frontiers in Blockchain

TECHNOLOGY AND CODE
published: 20 May 2020
doi: 10.3389/fbloc.2020.00023

Hyperparameter Optimization Using Sustainable Proof of Work in Blockchain

*Anshul Mittal and Swati Aggarwal**

Department of Computer Engineering, Netaji Subhas University of Technology, New Delhi, India

Hyperparameters are pivotal for machine learning models. The success of efficient calibration, often surpasses the results obtained by devising new approaches. Traditionally, human intervention is required to tune the models, however, this obtuse outlook restricts the proficiency and competence. Automating this crucial characteristic of learning sustainably, proffers a significant boost in performance and cost optimization. Blockchain technology has revolutionized industries utilizing its Proof-of-Work algorithms for consensus. This complicated solution generates a lot of useless computations across the nodes attached to the network and thus, fritters away a huge amount of precious energy. In this paper, we propose to exploit these inane computations for training deep learning models instead of calculating purposeless hash values, thus, suggesting a new consensus schema. This work distinguishes itself from other related works by capitalizing on the parallel processing prospects it generates for hyperparameter tuning of complex deep learning models. We address this aspect through the framework of Bayesian optimization which is an effective methodology for the global optimization of functions with expensive evaluations. We call our work, Proof of Deep Learning with Hyperparameter Optimization (PoDLwHO).

Keywords: blockchain, hyperparameter tuning, proof of work, deep learning, sustainability, bayesian optimization

OPEN ACCESS

Edited by:
Sooyong Park,
Sogang University, South Korea

Reviewed by:
Zehui Xiong,
Nanyang Technological University, Singapore
Neeraj Kumar,
Thapar University, India
Stefanos Leonardos,
Singapore University of Technology and Design, Singapore

***Correspondence:**
Swati Aggarwal
swati1178@gmail.com;
swati.aggarwal@nsut.ac.in

Specialty section:
This article was submitted to Blockchain Technologies, a section of the journal Frontiers in Blockchain

Received: *05 October 2019*
Accepted: *23 April 2020*
Published: *20 May 2020*

Citation:
Mittal A and Aggarwal S (2020) Hyperparameter Optimization Using Sustainable Proof of Work in Blockchain. Front. Blockchain 3:23. doi: 10.3389/fbloc.2020.00023

INTRODUCTION

The accelerating ubiquity of machine learning has radically altered the way technology caters to human needs (LeCun et al., 2015). Deep learning networks have steered their way into human lives, complementing their capabilities, in the form of voice assistants (Hoy, 2018), targeted advertisements (Perlich et al., 2014), recommendation services (Linden et al., 2003) and life crucial applications like cancer prognosis (Kourou et al., 2015). However, the operational performance of these composite structures is still swayed by the parametric configurations set manually before initiating their training phase. The distinction between a substandard setting and a favorable hyperparameter calibration can be contrasted to an unsustainable design and a model with cutting-edge efficiency (Pinto et al., 2009; Kotthoff et al., 2017).

Numerous endeavors have been undertaken to optimize hyperparameters like racing algorithms (Maron and Moore, 1994), gradient search (Bengio, 2000) and random search (Bergstra and Bengio, 2012), each ameliorating the performance and operational efficiency. Bayesian optimization (BO), a sophisticated global optimization algorithm, has exhibited several interesting characteristics rendering it suitable for automated exploration in the myriad of hyperparameter choices. It has outperformed the traditional methods and is considereda huge step toward

automating machine learning (Snoek et al., 2012; Feurer et al., 2015). However, the BO based approach suffers from the issue of "cold start," necessitating an exploration from scratch for every new problem flung to it (Swersky et al., 2013). It operates by sequentially generating samples, evaluating the expensive black box function over them and building up its appropriation using the induced history. Thus, the conventional process involves a lot of ensuant computations and time. Few parallel alterations have been proposed which run multiple evaluations asynchronously (Ginsbourger et al., 2011; Chevalier and Ginsbourger, 2013; González et al., 2016), thereby, significantly decreasing the processing time. PoDLwHO aims to leverage on the parallel prospects of this algorithm by imputing the results of parallel runs in the experiment using a blockchain network. Thus, after every block generation run on the network, the subsequent sample generation phase will utilize the results obtained parallelly across the nodes, ergo surveying a large space in short time.

The last decade has witnessed emergence of a substantial number of blockchains (For more details regarding blockchain see Section Blockchain in **Appendix**). The concept of peer-to-peer distributed ledger and double spending was popularized by Satoshi Nakamoto through his revolutionary paper on Bitcoin (Nakamoto, 2008). These blockchains are based on the postulations of setting up an immutable digital system which would dismantle the orthodox record maintenance architecture of the current financial and industrial sectors and democratize it, thereby, empowering the users across the globe with total control and transparency over their transactions and records. The certitude of blockchain's acceptance is justified by observing the rapid rise in the number of transactions being performed over them and the value of cryptocurrencies accompanying them (Beck, 2018). However, the scale of energy consumed purposelessly for all this to happen is certainly a caveat (O'Dwyer and Malone, 2014; Giungato et al., 2017). The conventional democratic approach of blockchain consensus demands the participating nodes to solve a cryptographic and mathematical puzzle, like, unearthing a hash with certain length of zeroes, in case of Bitcoin (Nakamoto, 2008). This phenomenon comes under the umbrella of *proof-of-work* (PoW) scheme.

We call our work, Proof of Deep Learning with Hyperparameter Optimization (PoDLwHO). PoDLwHO aims to incorporate hyperparameter optimization using BO with the core democratic architecture of PoW scheme. Consequently, the purposeless hash computations will instead be used to train and tune the deep learning model parameters. The participating nodes in blockchain will no longer compete with each other to solve the puzzle and lay claim to mine the block, rather, they will be contending with each other for yielding better performing models. The miner of a block will be the node which reaches some set threshold accuracy earliest. For a provided architecture and dataset, the nodes will be tuning and training that architecture, competing as well as assisting each other (due to parallel Bayesian evaluation approach described in Hyperparameter Optimization Using Bayesian Constructs and Optimization Algorithms) for generating superior set of hyperparameters. There are two major contributions of this paper:

1) Modifying the PoW scheme to distribute and aggregate the BO evaluations over the nodes participating in blockchain network.
2) Introducing the concept of reward as a substitute of minting fresh currency.

The major challenge involved in designing such an architecture is ensuring the democratic behavior and security of monetary as well as data transactions, so that adversaries cannot introduce invalid models and fool the nodes into performing inessential labor. Also, the network should be secure so that the identity of a node is not impersonated just like in a normal blockchain network (Lin and Liao, 2017).

Section Related Works reflects on existing literature related to hyperparameter tuning and blockchain architectures. In Section Method, the proposed method is described, followed by analysis and comparison with some prominent algorithms in Section Analysis and Comparison, experiment in Section Experimental Setup, Technology and Code and results obtained in Section Experiment and Results. Section Discussion discusses the limitations and viable resolutions succeeded by future prospects in Section Future Prospects.

RELATED WORKS

Hyperparameter tuning is a crucial and challenging aspect of machine learning (Bergstra et al., 2011). It is evident from recent studies that calibrating the deep learning models has sprung the outcome statistics more than scouting new strategy (Pinto et al., 2009; Kotthoff et al., 2017). Several optimization techniques have been explored by researchers like racing algorithms (Maron and Moore, 1994), grid search (Larochelle et al., 2007), random search (Bergstra and Bengio, 2012) and gradient search (Bengio, 2000). Bayesian optimization (Mockus et al., 1978) is widely used in hyperparameter tuning (Snoek et al., 2012; Feurer et al., 2015; Kotthoff et al., 2017; Jin et al., 2019), substantially improving the parameter search (Jones et al., 1998; Brochu et al., 2010), surpassing the conventional semi-manual routine as well as assisting in assessing the hyperparameter importance (Hoos and Leyton-Brown, 2014). It employs gaussian process (Williams and Rasmussen, 2006) to model the underlying black-box function and also exercises a surrogate acquisition function to determine the next sampling point. However, the major hurdle with BO is the issue of "cold-start" requiring fresh initiation and generation of new samples for every problem it is given (Swersky et al., 2013). There have been attempts to resolve this concern (Ginsbourger et al., 2011; Janusevskis et al., 2012; Swersky et al., 2013; Kandasamy et al., 2018). Some variations have been proposed which run multiple evaluations asynchronously (Ginsbourger et al., 2011; Chevalier and Ginsbourger, 2013; González et al., 2016), thereby, significantly reducing the processing time. PoDLwHO uses the asynchronous evaluation of black-box function followed by subsequent aggregation of the

Abbreviations: PoDLwHO, Proof of Deep Learning with Hyperparameter Optimization (Proposed work); DL, Deep Learning; AI, Artificial Intelligence; BO, Bayesian optimization; PoW, Proof-of-Work.

computed results across the distributed network for efficiently appropriating the next sample points.

Blockchains have numerous computation and parallel processing prospects. *Proof-of-work* was first introduced in Bitcoin (Nakamoto, 2008) and is currently employed in a lot of crypto blockchains like Namecoin, Mazacoin, etc. PoW offers the supremacy of being fully decentralized, avoids any double-spend attack but also has the caveat of huge, practically purposeless, resource consumption. Primecoin (King, 2013), Gridcoin[1], Proof of Useful Work for Orthogonal vectors problem (Ball et al., 2017) incentivize participants for performing scientific analysis, however, the general public is still far from receiving a visible benefit from it. In Permacoin (Miller et al., 2014), Proof of Retrievability, is proposed requiring clients to invest their physical resources as well as computations, however, the several restrictions imposed, render it unsuitable.

Many researchers have attempted to integrate the concept of artificial intelligence and blockchain. Singularity Net[2] aims to create a decentralized, synergic commercial launchpad integrating AI and blockchain. Deep Brain Chain[3] provides a platform wherein several nodes ranging from individuals to big cloud vendors can dedicate physical systems for training AI models reducing the cost for accessing such equipment. However, these propositions are more involved in integrating smart contracts and machine learning, establishing a new commercial infrastructure and thus, do not address the core problem at hand. Proofware (Dong et al., 2019) promotes a crowd-based computing system allowing developers to build their decentralized applications (dApps) and their own consensus protocols by providing an effective and transparent financial system, building an elastic and autonomous large-scale computing system. However, their aim is to solicit developers with independent consensus protocols and create a interoperable seamless financial system.

Numerous implementations and strategies have been proposed for consensus (Bach et al., 2018). Peercoin (King and Nadal, 2012), introduces *proof-of-stake,* a new consensus method where the miners put up their coins at stake instead of solving some puzzle, thus, saving on electricity consumption and also reduces the risk of vertical centralization. However, there are several restrictions attached to it. It doesn't allow the supply-distribution control exercised in real world for market and inflation regulation. Also, there is a risk of double payment arising due to double fork. Most importantly, proof-of-stake is more aligned toward making the rich richer, as the protocol appraises the forger's holdings and stake.

PoDLwHO offers novelty in a way that while it has the advantages offered by the PoW schema, the repurposed consensus algorithm also advances the prospect of training deep learning models in a quick and economic manner, therefore, capitalizing its biggest shortcoming. Coin.AI (Baldominos and Saez, 2019) presented a similar theoretical hypothesis premised on generating complete model architectures from scratch by following a context-free grammar. Since the multitude of architecture hyperspace consists of only a fractional number of architectures feasible for a data set, this leads to several unwise structures. Their concept of choosing problem by voting within community is also inconvenient for a large network and can create clique of users with similar interests who won't allow any other problem domain to get voted in. They have mentioned a few alternatives for these approaches due to their limitation. PoDLwHO only aims to optimize the hyperparameters for a specific architecture and not generate a structure from scratch, which is a feasible problem and has several prior researches as described before (Chenli et al., 2019) worked on a similar concept of reutilizing the blockchain PoW. However, the hyperparameters used by the model trainers are random, which leads to redundancy and sacrifice of exploration performed by other miners. Also, their concept of full nodes, introduces a disparity among the new nodes to determine a suitable role and maintain their economic advantage.

METHOD

Blockchain network proposed in PoDLwHO is analogous to conventional systems. There are however, certain alterations concerning the functioning and runnability.

Main Entities and Their Function

The nodes in blockchain network are categorized as *Requestor, Miner*. Requestor is a semi-honest adversary who outsources the DL model and its accompanying dataset. Requestor can be researchers, scholars or small enterprises who want to train their models. The requestor attaches a reward for successful training of the model architecture provided by them. We assume that the Requestor is impartial in terms of providing hosted data set to all the miners. We also surmise that the requestor will not introduce a problem to deliberately obstruct the normal operations of blockchain and that their objective is to obtain appropriate model for their data. Miners are the participatory nodes which receive the model architecture and dataset, spawn the hyperparameter constructs, and train the model. Each of the miner is an accessory in training of each other's model, as the hyperparameters evaluated by the miners, during a block generation phase, are used collectively by each miner to appropriate their next sampling point (See Hyperparameter Optimization Using Bayesian Constructs and Optimization Algorithms). This routine allows traversal of the myriad of parameter space in short time, asynchronously.

Block Generation and Consensus Validation

The transaction block is the foundation of distributed ledger which validates the transactions performed over the network and thus, emplaces the entire financial system. The transactions validated by the miners are selected from the Transaction pool (See Section Transaction Pool in **Appendix**) consisting of the transactions performed by the blockchain users which haven't been added to the blockchain yet. This is similar to the concept

[1]https://gridcoin.us/assets/img/whitepaper.pdf.
[2]https://public.singularitynet.io/whitepaper.pdf.
[3]https://www.deepbrainchain.org/assets/pdf/DeepBrainChainWhitepaper_en.pdf.

introduced in Bitcoin (Nakamoto, 2008). The transaction block generation phase, which will be referred as block generation phase subsequently, in the PoDLwHO, commences when a requestor lodges a deep learning architecture and corresponding data set for training, along with the allotted reward and threshold accuracy (See **Figure 1**). The previous information coupled with identification information of requestor is known as a *Data transaction* (See Section Data Transaction in **Appendix** for further details). The dataset is hosted by the requestor and only its URL link is published, therefore, reducing the network load from hosting such large files. Threshold accuracy is the minimal desired result preferred by the requestor. Data transaction also holds the current Bayesian statistics which are pairs of hyperparameters and associated results, as well as the details of the current best model and its trainer. *In our experiment, we are only comparing the accuracy of the models over the test data set to determine the current best model, i.e., the model with highest accuracy is best model at that timeframe.* The data transaction is first validated using the digital signatures of the requestor before getting accepted into the system by nodes. Using web sockets, it is broadcasted to all the other nodes. Elliptic curves are used for begetting the digital signatures and their verification.

In case, processing of a data transaction from some other requestor is already underway, the current one will be pushed to the *Data transaction pool* (See Section Data Transaction Pool in **Appendix** for further details) and will subsequently be introduced to the network after the transactions prior to it have underwent their execution.

The transaction block is generated after a stipulated time frame which can be adjusted according to the network statistics, just like the dynamic length of zeroes for hash calculation in Bitcoin (Nakamoto, 2008). A data transaction can last multiple block generation phases as explained in Threshold not reached by any model. A data transaction consummates only if after some generation phases, the threshold accuracy is achieved by some model or if the data transaction reaches the preset expiration age.

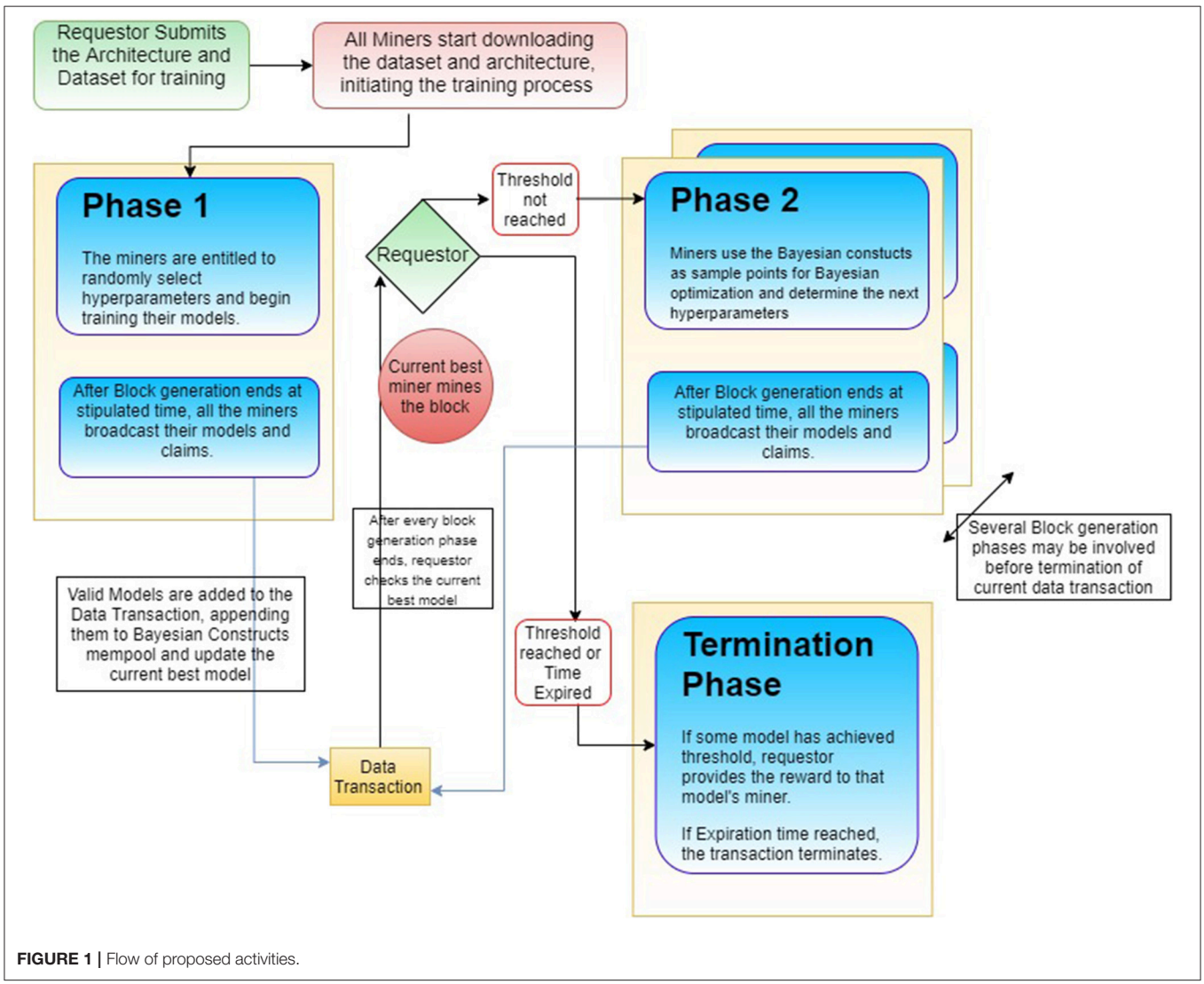

FIGURE 1 | Flow of proposed activities.

Hyperparameter Optimization Using Bayesian Constructs and Optimization Algorithms

Since the requestor only provides architecture and accompanying dataset, every node has the liberty to choose the hyperparameters on their own. *In our experiment we are considering only learning rate for optimization*.

Initially, when the requestor introduces the training dataset, the miners are unaware of the hyperparameters to be selected and their analogous performance. The miners thus, assay random hyperparameters within the acceptable domain range and churn up models, calculating their accuracy and loss results (Described as Phase 1 in **Figure 1**). These results are validated and updated in the data transaction. The updates are broadcasted to every other network node. Consequently, due to the large number of participating miners, a large parameter space will have been explored in an arbitrary fashion. When the stipulated time strikes, block generation phase ends. Now, the requestor may or may not accept the best model depending on the threshold it had set up. However, the miner of current best model mines the block without any reward. This way the blockchain propagates even if the data transaction never got completed. The process proceeds as follows depending on the whether the threshold was reached or not.

Threshold not reached by any model

Instead of disregarding the models other than the current best, the key-value pair of the hyperparameters used and corresponding accuracies achieved are stored, in a *Bayesian optimization construct mempool*. These will garner a large number of Bayesian constructs consisting of the evaluated samples. Now, using these samples, the miners apply BO and determine the next suitable set of hyperparameters that should be exercised in order to beget a better model, during the subsequent block generation phase (Described as Phase 2 in **Figure 1**). The acquisition function would require a trade-off between exploitation and exploration so that we are not always crowding a local maximum. This way the valid models (as defined in Warranting Validity of Models) generated by miners, unable to beat the current best model, assist in discovering the potential superior set of hyperparameters which may furnish an even better model.

Since, a single data transaction can span across multiple block generation cycles, the miners get an opportunity after every block generation to retrieve pairs of hyperparameters and corresponding accuracies from all the other miners in the networks and determine their next parameters. This helps in performing the BO asynchronously, diluting the time otherwise required for sequential execution.

However, this suffers from a limitation that a given set of hyperparameters are trained for only a very small amount of time. To counter this, instead of beginning afresh after every block generation phase, miners train upon their previous local models. Therefore, the weights realized by the model due to the successive training over the previous block generation phases, are stimulated for the new block generation phase using the newly enumerated hyperparameters. The model keeps on amending locally over a series of varying hyperparameters till consummation of data transaction. It may seem counterintuitive that a particular hyperparameter set is only utilized for training over a single block generation phase, which is a very short training period. It may yield optimal results if permitted several training passes, which is not the scenario here. However, due to the tradeoff between exploration and exploitation, this particular parameter set may reappear after some block generation phases and if it is consistent in enhancing the model's accuracy, the BO will be biased toward selecting its neighborhood, thereby, supporting the original claim.

Threshold reached by some model

The requester keeps inspecting the current best model statistics on cessation of every block generation phase. If a model reaches the desired threshold, the data transaction consummates with the assigned bounty rewarded to this model trainer and the block being mined by it (Described as Termination Phase in **Figure 1**). Requestor saves the architecture weights to their local system and thus, concludes the data transaction initiated by them. This terminated data transaction is popped out of the data transaction mempool and the subsequent data transaction is lined up for execution.

It may be reasonable to consider that the miner with better equipment will achieve a better model accuracy and consequently, always mine the block, rendering the nodes with subpar equipment fruitless. However, it is rational to assess that miners might not be successful in reaching the threshold accuracy at their first attempt, thereby, providing all the other nodes with the fruits of their exercise in form of hyperparameters used and corresponding results achieved. This adds to the parameter hyperspace reduction for all the participating nodes and allows them to seek a more favorable hyperparameter for their next execution cycle. Therefore, a node with budget hardware can use BO for undertaking training with calculated hyperparameters which puts up a better performance than a node with expensive gear but inferior hyperparameter. BO in PoDLwHO has a little random noise factor integrated in it which facilitates the miners to explore a neighborhood and not throng at a single point. A favorable hyperparameter calibration will lead to superior results (Pinto et al., 2009; Kotthoff et al., 2017) which can be achieved by any miner and the effect of substandard equipment will be reduced. Nevertheless, it will be irrational to ignore the performance advantages accompanying superior equipment.

Summary of Block Generation Activities

PoDLwHO inherits the block acceptance and generation policy similar to Bitcoin (Nakamoto, 2008), but substitutes the hash calculation with model training and introduces further refinement of hyperparameters to advance the operational accuracy.

Figure 1 demonstrates the flow of activities proposed. When the requestor submits the data set and architecture, all the nodes download them and begin the block generation phases. After every phase, the requestor checks whether the current best model has reached the set threshold limit or not. In case the threshold is reached, the data transaction ends there and the reward is awarded with the winner mining the block. The next

TABLE 1 | Theoretical overview of various algorithms used for hyperparameter tuning: Advantages and Challenges.

Algorithm	Underlying process	Brief overview	Parallel prospects	Challenges	Sustainability
Manual search	Intuition and experience.	Trivial and basic method. Utilizes the intuition and experience of experts.	Generated hyperparameters can be tested parallelly. across many processors.	Difficulty in reproducing results. Experience and intuition play a very crucial role.	Explicit resource consumption for tuning.
Grid search	Iterative computation and evaluation over a fixed domain range.	Simple to implement. Using computational loops, range of domains is evaluated removing the manual labor of experts involved.	Parallelization is trivial.	Suffers from the curse of dimensionality. Takes a very long time to process.	Explicit resource consumption for tuning.
Random search (Bergstra and Bengio, 2012)	Random hyperparameter sampling and evaluation from domain.	Utilizes the fact that function of interest is often more sensitive to certain dimensions, therefore, randomly samples the domain space in order to get optimum hyperparameters. No manual labor involved.	Easy to execute independent parallel runs, however, redundant repeated execution possible.	The sampled points are independent and random; thus, no statistical advantage of using the points already processed in determining the next batch.	Explicit resource consumption for tuning.
Sequential model based optimization (SMBO) (Snoek et al., 2012)	Surrogate model like GP and Acquisition functions like EI, MPI, UCB.	Heuristic approach is used to determine the next best set of points, thereby, leading to optimum results in a constrained and generative manner.	No parallel execution prospects.	The sequential process is time consuming and doesn't utilize the modern-day parallel processing prospects.	Explicit resource consumption for tuning.
Parallel evaluation of bayesian optimization	Surrogate model like GP and acquisition functions like EI, MPI, UCB.	The SMBO is performed parallelly by utilizing the multi-processor prospects for evaluating different points, parallelly distributing several steps involved.	Explicit execution of the parallel processors. Synchronizing wait for each processor to complete their set of execution before proceeding to the next evaluation cycle.	Explicit Execution and difficult synchronization of processors before proceeding to the next cycle of evaluation and discovery.	Explicit resource consumption for tuning.
q-EI Kriging metamodel (Ginsbourger et al., 2010, 2011)	Kriging believer, constant liar, gaussian process, monte carlo simulations.	Offers the parallel prospects of kriging, computational intelligence by using monte Carlo simulations. Using statistical algorithms, the sequential evaluations are reduced and expected values are employed.	Kriging is highly suitable for parallelization.	Maximizing q-EI becomes unaffordable as dimensions and number of points increases. Simulation is performed using monte carlo to resemble to expected results. Therefore, is not much accurate as compared to SMBO.	Explicit resource consumption for tuning.
Auto-WEKA (Thornton et al., 2013)	Tree structured parzen estimator and sequential model-based algorithm configuration.	Solves combined algorithm selection and hyperparameter optimization by BO techniques that iteratively build models of the algorithm/hyperparameter landscape and leverage these models to identify new points in the space that deserve investigation.	Parallelization is only used in executing parallel independent runs with random seed values and selecting the results with the lowest cross-validation error.	Parallelization is incorporated only to perform independent runs. The process itself is not distributed parallelly.	Explicit resource consumption for tuning.
Collaborative hyperparameter tuning (Bardenet et al., 2013)	For surrogate model - GP and Acquisition functions like EI, MPI, UCB; Transfer Learning technique	Exploits the ability of experienced practitioners at tuning by generalizing across similar learning problems. Knowledge from previous experiments is incorporated for new problems. A combination of surrogate-based ranking and optimization techniques is used for surrogate-based collaborative tuning. BO is used with the surrogate function.	No explicit information regarding parallelization is presented.	The evaluation metrics may not be comparable across different problems and datasets. A comprehensive database of problems is necessary. Feature construction is needed to define correlations between parameters.	Explicit resource consumption for tuning.

(Continued)

TABLE 1 | Continued

Algorithm	Underlying process	Brief overview	Parallel prospects	Challenges	Sustainability
Scalable bayesian optimization using deep neural networks (Snoek et al., 2015)	Uses neural networks to learn an adaptive set of basis functions for bayesian linear regression, deep networks for global optimization.	The GP traditionally used in BO is replaced with a model that not only scales linear fashion, but also retains most of the GP's desirable properties such as flexibility and well-calibrated uncertainty. Scales linearly in the number of observations, and cubically in the basis function dimensionality. Marginalizes over the possible outcomes of running experiments, thus, a set of fantasy outcomes is generated for each running experiment which is then used to augment the existing dataset.	Offers high degree of Parallelization.	Need to train a neural network for performing hyperparameter tuning. Performance is competitive with existing state-of-the-art approaches for BO.	Explicit resource consumption for tuning.
The parallel knowledge gradient method for batch bayesian optimization (q-KG) (Wu and Frazier, 2016)	For surrogate model - GP and for Acquisition function - KG.	q-KG consistently finds better function values than other parallel BO algorithms, such as parallel EI, batch UCB and parallel UCB with exploration. q-KG provides especially large value when function evaluations are noisy.	Offers high degree of Parallelization.	Computing q-KG and its gradient is very expensive. KG policy is more like a greedier approach. KG policy relies on the posterior distribution of the model for a guideline, in a sense, trusting and believing the model.	Explicit resource consumption for tuning.
PoDLwHO (Proposed Work)	For surrogate model - GP and for acquisition function - EI; blockchain	Repurposes the inconsequential computation cycles involved in the conventional proof-of-work schema of blockchain, thereby, utilizing the parallel prospects offered for not only tuning but also training the model architectures.	Offers high degree of Parallelization.	The BO algorithm used is rudimentary without exploiting the statistical modifications proposed by several researchers. Though, the robustness of architecture allows incorporation of any algorithm.	The architecture used is a sustainable surrogate to the otherwise, computation hungry consensus schema of blockchain. Provides viable utilization of electricity and computation resources. The process of blockchain consensus as well as hyperparameter tuning along with model training is accomplished, all through the same resource usage.

GP, Gaussian Process; EI, Expected Improvement; MPI, Maximum Probability of Improvement; UCB, Upper Confidence Bound; BO, Bayesian Optimization; KG, Knowledge Gradient.

data transaction is now put in process. However, if the threshold is not reached, the node with the current best model mines the block and the next block generation phase begins. Apart from the first block generation phase, where the hyperparameters are randomly sampled, all the other phases involve BO to determine the hyperparameters for training the model.

Warranting Validity of Models

The miners can try to deceive the networks by submitting models with exaggerated accuracies, therefore, any model must be validated before considering it either for best accuracy or even for adding it to Bayesian constructs mempool. The received model validation is transpired by downloading the model and performing a forward pass to match the claim. The dataset is already available with every miner. Forward passes are speedy and wouldn't consume a lot of clock time. Only if the accuracy asserts, the model is acknowledged.

Ensuring Authentic Inputs and Outputs

The authenticity of data transaction and its input data set is verified by using the signature of the requestor. The affirmation of trained model is also done in a similar way. Nodes always verify

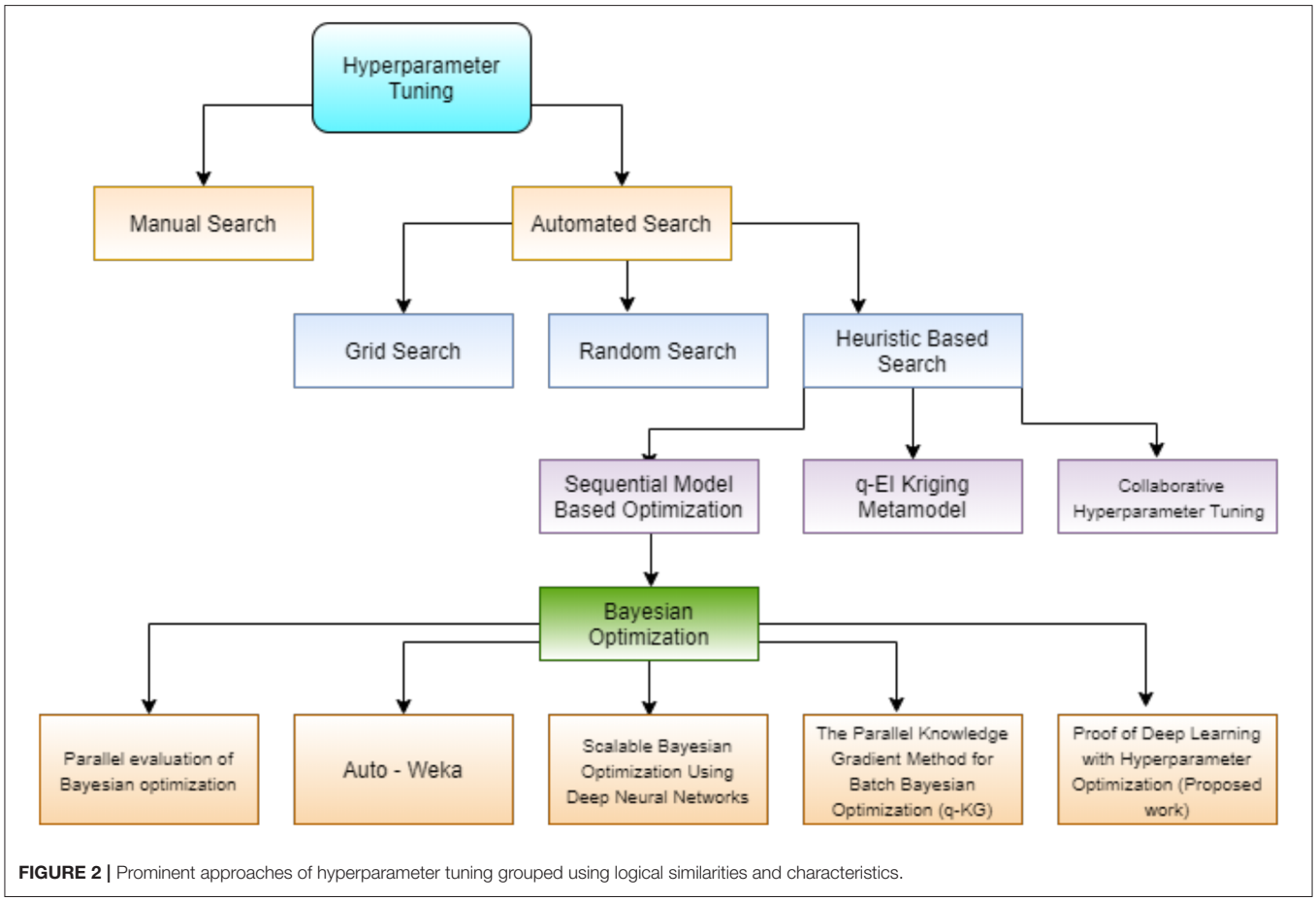

FIGURE 2 | Prominent approaches of hyperparameter tuning grouped using logical similarities and characteristics.

the model claims before accepting it (as described in Warranting Validity of Models). Every node saves their trained model locally and also hosts it via an accessible URL. They broadcast their trained model throughout the blockchain network along with the URL to access it, the hyperparameters used and their accuracy claim. Since the testing data set is the same for all miners in a block generation phase, the verifying nodes only need to download the model, they reuse the testing data set. Thus, any receiving node first downloads that model, compiles it using the hyperparameters and verifies the claim. Elliptic curves are used to beget digital signatures for each node as well as to authenticate the legitimacy of any transaction performed.

Setting Reward by the Requestor

The requesters don't have any advantage in colluding with the miners as they want the model that is optimal for their cause. However, miners have a certain profit attached if they are successfully able to connive with the requestor and retrieve the dataset in advance. If the miners are rewarded only by the blockchain system, then it may be profitable for them to apportion this reward with the requestor, access the dataset ahead of others, ergo get more time for drilling their model which should furnish superior results. However, in our proposition, the requestor is required to pay the reward attached to the model training and not the blockchain system. This corroborates a fair competition eliminating any collaboration between requestor and miner. It also dispenses with the requirement of minting a new cryptocurrency, permitting the network to simply use conventional currency as the medium of exchange in supporting the digital payments.

The rewards set up by the Requestor may be uneconomic for processing and thus, certain regulation is required for deciding the reward attached to a given data set.

Mining a New Block

The miner with the current best model mines the transaction block in a block generation phase. However, the data transaction currently in process may not conclude as a threshold accuracy is to be achieved by some model before committing the data transaction. Thus, the mining process sustains with the ongoing data transaction till its termination. The miner doesn't gain any incentive by mining this block. The only way miner can profit is by fostering a model which achieves the threshold accuracy set by the requestor. The limitation attached to this compensation strategy is discussed in Section Discussion and a viable alternate is also discussed.

The verification and linking of transaction blocks in the blockchain using hash of previous block, is performed in the

TABLE 2 | Comparison of various hyperparameter tuning algorithms over presence and absence of important characteristics.

Algorithm/process	Heuristic approach for determining hyperparameters	Requirement of intuition and experience	Parallel work distribution	Sustainability of resource utilization
Manual search	×	✓	×	×
Grid search	×	×	✓	×
Random search (Bergstra and Bengio, 2012)	×	×	✓	×
Sequential model based optimization (Snoek et al., 2012)	✓	×	×	×
Parallel evaluation of bayesian optimization	✓	×	✓	×
q-EI Kriging metamodel (Ginsbourger et al., 2010, 2011)	✓	×	✓	×
Auto-WEKA (Thornton et al., 2013)	✓	×	×	×
Collaborative hyperparameter tuning (Bardenet et al., 2013)	✓	×	–	×
Scalable bayesian optimization using deep neural networks (Snoek et al., 2015)	✓	×	✓	×
The parallel knowledge gradient method for batch bayesian optimization (q-KG) (Wu and Frazier, 2016)	✓	×	✓	×
PoDLwHO (proposed work)	✓	×	✓	✓

similar manner as described in Bitcoin (Nakamoto, 2008). However, the hash in PoDLwHO doesn't follow the PoW schema of Bitcoin with specific difficulty of prefix zeroes in hash generation, instead the hash is the SHA-256 hash of the block without any random nonce introduced. Instead, this block has the hyperparameters used, accuracy achieved by the miner (the trainer) of the corresponding data transaction in the block generation phase. Also, the transactions validated in a block are selected based on their time of arrival, and for current experiment, they do not entail any transaction fee associated with mining and validating them.

Security Threat Model

The blockchain technology has exhibited remarkable applications but it has also garnered unwanted attention of attackers and hackers raising concerns regarding the security and vulnerability (Li et al., 2017; Feng et al., 2019). The blockchain described under current scheme is also susceptible to 51% Attack, Double spending and private key security vulnerability (Li et al., 2017) and our experiment assumes the participating nodes to be honest and lawful preventing such attacks.

The requestor in PoDLwHO is vulnerable to a Distributed denial of service attack (DDoS) as all the miners are requesting and acquiring their data from the single requestor entity. We thus, assume the requestor to be capable of servicing all the requests without any prejudice or intended delay and function properly throughout the its operations. PoDLwHO is also exposed to the threat of code injection, since the data transactions exchanged between the nodes is followed by code execution for the purpose of model training and parameter evaluation. Suitable filters over the data exchanges are required to be implemented in order to prevent code injection.

ANALYSIS AND COMPARISON

Table 1 describes a brief overview of a few prominent algorithms proposed for performing hyperparameter tuning along with the advantages and challenges accompanying them. The algorithms have been stated in an order of priority to the concept of tuning models, building up on the fundamentals and enhancing them. **Figure 2** groups these algorithms using the similarities in their underlying characteristics and presents a broad outline of their resemblance and divergence. From the initial intuition and experience-based approach, the exercise of tuning, gradually moves toward a heuristic approach, automating the procedure and incorporating several statistical methods to assist in the task completion.

Table 2 sketches a contrast among these algorithms over certain parameters. Heuristic approach of determining hyperparameters refers to the automated and statistical approach of exploring the myriad of hyperparameter space and narrowing down to a few optimal samples, thereby, reducing the time involved in process. Since it is difficult even for experts to generate optimal hyperparameters, the Requirement of Intuition and Experience plays an important role in the practicability and adoption of approaches. Distributing the process parallelly not only helps in reducing the overall wall clock time but also provides justice to the modern-day arsenal of cheap and efficient parallel processors. Sustainability of resources is of utmost importance as the need for viable and environment friendly technology has amassed widespread attention due to the recent climatic changes leading to global warming and natural disasters.

It can be clearly observed from **Table 2** that other than PoDLwHO, no other process offers a foundation of sustainability with the advantages associated with statistical improvements. PoDLwHO essentially integrates the process of Parallel

evaluation of Bayesian optimization into the consensus schema of Blockchain and removes the challenge of synchronization by capitalizing on the block generation phases (as described in Block Generation and Consensus Validation) for achieving the same, and at the same time produces a viable and sustainable alternative. q-EI Kriging Metamodel (Ginsbourger et al., 2010, 2011) has the advantage of Kriging computation intelligence but the employment of Monte Carlo simulation renders it less accurate than the employed BO approach. Scalable Bayesian Optimization Using Deep Neural Networks (Snoek et al., 2015) requires training of a neural network for the process, which is a difficult task itself. The Parallel Knowledge Gradient Method for Batch Bayesian Optimization (q-KG) (Wu and Frazier, 2016) has good performance but the advantages of Expected Improvement over Knowledge gradient outweigh its precedence[4].

The comparison of characteristics among these algorithms as described in **Table 2** is only theoretical because wall clock comparison of experiments, and comparing PoDLwHO over BO benchmarks such as HPOLib package (Eggensperger et al., 2013) is unjustifiable as the novelty of PoDLwHO lies in the parallel architecture proposed which alters the PoW consensus mechanism of blockchain and not the rudimentary BO method used for Tuning. The robustness of architecture allows alteration to this optimization method, allowing ease in replacing it with more suitable surrogates. The future work would be directed toward finding a better optimization algorithm suitable to architecture operations.

EXPERIMENTAL SETUP, TECHNOLOGY, AND CODE

The experiment for setting up the blockchain and corresponding implementation of the PoDLwHO mechanism was performed on a laptop with Intel® Core(TM) i5-6200U CPU @ 2.30GHz × 4, 8 Gb RAM running on Ubuntu 18, a popular Linux distribution. The entire blockchain is custom built from scratch in NodeJS[5], a popular open-source JavaScript runtime and python. The deep learning models were implemented using TensorFlow[6], a very popular open-source machine learning library.

The code for the hypothesis proposed can be requested from the authors and will be provided on their discretion.

EXPERIMENT AND RESULTS

Data Set and Model Used

MNIST handwritten digit images data set (LeCun and Cortes, 1998) was used as the training and testing data set. A basic CNN was used as the architecture consisting of Conv2d, flatten, dense, dropout and dense layers. The current hyperparameter optimization is only targeting learning rate with its domains set as (0,0.001). Also, the epochs are fixed at 7 for each training phase.

[4]https://sigopt.com/blog/expected-improvement-vs-knowledge-gradient/.
[5]https://nodejs.org/en/.
[6]https://www.tensorflow.org/.

Analysis

Figure 3 shows the accuracy achieved by the model over the different learning rates which were proposed using the BO approach across the different nodes. It can be seen that the sample generated by BO is more concentrated around the learning rate of 0.001 from the figure.

Figures 4, **5** show the learning rate and accuracy achieved by the models across the span of 10 block generations for *Node 1*. It can be seen clearly that while the accuracy may not increase after every block generation, the BO algorithm sample the different prospective points that it deems fit. **Figure 3** shows that during the early stage when it tried sampling around 0.001 and later explored to 0.00078, on receiving an average response as compared to previous results, it pivots the sampling back to 0.001 as it had received the current best accuracy from that neighborhood. However, later during the block generation 9 stage, we can see that its samples around the learning rate of 0.0006 implying that it doesn't remain biased by what is current best neighborhood of sample points.

Figure 6, shows the current best accuracy as perceived by the requestor across the 10 generations of experiment. It can be

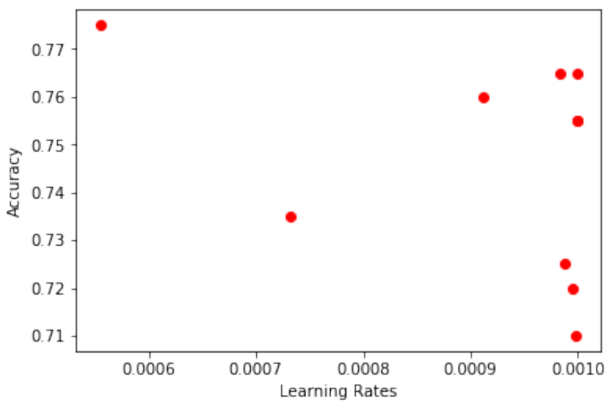

FIGURE 3 | Accuracy corresponding to learning rate samples.

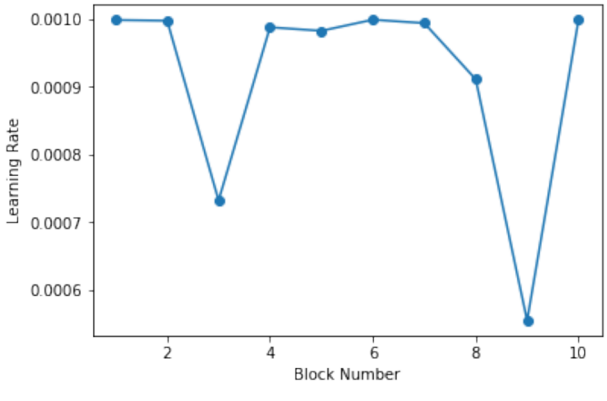

FIGURE 4 | Learning rate of the model for Node 1 for 10 block generations.

visually comprehended that the accuracy seems to increase after a few block generations phases.

Figure 7 shows the various domain points sampled across the block generation phases in the experimental run.

Thus, we can conclude that over the course of our experiment, we found that the accuracy of the best model received so far has improved after a period of 2-3 block generations. Also, another thing to note is that the sample points for hyperparameters set are not biased toward a local neighborhood, rather, the BO ensures a careful balance of exploration and exploitation. It is also worth considering that for a large of number of nodes several different set of hyperparameters will be under process in a block generation stage, not all nodes will have sampled on the same set of hyperparameters.

DISCUSSION

A given set of hyperparameters is trained for a fixed time frame, thus, it may steer the training toward a biased result. The hyperparameters processed later have the ascendancy of the model being already subjected to various training phases before.

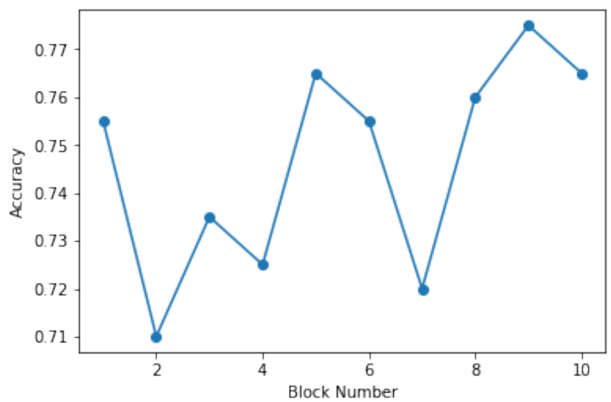

FIGURE 5 | Accuracy corresponding to the model of Node 1 for 10 block generations.

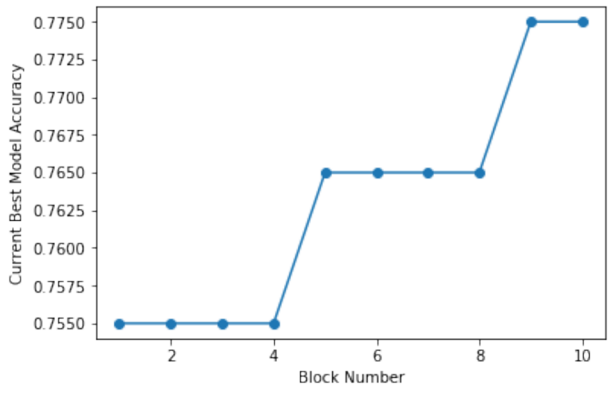

FIGURE 6 | Current best model accuracy across the block generations.

This can catalyze the hyperparameters sampled at later stage to exhibit superior findings. Even though the trade-off betwixt exploration and exploitation should account for this, yet there are some odds of bias crawling in.

Cumulating and processing all the Bayesian optimization constructs puts a heavy toll on the blockchain, therefore, the need to selectively transmit a certain proportion of these constructs to the next training phase arises. The selection must be conducted meticulously, preserving the tradeoff between exploration and exploitation. The requestor is presumed to be semi-honest, guaranteed to provide the incentive it had proffered. We can organize a smart-contract based system to circumvent this assumption. As mentioned in Setting Reward by the Requestor, certain regulation is required before requestor decides rewards for the data set as it may be uneconomical for the miners to process. Another interesting characteristic of the reward is that it is processed to the node whose model beats the threshold. However, this snubs the fact that, generation of such model undergoes a few block generation phases where different models may have been better. A suitable reward schema can be formulated where the bounty is administered across the blocks generated before consummation of the data transaction. The miners with the best models in each of these phases should be apportioned suitably from the given incentive.

An important aspect of PoW scheme is that the consensus schema should render a problem which is difficult to solve but quick to validate and verify. In PoDLwHO, the forward pass of deep learning model involved in the verification of claims by miner takes a few moments depending on the complexity of model and the size of data to be processed. Even though the data required for validation is already downloaded and available for evaluation, the forward pass takes time for performing computations and is not as swift as the hash validation involved in Bitcoin (Nakamoto, 2008). Thus, future work coordinated toward solving this issue is essential.

Even though, the blockchain is designed to minimize unwanted attacks and fraudsters, it is still susceptible to the assumptions as described in Security Threat Model and here. A robust model should work to evade such assumptions.

FUTURE PROSPECTS

While the training data is only downloaded once and reused, it institutes the requester as a performance bottleneck during the initiation phase of any data transaction, as, every miner is soliciting access to the dataset. We can contemplate a contrasting approach like, mini batch gradient descent, where instead of the entire data set, the requestor publishes only a limited set of random training and testing data after every phase. This unlocks a huge opportunity to study the incorporation of BO and mini-batch gradient descent. This way, miners procure a new training set every time and likelihood of performance improvement augments substantially.

Related works on asynchronous Bayesian optimization (Ginsbourger et al., 2011; González et al., 2016; Wang et al.,

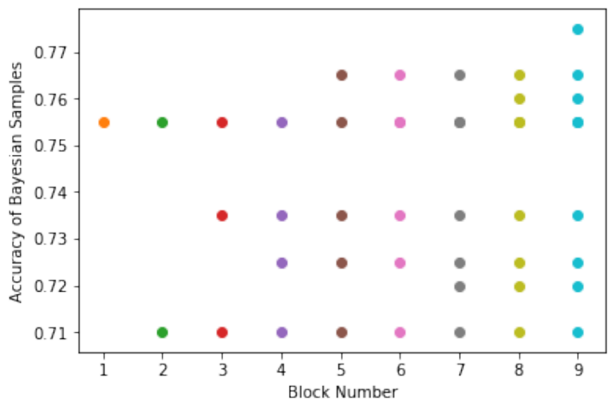

FIGURE 7 | Various samples for Node 1 across the block generations.

2016) can also be incorporated to further improve the processing and optimization.

CONCLUSION

We presented a novel proof-of-work consensus mechanism, Proof of Deep Learning with Hyperparameter Optimization, PoDLwHO, which assists in hyperparameter tuning of deep learning models. It not only repurposes the inconsequential computation cycles involved in the conventional proof-of-work schema, but also, furnishes the latitude of parallel processing prospects accompanying it. PoDLwHO institutes a mechanism where the participating nodes train deep learning models as their proof-of-work, ergo they compete with each other to yield models with superior performance. Concurrently, the proposed models are persistently refined by performing hyperparameter tuning using the approach of Bayesian optimization. By asynchronously evaluating the myriad of hyperparameter space, numerous parameter sets can be evaluated parallelly in a short time, saving sequential computation steps and simultaneously, allowing the multitude of nodes to compete and assist each other.

These deep learning architectures can be solicited by academia, researchers and budding startups that desire affordable training for their projects. Toll of training machine learning models ranges from expensive end, where the pricey equipment is purchased, to, comparatively low yet still costly, cloud rental stage. From prototyping stage to pilot phase, and subsequent maintenance and retrain cycles, the rising expense discourages people from assessing their postulates. PoDLwHO aims to parry such exorbitant costs by surrogating as a low-cost alternative. The economic benefit emerges as the computations which were otherwise generating impractical hash, like in Bitcoin, are now vindicated by training the compute extensive deep learning models. We also outlined the prospects of introducing a blockchain without minting a new cryptocurrency. Interestingly, it is also anticipated that the motivation of increasing accompanying rewards will encourage users to innovate and come up with new and compelling approaches for training the models in a better and efficient way, igniting more research in this domain.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found here: http://yann.lecun.com/exdb/mnist/.

AUTHOR CONTRIBUTIONS

AM and SA contributed to design, conceptualization of the study, to writing, reviewing, and editing of manuscript. AM performed analysis and investigation. SA provided supervision.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fbloc.2020.00023/full#supplementary-material

REFERENCES

Bach, L. M., Mihaljevic, B., and Zagar, M. (2018). "Comparative analysis of blockchain consensus algorithms," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (Opatija), 1545–1550. doi: 10.23919/MIPRO.2018.8400278

Baldominos, A., and Saez, Y. (2019). Coin.AI: a proof-of-useful-work scheme for blockchain-based distributed deep learning. *Entropy* 21:723. doi: 10.3390/e21080723

Ball, M., Rosen, A., Sabin, M., Vasudevan, P. N. (2017). *Proofs of Useful Work*. Cryptology ePrint Archive, Report 2017/203. Available online at: https://eprint.iacr.org/2017/203

Bardenet, R., Brendel, M., Kégl, B., and Sebag, M. (2013). "Collaborative hyperparameter tuning," in *International Conference on Machine Learning* (Atlanta), 199–207.

Beck, R. (2018). Beyond bitcoin: the rise of blockchain world. *Computer* 51, 54–58. doi: 10.1109/MC.2018.1451660

Bengio, Y. (2000). Gradient-based optimization of hyperparameters. *Neural Comput.* 12, 1889–1900. doi: 10.1162/089976600300015187

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). "Algorithms for hyper-parameter optimization," in *25th Annual Conference on Neural Information Processing Systems (NIPS 2011)*, eds J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger (Granada: HAL) 2546–2554.

Bergstra, J., and Bengio, Y. (2012). Random search for hyper-parameter optimization. *J. Mac. Learn. Res.* 13, 281–305. doi: 10.5555/2188385.2188395

Brochu, E., Cora, V. M., and De Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv [preprint]* arXiv:1012.2599, 23–40.

Chenli, C., Li, B., Shi, Y., and Jung, T. (2019). "Energy-recycling blockchain with proof-of-deep-learning," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (Seoul: IEEE), (19–23).

Chevalier, C., and Ginsbourger, D. (2013). "Fast computation of the multi-points expected improvement with applications in batch selection," in *International Conference on Learning and Intelligent Optimization* (Berlin: Springer), 59–69. doi: 10.1007/978-3-642-44973-4_7

Dong, Z., Lee, Y. C., and Zomaya, A. Y. (2019). Proofware: proof of useful work blockchain consensus protocol for decentralized applications. *arXiv [preprint]* arXiv:1903.09276.

Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., et al. (2013). "Towards an empirical foundation for assessing bayesian optimization

of hyperparameters," in *Proceedings of the NIPS Workshop on Bayesian Optimization in Theory and Practice*, 5.

Feng, Q., He, D., Zeadally, S., Khan, M. K., and Kumar, N. (2019). A survey on privacy protection in blockchain system. *J. Netw. Comput. Appl.* 126, 45–58. doi: 10.1016/j.jnca.2018.10.020

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). "Efficient and robust automated machine learning," in *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems - Vol. 2*, eds C. Cortes, D. D. Lee, M. Sugiyama, and R. Garnett (Cambridge, MA), 2755–2763. Available online at: https://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning

Ginsbourger, D., Le Riche, R., and Carraro, L. (2010). "Kriging is well-suited to parallelize optimization," in *Computational Intelligence in Expensive Optimization Problems*, eds Y. Tenne, C. K. Goh (Berlin: Springer), 131–162. doi: 10.1007/978-3-642-10701-6_6

Ginsbourger, D., Janusevskis, J., and Le Riche, R. (2011). *Dealing with Asynchronicity in Parallel Gaussian Process Based Global Optimization*. Available online at: https://hal.archives-ouvertes.fr/hal-00507632

Giungato, P., Rana, R., Tarabella, A., and Tricase, C. (2017). Current trends in sustainability of bitcoins and related blockchain technology. *Sustainability* 9:2214. doi: 10.3390/su9122214

González, J., Dai, Z., Hennig, P., and Lawrence, N. (2016). "Batch bayesian optimization via local penalization," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS), Vol. 51, JMLR Workshop and Conference Proceedings*, eds A. Gretton and C. C. Robert (Cadiz), 648–657. Available online at: https://ps.is.tuebingen.mpg.de/publications/gondaihenlaw16

Hoos, H., and Leyton-Brown, K. (2014). "An efficient approach for assessing hyperparameter importance," in *International Conference on Machine Learning* 754–762.

Hoy, M. B. (2018). Alexa, Siri, Cortana, and more: an introduction to voice assistants. *Med. Ref. Serv. Q.* 37, 81–88. doi: 10.1080/02763869.2018.1404391

Janusevskis, J., Le Riche, R., Ginsbourger, D., and Girdziusas, R. (2012). "Expected improvements for the asynchronous parallel global optimization of expensive functions: potentials and challenges," in *International Conference on Learning and Intelligent Optimization* (Berlin: Springer), 413–418. doi: 10.1007/978-3-642-34413-8_37

Jin, H., Song, Q., and Hu, X. (2019). "Auto-keras: an efficient neural architecture search system," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK: ACM), 1946–1956. doi: 10.1145/3292500.3330648

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *J. Glob. Optim.* 13, 455–492. doi: 10.1023/A:1008306431147

Kandasamy, K., Krishnamurthy, A., Schneider, J., and Póczos, B. (2018). "Parallelised bayesian optimisation via thompson sampling," in *International Conference on Artificial Intelligence and Statistics* (Lanzarote), 133–142.

King, S. (2013). *Primecoin: Cryptocurrency with Prime Number Proof-of-Work*. Available online at: http://primecoin.org/

King, S., and Nadal, S. (2012). "Ppcoin: peer-to-peer crypto-currency with proof-of-stake," in *Self-Published paper*. Available online at: https://decred.org/research/king2012.pdf

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2017). Auto-WEKA 2.0: automatic model selection and hyperparameter optimization in WEKA. *J. Mac. Learn. Res.* 18, 826–830. doi: 10.5555/3122009.3122034

Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., and Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. *Comput. Struct. Biotechnol. J.* 13, 8–17. doi: 10.1016/j.csbj.2014.11.005

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of the 24th International Conference on Machine Learning* (Corvallis, OR: ACM), 473–480. doi: 10.1145/1273496.1273556

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature* 521, 436–444. doi: 10.1038/nature14539

LeCun, Y., and Cortes, C. (1998). *The MNIST Database of Handwritten Digits*. Available online at: http://yann.lecun.com/exdb/mnist/

Li, X., Jiang, P., Chen, T., Luo, X., and Wen, Q. (2017). A survey on the security of blockchain systems. *Future Gen. Comput. Syst.* 170, 841–853. doi: 10.1016/j.future.2017.08.020

Lin, I. C., and Liao, T. C. (2017). A survey of blockchain security issues and challenges. *IJ Netw. Security* 19, 653–659.

Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Comput.* 7, 76–80. doi: 10.1109/MIC.2003.1167344

Maron, O., and Moore, A. W. (1994). "Hoeffding races: accelerating model selection search for classification and function approximation," in *NIPS'93: Proceedings of the 6th International Conference on Neural Information Processing Systems 1993*, eds J. D. Cowan, G. J. Tesauro, and J. Alspector (San Francisco, CA: Morgan Kaufmann Publishers Inc.), 59–66. Available online at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.5490

Miller, A., Juels, A., Shi, E., Parno, B., and Katz, J. (2014). "Permacoin: repurposing bitcoin work for data preservation," in *2014 IEEE Symposium on Security and Privacy* (Washington, DC: IEEE), 475–490. doi: 10.1109/SP.2014.37

Mockus, J., Tiesis, V., and Zilinskas, A. (1978). The application of Bayesian methods for seeking the extremum. *Towards Glob. Optim.* 2, 117–129.

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Available online at: http://bitcoin.org/bitcoin.pdf (accessed September 10, 2019).

O'Dwyer, K. J., and Malone, D. (2014). *Bitcoin mining and Its Energy Footprint* (Limerick: IET).

Perlich, C., Dalessandro, B., Raeder, T., Stitelman, O., and Provost, F. (2014). Machine learning for targeted display advertising: transfer learning in action. *Mac. Learn.* 95, 103–127. doi: 10.1007/s10994-013-5375-2

Pinto, N., Doukhan, D., DiCarlo, J. J., and Cox, D. D. (2009). A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Comput. Biol.* 5:e1000579. doi: 10.1371/journal.pcbi.1000579

Snoek, J., Larochelle, H., and Adams, R. P. (2012). "Practical Bayesian optimization of machine learning algorithms" in *NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems - Vol. 2*, eds F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Red Hook, NY: Curran Associates Inc). 2951–2959.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., et al. (2015). "Scalable bayesian optimization using deep neural networks," in *International Conference on Machine Learning* (JMLR.org), 2171–2180.

Swersky K., Snoek, J., and Adams, R. P. (2013). "Multi-task Bayesian optimization," in *NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems - Vol. 2*, eds C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Red Hook, NY: Curran Associates Inc). 2004–2012.

Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2013). "Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Chicago, IL: ACM), 847–855. doi: 10.1145/2487575.2487629

Wang, J., Clark, S. C., Liu, E., and Frazier, P. I. (2016). Parallel bayesian global optimization of expensive functions. *arXiv [preprint]* arXiv:1602.05149.

Williams, C. K., and Rasmussen, C. E. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Wu, J., and Frazier, P. I. (2016). "The parallel knowledge gradient method for batch Bayesian optimization," in *NIPS'16: Proceedings of the 30th International Conference on Neural Information Processing Systems 2016*, eds D. D. Lee, U. Von Luxburg, R. Garnett, M. Sugiyama, and I. M. Guyon (Red Hook, NY: Curran Associates Inc). 3126–3134.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.