Cortex - 链上人工智能

陈子祺
zchen@cortexlab.ai

王威扬
weiyang.wang@cortexlab.ai

田甲
mickey@cortexlab.ai

严枭
xiao.yan@cortexlab.ai

Abstract

当今的区块链世界里，内置了图灵完备的智能合约的链得到了广泛应用，吸引了大批应用开发者。然而由于过于理想化的世界计算机 (World Computer) 模型带来的昂贵成本，智能合约在设计时限制了自身的能力，并不能完全发挥图灵完备的巨大计算可能性，而仅限于编写短小的程序和访问极少量的资源。虽然一些极有可行性的应用与技术已经可以引入智能合约🄴，通过智能合约的扩容技术可以带来执行的性能提升，在合理的优化和硬件支持之下可以得到应用。本文描述了一条新的公有链 Cortex，通过修改和扩展指令集的方式，为智能合约增加人工智能算法的支持，从而使得任何人都可以为他们的智能合约增加人工智能模型。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，模型的贡献者亦可以得到奖励。正如其他专业领域正在发生进展的事情一样，拜 Cortex 的开放和共享特性所赐，Cortex 将产生许多超越人类能力极限的模型。作为一场社会实验，我们同时也非常期待 Cortex 链上能够诞生通用人工智能 AGI(Artificial General Intelligence)。

术语表

AI DApps	Decentralized Artificial Intelligence Applications 去中心化人工智能应用
ASGD	Asynchronous Stochastic Gradient Descent 异步随机梯度下降法
BFT	Byzantine Fault Tolerant 拜占庭容错
CNN	Convolutional Neural Network 卷积神经网络
CVM	Cortex Virtual Machine, Cortex 定义的虚拟机
ERC 20	一个基于以太坊的令牌标准协议
EVM	Ethereum Virtual Machine, 以太坊虚拟机
FHE	Fully Homomorphic Encryption 完全同态加密
GAN	Generative Adversarial Network 生成对抗网络
HE	Homomorphic Encryption 同态加密
Kaggle	一个机器学习领域里中心化的数据建模和分析平台
PoS	Proof of Stake 权益证明
PoW	Proof of Work 工作量证明
RL	Reinforcement Learning 增强学习
RNN	Recurrent Neural Network 递归神经网络
VAE	Variational Autoencoder 变分自编码机
zk-SNARKs	Zero-knowledge Succinct Non-Interactive Argument of Knowledge 非交互式简洁零知识证明
zk-STARKs	Zero-knowledge Succinct Transparent ARguments of Knowledge 透明可信简洁零知识证明
ZSL	Zero-knowledge Security Layer 零知识安全协议层

目录

1 介绍

"*Know Thyself.*" – 古希腊，阿波罗神谕

"*Two possibilities exist: either we are alone in the Universe or we are not. Both are equally terrifying.*" – 亚瑟 · 查尔斯 · 克拉克爵士

历经亿万年的进化，人类的智慧在巅峰闪耀，照亮着迷雾中的未来，与之相伴的孤独却愈加强烈。对未知的恐惧，对自身的迷惑和对实在的追求，使得人类作为一个整体，感受着前所未有的孤独。因此人类研究生命，倾注全力去制造新的自动机器，希望超越进化的速度，加速驶向未来。

2009 年 1 月 3 日，比特币 [1] 作为一种自持的 P2P 系统启动了创世区块，以巧妙的设计驱使参与者维持它的运转，并提供受限但极具颠覆性的金融功能至今。2015 年 6 月 30 日，以太坊 [2] 上线，为区块链增加了图灵完备的智能合约，可以对一些短小的程序的执行结果形成共识。相对于比特币，以太坊可以执行更复杂的计算，提供更丰富的响应，然而这些合约是不具备学习能力和自我进化规则的，是纯粹的基于简单规则 (rule-based) 与递归调用的子程序的集合。参考 Conway 的生命游戏 [3]，基于 P2P 技术的虚拟货币网络可以被界定为生存在互联网上的生命，通过提供金融功能维持自身的存在，只要还有一个全节点在，网络的状态就可以得到保存，并且能够响应来自外界的交互。然而人类渴望的智能还没有出现，这些原始的网络生命只停留在简单规则的水平。

Cortex 在此基础上更进一步，为区块链增加了人工智能的共识推断，所有全节点共同运作，对一个要求人工智能的智能合约的执行达成共识，为系统赋予智能响应的能力。Cortex 作为一条兼容 EVM 智能合约的独立公链，可以运行现有的合约和带有人工智能推断的合约，在创世区块发布后，也将作为一个更加智能的网络生命永续存在下去。在 Cortex 中，由于开源和天然的竞争机制，最优秀的模型终将会存留下来，提升网络的智能水平。从机器学习研究者的角度来讲，Cortex 平台集合了各种基本智能应用的公开模型，并且是当前的世界级水准 (state of the art)，这将大大加速他们的研究，并朝向 AI in All 的智能世界快速前进。这条公链同时使得模型在部署后的计算结果自动地得到全网公证。外星人存在与否我们尚不可知，但有人工智能的陪伴，人类不再孤独前行。

1.1 背景概述

现有的区块链合约，只能执行简单的智能合约计算，无法满足真实世界 AI 的应用。

区块链解决的是去中心化价值网络的传输和记账问题。比特币采用了工作量证明（Proof of Work，PoW），每一个区块分为三部分，分别是：

1. 上一个区块的哈希值作为本区块的 Block Header；
2. 在一段时间 T 内产生的交易 $(t_1, t_2, \cdots, t_n)$，将被打包哈希 (hashed) 进入 coinbase；

3. 连同挖矿者的地址（一般来讲是矿池地址）一同作为哈希函数的输入值 X，由矿池派发任务给矿机去寻找 $H(X, \text{nonce}) <$ 目标难度数值 (Target Difficulty)，其中 nonce 是后面拼接的整数随机猜测。计算结果将被全网节点验证从而出块获得奖励，进入下一轮出块计算，从而形成一条链。此外，还有如版本号、默克尔树 (Merkle tree) 、时间戳 (timestamp) 等信息。

整个挖矿过程概括为：

SHA256(SHA256(version+prev_hash+merkle_root+ntime+nbits+nonce)) < TARGET

以太坊利用了叔块 (uncle block) [4] 来提高并发性。应特别注意的是，以太坊网络和 Rootstock[5] 网络，都是针对链上智能合约来设计的。区块链的共识和不可篡改性保证了合约的强制执行，合约执行后，资金将自动划拨，从而去掉了对人或其他第三方的信任和依赖。

近年来人工智能的兴起，主要得益于计算能力的大幅度提升。针对人工智能领域的机器学习问题，可以泛化为如下形式：针对某个问题 Q 以及这个问题的输入数据集 D，给出度量（metric) P，从中获得模型 M，使得模型 M 在这个问题中的度量评价提高。在这种形式下，所有的机器学习问题都可以归结于以下几个元素：输入 (input)，这里用 D 表示，输出 (output)，这里用 Y 表示，度量 (metric)，这里用 P 表示，机器学习所需要解决的问题就是：

极大或者极小化：$\mathrm{P}(Y, \mathrm{M}(D))$。附录 A和附录 B针对具体问题进行了细节表述和数学表述。

优化目标函数的过程中，往往运用各种数值方法进行迭代，进行梯度下降，从而找到全局最优解 (Global Optima)，大规模的分布式学习往往采用异步随机梯度下降法 (ASGD) 来优化结果。有时由于问题不同，训练只能依某一分布获得一个离全局最优一定距离的次优解。

由于机器学习训练过程天生的中心化倾向，一个中心化的大集群比起松散连接的去中心化集群有无可比拟的优势，Cortex 将致力于搭建链下训练的集群，优化链上模型。

1.2 AI on Blockchain-可行性分析

目前有多个 AI on Blockchain 的项目，基本都是概念化阐释，没有具体实践方案，方向概括为：

1. 把分布式挖矿与强化学习结合。
2. 搭建数据交换和发布机器学习任务的平台。
3. 帮助所有链提供调用级别的机器学习接口。
4. 在训练过程中利用同态加密对用户的数据隐私进行保密。

目前区块链比较关心的研究领域在于私密性的保证，同态加密方案可以保护用户数据或者模型在云端计算的时候不被他人盗取。所谓同态加密是指一种加密手段，使得对明文进行运算符操作的结果等价于对密文进行运算符操作的结果（相当于加密运算在运算符 $\circ$ 下不变）：

$$E(x) \circ E(y) = E(x \circ y)$$

对于完全同态加密（FHE）是指针对任何运算符 $\circ$，都满足不变性。目前虽然有理论论文证明了完全同态加密的可行性，但是由于运算量过大，不具有工业化可行性。此外，有一种称为 Somewhat 同态加密 (SWHE)[6] 的加密方式，这种同态加密只支持一些特定的函数，比如多项式、代数乘法、加法等。对于机器学习问题，当前有三种情况：

1. 数据加密，模型参数不加密（在这种情况下，用户更加关心数据的保密性，比如病人的 CT 数据）。

2. 数据不加密，模型参数加密（在这种情况下，用户只是想训练或者测试自己的模型，并不关心数据如何）。

3. 数据和模型参数都加密。

针对第三种情况，如果双方的加密算法或者私钥不同，数据无法进行训练。比如 A 提供了一个模型，B 提供了数据，这个时候训练过程就无法进行。

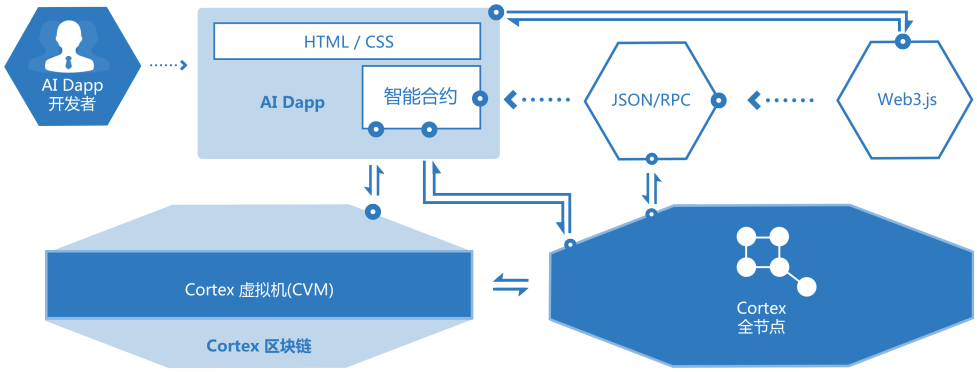
对于数据或者模型参数加密的情况，无法知道损失函数的具体数值，因为最后的残差计算出来也是被加密的。这个时候无法对模型进行评估，无法做交叉验证避免过拟合，无法进行超参调整（比如至关重要的学习率）。所有超参调整必须通过发布者解密后进行确认。

针对前两个情况，现阶段的同态加密算法使用 FHE，由于运算量过高，暂时不予考虑，只能使用计算复杂度较小的 SWHE 方法。经过测试，目前 state-of-the-art 的技术运算量依然非常高，相比于传统计算方法，如果使用同态加密对模型加密后进行训练和推断，运算量增加了 2-3 个数量级，这是无法承受的。

针对去中心化并行训练，由于目前无论是网络带宽，同步参数难度以及训练进度共识都有一定的困难，Cortex 针对这个问题已经引进了全球顶级分布式机器学习专家进行研究。

Cortex 的主要任务是在链上提供一批最好的机器学习模型，并且用户可以使用 Cortex 链上的智能合约进行推断。Cortex 的目标同时包括实现一个机器学习平台，用户可以在平台上发布任务，提交模型，利用智能合约调用模型进行推断，创建自己的 AI DApps (Artificial Intelligence Decentralized Applications) 等。

图 1: AI DApp



1.3 AI DApps

Cortex 给智能合约带来了人工智能能力，从而使得以下应用成为可能：

信息服务：个性化推荐系统，搜索引擎，新闻撰写服务等。

金融：征信，智能投顾。

人工智能助手：自动问答、行业知识图谱、语音合成、人脸属性预测。

模拟环境：自动驾驶、围棋等强化学习相关的应用。

2 系统架构

2.1 扩充智能合约和区块链的功能

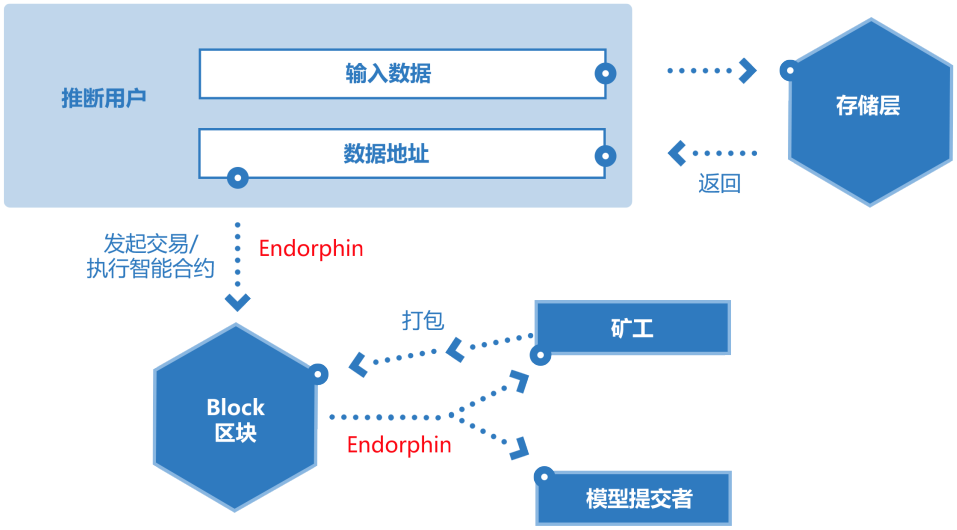
2.1.1 Cortex 智能推断框架

模型的贡献者将不限于 Cortex 团队，全球的机器学习从业人员都可以将训练好的相应数据模型上传到存储层，其他需要该数据模型的用户可以在其训练好的模型上进行推断，并且支付费用给模型上传者。每次推断的时候，全节点会从存储层将模型和数据同步到本地。通过 Cortex 特有的虚拟机 CVM (Cortex Virtual Machine) 进行一次推断，将结果同步到全节点，并返回结果。

将需要预测的数据进行代入计算到已知一个数据模型获得结果就是一次智能推断的过程。用户每发起一笔交易，执行带有数据模型的智能合约和进行推断都需要支付一定的 Endorphin，每次支付的 Endorphin 数量取决于模型运算难度和模型排名等。Endorphin 和 Cortex Coin 会有一个动态的转换关系，Endorphin 的价格由市场决定，反映了 Cortex 进行模型推断和执行智能合约的成本。这部分 Endorphin 对应的 Cortex Coin 会分成两个部分，一部分支付给智能合约调用 Infer 的模型提交者，另一部分支付给矿工作为打包区块的费用。对于支付给模型提交者的比例，Cortex 会为这个比例设定一个上限。

Cortex 在原有的智能合约中额外添加一个 Infer 指令，使得在智能合约中可以支持使用 Cortex 链上的模型。

图 2: 推断流程



下述伪代码表述了如何在智能合约里使用 Infer ，当用户调用智能合约的时候就会对这个模型进行一次推断：

Inference 代码

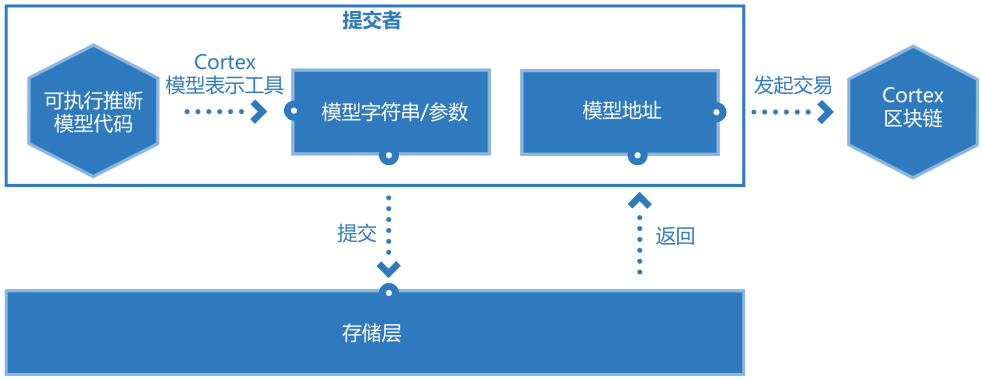
```
contract MyAIContract{
  InferType res;
  ...
  function myAIFunction(){
    ...
    res=infer(model_hash, data_hash);
    ...
  }
  ...
}
contract InferType{
  ...
}
```

2.1.2 模型提交框架

前面分析了链上训练的难处和不可行性，Cortex 提出了链下 (Offchain) 进行训练的提交接口，包括模型的指令解析虚拟机。这能够给算力提供方和模型提交者搭建交易和合作的桥梁。

用户将模型通过 Cortex 的 CVM 解析成模型字符串以及参数，打包上传到存储层，并发布通用接口，让智能合约编写用户进行调用。模型提交者需要支付一定的存储费用得以保证模型能在存储层持续保存。对智能合约中调用过此模型进行 Infer 所收取的费用会有一部分交付给模型提交者。提交者也可以根据需要进行撤回和更新等操作。对于撤回的情况，为了保证调用此模型的智能合约可以正常运作，Cortex 会根据模型的使用情况进行托管，并且保持调用此模型收取的费用和存储维护费用相当。Cortex 同时会提供一个接口将模型上传到存储层并获得模型哈希。之后提交者发起一笔交易，执行智能合约将模型哈希写入存储中。这样所有用户就可以知道这个模型的输入输出状态。

图 3: 模型提交流程



2.1.3 智能 AI 合约

Cortex 允许用户在 Cortex 链上进行和机器学习相关的编程，并且提交一些依赖其他合约的交互，这将变得十分有趣。比如以太坊上运行的电子宠物 Cryptokitties ，宠物之间的交互可以是动态的、智能的、进化的。通过用户上传的增强学习模型，赋予智能合约结合人工智能，可以很方便的实现类似带有人工智能的各种应用。

同时 Cortex 为其他链提供 AI 调用接口。比如在比特币现金和以太坊上，Cortex 提供基于人工智能的合约钱包地址上分析的调用结果。那些分析地址的模型将将不仅有助于监管科技 (RegTech)，也能给一般用户提供转账目标地址的动态风险评估。

2.2 模型和数据存储

Cortex 链并不实际存储模型和数据，只存储模型和数据的 Hash 值，真正的模型和数据存储在链外的 key-value 存储系统中。新模型和新数据在节点上有足够多的副本之后将可以在链上可用。

2.3 Cortex 共识推断标准

当用户发起一笔交易到某个合约之后，全节点需要执行该智能合约的代码。Cortex 和普通智能合约不同的地方在于其智能合约中可能涉及推断指令，需要全节点对于这个推断指令的结果进行共识。整个全节点的执行流程是：

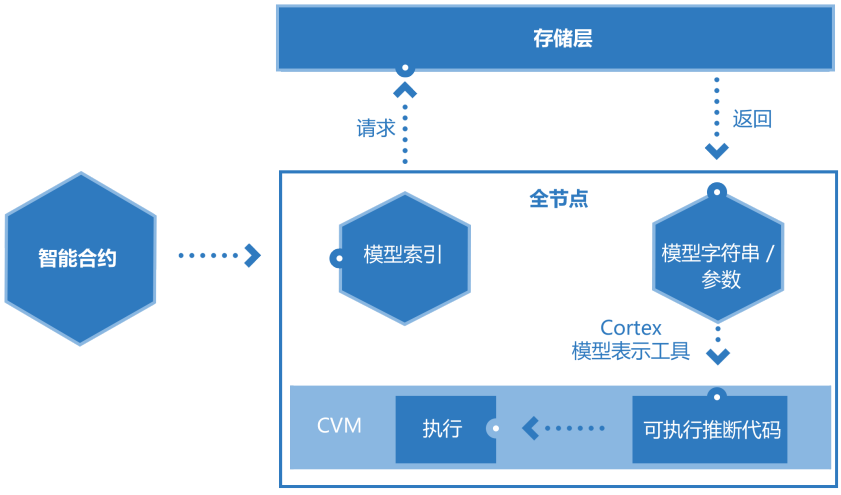
1. 全节点通过查询模型索引找到模型在存储层的位置，并下载该模型的模型字符串和对应的参数数据。
2. 通过 Cortex 模型表示工具将模型字符串转换成可执行代码。
3. 通过 Cortex 提供的虚拟机 CVM ，执行可执行代码，得到结果后进行全节点广播共识。

Cortex 模型表示工具的作用可以分为两部分：

1. 模型提交者需要将自己编写的模型代码通过模型表示工具转化为模型字符串之后才可以提交到存储层。
2. 全节点下载模型字符串之后通过模型表示工具提供的转换器转换成可执行代码后，在 Cortex 虚拟机中执行推断操作。

Cortex 虚拟机的作用在于全节点的每次推断执行都是确定的，章节 3.2和章节 3.3描述了 Cortex 模型表示工具和 Cortex 虚拟机的实现细节。

图 4: 全节点执行推断流程



2.4 如何挑选优秀的模型

Cortex 链不会对模型进行限制，用户可以依靠模型 infer 的调用次数作为相对客观的模型评价标准。当模型使用者对模型有不计计算代价的高精度需求时，Cortex 支持保留

原有模型参数使用浮点数来表示。从而，官方或者第三方机构可以通过自行定义对模型的排序机制（召回率，准确率，计算速度，基准排序数据集等），达成模型的甄选工作，并展示在第三方的网站或者应用中。

2.5 共识机制：PoW 挖矿

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 [7] 和 Zcash 的 Equihash[8] 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，采用 Cuckoo Cycle 的 PoW 进一步缩小 CPU 和显卡之间加速比的差距。同时 Cortex 链将充分发掘智能手机显卡的效能，使得手机和桌面电脑的显卡差距符合通用硬件平台测评工具（如 GFXBench ）的差距比例：比如，最好的消费级别显卡是最好的手机显卡算力的 10-15 倍。考虑到手机计算的功耗比更低，使得大规模用户在夜间充电时间利用手机挖矿将变得更加可行。

特别值得注意的一点是，出块加密的共识算法和链上的智能推断合约的计算并没有直接联系，PoW 保障参与挖矿的矿工们硬件上更加公平，而智能计算合约则自动提供公众推理的可验证性。

2.6 防作弊以及模型筛选

由于模型是完全公开的，所以可能会有模型被复制或抄袭等现象发生。在一般情况下，如果是一个非常优秀的模型，往往上线之后就会有很高的使用量，而针对这些模型进行抄袭并没有很大优势，但是，在一些特殊情况下，对一些很明显的抄袭或者完全复制的行为，Cortex 会进行介入并且仲裁，并通过链上 Oracle 的方式公示。

3 软件方案

3.1 CVM：EVM + Inference

Cortex 拥有自己的虚拟机，称为 Cortex Virtual Machine（CVM)。CVM 指令集完全兼容 EVM，此外，CVM 还提供对于推断指令的支持。Cortex 将在 0xc0 加入一个新的 INFER 指令。这条指令的输入是推断代码，输出是推断结果。CVM 使用的虚拟机指令包含的内容在表 1中说明。

3.2 Cortex 核心指令集与框架标准

人工智能的典型应用——图像问题，语音/语义/文本问题，与强化学习问题无一例外的需要以下张量操作。Cortex 以张量操作的代价作为 Endorphin 计费的一种潜在锚定手段，剖析机器学习以及深度学习的核心指令集。在不同计算框架中，这一术语往往被称为网络层 (network layer) 或者操作符 (operator)。

表 1: CVM 指令

数值	指令	描述
0s	Stop and Arithmetic Operations	包含函数和代数运算
10s	Comparison & Bitwise Logic Operations	比较和比特操作
20s	SHA3	SHA3
30s	Environmental Information	环境信息
40s	Block Information	区块信息
50s	Stack, Memory, Storage and Flow Operations	栈，内存，存储，流操作等
60s & 70s	Push Operations	压栈操作
80s	Duplication Operations	复制操作
90s	Exchange Operations	交换操作
a0s	Logging Operations	日志操作
c0s	Cortex Operations	Cortex 相关操作
0xc0	INFER	Inference
f0s	System operations	系统操作

- 张量的计算操作，包括：
  - 张量的数值四则运算：输入张量，数值与四则运算符
  - 张量之间的按位四则运算：输入两个张量与四则运算符
  - 张量的按位函数运算：输入张量与乘方函数、三角函数、幂与对数函数、大小判断函数、随机数生成函数、取整函数等。
  - 张量的降维运算：输入张量与满足结合律、交换律的操作符。
  - 张量之间的广播运算：输入张量，用维度低张量补齐维度后进行按位操作。
  - 张量之间的乘法操作：以 NCHW/NHWC 张量存储模式为例，包含张量与矩阵、矩阵与向量等张量乘法/矩阵乘法操作。
- 张量的重构操作，包括:
  - 维度交换，维度扩张与维度压缩
  - 按维度排序
  - 值补充
  - 按通道拼接
  - 沿图像平面拼接/剪裁
- 神经网络特定操作
  - 全连接
  - 神经网络激发函数主要依赖张量的按位函数运算的操作
  - 1 维/2 维/3 维卷积（包括不同尺度卷积核、带孔、分组等选项）
  - 通过上采样实现的 1 维/2 维/3 维反卷积操作与线性插值操作
  - 通用辅助运算（如对一阶/二阶信息的统计 BatchNorm）
  - 图像类辅助计算（如可形变卷积网络的形变参数模块）

– 特定任务辅助计算（如 ROIPooling, ROIAlign 模块）

Cortex 的核心指令集已覆盖主流的人工智能计算框架操作。受制于不同平台上 BLAS 的实现，Cortex 把拥有浮点数 (Float32, Float16) 参数的 Cortex 模型通过 DevKit 转化为定点数（INT8, INT6）参数模型 (Wu et al. [9]Han et al. [10])，从而支持跨平台的推断共识。

3.3 Cortex 模型表示工具

Cortex 模型表示工具创建了一个开放，灵活的标准，使深度学习框架和工具能够互操作。它使用户能够在框架之间迁移深度学习模型，使其更容易投入生产。Cortex 模型表示工具作为一个开放式生态系统，使人工智能更容易获得，对不同的使用者都有价值：人工智能开发人员可以根据不同任务选择正确的框架，框架开发人员可以专注于创新与更新，硬件供应商可以针对性的优化。例如，人工智能开发人员可以使用 PyTorch 等框架训练复杂的计算机视觉模型，并使用 CNTK 、Apache MXNet 或者 TensorFlow 进行推断。

模型表示的基础是关于人工智能计算的 Cortex 核心指令集的规范化。随着人工智能领域研究成果、软件框架、指令集、硬件驱动、硬件形式的日益丰富，工具链碎片化问题逐渐突显。很多新的论文站在前人的工作基础上进行微创新；理论过硬的科研成果得到的模型、数据、结论并不是站在过去最佳成果之上进行进一步发展，为精度的提高带来天花板效应；工程师为了解决特定问题而设计的硬编码更加无法适应爆发式增长的数据。

Cortex 模型表示工具被设计为

- 表征：将字符串映射为主流神经网络模型、概率图模型所支持的最细粒度的指令集
- 组织：将指令集映射为主流神经网络框架的代码
- 迁移：提供同构检测工具，使得不同机器学习/神经网络框架中相同模型可以互相迁移

3.4 存储层

Cortex 可以使用任何 key-value 存储系统来存储模型，可行的选择是 IPFS 和 libtorrent 。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，对于不同设备，Cortex 采取不同策略：

- 全节点常年存储公链数据模型
- 手机节点采取类似比特币轻钱包模式，只存储小规模的全模型

Cortex 只负责共识推断，不存储任何训练集。为了帮助合约作者筛选模型，避免过拟合的数据模型难题，合约作者可以提交测试集到 Cortex 披露模型结果。

一条进入合约级别的调用，会在内存池 (Mempool) 中排队，出块后，将打包进入区块确认交易。缓存期间数据会广播到包括矿池的全节点。Cortex 当前的存储能力，能够支持目前图片、语音、文字、短视频等绝大部分典型应用，足以覆盖绝大多数人工智能问题。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，一条就可能几十个 GB ，将在未来 Cortex 提升存储限制后加入支持。

对于 Cortex 的全节点，需要比现有比特币和以太坊更大的存储空间来存储缓存的数据测试集和数据模型。考虑到摩尔定律 (Moore's Law)，存储设备价格将不断下降，因此不会构成障碍。对于每个数据模型，Metadata 内将建立标注信息，用来进行链上调用的检索。Metadata 的格式在表 2中表述。

表 2: Metadata

	关键词	示例
Model:		
	MD5	8ac7b335978cf2fe8102c5c43e380ca1
	Field	Speech, Image, etc.
	Method	NasNet Large
	Is Training	False if model is deployed else True
	Loss Function	Softmax, Hinge loss, KL divergence
	Rank	1
	Model Size	$10^4 - 10^{10}$ Byte
Input:		
	Dataset	ImageNet22k + Place365
	Dimension	Dim of Input, e.g. 2 or 3 for Image
	Size	e.g. $1024 \times 1920 \times 3$
	Dtype	Float32
Output:		
	Range	[0,1],[0,255]
	Predict	Top 5 Predictions
	Dimension	Dim of output
	Size	e.g. 1

3.5 模型索引

Cortex 存储了所有的模型，在全节点中，对于每笔需要验证的交易，如果智能合约涉及共识推断，则需要从内存快速检索出对应的模型进行推断。Cortex 的全节点内存将为本地存储的模型建立索引，根据智能合约存储的模型地址去检索。

3.6 模型缓存

Cortex 的全节点存储能力有限，无法存下全网所有模型。Cortex 引入了缓存来解决这个问题，在全节点中维护一个 Model Cache 。Model Cache 数据模型的替换策略，有最近最常使用（LRU）、先进先出（FIFO）等，也可以使用任何其他方案来提高命中率。

3.7 全节点实验

针对全节点执行推断指令的吞吐情况，本章描述了一些在单机上实验的结果。测试平台配置为：

- CPU: E5-2683 v3
- GPU: 8x1080Ti
- 内存: 64 GB
- 硬盘: SSD 960 EVO 250 GB

实验中使用的测试代码基于 python 2.7 和 MXNet ，其中主要包含以下模型：

- CaffeNet
- Network in Network
- SqueezeNet
- VGG16
- VGG19
- Inception v3 / BatchNorm
- ResNet-152
- ResNet101-64x4d

所有模型都可以在 MXNet 的文档 [1] 中找到。实验分别在 CPU 和 GPU 中测试这些模型在平台上的推断速度，这些测试不考虑读取模型的速度，所有模型会提前加载到内存或者显存中。

测试结果如表 3，括号中是 Batch Size（即一次计算所传入的数据样本量），所有 GPU 测试结果都是在单卡上的测试。

[1] http://mxnet.incubator.apache.org/model_zoo/index.html

图 5: 模型大小

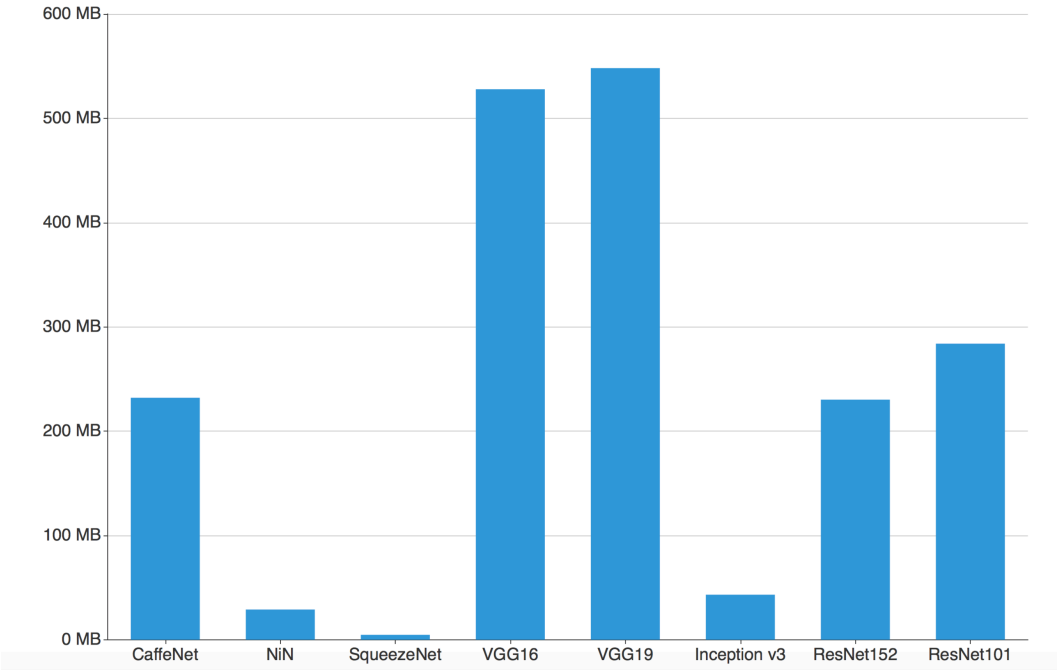


表 3: 推断速度表

模型	大小	CPU 推断 (1)	GPU 推断 (1)	GPU 推断 (64)
CaffeNet	232 MB	196 ms	2.23 ms	39.98 ms
Network in Network	28.97 MB	115 ms	2.12 ms	42.90 ms
SqueezeNet v1.1	4.72 MB	130 ms	2.16 ms	46.18 ms
VGG16	527.79 MB	657 ms	5.84 ms	177.95 ms
VGG19	548.05 MB	681 ms	6.70 ms	205.26 ms
Inception v3/BN	43.19 MB	1084 ms	8.53 ms	80.61 ms
ResNet-152	230.18 MB	4050 ms	23.93 ms	253.08 ms
ResNet101-64x4d	283.86 MB	2650 ms	14.19 ms	182.73 ms

以上是单机测试的结果。为了模拟真实的情况，试验平台上设置 10 万张的图片不断进行推断，每次推断选择随机的模型来进行并且 Batch Siz e 为 1，图片发放到 8 张带有负载均衡的显卡上。对于两种情况：

1. 所有模型都已经读取完毕并存放到显存中，其单张图片推断的平均速度为 3.16 ms。

2. 每次重新读取数据（包括模型和输入数据）而不是提前加载进显存，但是进行缓存，其单张图片推断的平均速度为 113.3 ms。

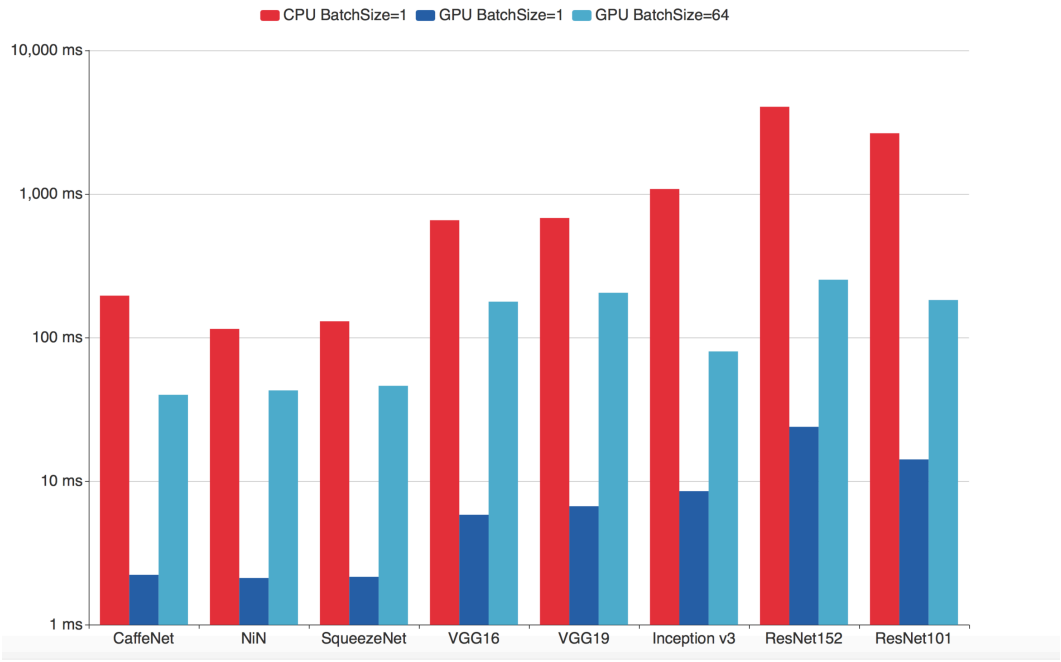


图 6: 不同模型推断时间对比（对数坐标）

结论 全节点在模型已经预读到显存之后，支持负载均衡，并且将同一模型进行显卡间并行推断，测试结果大约每秒能执行接近 300 次的单一推断。如果在极端情况下不进行显存预读，而只是进行缓存，每次重新读取模型和输入数据，大约每秒能进行 9 次左右的单一推断。以上实验都是在没有优化的情况下进行的计算，Cortex 的目标之一是致力于不断优化提高推断性能。

4 硬件方案

4.1 CUDA and RoCM 方案

Cortex 的硬件方案大量采用了 NVidia 公司的 CUDA 驱动与 CUDNN 库作为显卡计算的开发框架。同时，AMD OpenMI 软件项目采用了 RoCM 驱动与 HIP/HCC 库人工智能研发计划，并计划在 2018 年底推出后支持的开发框架。

4.2 FPGA 方案

FPGA 产品的特性是低位定点运算 (INT8 甚至 INT6 运算)，延时较低，但是计算功耗较高，灵活性较差；在自动驾驶领域、云服务领域已经有较好的深度学习部署方案。Cortex 计划对 Xilinx 与 Altera 系列产品提供 Infer 支持。

4.3 全节点的硬件配置需求 - 多显卡和回归传统的 USB 挖矿

"*Make mining great again!*" – 让挖矿再次伟大！

图 7: 比特币 USB 矿机和神经网络计算芯片

不同于传统的比特币和以太坊全节点，Cortex 对全节点的硬件配置需求较高。需要比较大的硬盘存储空间和多显卡桌面主机来达到最佳确认速度的性能，然而这并不是必需的。在比特币领域 USB 曾经是一种即插即用的比特币小型 ASIC 矿机，在大规模矿厂形成之前，这种去中心化的挖矿模式，曾经风靡一时，Cortex 全节点在缺少显卡算力的情况下可以配置类似的神经网络计算芯片或计算棒，这些设备已经在市场上逐渐成熟。与 USB 挖矿不同的是，计算芯片是做全节点验证的硬件补足，并非计算挖矿具体过程中需要的设备。

4.4 现有显卡矿厂需要的硬件改装措施

对于一个现有显卡算力的矿厂，特别是有高端显卡的矿厂，Cortex 提供改造咨询服务和整体技术解决方案，使得矿厂具有和世界一流 AI 公司同等水平的智能计算中心，硬件性价比将远远超过现有商用 GPU 云，多中心化的矿厂有机会出售算力给算法提供者，或者以合作的方式生成数据模型，和世界一流的互联网、AI 公司同场竞技。

具体的改造策略有：

- 主板和 CPU 的定制策略，满足多路 PCI-E 深度学习的数据传输带宽
- 万兆交换机和网卡的硬件解决方案
- 存储硬件和带宽解决方案
- 相关软件在挖 Cortex 主链、挖其他竞争显卡币和链下深度学习训练之间自动切换
- 相关的手机端监控收益、手动切换等管理软件

图 8: Facebook AI Research 的 Big Basin Wedge 100-32X 交换机可用于进行分布式训练的联盟链矿厂

4.5 手机设备和物联网设备挖矿和计算

平衡异构计算芯片 (CPU)、显示芯片 (GPU)、FPGA 与 ASIC 计算模块的算力收益比例，从而更加去中心的进行工作量证明挖矿，一直是主链设计的难点，特别是能够让算力相对弱小的设备，比如手机和 IoT 设备参与其中。同时，由于目前市场上的手机设备已经出现了支持 AI 计算的芯片或者计算库、基于手机 AI 芯片的计算框架也可以参与智能推断，只不过全节点的数据模型相对较大，移动端需要定制对可执行数据模型的规模做筛选。Cortex 主链将发布 Android 和 iOS 客户端 App：

- 闲置中具有显卡算力的设备能通过 SoC 、ARM 架构的 CPU/GPU 参与挖矿，比如市场中，电视盒子的显卡性能其实已经很不错了，而 90% 时间基本都在闲置
- 用户手机在上班充电和睡觉充电中都可以参与挖矿，只要算法上让手机的显卡得到公平的收益竞争力
- 手机或其他配有 AI 芯片的设备，能够自动在主链出块和执行智能推断之间切换

手机端的推断能力可能会受到芯片供应商的软件技术限制，不同软件供应商正在封装不同的计算协议，Cortex 将负责抽象层接口的编写和轻智能客户端的筛选。

5 应用和未来工作

5.1 应用案例

Cortex 上可以建立以下 AI DApps：

5.1.1 信息服务

- 个性化推荐系统：根据 User Profile 和 Show/Click Log 的交易事务，给用户推荐符合他需要的新闻
- 图片搜索引擎：根据图片数据，检索相似图片
- 新闻撰写：根据文本语料，生成风格相似的文本
- 自动摘要：基于文本语料库自动摘要文本语料

5.1.2 金融服务

- 征信：根据用户的链上数据计算信用评级
- 智能投顾：根据金融数据输出交易决策

5.1.3 人工智能助手

- 自动问答：聊天机器人，根据用户对话生成回答
- 行业知识图谱：专家系统，可以应用在医疗、咨询等行业
- 语音合成：根据用户的语音数据生成另一段风格相似的语音
- 人脸辨识：根据用户上传的人脸数据对年龄、性别、情感等属性作出判断

5.1.4 模拟环境

强化学习的应用可以通过上传环境（Environment）得到模型预测输出，来实现自动驾驶和围棋、游戏等的智能决策和验证。

5.2 未来工作

Cortex 的目标是在链上机器学习领域不断保持世界领先。对于未来可能发明的新技术会不断进行创新和融合，本章描述了部分未来工作方向。

5.2.1 数据的隐私

Cortex 实现了链上的共识推断，然而对于机器学习应用的典型场景，如辅助医疗、生物信息识别、语音识别等，均需要严格的隐私保护机制，保护数据隐私和模型的知识产权。相关的技术方向有：差分隐私（Differential Privacy）[11]，零知识证明（Zero-Knowledge Proof）[12]，同态加密（Homomorphic Encryption）[13] 等。Cortex 将密切关注相关进展，当有新技术可以实用时会集成进来。

5.2.2 扩容

受 PoW 共识机制的限制，Cortex 也会遇到链上扩容的问题，目前可能的方向有：PoS、DAG、跨链通信等等。本质上来讲，由于分布式系统 CAP 定理 [14] 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 也将对扩容问题进行研究，在不牺牲核心假设的前提下提升网络的性能。

5.2.3 移动设备 AI 芯片的机器学习适配

由于最新的移动设备中往往配备了专用的 AI 芯片，Cortex 推断框架也可以调用移动设备的 AI 芯片计算接口，适配各家芯片的指令集，开发手机端的轻量级智能推理虚拟机。

6 路线图

2018 年 Q1	发行 ERC20 代币
2018 年 Q1	登上主流交易所
2018 年 Q3	挖矿测试链 Bernard 上线
2019 年 Q1	智能测试链 Dolores 上线
2019 年 Q2	主链 Arnold 创世区块发布，发行 Cortex Coin 并销毁 ERC20 代币，开始挖矿

7 代币模型

7.1 Cortex Coin (CTXC)

7.1.1 模型提交者的奖励收益

传统的区块链对于每个打包区块的奖励是直接支付给矿工的，Cortex 为了激励开发者提交更加丰富和优秀的模型，调用合约需要支付的 Endorphin 不仅仅会分配给帮助区块打包的节点矿工，还会支付给模型的提供者。费用的收取比例采用市场博弈价格，类似以太坊中 Gas 的机制。

7.1.2 模型提交者成本支出

为了防止模型提交者进行过度的提交和存储攻击 - 比如，随意提交几乎不可用的模型以及频繁提交相同模型从而占用存储资源 - 每个模型提交者必须支付存储费用。这样可以促使模型提交者提交更加优秀的模型。这样调用者更多，模型提交者收益更大。

7.1.3 模型复杂度和 Endorphin 的耗费

Endorphin 用来衡量在推断过程中将数据模型带入合约时，计算所耗费的虚拟机级别硬件计算资源，Endorphin 的耗费正比于模型大小，同时 Cortex 也为模型的参数大小设置了 8GB 的上限，对应最多约 20 亿个 Float32 的参数。

7.2 代币分配

Cortex Coin (CTXC) 数量总共为 299,792,458[2]个。其中 60,000,000 (20.01%) 分配给早期投资者。

Cortex Coin 总量	299,792,458	100%
挖矿数量	150,000,000	50.03%
投资者	60,000,000	20.01%
基金会	74,792,458	24.95%
实验室 Cortex Lab	45,000,000	15.01%
基金会锁定	27,000,000	9.01%
挑战赏金	2,792,458	0.93%
顾问、学术、社区开发者	15,000,000	5.00%

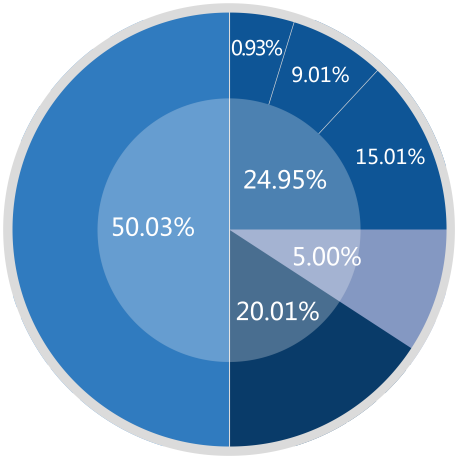
表 4: 代币分配细节 1

50.03%		Cortex Coin 挖矿者 (挖矿奖励)	矿工因提供算力贡献，维护 Cortex 区块链，并且运行 AI DApps 的全节点而获取奖励。
20.01%		投资者 (创世区块)	投资者的资金支持用来支持 Cortex 主链的开发，商业拓展，合作伙伴，和社区支持。
24.95%		基金会 (创世区块)	
	15.01%	实验室 Cortex Lab	负责实现 Cortex 主链协议的开发，数据模型的维护，软件升级，硬件解决方案。
	9.01%	基金会锁定	每季度加速解锁 1:2:3:4，用于奖励参与 Data Science / Kaggle 社区和实验室参与开发。
	0.93%	挑战赏金	用来奖励社区开发者提交有趣的项目，比如 BTC、XMR 地址分析挑战赛。
5.00%		顾问、学术、社区开发者 (创世区块)	人工智能的商业和学术机构，学者社区和开源框架开发者社区。

表 5: 代币分配细节 2

表 4和表 5描述了代币分配细节。

[2]光在真空中的速度为 $2.99792458 \times 10^8 m/s$



50.03% Cortex Coin 挖矿者 (挖矿奖励)

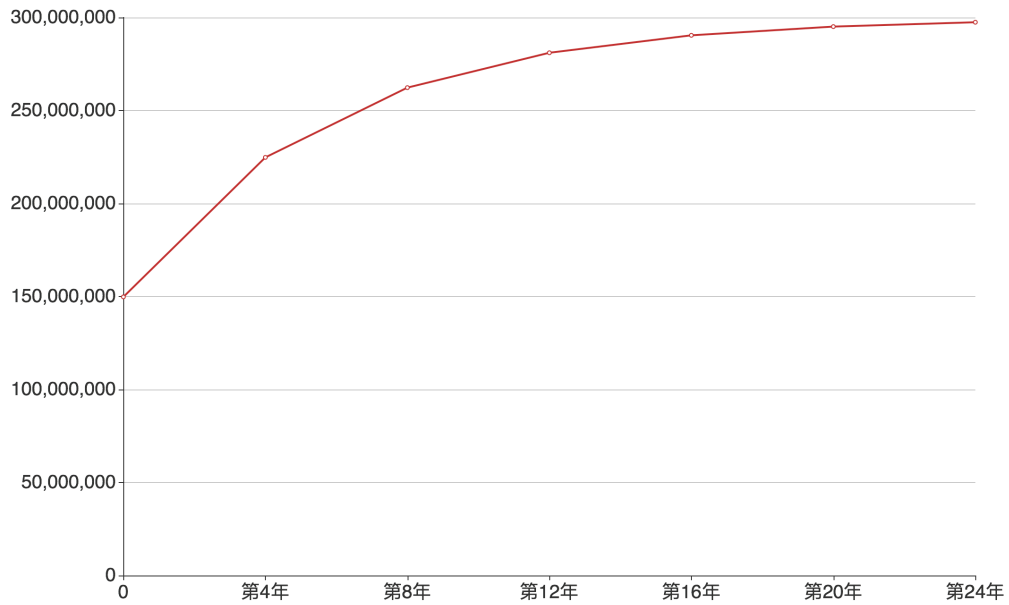
20.01% 投资者 (创世区块)

24.95% 基金会 (创世区块)

5.00% 顾问、学术、社区开发者 (创世区块)

7.3 代币发行曲线

Cortex Coin 发行总量为 299,792,458 个，其中 150,000,000 的 Cortex Coin 可以通过挖矿获得。



第一个 4 年　75,000,000
第二个 4 年　37,500,000
第三个 4 年　18,750,000
第四个 4 年　 9,375,000
第五个 4 年　 4,687,500
　　...

依此类推，发行量按每四年减半。

8 核心团队成员

陈子祺 创始人 *CEO* 清华大学土木工程系本科，后留美获卡耐基梅隆大学土木工程系硕士和加州大学圣克鲁斯分校计算机系硕士。在早期 AdaBoosting 和 Online Learning 的发源地，师从 David P. Helmbold 研究机器学习理论和各种算法应用，包括围棋算法。曾经在美国 SFTC 公司担任主研究科学家，负责有限元网格生成法，用于航天和武器研发。具有一线的电商创业经验和区块链全行业经验，Waterhole.io 北京遂视科技有限公司创始人，精通矿池、算力、钱包等业务，深度理解矿机、共识算法和公有链生态。在比特币、以太坊、Zcash 等加密货币提供算力。

王威扬 创始人 *CTO* 数学竞赛保送至清华大学，于航空航天学院工程力学系和经济管理学院获双学位，后留美于芝加哥大学获统计学硕士。师从逻辑回归创立者、1990 年考普斯总统奖获得者 Peter McCullagh 、随机森林创立者 Yali Amit 学习基于统计的机器学习和人工智能理论。利用深度学习领域多篇前沿论文进行 CNN/RNN 算法的研发，所在 DeepInsight 团队 4 个项目 OCR/迁移学习/人脸识别/分布式计算进入 Awesome MXNet 精品项目，Kaggle 竞赛获银牌。为 O' Reilly 出版社提供多篇人工智能报告的翻译。GARP 认证的金融风险分析师 (Certified FRM)。带领团队基于 IBM Hyperledger 区块链框架设计的资产证券化 (ABS) 系统获 2017 Dorahack 黑客马拉松冠军。精通数论、加密等相关领域，熟悉区块链技术架构。曾先后就职于京东金融，万达研究院。

严枭博士 人工智能专家 计算机竞赛保送至清华大学电子工程系，后直博就读于计算机科学与技术系，期间发表多篇顶会论文，如 ACM Siggraph 。精通 OI 竞赛编程、显卡 CUDA 计算研发、FPGA 硬件底层架构编程、密码学等，其研究领域包括有限元分析、张量分析、湍流、深度学习和增强学习算法。实际开发领域包括，机器学习、区块链、量化交易等。管理过规模 $10M 级别以上的私募基金。业余时间研究量子计算机，金融宏观分析等。对挖矿软件、分布式计算和共识算法有深入研究。曾在多家人工智能公司/私募基金担任过技术总监，量化总监，创始人或联合创始人。

杨阳 区块链工程总监 大连理工大学计算机系本科，清华大学计算机系硕士。精通分布式数据库、联盟链 Hyperledger/EEA 等，曾就职于英国电信，IBM 。曾开发过社交软件如:VisiDB/拨号精灵/蓝信等。10 年以上的 Java 和 C++ 后端构建经验。具有独立用 C/C++ 实现一个数据库的能力。区块链源代码和安全加密技术专家，具有多种加密数字货币开发经验。精通高并发长链接商业级软件-如 WhatsApp 、微信-等后台实现。精通各种智能合约和硬分叉开发。

9 顾问

9.1 技术顾问

田甲 首席科学家 物理和生物竞赛保送至清华大学计算机系，获本硕学位，分布式系统方向专家。曾在百度、阿里巴巴工作，日 PV 过亿的搜索引擎 (so.com) 和推荐引擎架构师。连续创业者，先后在多家创业公司任职，涉足搜索引擎、推荐引擎、人工智能、金融科技等方向。第一家公司卧龙云被阿里巴巴收购，后加入北京机器学习信息技术有

限公司担任 CTO ，研发 recsys 、chatbot 、医疗影像识别等系统，后又加入 Pony.ai 无人车创业公司，天使轮已得到红杉、IDG 投资。曾担任比特基金首席科学家，区块链研究者，在多家区块链技术公司任顾问，比特币、Zcash 早期投资人，世界最大比特币交易所 Bitfinex 投资人。对量子计算、核聚变和计算神经学有浓厚兴趣。

Matt Branton 技术顾问 Matt Branton 是一名分布式计算专家，在支付和交易系统设计和架构方面拥有十年以上的经验。他是 World Financial Desk 的创始合伙人，World Financial Desk 是一家专门从事固定收益和衍生产品的高频交易公司，负责数万亿美元的交易量。早期的比特币采用者，他创立了 Coinlock ，用于加密小额支付内容交付，并在过去五年中花费在 Ember Financial 建立智能合约和分布式账本系统。他拥有多项区块链专利，包括他在合成采矿和新型衍生产品方面的工作。

9.2 学术顾问

Whitfield Diffie 加密博士 密码学家和公钥密码学的先驱。迪菲于 1992 年获得瑞士联邦理工学院的荣誉博士学位。他还是马可尼基金会的会员，以及艾萨克牛顿研究所的访问学者。2008 年 7 月，他还获得了伦敦大学皇家霍洛威学院的科学博士学位（Honoris Causa）。他于 1981 年荣获 IEEE Donald G. Fink Prize Pape 奖，于 1997 年荣获 Franklin Institute 的 Louis E. Levy 奖章，于 1998 年荣获 IEEE 信息理论学会颁发的金禧奖技术创新奖以及于 2010 年荣获 IEEE Richard W. Hamming 奖章。迪菲于 2017 年被选为英国皇家学会的外籍成员（ForMemRS）.Diffie 与 Martin Hellman 一起赢得了 2015 年图灵奖，被公认为计算机科学领域最负盛名的奖项。

9.3 商业顾问

Vincent Zhou *FBG Capital* 创始人

Yahui Zhou *Beijing Kunlun Tech Co. (300418.SZ)* 首席执行官兼董事长

Heting Shen *China Metallurgical Group Corporation (601618.SH) and (01618.HK)* 前首席执行官兼董事长

Guy Corem *DAGlabs* 总裁

9.4 合作机构

学术界核心顾问团队:

清华大学

上海交通大学

加利福尼亚大学伯克利分校

斯坦福大学

10 投资机构

机构投资人：

Bitmain：比特大陆是一家世界顶级的提供加密数字货币和区块链软硬件解决方案的商业机构，同时也大力投入人工智能芯片领域的研发。

FBG Capital：全称 Fintech Blockchain Group Capital ，是亚洲和硅谷的一线区块链投资机构，曾早期投资 Dfinity 、OMG 、aelf 等众多区块链明星项目。

Nirvana Capital: 涅🅵资本是一家专注于区块链领域的投资基金。由姜普凡先生 (Lamento) 成立。他也是以太坊的早期投资人。

A Cortex 链上人工智能的数学表述

对传统人工智能定义和 Cortex 指令可以用数学表述进行说明

i: Cortex 指令

I: $\{i\}$, Cortex 指令集合

s: Cortex 指令参数, $s \in R^{N^d}$

S: Cortex 指令参数状态空间

r: $\in R$, Cortex 模型表达字符串

R: 有效 Cortex 模型表达字符串空间

m: $(i_1, i_2, ...i_k) \in M$, 有序 Cortex 指令集序列，Cortex 模型表达字符串 r 的完整描述

M: 有效的有序 Cortex 指令集序列空间

$\mathfrak{S}$: Cortex 模型参数空间 M^S

$\mathfrak{F}$: 支持的 AI 计算框架映射函数集合

$\mathfrak{f} \in \mathfrak{F}$: AI 计算框架映射函数 $\mathfrak{f} : f(m) \to c$

d: 数据样本

D: $\{d\}$, 数据集

$P : \{p : D \to (D_{\text{train}}, D_{\text{test}}, D_{\text{val}})\}$, 对数据集 D 划分的集合

$\mathfrak{M}$: 用于模型训练和模型排序服务的测度 $\mathfrak{M} : f(\mathfrak{I})(d) \to R$

$A : \{a\}$, 模型执行策略序列集合

一个定义完备的、在物理设备上可计算的机器学习/人工智能问题 $q \in Q$ 是一个四元组：

$Q : (\mathfrak{F}(M), P(D), \mathfrak{M}, A)$, 完备定义的问题集合

它具备特定框架下选择好的模型形式，以概率图模型或者神经网络图模型 $\mathfrak{f}(M)$

它具备了包含特定数据样本的一个划分 $p(D)$ 以进行训练、推断或者检查泛化性能

它具备了一个等待被优化，或者需要被排序的测度 $\mathfrak{m}$

它具备了机器学习的执行策略 a，以判断当前流程仅需要推断/共识还是需要训练。

B 深度学习原理和基本类型

对于章节 1.1中提到的机器学习框架，都可以表述出不同的机器学习问题，这里介绍几个经典的机器学习模式：

B.1 监督学习

所谓监督学习是指输出训练集数据（作为输入）给出，并且给出了对于训练集合对应的标签（也就是目标输出），然后计算机根据训练集和目标输出进行训练。比如数字图像识别问题，数据集 D 的形式可以是一些 64×64 的 8 位位图格式图片。标记 $Y\prime$ 是对应图像的类型，譬如图像有 10 种分类，0 对应的 $Y\prime$ 是 $[1,0,0,0,0,0,0,0,0,0]^T$ 的向量，19 以此类推。P 的形式可以是 $||Y\prime - M(D)||$。而模型 M 也可以选择各种模型，比如神经网络，SVM，朴素贝叶斯等等。

B.2 无监督学习

无监督学习是指不给出目标输出，而是希望模型发现其内在规律和特征（autoencoding）。比如聚类问题，这个时候 $Y\prime$ 不再存在，例如 K-means 算法，$P(Y\prime, M(D))$ 退化成为

$$\arg\max_{S} \sum_{i}^{k} \frac{1}{2|S_i|} \sum_{x,y \in S_i} ||x - y||^2$$

这里介绍几个典型的无监督模型：

- **GAN 网络**

  GAN 网络是一种无监督学习网络，广泛用于图像生成应用中，其本身是一个生成模型，由一个判别模型 $\mathbb{D}$ 和一个生成模型 $\mathbb{G}$ 组成，一般这两个模型都是神经网络。$\mathbb{G}$ 网络主要通过一个图像 $\boldsymbol{z}$ 作为输入（一般是随机噪声），生成一个结果 $\mathbb{G}(\boldsymbol{z})$，然后将 $\mathbb{G}(\boldsymbol{z})$ 放入判别网络 $\mathbb{D}$ 中，$\mathbb{D}$ 的目的是区分出 $\mathbb{G}(\boldsymbol{z})$ 和输入数据 d(将生成的和原始输入判定为不同的类型)。简单来说，$\mathbb{G}$ 的目的是欺骗 $\mathbb{D}$，$\mathbb{D}$ 的目的是识别出 $\mathbb{G}$ 生成的图像。损失函数可以写成：

  $$\min_{\mathbb{G}} \max_{\mathbb{D}} \mathbb{E}_{\boldsymbol{x} \in \chi}[\log \mathbb{D}(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \in Z}[\log(1 - \mathbb{D}(\mathbb{G}(\boldsymbol{z})))]$$

- **VAE 网络**

  VAE 网络也是一种生成模型网络，主要由两个部分组成，一个是编码器，将原始数据的分布投影到一个比较简单的多维分布中（e.g. $\mathcal{N}(0,1)$）。另一个是解码

器，通过从这个分布中的采样生成需要的原始数据。度量函数的形式为:

$$\mathcal{L}(\phi, \theta, \boldsymbol{x}) = D_{KL}(\boldsymbol{q}_\phi(z|x)||p\theta(\boldsymbol{z})) - \mathbb{E}_{\boldsymbol{q}_\phi(\boldsymbol{z}|\boldsymbol{x})}(\log p_\theta(\boldsymbol{x}|\boldsymbol{z}))$$

其中 ϕ,θ 为超参，$\boldsymbol{x}$ 变量满足输入数据的分布，$\boldsymbol{z}$ 是编码之后的变量，编码器的目的在于优化右手第一部分，解码器的目的在于优化第二部分。

- **流型学习**

  流型学习的核心问题是通过数据发现数据分布的流型几何拓扑结构。通过将数据从流型映射到平面从而可以使用一些线性聚类分类算法。

B.3 其他类型学习方式

B.3.1 半监督学习

对于半监督学习 [15] 而言，只给定一部分数据的目标输出，而另外一部分需要一些假设进行预估计，譬如连续性假设（空间距离较为接近的标签也应该类似），聚类假设 (聚类为一类的标签结果应该接近)，流形假设（D 投影到某个流形上的距离较近的标签应该类似)。

B.3.2 主动学习

在主动学习中，是没办法直接获得数据的目标标签的，目标 output 需要进行询问，然后再次训练然后再询问这种方式。询问的方法模型也有各种，比如选择分类最不确定的进行询问（比如落 SVM 的 margin 内的数据），选择使得 label 获取后模型 (参数) 改变量最多的数据，等等。这时候度量可以成为

$$P = P\prime(Y\prime_{D_1}, M_1(D_1)) - P\prime(Y\prime_{D_1+D_2}, M(D_1 + D_2))$$

。

B.3.3 增强学习

增强学习的目标是智能体（agent）在环境（environment) 中时间轴 T 上的每一个状态 S 通过执行某个行动 A 所获得的最大化累计奖励 $\sum R$。[16] 下面是几个定义:

- 状态 S：指的是当前环境的情况，比如下棋比赛某一时刻的棋盘情况。
- 行动 A：是指通过这个某个操作，使得状态进行变化，比如走一步棋子。
- 转移函数 T $S \times A \times S\prime \to [0, 1]$ 的应变量，表示从 S 状态通过 A 转移到 $S\prime$ 的概率表示为 $T(S, A, S\prime)$
- 奖励函数 R：状态奖励 $R : S \to R$, 行为奖励:$S \times A \times S \to R$ 在增强学习中，环境被定义为马尔可夫决策过程（MDP)，这里就是 (S, A, T, R) 的四元组，其

目标就是最大化总价值，在行为的过程中可以定义一个衰减系数，使得

$$\text{总价值}v = \text{当前奖励}R + \text{衰减系数}\gamma \times \text{上一个时间步的总价值}v$$

- 决策（Policy）就是根据 $S \to A$ 的一个映射函数，也可以是根据分布 π 的随机决策，其中 $\pi(s,a),\ s.t.\ \sum \pi(s,a) = 1$。

对于增强学习问题，如果 MDP 是已知的，那么整个就是一个动态规划过程，如果 MDP 未知，这个时候就需要进行学习。目前比较流行的学习方法是 Q Learning 和 Policy Gradient。

Q Learning 的核心思想来自于动态规划，主要是计算函数 $Q : S \times A \to R$。通过 Bellman 方程

$$v(s) = \mathbb{E}[R_{t+1} + \gamma v(S_{t+1})|S_t = s]$$

不断提高奖励直到收敛。此外，为了解决状态空间过大的问题，引进深度学习（DQN）来拟合 Q 函数。

Policy Gradient 方法主要是根据输入状态，以及模拟可能的行动，通过最终模拟过程中的奖励设定目标是，进行一次梯度下降来优化期望奖励，也就是说如果这次模拟结果更优，那么 agent 会趋向于选择这次模拟的行动。这样整个模型就是一个端到端的 $S \to A$ 的 policy 模型。

可以说 Q Learning 主要是基于某个状态下行动的期望值的（Value Based）来得到最优行动的，而 PG 是直接预测行为的。除此之外还有 policy 和 value 相结合的方式 Actor-Critic，Actor 和普通 PG 一样，而 Critic 对于每个行动 Action 都进行了 value 的估计，其目标是为了更加准确的评估出当前状态对应的行动的 value 。此外也有 DQN 和 Actor-Critic 相结合的方法（DDPG）等，这里不再表述。

B.3.4 迁移学习

迁移学习主要是根据以往训练的结果，迁移到新的学习任务上去，即使以前的域，任务和数据输入的分布情况和新的域中分布情况不一样，也可以使用迁移学习使得之前的结果可以复用。

- 归纳式迁移学习（ Inductive Transfer Learning ）

  在 Kaggle 竞赛参赛者或者一些入门的深度学习研究者当中，一个典型的使用该过程的例子被称为细化调整 (Fine-tuning)。如果拥有使用大数据集训练好的模型，可以用极少的训练代价达到对小数据集上物体识别或者检测性能极大提升的效果。在高性能计算中心用足量的数据、合理的方式训练一个优秀的模型可以提高整个科研圈的起跑线。

- 传导式迁移学习（ Transductive Transfer Learning ）

迁移学习可以在训练集和验证集的分布不相同，前者有标注，后者无标注的时候，显著的提高后者数据集上的分类能力，这种算法称为传导式迁移学习。一个典型的例证是 J-MMD 算法（Long Mingsheng, Michael Jordan）。通过这一算法，可以在不同图像质量、不同图像大小的数据集上完成对目标数据集上分类器/检测器识别率的极大提高。

- 无监督迁移学习 (Unsupervised Transfer Learning)

  广义的迁移学习也可以用于无监督学习，当所有样本不存在标注，或者不需要进行标注的时候，通过引入生成模型或者无监督学习中常见的损失函数形式，实现对样本之间『相似度』的度量，从而达到某些特定的目的。

  最近热度高的人工智能应用是『风格化迁移』(Style Transfer)。该应用可以将梵高的《星空》或者莫奈的《呐喊》这种艺术大作 S (= Style Picture) 中出现的色调与画风，通过训练或者非训练的方式，和另一张用户自己的照片 C(= Content Picture) 杂糅在一起，变成图像 M (Mixed Picture)，使得 M 的图像特征与 S、C 两张原始图片的特征距离是接近的。尽管这种算法完全没有成熟的的商业模式可言，但来自俄罗斯的网站 Ostagram 、美国的 Prisma App 和中国的 Philm App 都是较为成功的客户端案例，很快包括美图在内的很多相机类应用都加入了这一混战，而且受此启发，越来越多的艺术家和 AI 工程师开始研究 AI 美学，通过对人脸、自然环境的深入研究，对各种各样的图像、视频进行艺术美化、动漫化、加入智能滤镜。

C 分布式算法的云计算实践

尽管很多深度学习、机器学习框架已经实现了主流的单机单卡算法，在实践的过程中发现，为了突破单机单卡训练的能力壁垒，框架纷纷快速支持了单机多卡（主要靠 PCI-E 总线传输数据）以及多机多卡（主要靠万兆以太网传出数据）的训练方式。这部分章节不适合个人设备的参与，仅适合矿厂以联盟链的形式加入，且仅针对训练过程的算力合约销售。

分布式深度学习根据划分策略可以分为模型并行，数据并行，以及其他情况三类，这里分别展开：

C.1 模型并行

模型并行是指单个节点 (单个显卡，或者单机内存) 不容纳整个模型，分别计算然后汇总的做法。由于模型之间连接众多，将模型部署在多个节点通常会导致巨大的通信开销，因此模型并行适用场景有限。根据不同划分手段，模型并行可以分为两类：

1. 第一类是将网络的所有层切分成不同子集。换句话，不同节点运行网络的不同层。RNN 很容易采用这种方式，这在模型尺寸巨大单节点无法容纳的情况下非常有用。CNN 这样做的情况并不多，这是因为当前 CNN 的模型尺寸基本都能控制在显卡显存尺寸之内。不过在稍早时候，Google 的 One Weird Trick

Krizhevsky [17]通过将 CNN 中通信占比最高的全连接层单独划分出去的做法显著降低了 CNN 的通信开销，这种工程实践在大规模 CNN 网络中得到有效使用，例如百度识图搜索。

2. 第二类是将网络的层根据特征划分成不同子层，单个节点运行子层并最终进行汇总。这种做法并不常见，因为这会造成网络中连接巨大从而导致巨大的通信开销，因此不论是 RNN 还是 CNN 都很少在实践中采用。一个例外是 Amazon 开源的 dss [18]，这是一个采用稀疏模型的模型并行 DNN 实现，用于推荐引擎，由于稀疏性导致通信开销巨大的情况并未出现，但它既不属于 RNN 也不属于 CNN ，因此不是通用的解决手段。

总体而言，模型并行由于开销巨大，在不是万不得已的情况下没有采用的必要。尽管对于一些深度学习框架来说，已经内置了该功能，例如 TensorFlow ，Apache SINGA ，但由于性能损耗大，在实际场景中几乎没有使用价值，因此对于大规模分布式深度学习来说，数据并行是目前唯一可靠有效的并行训练手段。

C.2 数据并行

数据并行的工作出发点可以大致归结为两个方面---系统结构和最优化算法。从系统结构来说，深度学习主流框架采用 MPI 或者参数服务器，由于后者便于引入更多的技巧从而降低通信带宽损耗，因此系统结构上的研究工作许多围绕参数服务器来进行，这方面邢波在分布式机器学习的总结 Xing et al. [19]算入门之作，将数据并行分为 BSP 同步、ASP 异步、以及 SSP 半同步三类。BSP 同步模型是最简单和早期的并行手段，它的主要问题在于各节点需要同时更新参数，因此会带来带宽消耗大的问题，所以随着节点数量的增加模型训练加速会越来越受到影响；ASP 全异步可以降低各节点之间同时传递参数的概率，因此在三类当中可以最大程度节约带宽，但这个方法的问题在于各节点之间模型差异过大容易导致收敛缓慢，在实际使用当中反而起到浪费计算资源的作用。Google 第一代深度学习系统 DistBelief 就是采用该机制。为缓解收敛缓慢的问题，一个常用做法是 Model Average 模型平均 Zinkevich et al. [20]，将各节点更新的模型取平均之后再传递回各参数节点，这在 BSP 中也得到广泛使用，目的可以使得收敛更加平滑。总体上模型平均随着节点增加加速也会逐渐变差；而 SSP 有限异步则在证明收敛的基础之上通过有限异步来确保节点之间模型不至于差异过大，从而在带宽节约和模型收敛之间得到一个平衡，因此理论上是数据并行的最佳实践。不过需要指出的是，尽管 SSP 的收敛证明在分布式机器学习领域属于开创性工作，但邢波的工作仍然是围绕传统机器学习，在深度学习上的有效性并没有在实践当中得到验证。这在一些框架如 MXNet 、PaddlePaddle 的实验中已经得到确认，因此尽管邢波团队也随后进行了 SSP 在深度学习上同样收敛的证明 Keuper [21]，但综合各种因素，BSP 依然是实际当中被证明有效的主要手段，这也是为何 MPI 依然在深度学习框架被广泛采用的原因之一。

SSP 开启了并行训练的研究方向之一，后来的跟进者纷纷提出各种 idea，尤其是结合各种最优化算法，例如微软研究院的王太峰 Zheng et al. [22]提出 ASP 跟 SSP 本质上没有什么两样，它们都导致不同节点之间参数过期引起模型差异从而影响收敛，因此

在 SGD 的基础上做泰勒展开，并在不同节点之间利用一阶补偿来弥补差异。这个想法非常直接，尽管并没有在深度学习上收敛的证明，但在实践当中被证明有效，目前相关实现已经被一些深度学习框架如 MXNet 接受。这一类工作启发了不少希望从最优化算法本身取得突破来加速并行训练的研究者。目前所有的深度学习算法都采用 SGD 实现，它的迭代速度快，但是它的收敛速度不如 GD ，从复杂度上来看，SGD 能够在目标函数强凸的情况下做到 $O(1/\epsilon^2)$ 的次线性收敛，而梯度下降则在目标函数强凸时做到 $O(n/\epsilon)$ 的线性收敛。然而，梯度下降的训练会导致 GPU 显存不足，因此 SGD 一直被主流采用至今。SGD 后来也衍生出了很多变种，比如 Adagrad 、Momentum 等，但是并没有在收敛速度上实现突破，直到 SVRG Johnson and Zhang [23]发表。SVRG 的思路是，维护一个全量梯度，每过段时间 (1$\tilde{5}$个 epoch) 计算一次，每次更新用旧模型上算得的全量梯度和当前模型上得到的梯度差来修正。南京大学的李武军团队成功将 SVRG 用于分布式机器学习 Zhao et al. [24]得到显著线性加速。将 SVRG 用于分布式深度学习的努力仍在进行中，理论界已经有一系列工作力图达到一个结构与算法的 “最优” 组合---全异步和线性加速，如 Huo and Huang [25]等，也许在不久之后就能看到它们在深度学习中广泛运用。

更实际一些，由于 BSP 仍然在深度学习中占据统治地位，更多的工作则着眼于如何从结构上和最优化两方面来减少带宽传输。例如 Lecun 首先提出的 EASGD Zhang et al. [26]，可直接运用在 BSP 或者 ASP 上，其思路是在参数服务器的 Worker 和 Server 之间传递参数之前，首先计算两者参数的 Elastic Difference，用来减少实际传输的参数量。更进一步的，微软的 CNTK 提出 1-bit SGD Seide et al. [27]，通过将这种差异量化到 1 bit 的 clip 操作来最大化节约带宽，但它需要额外的预训练来确保收敛不被影响，此外根据 CNTK 试验，尽管 1-bit SGD 减少了带宽传输，但它对模型精度的损耗影响了更多训练节点时的线性加速。CNTK 默认推荐的做法是 Block Momentum Chen and Huo [28]，它是在 Model Average 模型平均 Zinkevich et al. [20]的基础之上增加了模型更新滤波的步骤，将每一轮更新的参数收集起来结合历史参数进行学习，用来确保每轮的更新更加平滑。因此 Block Momentum 本身并没有做带宽压缩的工作，而是通过让收敛更容易来加速训练，这种最优化手段使得 CNTK 是深度学习框架多机并行最快的实现之一。

结构方面一个较大的创新来自邢波团队的 Poseidon Xie et al. [29] Zhang et al. [30]，它在参数服务器的 Master-Worker 结构的基础之上进一步提出了 P2P 结构，每个 Worker 的参数传递可以动态在参数服务器和其他 Worker 之间动态决定，判断依据是传输带宽的估计。之所以 P2P 结构可行，是因为 Poseidon 提出了 SFB(Sufficient Factor Broadcasting) 的概念，将参数矩阵分解为两个低阶矩阵乘积，称作 SF(Sufficient Factor)，而 SF 的数据量在某些时刻是可以低于原始参数矩阵的。通过将 SF 在不同 Worker 之间的 P2P 结构传输，可以缓解参数服务器本身单点带宽阻塞的问题，因此在实际使用中得到了一定证明，例如在 Inception 网络中可以做到多机线性加速。

来自产业界的工程团队更多从 BSP 本身的结构优化来做工作，例如来自百度的工作 Gibiansky [31]，以及 Uber 根据百度工作实现的分布式 TensorFlow 版本 Horovod hor

[32]，通过变更传统 BSP 的 AllReduce 为 Ring AllReduce ，使得在更大规模集群上的并行训练可以得到有效加速。此外，IBM 未公开实现细节的 Power AI Cho et al. [33]则进一步提出了 Multi Ring AllReduce，宣称在 256 卡上做到了线性加速。

C.3 其他

由前述可见，提升并行训练性能的关键在于带宽节约和有效收敛之间兼顾。因此，从另外角度来看待，部分模型压缩技术本身也可归入并行深度学习的范畴。除了通过精度损失来近似参数 Dettmers [34]之外，一篇 ICLR 2018 双盲审的工作 authors [35] 值得关注，该工作通过动量矫正、局部梯度裁剪、动量因子屏蔽和 warm up train 的手段将并行训练梯度交换的参数压缩了上百倍，可以用于廉价的万兆以太网上实现大规模分布式训练。

D 实验室积累

Cortex 实验室积累的处于世界前沿的模型有：

OCR(ReCaptcha 99%+)

预计研发并公开给 Cortex 生态社区的模型包括：

Instance-Aware Segmentation 与 Mask-RCNN Keypoint Detection 图像语言描述 (Image Captioning, AIC 竞赛 Top 20 围棋算法 (将前沿图像模型整合进围棋算法)

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2009.

[2] Ethereum white paper. `https://github.com/ethereum/wiki/wiki/White-Paper`.

[3] Conway's game of life. `https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life`.

[4] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, pages 507–527, 2015. doi: 10.1007/978-3-662-47854-7_32. URL `https://doi.org/10.1007/978-3-662-47854-7_32`.

[5] Rsk:bitcoin powered smart contracts. `http://www.the-blockchain.com/docs/Rootstock-WhitePaper-Overview.pdf`.

[6] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the aes circuit. Cryptology ePrint Archive, Report 2012/099, 2012. https://eprint.iacr.org/2012/099.

[7] Dagger-hashimoto. https://github.com/ethereum/wiki/blob/master/Dagger-Hashimoto.md.

[8] Alex Biryukov and Dmitry Khovratovich. Equihash: Asymmetric proof-of-work based on the generalized birthday problem. Cryptology ePrint Archive, Report 2015/946, 2015. https://eprint.iacr.org/2015/946.

[9] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. *CoRR*, abs/1512.06473, 2015. URL http://arxiv.org/abs/1512.06473.

[10] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015. URL http://arxiv.org/abs/1510.00149.

[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. *Calibrating Noise to Sensitivity in Private Data Analysis*, pages 265–284. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-32732-5. doi: 10.1007/11681878_14. URL https://doi.org/10.1007/11681878_14.

[12] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304, New York, NY, USA, 1985. ACM. ISBN 0-89791-151-2. doi: 10.1145/22145.22178. URL http://doi.acm.org/10.1145/22145.22178.

[13] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-506-2. doi: 10.1145/1536414.1536440. URL http://doi.acm.org/10.1145/1536414.1536440.

[14] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, June 2002. ISSN 0163-5700. doi: 10.1145/564585.564601. URL http://doi.acm.org/10.1145/564585.564601.

[15] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[16] Martijn van Otterlo and Marco Wiering. *Reinforcement Learning and Markov Decision Processes*, pages 3–42. Springer Berlin Heidelberg, Berlin, Heidelberg,

2012. ISBN 978-3-642-27645-3. doi: 10.1007/978-3-642-27645-3_1. URL https://doi.org/10.1007/978-3-642-27645-3_1.

[17] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997, 2014. URL http://arxiv.org/abs/1404.5997.

[18] Amazon dsstne: Deep scalable sparse tensor network engine. 2016.

[19] Eric P. Xing, Qirong Ho, Pengtao Xie, and Dai Wei. Strategies and principles of distributed machine learning on big data. *Engineering*, 2(2):179 – 195, 2016. ISSN 2095-8099. doi: https://doi.org/10.1016/J.ENG.2016.02.008. URL http://www.sciencedirect.com/science/article/pii/S2095809916309468.

[20] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J. Smola. Parallelized stochastic gradient descent. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2595–2603. Curran Associates, Inc., 2010. URL http://papers.nips.cc/paper/4006-parallelized-stochastic-gradient-descent.pdf.

[21] Janis Keuper. Distributed training of deep neuronal networks: Theoretical and practical limits of parallel scalability. *CoRR*, abs/1609.06870, 2016. URL http://arxiv.org/abs/1609.06870.

[22] Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhiming Ma, and Tie-Yan Liu. Asynchronous stochastic gradient descent with delay compensation for distributed deep learning. *CoRR*, abs/1609.08326, 2016. URL http://arxiv.org/abs/1609.08326.

[23] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *International Conference on Neural Information Processing Systems*, pages 315–323, 2013.

[24] Shen Yi Zhao, Ru Xiang, Ying Hao Shi, Peng Gao, and Wu Jun Li. Scope: Scalable composite optimization for learning on spark. 2016.

[25] Zhouyuan Huo and Heng Huang. Asynchronous stochastic gradient descent with variance reduction for non-convex optimization. 2016.

[26] Sixin Zhang, Anna Choromanska, and Yann LeCun. Deep learning with elastic averaging SGD. *CoRR*, abs/1412.6651, 2014. URL http://arxiv.org/abs/1412.6651.

[27] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns. In *Interspeech 2014*, September 2014.

[28] Kai Chen and Qiang Huo. Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5880–5884, 2016.

[29] Pengtao Xie, Jin Kyu Kim, and Eric P. Xing. Large scale distributed multiclass logistic regression. *CoRR*, abs/1409.5705, 2014. URL `http://arxiv.org/abs/1409.5705`.

[30] Hao Zhang, Zeyu Zheng, Shizhen Xu, Wei Dai, Qirong Ho, Xiaodan Liang, Zhiting Hu, Jinliang Wei, Pengtao Xie, and Eric P. Xing. Poseidon: An efficient communication architecture for distributed deep learning on GPU clusters. *CoRR*, abs/1706.03292, 2017. URL `http://arxiv.org/abs/1706.03292`.

[31] Andrew Gibiansky. Bringing hpc techniques to deep learning.

[32] Horovod. `https://github.com/uber/horovod`, 2017.

[33] Minsik Cho, Ulrich Finkler, Sameer Kumar, David S. Kung, Vaibhav Saxena, and Dheeraj Sreedhar. Powerai DDL. *CoRR*, abs/1708.02188, 2017. URL `http://arxiv.org/abs/1708.02188`.

[34] Tim Dettmers. 8-bit approximations for parallelism in deep learning. *CoRR*, abs/1511.04561, 2015. URL `http://arxiv.org/abs/1511.04561`.

[35] Anonymous authors. Deep gradient compression-reducing the communication bandwidth for distributed training. *ICLR*, 2018.