Deuro: AI技术白皮书

# 摘要

当我们开发人工智能软件解决方案时，我们意识到大约10％到30％的预算用于人工智能云计算基础设施，并且至少有数百万美元用于标记的培训数据。 此外，我们还经历了企业客户提出的信任和安全问题。 这就是Deuro的用武之地.Deuro是世界上第一个也是唯一一个区块链上的人工智能生态系统。 通过权力下放，开放治理，网络安全和代币经济，Deuro可以帮助AI公司大幅降低计算成本，缩短AI系统开发周期，提高生产数据隐私和安全性。

## 由Deuro提供的 人工智能解决方案

        1. gStorage: 隐私保护去中心化的人工智能客户端驱动的数据存储系统

                1.a. 特征提取（数据预处理）

                1.b.去中心化的伪匿名多方密钥加密和解密

                1.c. AI数据作为资产

                1.d. 企业级别解决方案

                1.e. 消费者级别解决方案

                1.f. 建模集存储库

                1.g. 预测数据存储

                1.h. 模型中心

        2 gCrawl:  去中心化的高性能算法自动生成的训练数据解决方案

                2.a. 爬虫任务作为交易

                2.b. 爬到的数据作为资产

                2.c. 容错分布式高性能爬网

                2.d. RAFT一致性共识算法

                2.e.  去中心化的数据存储

                2.f.   DFSM解码路径

                2.g. 转移学习以更新模型参数

       3. gPredict: 去中心化的分布式梯度学习框架

                3.a. 训练任务作为交易

                3.b. 训练完备的模型作为资产

                3.c. 浮点映射

                3.d. 超参数路由

                3.e. 结果回归

       4. gCompute: 互联网规模的AI dApps解决方案

                4.a. AI模型作为资产

                4.b. 可扩展性

                4.c. 可控制性

                4.d. 系统稳定程度

                4.e. 可维护性

                4.f.  响应性

                4.g. 高并发性

# 关键词

**AI dApps**: 分散式人工智能应用程序，其后端通常是区块链

**Weak-AI**: 弱人工智能/应用人工智能

**AGI**: 通用人工智能

**Brute-force search**: 在计算机科学中，蛮力搜索或穷举搜索（也称为生成和测试）是一种非常普遍的问题解决技术，它包括系统地枚举解决方案的所有可能候选者，并检查每个候选人是否满足问题的陈述。

**AlphaGo**(missing citation): AlphaGo是一款玩电脑游戏Go的计算机程序。它由Alphabet Inc.在伦敦的Google DeepMind开发.

**GD**: 梯度下降是用于找到函数最小值的一阶迭代优化算法。 为了使用梯度下降找到函数的局部最小值，需要采用与当前点处函数的梯度（或近似梯度）的负值成比例的步长。 相反，如果采用与梯度的正值成比例的步长，则接近该函数的局部最大值; 然后将该过程称为梯度上升。

**Supervised-Learning**: 监督学习是学习函数的机器学习任务，该函数基于示例输入 - 输出对将输入映射到输出

**Unsupervised-learning**: 无监督学习是一种机器学习算法，用于从没有标记响应的输入数据组成的数据集中得出推论。

**NBC**: 在机器学习中，朴素贝叶斯分类器是一系列简单的“概率分类器”，它基于贝叶斯定理应用特征之间的强（天真）独立假设。

**CA**: 聚类分析的任务是对一组对象进行分组，使得同一组（称为集群）中的对象（在某种意义上）与其他组（集群）中的对象更相似（在某种意义上）

**DMR**: 维数减少或降维是通过获得一组主要变量来减少所考虑的随机变量数量[1]的过程。 它可以分为特征选择和特征提取

**PCA**: 主成分分析（PCA）是一种统计过程，它使用正交变换将可能相关变量的一组观察值转换为称为主成分的线性不相关变量的一组值。

**K-NN**: k-NN是一种基于实例的学习或懒惰学习，其中函数仅在本地近似，并且所有计算都推迟到分类。 k-NN算法是所有机器学习算法中最简单的算法之一。

**SVM**: 支持向量机（SVM，也支持向量网络）是带有相关学习算法的监督学习模型，用于分析用于分类和回归分析的数据

**RL**: 强化学习（RL）是受行为主义心理学启发的机器学习领域[引证需要]，关注软件代理应如何在环境中采取行动以最大化一些累积奖励的概念。

**AI-complete**: 在人工智能领域，最困难的问题是非正式地称为AI-complete或AI-hard，这意味着这些计算问题的难度等同于解决中心人工智能问题 - 使计算机像人一样聪明， 或强大的AI

**NP-complete**: 在计算复杂性理论中，NP完全决策问题是属于NP和NP难复杂类的问题。 在这种情况下，NP代表“非确定性多项式时间”。 NP完全问题的集合通常用NP-C或NPC表示

**DL**: 深度学习（也称为深度结构化学习或分层学习）是基于学习数据表示的更广泛的机器学习方法系列的一部分，而不是特定于任务的算法。 学习可以是监督，半监督或无监督

**CNN**: 卷积神经网络（CNN，或ConvNet）是一类深度前馈人工神经网络，已成功应用于分析视觉图像。

**RNN:**递归神经网络（RNN）是一类人工神经网络，其中单元之间的连接形成沿序列的有向图。

**LSTM:**长短期记忆（LSTM）单位（或块）是递归神经网络（RNN）的层的构建单元。 由LSTM单元组成的RNN通常称为LSTM网络。 公共LSTM单元由单元，输入门，输出门和忘记门组成

**Singularity**: 技术奇点（也就是简单地说，奇点）是人造超级智能（ASI）的发明将突然引发技术增长失控的假设，导致人类文明的不可思议的变化。 根据这一假设，可升级的智能代理（例如运行基于软件的人工通用智能的计算机）将进入自我改善周期的“失控反应”，每个新的和更智能的一代出现越来越快，导致 智能爆炸并产生强大的超级智能

**GAN**: 生成对抗网络（GAN）是一种用于无监督机器学习的人工智能算法，由两个神经网络系统在零和游戏框架中相互竞争实现

**SAAS**: 软件即服务是一种软件许可和交付模型，其中软件是基于订阅许可的，并且是集中托管的。

**Colo**: 托管中心（也称拼写共址或colo）或“运营商酒店”是一种数据中心，其中设备，空间和带宽可供零售客户租用。

**On-premise distribution**: 内部部署软件（有时是“内部部署”或缩写为“本地内部”）安装并使用软件在个人或组织的场所（建筑物内）的计算机上运行，而不是在远程设施上运行。 作为服务器场或云

**DFSM**: 确定性有限状态机是一种有限状态机，它接受和拒绝符号串，并且只为每个输入字符串产生自动机的唯一计算（或运行）

**MSE**: 均方误差

**BFT**: 拜占庭容错

# Deuro提供的解决方案

在这里，我们提出了基于区块链的创新AI解决方案，以解决上述问题。

**gStorage**: 对于数据安全和隐私问题，我们提出了gStorage解决方案。 考虑到AI特殊用例和区块链智能合约支持，我们引入了分布式端到端加密解决方案来存储/检索培训数据，生产预测数据和模型文件。

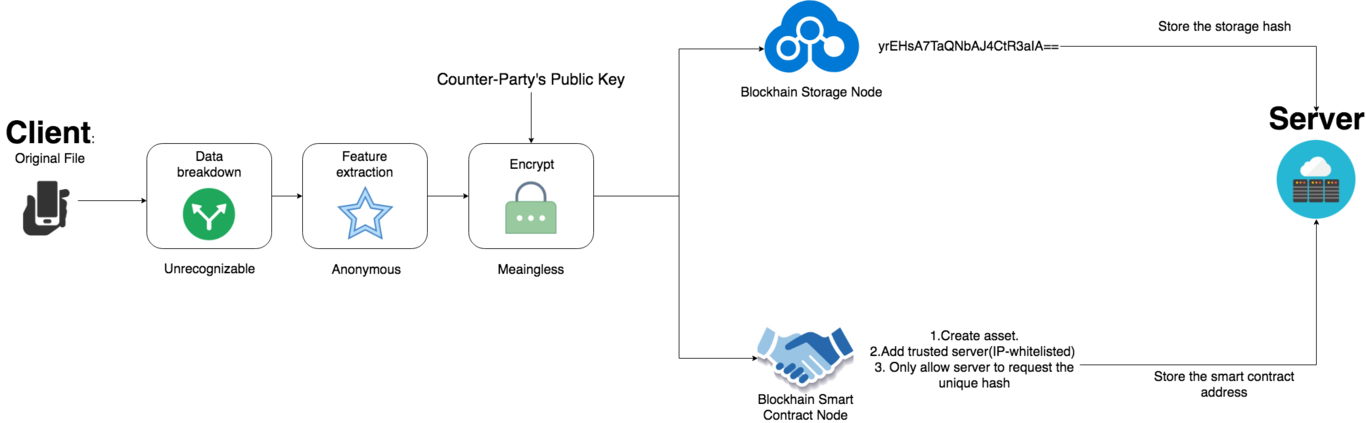
**gCrawl**: 由于缺乏标签数据问题，我们提出了创新的gCrawl架构。 gCrawl系统由容错分布式高性能数据爬行系统，有状态NoSQL数据库（重复数据删除以及元数据存储），分散存储，模型参数传输学习以完成反馈循环组成。

**gPredict**: 对于算法准确性问题，我们提出了一种创新的方法来训练/评估/测试区块链上的模型性能。 gPredict包括浮点映射，超参数路由和解码结果回归。

**gCompute**: 对于开发，测试和部署AI系统时的高硬件和软件成本，我们引入了gCompute模块。 gCompute模块具有可扩展性，可控性，难度，可维护性，响应性和并发性。

## 1. gStorage: 隐私保护去中心化的人工智能客户端驱动的数据存储系统

为了以无信任的方式安全地存储用户数据，我们必须确保客户的隐私和安全。 在这里，我们提出了一种在区块链上存储用户数据的创新方法。 以下是gStorage工作原理的工作流程图。



gStorage 工作流程图

## 1.a. 特征提取（数据预处理）

在这里，我们展示了语音识别和图像识别中使用的共同特征。 音频功能：时间特征，光谱特征。 图像特征：边缘，角落/兴趣点，斑点/感兴趣区域，脊。

## 时间特征

class TF(object):

    def \_\_init\_\_(self, nfilt=40, ncep=13,

                 lowerf=133.3333, upperf=6855.4976, alpha=0.97,

                 samprate=16000, frate=100, wlen=0.0256,

                 nfft=512):

        # Store parameters

        self.lowerf = lowerf

        self.upperf = upperf

        self.nfft = nfft

        self.ncep = ncep

        self.nfilt = nfilt

        self.frate = frate

        self.fshift = float(samprate) / frate

        # Build Hamming window

        self.wlen = int(wlen \* samprate)

        self.win = numpy.hamming(self.wlen)

        # Prior sample for pre-emphasis

        self.prior = 0

        self.alpha = alpha

        # Build mel filter matrix

        self.filters = numpy.zeros((nfft/2+1,nfilt), ‘d’)

        dfreq = float(samprate) / nfft

        if upperf > samprate/2:

            raise(Exception,

                   “Upper frequency %f exceeds Nyquist %f” % (upperf, samprate/2))

        melmax = mel(upperf)

        melmin = mel(lowerf)

        dmelbw = (melmax - melmin) / (nfilt + 1)

        # Filter edges, in Hz

        filt\_edge = melinv(melmin + dmelbw \* numpy.arange(nfilt + 2, dtype=’d’))

        for whichfilt in range(0, nfilt):

            # Filter triangles, in DFT points

            leftfr = round(filt\_edge[whichfilt] / dfreq)

            centerfr = round(filt\_edge[whichfilt + 1] / dfreq)

            rightfr = round(filt\_edge[whichfilt + 2] / dfreq)

            # For some reason this is calculated in Hz, though I think

            # it doesn’t really matter

            fwidth = (rightfr - leftfr) \* dfreq

            height = 2. / fwidth

            if centerfr != leftfr:

                leftslope = height / (centerfr - leftfr)

            else:

                leftslope = 0

            freq = leftfr + 1

            while freq < centerfr:

                self.filters[freq,whichfilt] = (freq - leftfr) \* leftslope

                freq = freq + 1

            if freq == centerfr: # This is always true

                self.filters[freq,whichfilt] = height

                freq = freq + 1

            if centerfr != rightfr:

                rightslope = height / (centerfr - rightfr)

            while freq < rightfr:

                self.filters[freq,whichfilt] = (freq - rightfr) \* rightslope

                freq = freq + 1

        # Build DCT matrix

        self.s2dct = s2dctmat(nfilt, ncep, 1./nfilt)

        self.dct = dctmat(nfilt, ncep, [numpy.pi/nfilt](http://numpy.pi/nfilt))

    def sig2s2mfc(self, sig):

        nfr = int(len(sig) / self.fshift + 1)

        mfcc = numpy.zeros((nfr, self.ncep), ‘d’)

        fr = 0

        while fr < nfr:

            start = round(fr \* self.fshift)

            end = min(len(sig), start + self.wlen)

            frame = sig[start:end]

            if len(frame) < self.wlen:

                frame = numpy.resize(frame,self.wlen)

                frame[self.wlen:] = 0

            mfcc[fr] = self.frame2s2mfc(frame)

            fr = fr + 1

        return mfcc

    def sig2logspec(self, sig):

        nfr = int(len(sig) / self.fshift + 1)

        mfcc = numpy.zeros((nfr, self.nfilt), ‘d’)

        fr = 0

        while fr < nfr:

            start = round(fr \* self.fshift)

            end = min(len(sig), start + self.wlen)

            frame = sig[start:end]

            if len(frame) < self.wlen:

                frame = numpy.resize(frame,self.wlen)

                frame[self.wlen:] = 0

            mfcc[fr] = self.frame2logspec(frame)

            fr = fr + 1

        return mfcc

    def pre\_emphasis(self, frame):

        # FIXME: Do this with matrix multiplication

        outfr = numpy.empty(len(frame), ‘d’)

        outfr[0] = frame[0] - self.alpha \* self.prior

        for i in range(1,len(frame)):

            outfr[i] = frame[i] - self.alpha \* frame[i-1]

        self.prior = frame[-1]

        return outfr

    def frame2logspec(self, frame):

        frame = self.pre\_emphasis(frame) \* self.win

        fft = numpy.fft.rfft(frame, self.nfft)

        # Square of absolute value

        power = fft.real \* fft.real + fft.imag \* fft.imag

        return numpy.log(numpy.dot(power, self.filters).clip(1e-5,numpy.inf))

    def frame2s2mfc(self, frame):

        logspec = self.frame2logspec(frame)

        return numpy.dot(logspec, self.s2dct.T) / self.nfilt

def s2dctmat(nfilt,ncep,freqstep):

    “”“Return the ‘legacy’ not-quite-DCT matrix used by Sphinx”“”

    melcos = numpy.empty((ncep, nfilt), ‘double’)

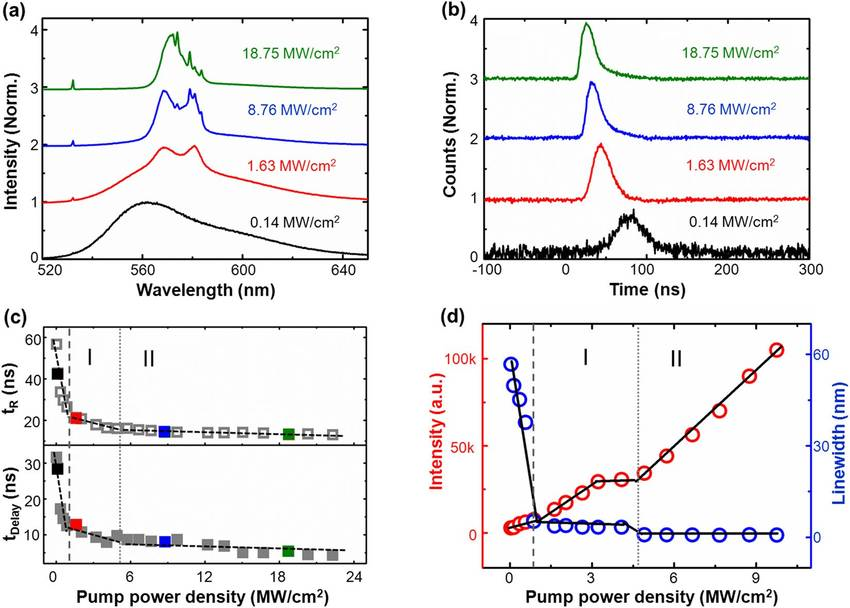
    for i in range(0,ncep):

        freq = numpy.pi \* float(i) / nfilt

        melcos[i] = numpy.cos(freq \* numpy.arange(0.5, float(nfilt)+0.5, 1.0, ‘double’))

    melcos[:,0] = melcos[:,0] \* 0.5

    return melcos



时间特征

**边缘 Edge**

边缘是两个图片区域之间存在限制（或边缘）的焦点。 通常，边缘可以具有相对自由的形状，并且可以包含交叉点。 通过边缘，边缘通常被表征为图像中具有立体角度范围的焦点组。 此外，一些常规计算然后将高倾斜度链接在一起朝向框架更完整地描绘边缘。 这些计算通常会对边缘的属性施加一些限制，例如形状，平滑度和倾向度。 在局部，边缘具有一维结构。

**角落/兴趣点 Corners**

术语角和关注焦点在某种程度上相反地被利用并且暗示图片中的点状高光，其具有邻域二维结构。 由于早期计算最初进行了边缘发现，因此出现了“角落”这个名称，然后在边缘发生故障以发现路线（角落）的快速变化。 然后，这些计算的目标是不再需要表达边缘识别，例如通过在图像倾斜中搜索异常的形状量。 然后可以看到，声称的角落同样被识别在图片的部分区域上，这些部分不是传统意义上的角落（例如，在昏暗的基础上可能会发现一个小亮点）。 这些焦点一下子被称为关注焦点，然而表达“角落”被自定义使用。

**团 /感兴趣的区域点 Bolb**

团给出了图片结构的相关描述，就像语言环境而言，而不是更像点的角落。 通过团描述符可以定期地包含优选点（管理员反应的最极端的邻域或重点），这意味着许多团标识符同样可以被视为兴趣点管理员。 斑点查找器可以区分图片中的区域，这些区域过于平滑，无法以任何方式被角落指示器识别。

**山脊 Ridges**

对于拉伸物品，边缘的思想是一种特征装置。 从昏暗级别图片处理的边缘描述符可被视为平均枢轴的推测。 从实用的角度来看，边缘可以被认为是与对称中心对话的一维弯曲，并且更具有与每个边缘点相关的邻域边缘宽度的质量。 可悲的是，尽管如此，在边缘高光与一般类别的暗级图像之间将边缘高光与边缘，角落或斑点高光区分开来在算法上更难。 通过边缘描述符，尽可能多地用于飞行图片中的街道提取以及用于去除恢复图片中的静脉 - 参见边缘识别。

## 1.b. 去中心化的伪匿名多方密钥加密和解密

在这里，我们将讨论gStorage的核心。 去中心化的伪匿名多方密钥加密和解密算法。

1. 它在某种意义上是去中心化的，我们将使用区块链作为后端，并将签名的交易广播到整个网络，而不会依赖在可信的第三方之上，因为后者可能会导致单点故障。
2. 然后，它在某种意义上是伪匿名的，我们只保护提取的功能并将加密的哈希暴露给整个网络。 就像比特币的工作原理一样（用户使用私钥在本地对交易进行签名，然后将交易哈希广播到整个网络进行验证）。
3. 它涉及多方，在某种意义上我们需要提前知道对方公钥，以便对方（服务器，物联网设备，网站，IP地址，智能合约）可以在以后使用他们自己的私钥来对加密的哈希解密 。

**保护提取的特征**

一旦我们为AI模型提取了特征表示，我们就需要保护它。 我们通过使用对方的公钥将提取的特征加密为密文来保护它。 这里我们将使用行业标准的公钥加密算法。

1. 每一方都有一个Pair(K，K-1)密钥：K是公钥，K-1是密钥，这样DK-1 [EK [M]] = M
2. 因为知道公钥和密码，计算私钥在计算上是不可行的因此公共密钥加密系统因此被称为非对称加密系统
3. 公钥K可以公开提供，例如，在公共可用索引之中
4. 许多人可以加密，只有一个人可以解密

密钥生成：

    选择大小相同，p和q的2个大素数。

    计算n = pq，并且Φ（n）=（q-1）（p-1）。

    选择随机整数e，1 <e <Φ（n），s.t。 gcd（e，Φ（n））= 1。

    计算d，1 <d <Φ（n）s.t. ed = 1 modΦ（n）。

    公钥：（e，n）密钥：d

加密：

    给定消息M，0 <M <n使用公钥（e，n），计算C(密文) = M mod n

解密:

    M  Zn− {0} CZn−{0}.

    给出密文C，使用私钥（d）解密.

    计算Cd modn=(Me modn)d modn=Med mod n = M

**广播加密的哈希**

一旦我们将提取的特征加密，我们将进入p2p分散数据库。 然后将加密特征的唯一散列注册到区块链中，并将此加密散列广播到整个网络中。 矿工将知道特征的加密哈希，可以找到属于单个用户的所有特征数据，但矿工无法解密它（因为矿工没有加密时的私钥）。 就像所有比特币矿工都可以找到属于单个用户的交易一样，但他们无法将资金转入矿工自己的账户，因为矿工没有对方账户的私钥。

**将资产所有权转让给对方**

一旦我们在区块链上注册了特征资产，服务器就可以请求属于特定用户的唯一哈希特征。 然后，服务器可以从p2p分散数据库中检索加密的密文，并使用自己的私钥解密特征的密文。

———-

Solidity ETH access\_control\_template()

———-

pragma solidity ^0.4.23;

contract gStorageAccessControl {

    address admin;

    address server\_address;

    modifier onlyByServer { if (msg.sender != server\_address) throw; \_ }

    modifier onlyByAdmin { if (msg.sender != admin) throw; \_ }

     address[] public features;

    event FindTheUniqueHash(address[] unique\_hash);

    function gStorageAccessControl(address server\_address\_ ) {

        admin = msg.sender;

        server\_address = server\_address\_;

    }

    function registerFeatures(address unique\_hash) onlyByAdmin{

        features.push(unique\_hash);

    }

    function transfer() onlyByServer {

        //Retrive data from the storage

        FindTheUniqueHash(features);

    }

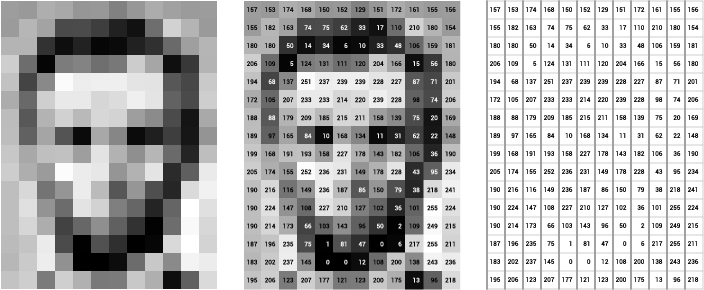
}

**保护隐私的伪匿名数据共享**

通过这种方式，我们通过应用特征提取来保护客户的隐私。 我们使用行业标准公钥加密来保护数据的安全性。 我们还通过将服务器的公钥传递给智能合约来强制执行权限控制。

## 1.c. AI数据（图像，语句）作为资产

图像数据有时有令人困惑的各种形式。 可以表示声音的基本方式的数量实际上相当小。 各种图像文件类型是由于存在很多压缩图像数据的方法和许多不同的数据打包方式。 我们可以将图像抽象为区块链上的资产。 资产也可以是状态机，其中状态转换在元数据中表示。 每次机器更改其状态时，都会触发事务以将元数据更新为新状态（使用WebSocket或HTTP rest方式监听它的状态变化）。 在我们的例子中，我们可以在这种资产状态机转换方式中实现timestamped\_private\_key。



图像矩阵表示形式

## 1.d. 企业级别解决方案

对于企业客户，我们将为他们部署开箱即用的解决方案。 客户需要购买我们的通证才能接收我们的服务。 完成服务后，我们将销毁他们购买的通证。

## 1.e. 消费者级别解决方案

对于面向消费者的应用程序，开发人员可以通过简单地安装我们的SDK快速让gStorage处理他们的应用程序。 我们将实施免费增值模式，一旦达到某些使用次数，我们将要求他们购买一定数量的通证作为储备。 我们将根据API调用销毁通证。

## 1.f. 建模集存储库

考虑到上述架构，我们可以轻松地将所有培训数据存储到区块链对等数据库中。 对于音频训练数据，我们可以存储原始的未压缩或无损压缩的wav文件和相应的标准答案。

## 1.g. 预测数据存储

对于AI dApps生成的所有生产数据，它也可以通过我们现有的gStorage管道。 并允许用户安全地检索以及发送文件。

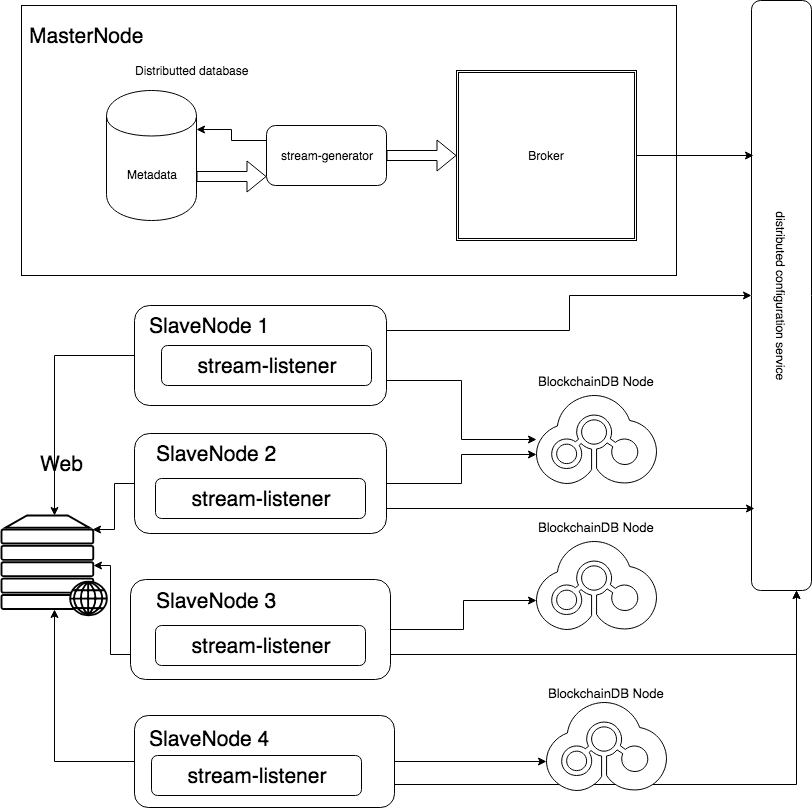
## 1.h. 模型中心

对于流行的深度学习框架（如Tensor-flow，Keras，Pytorch，Kaldi，Caffee，CNTK等）生成的模型，我们也可以使用上述方法从区块链数据库中安全地存储和获取它们。

## 2. gCrawl: 分布式高性能算法自动生成的训练数据解决方案

我们展示了一种名为“gCrawl”的全新机器学习架构，用于在监督分类任务中使用未标记的信息。 我们不假设未标记的数据采用与标记数据类似的类名或生成描述。 因此，我们可能想要利用从因特网任意下载的大量未标记图片（或声音示例或内容记录）来增强对给定图片（或声音或内容）任务的执行。 我们描述了一种gCrawl的方法，该方法使用简化的解码路径和传递学习来从未标记的数据迭代模型。

下面是gCrawl计算集群的体系结构，它是水平可扩展的。 在实践中，我们有能力处理 **上千万个爬虫任务!**



gCrawl爬虫系统

## 2.a. 将爬虫任务作为交易

我们将爬行任务抽象为区块链分类帐上的交易。

contract CrawlingTask {  
  
 struct CrawlingMachine {  
 bool crawled;  
 address location;  
 }  
   
 struct Task {  
 string url;  
 }  
  
 address submiter;  
 mapping(address => CrawlingMachine) machines;  
 Task[] tasks;  
   
 function crawl() {  
 //The crawling business logic  
 }  
}

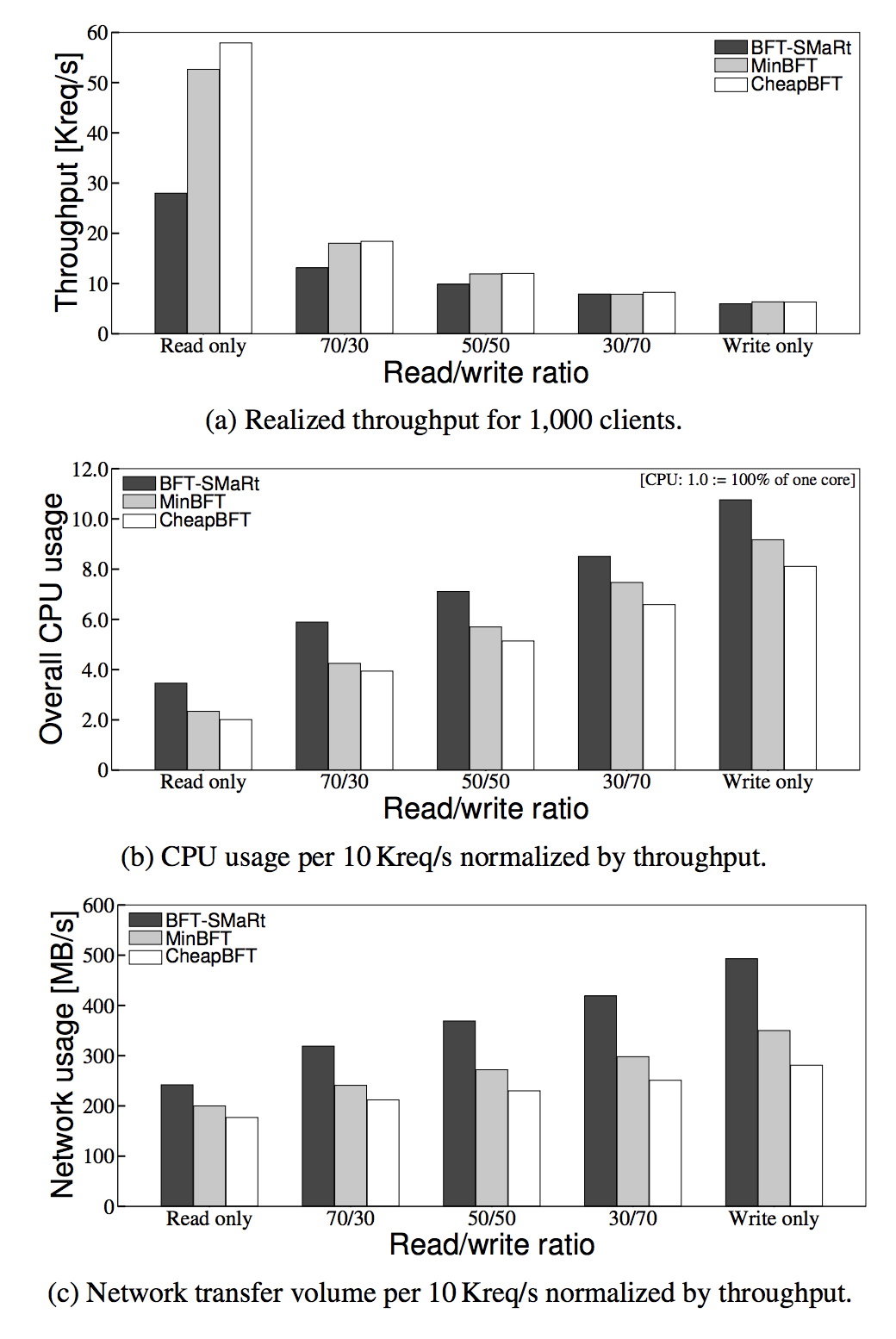
## 2.b. 爬到的数据作为资产

我们可以将已爬到的数据抽象为区块链中的资产

contract CrawledData {  
 string[] metadata;  
 string transcript;  
 address audio\_url;  
}

## 2.c. 高容错分布式高性能爬虫系统

ZooKeeper是一种容错协调服务，用作大型分布式框架的一部分，用于执行先决策，同步和故障识别等基本任务。 此部分显示了对类似ZooKeeper的BFT收益的评估，该收益依赖于BFT-SMaRt，MinBFT和CheapBFT分别用于请求传播。 ZooKeeper使客户能够在信息中心中存储和恢复（通常很少）数据块，这些信息中心在各种级别的树结构中受到监控。 我们评估了各种读取和撰写任务混合的三种执行方式。 在所有情况下，1,000个客户不止一次地访问各种信息中心，仔细阅读并撰写一个字节到两千字节之间任意大小的信息。结果表明，在执行阶段（即ZooKeeper应用程序）执行真正的工作（而不仅仅是发送答案），组合约定对框架执行的影响减弱了。 结果，三个ZooKeeper使用中的每一个都提供了比较吞吐量来组成压倒性的工作负载。尽管如此，资产印象在变化之间存在根本对比：与基于MinBFT的ZooKeeper相比，基于CheapBFT的变体中的复制品可以节省7-12％的CPU，并且可以通过系统减少12-20％的信息。 与BFT-SMaRt执行相比，基于CheapBFT的ZooKeeper的资产资金利用23-42％（CPU）。



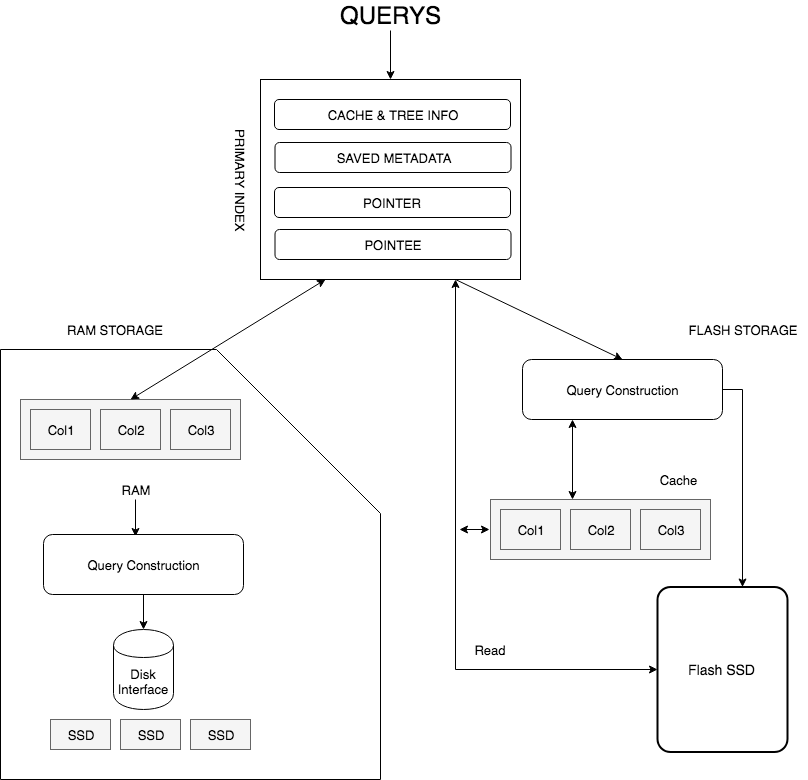
我们的ZooKeeper服务的不同BFT变体的执行和资源利用结果包括工作负载，包括独特的读写操作组合。

## 2.d. RAFT一致性算法

Raft是许多NoSQL数据库中使用的一致性算法。 我们使用NoSQL数据库来存储爬网任务的所有元数据信息。 以下是RAFT一致性算法的领导者选举的伪代码

### 

# follower not receiving heartbeats from leader  
if election\_timeout?  
 increment\_term()  
 change\_state\_to\_candidate()  
 vote\_myself()  
 send\_request\_vote\_to\_all\_partners()  
 while candidate?  
 wait\_feedback(random\_time)  
 process\_feedback()  
 foreach entry in append\_entries\_received()  
 if self.term() < entry.term()  
 mark\_current\_election\_as\_lost()  
 end  
 end  
 if i\_won\_election?  
 change\_state\_to\_leader()  
 start\_heartbeat\_signals()  
 else if partner\_won\_election?  
 change\_state\_to\_follower()  
 else if no\_one\_won\_election?  
 increment\_term()  
 end  
 end  
end  
  
# continue as leader or follower...



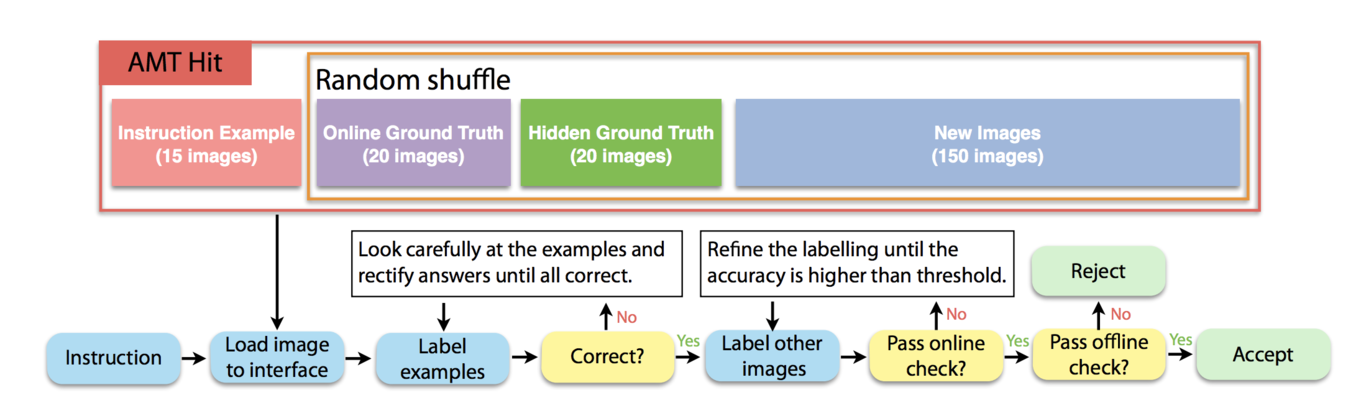
分布式数据库存储系统

## 2.e. 去中心化的数据存储

我们将使用不同的分散式数据区块链存储解决方案来存储已爬到的数据。 我们已经在第1节中详细介绍存储部分。

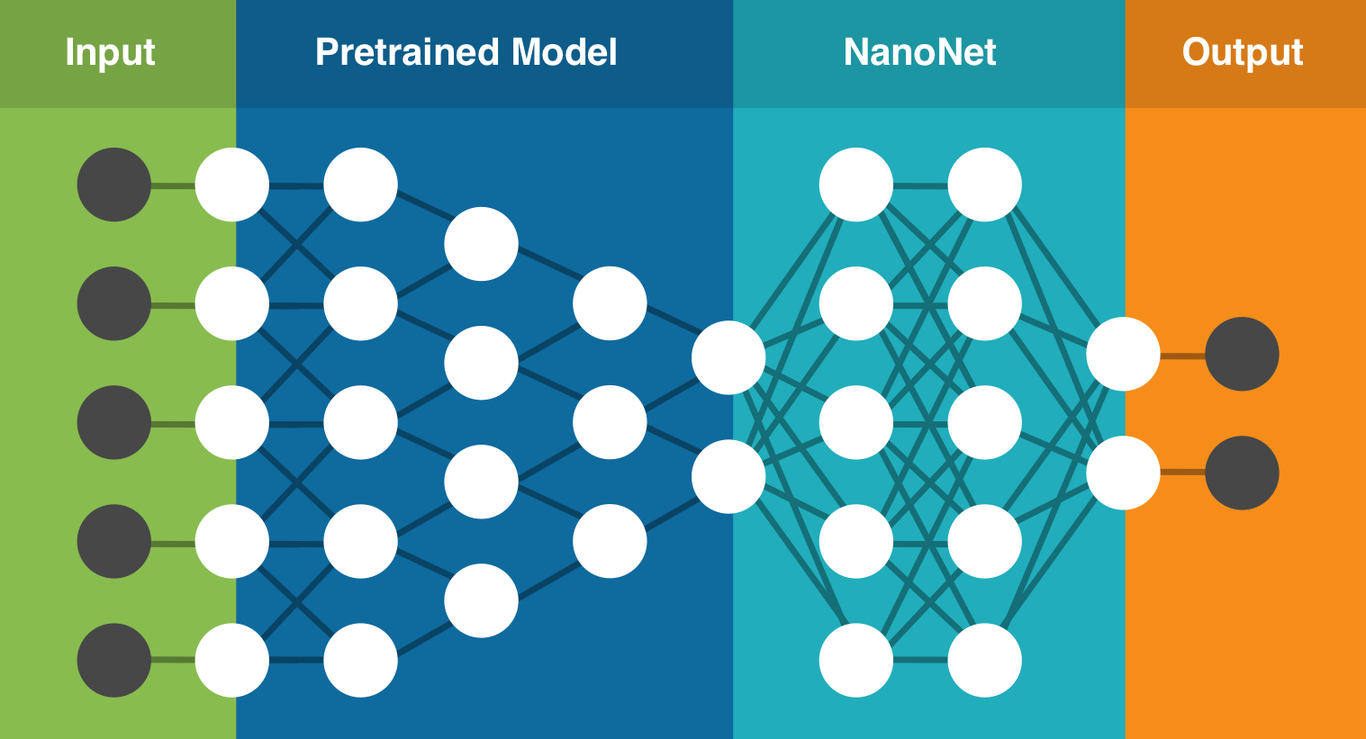
## 2.f. DFSM解码路径

对于每个类别，我们首先使用关键字搜索（通常~106-108张图片）收集大量潜在图片。 然后在某个时刻，我们重复请求个人标记一个小子集，在该子集上准备分类器，要求分类器预见标记和置信度，然后选择一个小的申请人集进一步考虑。



AMT（亚马逊Mechanical Turk图像生命周期）

## 2.g. 转移学习以更新模型参数



转移学习图

（转移学习）给定源域DS和学习任务TS，目标域DT和学习任务TT，转移学习旨在使用DS和TS中的知识帮助改进DT中目标预测函数fT（·）的学习， 其中DS！= DT，或TS！= TT。 在上面的定义中，域是一对D = {X，P（X）}。 因此，条件DS！= DT表示XS！= XT或PS（X）！= PT（X）。

## 3. gPredict: 去中心化的梯度学习框架

为了提高AI算法的性能，我们需要提高精度/召回率，降低MSE。 在本节中，我们提出了一系列减少区块链上AI模型培训周转时间的方法。

**Estimator**: 只是一个简单的函数来模拟数据样本的一些有意义的特征。 例如,  其中x（i）是从分布中得出的随机变量 D, i.e. .

**Estimator Bias**衡量我们的估算器在估算实数方面有多好.

**Estimator Variance** 衡量我们的估算器对采样的“跳跃”程度. . **Bias-variance decomposition for estimators:**将这两个属性组合在一个公式中**。** 对S随机变量的期望值。

上述结论的正式数学证明:

## 3.a. 训练任务作为交易

contract TrainingTask {  
 TrainedModel previousModel;  
 function training();  
 TrainedModel trainedModel;  
}

## 3.b. 训练完备的模型作为资产

contract TrainedModel {  
 string[][] configuration;  
 double[] weights;  
 state[] optimizerState;  
}

## 3.c. 浮点映射

根据我们对低精度定点环境中有限数精度的研究，我们发现通过在使用随机梯度下降时仅使用16位表示来控制舍入方案，我们可以更快地训练模型而不会损害任何精度。

**Limited Precision Arithmetic 有限精度算术**: 基于反向传播的深度学习系统的标准执行通常利用实数的32位浮点表示来进行数据存储和控制。 我们通过表示[IN.FR]来抽象这个问题，其中IN和FR分别与数字的整数和小数部分相关。 数字位数（NB）加上小数位数（FB）产生用于表示整数的总位数。 NB + FB的总和称为字长WL。

**Stochastic rounding 随机四舍五入**: 给定数字x和目标定点表示 <NB, FB>. 我们定义  作为最大的整数倍  <= x. S随机舍入是一种无偏舍入方案，并具有预期的舍入误差为零的理想特性, i.e. E(Round(x, <NB,FB>)) = x

                    所以, P(rounding x to ) 相对于x的近似值 :

                    然后,

**Multiply and accumulate operation 乘以和累加运算**: 考虑两个二维的向量 **d** 和 **h** 这样每个向量都以固定的定点表示 <NB,FB>,  然后定义 **e** = d.h 作为d和h的内积空间. 然后我们可以将计算分成以下步骤:

## 3.d. 超参数路由

gPredict的活动是找到一个最接近的标量，可能是随机函数的最佳估计，而不是对该函数的可能参数的排列。 虽然许多捆绑包期望这些数据源是从向量空间中提取的，但gPredict的独特之处在于它促使开发人员更详细地描绘搜索空间。 通过提供有关函数表征位置的更多数据以及最佳质量的位置，gPredict可以更有效地搜索最佳参数。

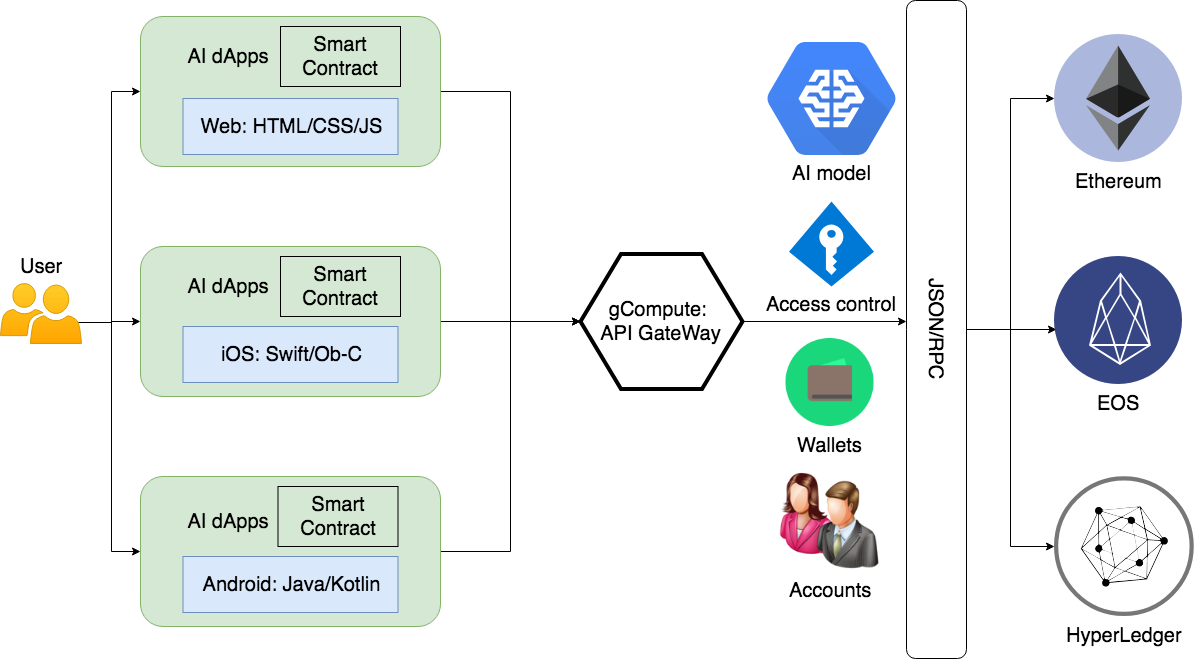
## 3.e. 结果回归

最后借助浮点映射和超参数路由，我们可以通过减少分布式系统的区块链类型来获得结果。

Reducer algorithm

class REDUCER  
 def REDUCE(string t, pairs [(s1,c1),(s2,c2)...])  
 sum <-- 0  
 cnt <-- 0  
 for all pair (s,c) belongs to pairs:  
 sum += s  
 cnt += c  
 rAvg = sum / cnt  
 EMIT(string t, integer rAvg)

## 4. gCompute: 互联网规模的AI dApps解决方案



gCompute API 网关

## 4.a. AI模型作为资产

与我们上面对训练完备模型的定义类似，我们可以将AI模型表示为区块链上的资产。

## 4.b. 可扩展性

**技术可扩展性**: 为了与谷歌，Facebook，阿里巴巴，腾讯等传统互联网巨头竞争，我们的底层平台区块链技术必须横向扩展。 如果我们有AI dApp注册并使用我们的平台轻松集成到所有现有的区块链后端，我们的系统能够处理数千万级别日常活跃用户。 对于这种类型的功能需求，我们将部署多个实例，将核心业务逻辑抽象为docker，并将负载均衡器放在我们的实例前面，以确保每个API / JSON调用顺利进行。

**商业模式的可扩展性**: 传统的互联网公司向我们证明，最好的商业模式是首先提供免费服务，然后提供各种收入模式（免费增值，高级版，订阅版）。 因此，我们的底层区块链技术平台应该具有类似的灵活性。

## 4.c. 可控制性

**不断升级**: 无论是大企业还是小型创业公司，甚至是个体开发者都应该能够灵活地不断地使用新功能升级他们的应用程序。 我们的平台将支持模型软件和智能合约升级。 即使谷歌Chrome，Facebook，腾讯微信，苹果iOS都受到不同的错误或系统故障，我们的平台和底层平台应该足够强大，可以修复错误并自动生成程序调用堆栈。

**持续集成**:  当开发人员为其现有软件添加新功能时，他们不想制动任何旧功能。 这就是持续集成解决方案的用武之地。如果某些提交破坏了旧功能（无法通过某些测试用例），代码托管平台将拒绝该提交并生成有关错误的报告。 我们的系统以及底层区块链平台也将支持这种良好实践。

**失败恢复**:  最糟糕的情况：如果我们遇到网络故障，系统故障，底层区块链平台DDOS攻击或任何用户密钥凭据被泄露，我们的平台以及底层区块链平台将具有故障恢复系统。 通过对不同版本的AI模型以及数据库状态进行版本控制，我们将能够为最终用户和开发人员提供此类功能。

## 4.d. 系统攻击难度

**PoW**:  对于公共区块链，PoW是最常采用的一致算法。 比特币是PoW的先驱，是区块链生态系统中最着名的共识算法。 PoW难以破解，需要超过50％的攻击才能破坏系统。 我们将能够安全地集成所有基于PoW的公共区块链，并在相应的区块链上完成大部分AI dApp业务逻辑。

**PoS**:  对于公共区块链，PoS / BFT-DPOS / DPOS是新的挑战者。 PoS被证明更加节能。 在这些算法下，在相应区块链平台上持有通证的用户能够选择块生成器并生成块。 我们的AI dApp平台将能够安全地与这些公共区块链进行通信，并实现我们所有的业务逻辑。

**SIEVE**:  对于获得许可的私有区块链，围绕共识算法有许多创新。 其中一个最有趣的是SIEVE。 简而言之，SIEVE通过将理论执行和检查阶段包括在内来扩展第一个PBFT计算：1）识别和筛选可能的非确定性请求并建立进入PBFT 3阶段断言协议的交换的确定性，以及2）启用协议 尽管经典PBFT提供的信息状态符合协议，但仍要继续运行验证器的收益条件。 SIEVE分别来自PBFT（由[Aublin et al。，TOCS’15]中描述的思想推动）通过重复使用PBFT看到改变惯例来降低其多方面的性质并且放弃执行另一个协议惯例的准备。通过性质 允许的私有区块链，我们的系统能够安全地连接这些区块链与实施的访问控制。

**Quantum resistance**:  后量子密码术，也称为量子安全密码术，可以经受量子PC的攻击。 鉴于特定算术领域中的麻烦问题，这种加密创新的改进采取更习惯的方式。 通过调查和创建计算，后量子安全加密创新可以在系统中连接，更重要的是，提供最异常的信息安全性。 我们期待看到这种算法的激动人心的用例在各种公共/私人区块链上付诸实践。

## 4.e. 可维护性

**可扩展性**:  现代软件的力量是建立在彼此之上的。 通过对第三方库和软件包的充满活力的支持，开发人员能够以更快的速度开发软件。 我们将提供一组库依赖关系管理工具，以允许开发人员轻松调用和管理已发布的智能合约或将其他合同作为库依赖项导入。

**可读性**:  我们将转换/解析给定区块链上的所有交易，并允许用户理解我们所有的系统AI dApps。 用户将能够看到谁创建AI智能合约，智能合约的主要交互，自动智能合约abi产生等。

**可探索性**:  我们应该能够探索整个区块链。 我们将提供有关区块链上最近开采的区块的见解。 我们还将提供已经开采并且当前附加到区块链网络的任何块中的任何交易的见解。 用户还可以查看任何公共地址的历史记录并审核余额，交易历史记录等。

## 4.f. 响应速度

**快速反应**: 卓越的客户体验需要可在几秒钟内获得即时反馈。 如果应用程序需要更长的响应时间，并且无响应的AI dApp导致竞争情况低于传统解决方案，则系统将会崩溃。 我们的gCompute平台将尽可能地优化以减少应用程序响应时间。

**网络优化**: 我们将尽量减少网络中的通信延迟。 网络优化应该有能力保证系统资源的理想使用，提高盈利能力以及社区的有效性。 网络优化会查看到服务器的各个工作站以及与之相关的工具和连接。 我们的系统可以包括流量整形，冗余数据消除，数据缓存和数据压缩以及数据协议的简化。

## 4.g. 高并发性

**同步执行**:  如果dApp本身需要顺序独立的步骤，我们也应该支持快速顺序执行。

**异步执行**:  即使计算机以顺序模式运行，并使用锁和上下文切换的组合来实现伪异步执行，现代应用程序也默认支持并发执行。 考虑到完整的互联网规模的AI dApp解决方案，我们将尽力优化我们的代码以实现异步执行。

# References