摘要

当今的区块链世界里，图灵完备的智能合约虚拟执行引擎得到了广泛应用，并吸引了大批应用开发者的关注和参与，基于此的去中心化金融、加密艺术、去中心化游戏等领域均得到长足发展。然而，基于区块链的世界计算机模型需要所有节点对计算结果进行验证和共识，这极大限制了智能合约的表示能力。现有链上智能合约语言以及虚拟执行机仅限于编写短小的程序和访问极少量的资源。而随着机器学习的发展，人工智能在图像识别、自然语言处理、模式识别等领域能发挥巨大作用，但机器学习模型对计算和存储有较大的需求，无法在区块链上得到应用。

Cortex 项目通过扩展智能合约底层指令集、增强存储层等技术，为智能合约增加人工智能算法的支持，使得任何人都可以为智能合约增加人工智能的能力。同时，Cortex 还提出了一种集体协作的激励机制，使得任何人都可以提交和优化 Cortex 上的模型，而模型的贡献者可以得到奖励。

Cortex 的最终目标是在于成为支持 AI 模型的更好的代币。为实现这一目的，Cortex 2.0 构建了更加完善的核心技术架构来增强链上 AI 智能合约的推断以及 Cortex 公链的安全性和性能，此外还通过形式化验证、可信执行环境等技术来增强 AI 推断的正确性、完备性以及隐私性。

Cortex 2.0 - 链上人工智能

The Cortex Team

Oct,30,2021

目录

1 介绍

2009 年，Bitcoin [1] 提出了区块链模型，能够支持去中心化、可信的数字货币交易系统。由于其内置的堆栈式语言并非图灵完备，譬如限制了循环等功能，这一设计限制了比特币系统的应用场景。2013 年，Ethereum [2] 项目将区块链上的可编程语言扩展到了图灵完备的 Solidity 语言，这一扩展为区块链系统带来了大量的应用。应对随之而来的图灵停机问题和拒绝服务攻击问题，Ethereum 提出了 Gas 机制，从经济学角度进行很好的解决。但是，Ethereum 的世界计算机模型仅能进行较小规模的计算和存储，这同样限制了 Ethereum 系统在人工智能等大数据场景的应用。后续出现的 PoS 公链 Algorand [3]，二层扩容方案 zkRollup [4]，Arbitrum [5]，新共识公链 Solana [6]、Avalanche [7]，去中心化存储 IPFS [8] 等方案，从不同角度提升了以太坊的性能，能运行更加复杂的智能合约，但是距离 AI 模型的计算和存储要求还有较大差距。

Cortex 项目 [9] 在 Ethereum 的基础上更进一步，打破了区块链系统和人工智能之间的障碍，为区块链系统引入了 AI 模型的分类、预测、生成等前所未有的功能。更大的突破带来了更多的挑战，为了应对区块链系统中的人工智能应用在计算、存储、网络等方面的负担，Cortex 提出了一系列方案进行解决：

- 实现 MRT 模型转换技术，将传统 AI 模型定点化；
- 提出 Cortex 虚拟机 CVM 实现链上 AI 推断计算；
- 引入 TorrentFS P2P 文件存储系统解决 AI 模型和数据的存储问题;

另一方面，由于 AI 技术需要的大规模数据和海量算力，而这两者具有聚集效应并主要掌握在大公司手中，可以预见的未来会形成垄断趋势并现已具雏形。为此，Cortex 系统提供了去中心化的 AI 模型市场，用户通过分享 AI 模型并从中获取收益，也使得更多人能自由享受 AI 技术的力量。我们相信，Cortex 项目会打破 AI 技术和普罗大众之间的藩篱，如同普罗米修斯为人类带来光明。

随着 Cortex Labs 在 2019 年 6 月完成了 1.0 主网上线，1.0 白皮书 [9] 中的计划基本实现，在区块链上部署运行 AI 模型已不再是天方夜谭。人工智能技术从垄断数据和算力的公司走了出来，去中心化的人工智能技术带给了我们更多的未来。每个普通人，都可以通过 Cortex 区块链享受到人工智能技术带来的便利，并且完全清楚了解自己的数据是如何被使用的。每个开发者，都能借助于人工智能技术构建更多可能的应用，充分发挥自己的天才想法，并从中受益。

Cortex Labs 不会停滞不前，整个项目将在系统架构、软件实现、硬件实现、应用生态等多方面取得进一步的突破。随着 Cortex 2.0 的完成，我们将看到一个功能更加完善，性能更加强大，协议更加稳定的去中心化 AI-on-blockchain 系统以及相辅相成的应用生态。

本文主要介绍 Cortex 2.0 设计的愿景，从核心架构和整体框架两个角度进行了详细的描述。其中核心架构主要围绕 Cortex 公链本身，介绍如何进一步提升性能、安全性以及去中心化程度。整体框架则是围绕 Cortex 公链进行的一系列链上链下工作介绍，包括帮助更加便利的使用 Cortex 区块链，保护用户的模型和数据隐私，更加易用、好用的链上 AI 生态等等。

2 核心架构

为了构建更加完善的支持 AI 模型的公链，Cortex 2.0 需要优化 AI 模型推断以及公链两个方面，一方面需要满足 AI 模型在链上执行的正确性以及功能的完备性，另一方面需要在共识、性能方面对现有 Cortex 链进行优化。Cortex 2.0 核心架构如图1所示，主要包含以下方面的技术突破：

1. 形式化验证: 通过 Z3 证明器 [10] 完成 AI 算子的形式化以及正确性验证，保证 Cortex 系统中所有节点对 AI 模型的推断结果保持一致以及正确。

2. AI 算子库：进一步完善 Cortex 支持的 AI 模型底层算子库，使得 Cortex 能实现更多 AI 模型的推断工作。

3. 共识算法：设计 RandomAI 工作量证明算法，进一步提升 Cortex 的去中心化程度。

4. 性能提升：通过零知识证明技术，逐步实现转账交易、智能合约以及 AI 推断的打包，提升 Cortex 主链性能。

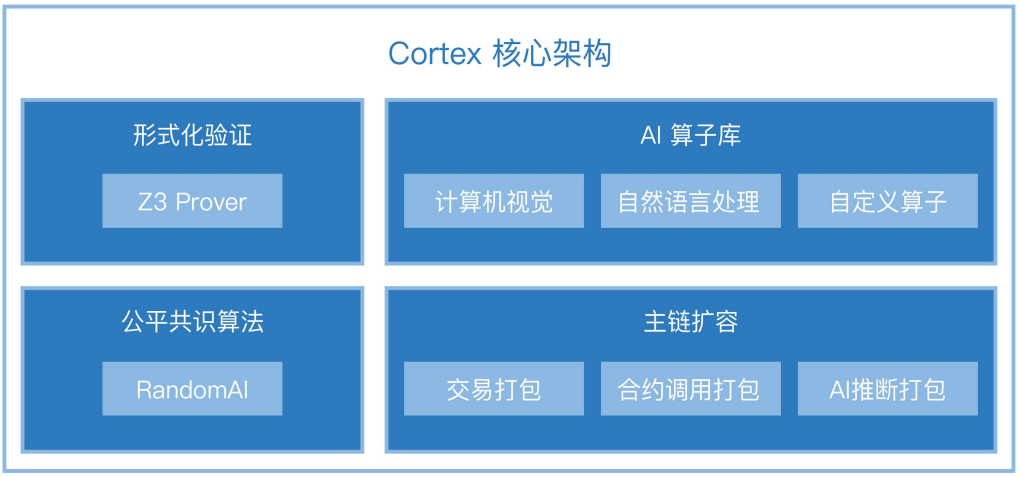

图 1: Cortex 2.0 核心架构

2.1 形式化验证: Z3-Prover

由于区块链上智能合约虚拟机中的指令执行和计算结果属于共识机制，这要求虚拟机中指令操作是确定性和可复现的，Cortex 1.0 将 AI 模型推断操作作为一条基本指令（INFER | IFNERARRAY）集成到虚拟机执行引擎（CVM）中，由此引申出 AI 推断操作在区块链上应具备的两大重要特性：确定性和可复现。

Cortex Labs 团队对上述两大重要特性给予足够的重视，并提出或计划提出一系列的可解释模型或方法，以保障定点化 AI 框架（CVM Runtime）中推断操作的完备性：

- 撰写并发布 MRT 量化论文以介绍定点化 AI 框架的必要性，完备性以及相关模型转换方法；

- 利用数学上严格的描述符号定义 AI 框架中算子的输入，输出和操作逻辑，保障算子计算执行具备理论可验证性；
- 采用第三方库 Z3-Prover 来验证 CVM Runtime 项目库中算子代码实现的正确性；

Cortex 2.0 计划发布的 MRT 量化论文主要介绍 Cortex Labs 团队为定点化 AI 框架 CVM Runtime 做出的相关模型转换工作，包含且不限于模型定点化的必要性，相关研究和主要成果。模型定点化的必要性主要在于区块链上计算要求的确定性同现有主流框架中浮点模型在并行架构中计算不确定性之间的矛盾，为此调研了相关量化研究论文并参考现有成果实现了适配定点化框架的 MRT 转换工具，更多细节请参考论文说明，第 5 节路线图中介绍了发布计划。

此外，Cortex 2.0 对 CVM Runtime 中已支持的算子代码进行逻辑抽象，形成数学上的严格操作定义，以保障理论上的可解释性，一致性和完备性。通过对算子的输入，输出和其他配置属性进行规范和前置说明，以数学表达式的方式对代码隐含的逻辑进行抽象并给出等式符号定义。在可能存在的不同推理代码版本出现计算结果不一致时，这种数学描述的形式符号能够为其提供理论完备的参考并给出唯一确定的计算结果。当前 CVM Runtime 项目库中已支持的算子数学符号描述已构建完善并添加至附录A。

最后，Cortex 2.0 拟借助可满足模理论（SMT）[11] 来完成代码库中算子实现的形式化验证。SMT 是对将布尔可满足性（SAT）的推广，添加了等式推理、算术、固定大小的位向量、数组、量词、以及其他有用的一阶理论。SMT 求解器是决定这些理论中公式的可满足性 (或双重有效性) 的工具，可以用于例如扩展静态检查、谓词抽象、测试用例生成和无限域上的有界模型检查等应用程序。

Z3 形式化验证器（Z3-Prover）[10] 是美国微软公司研发的一款新型 SMT 求解器。它的目标是解决问题在软件验证和软件分析中出现的问题，且整合了各种最新且全面广泛的理论研究工作且实现在项目库中供代码分析，代码审计等。Cortex 2.0 将调用 Z3-Prover 库对 CVM Runtime 中所有算

子进行代码的形式化验证，通过定义每个算子的输入和配置属性的数据规模和范围，验证输出或中间计算结果是计算无误且精度无溢出的。

2.2 链上 AI 推断引擎：更完善的算子库

CVM Runtime 项目库中定义了一系列算子集及其实现，并给出了严格的数学描述定义，规定算子在给定输入的情况下，按算子计算逻辑，输出确定性的结果。已支持的算子集合参考现有主流深度学习框架架构，结合常用 AI 模型涉及到的网络结构，囊括了诸如卷积，全联接，激活函数等必要算子集。目前，Cortex Labs 研发的 CVM Runtime 模型执行框架能够支持图像分类，物体识别等计算机视觉 CV 研究以及部分自然语言处理 NLP 的任务。

Cortex 2.0 版本在此基础上，拟进一步扩充算子集，通过实现更多算子提供更加完善的链上 AI 模型。一方面，Cortex 2.0 持续关注学术界与工业界提出的新模型中新增的算子。另一方面，Cortex 2.0 将算子集扩充到自然语言处理（语音，语义，文本）领域，增加 LSTM [12]、GRU [13]、RNN [14]、TRANSFORMER [15]、BERT [16] 等算子，极大增强 Cortex 2.0 在自然语言方面的推断能力。

除官方定义的算子外，Cortex 2.0 还将推出自定义算子功能，用户可以根据 Cortex 提供的协议以及工具完成自定义算子，并将算子上传到 Cortex 算子库中进行扩展，这使得用户自定义的范围从模型级别扩展到算子级别。社区贡献的算子可以有效创建更加完善的算子库，满足社区的需求。

为保证用户自定义算子的正确性以及安全性，所有算子均需在完善数学符号描述和代码形式化验证之后才能合并至 CVM Runtime 代码库中。

2.3 公平的工作量证明：RandomAI

一直以来，一机一票的加密数字货币社区设想并未实现。原因是 ASIC 的特殊设计使得计算加速比得到大幅提升。社区和学术界探索了很多内存瓶颈算法来对显卡和 CPU 挖矿更加友好，而无需花费大量资金购买专业

挖矿设备。近年来社区实践的结果显示，以太坊的 Dagger-Hashimoto [2] 和 Zcash 的 Equihash [17] 是比较成功的显卡优先原则的算法实践。

Cortex 链将进一步秉承一机一票优先，Cortex 1.0 版本采用基于 Cuckoo Cycle [18] 的工作量证明方案，缩小 CPU 和矿机之间加速比的差距。Cortex 2.0 版本中，将考察并设计 RandomAI 工作量证明算法，进一步保证共识算法的公平性。

通用型 CPU 相对于 ASIC 矿机的最大优势在于执行代码的通用性，因此采用的 PoW 算法必须是动态的。现有的 RandomX 通过生成随机计算程序，并要求节点完成生成的随机程序，提交转换后的结果作为工作量证明。这一随机计算程序可以提升 CPU 的优势，来对抗 ASIC 矿机。

Cortex 2.0 延续 RandomX 算法的思想，设计了 RandomAI 工作量证明算法。如图2所示，RandomAI 主要分为三个阶段，第一阶段根据前一个区块的信息 (无序内存状态)，利用 CVM 中给定的基算子随机生成随机网络和随机数据，第二阶段将随机数据输入到随机网络中获取推断结果，第三阶段将推断结果转换为工作量证明的标准格式进行提交。为了满足特定要求的工作量证明结果，该证明过程需要重复进行，不断生成不同的模拟网络进行尝试。这一过程中，由于随机模拟网络的存在，并不能通过定制电路的矿机获得加速。因此 RandomAI 工作量证明算法能有效激励网络中的节点采用通用的 CPU 和 GPU 进行工作量证明，可以有效提升 Cortex 的去中心化程度。

2.4 主链扩容: 零知识证明三部曲

区块链领域中，为了保障区块链系统的去中心化程度以及安全性，性能瓶颈一直困扰着相关研究者。为了提升区块链性能，目前主要有更换共识协议、DAG、zkRollup、分片、侧链等方案 [19]。

由于分布式系统 CAP 定理的限制，直接对区块链进行扩容将是一种权衡，在系统一致性、可用性和持久性之间进行折衷选择。Cortex Labs 通过对扩容问题进行了深入研究，希望在不牺牲核心安全假设的前提下提升网络的性能，最终选定了 zkRollup 扩容方案。

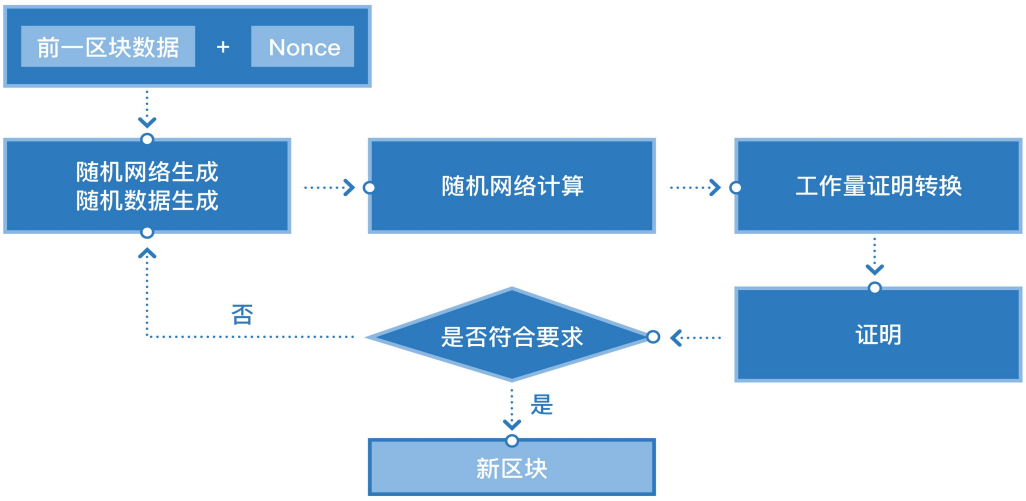

图 2: RandomAI 工作量证明算法

zkRollup 方是一种利用零知识证明将计算执行环节放在链下执行的技术，不需要每个节点都参与计算，只需要验证计算结果的正确性即可。相比于传统的验证需要重新计算一遍核对结果，zkRollup 技术采用概率可信证明 PCP，在极大概率下保障证明正确性，同时大大降低验证的计算开销。

Cortex 2.0 的世界状态存储在一棵 Merkle 树中，记录所有账户的余额、nonce、数据等数据。每笔交易会导致世界状态中部分账户状态转移。为了验证交易带来的状态转移是合法的，区块链上每个节点都需要重新执行该交易的计算过程，带来资源浪费以及性能瓶颈。

Cortex 2.0 采用 zkSNARK [20] 技术，将计算过程和验证过程拆开，各节点将需要打包的交易合并进行计算，转移前的世界状态 Merkle 树根和交易集合作为输入，转移后的世界状态 Merkle 树根作为输出，并将这一计算过程通过 zk-SNARK 生成证明提交到区块链上。区块链上的节点只需要验证该证明是否正确，即可判断这些交易的状态转移是否合法，可以极大提升整个区块链的性能。

如图3所示，Cortex 2.0 的扩容方案主要分为三个阶段：转账交易的零知识证明，智能合约的零知识证明，AI 模型的零知识证明。第一阶段采用 zkRollup 方案，实现对转账交易的零知识证明以及打包加速。第二阶段通过

增强零知识证明电路的通用性，实现 zk-CVM，可以对 CVM 中运行的非 AI 推断智能合约进行零知识证明并将交易打包上链。第三阶段通过采用适宜 AI 推断的零知识证明技术 [21]，对包含 AI 模型推断的交易实现 ZK 扩容。

图 3: Cortex 2.0 扩容方案三阶段

3 整体架构

为了更好地服务 AI 模型开发者以及 AI 应用开发者，Cortex 2.0 除了核心框架之外，还提供更加丰富的技术组件，构成一个完善的 AI 框架和应用生态，帮助用户更好享受 AI 区块链带来的便利。整体架构如图4所示。

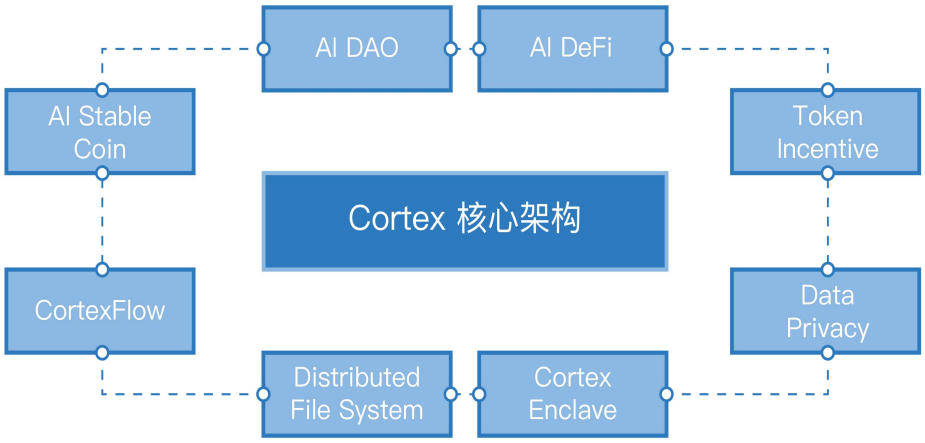

图 4: Cortex 2.0 整体架构

3.1 可信执行环境: Cortex Enclave

Intel 提出的 SGX（Software Guard Extensions）是一系列可信计算设计中广受欢迎的方案，旨在通过可信硬件来解决安全的远程计算问题，可信硬件中建立一个专用的安全容器 Enclave，远程计算服务用户将计算和数据上传到安全容器中，SGX 在执行计算时会保护数据的私密性和安全性。

Cortex 2.0 将基于 SGX 构建可信执行环境，用于执行 Cortex 链上的 AI 模型推断。一方面，SGX 可以保证 AI 模型推断的正确性，每当节点完成 AI 模型的推断并打包出块后，其他节点不需要重复进行复杂的模型推断计算，可以通过验证 SGX 签名来确保计算结果的正确性，减少了重复执行模型推断带来的计算开销。另一方面，SGX 可以保证数据的私密性，部分模型以及数据为用户的隐私信息，用户可以采用隐私模式，在 SGX 进行隐私计算，外部无法获取这些隐私数据，为模型和数据的隐私性提供了支持。

在此基础上，Cortex 2.0 未来将对 SGX 进行进一步改进，推出软硬件结合的可信执行环境 Cortex Enclave。Cortex Enclave 的硬件部分基于 Risk-V 架构实现，提供硬件层面上的数据加密。此外，为支持 Cortex 2.0 更加丰富的链上生态，Cortex Enclave 拟内置上述形式化 AI 算子，从硬件层面对算子进行加速优化，从而大幅提升 Cortex 2.0 中 AI 模型推断的性能效率。

Cortex Enclave 包括 enclave 保护模块、智能合约执行装置和加密模块。其中 Enclave 保护模块用于构建该芯片上的可信计算环境，保证芯片上所有数据和执行动作的可靠性，包括并不限于智能合约的执行、数据的签名、传感器数据的收集等；解码转化单元接受外部输入的二进制码，利用解码装置将二进制码反解析出合约原语序列，并根据解码装置中预嵌入的原语含义功能，将反解析出的原语序列转换生成指令执行单元所需要的指令代码，指令执行单元的处理结果交由加密协处理器对计算结果进行签名加密。可信计算芯片能够有效保证芯片在计算过程中的可信度，同时能够有效提升所有图灵完备的智能合约的执行效率，以供服务端调用。该技术方案已获取相关专利。

3.2 分布式存储

存储系统是支撑任何公链的重要组件，传统公链都采用区块链式存储结构，底层采用 levelDB 等键值对型数据库存储数据。这一存储体系保证了所有数据经过全网节点共识，保障一致性和防篡改性。但是这一存储体系由于较高的冗余，存在性能瓶颈和容量限制，不能直接应用于 AI 模型以及数据集的存储。

Cortex 针对不同类型的数据设计了不同的完整存储体系，包括高安全、高冗余的链上存储以及大规模、快捷访问的分布式文件系统。其中链上存储采用传统的键值对存储系统，分布式文件系统方案借鉴 Torrent File System 思路，调用 libtorrent 库，通过 DHT 网络来动态传播模型和数据，保证最终模型数据的状态一致性。

Cortex 设计上的良好抽象使得可以使用任何键值对存储系统来存储模型及数据。Cortex 的数据存储抽象层并不依赖于任何具体的分布式存储解决方案，分布式哈希表或者 IPFS 都可以用来解决存储问题。

Cortex 当前的存储能力，能够支持目前图片、语音、文字、短视频等大部分典型应用，足以覆盖绝大多数人工智能场景问题。而对于超出当前存储限制的模型和数据，比如医疗全息扫描数据，一条数据可能数十 GB，Cortex 2.0 将通过提升存储限制进行支持。Cortex 2.0 版本拟扩展底层分布式文件系统，通过支持 IPFS，数据库等更多存储系统来提升 Cortex 系统的扩展性以及存储层传输和存储性能。

3.3 链下 AI 框架 CortexFlow

模型开发者已经习惯于在 TensorFlow, PyTorch 等现有主流 AI 框架中完成模型的开发和训练，为了帮助模型开发者快速便捷的将模型上链，Cortex 打造了整套 AI 模型迁移工具 MRT（Model Representation Tool），从 AI 模型的定点化、实现传统 AI 模型往链上 AI 模型的简洁高效迁移。

传统框架下 AI 模型的量化研究方向存在诸多限制，最根本的原因是不具备理论上完备的浮点模型执行结果一致性解决方案，导致了 Cortex 重新

设计并实现了迁移工具 MRT 和完整的定点化 AI 执行框架：CVM Runtime。Cortex 2.0 计划开发完整的 AI 框架 CortexFlow，包括不限于训练，执行和部署等，以帮助模型开发者更好的参与进 Cortex 开发者生态。

Cortex 2.0 提出的 AI 框架 CortexFlow 中，包含模型开发、训练、测试等组件，其中模型构建可以直接采用 Cortex 2.0 支持并经过形式化验证的算子集合进行开发，同时可以通过导入的方式加入用户自定义并且经过形式化验证的算子集合。模型训练的过程中，相较于传统模型的训练方式，CortexFlow 在训练过程中自动加入定点化过程，保证模型训练得到的参数在部署过程中自动转换为链上可运行的定点化模型，并有效保证模型精度，满足工业场景下的应用要求。经过 CortexFlow 开发和训练的模型，可以直接部署到 CVM Runtime 中进行推断，无需任何修改。同时，CortexFlow 也提供了公开的测试数据集帮助模型进行测试。部分测试数据集来源于 Cortex 2.0 链上数据，支持用户设计链上数据的 AI 模型。

3.4 模型及数据隐私

区块链公开透明的属性对于用户数据的隐私是一种威胁，在 AI 领域，模型参数以及训练/测试数据均可能为隐私数据。仅依靠分布式存储可以提供数据可用性，但无法提供数据隐私性。为了解决这一问题，Cortex 2.0 计划提供整套隐私保护方案，包含数据隐私保护以及模型隐私保护。

借助 Cortex Enclave 的能力，Cortex 2.0 可以帮助用户将模型或者数据存储到 Enclave 中来保护隐私性，其他节点可以将数据输入到 Enclave 中，由 Enclave 完成推断过程并返回推断结果。这一方案保证用户模型与数据隐私性的同时，也由 Enclave 保障模型推断结果的正确性。

Cortex 2.0 拟采用 Enclave、全同态加密和零知识证明三套方案同步推进，对于公开 AI 模型，用户可以在本地完成推断并生成零知识证明，上传证明供节点进行验证和同步。对于支持全同态运算的隐私 AI 模型，用户可以采用全同态加密模式，先对自己的推断数据进行同态加密，然后上传加密后的数据供节点完成推断。加密推断结果可以解密还原为正确的推断结果。

3.5 跨链

随着越来越多区块链的出现，多个区块链之间的互操作性成为了一个重要的问题。跨链技术将源链上的交易或者合约调用转发到目标链上。根据跨链信任的基础，目前主要有原子交换、公证人机制以及中继机制三类跨链技术 [22]。其中原子交换技术 [23] 操作过于复杂，而且局限于链上资产交换，无法完成跨链合约调用。公证人机制由可信任的节点监听源链特定事件，将对应操作发送到目标链上，由合约验证签名来保证安全，需要大部分公证人节点可信。中继机制在目标链上通过智能合约实现源链的轻节点，验证交易记录到源链，中继机制无需信任中继节点。

Cortex 2.0 将通过搭建跨链桥增强 Cortex 链与其他公链的互操作性，使得 Cortex 上的 AI 能力能提供给更多链上应用。其他链上的应用可以通过跨链桥调用 Cortex 链上的 AI 合约完成 AI 推断，再将推断结果返回该链完成功能实现。

Cortex 将首先实现单节点公证人机制的跨链桥，支持双向资产跨链。然后实现多节点公证人机制的跨链桥，支持双向合约调用，通过多签提供安全性。最后，Cortex 2.0 将实现中继机制的跨链桥，进一步提升跨链桥的安全性。

4 应用

4.1 AI DAO

现有 DAO 组织受限于智能合约的约束，在功能表达和使用场景方面都有一定的限制。Cortex 2.0 推出 AI DAO，从 DAO in AI 和 AI in DAO 两个方向上推动去中心化自治组织的发展。

4.1.1 DAO in AI

AI 领域中，为了推动 AI 模型的创新发展，Kaggle 等中心化的平台举办 AI 比赛，由项目方提供具体的数据训练集以及测试集，召集全球的队伍

共同参加比赛，通过提交模型的计算结果进行排名。但中心化的平台举办的 AI 比赛，要求数据提交到该平台用于评分，难以保证数据的隐私性。另一方面，参与者需要提交模型至平台上，借由对平台项目方的完全信任来评判模型的优劣，难以保证模型的隐私性。

Numer.ai 项目 [24] 尝试提供加密数据集供模型开发者训练，开发者可以通过提交预测数据可以参与排名，或者提交模型进行实盘运行，根据实盘运行结果可以获取对应的 Numeraire 代币奖励。该项目已经有了 DAO 组织的精神和雏型，但是其中模型训练、预测以及实盘等过程均无法通过区块链进行监督和认证，仍然需要中心化机构进行运营和信用担保。

在该场景中，Cortex 2.0 支持采用 DAO 的形式实现去中心化的 AI 比赛，并且能保护模型以及数据的隐私安全。数据提供方可以通过全同态加密的方式对数据进行保护，参赛队伍在链下完成模型的训练和推断，并将结果以及证明存储在区块链上。

这一方案解决了现有 AI 模型所有者和数据所有者进行对接的两个重要问题，一方面能够通过同态加密保护数据所有者的隐私安全，不需要暴露数据即可获得希望的模型，同时也避免了用户隐私泄漏的潜在风险。另一方面能够保护模型所有者的模型隐私并保证计算结果的正确性，模型所有者不需要提供模型，通过 Cortex 2.0 的零知识 AI 模型计算结果以及证明，可以在保护隐私的同时完成正确性证明。

4.1.2 AI in DAO

现有的 DAO 系统仅采用内嵌的固定合约代码，在少量选择范围内通过投票等方式帮助组织进行治理，智能合约的执行能力极大限制了 DAO 的发展。Cortex 2.0 提供完备的链上 AI 模型，能使 DAO 在更多方面达成自动化和智能化管理，进一步实现自主决策。

例如，现有的 DAO 需要设计持有 token 或者 NFT 等机制来完成成员筛选，这种简单的机制可能导致女巫攻击等危险，而复杂的机制在现有区块链上会面临区块限制以及高昂的使用成本。Cortex 2.0 的 AI 模型可以高效解决这一问题，通过在区块链上收集 DAO 成员的所有历史交易以及各类数

据，可以快速学习到该组织希望接纳的成员。并且，随着成员在 DAO 中的各类操作做为反馈，AI DAO 可以进一步学习该组织的各类行为，随着数据的增加，AI DAO 可以在更多场景中替代人类完成公正的决策。

4.2 链上 AI 游戏

现有的区块链游戏受限于智能合约的功能和性能，仅仅能实现一些简单的游戏，Cortex 支持的 AI 功能赋予区块链游戏更丰富的设计以及更有趣的玩法。AI 功能可以使得游戏中的部分逻辑判断更加符合人类认知，因而增加了游戏的趣味性和易玩性。

例如，传统加密猫游戏在繁殖后代的时候采用简单的合约以及链上随机数完成基因组合，这一逻辑简单易实现，但是可玩性不足。Cortex 2.0 能通过 AI 模型增强链上游戏的可玩性，该基因的组合可以通过读取链上加密猫的历史交易数据，通过模型得到更加丰富的基因组合玩法。

4.3 链上 AI 金融服务

4.3.1 AI 聚合器

相比于普通的 DeFi 聚合器，Cortex 2.0 可以提供 AI-DeFi 聚合器，不仅仅对各个 DeFi 协议的收益率进行评估。还对用户更容易忽视的、各个 DeFi 理财平台背后隐藏的风险进行评估，利用 AI 推断能力对投资策略和投资组合作出综合决策。AI 聚合器从 DeFi 协议的链上原始数据出发，计算得到 centralization、liquidity、collateral 等一系列高阶特征，经过训练后的 AI 模型，最终得到每个投资项目的评分并得出最优投资组合。AI 聚合器将此投资组合的推断结果，跨链提供给以太坊的智能合约，并实现自动投资到不同的 DeFi 理财平台中去。完成对用户的一站式投资评估、咨询、操作的服务。

此外，投资者只需要在平台上回答一些投资调查问卷，Cortex 链上的 AI 模型便可以评估出投资者的风险偏好水平、在结合 DeFi 平台的 AI 评级结果，确定理财方案，自动生成最终的个性化投资配置组合。整个流程下来

所花的时间仅需几分钟，既做到真正的公开透明，又能达到高效、精准匹配用户资产管理目标。

4.3.2 去中心化 AI 信用评级

随着链上数据越来越丰富，对每个用户的资产、风险、信用等方面的评级成为可能，基于信用评级的借贷业务也随之成为可能。以 Cortex 链上 AI 模型为基础，对贷款账户的数据进行全面的评级，给出一个可参考的信用评级，借贷平台以此为依据开展信用借贷业务，并结合锁仓等风控手段。突破目前借贷市场还完全只停留在抵押借贷这一框架，实现无抵押借贷。抵押借贷的方式确实是一种相对有效的风控策略，但也正是这种单一粗暴的风控策略，形成了一些根本上的问题或者说给业务发展形成了比较明显的壁垒。突破壁垒，就将面对的是全新的蓝海市场，这无疑非常值得憧憬和期待。

4.3.3 AI 算法稳定币

算法稳定币 [25] 是区块链金融生态中重要的一环，目前的算法稳定币依靠预言机收集到的价格等简单数据对供给量、抵押率等数据进行调控。现有调控算法受限于智能合约的处理能力，仅能实现机械化的调控，这样的调控算法容易受到闪电贷等攻击。Cortex 2.0 中推出的 AI 算法稳定币能通过历史价格挖掘市场情绪等潜在数据，有前瞻性地调控算法稳定币中的重要参数。

5 路线图

- **Ritchie 2022 Q1** Solidity Compiler，MRT 版本更新。
- **Dijkstra 2022 Q2** 完成多节点公证人机制的跨链桥，兼容 EVM。发布 MRT 量化论文。
- **Bernoulli 2022 Q4** 完成转账交易和合约执行的 zk-Rollup，以及中继机制的跨链桥。
- **Galileo 2023 Q4** 完成通用分布式存储系统架构。

A 算子的数学形式化

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:

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

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)

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

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

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

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. $C = IC \cdot$ groups $\wedge\, OC$ 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:

$$\text{kernel}(n, j, p, q, o, i) = \sum_{k_i=0}^{\text{KH}} \sum_{k_j=0}^{\text{KW}} \text{pad}(p' + k_i * \text{DH}, q' + k_j * \text{DW}) \cdot W[o, i, k_i, k_j],$$

where $p' = p \cdot \text{SH} - \text{PH}$ and $q' = q \cdot \text{SW} - \text{PW}$ 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)
  - $ceil\_mode$, $boolean$.

$$PSH \in [0, H + 2PH + 1),$$
$$PSW \in [0, W + 2PW + 1),$$
$$PSH > PH \wedge PSW > PW$$

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

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

$$Y[d_0, d_1, \cdots, d_{N-1}] = \begin{cases} x, & x \geqslant 0 \\ -x, & x < 0 \end{cases},$$

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

where x 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

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

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

where x 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

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

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

where x 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 $(m_0, m_1, \cdots, m_{M-1})$.

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

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

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.

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

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 $[0, N)$.

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

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

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

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

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

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** :

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

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.

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

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

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

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

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**.

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

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

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

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**.

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

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

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

B CVM 实现的主流模型及性能测试

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

参考文献

[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.