TUM
Blockchain
Club

Verifibale Inference of big models

EIGENLAYER HACKERHOUSE
BERLIN 2025

Try Pitch

zkSNARK for proofs of inference

MODEL STAYS PRIVATE

- Because of the zero-knowledge property, the model weights remain private
- The proof does not reveal information about the model

COMPUTATION IS FULLY VERIFIABLE

- The proof is valid only when the computation has been done exactly as promised

Try Pitch

Big guarantees require big computation

NOT ALL APPLICATION REQUIRE HIDING
MODEL OWNER → SERVICE PROVIDER

LET'S USE SMALLER TOOLS

Try Pitch

TOPLOC: A Locality Sensitive Hashing Scheme for Trustless Verifiable Inference

Jack Min Ong [1] Matthew Di Ferrante [1] Aaron Pazdera [1] Ryan Garner [1] Sami Jaghouar [1] Manveer Basra [1]
Max Ryabinin [2] Johannes Hagemann [1]

Abstract

Large language models (LLMs) have proven to be very capable, but access to frontier models currently relies on inference providers. This introduces trust challenges: how can we be sure that the provider is using the model configuration they claim? We propose TOPLOC, a novel method for verifiable inference that addresses this problem. TOPLOC leverages a compact locality-sensitive hashing mechanism for intermediate activations, which can detect unauthorized modifications to models, prompts, or precision with 100% accuracy, achieving no false positives or negatives in

model outputs via APIs, have risen to meet the demands of users who lack the resources or expertise to operate large-scale inference pipelines themselves.

However, a critical challenge arises in this open ecosystem: trust. Users must trust that an inference provider is faithfully serving the model as advertised, without undisclosed modifications. A provider could, for instance, secretly reduce numerical precision to cut costs, fine-tune the model to introduce certain biases, or prepend an undisclosed system prompt to steer the model's outputs. Without robust verification methods, users can only rely on the provider's claims, leaving them vulnerable to having their outputs tampered with (Chen et al., 2023). There is thus a need for **verifiable**

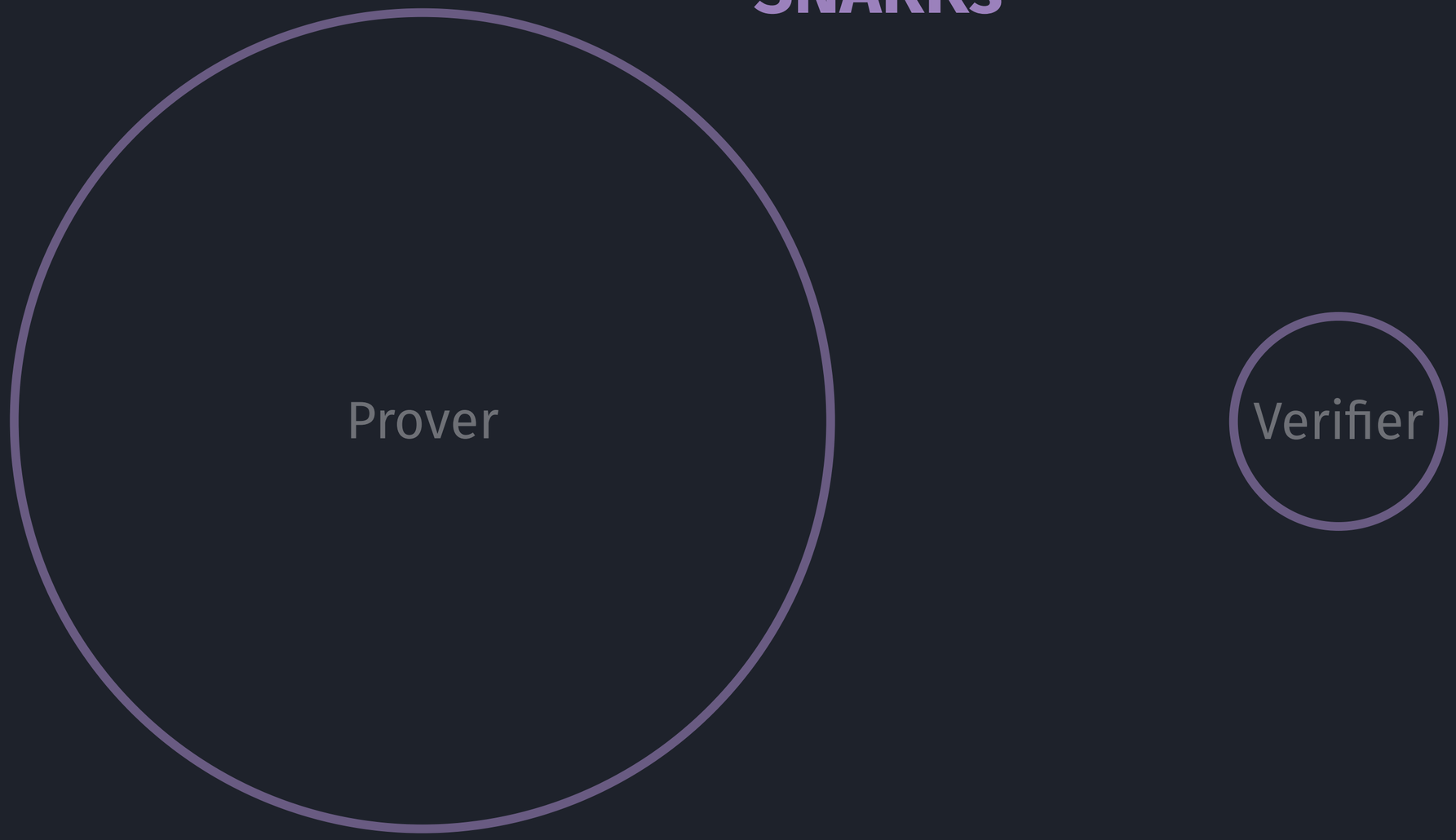
Try Pitch

TOPLOC in 20s

- Storing activation of just one layer become large really quickly
- We can do this smarter
  - Pick just k most important ones
  - Encode them as a polynomial to make this smaller
- In nutshell we create hash of the computation
- Now we have a small commitments for each prediction ✨✨

Try Pitch

SNARKs



Try Pitch

TOPLOC

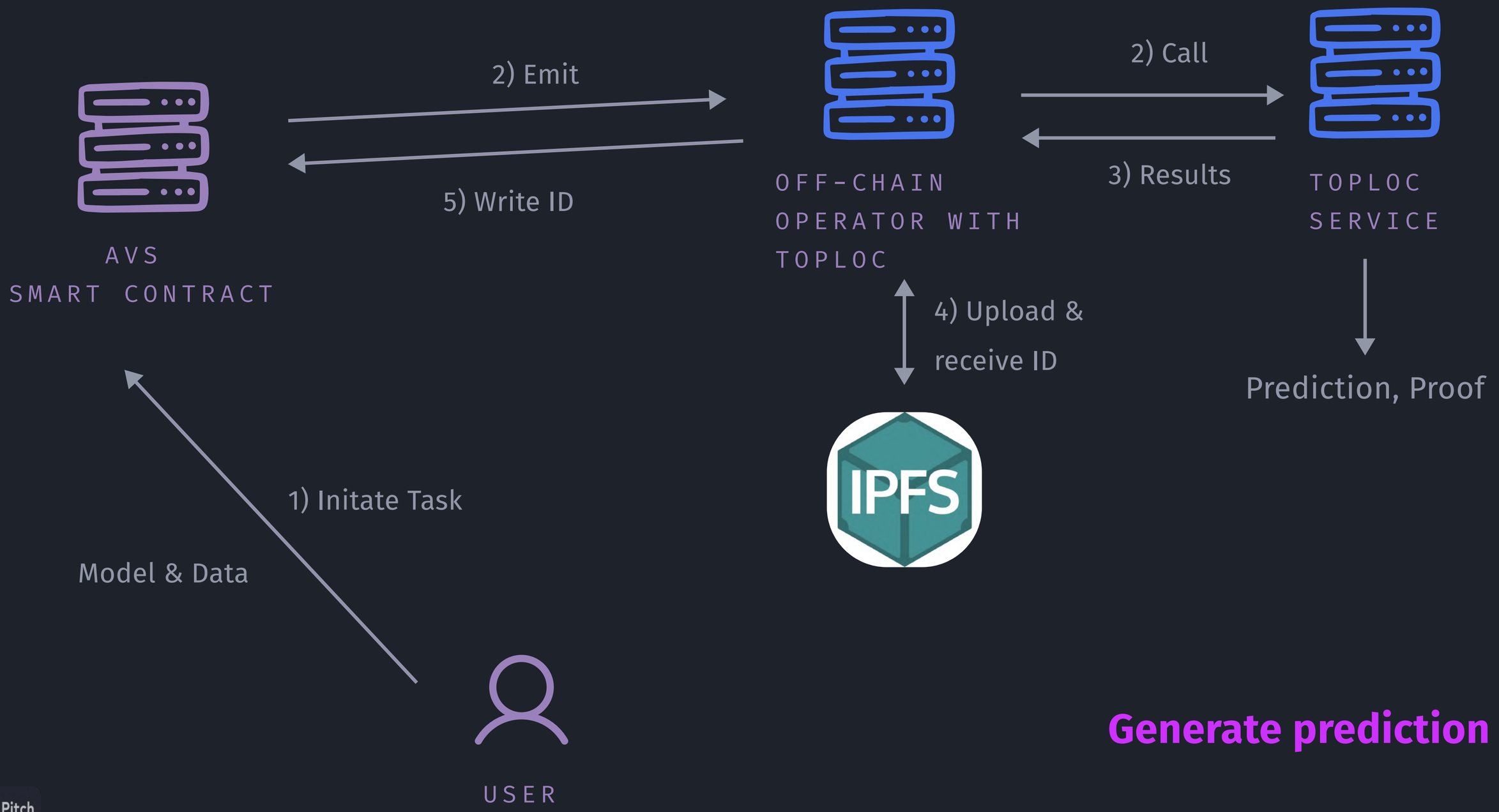
Prove everything, query subset for verification

Try Pitch

Verifiable Benchmarks

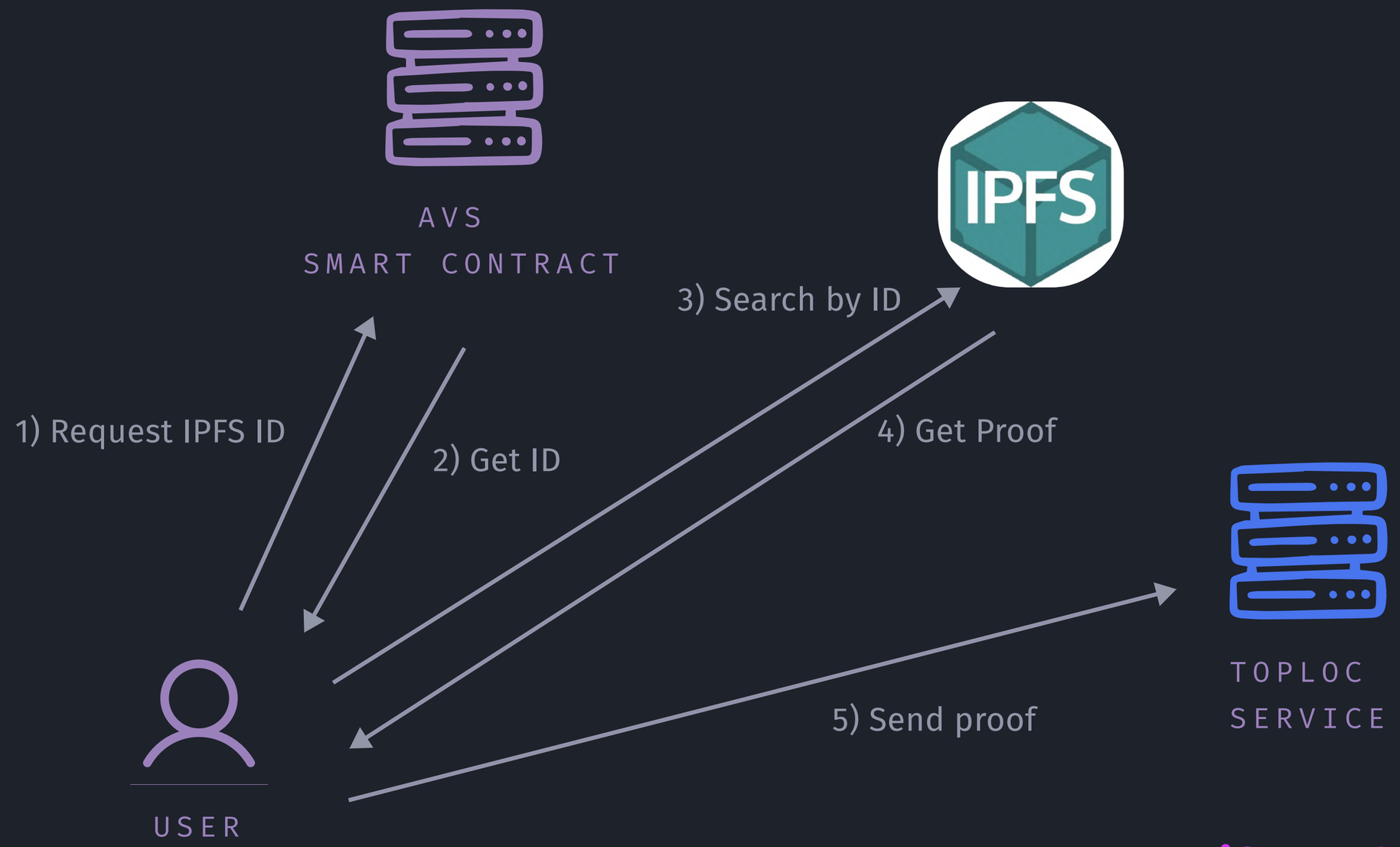
- Platfrom that allows to:
  - Generate prediction of your AI model on given data with proof
  - Verify proofs retrospectively
- Build as an AVS on Eigenlayer
  - Offchain component invokes TOPLOC service
- Using IPFS to store data, ID is written onchain

Try Pitch



Generate prediction

Try Pitch



Verify model predictions

Try Pitch

TOPLOC ML Verification

Cryptographic verification of ML inference results using locality-sensitive hashing and EigenLayer AVS

Generate Proof | Verify Proof

>_ Generate Proof

Provide your ML model and dataset URLs to generate a TOPLOC proof

Model URL

https://huggingface.co/model-name

Dataset URL

https://datasets.com/test-set

Generate TOPLOC Proof

<> Generated Proof

TOPLOC-encoded proof with Merkle tree aggregation

Generate a proof to see verification details

Try Pitch

TOPLOC ML Verification

Cryptographic verification of ML inference results using locality-sensitive hashing and EigenLayer AVS

Generate Proof | Verify Proof

Generate Proof

Provide your ML model and dataset URLs to generate a TOPLOC proof

Model URL

https://huggingface.co/benbencik/eigen-models/blob/main/SimpleNet_2025061

Dataset URL

https://huggingface.co/datasets/benbencik/eigen-datasets/blob/main/iris.csv

Generate TOPLOC Proof

Generated Proof

TOPLOC-encoded proof with Merkle tree aggregation

Generate a proof to see verification details

How TOPLOC Works

Locality-sensitive hashing with EigenLayer AVS verification

LSH Encoding

Hash top-k activation values and encode as polynomial congruence for 1000x compression

Merkle Aggregation

Bundle proofs into Merkle trees for efficient on-chain storage and batch verification

EigenLayer AVS

Decentralized validators verify proofs through independent recomputation

Try Pitch

benbencik/**eigen-models** like 0

Model card | Files and versions xet | Community

main eigen-models / SimpleNet_20250618210855.pt

benbencik Upload SimpleNet_20250618210855.pt 3d069d0 VERIFIED about 7 hours ago

download Copy download link history blame contribute delete pickle 3.67 kB

This file is stored with Git LFS . It is too big to display, but you can still download it.

Git LFS Details

SHA256: b18cb311c024b68ae865cd703c296178c63494810e3af9ec3831ff486d8436ab
Pointer size: 129 Bytes
Size of remote file: 3.67 kB
Xet backed hash: ee37bb542258cddcd65b0ee9ca52a9592ec2a175d77abc0aa29a255e54075eee

Raw pointer file

Git Large File Storage (LFS) replaces large files with text pointers inside Git, while storing the file contents on a remote server. More info.

Try Pitch

Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Community | Docs | Enterprise | Pricing | Log In | Sign Up

Datasets: benbencik/eigen-datasets | like 0

Modalities: Tabular | Formats: csv | Size: < 1K | Libraries: Datasets | pandas | Croissant | + 1

Dataset card | Data Studio | Files and versions | xet | Community 1

main | eigen-datasets / iris.csv

benbencik | Upload iris.csv | e1f9254 | VERIFIED | about 13 hours ago

raw | Copy download link | history | blame | contribute | delete | Safe | 2.78 kB

```csv
sepal length (cm),sepal width (cm),petal length (cm),petal width (cm),target
5.1,3.5,1.4,0.2,0
4.9,3.0,1.4,0.2,0
4.7,3.2,1.3,0.2,0
4.6,3.1,1.5,0.2,0
5.0,3.6,1.4,0.2,0
5.4,3.9,1.7,0.4,0
4.6,3.4,1.4,0.3,0
5.0,3.4,1.5,0.2,0
4.4,2.9,1.4,0.2,0
4.9,3.1,1.5,0.1,0
5.4,3.7,1.5,0.2,0
4.8,3.4,1.6,0.2,0
4.8,3.0,1.4,0.1,0
4.3,3.0,1.1,0.1,0
5.8,4.0,1.2,0.2,0
5.7,4.4,1.5,0.4,0
5.4,3.9,1.3,0.4,0
5.1,3.5,1.4,0.3,0
5.7,3.8,1.7,0.3,0
5.1,3.8,1.5,0.3,0
5.4,3.4,1.7,0.2,0
5.1,3.7,1.5,0.4,0
4.6,3.6,1.0,0.2,0
5.1,3.3,1.7,0.5,0
```

Try Pitch

```
def cleanup():
```

Problems 3 | Output | Debug Console | Terminal | Ports

```
2ff4c510f622: Verifying Checksum
2ff4c510f622: Download complete
47438591b681: Verifying Checksum
47438591b681: Download complete
7ee8b49a4227: Verifying Checksum
7ee8b49a4227: Download complete
2e5117df584b: Verifying Checksum
2e5117df584b: Download complete
7ee8b49a4227: Pull complete
2ff4c510f622: Pull complete
47438591b681: Pull complete
2e5117df584b: Pull complete
Digest: sha256:21a026ec90dab4a29a6f64d7125ec0176adcd6853fc9eb3dc2bc0463e8d22ddf
Status: Downloaded newer image for public.ecr.aws/z6g0f8n7/eigenlayer-hourglass:v0.1.0
{"level":"info","ts":"2025-06-19T02:29:48.201Z","caller":"caller/caller.go:68","msg":"Creating contract caller","AVSRegistrarAddress":"0x99aA73dA6309b8eC484eF2C95e96C131C1BBF7a0","TaskMailboxAddress":""}
Processing executor operators...
Generating registration data for executor operator: 0x15d34AAf54267DB7D7c367839AAf71A00a2C6A65
{"level":"info","ts":"2025-06-19T02:29:49.695Z","caller":"caller/caller.go:68","msg":"Creating contract caller","AVSRegistrarAddress":"0x99aA73dA6309b8eC484eF2C95e96C131C1BBF7a0","TaskMailboxAddress":""}
2025/06/19 04:29:49
2025/06/19 04:29:49 Save contract artefacts
2025/06/19 04:29:49
2025/06/19 04:29:49 Written contract output: contracts/outputs/devnet/taskMailbox.json
2025/06/19 04:29:49 Written contract output: contracts/outputs/devnet/taskAVSRegistrar.json
2025/06/19 04:29:49 Written contract output: contracts/outputs/devnet/avsTaskHook.json
2025/06/19 04:29:49 Written contract output: contracts/outputs/devnet/bn254CertificateVerifier.json
2025/06/19 04:29:49 Written contract output: contracts/outputs/devnet/HelloWorld.json
2025/06/19 04:29:49
2025/06/19 04:29:49 Devnet contracts deployed successfully in 37s
2025/06/19 04:29:50
2025/06/19 04:29:50 Registering AVS with EigenLayer...
2025/06/19 04:29:50
2025/06/19 04:29:53 Attempting to find AvsRegistrar in deployed contracts...
2025/06/19 04:29:53 Found AvsRegistrar: 'taskAVSRegistrar' at address 0x99aA73dA6309b8eC484eF2C95e96C131C1BBF7a0
2025/06/19 04:29:58 AVS registered with EigenLayer successfully.
2025/06/19 04:29:58 Registering operators with EigenLayer...
2025/06/19 04:29:58 Processing registration for operator at address 0x90F79bf6EB2c4f870365E785982E1f101E93b906
2025/06/19 04:30:05 Successfully registered operator 0x90F79bf6EB2c4f870365E785982E1f101E93b906 for OperatorSetID 0
2025/06/19 04:30:05 Processing registration for operator at address 0x15d34AAf54267DB7D7c367839AAf71A00a2C6A65
2025/06/19 04:30:11 Successfully registered operator 0x15d34AAf54267DB7D7c367839AAf71A00a2C6A65 for OperatorSetID 1
2025/06/19 04:30:11 Operator registration with EigenLayer completed.
Starting AVS components for environment: devnet
[+] Running 3/3
 ⠿ Network hourglass-local_hourglass-network  Created 0.1ss
 ⠿ Container hourglass-local-executor-1       Created 0.1ss
 ⠿ Container hourglass-local-aggregator-1     Created 0.1ss
hourglass-local-executor-1    | {"level":"info","ts":"2025-06-19T02:30:14.281Z","caller":"executor/run.go:42","msg":"Executor run"}
hourglass-local-aggregator-1  | {"level":"info","ts":"2025-06-19T02:30:15.476Z","caller":"aggregator/run.go:69","msg":"Loading core contracts from embedded config"}
hourglass-local-executor-1    | {"level":"info","ts":"2025-06-19T02:30:15.479Z","caller":"executor/run.go:71","msg":"Loading core contracts from embedded config"}
hourglass-local-aggregator-1  | {"level":"info","ts":"2025-06-19T02:30:15.480Z","caller":"inMemoryContractStore/inMemoryContractStore.go:58","msg":"Overriding contract","name":"TaskMailbox","previousAddress":"0x7306a64
9b451ae08781108445425bd4e8acf1e00","newAddress":"0x4B7099FD879435a087C364aD2f9E7B3f94d20bBe","chainId":31337}
hourglass-local-aggregator-1  | {"level":"info","ts":"2025-06-19T02:30:15.480Z","caller":"aggregator/run.go:92","msg":"Aggregator config: {Debug:false Operator:0x40002e0570 ServerConfig:{Port:9000 SecureConnection:fals
e AggregatorUrl:aggregator:9000} L1ChainId:31337 Chains:[0x40000ca9b0] Avss:[0x40000889c0] SimulationConfig:<nil> Contracts:[] OverrideContracts:0x400005e530}\n"}
hourglass-local-aggregator-1  | {"level":"info","ts":"2025-06-19T02:30:15.480Z","caller":"aggregator/run.go:93","msg":"Building aggregator components..."}
hourglass-local-aggregator-1  | {"level":"info","ts":"2025-06-19T02:30:15.480Z","caller":"caller/caller.go:68","msg":"Creating contract caller","AVSRegistrarAddress":"0x99aA73dA6309b8eC484eF2C95e96C131C1BBF7a0","TaskMa
ilboxAddress":"0x4B7099FD879435a087C364aD2f9E7B3f94d20bBe"}
hourglass-local-executor-1    | {"level":"info","ts":"2025-06-19T02:30:15.481Z","caller":"inMemoryContractStore/inMemoryContractStore.go:58","msg":"Overriding contract","name":"TaskMailbox","previousAddress":"0x7306a64
9b451ae08781108445425bd4e8acf1e00","newAddress":"0x4B7099FD879435a087C364aD2f9E7B3f94d20bBe","chainId":31337}
hourglass-local-executor-1    | {"level":"info","ts":"2025-06-19T02:30:15.482Z","caller":"caller/caller.go:68","msg":"Creating contract caller","AVSRegistrarAddress":"0x99aA73dA6309b8eC484eF2C95e96C131C1BBF7a0","TaskMa
ilboxAddress":"0x4B7099FD879435a087C364aD2f9E7B3f94d20bBe"}
hourglass-local-executor-1    | {"level":"info","ts":"2025-06-19T02:30:15.535Z","caller":"executor/executor.go:67","msg":"Initializing AVS performers"}
hourglass-local-executor-1    | {"level":"info","ts":"2025-06-19T02:30:15.535Z","caller":"executor/executor.go:121","msg":"Booting AVS performers"}
hourglass-local-aggregator-1  | {"level":"info","ts":"2025-06-19T02:30:15.540Z","caller":"aggregator/aggregator.go:171","msg":"Initializing chain pollers...","chains":[{"name":"ethereum","version":"","chainId":31337,"r
```

cd /Users/a...
go run cmd...
cd toploc ...
node ml-ve...

Try Pitch

toploc > main.py > ...

```
def cleanup():
```

Problems 3 | Output | Debug Console | **Terminal** | Ports

```
6f622f6d61696e2f697269732e6373760000000000000000000000000000000000000000
2025/06/19 04:31:44
2025/06/19 04:31:44 ## Setting up 1 EVM.
2025/06/19 04:31:44 ==========================
2025/06/19 04:31:44 Simulated On-chain Traces:
2025/06/19 04:31:44
2025/06/19 04:31:44   [480620] 0x4B7099FD879435a087C364aD2f9E7B3f94d20bBe::createTask(TaskParams({ refundCollector: 0x0000000000000000000000000000000000000000, avsFee: 0, executorOperatorSet: OperatorSet({ avs: 0x70997
970C51812dc3A010C7d01b50e0d17dc79C8, id: 1 }), payload: 0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000000400000000000000000000000000000000
0000000000000000000000000000c000000000000000000000000000000000000000000000000000000000
```

```
toploc > app.py > run_task
1   from flask import Flask, request
Problems 3   Output   Debug Console   Terminal   Ports
(i2dl) (base) ➜  toploc git:(main) ✗ curl -X POST http://localhost:6500/run -F "dataset_url=https://huggingface.co/datasets/benbencik/eigen-datasets/blob/main/iris.csv" -F "model_url=https://huggingface.co/benbencik/eigen-models/blob/main/SimpleNet_20250618210855.pt" -F "task=predict"
Proof stored: proofs/proof_1750300849636158713.json
(i2dl) (base) ➜  toploc git:(main) ✗ docker-compose up --build -d
[+] Building 111.7s (13/13) FINISHED
 => [internal] load build definition from Dockerfile                                                                    0.0s
 => => transferring dockerfile: 69B                                                                                     0.0s
 => [internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 72B                                                                                        0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                                                     1.5s
 => [internal] load build context                                                                                       0.0s
 => => transferring context: 6.64kB                                                                                     0.0s
 => [1/8] FROM docker.io/library/python:3.11-slim@sha256:9e1912aab0a30bbd9488eb79063f68f42a68ab0946cbe98fecf197fe5b085506  0.0s
 => CACHED [2/8] WORKDIR /app                                                                                           0.0s
 => CACHED [3/8] RUN apt-get update && apt-get install -y     gcc     g++     make     && rm -rf /var/lib/apt/lists/*    0.0s
 => CACHED [4/8] COPY requirements.txt .                                                                                0.0s
 => CACHED [5/8] RUN pip install --no-cache-dir -r requirements.txt                                                     0.0s
 => [6/8] COPY . .                                                                                                      0.0s
 => [7/8] RUN find . -name "*.so" -delete &&     find . -name "__pycache__" -type d -exec rm -rf {} + 2>/dev/null || true &&     pip install -e .   109.3s
 => [8/8] RUN mkdir -p trained_models models proofs                                                                     0.3s
 => exporting to image                                                                                                  0.4s
 => => exporting layers                                                                                                 0.4s
 => => writing image sha256:c783cb33b23462d834e49cbc87e22dd35990ab221f5087c58489444b5c2f880d                            0.0s
 => => naming to docker.io/library/toploc-toploc                                                                        0.0s
[+] Running 1/1
 ⠿ Container toploc-toploc-1  Started                                                                                  10.8s
(i2dl) (base) ➜  toploc git:(main) ✗ curl -X POST http://localhost:6500/run -F "dataset_url=https://huggingface.co/datasets/benbencik/eigen-datasets/blob/main/iris.csv" -F "model_url=https://huggingface.co/benbencik/eigen-models/blob/main/SimpleNet_20250618210855.pt" -F "task=predict"
/9k/BpQoIK+iMQ==~/9k/Ogfx7cjCrg==~/9k/BaZvXprqRw==~/9k/Jzi6uPlAug==~/9k+6DjYAC+kOQ==~/9k/GiaKM/mU/g==~/9k+8FEzQiyxbA==~/9k/Gbis45+WAg==~/9k/JksBzEGzdA==~/9k/NCBVU3SBtA==~/9k/Del0NgG3gw==~/9k/GlEfM/pqWg==~/9k/LfWMhTKlPQ
==~/9k+z9dLIqhEyw==~/9k+tssrjEbzag==~/9k+oybMAi6cGg==~/9k+y3XbTFEI4w==~/9k/DKZuAjocLQ==~/9k/MzKqSlJmSg==~/9k+5jLKX53Q5Q==~/9k/TVcmW5bhIw==~/9k/BhRGoJqjMQ==~/9k+G13eX4KmQw==~/9k/Wr6xLuu+ng==~/9k/QETaXZvZBA==~/9k/WpQAEn9
bHw==~/9k/M902prWJ0w==~/9k/GWNmXJdH1Q==~/9k/GPWWmoYP/g==~/9k/LIfqvAZJ3Q==~/9k/QvWCBUolUA==~/9k/QJoua9TKzQ==~/9k+jeOiRzZdKA==~/9k+kFeHfw6S9w==~/9k/O8rgDGVzfQ==~/9k/DWl9Um7Fug==~/9k/E3vDAjxG0Q==~/9k+zwIV1FuS9w==~/9k/DZQn
PRcwfA==~/9k/H/WVdv1eKw==~/9k+8oHrG5p2iQ==~/9k/c8rDzEkzhw==~/9k+5t1W0U5fMA==~/9k/NPWZKNDW/A==~/9k/LSBrk2nBqg==~/9k/Ot0nX6UmVA==~/9k+9Age0VQ0iw==~/9k/DKZuXprqRw==~/9k/B4HohjWL2w==~/9k/Gvur0FXN2Q==~/9lARo18h2ryYQ==~/9lAO
I2FI+gAvw==~/9lAUKvs8QJ2hA==~/9lAM5Opy2P3dg==~/9lAShm9lqQgOA==~/9lANgeO/CEJ4w==~/9lAO4FT2602jg==~/9lAC+8YFqPKyw==~/9lARDIhkY7JxQ==~/9lAIG8lwTqgJg==~/9lAIG8lJNuR8A==~/9lALGkHdh53iQ==~/9lAOujuUpJFyg==~/9lAQROwBG2T9g==~/9
lAFB/2cAEP+w==~/9lAO1awMRsL6g==~/9lAMtCTAmHxXg==~/9lAKmLy6tsO9w==~/9lAVI16lZ45mg==~/9lAKLIjZt50fg==~/9lAPw2UnLvdDw==~/9lALYFjuBuEvA==~/9lAWp+5RW26ig==~/9lAQZOgRGFUAA==~/9lAOLgvwD7kKw==~/9lAPQd/vDLJxQ==~/9lAVGLQHNEH2g==
~/9lAVj5CvTwF0g==~/9lAO4FaW7y2ew==~/9lAGNazosPEsw==~/9lAJtaxDYXaBQ==~/9lAIkqXvixsXw==~/9lAJxnXD5B8nQ==~/9lAVr4y2acUCQ==~/9lALsRpD5LR5Q==~/9lALVa5W73hHw==~/9lARo18Y+FAtQ==~/9lATbglB3hH0Q==~/9lAIVzeQ1cYGg==~/9lALSwdYtAvd
A==~/9lANcRoZNh8nQ==~/9lAO6v9eCoaIQ==~/9lALSwdqeKS8w==~/9lAEGkSPThbHQ==~/9lALu8MpM+R7w==~/9lAJPs/ey4jRQ==~/9lAKZ/c4brIwg==~/9lANZm+D5T8iQ==~/9lAAywqzWNExw==~/9lAKZ/cmqhlQw==~/9lAdBmplq0gNw==~/9lAWJmkD578iA==~/9lAgD4uEr
KwYg==~/9lAaB/Hk53BpQ==~/9lAdQdiJv7fFg==~/9lAj1aIsRaL1Q==~/9lAOmkFfTibEg==~/9lAhz4tErKwYg==~/9lAgQFEP1yoQw==~/9lAe8Q0cyfDrA==~/9lAVDgth21H0A==~/9lAaEpqaPnBpQ==~/9lAbtyZCpB7lw==~/9lAW41yOT9rWQ==~/9lAXWjkRGx+ow==~/9lAX0R
WwU7KyA==~/9lAZavieDQaIA==~/9lAhWK7OUmV/A==~/9lAnQE5Rn7Lyw==~/9lAYrgUFbWO4g==~/9lAc4E227c2jQ==~/9lATbgeXMbyYQ==~/9lAlNBbtCJqVA==~/9lAWO7qJPESAQ==~/9lAaTgjMiidGA==~/9lAeZN85+WFvg==~/9lAUKvsqfATBQ==~/9lAS9ystRrQ4A==~/9lA
cmjaPVNbGg==~/9lAd+KtY+jrRQ==~/9lAh2jQC5diNQ==~/9lAgQFEBoWL1Q==~/9lAc1aT6eyoQw==~/9lAVujV9hV3iA==~/9lAaPT9WsDP3A==~/9lAiURCFsR1fw==~/9lAYuK3h3CdGA==~/9lAYNyiX9UmTw==~/9lAR1CS9AwqXw==~/9lAalB/bQuGwg==~/9lAcpN9UqebEA==~/
9lAYyXdl6+Gwg==~/9lAWJmkD578iA==~/9lAeNBmCY4VDA==~/9lAcLgL8g3dDg==~/9lAZsQ+q/3gGg==~/9lAY9Bw16IcKA==~/9lAXladqfNoTQ==~/9lAWMRH+ihnSQ==~/9lATqXXCYY/sQ==
Accuracy: 92%
(i2dl) (base) ➜  toploc git:(main) ✗
```

curl -X POST http://localhost:6500/run -F "dataset_url=https://huggingface.co/datasets/benbencik/eigen-datasets/blob/main/iris.csv" -F "model_url=https://huggingface.co/benbencik/eigen-models/blob/main/SimpleNet_20250618210855.pt" -F "task=predict"

Try Pitch

Pinata

A

Workspace
Alexander's Workspace

IPFS
- Files
- Groups
- Gateways
- Analytics

DEVELOPER
- API Keys
- Webhooks
- Access Controls

EXTENSIONS
- Marketplace
- Integrations

Documentation

FILES

amber-biological-toad-707.mypinat...

+ Add

PUBLIC PRIVATE

Public files are accessible via IPFS

Search files and CIDs

NAME	CID	SIZE	CREATION DATE	FILE ID
benchmark_result.json	bafkr...udybq	411 B	6/19/2025	

Rows per page: 10

Try Pitch

TOPLOC ML Verification

Cryptographic verification of ML inference results using locality-sensitive hashing and EigenLayer AVS

Generate Proof | Verify Proof

>_ Generate Proof

Provide your ML model and dataset URLs to generate a TOPLOC proof

Model URL

https://huggingface.co/benbencik/eigen-models/blob/main/SimpleNet_2025061

Dataset URL

https://huggingface.co/datasets/benbencik/eigen-datasets/blob/main/iris.csv

Generate TOPLOC Proof

<> Generated Proof

TOPLOC-encoded proof with Merkle tree aggregation

Proof (split by ~)


```
/9k/BpQoIK+iMQ==
/9k/Ogfx7cjCrg==
/9k/BaZvXprqRw==
/9k/Jzi6uPlAug==
/9k+6DjYAC+kOQ==
/9k/GiaKM/mU/g==
/9k+8FEzQiyxbA==
/9k/Gbis45+WAg==
/9k/JksBzEGzdA==
/9k/NCBVU3SBtA==
/9k/Del0NgG3gw==
/9k/GlEfM/pqWg==
/9k/LfWMhTKlPQ==
```


Accuracy

92%

Try Pitch

Current implementation

Verifiable inference

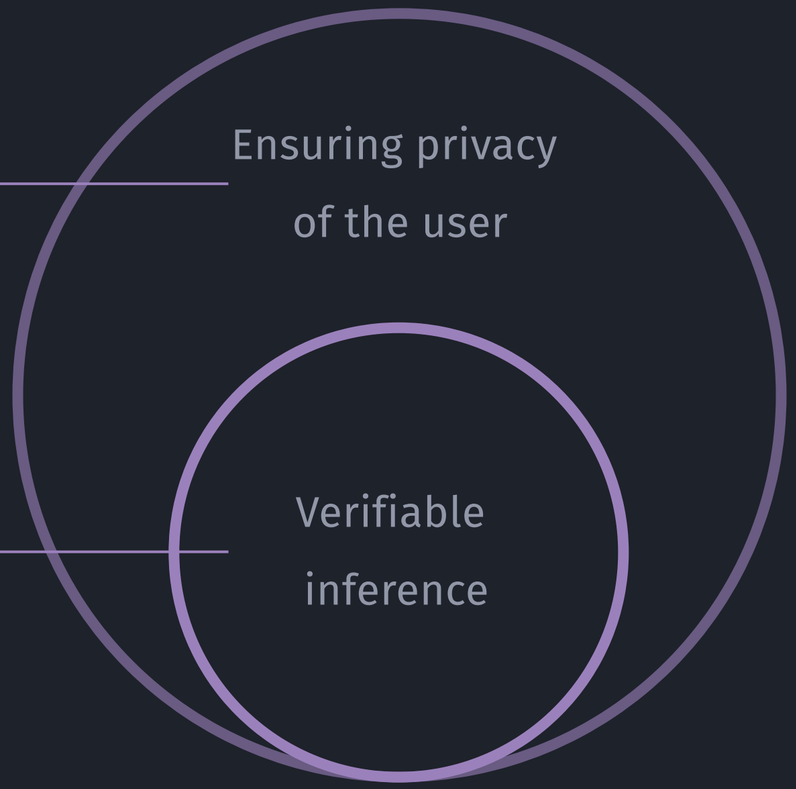
Try Pitch

FUTURE WORK

Encrypt the input bundled with proofs

- we do not want inputs of users to be public
- design a scheme to tackle this

Current implementation



Try Pitch

FUTURE WORK

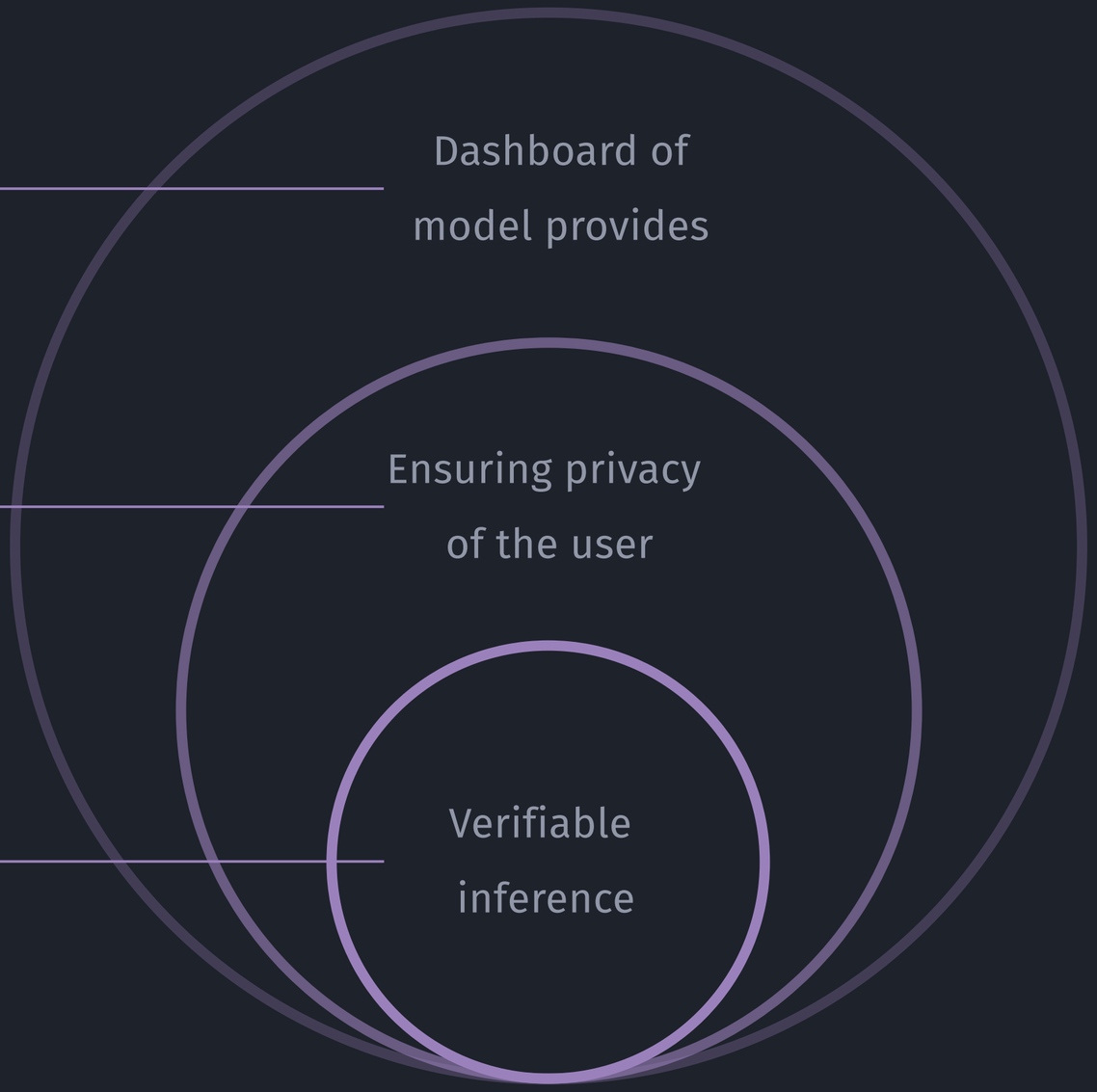
Aggregate information from the verification

- make a dashboard that updates rating for providers

Encrypt the input bundled with proofs

- we do not want inputs of users to be public
- design a scheme to tackle this

Current implementation



Try Pitch

LEFT
EXIT 12
Trust
the provider
Trust
the protocol
Me
imgflip.com

Try Pitch

ALEXANDER SEMENOV

BENJAMÍN BENČÍK

https://github.com/benbencik/verifiable-benchmarks

Try Pitch