and growing threat not only to computing systems but to society at large. Datasets of image, video, and voice deepfakes are being created to assist researchers in building strong defenses against these emerging threats. However, despite the growing number of datasets and the relative diversity of their samples, little guidance exists to help researchers select datasets and then meaningfully contrast their results against prior efforts. To assist in this process, this paper presents the first systematization of deepfake media. Using traditional anomaly detection datasets as a baseline, we characterize the metrics, generation techniques, and class distributions of existing datasets. Through this process, we discover significant problems impacting the comparability of systems using these datasets, including unaccounted-for heavy class imbalance and reliance upon limited metrics. These observations have a potentially profound impact should such systems be transitioned to practice - as an example, we demonstrate that the widely-viewed best detector applied to a typical call center scenario would result in only 1 out of 333 flagged results being a true positive. To improve reproducibility and future comparisons, we provide a template for reporting results in this space and advocate for the release of model score files such that a wider range of statistics can easily be found and/or calculated. To support this, we release every model we train, the data we use to train, and the source code of the entire project. Furthermore, we release our scores files along with the code used to produce the figures and tables in the paper. Our artifacts include precise steps to train, evaluate, and reproduce our results.

### A.2 Description & Requirements

We describe all of the information, data, and requirements needed to recreate our experiments.

#### A.2.1 Security, privacy, and ethical concerns

There are no security, privacy, or ethical concerns for the recreation of the work in this artifact.

#### A.2.2 How to access

Our source code is available via GitHub[1] and the data we use is available via Zenodo[2][3][4]. Our GitHub links to the Zenodo and vice versa. Due to file size limitations, we separate our datasets into three different Zenodos that all link to each other. The GitHub repository contains data download scripts and as such manually downloading files from Zenodo is not required, but provided for completeness.

#### A.2.3 Hardware dependencies

All training and evaluation scripts require a GPU for computation. We use 2x Nvidia GeForce RTX 2080 for all models except wav2vec, which uses 2x Nvidia DGX A100. The wav2vec models require substantially more GPU memory, thus requiring higher-performance GPUs. During training and inference, we allocate 50GB of RAM and 4 AMD EPYC 7742 2.25GHz CPUs to the process. While we test with the specific hardware listed, any suitable GPU and CPU combination will be sufficient.

However, it is worth noting that if simply recreating the figures/tables with our provided scores files, there are no GPU/CPU/memory requirements.

#### A.2.4 Software dependencies

Our source code has been verified to work on Ubuntu 22.04.4 LTS and Red Hat Enterprise Linux 8.9 (Ootpa). Our code requires Python 3.7 and 3.8 (these are specifically defined on the GitHub page). However, the operating systems are not limited and any modern operating system that supports Python should work for artifact evaluation.

[1] https://github.com/SethLayton/SoKTheGoodTheBadandTheUnbalanced/tree/753d89beb64929371f7460ead16c770888c4ae4b

[2] https://zenodo.org/records/12090252

[3] https://zenodo.org/records/12089727

[4] https://zenodo.org/records/12007844

### A.2.5 Benchmarks

The artifacts described herein depend on the ASVspoof, TIMIT, LJSpeech, WorldEnglishBible, CFAD, WenetSpeech, CIFAKE, CIFAR, and STL10 datasets. Each dataset is attributed in the paper, and linked to in the respective Zenodo repository. Additionally, the pre-trained wav2vec-XLS-R2-300m is used for the wav2vec model in the ASVspoof experiments.

Instructions to download all datasets and models are provided in the GitHub README.

## A.3 Setup

The most important setup piece is the decision of reproduction complexity. We facilitate three tiers of complexity when reproducing our artifacts.

**Maximum Complexity:** This entails retraining all models from source data, evaluating test data using these retrained models, and producing all figures and tables from this.
**Medium Complexity:** This entails evaluating test data using provided pretrained models, and producing all figures and tables from this.
**Minimum Complexity:** This entails producing all figures and tables from the provided scores files.

Maximum complexity requires upwards of a week of computation time, medium complexity requires upwards of half a day of computation time, and minimum complexity takes upwards of an hour of computation time.

### A.3.1 Installation

Complete installation steps are provided via the GitHub README file. Additionally, the GitHub repo provides download scripts for all required data for each complexity.
Stated concisely:

1. Clone the repository.
2. Decide reproduction complexity.
3. Create the conda environments.
4. Download the required data for the selected complexity.

### A.3.2 Basic Test

The simplest functionality test is to follow the steps in *STEP 3.3 – Generate Figures/Tables* in the GitHub README. This will produce the figures and tables for the paper using the provided scores files and this takes minimal time.

## A.4 Evaluation workflow

### A.4.1 Major Claims

**(C1):** Performance results from baseline detection models on deepfake datasets are not reasonably reproducible. This is proven by the experiment (E1) described in Section 3.2.1 Reproducibility whose results are illustrated/reported in Section 3.2.1 and Table 3.

**(C2):** Limited, and often singular, metrics as the sole performance measure do not sufficiently represent the behavior of deepfake detection models. This is proven by the experiments (E2) described in Section 3.2.2 Efficacy of Reported Metrics whose results are illustrated/reported in Section 3.2.2, Table 3, and Table 9 (companion website).

**(C3):** Composition of current deepfake datasets imposes bias on classification results. This is proven by the experiments (E3 and E4) described in Section 3.2.3 Class Distribution in Datasets whose results are illustrated/reported in Section 3.2.3, Figure 2, Figure 10 (companion website), Figure 11 (companion website), Figure 12 (companion website), Figures 8/9 (companion website), and Table 4 (companion website).

**(C4):** Current deepfake datasets perform poorly in a base-rate-aware environment. This is proven by the experiments (E4) described in Section 3.2.4 Characterizing Model Efficacy whose results are illustrated/reported in Section 3.2.4, Figure 3, Figure 4, Figure 13 (companion website), Table 3, and Table 4 (companion website).

### A.4.2 Experiments

Our experiments are not separated in any experimental capacity. We provide a single script to produce all figures and tables from the paper and companion website. Our division is defined by the complexity of reproduction as described above.

**Maximum Complexity** [1 human-hour + 1 compute-week]

**(E1):** [Reproducibility]: Produce performance metrics and compare with reported metrics.
**How to:** Follow steps 1.1, 2.1, and 3.1 in the GitHub README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Table 3 is reproduced.

**(E2):** [Performance Metrics]: Produce dataset-suggested performance metrics and compare with additional performance metrics.
**How to:** Follow steps 1.1, 2.1, and 3.1 in the GitHub README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Table 3, and Table 9 (companion website) are reproduced.

**(E3):** [Dataset Composition]: Produce TP,FP,TN,FN performance metrics for different training class distributions and evaluate against different test dataset class distributions.
**How to:** Follow steps 1.1, 2.1, and 3.1 in the GitHub

README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Figure 2, Figure 10 (companion website), Figure 11 (companion website), Figure 12 (companion website), Figures 8/9 (companion website), and Table 4 (companion website) are reproduced.

**(E4):** [Base-Rate Environment Evaluation]: Evaluate model performance metrics in a base-rate-aware environment for different training class distributions.
**How to:** Follow steps 1.1, 2.1, and 3.1 in the GitHub README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Figure 3, Figure 4, Figure 13 (companion website), Table 3, and Table 4 (companion website) are reproduced.

**Medium Complexity** [1 human-hour + 0.5 compute-days]

**(E1):** [Reproducibility]: Produce performance metrics and compare with reported metrics.
**How to:** Follow steps 2.2 and 3.2 in the GitHub README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Table 3 is reproduced.

**(E2):** [Performance Metrics]: Produce dataset-suggested performance metrics and compare with additional performance metrics.
**How to:** Follow steps 2.2 and 3.2 in the GitHub README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Table 3, and Table 9 (companion website) are reproduced.

**(E3):** [Dataset Composition]: Produce TP,FP,TN,FN performance metrics for different training class distributions and evaluate against different test dataset class distributions.
**How to:** Follow steps 2.2 and 3.2 in the GitHub README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Figure 2, Figure 10 (companion website), Figure 11 (companion website), Figure 12 (companion website), Figures 8/9 (companion website), and Table 4 (companion website) are reproduced.

**(E4):** [Base-Rate Environment Evaluation]: Evaluate model performance metrics in a base-rate-aware environment for different training class distributions.
**How to:** Follow steps 2.2 and 3.2 in the GitHub README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Figure 3, Figure 4, Figure 13 (companion website), Table 3, and Table 4 (companion website) are reproduced.

**Minimum Complexity** [1 human-hour + 1 compute-hour]

**(E1):** [Reproducibility]: Produce performance metrics and compare with reported metrics.
**How to:** Follow step 3.3 in the GitHub README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Table 3 is reproduced.

**(E2):** [Performance Metrics]: Produce dataset-suggested performance metrics and compare with additional performance metrics.
**How to:** Follow step 3.3 in the GitHub README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Table 3, and Table 9 (companion website) are reproduced.

**(E3):** [Dataset Composition]: Produce TP,FP,TN,FN performance metrics for different training class distributions and evaluate against different test dataset class distributions.
**How to:** Follow step 3.3 in the GitHub README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Figure 2, Figure 10 (companion website), Figure 11 (companion website), Figure 12 (companion website), Figures 8/9 (companion website), and Table 4 (companion website) are reproduced.

**(E4):** [Base-Rate Environment Evaluation]: Evaluate model performance metrics in a base-rate-aware environment for different training class distributions.
**How to:** Follow step 3.3 in the GitHub README.
**Execution:** Run the scripts defined in the aforementioned GitHub README sections.
**Results:** Figure 3, Figure 4, Figure 13 (companion website), Table 3, and Table 4 (companion website) are reproduced.

## A.5 Notes on Reusability

All of the model architectures we use in our experiments are direct replications of the source material. Thus, following the procedures defined in the source material it is possible to train and evaluate any of these models on any number of datasets. This is evident in the fact that our experiments use new/unseen data through training and testing phases for every model.

## A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.