**Abstract**

We present an efficient and succinct zero-knowledge proof application using zkSNARKs for remotely verifying the forward-pass execution of an arbitrarily-sized neural network with hidden inputs and model parameters. Our zero-knowledge guarantee allows the prover to hide information about the input and model parameters from the verifier while being able to attest to the integrity of these parameters and the model's execution.

Our approach is transformative for various applications such as nuclear treaty verification without the need to disclose sensitive data, computer vision auditing without the need to leak footage, and secure patient diagnosis without the need to disclose individually identifiable health information. We demonstrate an end-to-end implementation of this proof system using custom gadgets on a neural network for the classification of MNIST handwritten digits.

**System Architecture and Design**

A neural network such as input or weights while maintaining verification of the input to this program. We demonstrate the functionality of this system using custom gadgets on a neural network for the classification of MNIST handwritten digits.

**Problem Statement**

Neural Networks are increasingly being used for decision making and analysis; however, there does not exist an efficient method for remotely verifying the execution of a neural network or for hiding information about a neural network such as input or weights while maintaining verification of the providence of these hidden values.

In response to this problem, in this work, we present an efficient system for the remote verification of the execution of a neural network and the verification of the input to this program. We demonstrate the functionality of this program on the toy problem of digit classification using the MNIST database of handwritten digits.

**Background**

A Zero-Knowledge Proof ( zkSNARK ) is a way to prove a claim without leaking details about why the claim is true. zkSNARK says that a claim is true without revealing any additional information.

**RICS Optimization Strategies**

The standard pipeline for the creation of zkSNARK generators and verifiers from general computation involves the pipeline (right). Due to the size of the computational overhead that gets introduced in terms of both memory and total running time in this project, we wrote the program directly as a R1CS and perform optimizations fully and correctly.

**Conclusion and Future Work**

We present and demonstrate an optimized architecture and implementation for the efficient execution and remote verification of feed-forward neural networks and their inputs. Our implementation is heavily optimized with the largest number of constraints coming from the hashing algorithm and digital signature verification scheme. This program currently uses fewer constraints than would be generated from the compilation from a general circuit. We suggest the following avenues for future research which either expand on this project or would aid with increasing the efficiency of zkSNARK construction.

1. Implement additional neural network features which effectively leverage zkSNARKs for remote verification.
2. Implement a neural network specific compiler and optimizer for the reduction from Computation to RICS.
3. Develop a method that is more efficient than QAPs or SPPs for the expansion of RICS to a Linear PCP.
4. Develop and implement a post-quantum secure zkSNARK construction scheme and implement post-quantum cryptographic gadgets in RICS.

**Results**

We constructed and tested a large number of reusable gadgets for a variety of operations from logic gate verification to full neural network execution and RSA signature checking. Below is a hierarchical taxonomy of our gadgets.

**Conclusion and Future Work**

We present and demonstrate an optimized architecture and implementation for the efficient execution and remote verification of feed-forward neural networks and their inputs. Our implementation is heavily optimized with the largest number of constraints coming from the hashing algorithm and digital signature verification scheme. This program currently uses fewer constraints than would be generated from the compilation from a general circuit. We suggest the following avenues for future research which either expand on this project or would aid with increasing the efficiency of zkSNARK construction.

1. Implement additional neural network features which effectively leverage zkSNARKs for remote verification.
2. Implement a neural network specific compiler and optimizer for the reduction from Computation to RICS.
3. Develop a method that is more efficient than QAPs or SPPs for the expansion of RICS to a Linear PCP.
4. Develop and implement a post-quantum secure zkSNARK construction scheme and implement post-quantum cryptographic gadgets in RICS.