IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 5, 2021, accepted August 23, 2021, date of publication September 10, 2021, date of current version September 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3111898

Bayesian Graph Convolutional Neural Networks via Tempered MCMC

ROHITASH CHANDRA[1], (Senior Member, IEEE), AYUSH BHAGAT[2], MANAVENDRA MAHARANA[2], AND PAVEL N. KRIVITSKY[1]

[1]School of Mathematics and Statistics, UNSW Sydney, Kensington, NSW 2052, Australia
[2]Department of Computer Science Engineering, Manipal Institute of Technology, Manipal, Karnataka 576104, India

Corresponding author: Rohitash Chandra (rohitash.chandra@unsw.edu.au)

ABSTRACT Deep learning models, such as convolutional neural networks, have long been applied to image and multi-media tasks, particularly those with structured data. More recently, there has been more attention to unstructured data that can be represented via graphs. These types of data are often found in health and medicine, social networks, and research data repositories. Graph convolutional neural networks have recently gained attention in the field of deep learning that takes advantage of graph-based data representation with automatic feature extraction via convolutions. Given the popularity of these methods in a wide range of applications, robust uncertainty quantification is vital. This remains a challenge for large models and unstructured datasets. Bayesian inference provides a principled approach to uncertainty quantification of model parameters for deep learning models. Although Bayesian inference has been used extensively elsewhere, its application to deep learning remains limited due to the computational requirements of the Markov Chain Monte Carlo (MCMC) methods. Recent advances in parallel computing and advanced proposal schemes in MCMC sampling methods has opened the path for Bayesian deep learning. In this paper, we present Bayesian graph convolutional neural networks that employ tempered MCMC sampling with Langevin-gradient proposal distribution implemented via parallel computing. Our results show that the proposed method can provide accuracy similar to advanced optimisers while providing uncertainty quantification for key benchmark problems.

INDEX TERMS Bayesian neural networks, MCMC, Langevin dynamics, Bayesian deep learning, graph neural networks.

I. INTRODUCTION

Graph neural networks are a type of artificial neural network designed for data which features graph-based representation [1]–[4]. Graph-based representation can be used to analyse non-structured and non-sequential data, such as a social network comprising users and their activities [5]. Recently, a wide variety of graph-based deep learning network architectures has been introduced [3], such as graph convolutional neural networks (CNNs) [6]–[8], graph recurrent neural networks featuring long short-term memory (LSTM) networks [9]–[12], graph auto-encoders [13], [14], and graph generative adversarial networks (GANs) [15]. Applications of graph neural networks have included time series forecasting [16], traffic flow forecasting [17], [18], particle physics [19], molecular property prediction [20], sentiment analysis [21], recommender systems [22], and social media popularity prediction [23]. A review of applications of graph neural networks has been given in [24].

Deep learning methods, such as convolution neural networks [25], [26] (CNNs) and recurrent neural networks [27], [28] (RNNs) have been applied to image data and temporal sequences; however, these are structured, regular, Euclidean data, although they can be viewed as graphs (i.e., lattices). CNNs and RNNs are less applicable to unstructured or graph-based data with multi-layer hierarchical structure, with features that occur on different scales. On the other hand, graph neural networks (GNNs) use graph-based representation of data to propagate on each node. Aspects of the data such as the input order of the nodes are irrelevant, with the graph instead representing the dependencies between them; hence, GNNs can enable propagation guided by graph structure similar to the forward propagation in simple (canonical) neural networks [29], [30].

As the impact of graph neural networks on different deep learning architectures and applications grows, there is also a growing need for robust uncertainty quantification

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Ali.

in model parameters. Bayesian inference provides a means for robust uncertainty quantification in deep learning models by sampling from the posterior distribution that represents the model parameters [31], [32] using Markov-Chain Monte Carlo (MCMC) methods [33], [34]. Implementation of Bayesian inference via MCMC becomes very challenging with growing size of the data and the number of model parameters. MCMC methods have extensive computational requirements since thousands of samples (iterations) are needed for training, and hence limited work has been done in implementing deep learning models such as Bayesian CNNs via MCMC sampling. Variational inference, provides an alternative for uncertainty quantification in deep learning methods via *Bayes by backpropagation* [35]. At the same time, there has been progress in the MCMC approach, making use of parallel computing and advanced MCMC methods [33], [34], which can enable the framework for graph CNNs. Advanced proposal distribution in MCMC that incorporate gradients [32], [34], [36] has opened the door to Bayesian deep learning methods for novel deep learning methods.

In this paper, we present Bayesian graph convolutional neural networks that employs tempered MCMC sampling via parallel computing with Langevin-gradient proposal distribution. We apply the method to selected benchmark graph-based datasets obtained from research data repositories such as *PubMed*.[1] The parallel computing framework features inter-process communication for exchange of tempered MCMC replica states as demonstrated previously for Bayesian neural networks [34].

The remainder of the paper is organised as follows. In Section II, we review the background of the problem and related methods. Section III presents the proposed methodology, followed by experiments and results in Section IV. We discuss the implications of our work and directions of future work in Section V, and Section VI concludes the paper.

II. BACKGROUND AND RELATED WORK

A. GRAPH NEURAL NETWORKS

A graph G data structure consists of a set of vertices V (nodes) and edges E, which can be either directed or undirected [37], [38]. Each node represents a data element, and the edges denote the relationships between the data elements. Each node has its own graph embedding via a feature vector, which summarises the properties of that particular data element. The nodes send their graph embedding to their immediate neighbours in the form of messages [39].

The message received by node v at GNN layer index t, m_v^t is constructed by aggregating over the set of neighbours v, $N(v)$, the results of a message function M_t which takes three arguments: the feature vector of v itself (h_v^t), the feature vector of neighbour w (h_w^t), and the features of the edge between v and w (e_{vw}):

$$m_v^t = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}). \tag{1}$$

Based on this message m_v^t and the previous value h_v^t, the latter is updated via a function U_t:

$$h_v^{t+1} = U_t\,(h_v^t, m_v^t). \tag{2}$$

A variety of graph representations are possible, including directed graphs, heterogeneous graphs, edge information graphs, and dynamic graphs [24]. The goal of GNN is to learn a state embedding which incorporates the neighbourhood information of for each node; this state embedding can then be used for classification and other purposes. The choice of the propagation and the output step of a GNN are required to obtain the hidden states of nodes or edges and depend on the application. In relation to the canonical GNNs, the focus has been on refinements in the propagation step, while a simple feedforward neural network is retained in the output step. The major variants in the propagation step utilise different aggregators to gather information from each node's neighbors. Some of the key propagation step methods include attention aggregator (graph attention network [40] and gated attention network [41]), gated aggregator (gated graph neural networks [42] and graph LSTM [10]), skip connection (highway GNN [43] and jump-knowledge network [44]), hierarchical graph edge conditioned convolution [45]), and finally, the convolutional aggregator which features graph CNN with spatial [46] and spectral methods [30].

Spatial methods in graph CNNs include neural fingerprints (FPs) [47], dual graph convolution network (DGCN) [48], and model networks (MoNet) [49]. Some of the commonly used spectral methods in graph CNNs are ChebNet (Chebyshev polynomial approximation algorithm) [50], graph convolutional networks (GCNs) which is first-order approximation of graph convolutions [30], and adaptive graph convolutional networks (AGCN) [51]. In spectral convolution, the underlying structure of the graph is deduced by eigen-decomposition of the graph laplacian. The entire graph is processed simultaneously, which makes spectral convolution more computationally expensive. However, it is still widely used, because the spectral filters excel at capturing complex patterns. Spatial methods use information from neighbouring nodes and deduce properties of a node based on features of its closest k neighbours. The graph can be processed in batches of nodes, which help improve speed and efficiency.

A number of approaches can be used to update or training GNNs. They include neighbourhood sampling (Graph SAGE [52], Pin-Sage [53] and Fast-GCN [54]), receptive field control [55], data augmentation such as co-training and self-training [6], and unsupervised training such as graph autoencoder (GAE) [56] and adversarially regularized graph autoencoder (ARGA) [14].

B. BAYESIAN DEEP LEARNING

Research in area of Bayesian deep learning has been limited due to the limitations of canonical MCMC methods for large number of parameters, and other characteristics such as complex architectural properties of deep learning mod-

[1] https://pubmed.ncbi.nlm.nih.gov/

els [57], [58]. As we noted earlier, variational inference provides a computationally cheaper approach, with variational autoencoders [59], variational autoencoders and GANs [60], pruned variational CNNs [61]–[63], variational RNNs and long short-term memory (LSTM) networks [64], recurrent variational graph convolutions [65], and variational graph neural networks with focus on autoencoders [56] and Markov networks [66]. Most of these methods were developed in the last five years, particularly after 2017, with applications summarised in [57]. Although work has been done in area of variational CNNs [67], we did not find any work in the area of variational graph-CNNs.

Over the last decade, advanced proposal distributions incorporating gradients have been applied, such as Langevin and Hamiltonian MCMC for statistical models [32], [68]. However, only in last five years has there been progress in area of Bayesian neural networks with Hamiltonian MCMC [69], and graphic processing unit (GPU) implementation to enhance computation [70]. Langevin MCMC methods for neural networks include the use of tempered (parallel tempering) MCMC for simple neural networks applied to pattern classification and time series prediction problems [34]. Furthermore, surrogate assisted estimation via Langevin tempered MCMC has been developed for Bayesian neural networks which is useful when the model and data are computationally expensive [33]. Transfer learning has been used to take advantage of multiple sources of data in a Bayesian framework via Langevin MCMC sampling [36]. Although simple neural networks have been used in pattern classification problems [33], [36], some of the problems had large numbers of features, and hence the neural network models had more than 5,000 parameters, which is comparable to smaller deep learning models.

III. METHODOLOGY

In this section, we present details for Bayesian graph convolutional neural networks, which uses spectral convolution and tempered MCMC sampling framework with Langevin-gradient proposal distribution. The framework is used for classification of nodes in datasets with graph representation.

A. MODEL AND LIKELIHOOD FUNCTION

In conventional CNNs, the input data are multiplied by a matrix of weights having the same dimension as the input data. This matrix of weights is known as the *filter* or *kernel*. A single layer can have multiple filters to extract different features in the data. The output can then be fed into more convolution layers or pooling layers to extract the prominent features of the data. Given a CNN with multiple layers, the output Y_i^l for layer l with m_1 filters is computed as follows

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} K_{i,j}^{(l)} X_i^{(l-1)} \tag{3}$$

where, $B_i^{(l)}$ is the bias matrix, $X_i^{(l-1)}$ is the input data to the layer, and $K_{i,j}^{(l)}$ is the filter.

In comparison to conventional neural networks, graph neural networks do not operate on euclidean data, with a fixed dimension and a tabular structure. Graph information such as the edge directionality, node attributes, and edge attributes cannot necessarily be mapped to a higher dimension euclidean space. In our proposed Bayesian graph CNN (Bayes-GCNN), we use the fast approximate spectral graph convolution technique of Kipf and Welling [30]. We adapt Equation (3) such that it can work on graphs, where the number of node connections is dynamic and the nodes are unordered. Hence, the equation to compute the convolved signal matrix is given as follows

$$Z = D^{(-1/2)} A D^{(-1/2)} X \theta \tag{4}$$

where Z is the convolved signal matrix, D is the degree matrix, A is the adjacency matrix, X is the node feature vector and θ is a matrix of filter parameters.

Our focus in this paper is the application of Bayes-CNN to node classification using graph datasets. Therefore, we construct the likelihood function that is used for MCMC sampling of parameters (weights and biases) in Bayes-GCNN. The likelihood function enables the comparison of the training data featuring graph representation $\mathbf{g}$ with graph CNN output $\mathbf{g}' = z(\mathbf{g}, w, b)$; where, w and b are the Bayes-GCNN parameters (weights and biases combined) shown in Figure 1, respectively. These weights and biases include those in convolution and max-pooling layers of graph CNNs.

Pattern classification problems entail discrete outcomes, and thus we use a multinomial likelihood function. Suppose that we have K classes in the training data, and assume that the outputs $\boldsymbol{y} = (y_1, \ldots, y_N)$ are drawn from a multinomial distribution with parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)$, for $\sum_{k=1}^K \theta_k = 1$. We define indicator variables

$$z_{i,k} = \begin{cases} 1, & \text{if } y_i = k \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

for observations $i = 1, \ldots, N$ and classes $k = 1, \ldots, K$. Then, the multinomial likelihood function can be expressed as

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \prod_{i=1}^{N} \prod_{k=1}^{K} \theta_{i,k}^{z_{i,k}}, \tag{6}$$

where $\theta_{i,k}$ is the Bayes-GCNN model's predicted probability that observation i is in class k. We use a multinomial expit (softmax) function to link the output $f(x_i)$ (of inputs x_i) of the Bayes-GCNN to the predicted probability:

$$\theta_{i,k} = \frac{\exp(f_k(x_i))}{\sum_{j=1}^{K} \exp(f_j(x_i))}. \tag{7}$$

The prior distribution is given by

$$p(\boldsymbol{\theta}) = \frac{1}{(2\pi\sigma^2)^{P/2}} \times \exp\left\{ -\frac{1}{2\sigma^2} \left(\sum_{i=1}^{P} \theta_i^2 \right) \right\} \tag{8}$$

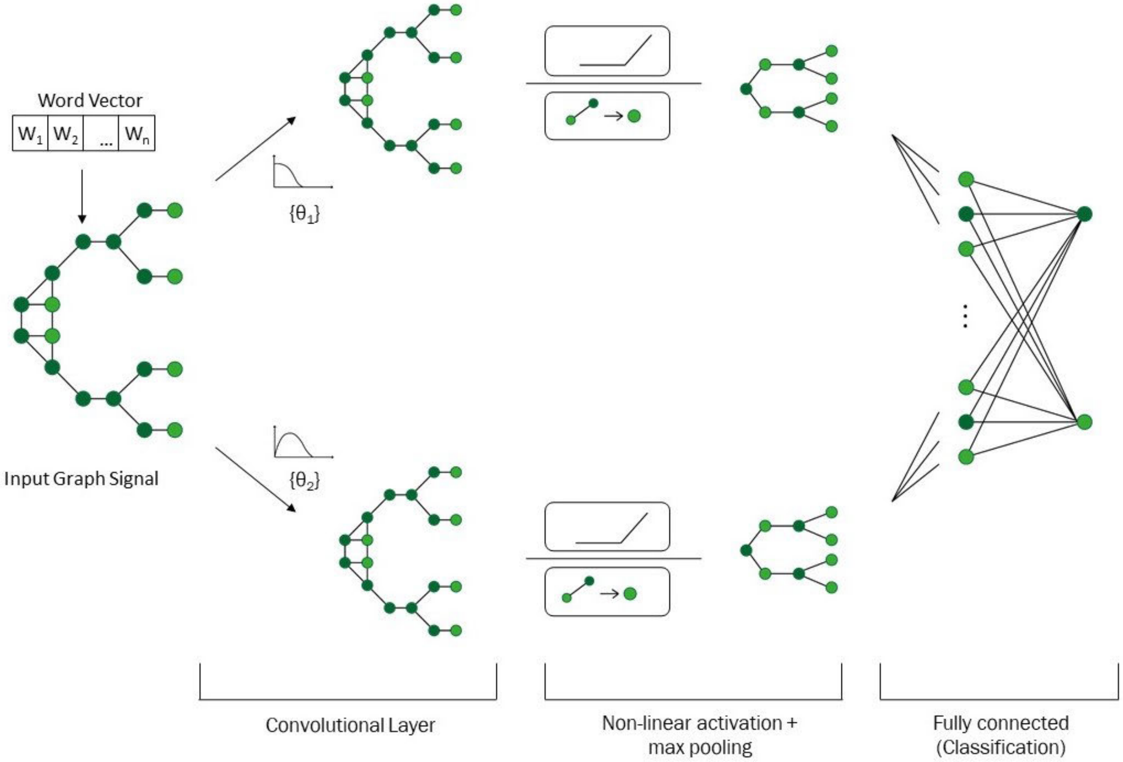

Word Vector
W1
W2
...
Wn
Input Graph Signal
{θ1}
{θ2}
Convolutional Layer
Non-linear activation + max pooling
Fully connected (Classification)

FIGURE 1. Graph convolutional neural network (GCNN) showing convolutional and pooling layers.

where $\boldsymbol{\theta}$ represents the GCNN parameters (weights and biases), P is the total number of parameters, and σ^2 is user-defined variance which is typically obtained from prior knowledge about distribution of parameters in trained neural networks.

Our implementation performs these calculations on the log scale to minimise numerical instability.

B. LANGEVIN-GRADIENT PROPOSAL DISTRIBUTION

Next, we use MCMC sampling to sample the posterior distribution of weights and biases of Bayes-GCNN. Deep learning models (such as GCNNs), can feature hundreds of thousands of parameters, hence state-of-art MCMC sampling methods are needed. Therefore, we employ 1) efficient proposal distribution via Langevin-gradients, 2) parallel computing that features inter-process communication, and 3) tempered MCMC to optimise sampling from multi-modal posterior distributions.

The Langevin-gradient proposal distribution essentially incorporates Gaussian noise with a gradient step taken using a single iteration (epoch) [32]. The gradient step can be either in form of stochastic-gradient descent (SGD) or adaptive gradient-descent, such as the Adam optimiser [34], [71]. Henceforth, we refer to SGD-based proposal distribution as Langevin-gradients (LG) and Adam-based proposal distribution as adaptive Langevin gradients (adapt-LG).

At a given step or chain position (n) of a MCMC sampler, we create a proposal denoted by superscript ($\star$) from a multivariate normal distribution $\boldsymbol{\theta}_n^\star$ as follows:

$$\boldsymbol{\theta}_n^\star \sim \mathcal{N}(\boldsymbol{\theta}_n + \nu_1 \bar{v}_n, \nu_2^2 I_P). \tag{9}$$

Here, I_P is an $P \times P$ identity matrix, scaled by the tuning parameter ν_2^2; and $\bar{v}_n$ is the Langevin gradient scaled by the user-defined learning rate ν_1. The Metropolis–Hastings probability (α) is used to accept/reject a proposed sample is as follows

$$\alpha = \min\left\{1, \frac{p(x|\boldsymbol{\theta}_n^\star)p(\boldsymbol{\theta}_n^\star)Q(\boldsymbol{\theta}_n|\boldsymbol{\theta}_n^\star)}{p(x|\boldsymbol{\theta}_n)p(\boldsymbol{\theta}_n)Q(\boldsymbol{\theta}_n^\star|\boldsymbol{\theta}_n)}\right\}, \tag{10}$$

where $Q(\boldsymbol{\theta}_n^\star|\boldsymbol{\theta}_n) = p(\boldsymbol{\theta}_n^\star|\boldsymbol{\theta}_n)$, the conditional proposal density and vice versa. Calculating the Q ratio is necessary because the Langevin-gradients are not symmetric at different steps in the chain.

Initially motivated by thermodynamics, tempered MCMC (also known as replica exchange and parallel tempering MCMC) [72], [72]–[74], samples an ensemble of M MCMC replicas $\Omega = [R_1, R_2, ..R_M]$. Each replica features temperature t from the temperature ladder $T = [1, \ldots T_{max}]$ which is typically geometrically spaced, with T_{max} defined by the user to control the extent of exploration. The likelihood of each replica in the ensemble is attenuated to form an attenuated posterior distribution $p_t(\boldsymbol{\theta}_n^t|x) \propto p(x|\boldsymbol{\theta}_n^t)^{1/t}p(\boldsymbol{\theta}_n^t)$. This ‘‘flattens’’ the distribution according to the temperature level of the replica, which increases the Metropolis–Hastings acceptance rates for replicas with higher temperature levels, which also helps in escaping from local minima. The neighbouring repli-

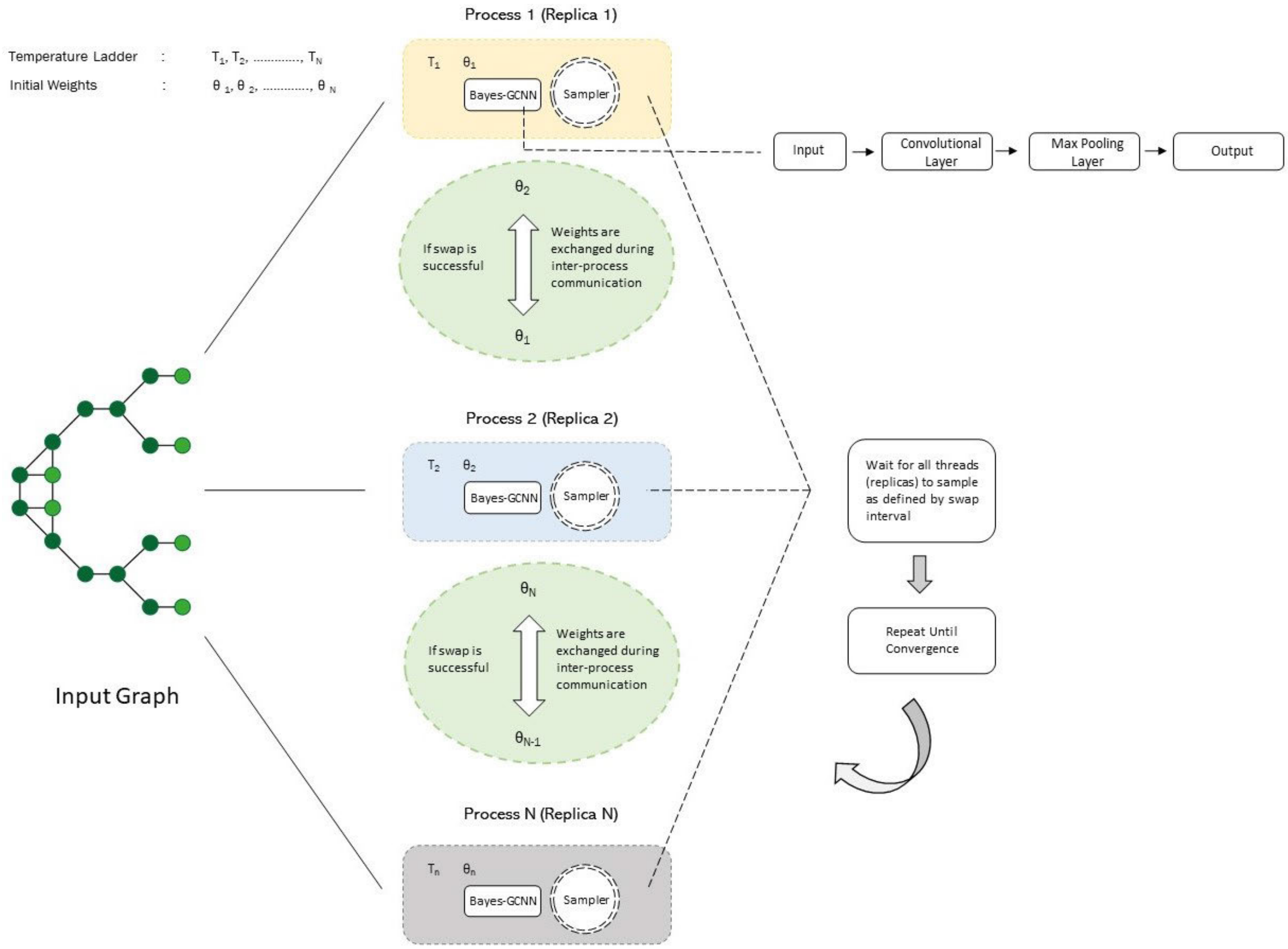

Temperature Ladder : T1, T2, ..., TN
Initial Weights : θ 1, θ 2, ..., θ N
Process 1 (Replica 1)
T1 θ1
Bayes-GCNN
Sampler
Input
Convolutional Layer
Max Pooling Layer
Output
θ2
If swap is successful
Weights are exchanged during inter-process communication
θ1
Input Graph
Process 2 (Replica 2)
T2 θ2
Bayes-GCNN
Sampler
Wait for all threads (replicas) to sample as defined by swap interval
Repeat Until Convergence
θN
If swap is successful
Weights are exchanged during inter-process communication
θN-1
Process N (Replica N)
Tn θn
Bayes-GCNN
Sampler

FIGURE 2. Bayesian GCNN framework implemented using tempered MCMC and parallel computing.

cas are periodically exchanged via a Metropolis–Hastings step, balancing exploration and exploitation [75], [76].

Effectively, this augments the sample space of $\boldsymbol{\theta}_n$ into $\boldsymbol{\theta}_n^t = (\theta, \tau^2, t)$, sampling jointly across $\boldsymbol{\theta}$ and t. Those realisations with $t = 1$ then have the target distribution. Therefore, periodically, a proposal is made to exchange the states of the neighbouring replicas, so that $\boldsymbol{\theta}_n^t = \boldsymbol{\theta}_n^{t+1}$ and $\boldsymbol{\theta}_n^{t+1} = \boldsymbol{\theta}_n^t$. The proposal distribution is symmetric and hence the priors cancel. Therefore, the acceptance probability is given by

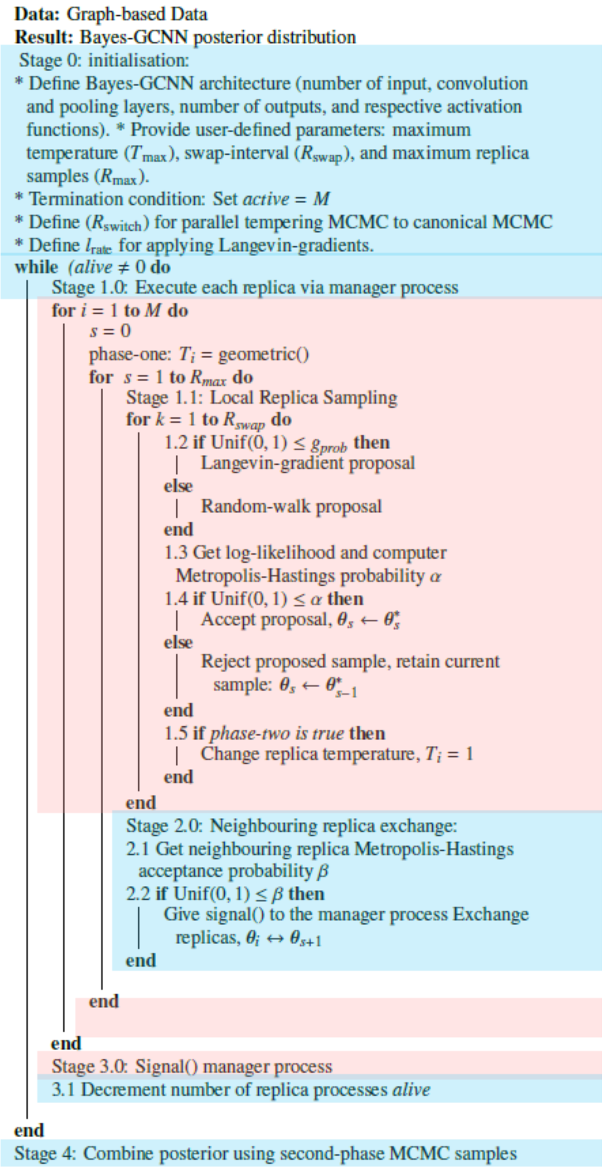
$$\beta = \min\left\{1, \frac{P(x|\boldsymbol{\theta}_n^t)^{1/(t+1)} P(x|\boldsymbol{\theta}_n^{(t+1)})^{1/t}}{P(x|\boldsymbol{\theta}_n^{(t+1)})^{1/(t+1)} P(x|\boldsymbol{\theta}_n^t)^{1/t}}\right\}. \tag{11}$$

.

C. BAYES-GCNN FRAMEWORK

Our Bayes-GCNN employs parallel processing for execution of tempered MCMC replicas that exchange states at regular interval via inter-process communication. The Bayes-GCNN framework that features parallel MCMC replica samplers, inter-process communication, and graph-based data is shown in Figure 2 and presented as an algorithm in Figure 3. We begin by defining the graph-CNN architecture that includes the size of convolution and max-pooling layers and number of output neurons given by the classification problem with given graph-based data representation for input as shown in Figure 1.

The execution begins with tempered replica sampling (Stage 1.1) with the manager process overseeing parallel replica processes. Each replica sample creates a proposal depending on the l_{rate} (Stage 1.2) using either random-walk or Langevin-gradients (9). In Stage 1.3, the log-likelihood is computed along with the Metropolis–Hastings probability (Stage 1.4) to accept/reject a proposed sample. Then, the algorithm checks (Stage 1.5) whether to carry on with tempered MCMC or change the replica temperature values to 1 for canonical MCMC. In this way, tempered MCMC is used in first-phase, then it switches to canonical MCMC in second-phase defined by R_{switch} to further balance global exploration with local and to ensure that the true posterior distribution is sampled during the second-phase as done in our previous works [34], [77]. Hence, the tempered MCMC is used as a burn-in sampling procedure which is not part of the posterior distribution. The tempered MCMC is used for exploration but the samples do not become part of the true



```
Data: Graph-based Data
Result: Bayes-GCNN posterior distribution
 Stage 0: initialisation:
* Define Bayes-GCNN architecture (number of input, convolution
  and pooling layers, number of outputs, and respective activation
  functions). * Provide user-defined parameters: maximum
  temperature ($T_{\max}$), swap-interval ($R_{\text{swap}}$), and maximum replica
  samples ($R_{\max}$).
* Termination condition: Set $active = M$
* Define ($R_{\text{switch}}$) for parallel tempering MCMC to canonical MCMC
* Define $l_{\text{rate}}$ for applying Langevin-gradients.
while (alive $\neq 0$ do
    Stage 1.0: Execute each replica via manager process
    for $i = 1$ to $M$ do
        $s = 0$
        phase-one: $T_i$ = geometric()
        for $s = 1$ to $R_{max}$ do
            Stage 1.1: Local Replica Sampling
            for $k = 1$ to $R_{swap}$ do
                1.2 if Unif(0, 1) $\leq g_{prob}$ then
                    Langevin-gradient proposal
                else
                    Random-walk proposal
                end
                1.3 Get log-likelihood and computer
                  Metropolis-Hastings probability $\alpha$
                1.4 if Unif(0, 1) $\leq \alpha$ then
                    Accept proposal, $\boldsymbol{\theta}_s \leftarrow \boldsymbol{\theta}_s^*$
                else
                    Reject proposed sample, retain current
                      sample: $\boldsymbol{\theta}_s \leftarrow \boldsymbol{\theta}_{s-1}^*$
                end
                1.5 if phase-two is true then
                    Change replica temperature, $T_i = 1$
                end
            end
            Stage 2.0: Neighbouring replica exchange:
            2.1 Get neighbouring replica Metropolis-Hastings
              acceptance probability $\beta$
            2.2 if Unif(0, 1) $\leq \beta$ then
                Give signal() to the manager process Exchange
                  replicas, $\boldsymbol{\theta}_i \leftrightarrow \boldsymbol{\theta}_{s+1}$
            end
        end
    end
    Stage 3.0: Signal() manager process
    3.1 Decrement number of replica processes alive
end
Stage 4: Combine posterior using second-phase MCMC samples
```

FIGURE 3. Tempered MCMC algorithm for Bayesian GCNN using parallel computing. Note that the Langevin-gradient proposal can be either LG or adapt-LG. The manager process is highlighted in blue and replica processes running in parallel are highlighted in pink.

posterior, since it features pseudo-posterior distribution (due to the temperature level affecting the replica log-likelihood).

Next the replica exchange is done depending on the replica swap-interval (R_{swap}) and Metropolis-Hastings probability (Stage 2.2) which considers the log-likelihood of neighboring replica processes. We note that the manager process is used to determine if the neighboring replicas can be swapped. In case if they are swapped, the chain position is exchanged via inter-process communication. Finally, the algorithm decrements the number of active replicas if the maximum number of replica samples ($R_{\max}$) are reached which enables the algorithm to end replica sampling. In post-replica sample stage (Stage 4), burn-in sampling period is removed (which includes tempered MCMC) and then combined with respective replica posterior distribution of Bayes-GCNN for further analysis.

In addition to the configuration of the GCNN, the user sets the number of replicas (M), maximum temperature of the temperature ladder ($T_{\max}$), neighbouring replica swap-interval (R_{swap}), and maximum number of replica samples ($R_{\max}$). The user must also set the Langevin-gradient rate (l_{rate}), which determines how often it is used for creating the proposal, as opposed to a random-walk proposal (effectively setting $\nu_1 = 0$).

IV. EXPERIMENTS AND RESULTS

We evaluate the distinct features of Bayes-GCNNs in terms of computational efficiency of tempered MCMC sampling, effect of different proposal distributions, and prediction accuracy for established benchmark datasets.

A. DATASET DESCRIPTION

We use Cora [78], CiteSeer [79] and PubMed [80] citation network datasets, which are commonly used to evaluate graph neural networks. Each dataset has one connected graph consisting of nodes representing a scientific publication. The edges of the graph serve as citation links between the scientific publications (nodes). Each publication in each dataset is described by a word vector indicating the absence/presence of the corresponding word from the dictionary. The details of the dataset in terms of number of nodes, edges, classes and training and test samples are provided in Table 1. We note that number of training instances is relatively low; however, this specific data split is used in the literature [29] and hence we used it for comparing results. The data split has 20 labels per class for training, and 1000 nodes for testing.

The Cora dataset consists of citation information for 2708 machine learning papers with 1433 unique words in its dictionary. The nodes are classified into 7 labels: "case-based", "genetic algorithms", "neural networks", "probabilistic methods", "reinforcement learning", "rule learning", and "theory". The CiteSeer dataset extracted from the CiteSeer digital library consists of 3327 scientific papers with 3703 unique words in it's dictionary. The nodes are classified into 6 labels: "agents", "artificial intelligence", "DB", "IR", "machine learning", "human computer interface". The PubMed dataset features papers about diabetes which contains 19717 scientific papers and 44,338 citation links with 500 unique words in the dictionary. The nodes are classified into 3 classes which include "diabetes mellitus, experimental", "diabetes mellitus Type 1" and "diabetes mellitus Type 2".

B. EXPERIMENT DESIGN

We first run experiments to assess the effect of the tempered MCMC tuning parameters (hyper-parameters) and report the computational time and classification accuracy. In all experiments, we use the following parameters determined from our trial experiments. In random-walk proposals, we create

TABLE 1. Overview of datasets using graph representation showing number (num.) of classes with training, and test instances.

Dataset	Nodes	Edges	Num. classes	Num. training	Num. testing
Cora	2708	5429	7	140	1000
CiteSeer	3327	4732	6	120	1000
PubMed	19717	44338	3	60	1000

TABLE 2. Bayes-GCNN topology showing the total number of parameters (weights and biases).

Dataset	Input Neurons	Output Neurons	Hidden Layers	Total parameters
Cora	1433	7	16	23063
CiteSeer	3703	6	16	59366
PubMed	500	3	16	8067

TABLE 3. Effect of adaptive Langevin-gradient (adapt-LG) rate on the classification accuracy (acc.) for the Cora dataset.

adapt-LG Rate	Train Acc.(Mean, Max, Std)	Test Acc. (Mean, Max, Std)	Swap Per.	Accept Per.	Time (min.)
0	14.25 22.14 1.51	16.85 32.20 7.39	50.89	40.00	72.30
0.25	98.32 100.00 4.50	74.52 79.50 3.81	47.74	40.75	79.13
0.5	98.73 100.00 3.45	74.95 79.30 3.18	46.58	43.04	83.43
0.75	98.94 100.00 2.28	75.35 79.30 2.17	47.04	45.25	87.72
1.0	99.05 100.00 2.06	75.52 79.20 1.99	48.43	49.75	85.01

TABLE 4. Effect of number of replicas in tempered MCMC for Cora dataset.

# Chains	Train Acc. (Mean, Max, Std)	Test Acc. (Mean, Max, Std)	Accept Per.	Swap Per.	Time (min.)
2	99.22 100.00 0.71	75.80 79.60 1.05	44.00	43.62	138.64
4	99.06 100.00 2.01	75.45 79.20 1.84	43.75	46.34	84.24
6	98.97 100.00 2.33	75.38 79.30 2.19	44.00	45.89	82.82
8	98.89 100.00 2.50	75.11 78.80 2.50	44.38	47.04	84.76
10	98.70 100.00 3.39	74.92 79.30 3.13	43.80	47.52	87.23

Gaussian noise with standard deviation $\nu_2 = 0.005$. We use maximum temperature ($T_{\max} = 2$) in tempered MCMC. We change tempered MCMC to canonical MCMC with $R_{\text{switch}} = 60$ percent of total samples. We use a swap interval $R_{\text{swap}} = 2$ samples and a maximum of 48,000 samples which are distributed across all the replicas.

We implement Bayes-GCNN using pyTorch[2] and pyTorch-geometric libraries[3] and Python multi-processing library for parallel MCMC replica processes. Table 2 provides the topology of Bayes-GCNN in terms of number of input, hidden, and output neurons for the respective datasets.

In the case of adaptive Langevin-gradients which is based on the Adam optimiser, the first and second order moments of the past gradient is used as a means of adapting the gradients [71]. In this case, we use a user defined learning rate based on the literature and trial experiments ($\nu_1 = 0.01$). In the case of canonical Langevin-gradients, we use user defined learning rate based on previous works [33], [34], [36] ($\nu_1 = 0.1$).

C. RESULTS

We begin by evaluating the effect of adaptive Langevin proposal (adapt-LG rate) on the proposal distribution with 8 replicas in tempered MCMC using the Cora dataset. Table 3 presents the classification accuracy on training and testing datasets (showing mean, best, and standard deviation), neighbouring replica swap rate (percentage), and percentage of accepted samples during sampling. We observe a positive association between the adapt-LG rate and percentage of acceptance samples which implies that adapt-LG provides better proposals when compared to random-walk proposal distribution. There is also an association between adapt-LG rate and computational time since computing gradients is expensive [34]. The accuracy for train and test dataset is much lower for adapt-LG rate of 0, which indicates that random-walk proposal distribution on its own cannot be used to train Bayes-GCNNs. However, higher adapt-LG rates do not appear to yield performance gains.

Table 4 shows the effect of number of replicas in tempered MCMC for the Cora dataset according to the same metrics as used previously. We observe a reduction in computational time as the number of replicas increases from 2 to 4; however, the classification performance does not change much. This can be directly attributed to speed and efficiency gained by having more replica processes running in parallel (one replica process/thread per processing core). We also observe that there is a gradual increase in the swap percentage (between neighboring replicas) for up to 8 replicas.

We then apply the Bayes-GCNN to the other datasets (CiteSeer and PubMed), using 8 replicas with adapt-LG rate of 0.75 and present the training and test classification accuracy in Table 5. The literature does not report the same result summaries (best, mean, standard deviation) that we do; hence, a direct comparison is not possible. We compare the results assuming those from literature as the best performance

[2] https://pytorch.org/

[3] https://pytorch-geometric.readthedocs.io/en/latest/

TABLE 5. Bayesian GCNN results comparison of established methods from the literature (unavailable values are left blank).

Method	Cora best (mean, std)	CiteSeer best (mean, std)	PubMed best (mean, std)
GCNN (SGD) [29] 2016	75.70 (,)	64.70 (,)	77.20 (,)
GCNN (Adam) [30] 2017	(81.50,)	(70.30,)	(79.00,)
Bayes-GCNN (LG)	54.70 (40.98, 5.04)	43.00 (30.82, 4.45)	69.50 (60.53, 3.01)
GCNN* (Adam)	81.70 (80.81, 0.65)	72.00 (70.51, 0.85)	79.50 (79.00, 0.60)
Bayes-GCNN (adapt-LG)	79.00 (74.71, 2.42)	68.90 (63.26, 2.74)	78.70 (74.94, 1.63)

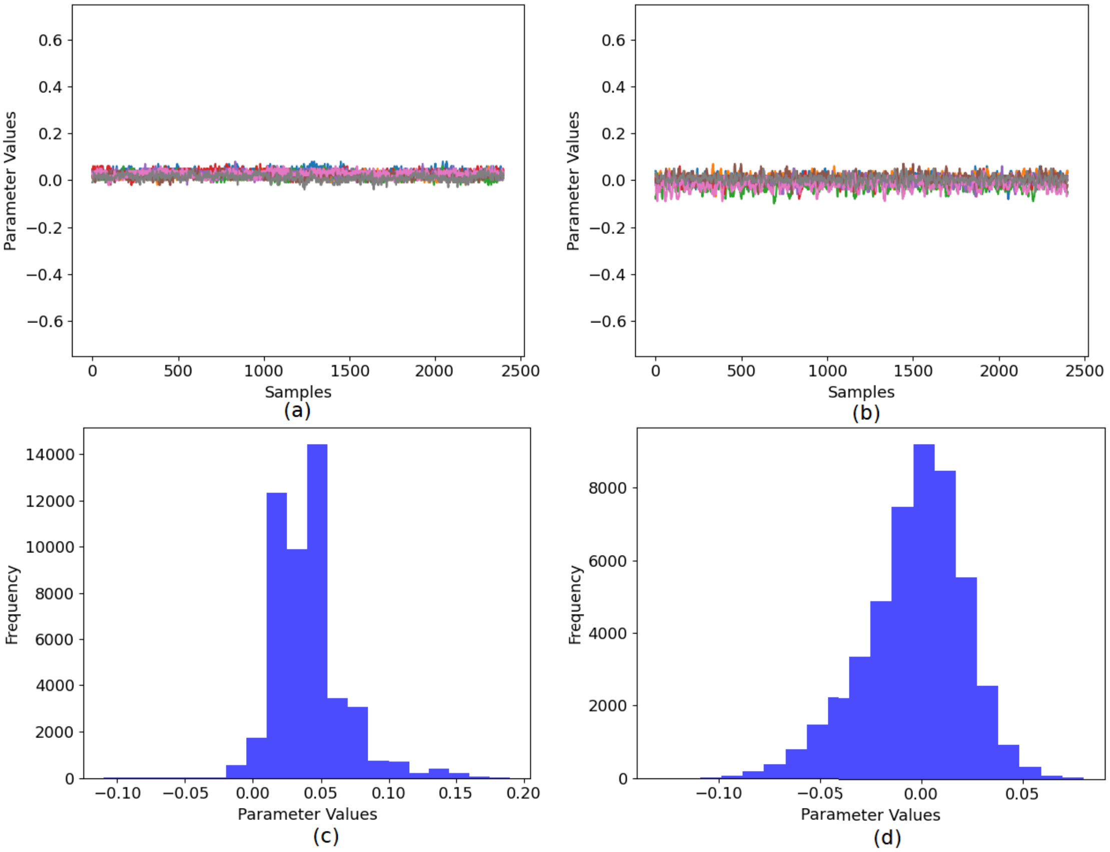


FIGURE 4. Posterior and trace plot for selected weights for Cora.

unless otherwise indicated. In the case of GCNN (SGD) [29], we report the mean classification accuracy. We also show performance by our own implementation of canonical GCNN* using Adam optimiser with 30 independent experiential runs with different initial weights and biases. Note in case of Bayes-GCNN, the mean, best and standard deviation are taken from posterior distribution of one experimental run.

Bayes-GCNN (adapt-LG) offers almost comparable classification accuracy to those in the literature (Table 5). Although Bayes-GCNN (adapt-LG) offers slightly lower classification accuracy (taking account the mean performance), it provides comprehensive uncertainty quantification in predictions. We also note that using LG rather than adapt-LG proposal distribution significantly deteriorates the performance of Bayes-GCNN.

Figures 4–6 show the Bayes-GCNN (adapt-LG) trace plot and posterior distribution for selected parameters (weights) from the respective datasets. The trace plots show 8 replica samples with different colours post the burn-in period (hence all replica temperature values are 1). Figure 4 presents the results from the Cora dataset showing the trace-plots (Panels a and c) and the posterior distribution (Panels b and d) for two selected weights with evidence of a unimodal posterior distribution. This can be seen in the dense histograms having a single peak for both the weights (Panels b and d). In the case for the CiteSeer dataset shown in Figure 5, we get a similar observation where the selected weights are similar in trace plots (Panels a and c), and both the posterior distributions (Panels b and d) show a single peak and indicate a single node in a unimodal posterior. In Figure 6 (posterior for PubMed

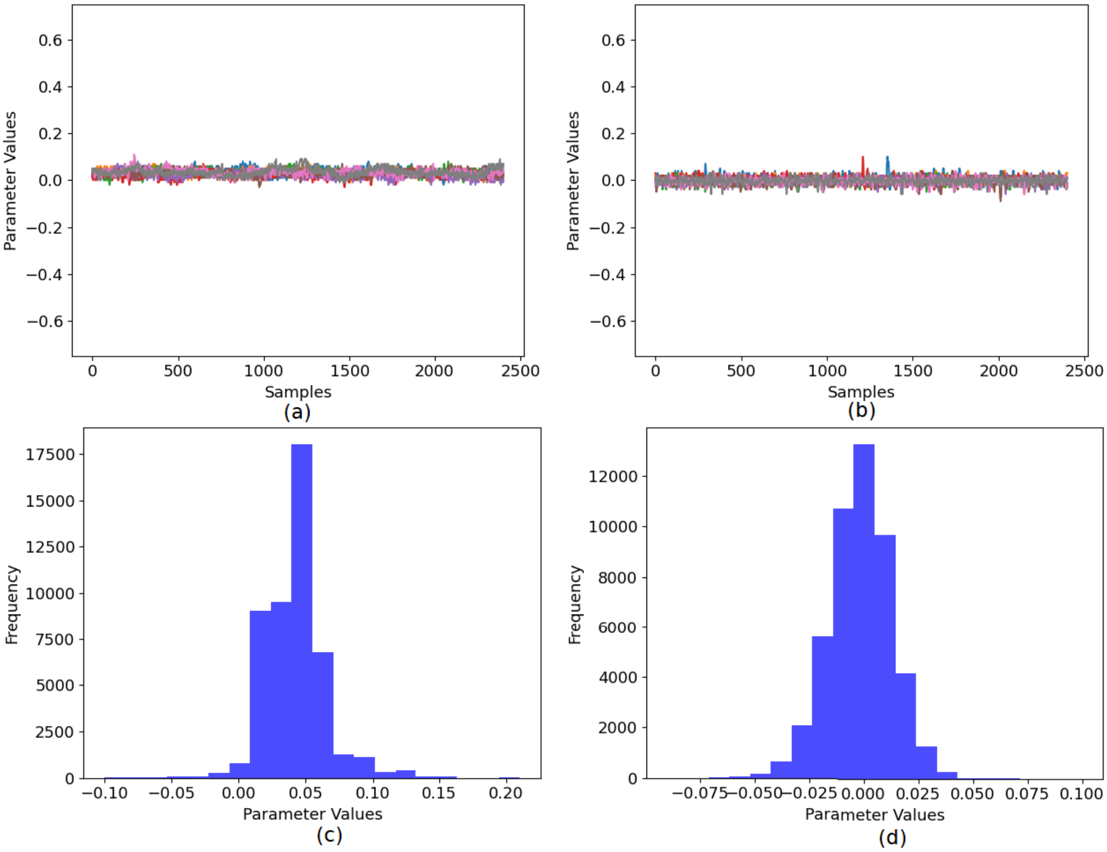


FIGURE 5. **Posterior and trace plot for selected weights for CiteSeer.**

dataset), we see a striking contrast between the trace plots (Panels a and c) of the selected weights: the posterior distribution of the first weight (Panel b) indicates a unimodal posterior, and the posterior distribution of the second weight (Panel d) indicates a bimodal posterior. An explanation of this is that Cora and CiteSeer datasets employ a much larger Bayes-GCNN model architecture given by the number of parameters when compared PubMed. (See Table 2.) All the trace plots show high correlation between the chains for all the respective datasets.

The reason behind the difference in the trace-plot and respective posterior distributions between the different datasets cannot be explained without further analysis. We note that we only selected two weights from thousands of parameters and this is merely for visualisation of the sampling process. The investigation as to why we get unimodal or multimodal posterior for different datasets and Bayes-GCNN architectures is beyond the scope of the paper.

Figure 7 shows the log-likelihood plot along with the training and test classification accuracy for a single MCMC replica (with temperature level of 1) for the respective problems. In the respective problems (Panel a, b, and c), we observe that the log-likelihood value increases in value over the time (samples), leading to higher training and test classification accuracy. We also notice that the case of PubMed dataset in Panel c, is slightly different than Cora and CiteSeer datasets (Panel a and b). We find that PubMed dataset has a higher variance in test classification accuracy over time when compared to others. This could be purely due to the application problem and size of the datasets and the Bayes-GCNN architecture as shown in Table 2.

Next, we present results to verify if the Bayes-GCNN (adapt-LG) has converged using the Gelman–Reubin diagnostic [81]. Table 6 shows their values for selected weights for their respective problems; $\hat{r}$ values close to 1 indicates convergence [81]. We use the different chain replicas for selected weights with unique identity number (Weight-ID) to determine if there is convergence between the MCMC replicas. The diagnostic uses the posterior distribution from all the replica chains after a burn-in of 60%. We observe that in Table 6, all the selected weight-IDs for the respective problems have values close to 1 and hence have obtained convergence.

V. DISCUSSION

This paper serves as a proof-of-concept of implementing Bayesian inference via tempered MCMC for GCNNs, achieving comparable performance with traditional methods (Table 5). Bayes-GCNNs provide a principled approach to uncertainty quantification for deep learning models. We observed that the adapt-LG proposal distribution performed significantly better than LG proposal distribution in

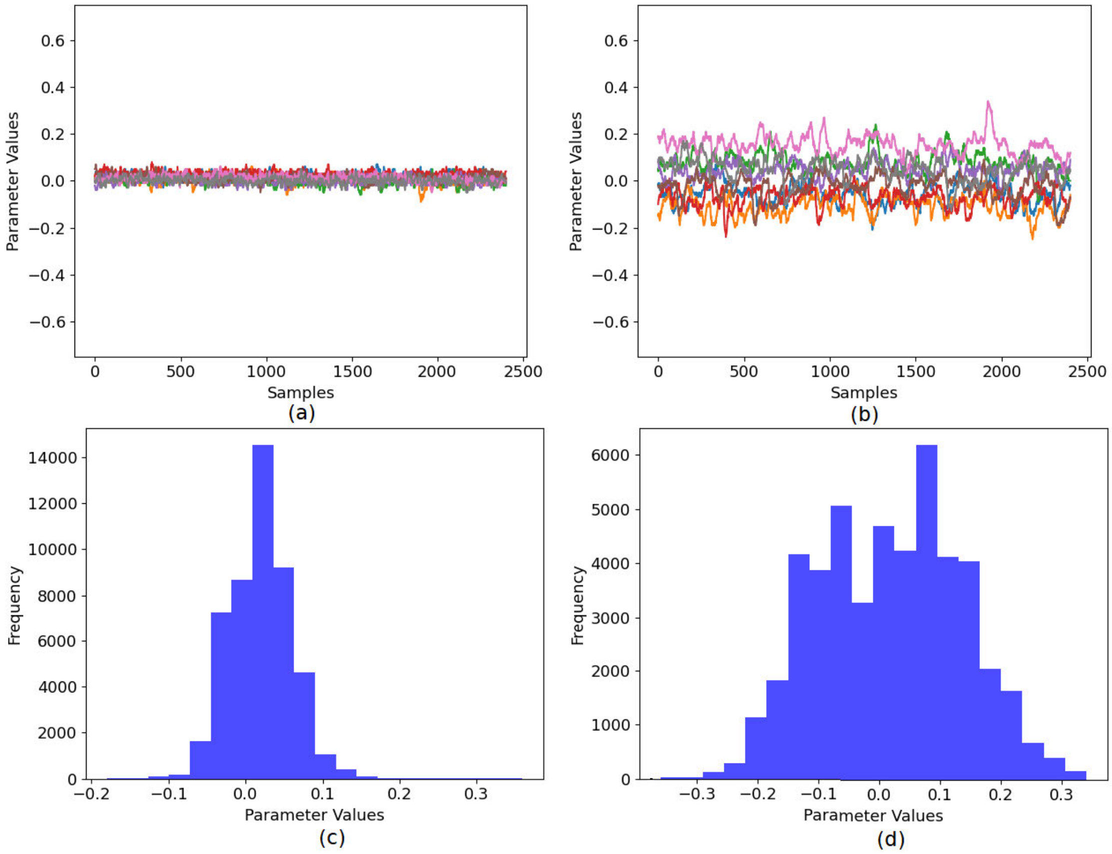
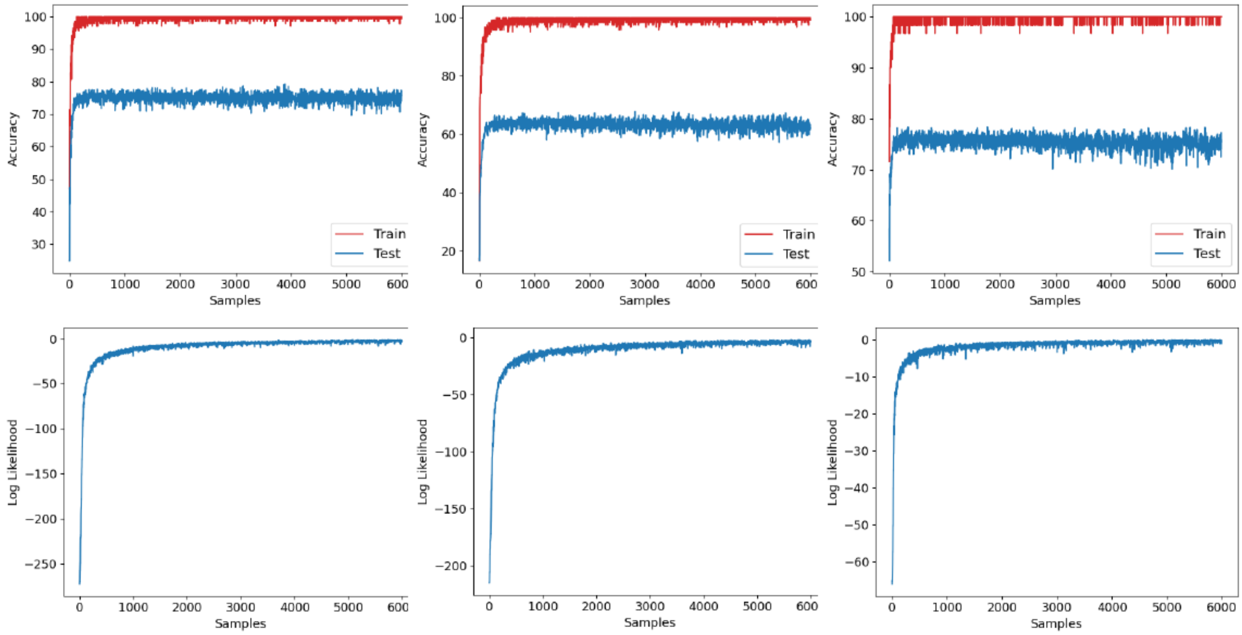


FIGURE 6. Posterior and trace plot for selected weights for PubMed.



(a) Cora

(b) CiteSeer

(c) PubMed

FIGURE 7. Accuracy And Log-Likelihood for different problems.

Bayes-GCNN. This can be attributed to the fact that LG uses a single and constant learning rate for all the weights whereas adapt-LG optimiser transforms the current gradient based on first and second moment of the past gradient. Hence, adapt-LG is better suited for convergence of large number of parameters when compared to conventional LG proposal

TABLE 6. Convergence diagnostics for the selected weights denoted by identity number (Weight-ID) for respective problems (Cora, CiteSeer, and PubMed).

Weight-ID	Cora	CiteSeer	PubMed
0	1.25	1.27	1.25
100	1.21	1.18	1.15
1000	1.15	1.16	1.23
5000	1.18	1.26	1.17
8000	1.22	1.28	1.20

distribution. Since a single and constant learning rate is used for all the weights in LG, only a portion of the weights may reach their local minima; however, some of the weights may not converge leading to poor performance or inability to train as shown in Table 5.

The adaptive nature adapt-LG based on the Adam optimiser may also be a mild violation of the Markov assumption underpinning MCMC, since the exact step length depends on the previous gradient and the Q-ratio might be approximate; these requirements can be relaxed somewhat, but need to be checked carefully [82], which can be the subject for future work.

The Gelman–Rubin diagnostics has been the most commonly used method to evaluate convergence of Markov chains, due to ease of implementation and availability in software packages. However, it has also been reported to sometimes give a premature and unreliable convergence diagnosis, particularly in cases where the Markov chains are stuck in a local maxima [83]. Moreover, the effectiveness of Gelman–Rubin diagnostics for large numbers of parameters is not well studied in the literature. Better convergence diagnosis methods needs to be developed, since deep learning models features tens of thousands of parameters. Moreover, auto-correlation and effective sample size has also been widely used for MCMC convergence [84], [85], and their applicability for Bayesian deep learning models can also be evaluated.

The comparison of results with the literature motivates the implementation of Bayesian framework for other deep learning models, which includes LSTM and CNN models [47]. The uncertainty quantification in predictions can be useful for application areas, such as traffic forecasting [86] and emotion recognition [87]. We addressed graph-based CNNs in this paper; however, the approach can be used for other graph deep learning architectures such as graph-LSTM networks [86], [87] and conventional graph neural networks [88].

The use of Bayesian inference via MCMC schemes in graph neural networks is largely unexplored. Future work could focus on implementing Bayesian inference via MCMC for other architectures of graph neural networks, such as graph LSTM models [86] and graph attention networks [40] using different datasets (such as Quantum Machine 9) [89], [90], bioinformatics (enzymes) [91], [92], and social news network (Reddit) [93]. Further work could also focus on the efficacy of other types of MCMC schemes such as Hamiltonian MCMC methods [94] to further improve the classification performance.

VI. CONCLUSION

We presented Bayes-GCNN that featured tempered MCMC sampling via parallel computing with adaptive Langevin-gradient proposal distribution. Our results indicated that while the mean accuracy of the Bayes-GCNN was around 4-5% lower for the CiteSeer problem, the maximum accuracy in general is on par with the accuracy of canonical GCNN for all the benchmark problems. Hence, Bayes-GCNN provides an alternative form of training that features a principled approach to quantify uncertainty in model parameters. Bayes-GCNN eliminates the need to run repetitive experiments with a probabilistic representation of weights and biases. In addition, canonical optimisers do not offer uncertainty quantification on their own which is needed for certain problems; hence, Bayes-GCNN has good potential for real-world applications.

ACKNOWLEDGMENT

(Rohitash Chandra and Ayush Bhagat contributed equally to this work.)

CODE AND DATA

Python-based implementation for Bayes-GCNN along with data is available.[4]

REFERENCES

[1] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2008.

[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[3] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 17, 2020, doi: 10.1109/TKDE.2020.2981333.

[4] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018, *arXiv:1810.00826*. [Online]. Available: https://arxiv.org/abs/1810.00826

[5] D. J. Cook and L. B. Holder, *Mining Graph Data*. Hoboken, NJ, USA: Wiley, 2006.

[6] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1416–1424.

[7] F. Wu, T. Zhang, A. H. de Souza Jr., C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," 2019, *arXiv:1902.07153*. [Online]. Available: https://arxiv.org/abs/1902.07153

[8] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.* Greece: Springer, 2018, pp. 593–607.

[9] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic object parsing with graph LSTM," in *Proc. Eur. Conf. Comput. Vis.* Amsterdam, The Netherlands: Springer, 2016, pp. 125–143.

[10] V. Zayats and M. Ostendorf, "Conversation modeling on reddit using a graph-structured LSTM," *Trans. Assoc. Comput. Linguistics*, vol. 6, pp. 121–132, Dec. 2018.

[11] J. Tang, X. Shu, R. Yan, and L. Zhang, "Coherence constrained graph LSTM for group activity recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Jul. 15, 2019, doi: 10.1109/TPAMI.2019.2928540.

[12] X. Shu, L. Zhang, Y. Sun, and J. Tang, "Host–parasite: Graph LSTM-in-LSTM for group activity recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 663–674, Feb. 2021.

[4] https://github.com/sydney-machine-learning/BayesianGraphNeuralNetworks

[13] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "Mgae: Marginalized graph autoencoder for graph clustering," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 889–898.

[14] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," 2018, *arXiv:1802.04407*. [Online]. Available: https://arxiv.org/abs/1802.04407

[15] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: Graph representation learning with generative adversarial nets," 2017, *arXiv:1711.08267*. [Online]. Available: https://arxiv.org/abs/1711.08267

[16] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," 2020, *arXiv:2005.11650*. [Online]. Available: https://arxiv.org/abs/2005.11650

[17] C. Chen, K. Li, S. G. Teo, X. Zou, K. Wang, J. Wang, and Z. Zeng, "Gated residual recurrent graph neural networks for traffic prediction," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 485–492.

[18] H. Peng, H. Wang, B. Du, M. Z. A. Bhuiyan, H. Ma, J. Liu, L. Wang, Z. Yang, L. Du, S. Wang, and P. S. Yu, "Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting," *Inf. Sci.*, vol. 521, pp. 277–290, Jun. 2020.

[19] J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics," *Mach. Learn., Sci. Technol.*, vol. 2, no. 2, Jan. 2021, Art. no. 021001.

[20] O. Wieder, S. Kohlbacher, M. Kuenemann, A. Garon, P. Ducrot, T. Seidel, and T. Langer, "A compact review of molecular property prediction with graph neural networks," *Drug Discovery Today, Technol.*, Dec. 2020, doi: 10.1016/j.ddtec.2020.11.009.

[21] M. Wang and G. Hu, "A novel method for Twitter sentiment analysis based on attentional-graph neural network," *Information*, vol. 11, no. 2, p. 92, Feb. 2020.

[22] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 968–977.

[23] Q. Cao, H. Shen, J. Gao, B. Wei, and X. Cheng, "Popularity prediction on social platforms with coupled graph neural networks," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 70–78.

[24] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," 2018, *arXiv:1812.08434*. [Online]. Available: https://arxiv.org/abs/1812.08434

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "ImageNet large scale visual recognition challenge," *Int. J. Conflict Violence*, vol. 115, no. 3, pp. 211–252, 2015.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[29] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," 2016, *arXiv:1603.08861*. [Online]. Available: https://arxiv.org/abs/1603.08861

[30] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: https://arxiv.org/abs/1609.02907

[31] R. M. Neal, "Sampling from multimodal distributions using tempered transitions," *Statist. Comput.*, vol. 6, no. 4, pp. 353–366, Dec. 1996.

[32] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 681–688.

[33] R. Chandra, K. Jain, A. Kapoor, and A. Aman, "Surrogate-assisted parallel tempering for Bayesian neural learning," *Eng. Appl. Artif. Intell.*, vol. 94, Sep. 2020, Art. no. 103700.

[34] R. Chandra, K. Jain, R. V. Deo, and S. Cripps, "Langevin-gradient parallel tempering for Bayesian neural learning," *Neurocomputing*, vol. 359, pp. 315–326, Sep. 2019.

[35] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1613–1622.

[36] R. Chandra and A. Kapoor, "Bayesian neural multi-source transfer learning," *Neurocomputing*, vol. 378, pp. 54–64, Feb. 2020.

[37] J. A. Bondy and U. S. R. Murty, *Graph Theory With Applications*. London, U.K.: Macmillan, 1976.

[38] D. B. West, *Introduction to Graph Theory*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[39] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.

[40] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*. [Online]. Available: https://arxiv.org/abs/1710.10903

[41] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "GaAN: Gated attention networks for learning on large and spatiotemporal graphs," 2018, *arXiv:1803.07294*. [Online]. Available: https://arxiv.org/abs/1803.07294

[42] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," 2015, *arXiv:1511.05493*. [Online]. Available: https://arxiv.org/abs/1511.05493

[43] A. Rahimi, T. Cohn, and T. Baldwin, "Semi-supervised user geolocation via graph convolutional networks," 2018, *arXiv:1804.08049*. [Online]. Available: http://arxiv.org/abs/1804.08049

[44] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-I. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5453–5462.

[45] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," 2017, *arXiv:1704.02901*. [Online]. Available: https://arxiv.org/abs/1704.02901

[46] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," 2015, *arXiv:1511.02136*. [Online]. Available: https://arxiv.org/abs/1511.02136

[47] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," 2015, *arXiv:1509.09292*. [Online]. Available: https://arxiv.org/abs/1509.09292

[48] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proc. World Wide Web Conf. (WWW)*, 2018, pp. 499–508, doi: 10.1145/3178876.3186116.

[49] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," 2016, *arXiv:1611.08402*. [Online]. Available: https://arxiv.org/abs/1611.08402

[50] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2011.

[51] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," 2018, *arXiv:1801.03226*. [Online]. Available: https://arxiv.org/abs/1801.03226

[52] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[53] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 974–983.

[54] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," 2018, *arXiv:1801.10247*. [Online]. Available: https://arxiv.org/abs/1801.10247

[55] J. Chen, J. Zhu, and L. Song, "Stochastic training of graph convolutional networks with variance reduction," 2017, *arXiv:1710.10568*. [Online]. Available: https://arxiv.org/abs/1710.10568

[56] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*. [Online]. Available: https://arxiv.org/abs/1611.07308

[57] H. Wang and D.-Y. Yeung, "A survey on Bayesian deep learning," *ACM Comput. Surv.*, vol. 53, no. 5, pp. 1–37, Oct. 2020.

[58] N. G. Polson and V. Sokolov, "Deep learning: A Bayesian perspective," *Bayesian Anal.*, vol. 12, no. 4, pp. 1275–1304, Dec. 2017.

[59] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, "Grammar variational autoencoder," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1945–1954.

[60] L. Mescheder, S. Nowozin, and A. Geiger, "Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2391–2400.

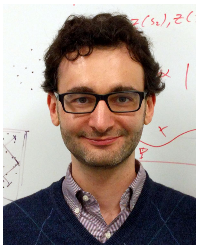
[61] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, "Variational convolutional neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 2780–2789.

[62] J.-T. Chien and K.-T. Kuo, "Variational recurrent neural networks for speech separation," in *Proc. 18th Annu. Conf. Int. speech Commun. Assoc. (INTERSPEECH)*, 2017, pp. 1193–1197.
[63] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational LSTM enhanced anomaly detection for industrial big data," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3469–3477, May 2021.
[64] J. Su, S. Wu, D. Xiong, Y. Lu, X. Han, and B. Zhang, "Variational recurrent neural machine translation," 2018, *arXiv:1801.05119*. [Online]. Available: https://arxiv.org/abs/1801.05119
[65] S. Bonner, A. Atapour-Abarghouei, P. T. Jackson, J. Brennan, I. Kureshi, G. Theodoropoulos, A. S. McGough, and B. Obara, "Temporal neighbourhood aggregation: Predicting future links in temporal graphs via recurrent variational graph convolutions," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 5336–5345.
[66] M. Qu, Y. Bengio, and J. Tang, "GMNN: Graph Markov neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5241–5250.
[67] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with Bernoulli approximate variational inference," 2015, *arXiv:1506.02158*. [Online]. Available: https://arxiv.org/abs/1506.02158
[68] R. M. Neal, "MCMC using Hamiltonian dynamics," in *Handbook of Markov Chain Monte Carlo*, vol. 2, no. 11. Boca Raton, FL, USA: CRC Press, 2011.
[69] D. Levy, M. D. Hoffman, and J. Sohl-Dickstein, "Generalizing Hamiltonian Monte Carlo with neural networks," 2017, *arXiv:1711.09268*. [Online]. Available: https://arxiv.org/abs/1711.09268
[70] A. D. Cobb and B. Jalaian, "Scaling Hamiltonian Monte Carlo inference for Bayesian neural networks with symmetric splitting," 2020, *arXiv:2010.06772*. [Online]. Available: http://arxiv.org/abs/2010.06772
[71] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[72] R. H. Swendsen and J.-S. Wang, "Replica Monte Carlo simulation of spin-glasses," *Phys. Rev. Lett.*, vol. 57, no. 21, p. 2607, 1986.
[73] K. Hukushima and K. Nemoto, "Exchange Monte Carlo method and application to spin glass simulations," *J. Phys. Soc. Jpn.*, vol. 65, no. 6, pp. 1604–1608, Jun. 1996.
[74] U. H. Hansmann, "Parallel tempering algorithm for conformational studies of biological molecules," *Chem. Phys. Lett.*, vol. 281, nos. 1–3, pp. 140–150, 1997.
[75] M. K. Sen and P. L. Stoffa, "Bayesian inference, Gibbs' sampler and uncertainty estimation in geophysical inversion," *Geophys. Prospecting*, vol. 44, no. 2, pp. 313–350, 1996.
[76] M. Maraschini and S. Foti, "A Monte Carlo multimodal inversion of surface waves," *Geophys. J. Int.*, vol. 182, no. 3, pp. 1557–1566, 2010.
[77] R. Chandra, R. D. Müller, D. Azam, R. Deo, N. Butterworth, T. Salles, and S. Cripps, "Multicore parallel tempering bayeslands for basin and landscape evolution," *Geochemistry, Geophysics, Geosystems*, vol. 20, no. 11, pp. 5082–5104, Nov. 2019.
[78] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, p. 93, 2008.
[79] Q. Lu and L. Getoor, "Link-based classification," in *Proc. 20th Int. Conf. Mach. Learn.*, T. Fawcett and N. Mishra, Eds. Washington, DC, USA, 2003, pp. 496–503.
[80] G. Namata, B. London, L. Getoor, B. Huang, and U. Edu, "Query-driven active surveying for collective classification," in *Proc. 10th Int. Workshop Mining Learn. Graphs*, 2012, pp. 1–8.
[81] D. Vats and C. Knudson, "Revisiting the Gelman-Rubin diagnostic," 2018, *arXiv:1812.09384*. [Online]. Available: https://arxiv.org/abs/1812.09384
[82] G. O. Roberts and J. S. Rosenthal, "Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms," *J. Appl. Probab.*, vol. 44, no. 2, pp. 458–475, Jun. 2007.
[83] J. M. Flegal, M. Haran, and G. L. Jones, "Markov chain Monte Carlo: Can we trust the third significant figure?" *Stat. Sci.*, vol. 23, no. 2, pp. 250–260, May 2008, doi: 10.1214/08-STS257.
[84] V. Roy, "Convergence diagnostics for Markov chain Monte Carlo," *Annu. Rev. Statist. Appl.*, vol. 7, no. 1, pp. 387–412, Mar. 2020.
[85] M. K. Cowles and B. P. Carlin, "Markov chain Monte Carlo convergence diagnostics: A comparative review," *J. Amer. Statist. Assoc.*, vol. 91, no. 434, pp. 883–904, Jun. 1996.
[86] T. Bogaerts, A. D. Masegosa, J. S. Angarita-Zapata, E. Onieva, and P. Hellinckx, "A graph CNN-LSTM neural network for short and long-term traffic forecasting based on trajectory data," *Transp. Res. C, Emerg. Technol.*, vol. 112, pp. 62–77, Mar. 2020.
[87] Y. Yin, X. Zheng, B. Hu, Y. Zhang, and X. Cui, "EEG emotion recognition using fusion model of graph convolutional neural networks and LSTM," *Appl. Soft Comput.*, vol. 100, Mar. 2021, Art. no. 106954.
[88] B. Donon, R. Clément, B. Donnot, A. Marot, I. Guyon, and M. Schoenauer, "Neural networks for power flow: Graph neural solver," *Electr. Power Syst. Res.*, vol. 189, Dec. 2020, Art. no. 106547.
[89] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Sci. Data*, vol. 1, no. 1, pp. 1–7, Dec. 2014.
[90] L. Ruddigkeit, R. van Deursen, L. C. Blum, and J.-L. Reymond, "Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17," *J. Chem. Inf. Model.*, vol. 52, no. 11, pp. 2864–2875, Nov. 2012.
[91] K. M. Borgwardt, C. S. Ong, S. Schonauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. 1, pp. i47–i56, Jun. 2005.
[92] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg, "Brenda, the enzyme database: Updates and major new developments," *Nucleic Acids Res.*, vol. 32, pp. D431–D433, Jan. 2004.
[93] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 1365–1374.
[94] M. Betancourt, "A conceptual introduction to Hamiltonian Monte Carlo," 2017, *arXiv:1701.02434*. [Online]. Available: https://arxiv.org/abs/1701.02434

ROHITASH CHANDRA (Senior Member, IEEE) is currently a Senior Lecturer in data science with the UNSW School of Mathematics and Statistics. He leads a program of research encircling methodologies and applications of artificial intelligence; particularly in areas of Bayesian deep learning, neuro-evolution, climate extremes, geoscientific models, and mineral exploration. He has developed novel methods for machine learning inspired by neural systems and learning behaviour that include transfer and multi-task learning, with the goal of modular deep learning. His current interests include uncertainty quantification and deep learning with applications to language models, vaccine research, and COVID-19.

AYUSH BHAGAT is currently pursuing the B.Tech. degree in computer science with a minor in big data and analytics with Manipal Institute of Technology. He is also working as an Analyst at Axxela, a proprietary trading firm. His research interests include machine learning, sentiment analysis, and algorithmic trading.

MANAVENDRA MAHARANA received the B.Tech. degree in computer science with a minor in big data and analytics from Manipal Institute of Technology. He is currently working with Microsoft as a Customer Success Account Manager. His research interests include machine learning, behavioral economics, misinformation, and ethical AI.



PAVEL N. KRIVITSKY is currently a Senior Lecturer in statistics with the University of New South Wales. His research interests include statistical modeling of social network data and processes for applications in epidemiology, social sciences, and defence, statistical software and computing, and survey sampling.