Zero-Knowledge Proofs for AI Model Validation

Ben Chase
University of New Brunswick
Fredericton, Canada
bchase1@unb.ca

Mackenzie Chase
University of New Brunswick
Fredericton, Canada
wlm.chase@unb.ca

Arthkumar Patel
University of New Brunswick
Fredericton, Canada
arth.patel@unb.ca

Eduardo Lazo Vera
University of New Brunswick
Fredericton, Canada
lazo.vluis@unb.ca

ABSTRACT

This study explores zero-knowledge proofs, such as zk-SNARKs and zk-STARKs, for validating artificial intelligence predictions without revealing sensitive inputs or model parameters. Targeting convolutional neural networks, we evaluate performance and scalability through prior research. The results underscore zero-knowledge proofs' value for private, reliable artificial intelligence verification.

ACM Reference Format:
Ben Chase, Mackenzie Chase, Arthkumar Patel, and Eduardo Lazo Vera. 2026. Zero-Knowledge Proofs for AI Model Validation. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (UNB - Foundations of Privacy).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Artificial Intelligence (AI) and machine learning (ML) have grown rapidly and have now been adapted for various applications that expands to nearly every aspect of our society. From enhancing diagnostic capabilities in healthcare and optimizing complex financial trading strategies to enabling autonomous vehicles to drive innovation and efficiency [18]. The increasing sophistication of their opaque nature and autonomy of these models offers powerful capabilities but simultaneously creates significant privacy challenges due to the vastness of their datasets.

The need for robust AI model validation is directly related to the high-stakes nature of many applications in which AI is utilized [18]. Validation processes are essential to ensure that models perform accurately, reliably and predictability, not just in laboratory conditions but also in real-world deployments. Failures in AI systems, whether due to algorithmic flaws, biases learned from the data, or unforeseen interactions with the environment, can have significant consequences, ranging from economic losses to unfair discrimination to safety hazards [32]. Consequently, verifying that an AI model meets its requirements and behaves ethically before and during its deployment is critical.

UNB - Foundations of Privacy, April 4, 2025, Fredericton, NB
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

When utilizing AI in any field there are several traits which are desired beyond simple accuracy, these desirable traits in AI models include robustness (the ability for a model to withstand against adversarial inputs), fairness (the ability for a model to avoid undue bias) , an transparency or explainability of decisions (allowing human understanding and accountability) [27]. Verifying the robustness might reveal vulnerabilities or proprietary defense mechanisms, thus achieving these traits without violating privacy becomes a challenge.

Therefore, for this project we investigated Zero-Knowledge - Proofs (ZKPs) as a powerful cryptographic solution specifically tailored to our privacy-validation dilemma [5]. ZKPs enable a party that possesses private information (Prover) to convince another party (Verifier) that a statement about this information is true. Using the ZKP methodology as a basis we explore its adaption to enable rigorous, verifiable validation of AI models while upholding the critical privacy requirements of both the data subjects and the model, hence forth fostering trust in AI through mathematically guaranteed privacy.

1.1 Background

The following is a concise overview of the foundational concepts required to understand modern privacy preserving AI model validation techniques.

1.2 Zero-Knowledge Proofs

In simple terms, zero-knowledge proofs (ZKPs) are used to convince [*] one entity (called a verifier) that another entity (called a prover) possesses secret information, without actually revealing any secret information. At first glance, this seems paradoxical; however, there are subtle ways to reveal indirect information.

The following example illustrates concepts involved in ZKPs without going into technical detail. Suppose you have a red card and a green card which are identical except for colour. You wish to convince a friend who is red-green colour blind that the two cards are indeed different colours. You give both cards to your friend, who hides them behind their back. They choose one, show it to you, and put it back behind their back. Then your friend decides whether or not they want to swap the cards, before showing you one again. This time, you correctly inform your friend if they swapped cards or showed you the same one. Your friend might not be persuaded after

[*]The term "proof" in ZKP does not refer to a mathematical proof.

one guess, so the process can be repeated until they are. This well-known analogy has originally been attributed to Oded Goldreich [14].

The formal setup for a ZKP has the following components: a prover P who holds the secret, and a a verifier V who wants to be convinced of a statement X which is a public claim about the witness. The goal of the setup is to generate a proof that convinces V that X is true, while revealing nothing else about X. ZKPs can be interactive, as in the previous scenario where the prover and verifier both must be present for the duration of the proof. Non-interactive ZKPs do not require a back-and-forth between two parties to execute a ZKP, reducing transmission costs.

Zero Knowledge Proofs require three canonical properties; completeness, soundness, and zero-knowledge. Completeness means that if a X is true and both P and V follow the protocol honestly, then V will always accept the proof. Soundness means that when X is false, a cheating P cannot convince an honest V that the statement is true. Zero-knowledge means that V learns nothing other than the fact that X is true and the proof does not leak any information about P.

1.3 Neural Networks

Neural networks are made up of individual *perceptrons*, inspired by biological neurons[†]. Perceptrons process inputs and produce outputs. They are grouped into layers, where each perceptron receives input from the previous layer and passes output to the next. A fully connected network links every perceptron in one layer to every perceptron in the next.

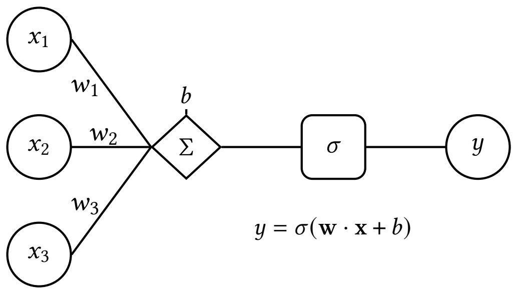

Figure 1: Single perceptron structure.

Figure 1 illustrates a simplistic perceptron, where weights $\mathbf{w}$ multiply inputs $\mathbf{x}$ as a dot product, $\mathbf{w}\cdot\mathbf{x}$, summing the weighted contributions. The bias b shifts this sum, adjusting the activation threshold. The activation function σ then introduces nonlinearity; the sigmoid function is often cited in literature, but modern implementations often use versions of ReLU, defined as $\text{ReLU}(x) = \max(0, x)$, which improves on sigmoid by avoiding the vanishing gradient problem [19].

1.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a specific type of neural network that are tailored for image processing [9]. They are composed of several layers – convolutional layers, activation layers, pooling layers, and fully connected layers. Before we discuss how CNNs work, we briefly explain how images are seen by a computer. Images are composed of pixels, or squares of information. These squares of information are simply numbers, telling the computer how much red is in this particular pixel for example. These pixels can carry other information such as luminosity, or other colour channels such as green and blue.

The feature extraction layers are the core component of what differentiates CNNs from other models. These layers consist of convolution, activation and pooling operations.

The convolutional and activation layers extract features by passing a kernel over an image. A kernel, is simply a matrix, typically 3×3 but can vary. The kernel effectively scans the image, does some computations, and outputs a new 'image'. This output is then sent to the pooling layer which compresses the output by removing unnecessary information, keeping only the most pertinent data. This process is repeated several times. Once the features have been extracted, they are processed by the fully connected layer tasked with classifying the image.

1.5 Freivald's Algorithm

Freivald's algorithm is a probabilistic technique that verifies the correctness of a matrix multiplication $A \times B = C$, without actually computing $A \times B$ (which would cost $O(n^3)$ [‡]). Instead, a random vector $\vec{r}$ is generated, consisting of entries with value 0 or 1 with probability $\frac{1}{2}$. The cost of computing the product of $B\vec{r}$ is $O(n^2)$, and the result is another vector. It follows that the total cost of computing $A \times (B\vec{r})$ is $O(n^2) + O(n^2) = O(n^2)$.

If $A \times B = C$, then $A \times (B\vec{r}) = C\vec{r}$ always. If $A \times B \neq C$, then it might still be possible that $A \times (B\vec{r}) = C\vec{r}$ (if $\vec{r}$ is the zero vector for example). It can be proven that the latter case happens with probability $\frac{1}{2}$. Freivald's algorithm checks the equality $A \times (B\vec{r}) = C\vec{r}$. If the equality holds, it increases confidence that $A \times B = C$. If the equality fails, it proves that $A \times B \neq C$. The verification is rerun k times, with a new random r generated each time, driving down the total probability of error to $\frac{1}{2^k}$.

1.6 Elliptic Curve Cryptography

The BN128 curve is an elliptic curve with pairing-based properties[1, 7] that are needed for SNARK systems such as Groth16 [17]. In general, an elliptic curve over a field $\mathbb{F}_p$ is a curve of the form $y^2 = x^3+ax+b$, where $a, b \in \mathbb{F}_p$ and the discriminant $4a^3+27b^2 \neq 0$. This set of points, plus a special point called the point at infinity, forms a group under a geometric addition operation defined on the curve.

The security of Elliptic Curve Cryptography (ECC) relies on the difficulty of the Elliptic Curve Discrete Logarithm Problem [28]. Given points P and $Q = cP$ on the curve (where $c \in \mathbb{F}_p$ is a scalar and cP is P summed c many times), it is computationally hard to find the value c.

[†]Unlike biological neurons in which electrical signals propagate continuously, perceptrons have discrete propagation cycles.

[‡]The best known complexity for multiplying two square matrices is approximately $O(n^{2.37286})$, found in 2014 [25].

2 PRIVACY-ENHANCING TECHNIQUES, PROBLEMS, AND NEW DEVELOPMENTS

2.1 Problems

ZKPs are computationally expensive and require large storage overhead. Even the most modern techniques will struggle in large models with thousands if not millions of parameters. Groth16, a SNARK developed in 2016 [17] had a total setup and proving time of 23 years for VGG16 – a CNN with only a handful of layers. In addition, the storage overhead required was thousands of terabytes.

For CNNs, the feature extraction layers are responsible for taking into account the spatial relationships in the input image, as well as doing heavy lifting by offloading computation from the fully connected layer. When implementing a SNARK for a CNN, it's these feature extraction operations (convolution specifically) which are a bottleneck.

2.2 zk-SNARKs

SNARKs are a particular scheme of ZKPs. They are currently in use in applications such as ZeroCash [4]. SNARKs rely on building arithmetic circuits C or directed acyclic graphs consisting only of affine equations where the parameters are over a finite field. SNARKs, built upon NARKs, consist of a preprocessing or setup phase S, a proving algorithm P, and a verification algorithm V. The setup S generates public parameters for both the prover and verifier p_p and v_p respectively.

The proving algorithm generates a proof π from the public parameter, a public statement, and a secret witness w known only to the prover: $P(p_p, x, w) = \pi$.

The proof π and the public statement x are sent to the verifier. The verifier runs the verification algorithm $V(v_p, x, \pi)$ that either accepts or rejects the proof and statement.

SNARK is an acronym for Succinct Non-interactive ARguments of Knowledge which describes its features. SNARKs are succinct in that the proof should be short, and verification should be quick. That is a proof π is sublinear to the size of the witness w and the verification time is sublinear to the size of the circuit C and linear to the size of the public statement x. They are non-interactive, meaning a prover and verifier do not need to communicate with each other beyond the exchange of the proof. Arguments of Knowledge are simply a statement or set of statements.

The 'zk' in zk-SNARKs stands for zero-knowledge. This addition designates that the SNARK does not leak any information. Zk-SNARKs follow the three properties of ZKPs: completeness, soundness, and zero knowledge. They additionally satisfy knowledge soundness, defined as follows: if a prover successfully convinces a verifier, then that prover knows a private input [24].

SNARKs come in different flavors such as Groth16 [17], PLONK (more recently) [13], and Marlin [8]. Each of these utilizes variations in mathematical techniques and modifications in the trusted setup.

2.3 Other Frameworks (zk-STARKs)

Scalable Transparent Arguments of Knowledge (STARKs) are a class of cryptography based ZKPs differentiating themselves from other proof systems by their reliance on minimal cryptographic assumptions along with eliminating the need for a trusted setup, while maintaining the privacy and verification properties.

STARKs are characterized by two primary distinguishing features: transparency and scalability. They require no trusted setup, deriving security solely from public-key cryptography-based collision-resistant hash functions and publicly verifiable randomness, making them highly transparent [3]. In zk-STARKs, proof generation time increases efficiently with the size of the underlying computation, while verification time remains asymptotically low, regardless of computational complexity. Moreover, the proof size stays relatively compact compared to the computation size, enabling high scalability suitable for verifying large-scale computations [6].

The property that gives STARKs an edge over SNARKs is their quantum resistance. This is achieved because STARKs primarily rely on the collision resistance of chosen hash functions, assuming that standard cryptographic hash functions (specifically Merkle hash trees) resist attacks by quantum algorithms (like Shor's Algorithm), enabling STARKs to create post-quantum secure zero-knowledge proofs (ZKPs). In contrast, SNARKs typically depend on elliptic curve cryptography (ECC) or other constructs vulnerable to quantum attacks, which could easily break these public-key cryptosystems by efficiently solving the Elliptic Curve Discrete Logarithmic Problem (ECDLP) that ECC relies on for security [30] [5].

SNARKs are desirable for blockchains and are actively being integrated into them. Cryptocurrencies benefit from SNARKs as they provide proof of transaction without revealing all the details such as who was involved, or the amounts being transferred. This maintains the integrity of the blockchain, that way no digital currency can be created out of thin air. Improvements specifically towards blockchain zero knowledge proofs are being pursued such as zk-rollups, which can effectively batch process multiple ZKPs at once [10].

Features	**SNARK**	**STARK**
Trusted Setup	Required (per-circuit) [5]	Not Required
Transparency	No (Since it requires trusted setup)	Yes
Post-Quantum Security	No (ECC vulnerable to Shor's Algorithm) [5]	Yes (Hash-based)
Proof Size	~100s of bytes	~100s of KB
Verification Time	Very Fast	Fast
Prover Time	Fast	Slower (But Scalable)
Scalability	Moderate	Excellent (Scales Logarithmically)
Cryptographic Assumptions	ECC Curve, Pairing-based	Collision-resistant hash function
Quantum Resistance	No	Yes

Table 1: Comparison of zk-SNARK and zk-STARKs

2.4 Circom Example

In order to utilize the SNARK architecture, the calculations involved must be transformed into a circuit. Typically, a high level language such as ZoKrates, or Circom is used to create the arithmetic circuit.

Computations must be performed over the field $\mathbb{F}_p$, and in the case of the curve BN128 used in Circom, the prime p is 254 bits. Therefore, *quantization schemes* exist to ensure values are properly transformed or scaled to integer values. This section illustrates this process by implementing a Groth16 SNARK circuit with Circom using the `snarkjs` Node library and a simple single perceptron example.

Suppose you are a tomato farmer who wants to prove to a skeptical buyer that a specific tomato plant is ready for transplanting based on a proprietary calculation, without revealing the exact measurements (number of leaves, days planted) or the specific weights used in your formula. You only want to reveal whether the calculated score meets a publicly agreed-upon threshold. The farmer creates a miniature "neural network" consisting of a single neuron which outputs 1 or 0, classifying tomato plants as ready to transplant or not. The following Circom code models this scenario.

```
1  // tomato.circom
2  pragma circom 2.2.2;
3
4  include "../node_modules/circomlib/circuits/comparators.circom";
5
6  template TomatoTransplant() {
7      // Private inputs
8      signal input numLeaves;
9      signal input daysPlanted;
10     signal input w1; // weight 1
11     signal input w2; // weight 2
12     signal input b;  // bias
13
14     // Public input
15     signal input threshold;
16
17     // Public Output
18     signal output isReady;
19
20     // Compute weighted sum (perceptron)
21     signal term1 <== numLeaves*w1;
22     signal term2 <== daysPlanted*w2;
23     signal score <== term1 + term2 + b;
24
25     // Check if score > threshold
26     component gt = GreaterThan(252);
27     gt.in[0] <== score;
28     gt.in[1] <== threshold;
29     isReady <== gt.out; // 1 if score > threshold, 0 otherwise
30 }
31
32 component main {public [threshold]} = TomatoTransplant();
```

Listing 1: Tomato Transplant Circuit in Circom

Groth16 requires that all logical operations must be converted into arithmetic expressions. This process involves representing the circuit as a Directed Acyclic Graph (DAG) and then converting it to a Rank-1 Constraint System (R1CS). In the Circom code `tomato.circom`, lines 21-23 split the score calculation into `term1` and `term2`. This is because Groth16 requires constraints to be at most quadratic. Directly calculating

```
score = numLeaves*w1 + daysPlanted*w2 + b
```

involves the sum of two quadratic terms, which cannot be represented by a single quadratic constraint, resulting in the error:

```
error[T3001]: Non quadratic constraints are not allowed!
    "tomato.circom":20:5
    signal score <== numLeaves*w1 + daysPlanted*w2 + b;
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ found here
   = call trace:
     ->TomatoTransplant
```

The next step is to compile the Circom code into WebAssembly, to be used in the `snarkjs` library which implements the Groth16 SNARK system.

```
$ circom tomato.circom --r1cs --wasm --sym
  template instances: 4
  non-linear constraints: 255
  linear constraints: 4
  public inputs: 1
  private inputs: 5
  public outputs: 1
  wires: 265
  labels: 271
  Written successfully: ./tomato.r1cs
  Written successfully: ./tomato.sym
  Written successfully: ./tomato_js/tomato.wasm
  Everything went okay
```

Note that three files are produced as output, the WebAssembly file `tomato_js/tomato.wasm`, which `snarkjs` uses to generate the witness, an R1CS file `tomato.r1cs` which contains the circuit's constraints, and the symbols file `tomato.sym` which provides symbolic information about the circuit (for debugging and analysis). The WebAssembly file, along with the utility script `generate_witness.js` (also generated during compilation), enables the creation of a witness. Recall that the witness contains a set of values for all the signals within the circuit, including the private inputs (`w1`, `w2`, `b`, `numLeaves`, `daysPlanted`) and the intermediate calculation results (`term1`, `term2`). This witness serves as proof that the farmer possesses valid private inputs that satisfy the circuit's constraints and lead to a specific output.

Following this compilation, the next step involves utilizing `snarkjs` to perform a *trusted setup* [2]. This ceremony generates the proving and verification keys specific to this circuit. First, a "powers of tau" ceremony (which generates a universal reference string that can be reused across circuits) is conducted:

```
$ npx snarkjs powersoftau new bn128 12 pot12_0000.ptau -v
$ npx snarkjs powersoftau contribute pot12_0000.ptau \
  pot12_0001.ptau  --name="Farmer's contribution" \
  -v -e="random entropy"
$ npx snarkjs powersoftau prepare phase2 pot12_0001.ptau \
  pot12_final.ptau -v
```

Then a circuit-specific setup is performed using the compiled `tomato.r1cs`.

```
$ npx snarkjs groth16 setup circuits/tomato.r1cs \
  pot12_final.ptau tomato_0000.zkey
[INFO]  snarkJS: Reading r1cs
[INFO]  snarkJS: Reading tauG1
[INFO]  snarkJS: Reading tauG2
[INFO]  snarkJS: Reading alphatauG1
[INFO]  snarkJS: Reading betatauG1
[INFO]  snarkJS: Circuit hash:
              2291e69e 31baf722 a007de1e 3e6a2114
              15b4d78d 7f6611b8 cb1863ef 0aeb967b
              c2076998 4af20c62 ca207677 98612065
              fd416acf ac57421f f026dc10 56a79945
$ npx snarkjs zkey contribute tomato_0000.zkey \
  tomato_0001.zkey --name="Farmer's contribution" \
  -v -e="more random entropy"
[INFO]  snarkJS: EXPORT VERIFICATION KEY STARTED
[INFO]  snarkJS: > Detected protocol: groth16
[INFO]  snarkJS: EXPORT VERIFICATION KEY FINISHED
```

These commands produce the proving key `tomato_0001.zkey` and verification key `verification_key.json`.

Subsequently, the tomato farmer can use these keys along with their private inputs to generate a witness. This witness, combined with the proving key and public inputs, allows for the creation of a Groth16 proof. The farmer creates an input file:

```
{
    "numLeaves": "5",
    "daysPlanted": "10",
    "w1": "2",
    "w2": "3",
    "b": "1",
    "threshold": "30"
}
```

Listing 2: Farmer's secret input.

Next, the witness is generated:

```
$ node circuits/tomato_js/generate_witness.js \
  circuits/tomato_js/tomato.wasm \
  inputs/farmer_input.json witness.wtns
```

Finally, a proof is generated:

```
$ npx snarkjs groth16 prove tomato_0001.zkey \
  witness.wtns proof.json public.json
```

This produces `proof.json` (the proof) and `public.json` (public signals: `threshold` and `isReady`).

The skeptical tomato plant buyer can use the verification key `verification_key.json` and the public signals `public.json` to verify the proof,

```
$ npx snarkjs groth16 verify verification_key.json \
  public.json proof.json
[INFO]  snarkJS: OK!
```

confirming the tomato plant's readiness without needing to know the farmer's private measurements or formula.

3 STATE-OF-THE-ART

This section gives a brief overview of three recent papers, outlining how the authors used the new developments in section 2. The focus of each paper is to optimize techniques regarding AI model validation in CNNs. This verification ensures that a potentially malicious server has performed the CNN classification correctly. Recall the Circom example from section 2.4, which illustrated how a single neuron's computation can be expressed as a circuit for verification using zk-SNARKs. While this example demonstrates the basic principle, scaling this approach to an entire CNN comprising numerous layers of complicated operations such as convolution and non-linear activations, is a challenge. Fortunately, each operation in the CNN can be expressed as a distinct set of constraints, allowing for proof generation and verification at each layer of the network.

3.1 vCNN

Verifiable CNNs (vCNN) is a novel approach proposed by Lee et al [26] that aims to improve efficiency in producing ZKPs in CNNs. They utilize quadratic arithmetic programs (QAP) to optimize convolutions, and quadratic polynomial programs (QPP) to optimize other components (rectified linear unit (ReLU) and Pooling). Linking the two together with CP-SNARKs (Commit and Prove), they achieve large improvements in setup time & proving time ($\approx$ 18 hours), and common reference string (CRS) size ($\approx$ 80 GB) in comparison to Groth16 and VGG16. Verification time remains the same across all systems.

3.2 validCNN

The 2024 paper *ValidCNN: A Large-Scale CNN Predictive Integrity Verification Scheme Based on zk-SNARK* by Fan et al. presents drastic improvements over vCNN. They accomplish this by targeting the preprocessing steps. Specifically, they use an `im2col` algorithm [21] to convert the convolution operations into matrix multiplication, then use an updated version of Freivald's algorithm [22] to speed up the convolution operation bottleneck. This improved Freivald's algortihm is deterministic; it doesn't need to be run multiple times to drive down the error rate as described in our section 1.5.

The validCNN implementation writes zk-SNARK circuits in C++, while pre-computation steps are handled with Python. In their experiment, they utilized the LeNet-5 and VGG16 CNNs with the MNIST and CIFAR-10 datasets. The results from the CIFAR-10 dataset are summarized in table 2.

	LeNet-5	VGG16
Setup time	22s	359s
Proof time	10s	194s
Verify time	55ms	220ms
CRS size	37.3MB	400.1MB
Proof size	0.87KB	2.61KB

Table 2: Comparison of validCNN applied to LeNet-5 and VGG16 across various metrics.

3.3 psvCNN

Fan et al. [12] have developed a novel solution, called psvCNN, to reduce storage and time overhead in CNNs. The primary improvement arises from dividing inter and intra layers into individually independent task blocks. They were able to divide layers, specifically the convolutional layers and fully connected layers as those are the most computationally challenging. Their work involved parallelizing the kernel functions in the convolutional layers of the network. Since each kernel's calculations are independent from one another, they can be done in parallel to efficiently extract features. From each block, they extract the parameters needed for the construction of a circuit. This, however, raises a problem of integrity. If individual blocks are incorrectly calculated, this may result in an incorrect ZKP, and/or an incorrect prediction. The researchers resolve this by using an integrity check by use of a hashing algorithm described in [11], which verifies the cumulative hash of individual blocks with the hash of the entire layer. Another disruption arises when dealing with inter-layer divisions. The data which is passed on from layer to layer may be private requiring a ZKP circuit to be constructed. If the data is considered public, the hashing algorithm is applied similarly as inter layer division. To ensure correctness of the prediction and reduce computational overhead for both the prover and verifier, they utilize an improved Freivald's algorithm. As mentioned earlier, the convolutional and fully connected layers are the most computationally intensive layers, this is due to the large amount of matrix multiplications. The improved Freivald's algorithm reduces the calculation to one round by constructing a random matrix r, instead of a vector, to compute $ABr = Cr$. They verified psvCNN satisfied the three properties of ZKPs – completeness, soundness, and zero-knowledge.

In their experiment, they utilized the LeNet-5 and VGG16 CNNs with the CIFAR-10 dataset. The results from psvCNN utilizing Freivald's improved algorithm are summarized in table 3.

	LeNet-5	VGG16
Setup time	1.09s	11.26s
Proof time	0.78s	7.65s
Verify time	10ms	68ms
CRS size	3.1MB	36.0MB
Proof size	8.8KB	1.2MB

Table 3: Comparison of psvCNN applied to LeNet-5 and VGG16 across various metrics.

4 THOUGHTS AND REMARKS

ZKPs are still a novel field, with advancements being made in the last 10 years. Despite the newfound traction, there are still gaps in research, namely the use of STARKs in CNNs. STARKs are an unlikely candidate for CNNs due to their large proof size, however it may be beneficial when a trusted setup is undesirable. A vast majority of research is focused on the use of ZKPs in the context of blockchain technologies and cryptocurrencies. It would be beneficial to see further research in other fields, such as e-voting, healthcare, and law. An area of possible conflict is the rising need for explainable artificial intelligence (XAI).

We noted that, While psvCNN provides faster execution times, validCNN requires less storage space.

Despite these substantial improvements in performance, there are still efficiencies to be had. In models with hundreds of layers, or millions of weights, this would still take too much time and space to construct circuits, proofs, and verifications that would be usable in real world scenarios. There is also interest at every level in improving and using ZKPs down to the hardware level [20, 31].

4.1 Proofs and Complexity

From a philosophical point of view, proofs are quintessentially human, a bridge between rationality and reality, which integrates the essences of Intelligence, Humanity, Love and aesthetics, as an effort to overcome the concrete ability to think to the abstract world of the imaginary that converges as a limit into the Arts [29].

Nondeterministic Polynomial time (NP) are class of decision problems with yes/no answer, if the answer is yes then there is a proof of a certificate that a verifier can confirm its validity in polynomial time. This has a big implication on the foundation of cryptosystems based on this property, providing a framework that can be used to prove membership to a particular domain of definition, when we can verify this without revealing any other fact other than the statement is true, we have a Zero-Knowledge Proof System. In fact, all NP class problems have a have Zero-Knowledge Proof System [16]

Gödel's incompleteness theorems provide a powerful tool to analyze complex axiomatic systems. It proves that these systems can be either complete or incomplete but not both at the same time [23]. This theorem has significant implications in the fields of mathematics and logic. It has led to the development of primitives and fundamentals in the field of axiomatic proofs. These developments have contributed to rigorous formalization and the discovery of the inherent limitations of Gödel's Incompleteness Theorems. Computationally Sound Proofs (CS Proofs) on the other hand takes consideration of Computational Complexity. This is the classical concept to verify the true of a statement evolve from absolute logical soundness to what can be achieved efficiently [29].

CS proofs completeness means every (computationally) true statement has succinct proof that is easy to find. The concept of inconsistency in this field means that in principle there could be bad statements that could have proof. However, these bad proofs are computationally infeasible to find to the point that even an efficient adversary cannot even produce them. Note that the notion of a true theorem has not changed, what has evolved is the notion of what it means to prove that a theorem is true taking into consideration Computational Complexity [16].

All Nondeterministic Polynomial time (NP) problems can hold a ZKP system that can be translated into a SNARK framework. It will depend on the requirements of the cryptosystem and the kind of scheme that can be applied. For instance, PLONK is a SNARK scheme that can be used for applications that can handle one-time universal trust setup, or Bulletproof is another type of SNARK that non-trusted setup is needed. In this conceptual structure it is very important to remember that any zk-SNARK can be constructed starting with CS proof. Then add a non-deterministic compiler to optimize the prover and finally add zero-knowledge. This is translating the computationally sound (CS) proof into a zk-SNARK. This is a wonderful achievement because it allows us to have ZKP anytime, anywhere [15].

5 CONCLUSION

In conclusion, zero-knowledge proofs are a unique cryptographic tool that can ensure both individual user privacy and correctness. This is particularly important in AI models, where we aim to maintain privacy for sensitive parameters in the model while ensuring the correctness of the calculations performed by the model in its decisions.

SNARK is a cryptographic scheme whose genesis lies in Computational Sound Proofs, that ensures that an NP statement can be interactively proven with completeness and computational soundness. These proofs are deliberately complete and inconsistent - a notion that would be unacceptable in the Gödelian world. However, when we consider Computational Complexity, it is evident that the classical approach of proof systems of Gödel is not a final statement and is not free to be influenced by historical context, indeed, the proof concept has evolved into what can be proven efficiently in the cryptographic field, considering its Computational Complexity.

REFERENCES

[1] Paulo S. L. M. Barreto and Michael Naehrig. 2006. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography – SAC 2005 (Lecture Notes in Computer Science, Vol. 3897)*, Bart Preneel and Stafford Tavares (Eds.). Springer-Verlag Berlin/Heidelberg, 319–331. http://cryptojedi.org/papers/#pfcpo.

[2] Jordi Baylina. 2019. snarkjs. https://github.com/iden3/snarkjs/blob/master/README.md.

[3] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. 2018. *Scalable, transparent, and post-quantum secure computational integrity.* Technical Report 2018/046. IACR Cryptology ePrint Archive. https://eprint.iacr.org/2018/046 Report 2018/046.

[4] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 459–474. https://doi.org/10.1109/SP.2014.36

[5] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*. 459–474. https://doi.org/10.1109/SP.2014.36

[6] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. 2016. Interactive Oracle Proofs. Cryptology ePrint Archive, Paper 2016/116. https://eprint.iacr.org/2016/116

[7] Jean-Luc Beuchat, Jorge Enrique González Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. 2010. High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves. Cryptology ePrint Archive, Paper 2010/354. https://eprint.iacr.org/2010/354

[8] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas Ward. 2019. Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS. Cryptology ePrint Archive, Paper 2019/1047. https://eprint.iacr.org/2019/1047

[9] Computerphile. 2017. Perceptron - Computerphile. https://www.youtube.com/watch?v=pj9-rr1wDhM

[10] Ethereum. 2024. Zero-Knowledge Rollups. https://ethereum.org/en/developers/docs/scaling/zk-rollups/.

[11] Y. Fan, X. Lin, G. Tan, Y. Zhang, W. Dong, and J. Lei. 2019. One secure data integrity verification scheme for cloud storage. *Future Generation Computer Systems* 96 (2019), 376–385.

[12] Yongkai Fan, Binyuan Xu, Linlin Zhang, Gang Tan, Shui Yu, Kuan-Ching Li, and Albert Zomaya. 2024. psvCNN: A Zero-Knowledge CNN Prediction Integrity Verification Strategy. *IEEE Transactions on Cloud Computing* 12, 2 (2024), 359–369. https://doi.org/10.1109/TCC.2024.3350233

[13] Ariel Gabizon, Zachary J. Williamson, and Oana-Madalina Ciobotaru. 2019. PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge. *IACR Cryptol. ePrint Arch.* 2019 (2019), 953. https://api.semanticscholar.org/CorpusID:201685538

[14] Oded Goldreich. 2003. From Weizmann's annual report 2003. https://www.wisdom.weizmann.ac.il/~/oded/poster03.html. Accessed: 2025-02-20.

[15] Oded Goldreich. 2019. *Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali.* ACM.

[16] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1991. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM (JACM)* 38, 3 (1991), 690–728.

[17] Jens Groth. 2016. On the Size of Pairing-based Non-interactive Arguments. Cryptology ePrint Archive, Paper 2016/260. https://eprint.iacr.org/2016/260

[18] Dan Hendrycks, Mantas Mazeika, and Thomas Woodside. 2023. An Overview of Catastrophic AI Risks. arXiv:2306.12001 [cs.CY] https://arxiv.org/abs/2306.12001

[19] Sepp Hochreiter. 1998. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (04 1998), 107–116. https://doi.org/10.1142/S0218488598000094

[20] Ingonyama. 2024. Ingonyama ZK GitHub Repository. https://github.com/ingonyama-zk.

[21] Hyungjun Kim, Taesu Kim, Jinseok Kim, and Jae-Joon Kim. 2018. Deep Neural Network Optimized to Resistive Memory with Nonlinear Current-Voltage Characteristics. *J. Emerg. Technol. Comput. Syst.* 14, 2, Article 15 (July 2018), 17 pages. https://doi.org/10.1145/3145478

[22] Ivan Korec and Jiří Wiedermann. 2014. Deterministic Verification of Integer Matrix Multiplication in Quadratic Time. In *SOFSEM 2014: Theory and Practice of Computer Science*, Viliam Geffert, Bart Preneel, Branislav Rovan, Július Štuller, and A. Min Tjoa (Eds.). Springer International Publishing, Cham, 375–382.

[23] Georg Kreisel. 1980. Kurt Gödel, 28 April 1906-14 January 1978.

[24] Ryan Lavin, Xuekai Liu, Hardhik Mohanty, Logan Norman, Giovanni Zaarour, and Bhaskar Krishnamachari. 2024. A Survey on the Applications of Zero-Knowledge Proofs. arXiv:2408.00243 [cs.CR] https://arxiv.org/abs/2408.00243

[25] François Le Gall. 2014. Powers of tensors and fast matrix multiplication. *Proceedings of the 39th international symposium on symbolic and algebraic computation (ISSAC 2014)* (2014), 296–303.

[26] Seunghwa Lee, Hankyung Ko, Jihye Kim, and Hyunok Oh. 2024. vCNN: Verifiable Convolutional Neural Network Based on zk-SNARKs. *IEEE Transactions on Dependable and Secure Computing* 21, 4 (2024), 4254–4270. https://doi.org/10.1109/TDSC.2023.3348760

[27] Haochen Liu, Yiqi Wang, Wenqi Fan, Xiaorui Liu, Yaxin Li, Shaili Jain, Yunhao Liu, Anil Jain, and Jiliang Tang. 2022. Trustworthy AI: A Computational Perspective. *ACM Transactions on Intelligent Systems and Technology* 14, 1 (Nov 2022), 1–59. https://doi.org/10.1145/3546872

[28] Kevin S. McCurley. 1990. The Discrete Logarithm Problem. *Proceedings of Symposia in Applied Mathematics* 42 (1990), 49–74. https://www.mccurley.org/papers/dlog.pdf

[29] S. Micali. 1994. CS proofs. In *Proceedings 35th Annual Symposium on Foundations of Computer Science.* 436–453. https://doi.org/10.1109/SFCS.1994.365746

[30] Peter W. Shor. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science.* 124–134.

[31] Supranational. 2024. Supranational. https://www.supranational.net/.

[32] Harini Suresh and John Guttag. 2021. A Framework for Understanding Sources of Harm throughout the Machine Learning Life Cycle. *Equity and Access in Algorithms, Mechanisms, and Optimization* (Oct 2021). https://doi.org/10.1145/3465416.3483305

A APPENDIX

Output code from Circom tomato plant example.

verification_key.json

```
{
 "protocol": "groth16",
 "curve": "bn128",
 "nPublic": 2,
 "vk_alpha_1": [
  "10569352019195793314114348506664292665
     0492493363832251131060499697004217902",
  "115391304991829357166180413567787030759
     37718363585909295304175378025389807048",
  "1"
 ],
 "vk_beta_2": [
  [
   "10423332362294020433511884857365027715
     377824157866888522940321211336572501250",
   "69176907469373842237266813206220822511
     44846969844842357154354159829775583419"
  ],
  [
   "86376566984482313406549498517646892115
     80168181103111379404436219071730176594",
   "12909760299356435929310987006298689940
     484624870960974700803078242463647502428"
  ],
  [
   "1",
   "0"
  ]
 ],
 "vk_gamma_2": [
  [
   "108570469990230571359445707622328294813
     70756359578518086990519993285655852781",
   "115597320329863871079910040213922857839
     25812861821192530917403151452391805634"
  ],
  [
   "849565392312343141760497324748927243841
     8190587263600148770280649306958101930",
   "408236787586343368133220340314543556831
     6851327593401208105741076214120093531"
  ],
  [
   "1",
   "0"
  ]
 ],
 "vk_delta_2": [
  [
   "198993844156512181782132421361771237252
     14479321046353511858798420351372781048",
   "550150101186258339432614465331849074787
     269663586572788368736017660989284104"
  ],
  [
   "152257760916273955691374398451304731390
     05190034594823272793294131875358964318",
   "561372023993925472184494061902148025904
     5435501364387591338801750102685867701"
  ],
  [
   "1",
   "0"
  ]
 ],
 "vk_alphabeta_12": [
  [
   [
    "17061189762986215824618659893439847900
     838832205402978217599976376124267001862",
    "19972364932363633806899930538016869410
     927385506146089740343994550508848284121"
   ],
   [
    "10244707815730863594960836147095939109
     510506226055308286889769172259272750003",
    "88809829587094047127960003390775202751
     2990227652927665864631742917774719478"
   ],
   [
    "18689271379161322392986238389264757862
     172911227105892412121289548923714128079",
    "13606233492167350233872229269722844893
     09854758911529626233470533637666580452"
   ]
  ],
  [
   [
    "11468819978010350784992616656641705113
     727336652977635428466052682049142325907",
    "19323038337059905636333823237116553015
     401112081072074727984655113778971403230"
   ],
   [
    "83613463882195594667695358616113773700
     66996793512368652773257903780058157211",
    "31347542996080891216297856853633596913
     99501374535538241687893421023483954882"
   ],
   [
    "31115343464799502658883855524871871495
     25969470746983643404409117345873134628",
    "92797612363775515524550799610615390604
     38973917664167299452021983671237222732"
   ]
  ]
 ],
 "IC": [
  [
   "88024587604872116984268625195449679103905258675000188807042916556531381 96197",
```

```
  [
   "8802458760487211698426862519544967910390525867500018880704291655653138196197",
   "1572683764510791502886108004393384590872799934696735862771414973883230902824 4",
   "1"
  ],
  [
   "2132629635259999683277682622917162924063931040917912457335780914680530362126 9",
   "7383953279935953384771677849551337065665489046196047609123158741541351209050",
   "1"
  ],
  [
   "1925769171650894795852968504278451721181776704215276359798056466943104736847 0",
   "9777777398939682387560614796308167722088288626041114280488763947858539358557",
   "1"
  ]
 ]
}
```

public.json

```
[
 "1",
 "30"
]
```

proof.json

```
{
 "pi_a": [
  "1288319701085257895109360148015430176698971496864444473295576770983174012759 6",
  "1927723869559294591107320514128532553245522319147582113046149703103450449203 3",
  "1"
 ],
 "pi_b": [
  [
   "1207031397474764285410421414281425744502112800077036728322893546888346980582 5",
   "1974744228759342034875136709569548039578947408350530784413293060158988926486 4"
  ],
  [
   "5904522416649542474486156573111956024644180459475208238398876686504122174235",
   "1401326563270943826457474870073130082077042713212242493378730057199192650389 5"
  ],
  [
   "1",
   "0"
  ]
 ],
 "pi_c": [
  "8215962667901302570031915871323566141014983223170777979586634785011761261333",
  "9003071391767135394349973144347335788290909362160032293214983255932949453799",
  "1"
 ],
 "protocol": "groth16",
 "curve": "bn128"
}
```