Neurocomputing 378 (2020) 54–64

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Bayesian neural multi-source transfer learning

Rohitash Chandra[a,b,*], Arpit Kapoor[a,c]

[a] Centre for Translational Data Science, The University of Sydney, NSW 2006, Australia
[b] School of Geosciences, The University of Sydney, NSW 2006, Australia
[c] Department of Computer Science and Enginnering, SRM Institute of Science and Technology, Chennai, Tamil Nadu 603203, India

ARTICLE INFO

Article history:
Received 11 May 2019
Revised 30 August 2019
Accepted 8 October 2019
Available online 20 October 2019

Communicated by Dr Xianming Zhang

Keywords:
Bayesian methods
Transfer learning
Neural networks
Data fusion
Bayesian neural networks

ABSTRACT

Although the use of deep learning and neural networks techniques are gaining popularity, there remain a number of challenges when multiple sources of information and data need to be combined. Although transfer learning and data fusion methodologies try to address this challenge, they lack robust uncertainty quantification which is crucial for decision making. Bayesian inference provides a rigorous approach for uncertainty quantification in decision making. Uncertainty quantification using Bayesian inference takes into consideration uncertainty associated with model parameters, as well as, the uncertainty in combining multiple sources of data. In this paper, we present a Bayesian framework for transfer learning using neural networks that considers single and multiple sources of data. We use existence of prior distributions to define the dependency between different data sources in a multi-source Bayesian transfer learning framework. We use Markov Chain Monte-Carlo method to obtain samples from the posterior distribution that consider the knowledge from the *source* datasets as *priors*. The results show that the framework provides a robust probabilistic approach for decision making with accuracy that is similar to gradient-based learning methods. Moreover, the results are comparable to related machine learning methods used for transfer learning in the literature.

1. Introduction

Traditional machine learning methodologies utilize the training and test datasets drawn from the same feature space or distribution [1,2]. Transfer learning features the reuse of knowledge learnt from different but related problems [3–6]. Transfer learning is useful in situations where the training dataset is not sufficient, hence data from related problems are considered as *source* datasets that help in learning. In these situations, the knowledge from trained sources belonging to same domain can be used to train a model on the required *target* task. Transfer learning has been successfully applied to a wide range of domains that include activity recognition [7], object recognition [8,9], text mining [10], and reinforcement learning [11]. Transfer learning has been used as a remedy for challenging problems such as learning from unlabeled data into an approach known as self-taught learning [12]. Some prominent surveys that covers the various methodologies and applications in transfer learning are given [5,6]. Transfer learning has also been used for the case of unsupervised learning which assumes absence of labels in the target domain [13–15]. A related domain for transfer learning is multi-task learning which focuses on improving the performance of all the problems (tasks), whereas, in transfer learning the main focuses on the target task [5]. Since transfer learning methodologies take multiple sources of data and information into account [6], the challenge increases when incorporating uncertainty quantification in decision making.

Bayesian methods provides a probabilistic representation of the model that naturally accounts for uncertainty in decision making which have a wide range of applications in various domains that include machine learning, econometrics, environmental and Earth sciences [16–21]. Bayesian neural networks, also known as Bayesian neural learning feature Markov Chain Monte-Carlo (MCMC) methods for sampling the model parameters that are represented as probability distributions [22,23]. Note that learning or optimization typically refers to the strategy of finding the set of model parameters (weights and biases) in a neural network that best describes or captures the training data. One can view the inference procedure as a form of learning, which highlights uncertainty quantification in decision making due to probabilistic representation of the model parameters as the posterior distribution, rather than single point estimates [24]. MCMC methods face a number of challenges given the curse of dimensionality; hence, the progress of Bayesian inference for neural networks

* Corresponding author at: Centre for Translational Data Science, The University of Sydney, NSW 2006, Australia.
E-mail addresses: rohitash.chandra@sydney.edu.au (R. Chandra), kapoor.arpit97@gmail.com (A. Kapoor).

https://doi.org/10.1016/j.neucom.2019.10.042
0925-2312/

has been slow. Recently, Bayesian deep learning has gained attention that features an approximation to MCMC sampling using dropouts that has been useful for large networks that feature millions of weights [25]. Other ways of addressing the shortcomings of canonical MCMC methods is by combining them with gradient-based methods, where the gradients are used for creating efficient proposals. This is known as *Metropolis-adjusted Langevin dynamics* that essentially features gradient-based proposals with stochastic noise [26–28], which has shown promising performance for neural networks [29,30].

In the past, several strategies have been used to incorporate the Bayesian methodology for transfer learning [31,32]. Luis et al. [31] used Bayesian networks (not neural networks), that considers both network structure and parameter learning. Cao et al. [32] proposed an efficient adaptive transfer learning algorithm based on Gaussian process models that showed that the adaptive transfer learning can avoid negative transfer learning, in the cases where the correlation between the source and target task is minimal. Karbalayghareh et al. [33] presented a homogeneous Bayesian transfer learning framework where the source and target domains are related through the *joint prior density* of the model parameters. Roy and Kaelbling [34] presented a hierarchical extension of the *naive Bayes classifier* by placing process priors on the parameters. Moreover, Bayesian approaches have been used in multi-task learning that features bi-directional transfer Gaussian process model learned from multiple tasks that were assumed to be drawn independently from the same Gaussian process prior [35,36]. Lawrence and Platt [35] proposed an algorithm for multi-task learning based on Gaussian Processes (GP) where the learning of the parameters is done through multiple tasks with an assumption that these are drawn from the same GP *prior*.

Although there has been some work done for Bayesian transfer learning that featured Gaussian process models [33,34], there has not been much work done using neural networks. Since Gaussian process models can naturally account for uncertainty, they did not feature MCMC methods for inference in the case of transfer learning. A major feature of Bayesian inference is the incorporation of prior information which is also known as *priors* [37]. Typically, the priors would refer to the information the model possess without examining the data [38]. In the case of transfer learning where multiple sources of data are present to guide the target data, our motivation is to uttilize knowledge from the source datasets or models during the sampling process. Furthermore, we need to take into consideration uncertainty associated with model parameters from multiple sources of data which has not been well addressed in the domain of Bayesian neural learning. Our work is also motivated by the Langevin-based Bayesian neural learning [29,30] that features gradient proposals for creating proposal distribution, that can also be extended for multi-source transfer learning. We note that the proposal distribution refers to the distribution from which the proposed samples are drawn during MCMC sampling. In a *Markov Chain*, the proposal distribution depends only on the previous state of the chain.

In this paper, we present a novel Bayesian framework for transfer learning using neural networks that consider single and multiple sources of data for both regression and pattern classification setting. The framework uses knowledge (configuration of neural network weights) from the source models to assist the target model during sampling. We provide an incremental learning approach via MCMC sampling where the source and target networks are sampled in a *round-robin fashion*. Given multiple source datasets with their respective models, at a given time, the framework selects the sources equally likely. In the experiments, we first provide an investigation of the effect of certain parameters that evaluate the quality of knowledge transfer for Bayesian neural learning. We use Langevin-gradient proposal distribution and compare their performance with random-walk proposal distribution. Furthermore, we provide comparision with related methods from the literature.

The rest of the paper is presented as follows. Section 2 provides a brief review of the transfer learning literature, Section 3 presents the methodology and Section 4 provides experimentation and results. Section 5 provides a discussion and Section 6 concludes the paper with discussion of future research.

2. Related work

2.1. Transfer learning

Transfer learning [5] has been motivated by the field of educational psychology which refers to the transfer of learning that occurs when learning in one context enhances (positive transfer) or undermines (negative transfer) the performance in another [39–41]. In machine learning, one of the earliest work in transfer learning was presented by Pratt et. al [3,4] with the idea of neural network transfer where learning on a target problem is enhanced by the weights obtained from a network trained for a related source task. Transfer learning literature also overlaps with *data fusion* that considers multiple sources which feature challenges such as interoperability, heterogeneity and relatedness of tasks [42,43]. Data fusion techniques combine data from multiple domains to enhance performance accuracy in the domain of interest. Whereas, transfer learning incorporates the knowledge from the *source task(s)* to improving learning process with the *target task*. The knowledge considered for transfer could be probability distributions (in the case of Gaussian process models) [33,34,44], weights (in the case of a neural networks), and instances of data from source task(s), which are combined with data in the target task. In some instances, the transfer learning techniques can be coupled with data fusion methods [45]. Moreover, multi-sensor data fusion refers to combining information from several sources to have a unified representation [43].

Transfer learning has been implemented in two major scenarios that includes single and multi-source transfer learning. In the single-source case, knowledge from one source is transferred to the target [32], whereas the multi-source case considers several source domains. [46–49]. The challenge in transfer learning also has been in considering different types of datasets for knowledge transfer. Heterogeneous transfer refers to the transfer of knowledge from source(s) to target, where the respective datasets have different feature space given by number and arrangement of the features [50,51].

Transfer learning is categorized according to different situations involving the source and target domains. [5]. In the case of *inductive transfer*, the source task is different from the target task, regardless of the domain. Yao and Doretto [52] proposed an extension of the *AdaBoost* algorithm to address the inductive transfer learning problems where two new algorithms, *MultiSource-TrAdaBoost* and *TaskTrAdaBoost* were introduced for object category recognition and object detection. In *transductive learning*, the source and the target tasks are in different domains, whereas the tasks are considered to be the same. In this case, the source and target either have different feature space or they follow different distributions [5]. The *unsupervised transfer learning* methodologies address the unsupervised learning tasks in the target domain; such as clustering, dimensionality reduction, and density estimation [53,54]. Wang et al. [54] proposed an algorithm named transferred discriminative analysis that uses clustering to generate class labels for the target unlabeled data, and dimensionality reduction with prior labeled data for subspace selection. Pan et al. considered transfer learning via dimensionality reduction with a low-dimensional latent feature space, where the distributions be-

tween the source and the target domains are close and treated as a bridge of transferring knowledge [55].

Transfer learning is highly related to *multi-task learning* which features learning multiple related tasks simultaneously, through a shared knowledge representation, for improved generalization performance [56,57]. Multi-task learning has been used in applications that include pattern classification and computer vision [58,59]. In contrast to transfer learning [36,60], knowledge is exchanged among the related multiple tasks (which could be considered as source or target tasks). Multi-task learning focuses on improving the performance of all the tasks, whereas transfer learning focuses on enhancing the target task only. Transfer learning is considered one way transfer of knowledge, whereas multi-task learning could be seen as multi-directional transfer of knowledge between the tasks.

2.2. Bayesian neural networks

Bayesian inference could also be viewed as learning with rigorous uncertainty quantification [61–64]. In Bayesian inference, the *posterior probability* is derived from the prior probability and the 'likelihood' function which provides a means to evaluate the model given observed data. As noted earlier, the prior is based on belief or expert opinion about the model without viewing evidence or data [24,65]. In the case of neural networks, the priors can be informative about the distribution of the weights and biases given the network architecture and expert knowledge. An example is knowledge about the mechanism known as *weight decay* that states that having smaller weights is better for generalization [66]. Such information can be incorporated into the priors [24,64].

As outlined earlier, the progress in Bayesian neural networks has been highly dependent on MCMC sampling methods. This is a problem when considering larger neural network architectures and datasets, especially in the case of deep learning. A number of techniques have been applied for improving MCMC methods by incorporating approaches from the optimisation literature. Neal et al. [67] for instance, presented *Hamiltonian dynamics* that use gradient information to form efficient MCMC proposals during sampling. On the other hand, as mentioned earlier, gradient-based learning using *Langevin dynamics* refer to use of gradient information with stochastic noise [28]. Furthermore, methods from the optimization literature have enhanced Bayesian neural learning for time series forecasting [68–71]. In the case of classification problems, Wan provided a Bayesian interpretation for neural networks [63]. Moreover, Richard and Lippmann [62] used Bayesian neural networks for classification and presented proofs that showed that Bayesian probabilities are estimated when desired network outputs are multinomial. Chandra et al. presented Langevin gradient Bayesian neural networks with parallel tempering MCMC that utilized high performance computing for time series prediction and pattern classification problems [19].

3. Methodology

3.1. Transfer learning

A domain, denoted by $D = \{\ \chi, P(X)\ \}$, consists of two components; feature space χ and marginal probability distribution $P(X)$, where $X = \{x_1, x_2, x_3, \ldots, x_n\}, x_i \in \chi$.

A task, defined as $T = \{\ Y, f(.)\ \}$, consists of two components; a label space Y and an objective predictive function $f(.)$, which is to be learned for pair $\{\mathbf{x_i}, y_i\}$, where $\mathbf{x_i} \in X$ and $y_i \in Y$. The function $f(.)$ can be used to predict the corresponding label $f(\mathbf{x_i})$, of a new instance $\mathbf{x_i}$. From a probabilistic viewpoint, $f(\mathbf{x_i})$ can be written as $P(y_i|\mathbf{x_i})$. For instance, given a binary classification task such as face verification, y_i can denote the outcome.

In the transfer learning case, source domain can be denoted as $D_s = \{\{\mathbf{x_{s_1}}, y_{s_1}\}, \ldots, \{\mathbf{x_{s_n}}, y_{s_n}\}\}$, where $\mathbf{x_{s_i}} \in \chi_s$ is the set of input features and $y_{s_i} \in Y_s$ is the corresponding class label. Similarly, the target domain can be denoted as $D_t = \{\{\mathbf{x_{t_1}}, y_{t_1}\}, \ldots, \{\mathbf{x_{t_n}}, y_{t_n}\}\}$, where $\mathbf{x_{t_i}} \in \chi_t$ is the target input instance and $\mathbf{y_{t_i}} \in Y_t$ is the corresponding class label. In most cases, number of instances in target domain is lower than source domain; $t_n << s_n$.

We define transfer learning problem as: given a source domain D_s and learning task T_s, a target domain D_t and learning task T_t, transfer learning aims to improve the learning of the target predictive function $f_t(.)$ in D_t using the knowledge in D_s and T_s, where $D_s \neq D_t$ or $T_s \neq T_t$.

3.2. Feedforward neural network

Let $\boldsymbol{\theta} = (\tilde{\mathbf{w}}, \mathbf{v}, \boldsymbol{\beta}, \mu)$ that features a total of L weights and biases. The feedforward neural network topology with one hidden layer is defined by the number of input (I), hidden (H) and output (O) neurons, respectively. Given single input instance $\mathbf{x}$ with corresponding label $\mathbf{y}$ from dataset Ω, the network computes the output $f_o(\mathbf{x})$ for a neuron o in the output layer by

$$f_o(\mathbf{x}) = g\Big(\beta_o + \sum_{j=1}^{H} v_{jo} \times g\Big(\mu_j + \sum_{d=1}^{I} w_{dj} x_d\Big)\Big) \tag{1}$$

where, β and μ are the bias variables for the output and hidden layer, respectively. v_{jo} is the weight matrix which links the hidden layer to the output layer. w_{dj} is the weight matrix which links input $\mathbf{x}$ to the hidden layer. $g(.)$ is the sigmoid activation function, given by

$$g(z) = \frac{1}{(1 + \exp(-z))} \tag{2}$$

for the hidden and output layer units. Note that other activation functions can also be used.

In order to perform Bayesian neural learning, we define the likelihood function as the multivariate normal probability density function given by

$$p(\mathbf{y}|\boldsymbol{\theta}) = -\frac{1}{(2\pi\tau^2)^{S/2}} \times \exp\left(-\frac{1}{2\tau^2}\sum_{i\in S}(y_i - E(\mathbf{y_i}|\mathbf{x_i}))^2\right) \tag{3}$$

where, $E(\mathbf{y}|\mathbf{x})$ is given by Eq. (1) and S defines the size of the dataset Ω given by number of instances.

We assume that the elements of $\boldsymbol{\theta}$ are independent *a priori*. In addition, we assume *a priori* that the weights and biases have a normal distribution with zero mean and variance σ^2. Our prior is now

$$p(\boldsymbol{\theta}) \propto \frac{1}{(2\pi\sigma^2)^{L/2}} \times \exp\left\{-\frac{1}{2\sigma^2}\Big(\sum_{h=1}^{H}\sum_{d=1}^{D} w_{dh}^2 + \sum_{k=1}^{K}\sum_{h=1}^{H}(\delta_{hk}^2 + v_{hk}^2) + \delta_o^2\Big)\right\} \times \tau^{2(1+\nu_1)} \exp\left(\frac{-\nu_2}{\tau^2}\right). \tag{4}$$

The quantity τ^2 is defined for the regression problems only. For the case of classification, the prior distribution doesn't have any contribution from τ^2.

For classification problems with discrete data, it would be inappropriate to model the data as Guassian. Hence, for such discrete data with K possible classes, we assume that $\mathbf{y} = (y_1, \ldots, y_n)$ is generated from a multinomial distribution with parameter vector $\pi = (\pi_1, \ldots, \pi_K)$, where $\sum_{k=1}^{K} \pi_k = 1$. The multinomial likelihood function is formulated by taking the neural network's prediction

and comparison with given data as shown in following equation:

$$p(\mathbf{y}|\boldsymbol{\pi}) = \prod_{\mathbf{i=1}}^{\mathbf{S}} \prod_{\mathbf{k=1}}^{\mathbf{K}} \pi_{\mathbf{k}}^{\zeta_{\mathbf{i,k}}} \tag{5}$$

where π_k, the output of the neural network is the probability that data is generated by category k. The dependence between this probability and the input features $\mathbf{x}$ is modelled as a multinomial logit function given in following equation:

$$\pi_k = \frac{\exp(f_k(x)}{\sum_{j=1}^{K} \exp(f_j(x))} \tag{6}$$

where $f_k(x)$ is given by Eq. (1) and $\zeta_{i,k}$ is an indicator variable for the given instance i and the class k as given in the dataset. The indicator is defined by:

$$\zeta_{i,k} = \begin{cases} 1, & \text{if } y_i = k \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

In general, the log posterior is

$$\log\left(p(\boldsymbol{\theta}|\mathbf{y})\right) = \log\left(p(\boldsymbol{\theta})\right) + \log\left(p(\mathbf{y}|\boldsymbol{\theta})\right) \tag{8}$$

3.3. Bayesian neural transfer learning

A number of issues need to be considered for implementation of the MCMC sampler for Bayesian neural transfer learning that considers a number of *source* dataset(s) and a single *target* dataset as defined in the literature [5]. The number of MCMC chains in the algorithm is defined by the number of source(s) and target task in the problem of interest, as shown in Fig. 1. Each dataset that is associated to either the target of different source(s) is designated to a feedforward neural network topology which defines the training *task* as shown in Fig. 1. The number of input features and outcomes is same for all the respective tasks, but the number of instances may vary. The target chain is the designated MCMC chain which gets knowledge from the source chain(s). We implement the *transfer of knowledge* by sampling each MCMC chain of the source dataset(s) in a round-robin fashion depending on the *transfer-interval* γ which is user defined. Note that the method is also applicable given a single source dataset.

Algorithm 1 begins by the construction of the input features X_{s_m} for source task m, for $m = 1, \ldots, M$; where, M is the number source(s). X_t represents the input features for the target task. The goal is to obtain the posterior distribution of weights and biases given by θ_t for the target neural network architecture via MCMC sampling that features transfer of weights and biases (proposals) given by θ_{s_m} from the source task(s) to the target task. Given the source and target task knowledge, the parameters θ_{s_m}, for $m = 1, \ldots, M$ and θ_t are initialized. As shown in Fig. 1, i denotes the current position which is same for all the chains and the $i+1$th position is sampled. The sampling considers each source task m and the target task, $\theta_{s_m}^{[i+1]}$ and $\theta_t^{[i+1]}$. Using random-walk, we propose new values of $\theta_{s_m}^{[p]}$ and $\theta_t^{[p]}$ by drawing from a normal distribution centered at the current values, $\theta_{s_m}^{[i]}$ and $\theta_t^{[i]}$, respectively. We use a user-chosen *step-size* (standard deviation) denoted by σ_λ; hence, $\theta_{s_m}^{[p]} = \theta_{s_m}^{[i]} + \lambda_s$ and $\theta_t^{[p]} = \theta_t^{[i]} + \lambda_t$, where $\lambda_s, \lambda_t \sim \mathcal{N}(0, \sigma_\lambda)$. Afterwards, the *Metropolis-Hastings* acceptance probability is computed for the proposals (θ^p) in each chain (either Source or Target) by

$$\alpha = min\left\{1, \frac{p(\theta^p)p(y|\theta^p)}{p(\theta^{[i]})p(y|\theta^{[i]})}\right\} \tag{9}$$

If the proposed values are accepted for the sources $\theta_{s_m}^{[i+1]} = \theta_{s_m}^{[p]}$, otherwise $\theta_{s_m}^{[i+1]} = \theta_{s_m}^{[i]}$. Similarly, for the target task, if the proposed values are accepted $\theta_t^{[i+1]} = \theta_t^{[p]}$, otherwise $\theta_t^{[i+1]} = \theta_t^{[i]}$.

Algorithm 1: Bayesian neural transfer learning for multiple sources of data.

Data: Source and target datasets
Result: Posterior of weights and biases $p(\boldsymbol{\theta}|\mathbf{y})$
i. Define M number of source datasets Ψ, indexed by m.
ii. Define single target dataset, Φ.
iii. Define neural network for each source dataset $y_m = f(\theta_m, \Psi_m)$
iv. Define neural network for target dataset $\bar{y} = f(\theta, \Phi)$
v. Set parameters σ^2,ν_1, ν_2 for priors, Eq. (4)
vi. Set user-defined transfer interval γ, and maximum number of samples Γ as termination criterion.
while *each w until* Γ **do**
  for *each m until M* **do**
    for *each i until* γ **do**
      1. Propose set of weights and biases for the target task using random-walk, Φ: $\boldsymbol{\theta}_t^{[p]} = \theta_t^{[i]} + \eta$, where η is either Gaussian noise or use Langevin-gradient (Eq. (13))
      2. Calculate likelihood for the source Ψ_m given in Eq. (8)
      3. Obtain acceptance probability α, Eq. (9)
      4. Draw from uniform $u \sim \mathcal{U}[0, 1]$ and accept/reject the proposal:
      if $u < \alpha$ **then**
        Set $\theta_{s_m}^{[i+1]} = \theta_{s_m}^{[p]}$
        5. Increment number samples, w.
    end
  end
  * *Transfer knowledge (proposal) to target chain:*
  1. Select source m by drawing from uniform distribution, $m \sim \mathcal{U}\{0, 1 \ldots M\}$
  2. Using Metropolis-Hastings criterion, accept/reject proposal from source m. Draw from uniform distribution, $u \sim \mathcal{U}[0, 1]$ and calculate the acceptance ration using Eq. (10).
  if $u < r$ **then**
    Source chain proposal becomes part of target chain, $\theta_t^{[i+1]} = \theta_{s_m}^{[p]}$
  for *each i until* γ **do**
    1. Propose set of weights and biases for the target task using random-walk, Φ:$\boldsymbol{\theta}_t^{[p]} = \theta_t^{[i]} + \eta$, where η is either Gaussian noise or use Langevin-gradient (Eq. (13))
    2. Calculate likelihood for the target task, Φ using Eq. (9)
    3. Obtain acceptance probability α using Eq. (9)
    4. Draw from uniform $u \sim \mathcal{U}[0, 1]$ and accept/reject the proposal:
    if $u < \alpha$ **then**
      Set $\theta_t^{[i+1]} = \theta_t^{[p]}$
      5. Increment number samples, w.
  end
end

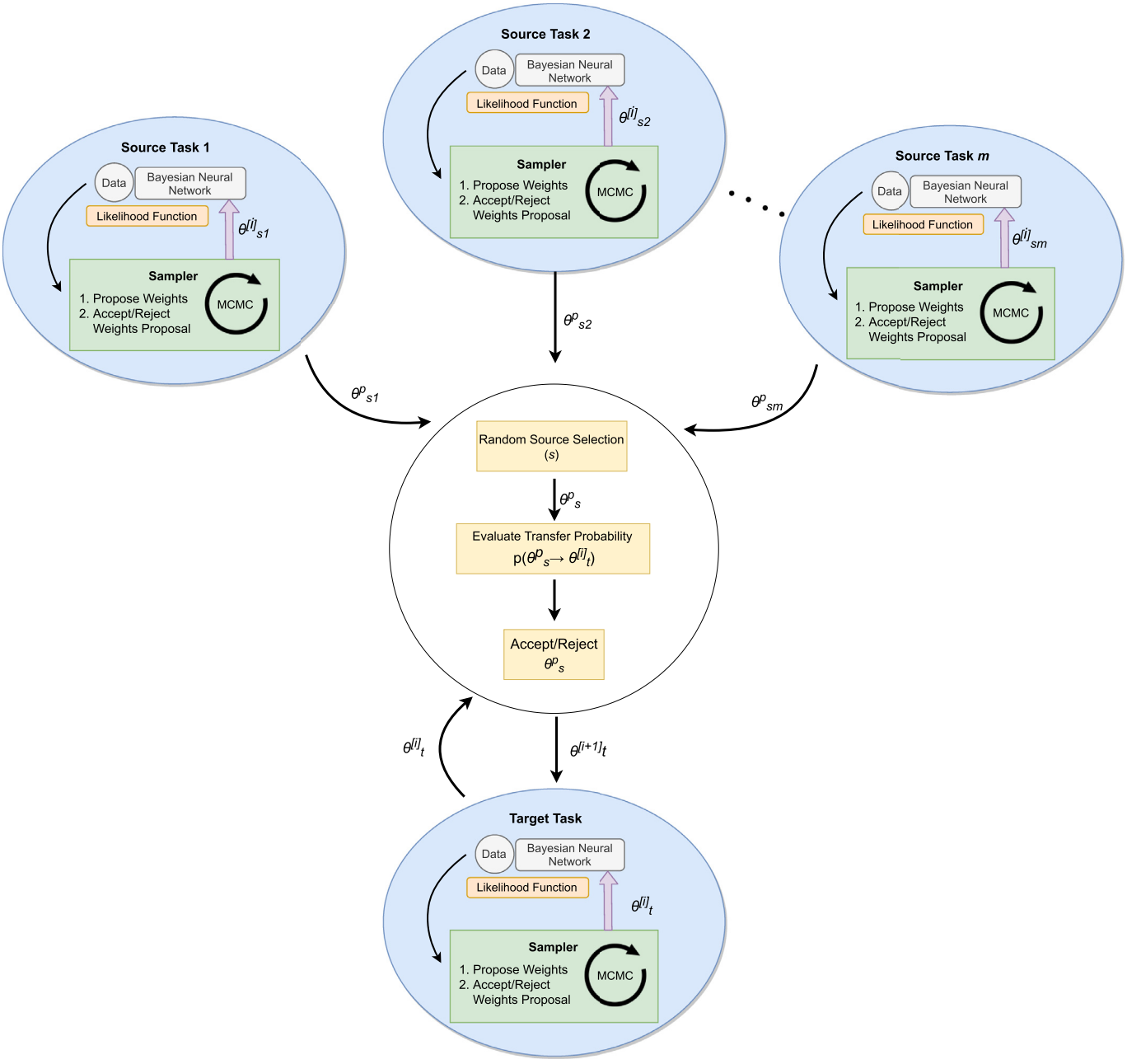


Fig. 1. Bayesian neural transfer learning via MCMC.

3.3.1. Transfer of knowledge

After the source task(s) are sampled as defined by the *transfer-interval*, we arbitrarily select a single source for transfer of knowledge. We either select or reject the transfer of source task proposal for the target task using Metropolis–Hastings acceptance ratio by

$$r = \frac{p(\theta_s^p|Y)q(\theta_t^{[i]}|\theta_s^p)}{p(\theta_t^{[i]}|Y)q(\theta_s^p|\theta_t^{[i]})} \tag{10}$$

where, $p(\theta_s^p|Y)$ and $p(\theta_t^{[i]}|Y)$ can be obtained with the help of Eqs. (4) and (8). Furthermore, $q(\theta_t^{[i]}|\theta_s^p) \sim \mathcal{N}(\theta_s^{[i]}, \sum_\theta)$ and $q(\theta_s^p|\theta_t^{[i]}) \sim \mathcal{N}(\theta_t^{[i]}, \sum_\theta)$.

$\sum_\theta = \sigma_\theta^2 I_L$ and I_L, are $L \times L$ identity matrix. Therefore, the transfer-interval determines when the network parameters from the source task is transferred to the target task, to be either accepted/rejected according to the Metropolis–Hastings criterion. Finally, the target neural network is sampled until the transfer-interval has been reached and then the process repeats considering the respective source tasks in a round-robin fashion.

This procedure continues until the *termination condition* is satisfied. The termination condition is defined by a maximum number of samples or when a predefined minimum criteria of prediction accuracy is reached by the Bayesian neural network.

3.4. Langevin gradient-based proposals

In order to enhance the efficiency of sampling, we employ Langevin gradient-based proposals which has been shown to be effective for Bayesian neural learning [29]. Langevin gradient-based proposals consider a single weight update via backpropagation that employs gradient descent with a small stochastic noise as proposal. This is slightly computationally expensive when compared to random-walk proposals; however, it can the performance as shown in previous work [30]. The Langevin gradient-based proposals is

Table 1
Description of datasets.

Dataset	Input	Output	Hidden	Source	Target	Samples
Wine	11	10	94	1	1	6000
Wifi	520	2	105	3	1	8000
SARCOS	21	1	25	1	1	4000
Synthetic	5	1	15	5	1	8000

given as follows.

$$\boldsymbol{\theta}^p \sim \mathcal{N}(\bar{\boldsymbol{\theta}}^{[i]}, \Sigma_\theta), \text{ where} \tag{11}$$

$$\bar{\boldsymbol{\theta}}^{[i]} = \boldsymbol{\theta}^{[i]} + r \times \nabla E_{\mathbf{y}}[\boldsymbol{\theta}^{[i]}],$$
$$E_{\mathbf{y}}[\boldsymbol{\theta}^{[i]}] = \sum_{t \in \mathcal{T}} (y_t - f(\mathbf{x}_t)^{[i]})^2,$$
$$\nabla E_{\mathbf{y}}[\boldsymbol{\theta}^{[i]}] = \left(\frac{\partial E}{\partial \theta_1}, \ldots, \frac{\partial E}{\partial \theta_L} \right) \tag{12}$$

r is the learning rate, So that the newly proposed value of $\boldsymbol{\theta}^p$, consists of 2 parts:

1. An gradient descent based weight update given by Eq. (12)
2. Add an amount of noise, from $\mathcal{N}(0, \Sigma_\theta)$.

The acceptance probability shown earlier in Eq. (9), for case of Langevin-gradients can be written as

$$\alpha = min\left\{1, \frac{p(\theta_s^p|Y)q(\theta^{[i]}|\theta^p)}{p(\theta^{[i]}|Y)q(\theta^p|\theta^{[i]})}\right\} \tag{13}$$

where, $p(\theta^p|Y)$ and $p(\theta^{[i]}|Y)$ can be obtained with the help of Eqs. (4) and (8). The value $q(\theta^p|\theta^{[i]})$ is defined as a shown in Eq. (11) and the proposal distribution is drawn from multivariate normal distribution, $q(\theta^{[i]}|\theta^p) \sim \mathcal{N}(\bar{\theta}^p, \sum_\theta)$, with $\bar{\theta}^p = \theta^p + r \times \nabla E_{\mathbf{y}}$.

4. Experiments and results

In this section, we present experiments and results that evaluate Bayesian neural transfer learning (BNTL) for multi-source transfer learning datasets. We show results for two strategies, that include random-walk (RW-BNTL) and Langevin-gradient (LG-BNTL) proposal distributions. The experiments are organised as follows

- Step 1: Evaluate the effect of the number of sources and the size of the task dataset using RW-BNTL;
- Step 2: Evaluate the effect the rate of Langevin-gradients in LG-BNTL for Synthetic dataset;
- Step 3: Compare RW-BNTL with LG-BNTL for the respective problems. Compare the results with canonical Bayesian neural learning using random-walk (BNL) that considers only the target dataset, to evaluate the performance via knowledge transfer;
- Step 4: Compare the results with related methods from the literature.

4.1. Experiment setting

In all the experiments, we use a single hidden layer feedforward neural network with sigmoid activation function in the hidden and output layers. Table 1 shows the network architecture and the number of source(s) for each problem. We select the value of transfer-interval $\gamma = 0.01 * N$; where N is the maximum number of samples per chain as shown in Table 1.

The proposals for θ for all the chains is drawn from a multivariate normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 0.02$. A normal distribution is also used to draw samples that define η with mean $\mu = 0$ and standard deviation $\sigma = 0.1$. These values gave acceptable results in trial runs and hence used in all the problems. The parameters for the prior given in Eq. (4) considered $\sigma^2 = 25$, $\nu_1 = 0$, $\nu_2 = 0$.

In all the experiments, the source dataset features the first five percent of the target data for training. The remaining ninety five percent of the target data is used as the test data. Table 1 shows the description of the network architecture used for each dataset and the number of sources used to train each target task for each problem. Note that software code in Python with the respective datasets for the given problems is provided online[1].

4.2. Synthetic dataset

We generate a regression dataset with multiple sources that considered five sources with a degree of similarity which was defined arbitrarily in comparison with the target task. We define a linear regression function $f(x) = w_0^T x + \epsilon$, where $w_0 \in \mathbb{R}^{100}$ and ϵ is a zero mean Gaussian noise. The target task considers learning the parameters of the linear regression model. We generate the target dataset with 500 points; out of which 50 are used for training and the remaining 450 are used for testing. The source tasks are defined by $g(x) = (w_0 + \delta(\Delta w))^T x + \epsilon$, where Δw is randomly generated vector and δ is the parameter controlling the difference in f and g (higher δ indicates lower similarity). We generate five source tasks with random δ values between [−1 and 1], such that each source task contains 500 data points. A similar dataset has been used in related papers in the literature [32,46].

4.3. Real world datasets

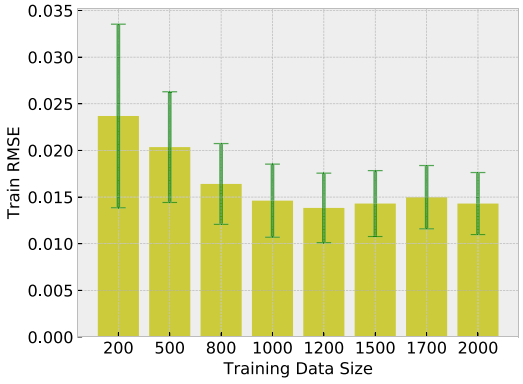
4.3.1. Wine quality

The task in the Wine quality dataset [72] is to predict the quality of white and red wine with a number of features that considered objective tests like *pH* value, sulphate content, etc. The quality of wine is given by experts with grades between 0 (poor quality) and 10 (best quality). The dataset contains 4898 records for white wine and 1599 records for red wine.[2] The quality estimation of white wine is taken as the source task, while the quality estimation of red wine is taken as the target task.

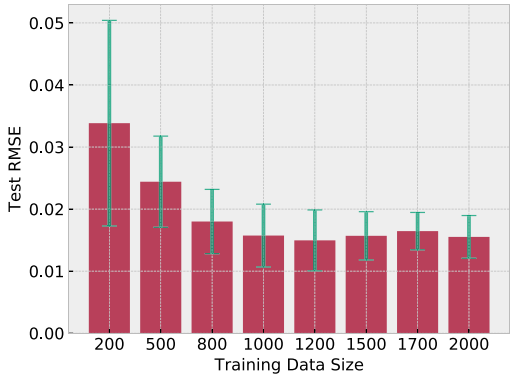
4.3.2. Wireless frequency access point: UJIndoorloc

The dataset UJIndoor [73] contains a collection of received signal strength (RSS) taken from the wireless frequency *(WiFi) access points* from three buildings.[3] Two of the buildings featured three floors each and one building featured a fourth floor. The task is to predict the location of the device based on the RSS values from 520 *WiFi access points*. The training data is taken as a set of (RSS values, location and the location of the device is given in terms of latitude and longitude values. There is a time difference between the train and test data. Thus, there exists a change in the distribution between the train data and the test data. For the *UJIndoorLoc* dataset, the outdated WiFi access-point data for each building is taken as the source task, while the new WiFi access-point data for the one of the buildings is taken as the target task. In WiFi location estimation, when we use the outdated data as the training data, the error can be very large. However, because the location information is constant across time, the knowledge from the outdated data can be transferred to the newer data. If this can be done successfully, we can save a lot of manual labeling effort for the new time period. Therefore, we wish to use the outdated data, which has more information as the source task to help predict the location for current signals.

[1] https://github.com/sydney-machine-learning/Bayesian-neural-transfer-learning.
[2] https://archive.ics.uci.edu/ml/datasets/wine+quality.
[3] https://archive.ics.uci.edu/ml/datasets/ujiindoorloc.

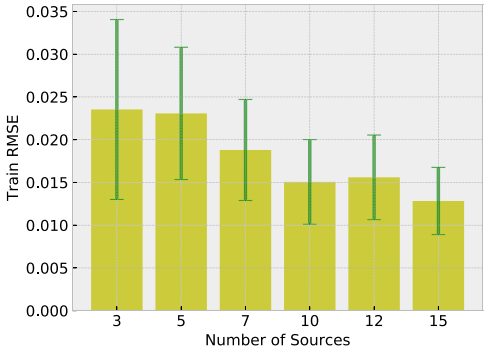


(a) Effect of training dataset size on target task

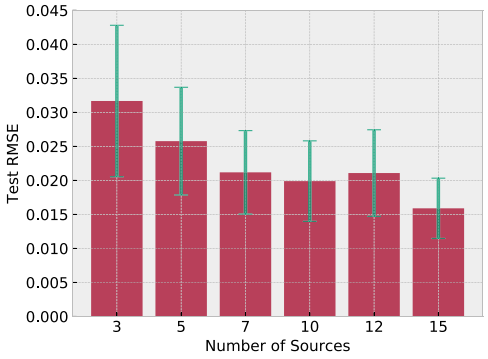


(b) Effect of test dataset size on target task

Fig. 2. An evaluation of performance accuracy (RMSE) on target task using synthetic dataset with varying source dataset size (number of instances in each source task) using RW-BNTL.



(a) Effect of number of sources on the target task train RMSE



(b) Effect of number of sources on the target task test RMSE

Fig. 3. An evaluation of performance accuracy (RMSE) on target task of synthetic dataset with different number of sources for training using RW-BNTL.

4.3.3. SARCOS robot arm

This dataset relates to an inverse dynamics for a seven degrees-of-freedom Sarcos anthropomorphic robot arm.[4] The data consists of 21-dimensional input space (7 joint positions, 7 joint velocities, 7 joint accelerations) and 7 joint torques. The task is to map the 21-dimensional orientation input to the corresponding torque values of 7 joints. Torque value estimation for each joint is taken as a separate task. We select one task as source task and another as the target task, thus forming the source-target pair. Since there are 7 torque estimation tasks, 42 such pairs are possible.

4.4. Results

We first present results regarding how the performance accuracy varies in terms of dataset size and the number of source tasks, for the synthetic dataset using RW-BNTL. For simplicity, in the first experiment, each source task features the same number of instances as the training dataset. The size of the target dataset is also initialised with the same number of data points as the source tasks, but as explained in Section 4.2, only 5% of the instances from the target dataset are used for training while remaining 95% of the instances are used for testing. The results for this experiment are presented in Fig. 2, which shows that the train the test performance accuracy on the target task improves (lower RMSE) with increase in the size of source task and the training data in the target task. The performance accuracy also improves with rise of the number of instances and the uncertainty of the model in the form of standard deviation also decreases with larger dataset. Moreover, we find that after a certain stage (1200 instances), there is not much change in the performance.

Next, we evaluate the performance with different number of sources using the synthetic dataset using RW-BNTL, and the results are shown in Fig. 3. We observe that there is improvement in the performance accuracy as the number of the source tasks increase.

Table 2 presents the results that show the effect of rate of applying the Langevin-gradients for LG-BNTL on the Synthetic dataset. We observe that as we increase the value of the Langevin rate (LG-Rate), the accuracy improves in terms of train and test RMSE. However, the improvement in accuracy comes at the expense of performance in terms of the computational time which increases significantly. Therefore, a trade off between time and accuracy is needed when selecting the parameters for experiments with large datasets.

Table 3 shows the results for the respective problems using *learning without transfer* for target only dataset (BNL-Target and LG-BNL-Target) and *learning with transfer* that considers the source and target datasets (BNTL and LG-BNTL). We observe that transfer learning with BNTL improves over BNL in all the cases, for both training and test datasets. The improvement in SARCOS case is minimal while the maximum improvement is shown for Wine and Synthetic dataset. Moreover, transfer learning with LG-BNTL shows the best performance in all the problems given both training and test datasets. We also observe that as the performance improves using the transfer learning methods, the percentage of accepted samples for the posterior reduces. This suggests that the method tends to accept more quality proposals when training with transfer,

[4] http://www.gaussianprocess.org/gpml/data/.

Table 2
Effect of rate of Langevin-gradient for LG-BNTL.

Problem	LG-Rate	Train (mean)	Train (std)	Test (mean)	Test (std)	Time (s)
Synthetic	0.1	0.01675	0.00869	0.01715	0.00881	733.80
	0.25	0.01496	0.00454	0.01638	0.00474	1027.98
	0.50	0.01144	0.00314	0.01249	0.00358	1379.78
	0.75	0.01085	0.00317	0.01173	0.00315	1812.30
	1.00	0.01031	0.00296	0.01125	0.00319	2126.31

Table 3
Performance accuracy (RMSE) on target tasks from different datasets.

Problem	Method	Train (mean)	Train (std)	Test (mean)	Test(std)	% Accepted
Wine Quality	BNL-Target-	0.55925	0.03829	0.56495	0.03720	93.14
	RW-BNTL	0.37193	0.01255	0.37884	0.01266	92.62
	LG-BNL-Target	0.51470	0.04797	0.51794	0.04850	65.01
	LG-BNTL	0.36498	0.04544	0.36777	0.04382	84.40
UJIndoorLoc	BNL-Target	0.15981	0.02315	0.21840	0.01424	32.75
	RW-BNTL	0.10111	0.04320	0.13446	0.04481	28.17
	LG-BNL-Target	0.16554	0.01838	0.24502	0.03239	23.24
	LG-BNTL	0.10497	0.02005	0.14198	0.02691	21.24
SARCOS	BNL (Target)	0.02692	0.00879	0.02757	0.00889	13.12
	RW-BNTL	0.02549	0.00477	0.02636	0.00482	9.62
	LG-BNL-Target	0.02242	0.00374	0.02239	0.00398	8.45
	LG-BNTL	0.01768	0.00250	0.01739	0.00247	8.05
Synthetic	BNL (Target)	0.05383	0.01654	0.08727	0.02901	71.30
	RW-BNTL	0.02107	0.00593	0.02266	0.00719	39.70
	LG-BNL-Target	0.04024	0.01252	0.04064	0.01222	54.67
	LG-BNTL	0.01294	0.00524	0.01301	0.00553	26.73

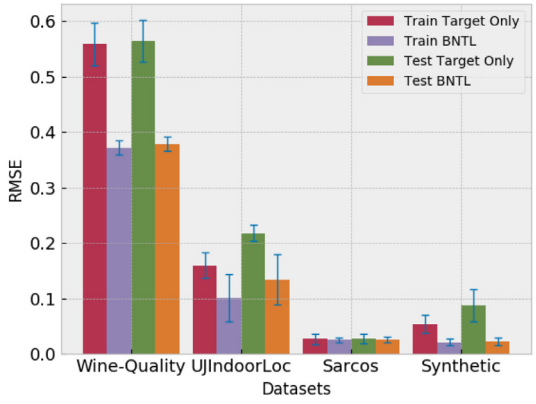
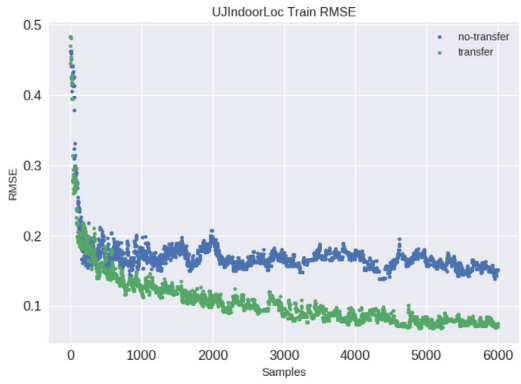
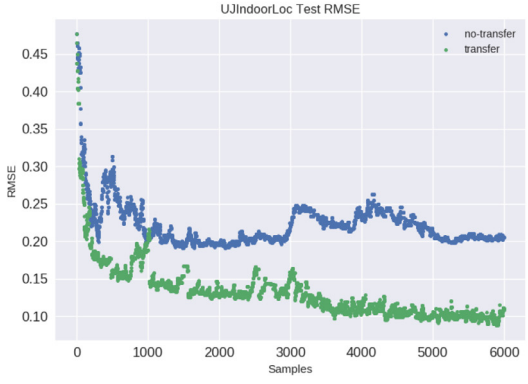


Fig. 4. Comparison of results showing performance accuracy (mean RMSE values) on different datasets with transfer and without transfer of knowledge via MCMC.



(a) Performance accuracy for the train dataset



(b) Performance accuracy for the test dataset

Fig. 5. Performance accuracy on train and test data of the target task data without transfer learning (no-transfer) and with transfer learning.

given more information from source datasets. Furthermore, with the Langevin gradients, the quality of the proposals improves and hence the percentage of proposals accepted further reduces, while the training and test performance is improved. Hence, it could be gathered that the performance of proposals accepted is related to the training and test performance. Fig. 4 gives further visualization of the results in Table 3.

We observe that RW-BNTL provider better convergence when compared to BNL without transfer (no-transfer), as shown by the RMSE of the posterior in Fig. 5 for the *UJIndoorLoc* dataset that highlights both training and test performance over the number of samples. The plots show convergence using performance accuracy over time (samples) given the train data for target only (training without transfer learning) and the transfer learning scenario. We notice that the RMSE plots for the target only and the transfer learning cases show that the target task with periodic transfer from the source task converges to a better solution. The difference in the convergence of the target task with and without transfer is more significant for the test data than the train data.

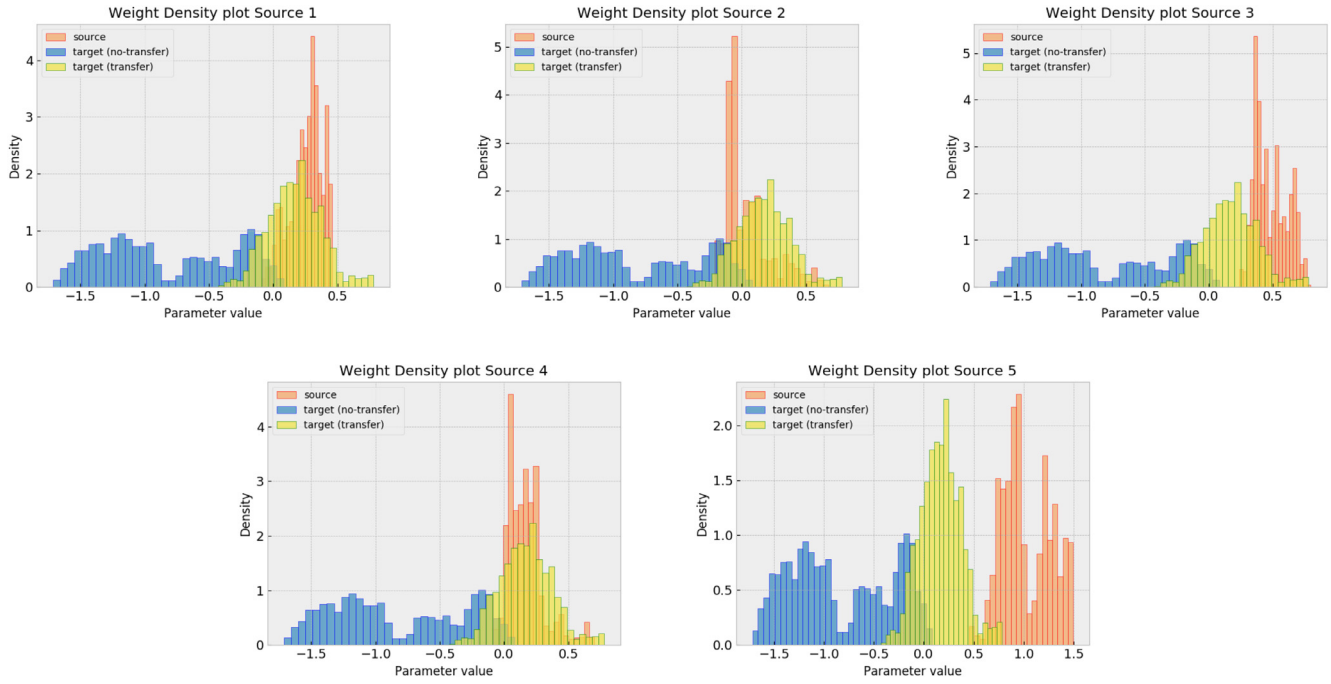


Fig. 6. Posterior distribution obtained for selected parameters (weights) between first input feature and first hidden node. Note that the case for transfer and no transfer learning is shown for the Bayesian neural network trained on SARCOS dataset.

Table 4
Results comparison with different transfer learning techniques. The standard deviation is given in brackets.

Dataset	BNL-Target	MTL-1 [35]	MTL-2 [44]	AT [32]	RW-BNTL	LG-BNTL
Wine Quality	1.81 (0.25)	1.69 (0.50)	1.27 (0.3)	1.16 (0.03)	1.47 (0.04)	1.31 (0.20)
SARCOS	0.33 (0.26)	0.24 (0.10)	0.26 (0.30)	0.18 (0.10)	0.20 (0.07)	0.18 (0.08)

Fig. 6 shows the posterior distribution for a selected weight connection (between first input unit and first hidden unit) of the Bayesian neural network for the SARCOS problem (BNL-Target vs BNTL) reported in Table 3. The posterior distribution for all the five source tasks in the dataset are compared with the posterior distribution for the target tasks (BNL-Target vs BNTL). It is clear that the mean of posterior distribution for the target dataset with transfer of knowledge (shown in yellow) is similar to one of the source datasets (shown in blue) for Source 1–4. We note that these distributions merely visualize the posterior without taking into account the prediction accuracy.

In Table 4, we compare the results of BNTL and LG-BNTL with other transfer learning algorithms taken from the literature for the same datasets. For the Wine and SARCOS dataset, we use NMSE (Normalised Mean Squared Error) as given in Eq. (14). For LG-BNTL, we use the LG-rate of 0.25 in the experiments.

$$NMSE = \frac{\sum_t^N (Y_t - O_t)^2}{\sum_t^N (Y_t - \overline{Y})^2} \tag{14}$$

where O_t is the predicted output from the neural network for input t, Y_t is the actual output and $\overline{Y}$ represents the mean of the actual output.

As seen in Table 4, the results of the proposed methodologies compares with transfer learning and multi-task learning algorithms for Gaussian Processes (MTL-1 [35], MTL-2 [44]) for the Wine Quality and SARCOS dataset. The results also show that NMSE values for LG-BNTL are much lower than BNTL. We note that both BNTL and LG-BNTL consider transfer of all the parameters θ to the target task and give acceptable results for these problems. The *adaptive transfer* (AT) algorithm [32] shows slightly better NMSE results for Wine Quality, but for SARCOS dataset, LG-BNTL gives similar results.

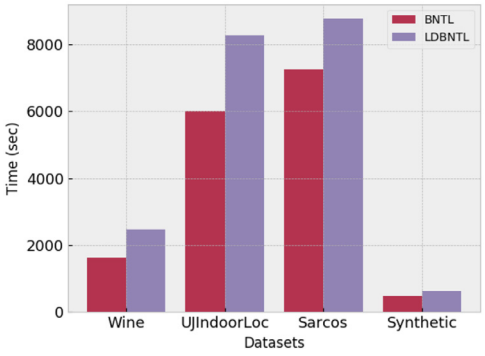


Fig. 7. A comparison of computational time between BNTL and LG-BNTL technique (where only up to 10% of samples use Langevin gradients).

Fig. 7 gives a snapshot of the computational time for the SARCOS dataset. We observe that the LG-BNTL takes significantly larger time, when compared to BNTL, due to the computational time used for calculating the gradients.

4.5. Discussion

In transfer learning, task relatedness, uncertainty quantification in model parameters and missing information in datasets pose major problems [5]. In cases when the target dataset is limited or very small in size, one needs to highly rely on source datasets for transfer learning. In such cases, even the test cases for the target dataset is small which means that there is greater need for uncertainty quantification while incorporating knowledge form the

source datasets. In the literature [5], the major focus of transfer learning has been to improve the performance of the test dataset in the target domain; however, this does not necessarily reflect real world cases where such test cases may also be limited. Hence, we presented a Bayesian transfer learning framework that probabilistically incorporates multiple sources of information in a rigorous fashion with uncertainty quantification. The goal of the proposed framework is not merely to improve the prediction performance, but also have a rigours framework for uncertainty quantification during learning and knowledge transfer.

We highlight that the number of posterior distributions (weights and biases) is large when compared to conventional problems, as in our previous work with time series prediction that considered less than hundred variables [29]. In this case, the number is in thousands (more than 50 thousand for Wine Quality problem) which poses problems for MCMC sampling methods in general. Hence, it is useful to incorporate Langevin gradient-based proposals which achieved better accuracy.

The sampling time would increase when the size of number of source datasets increases. This can be addressed with parallel and distributed computing and enhanced MCMC methods such as parallel tempering that naturally suits these computing environments. In Earth sciences, where model evaluation also poses a major computational problem, it has been shown that high performance computing with parallel tempering MCMC helps in alleviating the problem [19,74]. Furthermore, parallel tempering MCMC is useful for multimodal posterior distributions, which could arise from a number of source datasets and nature of problem at hand. The size of the dataset directly affects the network topology, in terms of the input and the hidden neurons which determine the number of weights and biases.

Given the advantage of using Langevin gradient based updates, the proposed framework could be extended to computer vision applications where transfer learning has been very useful. The framework could also incorporate dropouts which has shown to be a way for approximating Bayesian inference for deep learning [25]. This could be beneficial in providing a rigorous approach to uncertainty quantification for computer vision tasks which could be very useful in areas that require robust uncertainty quantification such as medical imaging [75].

5. Conclusions and future work

We presented a novel approach for transfer learning using a Bayesian framework that features a rigorous approach to uncertainty quantification during learning. The results shows that the framework is very promising for the transfer of knowledge from multiple sources to the target task. The approach incorporated Langevin gradients for MCMC proposals, which although is computationally expensive, improves performance accuracy. A balance between computational time and performance accuracy is needed for differed types of problems.

Future work can consider the proposed framework for different neural network architectures that includes recurrent and convolutional neural networks. In doing so, a number of challenges may occur since Bayesian inference has limitations given the size of the parameters, when number of posterior distributions increases drastically for larger models. Moreover, a wide range of applications in medicine that require rigorous form of uncertainty quantification could benefit from the framework.

Declaration of Competing Interest

The authors declare that they do not have any financial or non-financial conflict of interests.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.neucom.2019.10.042.

References

[1] Q. Yang, X. Wu, 10 Challenging problems in data mining research, Int. J. Inf. Technol. Decision Making 5 (04) (2006) 597–604.
[2] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, S.Y. Philip, et al., Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2008) 1–37.
[3] L.Y. Pratt, J. Mostow, C.A. Kamm, A.A. Kamm, Direct transfer of learned information among neural networks, in: Proceedings of AAAI-91, Citeseer, 1991.
[4] L.Y. Pratt, Discriminability-based transfer between neural networks, in: Proceedings of the Advances in Neural Information Processing Systems, 1993, pp. 204–211.
[5] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.
[6] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, G. Zhang, Transfer learning using computational intelligence: a survey, Knowl. Based Syst. 80 (2015) 14–23.
[7] D. Cook, K.D. Feuz, N.C. Krishnan, Transfer learning for activity recognition: a survey, Knowl. Inf. Syst. 36 (3) (2013) 537–556.
[8] A. Ahmed, K. Yu, W. Xu, Y. Gong, E. Xing, Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks, in: Proceedings of the European Conference on Computer Vision, Springer, 2008, pp. 69–82.
[9] R. Gopalan, R. Li, R. Chellappa, Domain adaptation for object recognition: an unsupervised approach, in: Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV), IEEE, 2011, pp. 999–1006.
[10] W. Pan, E. Zhong, Q. Yang, Transfer learning for text mining, in: Mining Text Data, Springer, 2012, pp. 223–257.
[11] M.E. Taylor, P. Stone, Transfer learning for reinforcement learning domains: a survey, J. Mach. Learn. Res. 10 (Jul) (2009) 1633–1685.
[12] R. Raina, A. Battle, H. Lee, B. Packer, A.Y. Ng, Self-taught learning: transfer learning from unlabeled data, in: Proceedings of the Twenty-forth International Conference on Machine Learning, ACM, 2007, pp. 759–766.
[13] D. Das, C.G. Lee, Sample-to-sample correspondence for unsupervised domain adaptation, Eng. Appl. Artif. Intell. 73 (2018) 80–91, doi:10.1016/j.engappai.2018.05.001.
[14] N. Courty, R. Flamary, D. Tuia, A. Rakotomamonjy, Optimal transport for domain adaptation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (9) (2016) 1853–1865.
[15] O. Sener, H.O. Song, A. Saxena, S. Savarese, Learning transferrable representations for unsupervised domain adaptation, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 2110–2118.
[16] G.E. Box, G.C. Tiao, Bayesian Inference in Statistical Analysis, 40, John Wiley & Sons, 2011.
[17] G. Chamberlain, G.W. Imbens, Nonparametric applications of Bayesian inference, J. Busin. Econ. Stat. 21 (1) (2003) 12–18.
[18] J. Geweke, Bayesian inference in econometric models using monte carlo integration, Econometr. J. Econometr. Soc. (1989) 1317–1339.
[19] R. Chandra, R.D. Müller, D. Azam, R. Deo, N. Buttersworth, T. Salles, S. Cripps, Multi-core parallel tempering Bayeslands For basin and landscape evolution, *Geochemistry, Geophysics, Geosystems.* 2019. doi:10.1029/2019GC008465.
[20] J. Pall, R. Chandra, D. Azam, T. Salles, J.M. Webster, S. Cripps, BayesReef: a Bayesian inference framework for modelling reef growth in response to environmental change and biological dynamics, 2018. arXiv preprint arXiv:1808.02763.
[21] R. Scalzo, D. Kohn, H. Olierook, G. Houseman, R. Chandra, M. Girolami, S. Cripps, Efficiency and robustness in monte carlo sampling for 3-d geophysical inversions with obsidian v0. 1.2: setting up for success, Geosci. Model Dev. 12 (7) (2019) 2941–2960.
[22] D.J. MacKay, A practical bayesian framework for backpropagation networks, Neural Comput. 4 (3) (1992) 448–472.
[23] D.J. MacKay, Hyperparameters: Optimize, or Integrate Out? in: Maximum Entropy and Bayesian Methods, Springer, 1996, pp. 43–59.
[24] R.M. Neal, Bayesian Learning for Neural Networks, 118, Springer Science & Business Media, 2012.
[25] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 1050–1059.
[26] M. Girolami, B. Calderhead, Riemann manifold Langevin and hamiltonian monte carlo methods, J. R. Stat. Soc. Ser. B Stat. Methodol. 73 (2) (2011) 123–214.
[27] G.O. Roberts, J.S. Rosenthal, Optimal scaling of discrete approximations to Langevin diffusions, J. R. Stat. Soc. Ser. B Stat. Methodol. 60 (1) (1998) 255–268.
[28] M. Welling, Y.W. Teh, Bayesian learning via stochastic gradient Langevin dynamics, in: Proceedings of the Twenty-eighth International Conference on Machine Learning (ICML-11), 2011, pp. 681–688.
[29] R. Chandra, L. Azizi, S. Cripps, Bayesian neural learning via Langevin dynamics for chaotic time series prediction, in: D. Liu, S. Xie, Y. Li, D. Zhao, E.-S. M. El-Alfy (Eds.), Proceedings of the Neural Information Processing, Springer International Publishing, Cham, 2017, pp. 564–573.

[30] R. Chandra, K. Jain, R.V. Deo, S. Cripps, Langevin-gradient parallel tempering for Bayesian neural learning, Neurocomputing 359 (2019) 315–326 In this issue, doi:10.1016/j.neucom.2019.05.082.
[31] R. Luis, L.E. Sucar, E.F. Morales, Transfer learning for Bayesian networks, in: H. Geffner, R. Prada, I. Machado Alexandre, N. David (Eds.), Proceedings of the Advances in Artificial Intelligence – IBERAMIA 2008, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 93–102.
[32] B. Cao, S.J. Pan, Y. Zhang, D.-Y. Yeung, Q. Yang, Adaptive transfer learning, 2010.
[33] A. Karbalayghareh, X. Qian, E.R. Dougherty, Optimal Bayesian transfer learning, IEEE Trans. Signal Process. 66 (14) (2018) 3724–3739, doi:10.1109/TSP.2018.2839583.
[34] D.M. Roy, L.P. Kaelbling, Efficient Bayesian task-level transfer learning., in: Proceedings of the IJCAI, 7, 2007, pp. 2599–2604.
[35] N.D. Lawrence, J.C. Platt, Learning to learn with the informative vector machine, in: Proceedings of the Twenty-first International Conference on Machine Learning, ACM, 2004, p. 65.
[36] A. Schwaighofer, V. Tresp, K. Yu, Learning gaussian process kernels via hierarchical bayes, in: Proceedings of the Advances in Neural Information Processing Systems, 2005, pp. 1209–1216.
[37] E.T. Jaynes, Prior probabilities, IEEE Trans. Syst. Sci. Cybern. 4 (3) (1968) 227–241.
[38] H.K.H. Lee, Priors for neural networks, in: D. Banks, F.R. McMorris, P. Arabie, W. Gaul (Eds.), Proceedings of the Classification, Clustering, and Data Mining Applications, 2004, pp. 141–150.
[39] C.W. Bray, Transfer of learning, J. Exp. Psychol. 11 (6) (1928) 443.
[40] D.N. Perkins, G. Salomon, et al., Transfer of learning, Int. Encyclopedia Educ. 2 (1992) 6452–6457.
[41] T.T. Baldwin, J.K. Ford, Transfer of training: a review and directions for future research, Pers. Psychol. 41 (1) (1988) 63–105.
[42] D.L. Hall, J. Llinas, An introduction to multisensor data fusion, Proc. IEEE 85 (1) (1997) 6–23.
[43] B. Khaleghi, A. Khamis, F.O. Karray, S.N. Razavi, Multisensor data fusion: a review of the state-of-the-art, Inf. Fus. 14 (1) (2013) 28–44, doi:10.1016/j.inffus.2011.08.001.
[44] E.V. Bonilla, K.M. Chai, C. Williams, Multi-task gaussian process prediction, in: Proceedings of the Advances in Neural Information Processing Systems, 2008, pp. 153–160.
[45] Y. Zheng, Methodologies for cross-domain data fusion: an overview, IEEE Trans. Big Data 1 (1) (2015) 16–34, doi:10.1109/TBDATA.2015.2465959.
[46] P. Wei, R. Sagarna, Y. Ke, Y.-S. Ong, C.-K. Goh, Source-target similarity modelings for multi-source transfer Gaussian process regression, in: Proceedings of the International Conference on Machine Learning, 2017, pp. 3722–3731.
[47] Z. Xu, S. Sun, Multi-source transfer learning with multi-view adaboost, in: Proceedings of the International Conference on Neural Information Processing, Springer, 2012, pp. 332–339.
[48] Y. Yao, G. Doretto, Boosting for transfer learning with multiple sources, in: Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2010, pp. 1855–1862.
[49] L. Duan, I.W. Tsang, D. Xu, T.-S. Chua, Domain adaptation from multiple sources via auxiliary classifiers, in: Proceedings of the Twenty-sixth Annual International Conference on Machine Learning, ACM, 2009, pp. 289–296.
[50] X. Ling, W. Dai, G.-R. Xue, Q. Yang, Y. Yu, Spectral domain-transfer learning, in: Proceedings of the Forteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2008, pp. 488–496.
[51] Q. Yang, Y. Chen, G.-R. Xue, W. Dai, Y. Yu, Heterogeneous transfer learning for image clustering via the social web, in: Proceedings of the Joint Conference of the Forty-seventh Annual Meeting of the ACL and the Firth International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1 ACL '09, Association for Computational Linguistics, Stroudsburg, PA, USA, 2009, pp. 1–9.
[52] Y. Yao, G. Doretto, Boosting for transfer learning with multiple sources, in: Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, pp. 1855–1862, doi:10.1109/CVPR.2010.5539857.
[53] W. Dai, Q. Yang, G.-R. Xue, Y. Yu, Self-taught clustering, in: Proceedings of the Twenty-fifth International Conference on Machine Learning, ACM, 2008, pp. 200–207.
[54] Z. Wang, Y. Song, C. Zhang, Transferred dimensionality reduction, in: Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2008, pp. 550–565.
[55] S.J. Pan, J.T. Kwok, Q. Yang, Transfer learning via dimensionality reduction., in: Proceedings of the AAAI, 8, 2008, pp. 677–682.
[56] R. Caruana, Multitask learning, Mach. Learn. 28 (1) (1997) 41–75.
[57] T. Evgeniou, C.A. Micchelli, M. Pontil, Learning multiple tasks with kernel methods, J. Mach. Learn. Res. 6 (Apr) (2005) 615–637.
[58] H. Zheng, X. Geng, D. Tao, Z. Jin, A multi-task model for simultaneous face identification and facial expression recognition, Neurocomputing 171 (2016) 515–523.
[59] T. Zeng, S. Ji, Deep convolutional neural networks for multi-instance multi-task learning, in: Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM), 2015, pp. 579–588.
[60] K. Yu, V. Tresp, A. Schwaighofer, Learning gaussian processes from multiple tasks, in: Proceedings of the Twenty-second International Conference on Machine Learning, ACM, 2005, pp. 1012–1019.
[61] D.F. Specht, Probabilistic neural networks, Neural Netw. 3 (1) (1990) 109–118.
[62] M.D. Richard, R.P. Lippmann, Neural network classifiers estimate Bayesian a posteriori probabilities, Neural Comput. 3 (4) (1991) 461–483.
[63] E.A. Wan, Neural network classification: a Bayesian interpretation, IEEE Trans. Neural Netw. 1 (4) (1990) 303–305.
[64] D.J. MacKay, Probable networks and plausible predictionsâa review of practical Bayesian methods for supervised neural networks, Netw. Comput. Neural Syst. 6 (3) (1995) 469–505.
[65] B. Cheng, D.M. Titterington, Neural networks: a review from a statistical perspective, Stat. Sci. (1994) 2–30.
[66] A. Krogh, J.A. Hertz, A simple weight decay can improve generalization, in: Proceedings of the Advances in Neural Information Processing Systems, 1992, pp. 950–957.
[67] R.M. Neal, et al., Mcmc using hamiltonian dynamics, Handb. Markov Chain Monte Carlo 2 (11) (2011).
[68] F. Liang, Bayesian neural networks for nonlinear time series forecasting, Stat. Comput. 15 (1) (2005) 13–29.
[69] O. Kocadağlı, B. Aşıkgil, Nonlinear time series forecasting with Bayesian neural networks, Expert Syst. Appl. 41 (15) (2014) 6596–6610.
[70] D.T. Mirikitani, N. Nikolaev, Recursive Bayesian recurrent neural networks for time-series modeling, IEEE Trans. Neural Netw. 21 (2) (2010) 262–274.
[71] H.S. Hippert, J.W. Taylor, An evaluation of bayesian techniques for controlling model complexity and selecting inputs in a neural network for short-term load forecasting, Neural Netw. 23 (3) (2010) 386–395.
[72] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, Decis. Support Syst. 47 (4) (2009) 547–553.
[73] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J.P. Avariento, T.J. Arnau, M. Benedito-Bordonau, J. Huerta, Ujiindoorloc: a new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems, in: Proceedings of the 2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN), IEEE, 2014, pp. 261–270.
[74] M. Sambridge, A parallel tempering algorithm for probabilistic sampling and multimodal optimization, Geophys. J. Int. 196 (1) (2013) 357–374.
[75] T. Pun, G. Gerig, O. Ratib, Image analysis and computer vision in medicine, Comput. Med. Imaging Graph. 18 (2) (1994) 85–96.

Dr. Rohitash Chandra holds a Ph.D. in Computer Science (2012) from Victoria University of Wellington, M.S. in Computer Science (2008) from the University of Fiji and B.Sc. in Computer Science and Engineering Technology (2006) at the University of the South Pacific. Dr. Rohitash Chandra is currently USyd Research Fellow at the Centre for Translational Data Science and School of Geosciences, University of Sydney, Australia. His research interests are in areas of deep learning, neuro-evolution, Bayesian methods, solid Earth Evolution, reef modelling and mineral exploration.

Arpit Kapoor holds a Bachelor of Engineering (2019) from SRM Institude of Technology, Chennai, India. He is currently a Machine Learning Developer and his research interests include deep learning and Bayesian neural networks.