USENIX'23 Artifact Appendix: Unique Identification of 50,000+ Virtual Reality Users from Head & Hand Motion Data

Vivek Nair
UC Berkeley

Wenbo Guo
UC Berkeley

Justus Mattern
RWTH Aachen

Rui Wang
UC Berkeley

James F. O'Brien
UC Berkeley

Louis Rosenberg
Unanimous AI

Dawn Song
UC Berkeley

A Artifact Appendix

Metaverse Research @ Berkeley RDI

Unique Identification of 50,000+ Virtual Reality Users from Head & Hand Motion Data
rdi.berkeley.edu/metaverse/identification

A.1 Abstract

We present source code and data that can be used to replicate our result of uniquely identifying over 55,000 users based on head and hand motion data in VR. Our source code includes Python scripts for training and testing a LightGBM-based identification model using our novel hierarchical approach. Our dataset includes both 4.7 TB of raw motion data from over 100,000 VR users, as well as about 20 GB of preprocessed features using our described featurization approach. Together, the scripts and data can be used to reproduce the main experiments, results, and figures presented in our paper.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our artifact is non-destructive, but requires careful ethical consideration as it involves real user data from over 55,000 users. We have taken steps to anonymize these users before publishing our dataset. Evaluators should be careful to respect the privacy of these users by not attempting to deanonymize or contact these users, to infer sensitive attributes, or to otherwise violate our ethical data use guidelines available at https://rdi.berkeley.edu/metaverse/boxrr-23/dua.pdf.

A.2.2 How to access

The source code and processed features are available on Zenodo at https://zenodo.org/record/7935034.

The raw data from over 100,000 users can be found at https://rdi.berkeley.edu/metaverse/boxrr-23/. It cannot be uploaded to a standard repository due to its size (4.7 TB).

A.2.3 Hardware dependencies

The training and testing code for our machine learning models can be run on any machine or cluster with at least 96 GB of RAM per node. We performed our main evaluation using a distributed machine learning cluster of 10 nodes, each with 16 vCPU cores and 128 GB of RAM. LightGBM also supports CUDA acceleration, and performs slightly better with a GPU. On a single machine with a RTX 3090 GPU, Ryzen 9 5950X CPU, and 128 GB of RAM, the execution time was as follows:

- Training (Layer 1): 1d 7h 38m 43s
- Training (Layer 2): 1d 9h 16m 27s
- Clustering: 4d 16h 42m 49s
- Training (Layer 3): 1h 7m 26s
- Testing: 4d 16h 43m 37s

Finally, we recommend at least 128 GB of free disk space. Alternatively, the subset evaluation (E5) can be run with as little as 8 GB of RAM and 4 GB of disk.

A.2.4 Software dependencies

A wide variety of operating systems are supported; our main cluster used Ubuntu 20.04, while the single-note benchmarking machine used Windows 10. Our scripts were designed for Python 3.10.2 with the following required Python packages:

- PyTorch (torch) v1.13.1
- pandas v1.5.2
- tqdm v4.64.1
- scikit-learn (sklearn) v1.2.0
- NumPy (numpy) v1.24.0
- LightGBM (lightgbm) v3.3.3
- Joblib (joblib) v1.2.0
- NetworkX (networkx) v3.0
- Matplotlib (matplotlib) v3.6.2

A.2.5 Benchmarks

Our evaluation is based on 3.96 TB of raw motion-tracking data from over 2.65 million recordings of over 55,000 virtual reality users. The dataset has since grown in size to 4.71 TB and now includes over 100,000 users, and can be accessed at https://rdi.berkeley.edu/metaverse/boxrr-23/.

To help evaluators reproduce our results without downloading the entire 4.71 TB dataset, we have also provided the pre-processed engineered features according to the featurization technique described in our papers. These features are provided in our Zenodo archive as four separate PyTorch (.pt) data files:

- train.pt (14.4 GB): 55540 Users, 150 Events per User
- validate.pt (0.5 GB): 55540 Users, 5 Events per User
- cluster.pt (4.8 GB): 55540 Users, 50 Events per User
- test.pt (4.8 GB): 55540 Users, 50 Events per User

Additionally, we have provided our trained model files and pre-computed identification results at https://boxrr-23.mfkdf.com/?identification-supplemental=1. These files cannot be uploaded to Zenodo due to their size (90 GB).

A.3 Set-up

A.3.1 Installation

1. Download and install version 3.10.2 of Python from https://www.python.org/downloads/.
2. Use `pip install` to download the suggested version of each of the packages listed in §A.2.4 above.
3. Download the source code and data from Zenodo at https://zenodo.org/record/7935034.
4. Optionally, follow the instructions to set up GPU acceleration for LightGBM: https://lightgbm.readthedocs.io/en/latest/GPU-Tutorial.html.
   (a) If you prefer to use CPU, change `device_type='gpu'` to `device_type='cpu'` in all three training scripts (scripts 1, 2, and 5).
   (b) Change `n_jobs=16` to `n_jobs=N` where N is the number of physical CPU cores in your machine.
5. Optionally, download the supplemental data (trained models and inference results) at https://boxrr-23.mfkdf.com/?identification-supplemental=1.
6. Optionally, download the subset data (scripts and datasets) at https://zenodo.org/record/8137817.

A.3.2 Basic Test

After installing the required software, unzip the supplemental data into the same folder as the source code and data from Zenodo, then run this command: `py 7-stats_final.py`

If everything is correctly setup, the script should run without errors, and will calculate the identification accuracy of around 95%. The estimated run time is around 2 minutes.

A.4 Evaluation workflow

Note: A full replication of our main result (C1) requires over 12 days of computing time on a high-end workstation PC. While this is certainly an option (E1), we have also provided an alternative (E2) that allows evaluators to reproduce each step of our training and testing pipeline within a day without completely re-computing every intermediate result for all 55,540 users. Either way, evaluators will be able to efficiently conduct E3 and E4 to validate claims C2 and C3. A third option (E5) allows evaluators with limited computational resources to validate C1, C2, and C3 on a representative subset of the data, which can be extrapolated to the full 55,540 users.

A.4.1 Major Claims

(C1): Our system can uniquely identify over 50,000 VR users with over 90% accuracy from head and hand motion.
(C2): Over 50% of the information used to identify users comes from actual motion rather than static features.
(C3): Our system can correctly classify whether a given user has previously been seen or not with over 90% accuracy.

A.4.2 Experiments

In all experiments, these warnings can be safely ignored:

- `[Warning] min_data_in_leaf is set=20, min_child_samples=20 will be ignored.`
- `LineSearchWarning: The line search algorithm did not converge`
- `UserWarning: Line Search failed`

(E1): [Full Reproduction] [1 human-hour + 320 compute-hour + 128GB disk]: re-run our entire training and testing pipeline to validate our main identification result (C1).
Preparation: Complete steps 1–4 of the installation instructions given in §A.3.1. Do not complete step 5.
Execution: Run the first seven Python scripts in labeled numerical order, starting with `1-train_layer_1.py` and ending with `7-stats_final.py`. See §A.2.3 above for the estimated execution time of each step.
Results: After running `7-stats_final.py`, a results table will be displayed. The final identification accuracy, as shown in the last row, should be 90% or higher.

(E2): [Partial Reproduction] [1 human-hour + 8 compute-hour + 128GB disk]: re-run part of each step of our pipeline on a subset of users to validate our main identification result (C1); a reasonable alternative to (E1).
Preparation: Complete steps 1–5 of the installation instructions given in §A.3.1. Unzip the supplemental data and source code from Zenodo into the same folder.
Execution:

1. Run `py 1-train_layer_1.py` to train the first layer, but only train the first model (you can write `exit()` at the end of the main loop to do so).
   (a) This step should take around 3 hours.
   (b) After running, the final epoch should display a validation error (`valid_0's multi_error`) of 0.40 or less. If so, this step is verified, and it is safe to use our models for the rest of layer 1.
2. Run `py 2-train_layer_2.py` to train the second layer, but only train the first model (you can write `exit()` at the end of the main loop to do so).
   (a) This step should take around 3 hours.
   (b) After running, the final epoch should display a validation error (`valid_0's multi_error`) of 0.40 or less. If so, this step is verified, and it is safe to use our models for the rest of layer 2.
3. Run `py 3-test_and_cluster.py` to test the first two layers. When you see `Testing Accuracy...`, wait a few minutes for the accuracy to stabilize.
   (a) This step should take around 10 minutes.
   (b) The accuracy value should be around 85.0% or higher. If so, this step is verified, and the models in layers 1 and 2 are performing as expected.
4. Run `py 4-generate_groups.py` to cluster users into groups based on the test results.
   (a) This step should take around 5 minutes.
   (b) If you see `Created N groups of size [...]`, this step was successful and you can proceed.
5. Run `py 5-train_layer_3.py` to train all models in the third and final layer.
   (a) This step should take around 1 hour.
   (b) The final epoch of each model should display a validation error (`valid_0's multi_error`) of 0.50 or less on average. If so, this step is verified, and layer 3 is performing as expected.
6. Run `py 6-test_layer_3.py` to test the accuracy of all models in the third layer.
   (a) This step should take around 5 minutes.
   (b) The accuracy of each model should be around 75.0% or higher on average. If so, this step is verified, and layer 3 is performing as expected.
7. Run `py 7-stats_final.py` to test final accuracy.
   (a) This step should take around 5 minutes.

Results: After running `7-stats_final.py`, a results table will be displayed. The final identification accuracy, as shown in the last row, should be 90% or higher.

(E3): [Feature Importance] [5 human-minutes + 5 compute-minutes + 128GB disk]: run our model explainability script to validate our motion feature importance (C2).
Preparation: Complete steps 1–5 of the installation instructions given in §A.3.1. Unzip the supplemental data and source code from Zenodo into the same folder.
Execution: Run `py 8-explain.py`. This script should take at most 3 to 5 minutes to execute.
Results: After running `8-explain.py`, a graph of the results is stored in `stats/features.pdf`. This should be similar to Figure 16 in our paper, and motion features (blue) should constitute most of the *area* of the graph.

(E4): [Open World] [5 human-minutes + 3 compute-minutes + 128GB disk]: run our secondary evaluation to validate our open-world performance claims (C3).
Preparation: Complete steps 1–5 of the installation instructions given in §A.3.1. Unzip the supplemental data and source code from Zenodo into the same folder.
Execution: Run `py 9-open_world.py`. This script should take at most 1 to 3 minutes to execute.
Results: After running `9-open_world.py`, the value of `Overall Accuracy` shown should be at least 0.90.

(E5): [Subset] [5 human-minutes + 1 compute-hour + 4GB disk]: evaluate claims C1–C3 on a subset of users.
Preparation: Complete steps 1, 2, and 6 of the installation instructions given in §A.3.1; unzip subset scripts.
Execution: Run `C1.py`, `C2.py`, and `C3.py` in sequence. These scripts should take about 30 minutes to execute.
Results: For C1, the projected identification accuracy should be at least 90%. For C2, the motion features should account for more than 50% of the entropy gained. For C3, the overall accuracy should be at least 90%.

A.5 Notes on Reusability

We encourage researchers to build upon and improve our work, and to this end have released our entire source code under an MIT license. We further created the "BOXRR-23" dataset, the largest known motion capture dataset, to help researchers try new featurization techniques and model architectures beyond those included in our code. We encourage researchers interested in any topic involving human motion data to consider using the BOXRR-23 dataset, found here:

https://rdi.berkeley.edu/metaverse/boxrr-23/

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2023/.