

Hydra: Succinct Pipelineable Interactive Arguments of Knowledge for Verifiable Neural Networks

ABSTRACT

We present Hydra, a novel verifiable computation system for neural networks. In this environment, a client (verifier) requests a server (prover) to evaluate a computation represented as an arithmetic circuit. The server returns the result of said computation, as well as a proof that attains to its correctness. Hydra introduces two new disjoint interactive argument scheme protocols geared towards the efficient pipelining of general arithmetic circuit verification, where layers of the circuit can be proven asynchronously. Compared to non-interactive SNARKs which rely on either knowledge type assumptions or the Random Oracle model and theoretical non-interactive arguments based on standard assumptions that are not useful in practice, Hydra achieves a sweet spot with a practical approach. From standard assumptions, we collapse the round complexity to polylogarithmic to the width of the circuit, but only incur polylogarithmic blowup in bandwidth and verifier time complexity. We demonstrate the effectiveness of this protocol in general cloud computing environments, and also apply this protocol in the specific context of cloud-based deep neural network verification. We propose a new interactive neural network quantization algorithm to convert a neural network into a provable arithmetic circuit representation, and leverage Hydra's pipelined approach to efficiently verify neural networks composed of many layers in both inference and training. We obtain protocol time efficiency improvements of up to 34.8× for general computation while also reaching state of the art accuracy (99.5%) on MNIST for our neural network framework.

ACM Reference Format:

. 2021. Hydra: Succinct Pipelineable Interactive Arguments of Knowledge for Verifiable Neural Networks. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

1 INTRODUCTION

Deep learning with neural networks, especially in regards to convolutional neural networks (CNNs), has grown prominent in recent years with a wide variety of growing applications such as image recognition and classification. A catalyst to this significant popularization of deep learning is cloud computing. As training deep learning models and performing inferences with them are computationally expensive, cloud service providers allow for the delegation

of such tasks to a computationally powerful machine at an economical cost.

However, this environment motivates the need for security and trust. Outsourcing the computation arises the need for the efficient verification that the cloud has indeed performed the training/inference correctly. In cases where the cloud is "lazy" and perhaps uses a simpler neural network to perform and return inference results, or even worse maliciously tampers with the training and/or inference results to bypass security checks, it is evident that frameworks are needed to verify such actions with neural networks and hold these cloud services accountable.

With the rise of cloud based computing, the demand for security and trust in the accurate execution of a delegated task has grown prominent. This has motivated the field of verifiable computation and the concept of an *interactive proof*, where a computationally weak verifier (denoted as \mathcal{V} , acting as a client) can be efficiently convinced to a high degree of probability that a computationally unbounded prover (denoted as \mathcal{P} , acting as a server) has correctly computed a requested task. The key property in this notion is that \mathcal{V} is able to check the proof of correctness that \mathcal{P} provides in asymptotically less time than it would take to recompute said task by itself, thus resulting in an advantageous benefit for \mathcal{V} in outsourcing the computation.

Interactive proofs were first introduced by [2] in the 1980's and have been discussed in the literature for quite some time, leading to important advances in cryptography and complexity theory such as the celebrated proof of $IP=PSPACE$ [5]. However, they were largely considered to be merely of theoretical promise and not applicable for production or practical use in any sense, mainly due to the incredibly high time and computation overhead required to generate such proofs. It only recently that work to refine and scale these theoretical concepts for practical use became prevalent.

Particularly, a significant amount of such research has been based on the seminal general purpose interactive proof due to Goldwasser, Kalai, and Rothblum [8] (henceforth referred to as GKR or the GKR protocol). In this environment, the prover convinces the verifier of the validity of an arithmetic circuit evaluation composed of addition and multiplication gates of fan-in size two over some finite field. A notable property of their powerful protocol is that it is doubly efficient, meaning that the prover runs in polynomial time in the circuit size and the verifier runs in time sublinear to the circuit size. Despite this progress, the underlying GKR protocol runtime as well as the round complexity still remain entirely dependent on the depth of the circuit, as the protocol is fundamentally based on a layer-by-layer proof approach to ensuring the validity of the output. This quickly becomes a major bottleneck for a variety of deep circuits comprised of many layers, where the protocol efficiency begins to seriously degrade. Unfortunately, parallelism cannot be exploited naively across the depth of the circuit to account for this problem because at a high level, it would expose information about

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA

other layers while they are being proved, which would destroy the security of the protocol. As a result, GKR-based proofs have been generally considered to be impractical in such instances with deep circuits and restricted only to circuits of shallow constant-bounded depth.

In this work, we contribute theoretical proposals and practical improvements which come together to form a complete, robust framework that has wide applicability. In summary, this paper introduces the following succinct¹ interactive argument protocols for general cloud computation:

- The subcircuit protocol. This argument scheme allows for a large, deep circuit to be split up depthwise into multiple smaller data-dependent subcircuits. Each of these subcircuits are then proved in parallel and the proofs are combined at the end. With this protocol, a streaming upload system can be utilized where the proving process begins during the subcircuit uploading process.
- The Hydra protocol. This allows for the complete layer-wise independent parallelization of the GKR protocol. This argument scheme removes the need for a sequential layer-by-layer evaluation proof so that all layers can be pipelined at once. Hydra achieves a sweet spot between non-interactive SNARKs that rely on non-standard assumptions such as the Fiat-Shamir heuristic and theoretical non-interactive proofs based on standard assumptions that are not applicable in practice. From standard assumptions, we reduce round complexity of the GKR protocol to polylogarithmic to the width of the circuit, but only incur polylogarithmic increase to the bandwidth and verifier complexity.

We provide a full implementation of both protocols. Through our experimental evaluations, we show efficiency improvements of up to 34.8× for the subcircuit protocol and 4.3× for the Hydra protocol on parallelized hardware.

We then apply our pipelined verification approach in the context of secure neural network verification. As the architecture of deep neural networks reflect that of deep arithmetic circuits, we propose

- A novel neural network quantization algorithm. As the weights and biases in neural networks are floating-point and the arithmetic circuits used in verifiable computing only accept field elements in some finite field \mathbb{F} , an algorithm is needed to convert a traditional neural network into a provable arithmetic circuit. Neural network quantization is typically used for hardware acceleration (see [37] for a survey of existing methods) and not verifiable computing. Here we propose, to the best of our knowledge, the first *provable* interactive quantization algorithm for verifiable computation of neural networks *intra-training*. With this algorithm, the neural network can be efficiently quantized by separating the network into subnetworks using a bottom-up approach. As the protocol proceeds up subnetworks of the CNN, not only will they be efficiently quantized, but proofs attaining to the correctness of previous subnetworks will also be verified.
- A pipelined protocol of the Hydra concurrent layer proving technique applied to the quantized neural network. Our approach, to the best of our knowledge, is the first time such

GKR-based verifiable computation scheme is efficiently able to verify ultra-deep neural networks, where hundreds or even thousands of layers can be verified concurrently.

We provide a full implementation of the quantization algorithm and the verifiable computation framework. We show our empirical results with verification efficiency and performance details/analyses on various compositions of neural networks, and demonstrate state-of-the-art performance on benchmark classification tasks for MNIST.

2 RELATED WORKS

Significant advancements have been made to GKR protocol, taking it to near practicality. The prover runtime was reduced to $O(|C|\log|C|)$ in [10], close to linear for specific circuit compositions in [13], and eventually reduced to where it is linear $O(|C|)$ with respect to the size of the circuit in [31]. In a practical setting, the utilization of parallelism has additionally introduced great speedups for the proving process within individual layers of the circuit [11], across data-independent parallel circuits [21], and in different sub-copies [30].

Also, cryptographic primitives have been used with the GKR protocol for further improvements. The Fiat-Shamir heuristic [3] is commonly used to convert interactive proofs such as GKR to non-interactive arguments that can be achieved with a single round protocol. However, it is important to note that these arguments are based on non-standard assumptions (either the Random Oracle model [6] or knowledge type assumptions) that are non-falsifiable, and have been proven insecure by [7] in the absence of a random oracle. In addition, polynomial commitments and other primitives are also utilized alongside GKR as the base for efficient argument schemes [16] [22] [26] [29] [31] [34], especially ones that guarantee zero-knowledge (zk-SNARKs). These zero-knowledge schemes are applicable in authentication systems and blockchains where \mathcal{P} wants to prove to \mathcal{V} that it knows some value x without revealing any information about x itself. One important aspect of these schemes is that they all rely on similar Fiat-Shamir transformations to convert them into non-interactive protocols. In fact, it is known due to [9] that one cannot construct such SNARKs with constant $O(1)$ round complexity from standard assumptions.

On the theoretical side, the latest state of the art non-interactive protocol from standard assumptions is due to [28], with a protocol runtime of $O(\text{poly}(|C|))$.

Applications of these verifiable computation protocols to neural networks is still a very new and emerging niche. [27] first introduces this concept with SafetyNets, which proves neural network inferences based off of the GKR protocol. However, due to the inefficiency of the protocol for deep circuits, empirical testing only showed feasibility for inference verification on 2-3 layered CNNs. Other works [33] [35] [36] [38] relating to neural network inference verification are generally centered on zero-knowledge single-round variants, where the arithmetic circuit is hidden during the proving process. However, these protocols fall victim to the same consequences of relying upon non-standard assumptions, and also typically require heavy prover-side cryptographic compute costs.

¹“succinct” refers to polylogarithmic in the size of the circuit C

Table 1 provides a comparison of the Hydra protocol and existing verifiable computation systems. Compared to other state-of-the-art protocols, Hydra’s main advantage is a near constant round complexity from standard assumptions, while keeping reasonable prover and verifier time complexity.

	libSNARK ² [16]	Ligero [19]	Bulletproofs [26]
\mathcal{P}	$O(C \log C)$	$O(C \log C)$	$O(C)$
\mathcal{V}	$O(1)$	$O(C)$	$O(C)$
\mathcal{R}	$O(1)$	$O(\sqrt{C})$	$O(\log C)$
$ \pi $	$O(1)$	$O(\sqrt{C})$	$O(\log C)$
	Hyrax [29]	libSTARK [23]	Aurora [24]
\mathcal{P}	$O(C \log C)$	$O(C \log^2 C)$	$O(C \log C)$
\mathcal{V}	$O(\sqrt{n} + d \log C)$	$O(\log^2 C)$	$O(C)$
\mathcal{R}	$O(d \log C)$	$O(\log^2 C)$	$O(\log^2 C)$
$ \pi $	$O(\sqrt{n} + d \log C)$	$O(\log^2 C)$	$O(\log^2 C)$
	[28]	Libra [31]	Hydra ³
\mathcal{P}	$O(\text{poly}(C))$	$O(d \cdot n)$	$O(d \cdot n \log n)$
\mathcal{V}	$O((d+n) \cdot \text{polylog}(C))$	$O(d \log n)$	$O(d \log^2 n)$
\mathcal{R}	$O(1)$	$O(d \log n)$	$\tilde{O}(1)$
$ \pi $	$O(d \cdot \text{polylog}(C))$	$O(d \log n)$	$O(d \log^2 n)$

Table 1: Comparison of the Hydra protocol to existing state of the art proof systems, specifically the interactive versions of protocols that only rely on standard assumptions (e.g., discrete logarithm, bilinear maps, etc.). Notably, this does not include knowledge type assumptions and Fiat-Shamir heuristic under the Random Oracle model. \mathcal{P} , \mathcal{V} , \mathcal{R} , and $|\pi|$ are the prover time, verification time, round complexity, and proof size, respectively. C is the size of a logspace uniform circuit with depth d and width n .

3 BACKGROUND

3.1 Interactive Proofs

We formally define an interactive proof below. Given a function g , prover \mathcal{P} and a verifier \mathcal{V} , an interactive proof allows for \mathcal{P} to convince \mathcal{V} that $g(x) = y$ through a multi-round conversation, where x is an input given by \mathcal{V} , and y is the output claimed by \mathcal{P} . An interactive proof must satisfy two conditions as follows:

- **Completeness.** For every x such that $g(x) = y$, it holds that

$$\Pr[\langle \mathcal{P}, \mathcal{V} \rangle(x) = \text{accept}] = 1.$$

²requires an $O(C)$ per-statement trusted setup phase

³round complexity $O(\log n)$ is constant in the depth of C , logarithmic in the width of C

In other words, a prover who follows the protocol honestly will always convince the verifier of the validity of the function evaluation.

- **Soundness.** For any x with $g(x) \neq y$ and any malicious \mathcal{P}^* , it holds that

$$\Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle = \text{accept}] \leq \epsilon.$$

In other words, the verifier will only accept a cheating prover with probability less than equal to some ϵ . Formally, $\epsilon = \frac{1}{3}$, however, in practical settings this can be made arbitrarily small.

3.2 Sumcheck Protocol

The sumcheck protocol was introduced by [4], providing an interactive proof for the problem of summing a multivariate polynomial $f : \mathbb{F}^\ell \rightarrow \mathbb{F}$ over the Boolean hypercube:

$$\sum_{b_1, b_2, \dots, b_\ell \in \{0,1\}} f(b_1, b_2, \dots, b_\ell)$$

The protocol proceeds in ℓ rounds as follows, where in each round \mathcal{V} samples a random field element $r \in \mathbb{F}$, commonly referred to as a “coin toss”.

- (1) For the first round 1, \mathcal{P} sends the claimed summation value H and a univariate polynomial

$$f_1(x_1) := \sum_{b_2, \dots, b_\ell \in \{0,1\}} f(x_1, b_2, \dots, b_\ell)$$

\mathcal{V} checks if $H = f_1(0) + f_1(1)$. Afterwards, \mathcal{V} samples a random field element $r_1 \in \mathbb{F}$ and sends it to \mathcal{P} .

- (2) For every round j where $2 \leq j \leq \ell - 1$, \mathcal{P} sends a univariate polynomial

$$f_j(x_j) := \sum_{b_{j+1}, \dots, b_\ell \in \{0,1\}} f(r_1, \dots, r_{j-1}, x_j, b_{j+1}, \dots, b_\ell)$$

\mathcal{V} checks if $f_{j-1}(r_{j-1}) = f_j(0) + f_j(1)$ and sends another random field element $r_j \in \mathbb{F}$ to \mathcal{P} .

- (3) For the last round ℓ , \mathcal{P} sends a univariate polynomial

$$f_\ell(x_\ell) := f(r_1, r_2, \dots, r_{\ell-1}, x_\ell)$$

which \mathcal{V} then checks if $f_{\ell-1}(r_{\ell-1}) = f_\ell(0) + f_\ell(1)$. Then, \mathcal{V} samples another random field element $r_\ell \in \mathbb{F}$ that is not revealed to \mathcal{P} , and evaluates $f(r_1, r_2, \dots, r_\ell)$ on its own or with the help of an oracle. \mathcal{V} accepts if $f_\ell(r_\ell) = f(r_1, r_2, \dots, r_\ell)$. Otherwise, if any equality check does not hold during the protocol, \mathcal{V} rejects.

The sumcheck protocol is complete and has soundness $\epsilon = \frac{d\ell}{|\mathbb{F}|}$, where d is the total degree of f . The proof size is $O(d\ell)$ and the verifier time is also $O(d\ell)$.

3.3 GKR Protocol

As stated before, the GKR protocol [8] is a powerful interactive proof technique that allows for the efficient verifiable evaluation of circuit computations. From a bird’s-eye view, the protocol proceeds layer by layer, starting from the output layer and ending at the input layer. For each layer, \mathcal{P} gives a claim about the values in that layer i and reduces it to a claim about the values in the subsequent

layer $i+1$ through an instance of the sumcheck protocol on a certain polynomial. Since \mathcal{V} does not have access to the values in intermediate layers (computing those values would require evaluating the circuit, which is precisely what \mathcal{V} wants to avoid) \mathcal{V} cannot check these claims directly. Therefore, this cycle continues until \mathcal{P} gives a claim about the values in the input layer, which \mathcal{V} can check itself and conclude the protocol.

3.3.1 Notation. Here, we describe the notations that we will be utilizing throughout the rest of the paper. Formally, the GKR protocol environment is one where \mathcal{P} and \mathcal{V} agree on a layered logspace uniform arithmetic circuit C over a finite field \mathbb{F} , of fan-in size 2 and composed of addition and multiplication gates. The objective is for \mathcal{P} to convince \mathcal{V} that the circuit evaluation (the gate values at the output layer) is correctly computed.

Let d be the depth of the circuit, with layer 1 as the output layer and layer d as the input layer. We use S_i to denote the number of gates in the i -th layer, and $s_i := \log(S_i)$. Then, we define a function $W_i : \{0, 1\}^{s_i} \rightarrow \mathbb{F}$ that takes in a binary string $b \in \{0, 1\}^{s_i}$, which is the label of a gate at layer i , and returns the value of that gate. Thus, W_1 is the function for the values of the output layer, and W_d is the function for the values of the input layer.

We also define two more functions, add_i and $mult_i : \{0, 1\}^{s_i} \times \{0, 1\}^{s_{i+1}} \times \{0, 1\}^{s_{i+1}} \rightarrow \{0, 1\}$, known as the wiring predicate functions. In essence, these encode how the gates from level i are connected to the wires from level $i+1$. These functions take in a gate label a at level i and two gate labels b and c at level $i+1$. They will return 1 if the a gate takes in the values of the b and c gates, and the gate type of a is of the corresponding type to the function (addition for add_i , multiplication for $mult_i$). Otherwise, they will return 0.

With this in mind, we provide the expression for W_i as

$$W_i(x) = \sum_{(a,b) \in \{0,1\}^{2s_{i+1}}} [add_i(x, a, b)(W_{i+1}(a) + W_{i+1}(b)) + mult_i(x, a, b)(W_{i+1}(a) \cdot W_{i+1}(b))] \quad (1)$$

for $x \in \{0, 1\}^{s_i}$. As W_i is expressed as a summation, \mathcal{P} and \mathcal{V} can engage in a sumcheck protocol in order to verify the validity of a claimed evaluation. We can rewrite the expression as a polynomial in the field \mathbb{F} by use of *multilinear extensions*. Namely, given a function $f : \{0, 1\}^\ell \rightarrow \mathbb{F}$, the multilinear extension of a function is the unique polynomial $\tilde{f} : \mathbb{F}^\ell \rightarrow \mathbb{F}$ where $\tilde{f}(x) = f(x) \quad \forall x \in \{0, 1\}^\ell$ and the degree of the polynomial in each variable is at most 1. We represent this polynomial by the Lagrange interpolation as

$$\tilde{f}(x_1, x_2, \dots, x_\ell) = \sum_{b \in \{0,1\}^\ell} \prod_{i=1}^{\ell} [(1-x_i)(1-b_i) + x_i b_i] \cdot f(b)$$

Applying this to W_i , we get

$$\tilde{W}_i(x) = \sum_{(a,b) \in \{0,1\}^{2s_{i+1}}} \left[\widetilde{add}_i(x, a, b)(\tilde{W}_{i+1}(a) + \tilde{W}_{i+1}(b)) + \widetilde{mult}_i(x, a, b)(\tilde{W}_{i+1}(a) \cdot \tilde{W}_{i+1}(b)) \right] \quad (2)$$

where $x \in \mathbb{F}^{s_i}$.

3.3.2 Protocol. Let \mathbb{F} be a prime field and C be a layered arithmetic circuit with depth d . The protocol proceeds in d instances as described below.

- (1) \mathcal{P} first sends the claimed output of the circuit to \mathcal{V} . \mathcal{V} samples a random $x^{(1)} \in \mathbb{F}^{s_1}$ and sends it to \mathcal{P} . \mathcal{P} and \mathcal{V} both compute $\tilde{W}_1(x^{(1)})$. Note that $\tilde{W}_1(x^{(1)})$ is based on the claimed output values sent by \mathcal{P} . The following steps reduces the truthfulness of $\tilde{W}_1(x^{(1)})$ all the way to the truthfulness of the input layer, which is known by \mathcal{V} .
- (2) For all layers $1 \leq i \leq d$, \mathcal{P} and \mathcal{V} execute sumcheck over

$$\tilde{W}_i(x^{(i)}) = \sum_{(a,b) \in \{0,1\}^{2s_{i+1}}} \left[\widetilde{add}_i(x^{(i)}, a, b)(\tilde{W}_{i+1}(a) + \tilde{W}_{i+1}(b)) + \widetilde{mult}_i(x^{(i)}, a, b)(\tilde{W}_{i+1}(a) \cdot \tilde{W}_{i+1}(b)) \right] \quad (3)$$

Note that at the end of the sumcheck (the last round), \mathcal{V} cannot directly check the evaluation of \tilde{W}_i at a random point through an oracle. \mathcal{V} performs the \widetilde{add}_i and \widetilde{mult}_i calculations by itself, however \mathcal{V} receives claims from \mathcal{P} about $\tilde{W}_{i+1}(u^{(i+1)})$ and $\tilde{W}_{i+1}(v^{(i+1)})$ for the final check on that layer. Therefore, \mathcal{V} needs to be convinced that those values are correctly provided with sumchecks on layer $i+1$. Observe that a claim is reduced to two more claims on the subsequent layer. Therefore, if this continues, the number of claims would increase exponentially in terms of the circuit depth d . In order to prevent an exponential blowup in the number of sumchecks, we combine the two claims $\tilde{W}_{i+1}(u^{(i+1)})$ and $\tilde{W}_{i+1}(v^{(i+1)})$ at each layer into one claim as follows.

- \mathcal{V} defines a canonical line z such that $z_i(0) = u^{(i+1)}$ and $z_i(1) = v^{(i+1)}$ and sends $z_i(x)$ to \mathcal{P} .
 - \mathcal{P} sends a degree s_{i+1} univariate polynomial $h_i(x) = \tilde{W}_{i+1}(z_i(x))$.
 - \mathcal{V} checks if $h_i(0) = \tilde{W}_{i+1}(u^{(i+1)})$ and $h_i(1) = \tilde{W}_{i+1}(v^{(i+1)})$. \mathcal{V} then samples $r \in \mathbb{F}$ and computes $h_i(r) = \tilde{W}_{i+1}(z_i(r))$. In this sense, \mathcal{P} and \mathcal{V} can now go to the next level on $x^{(i+1)} = z_i(r)$.
- (3) At the input layer d , \mathcal{V} will receive two claims $\tilde{W}_d(u^{(d)})$ and $\tilde{W}_d(v^{(d)})$. As \mathcal{V} has complete access to the input layer gates, it can simply calculate the polynomials at those two random locations and see if they are equal to the claimed values. If they are, \mathcal{V} is sufficiently convinced and accepts. Otherwise, if any check has failed, \mathcal{V} rejects.

The GKR protocol is complete and has soundness $O(\frac{d \cdot \log|C|}{|\mathbb{F}|})$. For bounded-depth circuits, it is composed of $O(d \cdot \log|C|)$ interactive rounds. The verifier time is $O(\log|C|)$, and with the advancements for computation in [31], the prover time is $O(|C|)$.

4 NAIVE PARALLELIZATION INSECURITIES

Our main area of study in this work is the concept of GKR-based protocols that remove the requirement for a sequential layer-by-layer approach to proving the circuit evaluation correct. In this section, we first present the case why a simple natural parallelized-round protocol approach is not secure. Next, we show an attempted

fix of such a naive parallelized-round protocol, and show why it is again insecure.

4.1 Natural Method

At a first glance, the GKR protocol seems trivially parallelizable. A naive approach to achieve layer-wise parallelism would be simply to have \mathcal{V} initially choose $r^{(i)} \in \mathbb{F}^{s_i}$ for all i in the number of layers d . Then, d parallel instances of sumcheck would be executed over

$$\begin{aligned} \widetilde{W}_i(x^{(i)} = r^{(i)}) = & \\ \sum_{(a,b) \in \{0,1\}^{2s_{i+1}}} & \left[\widetilde{add}_i(r^{(i)}, a, b)(\widetilde{W}_{i+1}(a) + \widetilde{W}_{i+1}(b)) \right. \\ & \left. + \widetilde{mult}_i(r^{(i)}, a, b)(\widetilde{W}_{i+1}(a) \cdot \widetilde{W}_{i+1}(b)) \right] \end{aligned} \quad (4)$$

However, we establish that this is not secure.

The problem with this approach is that the random points are not necessarily correlated such that \mathcal{V} can conclude anything about the two claims $\widetilde{W}_{i+1}(u^{(i+1)})$ and $\widetilde{W}_{i+1}(v^{(i+1)})$ given at the last step of the sumcheck protocol for each layer i . Recall that in the original protocol $u^{(i+1)}$ and $v^{(i+1)}$ were correlated with the help of a line z , and another degree s_i polynomial h to determine the expected value of the evaluation at a random point on that line. Therefore, the next sumcheck for layer $i+1$ took place at $W_{i+1}(z(r))$, which was expected to be equal to $h(r)$. However, in this case the soundness of the protocol overall is compromised. Intuitively, \mathcal{P} already knows the randomness that \mathcal{V} provided about the other layers before going further through each sumcheck.

4.2 Correlation Fix

In this protocol, we make a slight change to the naive parallelization in order to establish a correlation of points. It is important to note that this "fix" is in fact still insecure, and an attack will be present in the later section.

Recall that in the traditional protocol, two claims $\widetilde{W}_{i+1}(u^{(i+1)})$ and $\widetilde{W}_{i+1}(v^{(i+1)})$ are reduced to a single claim with the help of a line $z(x)$ such that $z(0) = u^{(i+1)}$ and $z(1) = v^{(i+1)}$. \mathcal{P} sends a degree s_{i+1} univariate polynomial $h(x) = \widetilde{W}_{i+1}(z(x))$ which \mathcal{V} checks if $h(0) = \widetilde{W}_{i+1}(u^{(i+1)})$ and $h(1) = \widetilde{W}_{i+1}(v^{(i+1)})$. \mathcal{V} then samples $r \in \mathbb{F}$ and computes $h(r) = \widetilde{W}_{i+1}(z(r))$. \mathcal{P} and \mathcal{V} finally go to the next level on the point $x^{(i+1)} = z(r)$.

To maintain correlation of points, we propose a protocol in which we adopt a similar policy with concurrent layer proving. In this case, $u^{(i+1)}$ and $v^{(i+1)}$ are sampled by \mathcal{V} before the online phase, along with the line $z(x)$. During the online phase of the protocol, \mathcal{V} can designate the random point to evaluate on by sampling $r \in \mathbb{F}$ and requesting the evaluation point at $z(r)$ such that \mathcal{P} gives a claimed value for $\widetilde{W}_i(z(r))$ at each layer i , which is reduced to claims for $\widetilde{W}_i(u^{(i+1)})$ and $\widetilde{W}_i(v^{(i+1)})$ via sumcheck. Hence, when the end of the sumcheck is reached, \mathcal{V} can request $h(x)$ (the prover's claimed polynomial) and evaluate it at r to check for consistency in the $\widetilde{W}_{i+1}(z(r))$ claim for the subsequent layer.

4.3 Technical Description

Let \mathbb{F} be a prime field and C be a layered arithmetic circuit with depth d , with circuit layers labeled from the output layer 1 to the

input layer d . S_i denotes the number of gates in the i -th layer, and $s_i := \log(S_i)$. The protocol proceeds as follows.

Protocol: Naive Parallelized GKR (Correlation Fix)

- (1) \mathcal{V} randomly samples points $u^{(i+1)}, v^{(i+1)} \in \mathbb{F}^{s_{i+1}}$ and line $z_i(x)$ such that $z_i(0) = u^{(i+1)}$ and $z_i(1) = v^{(i+1)}$ for all $1 \leq i < d$.
- (2) For all $1 \leq i < d$, \mathcal{V} randomly samples point $r^{(i)} \in \mathbb{F}$. \mathcal{P} and \mathcal{V} then execute sumcheck over $\widetilde{W}_i(x^{(i)} = z_i(r^{(i)})) =$

$$\sum_{(a,b) \in \{0,1\}^{2s_{i+1}}} \left[\widetilde{add}_i(z_i(r^{(i)}), a, b)(\widetilde{W}_{i+1}(a) + \widetilde{W}_{i+1}(b)) \right. \\ \left. + \widetilde{mult}_i(z_i(r^{(i)}), a, b)(\widetilde{W}_{i+1}(a) \cdot \widetilde{W}_{i+1}(b)) \right] \quad (5)$$

where $x^{(i)}$ is the input to the sumcheck. The first half of the sumcheck's coin tosses will be set to $u^{(i+1)}$, and the second half of the sumcheck's coin tosses will be set to $v^{(i+1)}$.

- (3) For all $1 \leq i < d$, \mathcal{P} sends \mathcal{V} the s_{i+1} degree univariate polynomial $h_i(x) = \widetilde{W}_{i+1}(z_i(x))$. \mathcal{V} checks that $h(0) = \widetilde{W}_{i+1}(u^{(i+1)})$ and $h(1) = \widetilde{W}_{i+1}(v^{(i+1)})$.
- (4) \mathcal{V} checks if the claim from the sumcheck in the subsequent layer $\widetilde{W}_{i+1}(z_i(r^{(i+1)}))$ is equal to $h_i(z_i(r^{(i+1)}))$. If it is equivalent across all $1 \leq i < d$ and the claims for the input layer d are independently verified to be correct, \mathcal{V} accepts. Otherwise, if any check has failed, \mathcal{V} rejects.

4.4 Attack

We demonstrate a simple attack method that shows the insecurity of a traditional layer-wise independent GKR protocol, even when the points are correlated. Our attack stems from the fact that the unlike the traditional protocol, in the current form the evaluation point r can be deduced by \mathcal{P} before sending h to \mathcal{V} . Notice that after $u^{(i+1)}$ and $v^{(i+1)}$ are revealed to \mathcal{P} , it can deduce the line $z(x)$, and subsequently determine at which r point \mathcal{V} requested the evaluation of $\widetilde{W}_{i+1}(z(r))$ because \mathcal{P} knows $z(r)$. Once a malicious \mathcal{P}^* knows r , it can craft the degree s_{i+1} polynomial $h(x)$ such that it agrees with \mathcal{V} 's checks for $h(0) = \widetilde{W}_{i+1}(u^{(i+1)})$ and $h(1) = \widetilde{W}_{i+1}(v^{(i+1)})$, but pegging $h(r)$ to equal an incorrect $\widetilde{W}_{i+1}^*(z(r)) \neq \widetilde{W}_{i+1}(z(r))$. Thus, when \mathcal{V} checks for consistency, $h(r)$ is already compromised.

5 SUBCIRCUIT PROTOCOL

In this section, we propose a method to introduce some degree of depth-wise parallelism while circumventing the security issues previously shown with a naive parallelized GKR protocol.

Using pipelining, we propose splitting up a circuit C depth-wise by treating C as k subcircuits (denoted as c_i for all $1 \leq i \leq k$), where the input of subcircuit c_i directly corresponds with the output of subcircuit c_{i-1} . In other words, for all $i > 1$, the input of a subcircuit c_i is precisely the output of the subcircuit c_{i-1} above it. Then, the original GKR protocol can be executed on each subcircuit concurrently, maintaining the security guarantee of GKR with the

scope of each subcircuit. The only remaining need to guarantee full security of the protocol in regards to the entire circuit C would be proving the relationship between the output and input of adjacent subcircuits.

Recall in the GKR protocol the prover generates claims about the multilinear polynomial $\tilde{W}_i(x)$. For any two adjacent subcircuits, Let $\tilde{W}_{out}(r)$ be the claim regarding the gate values of the output level of the first circuit, and $\tilde{W}_{in}(r)$ be the claim regarding the gate values of the input level of the second circuit. As the claims should effectively be the same, \mathcal{P} wants to convince \mathcal{V} that $\tilde{W}_{out}(r) = \tilde{W}_{in}(r)$.

5.1 Naive Approach

Naively, \mathcal{P} and \mathcal{V} can simply iterate over the gate values at each input and output level, explicitly providing the guarantee that they are in fact equal. In this case, all of the gate values that comprise those specific levels would be revealed, and \mathcal{V} would perform a check for the $\tilde{W}_i(r)$ values claimed by \mathcal{P} for all random points sampled $r \in \mathbb{F}^{s_i}$.

However, by revealing the plain points of those layers, \mathcal{V} incurs a huge overhead in terms of computation and communication costs. Now, instead of receiving and sending only s_i points and evaluations, \mathcal{P} has to give \mathcal{V} all 2^{s_i} gate values, which is an exponential increase that we want to avoid.

5.2 Polynomial Commitment Scheme

A better approach would be to utilize cryptography, specifically *polynomial commitments*, which are crucial in succinct arguments in that they force \mathcal{P} to answer the queries by \mathcal{V} such that they must be in accordance with a specific bounded-degree polynomial. In essence, these polynomial commitments start with \mathcal{P} sending a value $s \in \mathbb{F}$ which is the claimed value of an $f(z)$ polynomial evaluation where \mathcal{V} knows z . Then, \mathcal{P} sends a corresponding opening proof π that the evaluation is indeed correct.

More specifically, we use the efficient constant-size polynomial commitment scheme described in [32] and let \mathcal{P} commit to the polynomials $\tilde{W}_{in}(x)$ and $\tilde{W}_{out}(x)$ (which should be the exact same) on every subcircuit. This way, the gap in security between the subcircuits is effectively bridged, and the verification process can be efficiently pipelined across the subcircuits.

5.3 Technical Description

Let \mathbb{F} be a prime field and C be a layered arithmetic circuit with depth d , with circuit layers labeled from the output layer 1 to the input layer d . C is split into k equal-depth subcircuits denoted as c_i for all $1 \leq i \leq k$. The input of subcircuit c_i directly corresponds with the output of subcircuit c_{i-1} . For each subcircuit c_i , let \tilde{W}_{in_i} denote the multilinear extension of the input layer, and \tilde{W}_{out_i} denote the multilinear extension of the output layer. Figure 1 depicts the protocol, which proceeds as follows.

Protocol: Subcircuit (Figure 1)

- (1) For all $1 \leq i \leq k$, \mathcal{P} commits (polynomial commitment) to \tilde{W}_{in_i} and \tilde{W}_{out_i} in subcircuit c_i .
- (2) \mathcal{P} and \mathcal{V} engage in batched traditional GKR protocol for all k subcircuits.

- (3) For all $1 \leq i \leq k$, \mathcal{P} opens the proofs at all randomly sampled locations for the evaluations of \tilde{W}_{in_i} and \tilde{W}_{out_i} during the batched traditional GKR protocol and sends these proofs to \mathcal{V} .
- (4) \mathcal{V} checks the proofs to verify that $\tilde{W}_{in_i}(x) = \tilde{W}_{out_i}(x)$ at all randomly sampled points for all adjacent subcircuits.

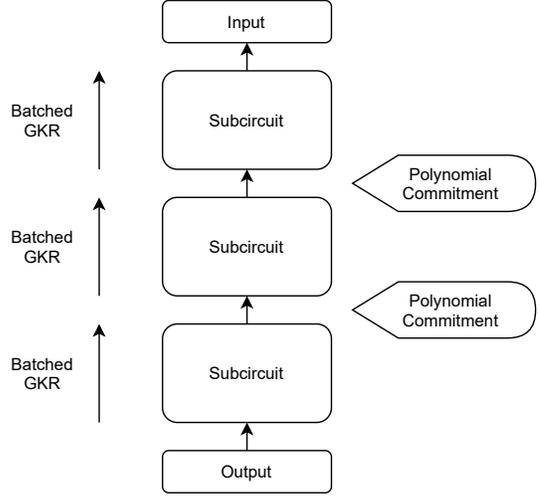


Figure 1: Subcircuit Protocol

5.4 Trade-off

We recognize that the total number of layers in the representation of C increases by an additive factor of k , as in each adjoining subcircuit the input and output are repeated. However, in the perspective of the proposed subcircuits, we essentially slash the depth and number of rounds by a *multiplicative* factor of k . This has great appeal for \mathcal{P} and \mathcal{V} assuming they have access to multithreaded hardware and can concurrently run the protocol on the separate subcircuits. In addition, assuming the subcircuits are all of the same depth and the layers are logspace uniform (s_i is equivalent across all i), the number of rounds can decrease by a factor of k as well. \mathcal{V} can use the same random coin tosses for sumchecks across all the subcircuits simultaneously and batch the proofs across the circuit.

6 HYDRA PROTOCOL

In this section, we present a GKR-based protocol with full layer-wise independence. Although the subcircuit protocol presented previously does allow for splitting a deep circuit up into multiple batched subcircuits, it does not allow for us to completely parallelize the proof across all of the layers in the circuit. Intuitively, this is because when we split the circuit up, the internal subcircuit itself still has to engage in a traditional sequential GKR protocol. Here, we propose an interactive protocol with a series of $O(\log n)$ rounds. Recall that n is the circuit width. We run $d \log n$ sumchecks concurrently. Despite the increase in number of sumchecks, the advantage of this protocol is the reduction in round complexity from the traditional $O(d \log n)$ rounds. Compared to the subcircuit

protocol, the Hydra protocol does not require the duplication of adjacent layers. More importantly, the round complexity is no longer dependent on the depth, so it does not degrade based on the number of layers in the circuit. Notably, Hydra does not depend on any non-standard assumptions (knowledge type, Fiat-Shamir/Random Oracle). Instead, Hydra only relies on the standard cryptographic assumptions presented in [32].

6.1 Context

For each layer of the traditional GKR protocol, the claim of $\widetilde{W}_i(x^{(i)})$ is reduced into claims of $\widetilde{W}_{i+1}(a^{(i)})$ and $\widetilde{W}_{i+1}(b^{(i)})$, where $x^{(i)} \in \mathbb{F}^{s_i}$ and $a^{(i)}, b^{(i)} \in \mathbb{F}^{s_{i+1}}$. More specifically, the polynomial evaluation

$$\begin{aligned} \widetilde{W}_i(x^{(i)}) = & \\ & \sum_{(a,b) \in \{0,1\}^{2s_{i+1}}} \left[\widetilde{add}_i(x^{(i)}, a, b)(\widetilde{W}_{i+1}(a) + \widetilde{W}_{i+1}(b)) \right. \\ & \left. + \widetilde{mult}_i(x^{(i)}, a, b)(\widetilde{W}_{i+1}(a) \cdot \widetilde{W}_{i+1}(b)) \right] \end{aligned} \quad (6)$$

over the sumcheck ultimately results in 2 claimed values for \widetilde{W}_{i+1} at the locations $a^{(i)}$ and $b^{(i)}$ determined by the random points sampled by \mathcal{V} during the protocol. During the sumcheck, each coin toss for a and b determines an additional dimension for $a^{(i)}$ and $b^{(i)}$, respectively.

At the beginning of each sumcheck, \mathcal{P} claims the value for $\widetilde{W}_i(x^{(i)})$. Let $x^{(i)}$ be known as a *query point*, and $\widetilde{W}_i(x^{(i)})$ be known as a *query point evaluation*. At the end of each sumcheck, \mathcal{P} claims the values for $\widetilde{W}_{i+1}(a^{(i)})$ and $\widetilde{W}_{i+1}(b^{(i)})$. Let $a^{(i)}, b^{(i)}$ be known as *challenge points*, and $\widetilde{W}_{i+1}(a^{(i)}), \widetilde{W}_{i+1}(b^{(i)})$ be known as *challenge points evaluations*.

The concept of layer-wise independence ultimately boils down to proving that the claimed challenge point evaluations ($\widetilde{W}_{i+1}(a^{(i)})$ and $\widetilde{W}_{i+1}(b^{(i)})$) given at the last step of the sumcheck at layer i is consistent with the claimed query point evaluation ($\widetilde{W}_{i+1}(x^{(i+1)})$) given at the beginning of the sumcheck run at layer $i+1$. Note that $a^{(i)}, b^{(i)}$, and $x^{(i+1)}$ should be uniformly random and uncorrelated in the perspective of \mathcal{P} . This is important because all of the query points are revealed at once to \mathcal{P} . If $a^{(i)}, b^{(i)}$ can be deduced given $x^{(i+1)}$, the soundness of sumcheck will be compromised.

Thus, the core of our protocol lies in the power of polynomial interpolation. If we let \mathcal{V} sample m query points ($x_1^{(i+1)}, x_2^{(i+1)}, \dots, x_m^{(i+1)}$) for layer $i+1$, it is simple to see how the evaluation claims for these values could be obtained through m instances of the sumcheck protocol at layer $i+1$. Then, \mathcal{V} can interpolate using points in the form of $(x_j^{(i+1)}, \widetilde{W}_{i+1}(x_j^{(i+1)}))$ for $1 \leq j \leq m$ to obtain the unique polynomial \widetilde{W}_{i+1} assuming m is large enough. Once \widetilde{W}_{i+1} is established, \mathcal{V} can use it to verify against the claimed challenge points evaluations $\widetilde{W}_{i+1}(a^{(i)})$ and $\widetilde{W}_{i+1}(b^{(i)})$ given at the last step of sumcheck at layer i . Given a set of $n+1$ points in the form of (x_i, y_i) where x_i is unique across all points, polynomial interpolation defines a linear bijection such that there exists a unique polynomial p of total degree at most n that agrees with all of the points.

If the query points are sampled from the full space of $\mathbb{F}^{s_{i+1}}$, \mathcal{V} would need $2^{s_{i+1}}+1$ claimed evaluation points in order to interpolate $\widetilde{W}_{i+1}(x)$. Note that \widetilde{W} is multilinear, therefore, the total degree is $2^{s_{i+1}}$. Unfortunately, similar to the plain gate reveal in the naive method for the subcircuit protocol, this is an exponential increase in claims that we want to avoid.

6.2 Subspace Reduction

Here, we investigate reducing this evaluation $\widetilde{W}_{i+1}(x^{(i+1)})$ by restricting the query and challenge points over a *subspace* in terms of a polynomial $f : \mathbb{F} \rightarrow \mathbb{F}^{s_{i+1}}$ (not to be confused with the f polynomial presented in the sumcheck background) sampled by \mathcal{V} , the coefficients of which are not revealed to \mathcal{P} . Evaluations and claims will all take place over $\widetilde{W}_{i+1}(f(x))$ at random points $x \in \mathbb{F}$. For example, given a multilinear function f and three randomly sampled field elements $\mu_j, \phi_j, \theta_j \in \mathbb{F}$ where $1 \leq j \leq m$, \mathcal{P} and \mathcal{V} will run sumchecks of the form

$$\begin{aligned} \widetilde{W}_i(x^{(i)} = f(\mu_j)) = & \\ & \sum_{(a,b) \in \{0,1\}^{2s_{i+1}}} \left[\widetilde{add}_i(f(\mu_j), a, b)(\widetilde{W}_{i+1}(a) + \widetilde{W}_{i+1}(b)) \right. \\ & \left. + \widetilde{mult}_i(f(\mu_j), a, b)(\widetilde{W}_{i+1}(a) \cdot \widetilde{W}_{i+1}(b)) \right]. \end{aligned} \quad (7)$$

When the sumchecks are run, the first half of \mathcal{V} 's s_{i+1} random coin tosses will be set equal to $f(\theta_j)$, and the second half of \mathcal{V} 's s_{i+1} random coin tosses will be set equal to $f(\phi_j)$ for all j . Thus, at the end of a single sumcheck, one claimed query point evaluation, namely $\widetilde{W}_i(f(\mu_j))$, is reduced to two claimed challenge points' evaluations on \widetilde{W}_{i+1} , namely $\widetilde{W}_{i+1}(f(\phi_j))$ and $\widetilde{W}_{i+1}(f(\theta_j))$. We propose $\deg(f) \cdot s_i + 1$ sumchecks to be executed per layer ($m = \deg(f) \cdot s_i + 1$), such that for each layer i , at the end of all sumchecks \mathcal{V} has $\deg(f) \cdot s_i + 1$ claimed query point evaluations and receives $2(\deg(f) \cdot s_i + 1)$ claimed challenge point evaluations. Then, \mathcal{V} can interpolate the query point evaluations to obtain the new polynomial $g_i(x) := \widetilde{W}_i(f(x))$. \mathcal{V} can check that $g_i(x)$ is in fact a degree $\deg(f) \cdot s_i$ polynomial, and subsequently verify it on the claimed challenge point evaluations. As no two distinct degree $\deg(h)$ polynomials can agree on more than $\deg(h)$ points, this setup protects from a dishonest \mathcal{P} by the Schwartz-Zippel lemma [1].

6.3 Maintaining Sumcheck Soundness

In addition, care must be taken to ensure that \mathcal{P} cannot predict any challenge points. If \mathcal{P} determines f or what value $f(x)$ will equal before the entire vector is revealed, it can bypass the soundness of sumcheck. Recall that for the sumcheck protocol, $f(\mu_j)$ is revealed to \mathcal{P} as it is the randomly sampled vector that \widetilde{W}_i is to be evaluated at (the input of the sumcheck). We observe that the two locations of claims ($f(\phi_j)$ and $f(\theta_j)$) on \widetilde{W}_{i+1} are the random coin tosses during the interactive sumcheck and are also revealed. Thus, if f is known to \mathcal{P} , after the first random coin toss (the first field element of an $f(x)$ evaluation) is sent, \mathcal{P} can determine the composition of the rest of the vector, which corresponds to the following $s_i - 1$ coin tosses. As the soundness of sumcheck relies on the fact that subsequent coin toss challenges are not known to \mathcal{P} , the security is compromised. We assert that for any polynomial $f : \mathbb{F} \rightarrow \mathbb{F}^{s_i}$, the

number of points for interpolation to determine $g_i(x) = \widetilde{W}_i(f(x))$ is $O(\deg(f) \cdot s_i + 1)$. However, f and the points f is evaluated at must not be revealed to preserve security. If f is revealed, the evaluation point x can be easily determined by \mathcal{P} . In addition, as \mathcal{V} reveals more than $\deg(f)$ evaluations of $f(x)$, x needs to be hidden as well because f can be interpolated by \mathcal{P} itself when given more than $\deg(f)$ points. The full proof for this concept is presented in the soundness proof later (see Section B.2).

The benefit of the subspace reduction method is twofold. First, the number of claims needed is no longer exponential. In this case, the number of rounds (claims needed per layer) is a proportional to s_i , which is ideal for even wide circuits. The degree of the polynomial f can subsequently be seen as a security parameter, in which a higher degree will require more points of evaluation in order for \mathcal{V} to interpolate the function and be fully convinced. Second, as the coefficients of f are not known to \mathcal{P} and f and x are hidden, from the perspective of \mathcal{P} the evaluation points are completely random and not correlated. Thus, \mathcal{P} cannot predict ahead of time the random coin tosses of any sumcheck, nor the challenge points for another layer before they are revealed by \mathcal{V} .

6.4 Malicious Interpolation Points

Finally, we note that the query point evaluations for interpolation that come from the $\widetilde{W}_{i+1}(f(x))$ polynomial need additional attention. As the initial claims are exactly what \mathcal{V} will be checking upon when \mathcal{V} interpolates, a malicious \mathcal{P} could be able to choose an incorrect \widetilde{W}_i^* such that it interpolates to the correct values *only in the given subspace of the protocol*. In other words, \mathcal{P} can proceed with an incorrect polynomial that in the scope of \mathcal{V} is completely consistent, and completely cooperate with \mathcal{V} throughout all sumchecks. After the sumchecks are completed, the values \mathcal{V} interpolates will be correct only in the scope of those claimed points. We present a formalized attack below.

- (1) Within any layer i in the circuit, \mathcal{P} chooses an incorrect polynomial \widetilde{W}_i^* constructed such that it agrees with the correct $\widetilde{W}_{i+1}(f(x))$ in the subspace provided. Namely, for each layer i the specific subspace the incorrect polynomial agrees upon is composed of the $f(x)$ evaluations given to \mathcal{P} on $\widetilde{W}_{i+1}(f(x))$.
- (2) For each sumcheck, \mathcal{P} engages in a completely honest interaction with \mathcal{V} , answering correctly according to the polynomial \widetilde{W}_i^* for all of \mathcal{V} 's challenges.
- (3) \mathcal{V} interpolates \widetilde{W}_i^* . The malicious polynomial bypasses this interpolation check. In the scope of the given claims, the interpolation shows complete consistency between adjacent layers.

Therefore, this attack will inductively lead to an incorrect claimed output that will be verified as correct by \mathcal{V} .

The reason why this is a significant vulnerability is because at the end of each sumcheck for a layer, \mathcal{P} cannot give \mathcal{V} oracle access to $\widetilde{W}_{i+1}(x)$ in order to verify the claims presented. In the original GKR protocol, the entire protocol on the subsequent layers $\geq i + 1$ essentially serve as the oracle for the current sumcheck on layer i .

In order to prevent this scenario, the protocol must guarantee that \widetilde{W}_i is not malicious by having \mathcal{P} engage in a polynomial commitment to the polynomial, similar to the subcircuit protocol. In such an environment, an adversarial \mathcal{P} will not be able to cheat with an incorrect \widetilde{W}_i because the polynomial commitment re-enables the oracle. \mathcal{P} is forced to bind to the polynomial before it knows the points that \widetilde{W}_{i+1} will be evaluated on.

6.5 Technical Description

Let \mathbb{F} be a prime field and C be a layered arithmetic circuit with depth d . Let $f : \mathbb{F} \rightarrow \mathbb{F}^{s_{i+1}}$ be a random polynomial of degree $\deg(f)$ sampled by \mathcal{V} for all $1 \leq i \leq d$. Recall that we use S_i to denote the number of gates in the i -th layer, and $s_i := \log(S_i)$. For the sake of simplicity, assume all of the layers obey the same logspace and let all of the individual functions in the dimension of f be linear. Figure 2 depicts the protocol, which proceeds as follows.

Protocol: Hydra (Figure 2)

- (1) For each $1 \leq j \leq s_i \cdot \deg(f) + 1$, \mathcal{V} samples random $\mu_j, \phi_j, \theta_j \in \mathbb{F}$. Let

$$\begin{cases} r_{i,j} = f(\mu_j) \\ a_{i,j} = f(\phi_j) \\ b_{i,j} = f(\theta_j) \end{cases}$$

for all $1 \leq i \leq d$.

- (2) For each $1 \leq i \leq d$, \mathcal{P} commits (polynomial commitment) to $\widetilde{W}_i(x)$.
- (3) For $1 \leq i \leq d$ and $1 \leq j \leq s_i \cdot \deg(f) + 1$, \mathcal{P} and \mathcal{V} concurrently execute sumchecks over

$$\begin{aligned} \widetilde{W}_i(x^{(i)} = r_{i,j}) = \\ \sum_{(a,b) \in \{0,1\}^{2s_{i+1}}} \left[\widetilde{add}_i(r_{i,j}, a, b) (\widetilde{W}_{i+1}(a) + \widetilde{W}_{i+1}(b)) \right. \\ \left. + \widetilde{mult}_i(r_{i,j}, a, b) (\widetilde{W}_{i+1}(a) \cdot \widetilde{W}_{i+1}(b)) \right] \end{aligned} \quad (8)$$

When the sumcheck is run, the first half of \mathcal{V} 's s_{i+1} random coin tosses will be set to $a_{i,j}$, and the second half of \mathcal{V} 's s_{i+1} random coin tosses will be set to $b_{i,j}$. At the beginning of sumcheck, \mathcal{P} claims the value for $\widetilde{W}_i(r_{i,j}) = \widetilde{W}_i(f(\mu_j))$. At the end of each sumcheck at layer i , \mathcal{P} provides claimed values for $\widetilde{W}_{i+1}(a_{i,j}) = \widetilde{W}_{i+1}(f(\phi_j))$ and $\widetilde{W}_{i+1}(b_{i,j}) = \widetilde{W}_{i+1}(f(\theta_j))$. Let the the beginning claims for $\widetilde{W}_i(r_{i,j})$ across $s_i \cdot \deg(f) + 1$ points be referred to as the *query point evaluations*, and the ending claims for $\widetilde{W}_{i+1}(b_{i,j})$ at a total of $2(s_{i+1} \cdot \deg(f) + 1)$ points be referred to as the *challenge point evaluations*.

- (4) For each $1 \leq i \leq d$ and $1 \leq j \leq s_i \cdot \deg(f) + 1$, \mathcal{P} opens and sends the polynomial commitment proofs of $\widetilde{W}_i(x)$ for all points $r_{i,j}$.
- (5) For all layers $1 < i < d$, \mathcal{V} uses the $s_i \cdot \deg(f) + 1$ query point evaluations from layer i to interpolate the unique polynomial $g_i(x) := \widetilde{W}_i(f(x))$. \mathcal{V} then checks to see if $g_i(x)$ is indeed a degree $s_i \cdot \deg(f)$ polynomial, and verifies consistency by evaluating $g_i(x)$ at the $2(s_i \cdot$

$deg(f) + 1$ challenge point evaluations reduced from layer $i - 1$, accepting only if $g_i(\phi_j) = \bar{W}_i(a_{i-1,j})$ and $g_i(\theta_j) = \bar{W}_i(b_{i-1,j})$ for all ϕ_j, θ_j points. As \mathcal{V} already has access to the input layer d and the output layer 1, it can check the \bar{W}_d and \bar{W}_1 claims on its own.

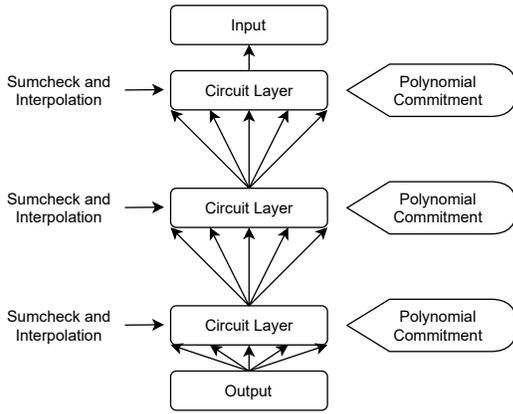


Figure 2: Hydra Protocol

7 EXPERIMENTAL EVALUATION

7.1 Environment

We used the Frigate [18] compiler to convert C-style code to logical circuits, and a parser to convert the circuits into provable representations for both the subcircuit protocol and the Hydra protocol. We implemented our proposals in around 1,500 lines total of C++ code with the polynomial and arithmetic logic based on Thaler’s implementation of [10]. Finally, we used the PolyCommit Rust library [39] from Marlin [32] for the multilinear extension composition-based polynomial commitments. We anticipate to release the source code for Hydra as fully open-source software. Our experiments were executed on 40 physical Intel Xeon CPU E7-4850 cores with hyperthreading (80 virtual cores) and 128 GB of RAM.

7.2 Results and Discussion

In this section, we present the results of our experiments, along with discussion and analysis. All experiments described in the following two subsections were run on randomly generated layered circuit structure and random input values which obeyed certain depth and width properties. The benchmark we compared our protocols against was a traditional implementation of the GKR protocol. In the case of the subcircuit protocol, we benchmark against a subcircuit composed of the entire circuit itself such that it transposes into a traditional GKR protocol. In the case of the Hydra protocol, we benchmark separately using the traditional GKR protocol under the same conditions.

Both the subcircuit protocol and the Hydra protocol were also tested with the practical concept of a *verifiable delay function* (VDF) [25]. In simple terms, a VDF is a sequential step-by-step evaluation that produces a specific output. In particular, we choose to iterate the SHA-256 cryptographic hash function using the efficient boolean SHA-256 logic circuit from [20].

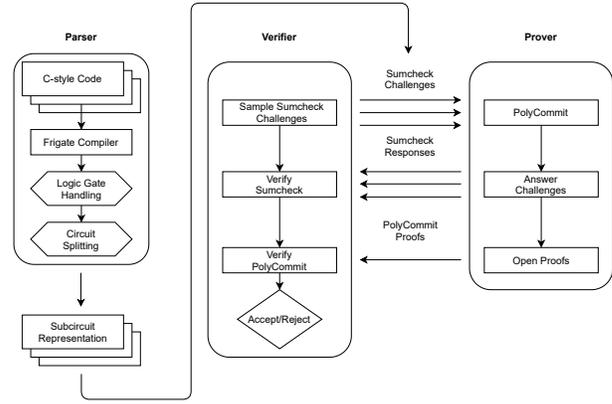


Figure 3: Subcircuit Environment

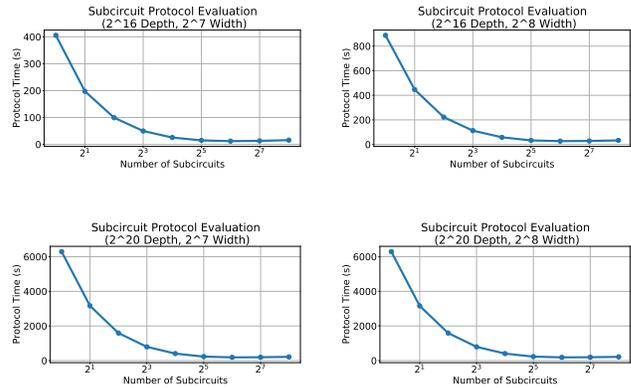


Figure 4: Subcircuit Evaluations

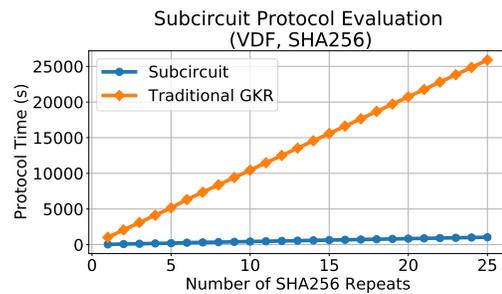


Figure 5: Subcircuit Protocol Evaluation (VDF, SHA256)

7.2.1 *Subcircuit Protocol*. For the subcircuit protocol (Figure 3), we evaluated multiple different circuit compositions, and tested various subcircuit depths and amounts. We mainly focused on testing long, skinny circuits that are known to be inefficient to prove with the traditional GKR approach. The tests were evaluated at two main circuit depths, $2^{16} = 65536$ and $2^{20} = 1048576$, as well as two main circuit widths, $2^7 = 128$ and $2^8 = 256$. We split the circuit

into evenly distributed subcircuits ranging from $2^0 = 1$ subcircuit (serving as the control, where the entire circuit is treated as one large subcircuit) up to $2^8 = 256$ subcircuits.

The results (Figure 4) of our tests were as expected, the protocol time decreased in proportion to the number of subcircuits the circuit was split up into. We note that the protocol time in fact started to increase ever so slightly after $2^6 = 64$ subcircuits. This is explained as the virtual core count of the machine we ran the tests was exceeded so more computational resources were wasted in scheduling/queue on the threads. Thus, we assert that as soon as the number of subcircuits exceed the number of virtual cores, increased parallelism is no longer profitable.

We see that the best increase in protocol time was when we tested with long and deep circuits that were split up into as subcircuits as the thread count would allow. In our case, the number of virtual cores was 80, and once the number of subcircuits increased from 2^6 to 2^7 , the protocol time began to increase. For depth 2^{16} and width 2^7 we see efficiency increase of 33.6 \times , and for depth 2^{16} and width 2^8 we see efficiency increase of 32.8 \times . Furthermore, for depth 2^{20} and width 2^7 we see efficiency increase of 33.8 \times , and for depth 2^{20} and width 2^8 we see efficiency increase of 34.8 \times .

For an ideal verifiable computation setup, the circuit would be split exactly in proportion with the number of virtual cores that are available. The case that subcircuits are dynamically added to be proven does not pose a problem because we take advantage of the streaming upload. Intuitively, the subcircuits earlier on are already proven, which frees computational resources for the subcircuits that are just being added.

With the SHA-256 VDF circuit evaluations (Figure 5), we also see favorable results with the subcircuit protocol. With around ~80 subcircuits generated per iteration, we see efficiency improvements of up to 26 \times , which is consistent with our self-constructed randomly generated circuits.

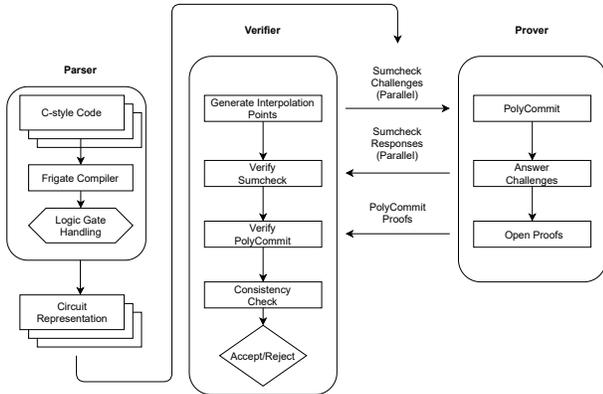


Figure 6: Hydra Environment

7.2.2 *Hydra Protocol.* For the Hydra protocol (Figure 6), we tested circuits composed of depth $2^{16} = 65536$ and widths of $2^7 = 128$ and $2^8 = 256$ (Table 3). Because Hydra requires more overall computational power than a traditional approach as it consists of $d(s_i \cdot deg(f) + 1)$ sumchecks instead of only d sumchecks, it

Protocol	n	Protocol Time (s)		
		0ms Latency	10ms Latency	20ms Latency
GKR	2^7	399.8s	9574.8s	18749.9s
Hydra	2^7	4384.4s	4384.5s	4384.7s
GKR	2^8	883.5s	11369.3s	21855.0s
Hydra	2^8	8459.9s	8460.1s	8460.2s

Table 2: GKR vs Hydra: 2^{16} Depth, n Width

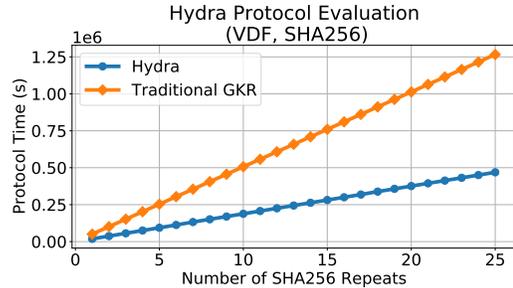


Figure 7: Hydra Protocol Evaluation (VDF, SHA-256)

is best utilized when the parallel core count of the server exceed $d(s_i \cdot deg(f) + 1)$. As the core count of our machine was restricted, we choose to demonstrate our results with an artificial latency that highlights the benefit of Hydra’s reduced round complexity. We argue that this latency is more representative of a real-life scenario. For example, when ensuring the validity of a cloud computation, there will clearly be a non-negligible amount of latency for conversation rounds between the client and the server. We evaluate Hydra with a 0ms latency as the ground truth, and from those results show the protocol times with 10ms and 20ms artificial latency period in between \mathcal{P} and \mathcal{V} interactions.

Due to significant limitations in the virtual core count of the machine we tested on, we see that with no latency, a traditional GKR protocol outperforms the Hydra protocol. However, once any amount of latency is added the true power of Hydra is revealed, even with virtual core counts that dwarf the circuit depth. With only a 10ms latency, Hydra comfortably approaches 2 \times protocol time efficiency for 2^{16} depth and 2^7 width, also improving by more than 1.3 \times for 2^{16} depth and 2^8 width. With 20ms of latency, these results are further compounded by efficiency improvements of 4.3 \times and 2.6 \times , respectively. We observe that these results play well with the advantageous long and skinny circuit structure, where the longer/skinnier the circuit is, the more the efficiency increases over the traditional GKR approach.

Even with core count limitations, with 2^9 ms latency per conversation round Hydra is able to improve over the traditional GKR protocol by a factor of 2.7 \times in the case circuits with rather long widths and short depths, such as with the SHA-256 circuit we tested with the VDF implementation (Figure 7). These results show that Hydra is superior in scenarios where high latency is a driving factor and bandwidth is available to spare.

8 APPLICATION TO NEURAL NETWORKS

Here, we explore applying Hydra's concurrent-layer proving technique to efficiently verify neural networks. We first provide necessary context behind applications of GKR-based protocols to verifiable neural network execution, and then introduce a novel interactive quantization algorithm that allows aspects of the training process to be verified as well. We then describe the execution verification process of our framework, and finally share some empirical results.

8.1 Context

A convolutional neural network (CNN) is composed of a series of layers. More specifically, most neural networks include a combination of convolutional layers, activation layers, and pooling layers followed by several fully connected layers and an output layer. Starting from the input layer, at every layer, input is received and process into an output for the next layer until the output layer is reached.

We use the SafetyNets [27] method as the basis to represent a neural network as an arithmetic circuit, which is summarized below.

8.1.1 Quadratic Activation. Here, we use a quadratic activation in all layers except the output layer, where the output is simply the square of the input. We choose this because the functions must have coefficients in the field \mathbb{F} when represented as an arithmetic circuit, and neural networks with quadratic activations have been shown to have the same power as networks with more commonly used activation functions (softmax, ReLU) [17].

8.1.2 Sum Pooling. CNNs use pooling layers to reduce the network size, effectively summarizing the features in a region. We use a sum pooling technique, where the pooling layer sums the activations in each region. Other pooling techniques [12] [14] are restricted due to highly inefficient max and division operations in arithmetic circuits.

8.1.3 Neural Network Model. Let the input to the network be $x \in \mathbb{F}^{n_0}$ where n_0 is the dimension of each input. The network is composed of d layers. Each layer i for $1 \leq i \leq d$ has n_i output neurons and is composed using a weight matrix $w_{i-1} \in \mathbb{F}^{n_{i-1} \times n_i}$ and biases $b_{i-1} \in \mathbb{F}^{n_i}$. We represent the neural network y_i for $1 \leq i \leq d$ as follows.

$$y_i = \sigma(w_{i-1} \cdot y_{i-1} + b_{i-1})$$

Where σ is the quadratic activation function for $1 \leq i < d$, and is the activation function for the output when $i = d$.

8.1.4 Finite Field Conversion. As for all arithmetic circuits, elements (in this case, weights and biases of the model) must lie in the finite field \mathbb{F} . Because the training process is done with floating point values, a conversion process is used to convert floating point weights and biases into field elements \mathbb{F} , while still preserving inference accuracy. SafetyNets uses a simple technique to scale this. Namely,

Let $w'_i \in \mathbb{R}^{n_{i-1} \times n_i}$ and $b'_i \in \mathbb{R}^{n_i}$ be the weights and biases after training the network for each layer $1 \leq i \leq d$ where d is the depth of the network.

- (1) Choose a large constant $\beta > 1$ and round w_i such that $w_i = \lfloor \beta w'_i \rfloor$.
- (2) Similarly, choose a scaling factor α and round the input such that $x = \lfloor \alpha x' \rfloor$.
- (3) Finally, to ensure the values in the network scale as expected, we set $b_i = \lfloor \alpha^{2^{i-1}} \beta^{2^{i-1}+1} b'_i \rfloor$.

8.2 Interactive Quantization Algorithm

Although the finite field conversion technique presented in SafetyNets is simple and easy to implement, as with many techniques for conversion in this area of GKR-based verifiable neural networks, this technique takes places completely after training, and done outside of the verification process. In addition, it is commonplace that many values of α and β need to be tested in order to achieve the best accuracy of the model possible.

In this section, we aim to address these issues with a novel interactive quantization algorithm, that combined with the subcircuit protocol introduced earlier, allows for a verifiable training process and reduces the guesswork that comes along with choosing constants to quantize with.

Our main observation here is that as we move towards the layers closer to the output layer, rounding the weights incurs less of an impact on final accuracy. As such, a simple method to quantizing the neural network would be to round the layers at the bottom layers first, retraining the network with said layers frozen, and repeating this process until the upper layers have been reached and all of the layers have been quantized.

This method is simple and intuitive, and allows for a network to be converted to an arithmetic circuit intra-training. In addition, recall that a key component of the SafetyNets quantization method is a trial and error approach for choosing appropriate α and β values. With our approach, the importance of choosing the best α and β constants is significantly diminished because at every step the model has the ability to reconverge and account for any poorly chosen values.

To add verification to this process, we can use the subcircuit protocol and break apart the neural network layers into different sections. With separate sections of the neural network, we can engage in a similar sequential bottom-up quantization technique where sections of the network are quantized and the model is retrained with quantized layers frozen. Then, we can also engage in the interactive subcircuit protocol to verify sections of the neural network that have already been quantized into an arithmetic circuit. Here, we present full description of such an approach.

Interactive Quantization Algorithm (Figure 8) Let $y_i = \sigma(w_{i-1} \cdot y_{i-1} + b_{i-1})$ represent a CNN with d layers, $1 \leq i \leq d$. Denote t as an accuracy threshold for the model, and let ℓ be number of layers per section that the neural network will be split into depth-wise.

- (1) Train the neural network using standard floating point weights and biases.
- (2) Starting from $i = d$ and moving upwards, group the last ℓ layers, namely $y_{d-\ell}$ to y_d . Choose large $\alpha, \beta \in \mathbb{F}$ and quantize the neural layers in the section in accordance

- with the SafetyNets technique such that $w_i = \lfloor \beta w'_i \rfloor$.
- (3) Freeze the weights/biases of the aforementioned section, and continue training the model until convergence. Stop training when the accuracy meets the threshold t .
 - (4) Concurrently, engage in an interactive subcircuit protocol that proves the validity of training data passing through pre-quantized sections of the network, which are treated as subcircuits.
 - (5) Proceed upwards to the next section (upper ℓ layers) of the network and repeat this process until all sections have been quantized and weights have been rounded.

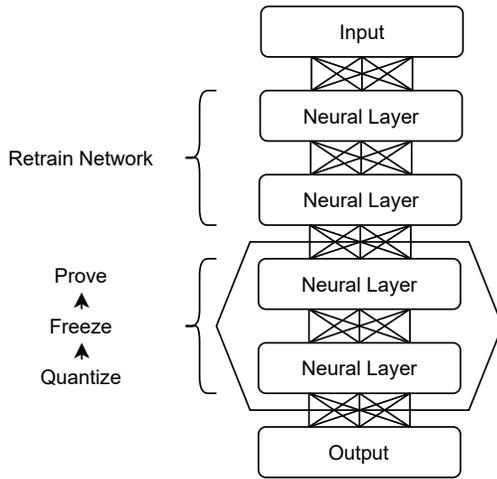


Figure 8: Interactive Quantization Algorithm

8.3 Execution Verification

After the model is trained and converted to a finite field arithmetic circuit, we can directly apply our concurrent layer proving technique (either the subcircuit protocol or the Hydra protocol) to said arithmetic circuit. Unlike in previous works such as SafetyNets where the depth of the network was limited to only two or three layers, with our approach we leverage the improvements to deep computation presented earlier and can efficiently verify “ultra-deep” CNNs composed of hundreds or even thousands of layers.

8.4 Empirical Results

We implemented both the neural network inference and execution process with different neural network architecture compositions. Our experiments were executed on Intel Core i7-11700F core with 32 GB of RAM. With the aforementioned methodology based off the original implementation from [15] and [27], we achieve 99.5% accuracy. As we obtain similar results to [27] in terms of accuracy, we defer to [27] for comparison with other related works. Our results were consistent with the observation from [15] that adding additional layers for this baseline classification task had insignificant impact on model accuracy.

For neural network verification, we tested our “HydraNet” protocol (same general computation subcircuit protocol, applied to a neural network arithmetic circuit) on two different architectures. Namely, we tested a model with an input batch size of 512, neural network input size of 256, and 100 hidden layers with 512 neurons each, and an output layer size of 128. In addition, we tested another model with the same parameters as aforementioned but consisting of 1000 hidden layers instead. We share our superior results (4.1× and 3.9× respectively benchmarked against SafetyNets [27]) below.

Protocol	n	Protocol Time (s)
Hydra	100	40.2s
SafetyNets [27]	100	164.7s
Hydra	1000	420.2s
SafetyNets [27]	1000	1654.8s

Table 3: Hydra vs SafetyNets: n hidden layers

9 FUTURE WORK

Our protocols rely on the cryptographic primitive of polynomial commitments to guarantee security. Therefore, they are formally categorized as arguments of knowledge. We are very interested to see if this dependence is truly necessary for our protocols. If it is not, this would result in near constant-round information theoretically secure interactive proofs, which would be a truly significant advancement for the theory community.

However, more research is needed to see if this approach is viable. Without the polynomial commitments, \mathcal{P} can cheat in the subcircuit protocol by executing GKR on malicious circuit compositions where the input and output layers of adjacent subcircuits do not match. Similarly, without polynomial commitments in the Hydra protocol, \mathcal{P} can cheat by providing incorrect values that only agree in the certain subspace in which \mathcal{V} interpolates.

In regards to experimental evaluations, we hope to test our protocols on higher core count compute to take full advantage of parallelization. On the verifiable neural network side, we are interested in testing our approaches with more powerful compute and a wider variety of datasets, such as CIFAR-10, to better demonstrate the superiority in both inference and training of “ultra-deep” CNNs.

10 CONCLUSION

In this paper, we have described two new verifiable computation protocols: the subcircuit protocol for breaking a large circuit up into smaller circuits that can be proven in batches/streaming upload, and the Hydra protocol for the generalized parallel proving of all layers in the circuit. Experimental results for general computation show that compared to traditional GKR implementations we improve by up to 34.8×. In addition, we apply our protocols in the context of neural networks, where we introduce a novel interactive quantization algorithm that converts a neural network into an arithmetic circuit, while also allowing aspects of the training process to be verified. We obtain protocol efficiency improvement of 4.1× and reach state of the art performance on MNIST.

REFERENCES

- [1] J. T. Schwartz. "Fast Probabilistic Algorithms for Verification of Polynomial Identities". In: *J. ACM* (1980).
- [2] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems". In: *ACM Symposium on Theory of Computing*. 1985.
- [3] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Proceedings on Advances in Cryptography*. 1987.
- [4] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. "Algebraic Methods for Interactive Proof Systems". In: *J. ACM* (1992).
- [5] Adi Shamir. "IP = PSPACE". In: *J. ACM* (1992).
- [6] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *ACM CCS*. 1993.
- [7] Shafi Goldwasser and Yael Tauman Kalai. "On the (In)security of the Fiat-Shamir Paradigm". In: *IEEE Symposium on Foundations of Computer Science*. 2003.
- [8] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating Computation: Interactive Proofs for Muggles". In: *ACM Symposium on Theory of Computing*. 2008.
- [9] Craig Gentry and Daniel Wichs. "Separating Succinct Non-Interactive Arguments From All Falsifiable Assumptions". In: *ACM Symposium on Theory of Computing*. 2011.
- [10] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. "Practical Verified Computation with Streaming Interactive Proofs". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. 2012.
- [11] Justin Thaler, Mike Roberts, Michael Mitzenmacher, and Hanspeter Pfister. "Verifiable Computation with Massively Parallel Interactive Proofs". In: *USENIX HotCloud*. 2012.
- [12] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. "Maxout Networks". In: *ICML*. 2013.
- [13] Justin Thaler. "Time-Optimal Interactive Proofs for Circuit Evaluation". In: *IACR CRYPTO*. 2013.
- [14] Matthew D. Zeiler and Rob Fergus. *Stochastic Pooling for Regularization of Deep Convolutional Neural Networks*. 2013. arXiv: 1301.3557 [cs.LG].
- [15] Yuchen Zhang, Percy Liang, and Martin J. Wainwright. "Convexified Convolutional Neural Networks". In: *ICML*. 2013.
- [16] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. "Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture". In: *USENIX Security*. 2014.
- [17] A Gautier, Q Nguyen, and M Hein. "Globally Optimal Training of Generalized Polynomial Neural Networks with Nonlinear Spectral Methods". In: *Conference on Neural Information Processing Systems*. 2016.
- [18] Benjamin Mood, Debayan Gupta, Henry Carter, Kevin R.B. Butler, and Patrick Traynor. "Frigate: A Validated, Extensible, and Efficient Compiler and Interpreter for Secure Computation". In: *IEEE European Symposium on Security and Privacy*. 2016.
- [19] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkatasubramanian. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: *ACM CCS*. 2017.
- [20] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. "Zero-Knowledge Contingent Payments Revisited: Attacks and Payments for Services". In: *ACM CCS*. 2017.
- [21] Riad S. Wahby, Ye Ji, Andrew J. Blumberg, abhi shelat abhi, Justin Thaler, Michael Walfish, and Thomas Wies. "Full Accounting for Verifiable Outsourcing". In: *ACM CCS*. 2017.
- [22] Yupeng Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. "vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases". In: *IEEE Symposium on Security and Privacy*. 2017.
- [23] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. *Scalable, Transparent, and Post-quantum Secure Computational Integrity*. Cryptology ePrint Archive, Report 2018/046. <https://eprint.iacr.org/>. 2018.
- [24] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. *Aurora: Transparent Succinct Arguments for R1CS*. Cryptology ePrint Archive, Report 2018/828. <https://eprint.iacr.org/>. 2018.
- [25] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. "Verifiable Delay Functions". In: *IACR CRYPTO*. 2018.
- [26] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *IEEE Symposium on Security and Privacy*. 2018.
- [27] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. "SafetyNets: Verifiable Execution of Deep Neural Networks on an Untrusted Cloud". In: *Conference on Neural Information Processing Systems*. 2018.
- [28] Yael Kalai, Omer Paneth, and Lisa Yang. *On Publicly Verifiable Delegation From Standard Assumptions*. Cryptology ePrint Archive, Report 2018/776. <https://eprint.iacr.org/>. 2018.
- [29] Riad S. Wahby, Ioana Tzialla, abhi shelat abhi, Justin Thaler, and Michael Walfish. "Doubly-Efficient zkSNARKs Without Trusted Setup". In: *IEEE Symposium on Security and Privacy*. 2018.
- [30] Yupeng Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. "vRAM: Faster Verifiable RAM with Program-Independent Preprocessing". In: *IEEE Symposium on Security and Privacy*. 2018.
- [31] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation". In: *IACR CRYPTO*. 2019.
- [32] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicolas Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *IACR EUROCRYPT*. 2020.
- [33] Seunghwa Lee, Hankyung Ko, Jiye Kim, and Hyunok Oh. *vCNN: Verifiable Convolutional Neural Network based on zk-SNARK*. Cryptology ePrint Archive, Report 2020/584. <https://eprint.iacr.org/>. 2020.
- [34] Srinath Setty. "Spartan: Efficient and general-purpose zkSNARKs without trusted setup". In: *Annual International Cryptology Conference*. 2020.
- [35] Jiaheng Zhang, Zhiyong Fang, Yupeng Zhang, and Dawn Song. "Zero Knowledge Proofs for Decision Tree Predictions and Accuracy". In: *ACM CCS*. 2020.
- [36] Boyuan Feng, Lianke Qin, Zhenfei Zhang, Yufei Ding, and Shumo Chu. *ZEN: Efficient Zero-Knowledge Proofs for Neural Networks*. Cryptology ePrint Archive, Report 2021/087. <https://eprint.iacr.org/>. 2021.
- [37] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. *A Survey of Quantization Methods for Efficient Neural Network Inference*. 2021. arXiv: 2103.13630 [cs.CV].
- [38] Tianyi Liu, Xiang Xie, and Yupeng Zhang. *zkCNN: Zero Knowledge Proofs for Convolutional Neural Network Predictions and Accuracy*. Cryptology ePrint Archive, Report 2021/673. <https://eprint.iacr.org/>. 2021.
- [39] Arkworks. *Poly-Commit Rust Library*. <https://github.com/arkworks-rs/poly-commit>.

A SUBCIRCUIT PROTOCOL GUARANTEES

Here, we provide the complete proof behind the validity of the subcircuit protocol as described.

A.1 Completeness

The completeness is straightforward from the completeness of sumcheck and polynomial commitments, as well as the protocol description.

A.2 Soundness

In the scope of the individual subcircuits, the soundness is guaranteed by the GKR protocol. With regards to connecting the input and output layers of adjacent subcircuits, the extractability of polynomial commitments in [32] for bounded polynomial-time \mathcal{P} and \mathcal{V} guarantees its soundness. A formal proof follows.

Let \mathbb{F} be a prime field. Let C be a layered arithmetic circuit, given an *input* and a claimed *output*. C has depth d , with circuit layers labeled from the output layer 1 to the input layer d . C is split into k equal-depth subcircuits denoted as c_i for all $1 \leq i \leq k$. The *subinput* $_i$ of c_i (input of subcircuit c_i) directly corresponds with the *suboutput* $_{i-1}$ of c_{i-1} (output of subcircuit c_{i-1}). For each subcircuit c_i , let \tilde{W}_{in_i} denote the multilinear extension of the input layer, and \tilde{W}_{out_i} denote the multilinear extension of the output layer.

Suppose that $C(\text{input}) \neq \text{output}$. Assume there exists an adversarial \mathcal{P}^* such that

$$\Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle = \text{accept}] = p.$$

Let A be the event in which $\langle \mathcal{P}^*, \mathcal{V} \rangle = \text{accept}$. For all $1 \leq i \leq k$, let G_i be the event where $c_i(\text{subinput}_i) \neq \text{suboutput}_i$ and let P_i be the event where $\tilde{W}_{in_i} \neq \tilde{W}_{out_i}$. We observe that

$$\begin{aligned}
 p &= \Pr[A] \\
 &\leq \Pr[\exists i \in [k] \text{ s.t. } A \wedge (G_i \vee P_i)] \\
 &\leq \sum_{i=1}^k \Pr[A \wedge (G_i \vee P_i)] \\
 &\leq \sum_{i=1}^k (\Pr[A \wedge G_i] + \Pr[A \wedge P_i]). \tag{9}
 \end{aligned}$$

The soundness of the GKR protocol implies that

$$\begin{aligned}
 &\sum_{i=1}^k (\Pr[A \wedge G_i]) \\
 &\leq \sum_{i=1}^k \frac{s_i \cdot \frac{d}{k}}{|\mathbb{F}|} \\
 &\leq O\left(\frac{s_i \cdot d}{|\mathbb{F}|}\right). \tag{10}
 \end{aligned}$$

From the the computational extractability of the polynomial commitment scheme, we see that

$$\begin{aligned}
 &\sum_{i=1}^k (\Pr[A \wedge P_i]) \\
 &\leq \sum_{i=1}^k \lambda \\
 &\leq O(k(\lambda)). \tag{11}
 \end{aligned}$$

where λ is the security parameter used in [32] of the commitment. Finally, by the union bound we obtain

$$\begin{aligned}
 p &= \Pr[A] \\
 &\leq \Pr[\exists i \in [k] \text{ s.t. } A \wedge (G_i \vee P_i)] \\
 &\leq \sum_{i=1}^k \Pr[A \wedge (G_i \vee P_i)] \\
 &\leq \sum_{i=1}^k (\Pr[A \wedge G_i] + \Pr[A \wedge P_i]) \\
 &\leq O\left(\frac{s_i \cdot d}{|\mathbb{F}|} + k(\lambda)\right). \tag{12}
 \end{aligned}$$

B HYDRA PROTOCOL GUARANTEES

Here, we provide the complete proof behind the validity of the Hydra protocol as described.

B.1 Completeness

The completeness is straightforward from the completeness of sum-check and polynomial commitments, and the protocol description.

B.2 Soundness

In order to guarantee the security of the Hydra protocol, we must first guarantee the security of the sumcheck's coin tosses. As stated earlier, if at any point f or its evaluation points are compromised, a

malicious prover can completely bypass the soundness of the sum-check protocol as it knows the coin tosses ahead of time. Thus, we first prove that in the perspective of \mathcal{P} , all coin tosses are completely random and probabilistically unpredictable.

Let $n = s_i$ and $m = s_i \cdot \text{deg}(f) + 1$. For the sake of simplicity, assume $\text{deg}(f) = 1$ (f is linear in each of its dimensions). At each layer, \mathcal{P} views a system of polynomial equations as follows:

$$\begin{aligned}
 x_1 y_1 + z_1 &= c_{1,1} & \dots & & x_1 y_m + z_1 &= c_{1,m} \\
 & \dots & & & & \dots \\
 x_n y_1 + z_n &= c_{n,1} & \dots & & x_n y_m + z_n &= c_{n,m}.
 \end{aligned}$$

where x_i, z_i are variables for $1 \leq i \leq n$, y_j is a variable for $1 \leq j \leq m$, and $c_{i,j}$ is a constant for $1 \leq i \leq n$ and $1 \leq j \leq m$. At first glance, this seems like an overdetermined system of polynomials, as the total number of variables is $2n+m$ and the total number of equations is $n \cdot m$. However, we establish that because of the structure of such equations, this is not the case.

We can form a reduction by linear combination of such a system by subtracting $x_i y_j + z_i = c_{i,j}$ from $x_i y_1 + z_i = c_{i,1}$ so that we obtain

$$\begin{aligned}
 x_1 (y_1 - y_2) &= c_{1,1} - c_{1,2} & \dots & & x_1 (y_1 - y_m) &= c_{1,1} - c_{1,m} \\
 & \dots & & & & \dots \\
 x_n (y_1 - y_2) &= c_{n,1} - c_{n,2} & \dots & & x_n (y_1 - y_m) &= c_{n,1} - c_{n,m}.
 \end{aligned}$$

Factoring out common x_i will reduce this system to dependent ratios in the form of

$$\begin{aligned}
 \frac{y_1 - y_2}{y_1 - y_3} &= \frac{c_{1,1} - c_{1,2}}{c_{1,1} - c_{1,3}} = \dots = \frac{c_{n,1} - c_{n,2}}{c_{n,1} - c_{n,3}} \\
 & \dots & & & \dots \\
 \frac{y_1 - y_2}{y_1 - y_m} &= \frac{c_{1,1} - c_{1,2}}{c_{1,1} - c_{1,m}} \dots = \frac{c_{n,1} - c_{n,2}}{c_{n,1} - c_{n,m}}
 \end{aligned}$$

Note that the chain of equalities must hold by construction: \mathcal{V} generated the system of polynomials such that it is valid. Therefore, in order for a correct solution to exist, the ratio of the constants shown must be equivalent and consistent.

We now demonstrate how a solution of the reduced system yields a valid solution to the original system. Given that the above reduction holds, we see that there must exist an $x_i \in \mathbb{F}$ for all $1 \leq i \leq n$ such that

$$\begin{aligned}
 \frac{x_1 (y_1 - y_2) = c_{1,1} - c_{1,2}}{x_1 (y_1 - y_3) = c_{1,1} - c_{1,3}} & \dots & \frac{x_n (y_1 - y_2) = c_{n,1} - c_{n,2}}{x_n (y_1 - y_3) = c_{n,1} - c_{n,3}} \\
 & \dots & \dots \\
 \frac{x_1 (y_1 - y_2) = c_{1,1} - c_{1,2}}{x_1 (y_1 - y_m) = c_{1,1} - c_{1,m}} & \dots & \frac{x_n (y_1 - y_2) = c_{n,1} - c_{n,2}}{x_n (y_1 - y_m) = c_{n,1} - c_{n,m}}.
 \end{aligned}$$

generalizing this for $1 \leq i \leq n$ and $1 \leq j \leq m$ we get

$$x_i (y_1 - y_j) = c_{i,1} - c_{i,j}$$

which expands to

$$x_i y_1 - x_i y_j = c_{i,1} - c_{i,j}$$

Corresponding to the original system of equations, we want to find z_i such that

$$\begin{aligned}
 x_i y_1 + z_i &= c_{i,1} \\
 x_i y_j + z_i &= c_{i,j}
 \end{aligned}$$

Notice that

$$x_i y_1 - x_i y_j = c_{i,1} - c_{i,j} \implies c_{i,1} - x_i y_1 = c_{i,j} - x_i y_j.$$

Thus, if we set z_i equal to $x_i y_1 - c_{i,1} = x_i y_j - c_{i,j}$, we obtain

$$\begin{aligned} x_i y_1 + (c_{i,1} - x_i y_1) &= c_{i,1} \\ x_i y_j + (c_{i,j} - x_i y_j) &= c_{i,j}. \end{aligned}$$

We can now see by the transitive property of equality that this is indeed a valid solution for $x_i y_j + z_i = c_{i,j}$ across all $1 \leq i \leq n$ and $1 \leq j \leq m$. It is clear how y_1 and y_2 can be the two degrees of freedom in this system. Any y_1 and y_2 fixed in the space of \mathbb{F} will result in a valid system. The remaining y_3, \dots, y_m are easily derivable, and once the y_i values are known for all $1 \leq i \leq m$, plugging them back into the original system will quickly yield x_i and z_i for all $1 \leq i \leq m$. Thus, the probability of \mathcal{P} bypassing soundness when f is linear in each of its dimensions is $\frac{1}{|\mathbb{F}|^2}$. With this in mind, we now proceed with the remainder of the proof.

Suppose that $C(\text{input}) \neq \text{output}$. Assume there exists an adversarial \mathcal{P}^* such that

$$\Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle = \text{accept}] = p.$$

Recall that the Hydra protocol takes $s_i \cdot \text{deg}(f) + 1$ claims on \widetilde{W}_i for every $1 \leq i \leq d$ layers, which reduces to $2(s_{i+1} \cdot \text{deg}(f) + 1)$ claims on \widetilde{W}_{i+1} . Namely, across all $i \in [d]$, $s_i \cdot \text{deg}(f) + 1$ claims that $\widetilde{W}_i(r_{i,j}) = R_{i,j}$ are reduced to $s_{i+1} \cdot \text{deg}(f) + 1$ verifications of $\widetilde{W}_{i+1}(a_{i,j}) = A_{i,j}$ and $s_{i+1} \cdot \text{deg}(f) + 1$ verifications of $\widetilde{W}_{i+1}(b_{i,j}) = B_{i,j}$.

As we assume the layers in C obey the same logspace, s_i is the equivalent across all $1 \leq i \leq d$. Note that $r_{i,j}$, $a_{i,j}$, and $b_{i,j}$ are all points on f , however, because neither f nor the locations where f is evaluated at are revealed, in the perspective of \mathcal{P} the random coin tosses of the sumchecks are in fact completely random and probabilistically unpredictable.

Let A be the event in which $\langle \mathcal{P}^*, \mathcal{V} \rangle = \text{accept}$. For all $1 \leq i \leq d$, let T_i be the event in which indeed all $\widetilde{W}_i(r_{i,j}) = R_{i,j}$ for layer i across all $1 \leq j \leq s_i \cdot \text{deg}(f) + 1$. We observe that

$$\begin{aligned} p &= \Pr[A] \\ &= \Pr[A \wedge \neg(T_1) \wedge T_d] \\ &\leq \Pr[\exists i \in [d] \text{ s.t. } A \wedge \neg(T_{i-1}) \wedge T_i] \\ &\leq \sum_{i=1}^d \Pr[A \wedge \neg(T_{i-1}) \wedge T_i]. \end{aligned} \quad (13)$$

Let E_i be the event in which indeed all $\widetilde{W}_{i+1}(a_{i,j}) = A_{i,j}$ and $\widetilde{W}_{i+1}(b_{i,j}) = B_{i,j}$ for $i \in [d]$. We can see that

$$\begin{aligned} &\sum_{i=1}^d \Pr[A \wedge \neg(T_{i-1}) \wedge T_i] \\ &= \sum_{i=1}^d (\Pr[A \wedge \neg(T_{i-1}) \wedge T_i \wedge E_i] \\ &\quad + \Pr[A \wedge \neg(T_{i-1}) \wedge T_i \wedge \neg(E_i)]). \end{aligned} \quad (14)$$

First, the soundness property of the sumcheck protocol across $s_i \cdot \text{deg}(f) + 1$ instances for d layers implies that

$$\begin{aligned} &\sum_{i=1}^d \Pr[A \wedge \neg(T_{i-1}) \wedge T_i \wedge E_i] \\ &\leq \sum_{i=1}^d \Pr[A \wedge \neg(T_{i-1}) \wedge E_i] \\ &\leq \sum_{i=1}^d \frac{s_i(s_i \cdot \text{deg}(f) + 1)}{|\mathbb{F}|} + \frac{1}{|\mathbb{F}|^2} \\ &\leq d \left(\frac{s_i(s_i \cdot \text{deg}(f) + 1)}{|\mathbb{F}|} + \frac{1}{|\mathbb{F}|^2} \right) \\ &\leq O\left(\frac{d \cdot s_i^2 \cdot \text{deg}(f)}{|\mathbb{F}|} \right). \end{aligned} \quad (15)$$

Second, note that a ground truth T_i will result in the correct claims of the query points where \mathcal{V} interpolates on to obtain the polynomial $h_i(x)$, which subsequently is indeed equal to $\widetilde{W}_i(f(x))$. If that is the case, there exists no situation where \mathcal{V} accepts a false E_i event where the claimed values for the challenge points are not consistent. In regards to the soundness of the polynomial commitment, [32] shows that the extractability of their construction guarantees that it is computationally sound. Thus, given a correct $h_i(x)$, \mathcal{V} will always reject $\neg(E_i)$ with probability negligibly close to 1 owing to the polynomial commitment. Therefore,

$$\begin{aligned} &\sum_{i=1}^d \Pr[A \wedge \neg(T_{i-1}) \wedge T_i \wedge \neg(E_i)] \\ &\leq \sum_{i=1}^d \Pr[A \wedge T_i \wedge \neg(E_i)] \\ &\leq O(d(\lambda)) \end{aligned} \quad (16)$$

where λ is the security parameter used in [32] of the commitment for polynomial-time \mathcal{P} . Finally, by the union bound we obtain

$$\begin{aligned} p &= \Pr[A] \\ &\leq \sum_{i=1}^d \Pr[A \wedge \neg(T_{i-1}) \wedge T_i] \\ &\leq \sum_{i=1}^d (\Pr[A \wedge \neg(T_{i-1}) \wedge T_i \wedge E_i] \\ &\quad + \Pr[A \wedge \neg(T_{i-1}) \wedge T_i \wedge \neg(E_i)]) \\ &\leq O\left(\frac{d \cdot s_i^2 \cdot \text{deg}(f)}{|\mathbb{F}|} \right) + O(d(\lambda)) \\ &\leq O\left(d \left(\frac{s_i^2 \cdot \text{deg}(f)}{|\mathbb{F}|} + \lambda \right) \right). \end{aligned} \quad (17)$$