Abstract

In the current blockchain world, a Turing-complete smart contract virtual machine has been widely used and attracted the attention and participation of application developers. As a result, many fields like decentralized finance, crypto art, and decentralized game have made great progress. However, the current blockchain infrastructure requires all nodes to verify and agree on the calculation results, which severely limits the ability of smart contracts. The existing smart contract languages and virtual machines are limited to writing short programs and accessing very few resources. Yet, machine learning models require a massive demand for computing and storage resources to be applied on the blockchain environment. With the emergence of artificial intelligence services, machine learning plays a huge role in image recognition, natural language processing, pattern recognition and many other fields.

The Cortex project adds AI algorithm support to smart contracts by expanding the underlying instruction set of smart contracts and enhancing the storage layer so that anyone can add AI capabilities to smart contracts. At the same time, the proposed incentive mechanism prompts model contributors to submit and optimize models on Cortex chain and receive rewards.

The ultimate goal of Cortex is to become a better token which supports all AI models. To achieve this goal, Cortex 2.0 builds a more comprehensive core technical architecture to enhance the on-chain AI inference in smart contracts and improve the security and performance of the Cortex chain. In addition, Cortex 2.0 also put a great emphasis on the accuracy and privacy of on-chain AI inference results by implementing formal verification, trusted execution environment, etc.

Cortex 2.0 - AI on Blockchain

The Cortex Team

Jan,30,2021

Contents

1 Introduction

In 2009, Bitcoin [1] proposed a blockchain model that can support a decentralized and credible digital currency exchange system. However, its built-in stacked language is not Turing-complete, such as limiting loop functions, this design restricts the real-world application scenarios of the Bitcoin system. In 2013, the Ethereum [2] project extended the programmable language on blockchain to the Turing-complete Solidity language. This expansion has brought a large number of applications to the blockchain system. Ethereum proposed the Gas mechanism to cope with the subsequent Turing shutdown issue and denial of service attack problems. Although it is a good solution from an economic point of view, but Ethereum can only perform small-scale calculations and storage, limiting the application of the Ethereum system in big data scenarios such as machine learning. There are projects trying to improve the performance of Ethereum to run more complex smart contracts, such as the PoS public chain Algorand [3], the two-layer expansion plan zk-Rollup [4], Arbitrum [5], the new consensus public chain Solana [6], Avalanche [7], decentralized storage IPFS [8] and other solutions. Yet, there is still a big gap between the computing and storage requirements for machine learning.

The Cortex project [9] based on Ethereum but expands the fundamentals and breaks the barrier between the current blockchain system and artificial intelligence. Cortex chain introduces unprecedented capabilities such as classification, prediction, and generation of AI models to the blockchain system. Massive breakthroughs come with more challenges. As such, to solve the computing, storage, and networking burdens of AI applications in the blockchain system, Cortex has proposed a series of solutions:

- Implement the Model Representation Tool (MRT) to convert traditional AI models into fixed-point models that can be executed on the blockchain;

- Propose the Cortex virtual machine, CVM, to realize on-chain AI inference calculations;
- Introduce TorrentFS P2P file storage system to solve models and data storage problem;

Moreover, AI services require large-scale data and massive computing power, which both are mainly in the hand of large companies, forming a unhealthy monopolistic market. To break the barrier, the Cortex ecosystem provides a decentralized AI model marketplace where users can share AI models and gain rewards, allowing more people to enjoy AI technology. We believe the Cortex project will bring together AI technology and the general public, just as Prometheus brings light to humanity.

With the completion of the Cortex 1.0 mainnet launch in June 2019, The 1.0 white paper [9] has been realized, and it is no longer a fantasy to deploy and run machine learning models on the blockchain. Large companies are no longer the only ones with big data and computing power. Everyone can enjoy the convenience brought by AI through the Cortex chain and fully understand how their data is used. Developers can now access AI models to build more complex applications and benefit from them.

With the introduction of Cortex 2.0, the Cortex Labs team aims to make further breakthroughs in system architecture, software implementation, hardware implementation, and app ecosystem. we hope to achieve a decentralized AI-on-blockchain ecosystem with enhanced features, more powerful performance, more stable protocols, and cross-chain capabilities to bring AI technology to other blockchain systems.

This paper introduces the vision of Cortex 2.0 design, its core architecture, and overall framework. The core architecture section describes how to further improve performance, security, and decentralization of Cortex chain. The overall framework section is a series of on-chain and off-chain work introductions, including facilitating the use of the Cortex chain, protecting the user's model and data privacy, and making it easier and better to use on-chain AI models, etc.

2 The Core Framework

To build a public chain that supports AI models, Cortex 2.0 needs to optimize the AI model inference and the underlying blockchain infrastructure. Not only does it needs to improve the on-chain model accuracy and consistency, but it also needs to optimize the existing Cortex chain in terms of consensus and performance. The Cortex 2.0 core architecture is shown in Figure 1 and contains the following technological breakthroughs:

1. Formal verification: The formalization and correctness verification of AI operators are completed through the Z3 prover [10] to ensure that all model inference results in the Cortex chain are deterministic and reproducible.
2. AI operator library: Further improve the underlying operator library of AI model supported by Cortex, so that Cortex can achieve more AI model inference work.
3. Consensus algorithm: The RandomAI workload proof algorithm to enhance the decentralization of Cortex furthermore.
4. Performance improvement: Gradually realize the packaging of transfer transactions, smart contracts, and AI inference, and improve the performance of the Cortex chain through zero-knowledge proof technology.

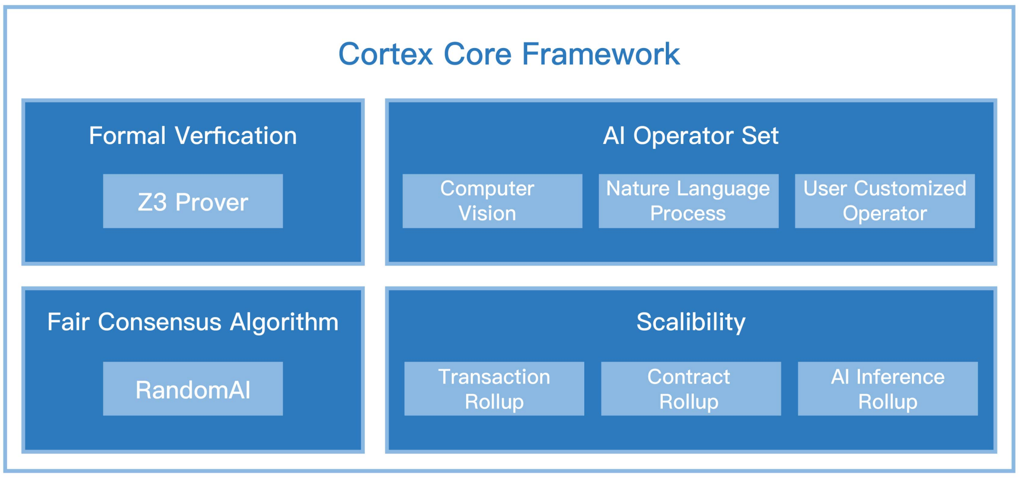

Figure 1: Cortex 2.0 Core Framework

2.1 Formal Verification: Z3-Prover

The instruction execution and calculation results in the smart contract virtual machine belong to the consensus mechanism of blockchain, which requires the instruction operation in the virtual machine to be deterministic and reproducible. Cortex 1.0 regards the AI model inference operation as a basic instruction (INFER | IFNERARRAY) integrated into the virtual machine (CVM), and this leads to two important characteristics that AI inference operation should have on blockchain: determinism and reproducibility. The Cortex Labs team pays sufficient attention to the above features, and proposes or plans to propose a series of interpretable models or methods to ensure the completeness of the inference operation within the deterministic AI framework (CVM Runtime):

- Write and publish an MRT quantitative paper to introduce the necessity, completeness of the deterministic AI framework and related model transformation method;
- Use mathematically strict descriptive symbols to define the input, output,

and process logic of the operator in the AI framework, to ensure that the calculation of the operator is theoretically verifiable;

- Use the third-party library Z3-Prover to verify the correctness of source code implementation in the CVM Runtime project library;

Cortex 2.0 plans to publish the MRT quantitative paper, which focus on the model transformation work made by the Cortex Labs team. This work is for the deterministic AI framework CVM Runtime, including but not limited to the necessity of model fixed-pointization, related research and key achievements. The calculation requirements on the blockchain includes certainty, while the parallel architecture of floating-point models in existing mainstream AI framework has uncertainty of calculation. The necessity of model fixed-pointization lies mainly in the contradiction between the certainty on the blockchain and the uncertainty in the AI framework. For this reason, relevant quantitative research papers have been investigated and related achievements have been referred to implement the MRT conversion tool, which is adapted to the deterministic AI framework. For more details please refer to the paper description and the release plan is described in the roadmap section 5.

In addition, Cortex 2.0 logically abstracts the operator codes already supported in the CVM Runtime to form a strict mathematical operation definition to ensure theoretical interpretability, consistency, and completeness. By standardizing and pre-stating the input, output and other configuration attributes of the operator, the processing logic implicit in the code is abstracted in the form of mathematical expressions and the equation symbol definition, aka formula, is given. The formal symbol of mathematical description can provide a theoretical complete reference and give a unique and definite calculation result when the calculation results are inconsistent between possible inference code versions. The operator mathematical notation description supported in the current CVM Runtime project library has been constructed and added to appendix A.

Finally, Cortex 2.0 intends to use the satisfiability modulo theories (SMT) [11] to complete the formal verification of the operator implementation in the codebase. SMT is a generalization of Boolean satisfiability (SAT), adding equation reasoning, arithmetic, fixed-size bit-vectors, arrays, quantifiers, and other useful first-order theories. The SMT solver is a tool to determine the satisfiability (or double validity) of the formulas in these theories. It can be used in applications such as extended static checking, predicate abstraction, test case generation, and bounded model checking on infinite domains.

Z3 formal validator (Z3-Prover) [10] is a new SMT solver developed by Microsoft in the United States. Its goal is to solve problems that arise in software verification and software analysis, and integrates a variety of the latest and comprehensive theoretical research work and implements it in the project library for code analysis, code audit, and so on. Cortex 2.0 will invoke the Z3-Prover library to perform formal verification for all operators' code in the CVM Runtime. By defining the data scale and range of the input and configuration attributes of each operator, it is verified that the output or intermediate calculation results are computationally correct and precision overflow free.

2.2 More AI models on Cortex: Extended Operator Set

A series of operator sets and their implementation are defined in the CVM Runtime project library. And a strict mathematical description definition is described above, stipulating the logic of the operator to calculate under given input and output a deterministic result. The supported operator set refers to the existing mainstream deep learning framework architecture and combines the network structure involved in AI models used most often, including necessary operators such as convolution, fully connected, and activation function. At present, the CVM Runtime execution framework developed by the Cortex Labs team can support computer vision CV research and some natural language processing NLP tasks, where CV contains image classification and object recognition.

Based on this, Cortex version 2.0 intends to implement more operators and expand the operator set to round out the on-chain AI model. On the one hand, Cortex 2.0 pays sustained attention to the new operators proposed by academia and industry. On the other hand, Cortex 2.0 extends the operator set to the field of natural language processing (speech, semantics, and text), adding LSTM [12], GRU [13], RNN [14], TRANSFORMER Operators such as [15], BERT [16] and other operators, which will enhance the inference capability of Cortex 2.0 in natural language tremendously.

In addition to the officially defined operators, Cortex 2.0 will also launch a custom operator function. Users can complete the custom operator according to the protocol and tools provided by Cortex, and upload the operator to the Cortex operator library for the extension. That extends the user-defined range from the model level to the operator level. Operators contributed by the community can effectively create a practicable operator library to meet their needs.

Besides, for the consideration of correctness and safety with user-defined operators, the CVM Runtime codebase will merge the customized operators only after completing the mathematical symbol description and code formal verification.

2.3 Fair Proof of Work Consensus: RandomAI

For a long time, the idea of one-machine-one-vote in the cryptocurrency community has not been materialized due to the design of the expensive ASIC chips which enhance the computational acceleration ratio. The community and academia have explored many memory bottleneck algorithms to be more friendly to the GPU and CPU mining without spending a lot of money on specialized mining equipment like ASIC. Ethereum's Dagger-Hashimoto [2] and Zcash's Equihash [17] are among those algorithmic practices that focus on GPU mining.

The Cortex chain will continue to prioritize one-machine-one-vote. The Cortex 1.0 adopts a proof-of-work scheme based on Cuckoo Cycle [18] to narrow the speed gap between CPU and special mining machine. In the Cortex 2.0, the RandomAI POW algorithm will be investigated and designed to ensure the fairness of the consensus algorithm.

The significant advantage of a general-purpose CPU over an ASIC mining machine is the versatility of executing code, so the PoW algorithm to be used must be dynamic. The existing RandomX generates a random calculation program, requires nodes to complete it, and submits the converted result as proof of work. This random calculation program can increase the CPU advantages against ASIC mining machines.

Cortex 2.0 will base on the concept of the RandomX algorithm and design RandomAI proof-of-work algorithm. As shown in the figure 2, RandomAI has three stages. Based on the information of the previous block (disordered memory state), the first stage uses the pre-defined CVM operator to generate random networks and data. In the second stage, random data are input into the random network to obtain the inferred result, and the third stage converts the inference result into the standard format of proof of work for submission. To meet the specific requirements of the POW result, the proof process needs to be repeated, constantly generate different simulation networks to try. In this process, due to the existence of a random simulation network, it cannot be accelerated by a custom-made circuit mining machine. Therefore, the RandomAI workload proof algorithm can effectively motivate nodes in the network to use general-purpose CPUs and GPUs for workload certification, improving the degree of decentralization for Cortex.

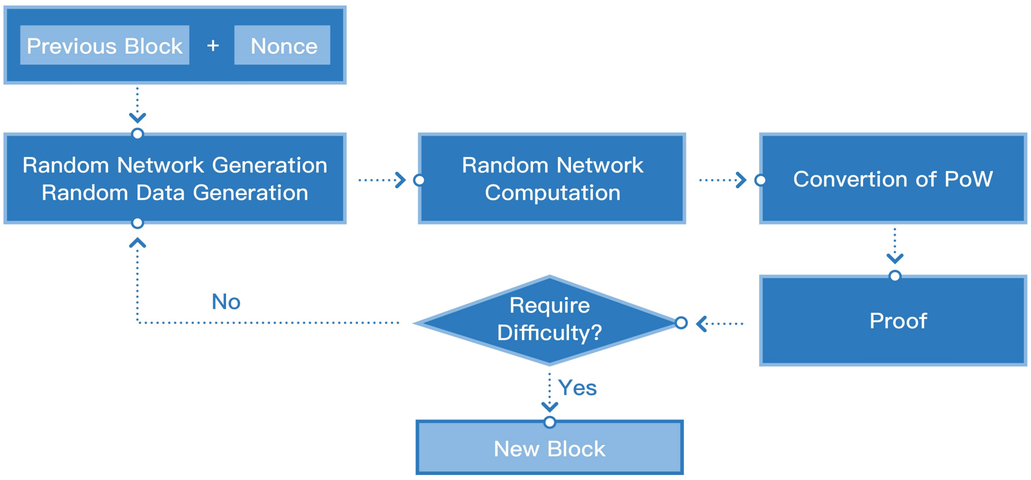

Figure 2: RandomAI PoW algorithm

2.4 Performance Improvement: three-phase of Zero Knowledge Proof

In the blockchain field, performance bottlenecks have always plagued relevant researchers to ensure the decentralization and security of the blockchain system. Up to now, there are many solutions to improve the blockchain performance, such as the consensus protocol replacement, DAG, zk-Rollup, sharding, and sidechains [19].

Due to the limitation of the CAP theorem of distributed systems, scaling up the blockchain will be an option, a trade-off between system consistency, availability, and persistence. The Cortex Labs team has conducted a series of in-depth research on the capacity expansion issue, hoping to improve the network performance without sacrificing core security assumptions. And we finally selected the zk-Rollup as the solution.

The zk-Rollup is a technology that uses zero-knowledge proof to put the calculation execution section off-chain. It does not require every node to participate in the calculation but only needs to verify the correctness of the calculation results. Traditional verification requires a recalculation of the verification result and takes up a lot of resources. And zk-Rollup reduces the computation overhead of verification hugely by adopting probabilistic confidence proof PCP, which guarantees the proof correctness under the maximum probability.

The world state of Cortex 2.0 is put in a Merkle tree and records all account balance, nonce, data, and so on. Each transaction causes a partial account state transfer in the world state. For the legality verification of state transfer brought about by transaction, each blockchain node needs to re-execute the transaction calculation process, resulting in wasted resources and performance bottlenecks.

Cortex 2.0 will use zk-SNARK [20] technology to separate the calculation process and the verification process. Each node merges the transactions and packages them for calculation, inputting The Merkle tree root and transaction set before the transfer and outputting the Merkle tree root in the world state after the transfer. The proof of calculation process will be generated by zk-SNARK and then submitted to the blockchain. Blockchain nodes can improve the performance of the entire blockchain significantly by just doing the verification for the correctness of the proof, which will represent the validity of state transfer of these transactions.

As shown in the figure 3, the expansion plan of Cortex 2.0 is divided into three stages: zero-knowledge proof for transfer transactions, zero-knowledge proof for smart contracts, and zero-knowledge proof for AI models. The first phase adopts the zk-Rollup solution to realize ZKP of transfer transactions and acceleration of packaging. In the second stage, by enhancing the versatility of the zero-knowledge proof circuit, zk-CVM is realized to perform ZKP and transaction packaging, to support smart contracts without AI inference. In the third stage, by adopting the

AI-compatible ZKP technology [21] of inference, the blockchain capacity will expand for transactions that include AI model inference.

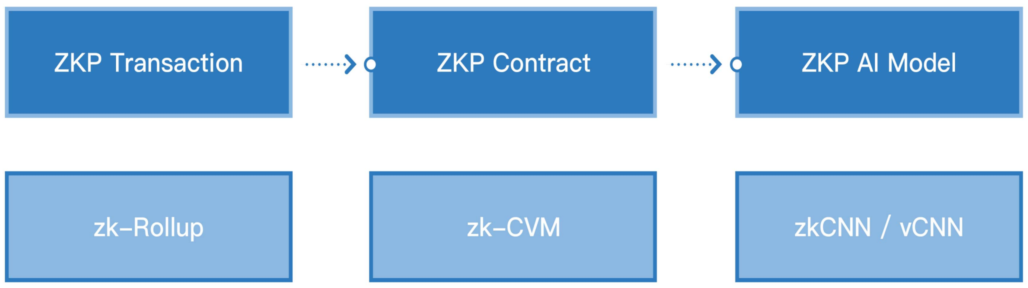

Figure 3: Cortex 2.0 Three-Phase Scaling

3 The Whole Architecture

To better serve the AI model developers and AI application developers, Cortex 2.0 will provide richer technical components in addition to the core framework, forming a complete AI framework and application ecosystem to help users enjoy the convenience brought by AI blockchain better. The overall architecture is shown in Figure 4.

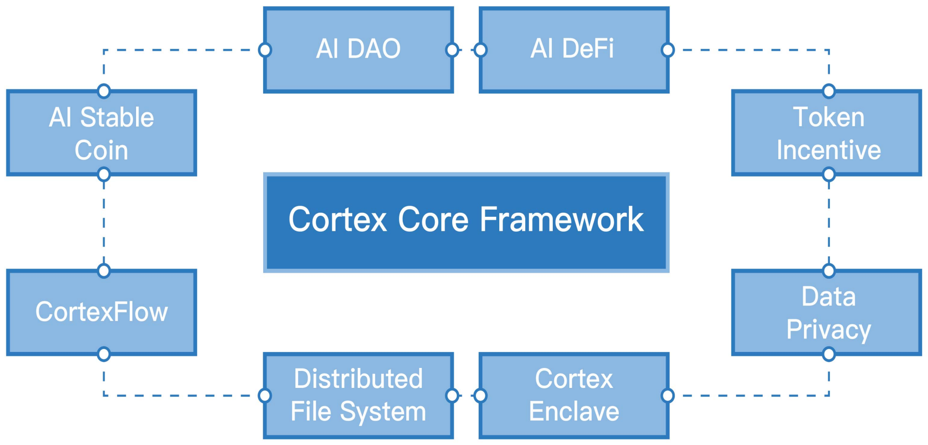

Figure 4: the Whole Architecture

3.1 Trusted Execution Environment: Cortex Enclave

SGX (Software Guard Extensions) proposed by the Intel Corporation is a popular solution in many designs of trusted computing. It aims to solve the problem of secure remote computing through trusted hardware. The trusted hardware will establish a dedicated secure container Enclave where computing service users upload calculations and data to the secure container, and SGX protects the privacy and security of data when performing calculations.

Cortex 2.0 will build a trusted execution environment based on SGX to execute AI model inference on the Cortex chain. On the one hand, SGX can ensure the correctness of the AI model inference. Whenever a node completes an AI model inference and packs it into blocks, other nodes do not need to repeat complex model inference calculations. The SGX signature can be certified to ensure that the calculation results are correct. It reduces the computational overhead caused by repeated execution of model inference. On the other hand, SGX can guarantee data privacy. Users can use the privacy mode to perform non-public calculations in SGX. This personal data can never be obtained externally, and that provides the privacy support for models and data.

On this basis, Cortex 2.0 plans to improve SGX and launch the Cortex Enclave, a trusted execution environment that combines software and hardware. The hardware part of Cortex Enclave is implemented based on the Risk-V architecture and provides data encryption at the hardware level. In addition, to support the rich on-chain ecosystem of Cortex 2.0, Cortex Enclave intends to embed in the above formalized AI operators to accelerate and optimize the operators from the hardware level, thereby improving the inference efficiency of the AI model in Cortex 2.0.

Cortex Enclave includes an enclave protection module, smart contract execution device, and encryption module. The enclave protection module builds a trusted computing environment on the chip to ensure the reliability of all data and execution actions, including but not limited to the execution of smart contracts, data signatures, sensor data collection, etc. The decoding conversion unit accepts the binary code input from the outside and uses the decoding device to de-analyze the binary code into the contract primitive sequence. By invoking the pre-embedded primitive meaning function, it then converts the reverse-analyzed primitive sequence into instruction code. The instruction execution unit parses the instruction code to execute. The processing result of the instruction execution unit is handed over to the encryption coprocessor to sign and encrypt. The trusted computing chip can ensure reliability in the prediction procedure and improve the execution efficiency of all Turing-complete smart contracts at the same time for the server to call. This technical solution has obtained related patents.

3.2 Distributed File System

Storage system is an important component in every public chain. The traditional public chain uses a linked storage structure of block data, and the bottom layer uses a key-value pair database such as levelDB to store data. This storage system ensures that all data passes the consensus of the entire network node, ensuring consistency and tamper-proof modification. However, due to high redundancy, this storage system has performance bottlenecks and capacity constraints, making it hard to store AI models and datasets.

Cortex has designed compatible storage systems for different data types, including high-security, high-redundancy on-chain storage, which adopts the traditional key-value pair storage system; and distributed file systems with large-scale and fast access that uses the Torrent File System, invokes the libtorrent library, and transmits the model and data dynamically through the DHT network. It would ensure the final consistency state of the model and data. The well-designed abstraction by Cortex makes it possible to use any key-value storage system to store models and data. Cortex's abstraction layer of data storage does not rely on any specific distributed storage solutions. DHT and IPFS can both be applied to solve storage problems.

Cortex's current storage capacity can support most typical applications such as pictures, voices, texts, and short videos, and is sufficient to cover most AI scenarios. For models and data that exceed the current storage limit, such as medical holographic scan data, which can be tens of GB per data piece, Cortex 2.0 will support it by increasing the storage limit. Cortex 2.0 version intends to extend the underlying distributed file system, and improve the scalability of the Cortex chain and the performance of transmission and storage in the storage layer by supporting more projects such as IPFS and databases.

3.3 Offchain AI Framework - CortexFlow

Model developers have become accustomed to completing model development and training in existing mainstream AI frameworks such as TensorFlow and PyTorch. Cortex has created Model Representation Tool (MRT) to help model developers upload models to the chain quickly and conveniently. MRT realizes the simple and efficient migration from the traditional AI model to the on-chain AI model with the determinism of the AI model.

There are many restrictions on the quantitative research of AI models under the traditional framework. The fundamental reason is that they do not have a theoretically complete solution for the consistency of floating-point model execution, which led to the redesign to a fixed-point AI execution framework: CVM Runtime and the realization of the migration tool MRT. Cortex 2.0 plans to develop an

AI framework CortexFlow, including but not limited to training, execution, and deployment, to help model developers better contribute in the Cortex ecosystem.

CortexFlow, the AI framework proposed by Cortex 2.0, contains many components such as model development, training, testing, etc. The model can be constructed directly using a formalized verification set of operators supported by Cortex 2.0. And user-defined operators can be imported into the set after passing the formalized verification. In the model training process, compared to the traditional model training method, CortexFlow automatically adds a fixed-point process during the training process. Then the deployment process would automatically convert the parameters obtained from training into a fixed-point model that can execute on the chain, and effectively ensure model accuracy to meet application requirements in industrial scenarios. The model developed and trained by CortexFlow can be directly deploy to the CVM Runtime for inference without any modification. At the same time, CortexFlow also provides a public test dataset to help the model testing. Part of the dataset comes from the Cortex 2.0 chain, which allows users to design the AI model of the on-chain data.

3.4 Privacy for Model and Data

The openness and transparency of the blockchain are a threat to the privacy of user data. In the AI field, model parameters and training/test data may all be private data. Relying on distributed storage alone can provide data availability, but not data privacy. To address this, Cortex 2.0 provides a full suite of privacy protection solutions, including data privacy and model privacy.

With the help of Cortex Enclave, Cortex 2.0 can help users store their models or data in the Enclave to protect privacy. Other nodes can input data into the Enclave, and the Enclave executes the inference and returns the result. This scheme guarantees the user model, data privacy, and inferred result correctness with the help of the Enclave model.

Cortex 2.0 intends to use Enclave, fully homomorphic encryption, and zero-knowledge proof to advance simultaneously. For public AI models, users can infer results and generate zero-knowledge proofs locally, then upload the proofs to nodes to verify and synchronize. Users can select a fully homomorphic encryption mode for AI model privacy. First, conduct homomorphic encryption on their inferred data, then upload the encrypted data for nodes for inference. Notice that the encrypted result can be decrypted back to the correct one.

3.5 Cross Chain

With the emergence of more and more blockchain systems, interoperability between multiple blockchain systems becomes an important issue. Cross-chain capability forwards transactions or contract calls on the source chain to the target chain. According to the trust foundation of cross-chain, there are three types of cross-chain technologies: atomic exchange, notary mechanism, and relay mechanism [22]. The operation of atomic exchange [23] is too complicated, it is limited to the exchange of assets on the chain and impossible to complete the cross-chain of contract calls. In the notary mechanism, trusted nodes monitor specific events in the source chain and send corresponding operations to the target chain. The signature is verified by the contract to ensure security, and most of the notary nodes are required to be trusted. The relay mechanism implements the light nodes of the source chain through smart contracts on the target chain and verifies the transaction records to the source chain. The cross-chain bridge on the relay mechanism does not need to trust the relay node.

Cortex 2.0 will enhance the interoperability of the Cortex chain with other blockchain systems by building a cross-chain bridge so that the AI capabilities on Cortex can be provided to more applications based on other blockchain infrastructures. Applications on other blockchain system can use the cross-chain bridge to call the AI contracts on the Cortex chain for AI inference, and then return the inference results to the blockchain to complete the function.

Cortex 2.0 will first implement a cross-chain bridge with a single-node notary mechanism to support two-way asset cross-chain. Then Cortex 2.0 will realize the cross-chain bridge on the multi-node notary mechanism, which supports two-way contract calls, and provide security through multi-signature. Finally, Cortex 2.0 will implement the cross-chain bridge of the relay mechanism to further improve the security of the cross-chain bridge.

4 Applications

4.1 AI DAO

Existing DAO(Decentralized Autonomous Organization) has certain restrictions in terms of function expression and usage scenarios due to the limitation of underlying smart contracts. Cortex 2.0 will launch AI DAO, promoting the development of decentralized autonomous organizations from the two directions: DAO in AI and AI in DAO.

4.1.1 DAO in AI

In the AI field, centralized platforms such as Kaggle hold AI competitions to promote innovation and development of AI models. These platforms provide a dataset for contestants to solve a specific issue. The competition often attracts AI developers from all over the world to participate in the competition, and submit their prediction results for ranking and rewards. However, AI competitions held by centralized platforms require data from companies to be submitted to the platform for scoring, which makes it difficult to ensure data privacy. Moreover, participants need to submit their model to the platform which is difficult to ensure the model privacy.

The Numer.ai project [24] tries to provide encrypted data sets for model developers to develop their models. Developers can then participate in competition by submitting predicted results data or models for ranking and receive cryptocurrency rewards. The Numer.ai project adopts the concept of DAO, but the process of model training, prediction, and firm offer cannot be supervised and certified through the blockchain, and it still requires centralized institutions for operation and credit guarantees.

In this scenario, Cortex 2.0 supports the implementation of decentralized AI competitions in the form of DAO and protects the privacy and security of models and data. The data provider can protect the data through fully homomorphic encryption. The participating teams complete the training and inference of the model off-chain and store the results and proofs on the blockchain.

This solution solves the two important problems between existing AI model owners and data owners. On the one hand, it can protect the privacy of data owners through homomorphic encryption, and the desired model can be obtained without exposing the data. At the same time, It also avoids the potential risk of user privacy leakage. On the other hand, it can protect the model owner's model privacy and ensure the correctness of the calculation results. The model owner does not need to provide a model. Through the calculation results and proof of the Cortex 2.0 zero-knowledge AI model, the correctness proof can be completed while protecting privacy.

4.1.2 AI in DAO

Current DAO only uses built-in fixed contract codes and helps organizations to govern through voting within a small range of choices. The capability of smart contracts greatly limits the development of DAO. Cortex 2.0 provides complete on-chain AI models, which enables DAO to achieve automation and intelligent management in more aspects, and further realize independent decision.

For example, existing DAO needs to design mechanisms such as holding tokens or NFTs to filter DAO members. This simple mechanism may lead to dangers such as sybil attacks, while complex mechanisms face block restrictions and high cost on existing blockchains. The AI model of Cortex 2.0 can efficiently solve this problem by collecting all historical transactions and relative data of DAO members on the blockchain. The models can quickly learn whether the members could be accepted by the DAO. Moreover, using the members' various operations in the DAO as feedback, AI DAO can further learn the various behaviors of the organization. With the increase of data, AI DAO can replace humans in more scenarios to complete smart decisions.

4.2 On-chain AI game

Existing blockchain games are limited by the functions and performance of smart contracts, and can only implement simple game logic. The AI model supported by Cortex gives games a richer design and more advanced gameplay. The AI models provides more logical judgments within the game, increasing the fun and ease of play in the games.

For example, CryptoKitties uses simple contracts and random numbers on the chain to complete gene combinations when breeding offspring. This logic is simple and easy to implement, but the playability is insufficient. Cortex 2.0 can enhance the playability of CryptoKitties on the chain through the AI model. The offspring combination can read the historical transaction data of the crypto cats on the chain, and get richer gene combination gameplay through the model.

4.3 On-chain AI financial services

4.3.1 AI DeFi Aggregator

Compared with traditional DeFi aggregators, Cortex 2.0 provides AI DeFi aggregator. It not only evaluates the profitability of each DeFi protocol, but also evaluates the potential risk behind each DeFi protocol that users are more likely to ignore. AI DeFi aggregator uses AI inference capabilities to make comprehensive decisions on investment strategies and portfolios. Starting from the original data on the chain of the DeFi protocol, a series of high-level features such as centralization, liquidity, and collateral are calculated. The score of each investment project is finally obtained and the optimal investment portfolio is obtained by the trained model. The combined inference result is provided to Ethereum's smart contract by the cross-chain bridge, and it can be automatically invested in different DeFi platforms. The solution offers a complete one-stop investment evaluation, consultation, and operation services for users.

In addition, investors only need to answer some investment questionnaires on the platform, and the AI model can assess the investor's risk preference, combine with the AI rating results of the DeFi platform, determine the financial management plan, and automatically generate the final personality Optimized investment allocation portfolio. The entire process takes only a few minutes, which achieves transparency, efficiency and accurate matching of user asset management goals.

4.3.2 Decentralized AI credit rating

As data on blockchain becomes more and more abundant, it becomes possible to rate each user's assets, risks, credit, etc. Lending based on credit rating becomes possible. Based on the AI models on the Cortex chain, a comprehensive rating of loan account data is given, and a reference credit rating can be calculated. The lending platform uses this as a basis to carry out credit lending business, combined with risk control methods such as staking. The current lending market is still stuck in the framework of over collateralization, while AI credit lending realize unsecured loan. Mortgage lending is indeed a relatively effective risk control strategy, but it has also formed some fundamental problems or a relatively obvious barrier to business development. Breaking through the barriers, we will face a new blue ocean market, which is undoubtedly worth looking forward to.

4.3.3 AI Stablecoin

Algorithmic stablecoin [25] is an important part of the blockchain financial ecosystem. The current algorithmic stablecoin relies on simple data such as prices collected by oracles to regulate supply, mortgage rate, and other parameters. Existing regulation algorithms are limited by the process capabilities of smart contracts and can only achieve rigid control. Such regulation algorithms are vulnerable to attacks such as lightning loan. The AI algorithmic stablecoin launched in Cortex 2.0 can mine potential data such as market sentiment through historical data, and prospectively regulate important parameters in the algorithmic stablecoin.

5 RoadMap

- **Ritchie 2022 Q1** Complete Solidity Compiler and MRT Upgrade.

- **Dijkstra 2022 Q2** Deploy the cross-chain bridge of the multi-node notary mechanism which is compatible with EVM. Publish papers for MRT quantification.
- **Bernoulli 2022 Q4** Complete the zk-Rollup for the transactions and contracts. Propose the cross-chain bridge of relay version.
- **Galileo 2023 Q4** Integrate the general distributed storage system architecture.

A Operators' Math Formalization

This section provides an extensive explanation to CVM-Runtime operators. Note that all numbers referred to by the symbol are integers by default.

All the operators' formalization obeys the unified format:

$$\begin{aligned} Y[y_{\text{indices}}] = X[x_{\text{indices}}], \\ \forall \text{given range}, \\ \text{where } \text{condition}_1 \text{ and } \text{condition}_2 \text{ and } \cdots \text{condition}_n \end{aligned}$$

which means that for given value range, the formula in the first line is always true, subjecting to the constraints listed as the condition statements.

A.1 Reduce Operators

A reduce operator performs the reduction function to input data based on the parameters, and the process logic over all kind of operators are the same. Reduction is performed on the given axes, other dimensions remains the same and the result are stored in those places. We abstract the common reduce logic as formalization here and specify the reduce function for each operators respectively.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$
- Output: Y
- Attribute:

- $axes$ (TShape), which is M-length vector, where $M \in [0, N+1)$
- $keepdims$ (bool)
- $exclude$ (bool)

$$T = \left\{ x \mid i \in \text{axes} \wedge x = \begin{cases} i, & \text{if } i \geqslant 0 \\ i + N, & \text{otherwise} \end{cases} \right\},$$

$$\text{where } card\{T\} = M \text{ and } j \in [0, N), \forall j \in \text{T}$$

$$U = \{0, 1, \cdots, N-1\}$$

$$R = \begin{cases} U - T, & \text{if exclude is true} \\ T, & \text{otherwise} \end{cases}$$

$$r = card\{R\}$$

1. Case *exclude* is **True** and $M = N$

$$Y = X$$

2. Case *exclude* is **False** and $M = 0$

$$Y[\underbrace{0, 0, \cdots, 0}_{K}] = \text{REDUCE\_FUNC}(X),$$

$$\text{where } K = \begin{cases} 1, & \text{if keepdims is false} \\ N, & \text{otherwise} \end{cases}$$

3. Case *keepdims* is **False**

$$\begin{aligned} Y[d_{I(0)}, d_{I(1)}, \cdots, d_{I(N-r-1)}] = \text{REDUCE\_FUNC}(Z), \\ \forall d_{I(i)} \in [0, n_{I(i)}) \wedge i \in [0, N-r), \\ \text{where } I : [0, N-r) \to U - R, \text{s.t. } \forall i < j, I(i) < I(j) \text{ and} \\ J : [0, r) \to R, \text{s.t. } \forall i < j, J(i) < J(j) \text{ and} \\ Z = \{X[d_0, d_1, \cdots, d_{N-1}] \mid d_i \in [0, n_i) \wedge i \in J\} \end{aligned}$$

4. Otherwise

$$\begin{aligned} Y[d_0, d_1, \cdots, d_{N-1}] = M[d_{I(0)}, d_{I(1)}, \cdots, d_{I(N-r-1)}], \\ \forall d_i \in [0, n_i) \wedge i \in U - R \wedge d_j = 0 \wedge j \in R, \\ \text{where } I : [0, N-r) \to U - R, \text{s.t. } \forall i < j, I(i) < I(j) \text{ and} \\ J : [0, r) \to R, \text{s.t. } \forall i < j, J(i) < J(j) \text{ and} \\ M = \text{OP\_NAME}(X, \text{axes=axes, keepdims=false, exclude=exclude}) \end{aligned}$$

A.1.1 sum

- Set OP_NAME as sum
- Set REDUCE_FUNC as

$$\text{REDUCE\_FUNC}(Z) = \sum Z$$

A.1.2 max

- Set OP_NAME as max

- Set REDUCE_FUNC as

$$\text{REDUCE\_FUNC}(Z) = \max Z$$

A.2 Broadcast Operators

A broadcast operator performs the broadcast function to input data, and the process logic over all kinds of operators are the same.

Math Formalization

- Input: There are 2 inputs.
  - A, a tensor of M dimensions, namely $(m_0, m_1, \cdots, m_{M-1})$
  - B, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$

  The two input shapes of tensor must satisfy the assertions as below:

  $P = \min(M, N)$

  $Q = \max(M, N)$

  $m_i = n_i$ or $m_i = 1$ or $n_i = 1, \forall i \in [0, P)$

- Output: Y, a tensor with Q dimensions, the higher dimension of the two inputs, and it's shape is identical to the input with higher dimension.

$$
\begin{aligned}
Y[d_0, d_1, \cdots, d_{K-1}] = A[a_0, a_1, \cdots, a_{M-1}] \text{ OP } B[b_0, a_1, \cdots, a_{N-1}],\\
\forall i \in [0, Q) \wedge d_i \in [0, \max(em_i, en_i)),\\
\text{where } a_j = d_{Q-M+j} \text{ if } d_{Q-M+j} < m_j \text{ else } 0, \forall j \in [0, M) \text{ and}\\
b_j = d_{Q-N+j} \text{ if } d_{Q-N+j} < n_j \text{ else } 0, \forall j \in [0, N) \text{ and}\\
em_i = \begin{cases} 1, & i < Q - M \\ m_{Q-M+i}, & \text{otherwise} \end{cases}, \forall i \in [0, Q) \text{ and}\\
en_i = \begin{cases} 1, & i < Q - N \\ n_{Q-N+i}, & \text{otherwise} \end{cases}, \forall i \in [0, Q)
\end{aligned}
$$

A.2.1 broadcast_add

set OP to add.

A.2.2 broadcast_sub

set OP to sub.

A.2.3 broadcast_mul

set OP to mutiply.

A.2.4 broadcast_div

set OP to divide.

A.2.5 broadcast_max

set OP to max.

A.3 NN Operators

We provide NN operators for users. In fact, NN operators stand for neural network operators, the core of neural network learning mechanism. NN operators have parameters to be trained and logic for linear or non-linear transformation in a model graph.

A.3.1 conv2d

We only supported 2-D convolution operator. Also alias Group-wise Convolution.

Math Formalization

- Input: there are 2 or 3 inputs.
  - X, input data to be calculated whose shape is (N, C, H, W)
  - W, convolution kernel weight whose shape is (OC, IC, KH, KW)
  - B, bias data. If B is not $None$, it's shape is (OC,).
- Output: Y
- Attributes:
  - **padding**, a **TShape** of length 2, namely $(PH, PW), PH, PW \in [min\_attr, max\_attr)$, indicating padding size.

- **stride**, a **TShape** of length 2, namely $(SH, SW) \in [1, max\_attr)$, indicating strides.
- **dilation**, a **TShape** of length 2, namely $(DH, DW) \in [1, max\_attr)$, parameter used in dilation convolution.
- **groups**, an **int** in range$[1, C]$, indicating group number. $C = IC \cdot \text{groups} \wedge OC \text{ mod groups} = 0$

$$Y[n, oc, p, q] = \sum_{ic=0}^{IC-1} \text{kernel}(n, (oc \div OPG) * IC + ic, p, q, oc, ic) + \begin{cases} 0, & \text{if B is None} \\ B[oc], & \text{otherwise} \end{cases},$$
$$\forall n \in [0, N) \wedge oc \in [0, OC) \wedge p \in [0, \text{Y\_HMAX}) \wedge q \in [0, \text{Y\_WMAX}),$$
$$\text{where Y\_HMAX} = \left\lfloor \frac{H + 2 \cdot \text{PH} - \text{DH} \cdot (\text{KH} - 1) - 1}{\text{SH}} \right\rfloor + 1 \text{ and}$$
$$\text{Y\_WMAX} = \left\lfloor \frac{W + 2 \cdot \text{PW} - \text{DW} \cdot (\text{KW} - 1) - 1}{\text{SW}} \right\rfloor + 1 \text{ and}$$
$$OPG = OC/\text{groups}, OPG \in \mathbb{N}^+ \text{ since } OC \text{ mod groups} = 0$$

where kernel function does the 2D image convolution calculation, and the formulation is:

$$\text{kernel}(n, j, p, q, o, i) = \sum_{k_i=0}^{\text{KH}} \sum_{k_j=0}^{\text{KW}} \text{pad}(p' + k_i * \text{DH}, q' + k_j * \text{DW}) \cdot W[o, i, k_i, k_j],$$
$$\text{where } p' = p \cdot \text{SH} - \text{PH} \text{ and } q' = q \cdot \text{SW} - \text{PW} \text{ and}$$
$$\text{pad}(p, q) = \begin{cases} X[n, j, p, q], & \text{if } p \in [0, H) \wedge q \in [0, W) \\ 0, & \text{otherwise} \end{cases}$$

A.3.2 dense

Dense operator provides a full connected layer.

Math Formalization

- Input: there are 2 or 3 inputs.
  - X, a matrix of shape (M, K)
  - W, a matrix of shape (N, K)
  - B, bias, of type **Optional<DLTensor>**. If B is not $NONE$, it's shape is $(N,)$.
- Output: Y, a matrix of shape (M, N)

$$Y = XW^T + \begin{cases} 0, & \text{if B is None} \\ B, & \text{otherwise} \end{cases}$$

A.3.3 relu

Relu performs elementwise rectified linear unit function.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{n-1})$
- Output: Y, the same shape as X

$$Y[d_0, d_1, \cdots, d_{n-1}] = max(0, X[d_0, d_1, \cdots, d_{n-1}]),$$
$$\forall i \in [0, N) \wedge d_i \in [0, n_i)$$

A.3.4 max_pool2d

Max_pool2d performs max pooling over every plane for each batch and channel.

Math Formalization

- Input: X, of shape (N, C, H, W), indicating there are N batches, C channels and all the planes are of size (H, W)
- Output: Y
- Attributes:
  - $pool\_size$, a $TShape$ of length 2, namely (PSH, PSW)
  - $padding$, either a $TShape$ of length 2, namely $(PH, PW) \in [min\_attr, max\_attr)$, or an int indicating $PH = PW$
  - $strides$, a $TShape$ of length 2, namely (SH, SW)
  - $ceil\_mode$, $boolean$.

$$\begin{aligned} PSH &\in [0, H + 2PH + 1), \\ PSW &\in [0, W + 2PW + 1), \\ PSH &> PH \wedge PSW > PW \end{aligned}$$

$$Y[n,i,p,q] = \text{max}\{\text{pad}(n,i,p',q')$$
$$\mid p' \in [p \cdot \text{SH} - \text{PH}, p \cdot \text{SH} - \text{PH} + \text{PSH}), q' \in [q \cdot \text{SW} - \text{PW}, q \cdot \text{SW} - \text{PW} + \text{PSW})\},$$
$$\forall n \in [0,N) \wedge i \in [0,C) \wedge$$
$$p \in \left[0, \text{ceil\_func}\left(\frac{H + 2 \cdot \text{PH} - \text{PSH}}{\text{SH}}\right) + 1\right) \wedge$$
$$q \in \left[0, \text{ceil\_func}\left(\frac{W + 2 \cdot \text{PW} - \text{PSW}}{\text{SW}}\right) + 1\right),$$
$$\text{where ceil\_func(val)} = \begin{cases} \lceil \text{val} \rceil, & \text{if ceil\_mode is true} \\ \lfloor \text{val} \rfloor, & \text{otherwise} \end{cases} \text{ and}$$
$$\text{pad}(n,i,p,q) = \begin{cases} X[n,i,p,q], & \text{if } p \in [0,H) \wedge q \in [0,W) \\ INT32\_MIN, & \text{otherwise} \end{cases}$$

A.3.5 upsampling

Upsampling operator performs upsampling to the input data by copying the value in each position serveral times.

Math Formalization

- Input: X, whose shape is (N,C,H,W)
- Output: Y
- Attributes: $scale$, in range $[1, max\_attr)$

$$Y[n, i, h, w] = X[n, i, \left\lfloor \frac{h}{\text{scale}} \right\rfloor, \left\lfloor \frac{w}{\text{scale}} \right\rfloor],$$
$$\forall n \in [0, N) \wedge i \in [0, C) \wedge h \in [0, H \cdot \text{scale}) \wedge w \in [0, W \cdot \text{scale})$$

A.4 Elemwise Operators

An elemwise operator performs the elementwise function to input data and the process logic over all kinds of operators are alike. There might be 1 or 2 input tensors and the logic might be complicated for someone. That's way we don't abstract them but describe each one.

A.4.1 abs

This operator calculates absolute value of input data.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n\_0, n\_1, \cdots, n\_N - 1)$.
- Output: Y, a tensor whose shape is same as X

$$Y[d_0, d_1, \cdots, d_{N-1}] = \begin{cases} x, & x \geqslant 0 \\ -x, & x < 0 \end{cases},$$
$$\forall i \in [0, N) \wedge d_i \in [0, n_i),$$
$$\text{where } x \text{ denotes } X[d_0, d_1, \cdots, d_{N-1}]$$

A.4.2 cvm_precision

The precision operator gives how many bits the absolute value of a number takes. 1 takes 1 bit. 2, 3 take 2 bits, etc. A special case is that 0 always takes at least 1 bit.

Math Formalization

- Input X, a tensor of N dimensions, namely $(n\_0, n\_1, \cdots, n\_N - 1)$.
- Output Y, a tensor whose shape is same as X

$$Y[d_0, d_1, \cdots, d_{N-1}] = \begin{cases} \lceil log_2(abs(x)+1) \rceil, & x \neq 0 \\ 1, & x = 0 \end{cases},$$

$$\forall i \in [0, N) \wedge d_i \in [0, n_i),$$

$$\text{where } x \text{ denotes } X[d_0, d_1, \cdots, d_{N-1}]$$

A.4.3 elemwise_add

This operator performs elementwise add to the 2 input tensors.

Math Formalization

- Input: there are 2 inputs, of the same shape.
  - A, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$.
  - B, whose shape is same as A.
- Output: Y, a tensor whose shape is same as A and B.

$$Y = A + B$$

A.4.4 elemwise_sub

This operator performs elementwise subtraction to the 2 input tensors.

Math Formalization

- Input: there are 2 inputs, of the same shape.
  - A, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$.
  - B, whose shape is same as A.
- Output: Y, a tensor whose shape is same as A and B.

$$Y = A - B$$

A.4.5 negative

This operator performs elementwise negative to the input tensor.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$.
- Output: Y, a tensor whose shape is same as X.

$$Y = -X$$

A.4.6 clip

This operator performs clip, cutting the data into a range, to the input tensor.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$.
- Output: Y, a tensor whose shape is same as X.
- Attributes:
  - a_min
  - a_max

$$Y[d_0, d_1, \cdots, d_{N-1}] = \begin{cases} \text{a\_max}, & x \geqslant \text{a\_max} \\ x, & x \in (\text{a\_min}, \text{a\_max}) \\ \text{a\_min}, & x \leqslant \text{a\_min} \end{cases},$$

$$\forall i \in [0, N) \wedge d_i \in [0, n_i),$$

$$\text{where } x \text{ denotes } X[d_0, d_1, \cdots, d_{N-1}]$$

A.4.7 cvm_cilp

This operator clips the input data into a certain CVM precision.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$.

- Output: Y, a tensor whose shape is same as X.
- Attribute: **precision**, an int in range $[1, 33)$

$$Y = clip(X, \text{a\_min} = -\alpha, \text{a\_max} = \alpha),$$
$$\text{where } \alpha = 2^{\text{precision-1}} - 1$$

A.4.8 cvm_right_shift

This operator performs right shift. Slightly different from C right shift, the result of this operator would be rounded to nearest integer. A special case is that negative half number will be rounded up, -1.5 rounded to -1 for example.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$.
- Output: Y, a tensor whose shape is same as X.
- Attribute:
  - **precision**, an int in range $[1, 33)$
  - **shift_bit**, an int in range $[1, 33)$

$$Y = clip(T, \text{a\_min} = -\alpha, \text{a\_max} = \alpha),$$
$$\text{where } T = \left\lfloor \left(\left\lfloor \frac{X}{2^{\text{shift\_bit}-1}} \right\rfloor + 1 \right) \div 2 \right\rfloor \text{ and } \alpha = 2^{\text{precision}-1} - 1$$

A.4.9 cvm_left_shift

This operator performs left shift to the input tensor, same as C left shift operator.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$.
- Output: Y, a tensor whose shape is same as X.
- Attribute:
  - **precision**, an int in range $[1, 33)$
  - **shift_bit**, an int in range $[1, 33)$

$$Y = clip(T, \text{a\_min} = -\alpha, \text{a\_max} = \alpha),$$
$$\text{where } T = X * 2^{\text{shift\_bit}} \text{ and } \alpha = 2^{\text{precision}-1} - 1$$

A.5 Transform Operators

transform operator do not do the calculation on the data but simply reshape, copy or select part of it. The process logic over all kinds of operators are quite different.

A.5.1 repeat

This operator repeats the input data by **repeats** times along the given **axis**. Each element is repeated right after itself.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{\text{axis}}, \cdots, n_{N-1})$
- Output: Y, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{\text{axis}} \cdot repeats, \cdots, n_{N-1})$
- Attribute:
  - **axis**, an int in range $[0, N)$, indicating on which axis is the repetition performed.
  - **repeats**, an int in range $[1, +\infty)$, indicating how many times the data is repeated.

$$Y[d_0, d_1, \cdots, d_{\text{axis}}, \cdots, d_{N-1}] = X[d_0, d_1, \cdots, \left\lfloor \frac{d_{\text{axis}}}{\text{repeats}} \right\rfloor, \cdots, d_{N-1}],$$
$$\forall d_0 \in [0, n_0) \wedge \cdots \wedge d_{axis-1} \in [0, n_{axis-1}) \wedge d_{axis} \in [0, n_{axis} \cdot \text{repeats}) \wedge$$
$$d_{axis+1} \in [0, n_{axis+1}) \wedge \cdots \wedge d_{N-1} \in [0, n_{N-1})$$

A.5.2 tile

This operator tiles the input data several times on several dimensions. Different from **Repeat** operator repeating each element right after itself, this operator tiles the whole data after the whole data.

Math Formalization

- Input: X, a tensor of :math:‘N‘ dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$,
- Output: Y
- Attribute: **reps**, a tensor of M dimensions, namely $(m_0, m_1, \cdots, m_{M-1})$.

$$r \in [1, max\_attr), \forall r \in \text{reps}$$

$$\begin{aligned}
Y[k_0, \cdots, k_{K-N-1}, k_{K-N}, k_{K-N+1}, \cdots, k_{K-1}] = \\
X[k_{K-N+0} \text{ mod } n_0, k_{K-N+1} \text{ mod } n_1, \cdots, k_{K-N+N-1} \text{ mod } n_{N-1}], \\
\forall k_0 \in [0, S_0) \wedge \cdots \wedge k_{K-1} \in [0, S_{K-1}), \\
\text{where } K = \text{max}\{M, N\} \text{ and } S_i = SX_i \cdot SR_i \text{ and} \\
SX_p = \begin{cases} n_{p-K+N}, & p \in [K-N, K-1) \\ 1, & p \in [0, K-N) \end{cases} \text{ and} \\
SR_q = \begin{cases} m_{q-K+M}, & q \in [K-M, K-1) \\ 1, & q \in [0, K-M) \end{cases}
\end{aligned}$$

A.5.3 flatten

This operator flattens the input tensor data to an array in a row-major order.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$.
- Output: Y.

$$\begin{aligned}
Y[\text{flatten\_index}(d_0, d_1, \cdots, d_{N-1}, n_0, n_1, \cdots, n_{N-1})] = \\
X[d_0, d_1, \cdots, d_{N-1}], \\
\forall d_0 \in [0, n_0) \wedge d_1 \in [0, n_1) \wedge \cdots \wedge d_{N-1} \in [0, n_{N-1})
\end{aligned}$$

where flatten_index is

$$\text{flatten\_index}(d_0, d_1, \cdots, d_{N-1}, n_0, n_1, \cdots, n_{N-1}) =$$
$$d_0 \cdot \prod_{i=1}^{N-1} n_i + d_1 \cdot \prod_{i=2}^{N-1} n_i + \cdots + d_{N-2} * n_{N-1} + d_{N-1}$$

A.5.4 concatenate

This operator concatenates tensors along a given axis.

Math Formalization

- Inputs: M tensors, namely $I^0, I^1, \cdots, I^{M-1}$. They all have :math:'N' dimensions, namely I^i's shape is $(n_0^i, n_1^i, \cdots, n_{N-1}^i)$. All dimensions except the axis to be concatenated along must have the same length.
- Output: Y
- Attribute: **axis**, an int in range $[0, N)$.

$$n_j^i = n_j^0, \forall i \in [1, M) \wedge j \in [0, N) \wedge j \neq \text{axis}$$

$$\begin{aligned} Y[d_0, d_1, \cdots, d_{\text{axis-1}}, \text{new\_idx}, d_{\text{axis+1}}, \cdots, d_{N-1}] = I^i[d_0, d_1, \cdots, d_{N-1}], \\ \forall d_0 \in [0, n_0^i) \wedge \cdots \wedge d_{N-1} \in [0, n_{N-1}^i) \wedge i \in [0, M), \\ \text{where new\_idx} = \sum_{j=0}^{i-1} n_{\text{axis}}^j + d_{\text{axis}} \end{aligned}$$

A.5.5 transpose

This operator transposes the input data with a certain sequence.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$
- Output: Y,
- Attribute: **axes**, a TShape of length $M \in \{0, N\}$, which means **axes** is either empty or a permutation of $\{0, 1, \cdots, N-1\}$

$$\text{axis} \in [-N, N), \forall \text{axis} \in \text{axes}$$

$$\begin{aligned} Y[d_{\text{real\_axes}_0}, d_{\text{real\_axes}_1}, \cdots, d_{\text{real\_axes}_{N-1}}] = X[d_0, d_1, \cdots, d_{N-1}], \\ \forall d_0 \in [0, n_0) \wedge \cdots \wedge d_{N-1} \in [0, n_{N-1}), \\ \text{where real\_axes}_i = \begin{cases} \text{axes}_i, & M = N \wedge \text{axes}_i \geqslant 0 \\ \text{axes}_i + N, & M = N \wedge \text{axes}_i < 0 \\ N - 1 - i, & M = 0 \end{cases} \text{ and} \\ card \left\{\text{real\_axes}_i \mid i \in [0, N)\right\} = N \end{aligned}$$

A.5.6 slice

This operator slices an input array with given attribute. For each dimension, strides (default to be 1), start (default to be 0) and end (default to be length of this dimension) coordinates can be assigned by user.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$
- Output: Y,
- Attributes:
  - **begin**, B dimensions.
  - **end**, E dimensions.
  - **strides**: S dimensions.

  B, E, S can be different.

$$\text{b\_arr}[b] = \begin{cases} \text{begin}[b] + n_i, & b \in [0, N) \wedge b < B \wedge begin[b] < 0 \\ \text{begin}[b], & b \in [0, N) \wedge b < B \wedge begin[b] \geqslant 0 \\ 0, & b \in [0, N) \wedge b \geqslant B \end{cases}, b \in [0, N)$$

$$\text{e\_arr}[e] = \begin{cases} \text{end}[e] + n_i, & e \in [0, N) \wedge e < E \wedge end[e] < 0 \\ \text{end}[e], & e \in [0, N) \wedge e < E \wedge end[e] \geqslant 0 \\ n_e, & e \in [0, N) \wedge e \geqslant E \end{cases}, e \in [0, N)$$

$$\text{s\_arr}[s] = \begin{cases} \text{stride}[s], & s \in [0, N) \wedge s < S \\ 1, & s \in [0, N) \wedge s \geqslant S \end{cases}, s \in [0, N)$$

$$\forall\{i \mid i \in [0, N)\} : \text{s\_arr}[i] \neq 0$$

$$\text{b\_range}(i) = \begin{cases} -1, & \text{s\_arr}[i] < 0 \\ 0, & \text{s\_arr}[i] \geqslant 0 \end{cases}$$

$$\text{e\_range}(i) = \begin{cases} n_i - 1, & \text{s\_arr}[i] < 0 \\ n_i, & \text{s\_arr}[i] \geqslant 0 \end{cases}$$

$$\text{b\_vec}[b] = clip(\text{b\_arr}[b], \text{a\_min} = \text{b\_range}(b), \text{a\_max} = \text{e\_range}(b) - 1), b \in [0, N)$$

$$\text{e\_vec}[e] = clip(\text{e\_arr}[e], \text{a\_min} = \text{b\_range}(e), \text{a\_max} = \text{e\_range}(e) - 1), e \in [0, N)$$

$$\forall\{i \mid i \in [0, N)\} : \begin{cases} \text{b\_vec}[i] < \text{e\_vec}[i], & \text{s\_arr}[i] > 0 \\ \text{e\_vec}[i] < \text{b\_vec}[i], & \text{s\_arr}[i] < 0 \end{cases}$$

$$\begin{aligned} Y[d_0, d_1, \cdots, d_{N-1}] &= X[K[0], K[1], \cdots, K[N-1]], \\ \forall d_j \in [0, &\left\lceil \frac{\text{e\_vec}[j] - \text{b\_vec}[j]}{\text{s\_arr}[j]} \right\rceil) \wedge j \in [0, N), \\ &\text{where } K[i] = \text{b\_vec}[i] + \text{s\_arr}[i] * d_i \end{aligned}$$

A.5.7 take

This operator takes some elements from the input data. If axis is not given, it flattens the input data and takes elements; if axis is given, takes the input data on this axis for every coordinates in other axes.

Math Formalization

- Input: there 2 inputs
  - X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$
  - $indices$, a tensor of M dimensions, namely $(m_0, m_1, \cdots, m_{M-1})$
- Output: Y,
- Attribute: **axis** None or int.

1. Case axis is **None** :

$$\begin{aligned} T &= flatten(X) \\ Y[d_0, d_1, \cdots, d_{M-1}] &= T[clip(\text{xidx}, \text{a\_min} = 0, \text{a\_max} = |T| - 1)], \\ &\forall (d_0, d_1, \cdots, d_{M-1}), \\ &\text{where } d_j \in [0, m_j) \wedge j \in [0, M) \text{ and} \\ &\text{xidx} = \text{indices}[d_0, d_1, \cdots, d_{M-1}] \end{aligned}$$

2. Case axis is **int** :

$$\text{axis} \in [-N, N)$$

$$\text{real\_axis} = \begin{cases} \text{axis}, & \text{axis} \geqslant 0 \\ \text{axis} + N, & \text{axis} < 0 \end{cases}$$

$$Y[d_0, d_1, \cdots, d_{M+N-2}] = X[d_0, \cdots, d_{\text{real\_axis}-1}, \text{xdix}, d_{\text{real\_axis}+M}, \cdots, d_{M+N-2}],$$

$$\forall d_j \in \begin{cases} [0, n_j), & j < \text{real\_axis} \\ [0, m_{j-\text{real\_axis}}), & j \in [\text{real\_axis}, \text{real\_axis} + M) \\ [0, n_{j-M+1}), & j \in [\text{real\_axis} + M, M + N - 1) \end{cases},$$

$$\text{where xidx} = clip(\text{indices}[d_{\text{real\_axis}}, d_{\text{real\_axis}+1}, \cdots, d_{\text{real\_axis}+M-1}],$$

$$\text{a\_min} = 0, \text{a\_max} = n_{\text{real\_axis}} - 1)$$

A.5.8 cvm_lut

This operator is a alias for a take where axis is None.

Math Formalization

- Input: there are 2 inputs.
  - X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$
  - $indices$, a tensor of M dimensions, namely $(m_0, m_1, \cdots, m_{M-1})$
- Output: Y.

$$Y = take(X, \text{indices}, \text{axis} = \text{None})$$

A.5.9 expand_dims

This operator expands some new dimensions right from the given axis.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$,
- Output: Y
- Attributes:
  - **axis**, an int in range $[-N-1, N+1)$, indicating where the new dimensions starts. Note that $N+1$, instead of N, will be added to negative axis.
  - **num_newaxis**, an int in range $[min\_attr, max\_attr)$

$$Y[d_0, d_1, \cdots, d_{\text{real\_axis}-1}, \underbrace{0, 0, \cdots, 0}_{\text{num\_newaxis}}, d_{\text{real\_axis}}, \cdots, d_{N-1}] = X[d_0, d_1, \cdots, d_{N-1}],$$

$$\forall d_0 \in [0, n_0) \wedge \cdots \wedge d_{N-1} \in [0, n_{N-1}),$$

$$\text{where real\_axis} = \begin{cases} \text{axis}, & \text{axis} \geqslant 0 \\ \text{axis} + N, & \text{axis} < 0 \end{cases}$$

A.5.10 reshape

This operator reshapes the input data.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$,
- Output: Y,
- Attribute: **target_shape**, a **TShape** of length **M**, namely $(m_0, m_1, \cdots, m_{M-1})$, s.t. $m_0 * m_1 * \cdots * m_{M-1} = n_0 * n_1 * \cdots * n_{N-1}$.

$$Y[d_0, d_1, \cdots, d_{M-1}] = T[\text{flatten\_index}(d_0, d_1, \cdots, d_{M-1}, m_0, m_1, \cdots, m_{N-1})],$$
$$\forall d_0 \in [0, m_0) \wedge \cdots \wedge d_{N-1} \in [0, m_{N-1}),$$
$$\text{where } T = \text{flatten}(X)$$

A.5.11 squeeze

This operator removes dimensions of length 1.

Math Formalization

- Input: X, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$
- Output: Y.
- Attribute: **axes**, a **TShape** of length M.

$$\text{axis} \in [-N, N), \forall \text{axis} \in \text{axes}$$

$$\text{real\_axes} = \begin{cases} \{\text{axis} \mid \text{axis} \geqslant 0 \wedge \text{axis} \in \text{axes}\} \bigcup \{\text{axis} + N \mid \text{axis} < 0 \wedge \text{axis} \in \text{axis}\}, & M > 0 \\ \{\text{axis} \mid n_{\text{axis}} = 1 \wedge \text{axis} \in [0, N)\}, & M = 0 \end{cases}$$

$$n_{\text{axis}} = 1, \forall \text{axis} \in \text{real\_axis}$$

$$\begin{aligned} Y[d_{I(0)}, d_{I(1)}, \cdots, d_{I(N-K-1)}] = X[d_0, d_1, \cdots, d_{N-1}], \\ \forall d_0 \in [0, n_0) \wedge d_1 \in [0, n_1) \wedge \cdots \wedge d_{N-1} \in [0, n_{N-1}), \\ \text{where } K = card\ \{\text{real\_axes}\} \text{ and} \\ I : \{i \mid i \in [0, N-K)\} \to \{i \mid i \in [0, N) \wedge i \notin \text{real\_axes}\}, \\ \text{s.t. } I(i) < I(j), \forall 0 \leqslant i < j < N - K \end{aligned}$$

A.5.12 where

This operator selects data from 2 inputs with condition given.

Math Formalization

- Input: there are 3 inputs
  - $Cond$, either a tensor whose shape is same as others, or a 1d tensor whose length is same as the length of others' first dimension.
  - A, a tensor of N dimensions, namely $(n_0, n_1, \cdots, n_{N-1})$
  - B, a tensor whose shape is same as A

- Output: :math:‘Y‘

1. Case shape of $Cond$ is same as others:

$$Y[d_0, d_1, \cdots, d_{N-1}] = \begin{cases} A[d_0, d_1, \cdots, d_{N-1}], & Cond[d_0, d_1, \cdots, d_{N-1}] \neq 0 \\ B[d_0, d_1, \cdots, d_{N-1}], & Cond[d_0, d_1, \cdots, d_{N-1}] = 0 \end{cases},$$

$$\forall d_i \in [0, n_i) \wedge i \in [0, N)$$

2. Case $Cond$ is a 1d tensor:

$$Y[d_0, d_1, \cdots, d_{N-1}] = \begin{cases} A[d_0, d_1, \cdots, d_{N-1}], & Cond[d_0] \neq 0 \\ B[d_0, d_1, \cdots, d_{N-1}], & Cond[d_0] = 0 \end{cases},$$

$$\forall d_i \in [0, n_i) \wedge i \in [0, N)$$

A.6 Vision Operators

We provide 2 operators for vision scenario. Since the scenario is narrow, regulation of the input data is stricter than other operators. If there’s no other specification, the input data should be 3D, namely (B, N, K), indicating number of batches, number of result for each batch and the length (K) of a result. The first value should be ID and the second should be the score.

A.6.1 get_valid_count

This operator counts how many results are more than a threshold and what are they.

Math Formalization

- Input: X, a 3D tensor of shape $(B, N, K), 2 \leqslant K \leqslant 32$
- Output:
  - valid_count,
  - Y,
- Attribute: **score_threshold**, an **int**.

$$\text{valid\_count}[b] = card\{q \mid q \in [0, N) \wedge X[b, q, 1] > \text{score\_threshold}\},\\ \forall b \in [0, B)$$

$$Y[b, \text{idx}, k] = X[b, n, k],\\ \forall b \in [0, B) \wedge n \in [0, N) \wedge k \in [0, K) \wedge X[b, n, 1] > \text{score\_threshold},\\ \text{where idx} = card\{q \mid q \in [0, n) \wedge X[b, q, 1] > \text{score\_threshold}\}$$

$$Y[b, n, k] = -1, \forall b \in [0, B) \wedge n \in [\text{valid\_count}[b], N) \wedge k \in [0, K)$$

A.6.2 non_max_suppression

This operator implements the nms algorithm, finding valid bounding boxes.

Math Formalization

- Input: there are 2 inputs.

  - X, a 3D tensor of shape $(B, N, K), K = 6$
  - **valid_count**, a 1D tensor of length B

- Output: Y
- Attributes:
  - **iou_threshold**, an **int**, representing the percentage of intersection over union with value in range $(0, +\infty)$ where 101 stands for bounding box full-overlap specifically and larger integer is equivalent to that.
  - **max_output_size**, an **int**
  - **force_suppress**, a **boolean**
  - **top_k**, an **int**.

$$\begin{aligned}
R[b, i, k] = X[b, I(i), k],\\
\forall b \in [0, B) \wedge i \in [0, T) \wedge k \in [0, K),\\
\text{where } T = \max\{\min(N, \text{valid\_count}[b]), 0\} \text{ and}\\
I : \{i \mid i \in [0, T)\} \to \{i \mid i \in [0, T)\},\\
\text{s.t. } X[b, I(i), 1] > X[b, I(j), 1] \vee\\
(X[b, I(i), 1] = X[b, I(j), 1] \wedge I(i) < I(j)), \forall 0 \leqslant i < j < T
\end{aligned}$$

$$
\begin{aligned}
&Y[b,n,k] = R[b,\text{IDX}(n),k],\\
&\forall b \in [0,B) \wedge n \in [0,\min\{T,\text{MOS},card\{U\}\}) \wedge k \in [0,K),\\
&\text{where TK} = \begin{cases} +\infty, & \text{if top\_k < 0} \\ \text{top\_k}, & \text{otherwise} \end{cases} \text{ and}\\
&\text{MOS} = \begin{cases} +\infty, & \text{if max\_output\_size < 0} \\ \text{max\_output\_size}, & \text{otherwise} \end{cases} \text{ and}\\
&\text{iou}(p,q) = \begin{cases} \text{OLR}(R[b,p,:],R[b,q,:]), & \text{force\_suppress is true or } R[b,p,0] = R[b,q,0] \\ 0, & \text{otherwise} \end{cases} \text{ and}\\
&\text{U} = \{p \mid p \in [0,min\{TK,T\}) \wedge R[b,p,0] >= 0 \wedge\\
&\text{iou}(p,q) < \text{iou\_threshold}, \forall q \in U \wedge q < p\} \text{ and}\\
&\text{IDX} : \{i \mid i \in [0,card\{U\})\} \rightarrow U, \text{s.t. IDX}(i) < \text{IDX}(j), \forall i < j
\end{aligned}
$$

$$
\begin{aligned}
&Y[b,n,k] = -1,\\
&\forall b \in [0,B) \wedge k \in [0,K) \wedge n \in [min\{T,\text{MOS},card\{U\}\},N)
\end{aligned}
$$

B The Models Implemented in CVM and Performance Testing

model	Jetson Nano-Cortex-A57(s)	Intel E5-2650(s)	Jetson Nano - GPU(128 CUDA Cores)(s)	1080Ti(3584 CUDA Cores)(s)
yolo_tfm			1.076	0.043
resnet50_mxg	1.2076	0.3807	0.147	0.009
resnet18_v1			0.055	0.004
qd10_resnet20_v2			0.064	0.010
resnet50_v2	1.4674	0.5005	0.185	0.010
qd10_resnet20_v2	0.2944	0.1605	0.065	0.012
trec	0.0075	0.0028	0.002	0.001
dcnet_mnist_v1	0.0062	0.0057	0.002	0.001
mobilenetv1.0_imagenet	0.3508	0.1483	0.039	0.002
resnet50_v1_imagenet	1.2453	0.3429	0.150	0.009
animal10	0.3055	0.1466	0.065	0.010
vgg16_gcv	4.3787	0.6092	0.713	0.021
sentiment_trec	0.0047	0.0022	0.002	0.001
vgg19_gcv	5.1753	0.7513	0.788	0.023
squeezenet_gcv1.1	0.3889	0.0895	0.044	0.002
squeezenet_gcv1.0	0.1987	0.1319	0.064	0.003
shufflenet	1.4575	0.7697	0.140	0.004
ssd			0.773	0.030
ssd_512_mobilenet1.0_coco_tfm			0.311	0.016
ssd_512_mobilenet1.0_voc_tfm			0.220	0.014

References

[1] Satoshi Nakamoto and A Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 4, 2008.

[2] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[3] Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183, 2019.

[4] Vitalik Buterin. On-chain scaling to potentially 500 tx/sec through mass tx validation. *Ethereum Blog*, 2018.

[5] Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, and Edward W Felten. Arbitrum: Scalable, private smart contracts. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1353–1370, 2018.

[6] Anatoly Yakovenko. Solana: a new architecture for a high performance blockchain v0. 8.14, 2021.

[7] Dmitry Tanana. Avalanche blockchain protocol for distributed computing security. In *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pages 1–3. IEEE, 2019.

[8] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[9] Ziqi Chen, Weiyang Wang, Xiao Yan, and Jia Tian. Cortex-ai on blockchain. *Cortex Labs Pte. Ltd., Singapore, Tech. Rep. C*, 201803307:2018, 2018.

[10] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[11] Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: introduction and applications. *Communications of the ACM*, 54(9):69–77, 2011.

[12] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[13] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[14] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[17] Alex Biryukov and Dmitry Khovratovich. Equihash: Asymmetric proof-of-work based on the generalized birthday problem. *Ledger*, 2:1–30, 2017.

[18] John Tromp. Cuckoo cycle: a memory-hard proof-of-work system. *IACR Cryptol. ePrint Arch.*, 2014:59, 2014.

[19] Bin Cao, Zhenghui Zhang, Daquan Feng, Shengli Zhang, Lei Zhang, Mugen Peng, and Yun Li. Performance analysis and comparison of pow, pos and dag based blockchains. *Digital Communications and Networks*, 6(4):480–485, 2020.

[20] Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016.

[21] Tianyi Liu, Xiang Xie, and Yupeng Zhang. zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy.

[22] Kaihua Qin and Arthur Gervais. An overview of blockchain scalability, interoperability and sustainability. *Hochschule Luzern Imperial College London Liquidity Network*, 2018.

[23] Maurice Herlihy. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM symposium on principles of distributed computing*, pages 245–254, 2018.

[24] Richard Craib, Geoffrey Bradway, Xander Dunn, and Joey Krug. Numeraire: A cryptographic token for coordinating machine intelligence and preventing overfitting. *Retrieved*, 23:2018, 2017.

[25] Amani Moin, Kevin Sekniqi, and Emin Gun Sirer. Sok: A classification framework for stablecoin designs. In *International Conference on Financial Cryptography and Data Security*, pages 174–197. Springer, 2020.