ZK-friendly ML model explorations - Milestone 1

saeyoon17

January 2024

This article outlines the progress made in acceleration program by Ethereum PSE team. Article first restates the goals made for the report, concluding with the progress made and future plans.

1 Goals

In milestone 1, the goal is to provide the source code and documentations of how one can train a neural network, using the heart failure dataset and make heart failure prediction with it. The code should contain evaluation pipeline where one can check the model accuracy. It also should allow one to prove that the prediction was made using the correct circuit.

- Functionality: Train/Test/Inference pipeline using neural network.
- Functionality: Converting neural network model to ZK circuits using Circom or EZKL.
- Functionality: Proof generation / Verification pipeline with utilities to check time/memory complexity.

2 Progress

I have my work publically available at https://github.com/saeyoon17/zk-models. The progress made are as follows.

- Pipeline where one can train pytorch neural networks. Currently, linear regression model and multi-layer perceptrons are available. Yet, one can easily adopt other model architectures by adding pytorch models under **models** directory.
- While the goal was to implement proving/verification pipeline using either Circom or EZKL, I wanted to compare the costs between them. Therefore I implemented two pipelines where one can prove their neural network using both Circom and EZKL. For Circom, groth16 is used to prove model whereas PLONK is used for EZKL.

- **prove_model_circom** and **prove_model_ezkl** both contains utilities to check following criterias.
  - number of parameters
  - model accuracy
  - zk-circuit accuracy
  - time taken for each proof / verification step (different for each Circom / EZKL)

  One important thing to note here is that the ZK proof system mostly works under finite field. This leads the necessity of converting floating points to integers. Usually, scaling factors are used to scale both network weights and input features. Yet, this may lead to performance degradation depending on different scaling approaches. Therefore, we calculate zk-circuit accuracy, which calculates the accuracy of the transformed-neural network model in which the ZK protocols prove upon.

  For EZKL, we use their predetermined strategy for scaling laws. For Circom, we use scaling factor of 1e6 for linear regression and 1e4 for MLP due to overflow.

  Here we present performance test results of linear regression models. (20 trials)

  - EZKL
    - setting_time: 0.02±0.03
    - calibration_time: 0.36±0.29
    - compile_time: 0.005±0.011
    - get_srs_time: 0.003±0.004
    - witness_generation_time: 0.14±0.09
    - setup_time: 0.72±0.66
    - proof_generation_time: 0.62±0.31
    - verification_time: 0.04±0.03
    - num_params: 38.0
    - total: 276.0
    - correct: 145.0
    - zk_total: 276.0
    - zk_correct: 145.0
    - accuracy: 52.53623188405797
    - zk accuracy: 52.53623188405797
  - Circom
    - witness_generation_time: 0.11±0.03
    - proof_generation_time: 1.06±0.14
    - verification_time: 0.76±0.089

    * num_params: 38.0
    * total: 276.0
    * correct: 145.0
    * zk_total: 276.0
    * zk_correct: 145.0
    * accuracy: 52.53623188405797
    * zk accuracy: 52.53623188405797

For simple linear regression model, we observe no performance degradation. Below we present the result for MLP with 1 hidden layer with hidden dimension of 8. While Poly-activation have shown its great performance, we observe that this without normalizing input data leads to unstable training. Therefore, we stick with ReLU.

- EZKL
  - setting_time: 0.25
  - calibration_time: 1.44
  - compile_time: 0.056
  - get_srs_time: 0.07
  - witness_generation_time: 0.54
  - setup_time: 71.85
  - proof_generation_time: 439.38
  - verification_time: 1.54
  - num_params: 242.0
  - total: 276.0
  - correct: 164.0
  - zk_total: 276.0
  - zk_correct: 110.0
  - accuracy: 59.42028985507246
  - zk accuracy: 39.85507246376812
- Circom
  - witness_generation_time: 4.91
  - proof_generation_time: 138.87
  - verification_time: 10.49
  - num_params: 242.0
  - total: 276.0
  - correct: 164.0

- zk_total: 276.0
- zk_correct: 164.0
- accuracy: 59.42028985507246
- zk accuracy: 59.42028985507246

EZKL tends to consume more time, due to lookup table generation. Yet, above result does not contain the time for Circom circuit compilation and groth16 setup.

EZKL shows huge performance degradation when converted to zk circuit. I believe this can be due to hyperparameters being used in calibration step which for now is chosen as the setup used in EZKL demo. I am looking forward to try out different hyperparameters as well.

3 Future Works

While the previous plans for Milestone 2 was to focus on implementing the fixed solution of linear regression, I would like to make modifications if possible. I first plan to refactor the code for more usability (current version requires modifications for switching between different network architectures). Moreover, I would like to explore more about how the time complxity grows as the number of parameters go up (scaling law for zk-proving). Finally, I plan to explore more about scaling approaches, including looking up specifically how EZKL does the scaling.