UNCERTAINTY-GUIDED LIFELONG LEARNING IN BAYESIAN NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The ability to learn in a setting where tasks arrive in a sequence without access to previous task data is difficult for learning algorithms when restricted in capacity. In this lifelong learning setting a single model is challenged to learning a new task, while at the same time not forgetting about previous tasks and freeing up capacity for future tasks. We argue that the ability to identify network parameters which are most critical for a learned task plays a critical role to decide which ones to remember. In this work we propose to rely on Bayesian Networks, which inherently model the distribution of a parameter rather than a single value of a parameter. More specifically, we formulate lifelong learning in the Bayesian-by-Backprop framework, exploiting the parameter uncertainty for two lifelong learning directions. First, weight pruning, where a hard selection is made on which parameters to select per task and, second, weight regularization which can be seen as a softer version to keep important parameters, respectively. We show the benefit of our approach using diverse object classification datasets in both cases.

1 INTRODUCTION

Humans can easily accumulate and maintain the knowledge gained from previously observed tasks and continuously learn to solve new problems or tasks. Inspired by this capability of humans, Lifelong Learning (LLL) intends to learn sequential tasks without forgetting leaned concepts as new tasks are being introduced. Unlike the powerful capabilities observed in mammalian brains in acquiring and preserving the knowledge, artificial agents still struggle in continual learning.

One of the main challenges for a lifelong learning agent is known as *catastrophic forgetting* (McCloskey & Cohen, 1989; McClelland et al., 1995) which refers to the significant drop in the performance of a learner when switching from a trained task to a new one. This phenomenon occurs because trained parameters on initial task require to change in favor of learning new objectives. This is the reason that naive finetuning intuitively suffers the most from *catastrophic forgetting*. Given limited capacity of a network, one way to see this problem as to deciding which part of network is best to keep and which part is less important and can thus be used by future tasks without significantly effecting current task performance. In this work we propose to turn to Bayesian Networks as introduced by Blundell et al. (2015). These networks represent each parameter with a distribution defined by a mean and variance which more naturally allows to determine which parameters are most important for a task. We integrate this idea with two different LLL paradigms: Hard pruning of part of the network and soft regularization based importance weighting of network parameters.

To avoid adding capacity when learning new tasks, one can first prune or compress the network (Liu et al., 2015; Aghasi et al., 2017) and then learn new tasks using free parameters as proposed by Mallya & Lazebnik (2017; 2018). Here a part of the network is frozen, while the remaining network is free to adapt for future tasks. While this eliminates forgetting over time, it reduces flexibility when learning large number tasks as the network fills up over time as the percentage which is frozen by previous tasks increases over time.

The second paradigm we study is regularization based LLL where forgetting is reduced by introducing a loss which penalizes when *important* weight parameters are changed while learning new tasks. In contrast to the hard pruning this is "softer" way to avoid forgetting. The main challenge for the approaches in this domain is how to define and measure importance. Different techniques have explored different quantities to measure importance with Kirkpatrick et al. (2017); Zenke et al.

(2017); Aljundi et al. (2017). While we go over more details about these methods in section 4.2 we argue that the externally defined importance definition used in these woks might yield sub-optimal regularization performance. Moreover, importance computation in these methods requires an extra computational step (whether online or offline). Despite of the difficulties in defining and measuring importance, these approaches have the benefit of good scalability as well as high efficiency.

Most of the work in lifelong learning have been proposed within the realm of non-Bayesian models where model parameters tend to overconfidently fit to training data and there is no internal definition of weight importance. In contrast, Bayesian models, enhanced by their statistically-grounded theoretical framework, provide clear definitions of uncertainty in model weights and/or predictions. Although exact Bayesian inference remains intractable in neural networks, variational approximations have provided a powerful alternative to this problem. In this work we exploit an efficient back-propagation-compatible Bayesian formulation in neural networks, called Bayes-by-Backpop (BBB) Blundell et al. (2015), to obtain weights' uncertainty as a guidance to perform continual learning. We show how lifelong learning using pruning and regularization can be enhanced when accurate weights importance are present to identify task-specific parameters. The main intuition behind is that Bayesian neural network provide precise uncertainty measurements in weights which is the bottleneck for regularization methods.

2 Related Work

Here we list some of the previously introduced approaches taken in lifelong learning that mitigate catastrophic forgetting to different extents. Conceptually, these approaches can be divided into the following categories. We list pros and cons for each and explain why (not) our approach is inspired or build upon them.

Dynamic architectural methods: In this setting the architecture will grow such that it carries over the past knowledge as well as storing new gained knowledge in different forms such as additional layers, nodes, or dual-architecture. In this approach the objective function remains fixed whereas the model capacity grows often exponentially with the number of tasks. Progressive networks Rusu et al. (2016); Schwarz et al. (2018) was one of the earliest work in this path which was successfully applied on reinforcement learning problems where the base architecture is duplicated and lateral connections are added in response to new tasks. Roy et al. (2018) also uses hierarchical network, where CNNs at multiple levels, could grow in tree-shape manner to accommodate new tasks. Dynamically Expandable Network (DEN) Yoon et al. (2018) also expands the network by selecting the *drifting* units and retraining them separately based on the current tasks in hand. Despite the ability of these methods in preventing forgetting, they have large overheads upon arrival of new tasks. We have completely avoided using architectural growth in our method due to this reason.

Memory-based methods: In this regime, previous information is partially stored in different fashions to be used later as a form of *rehearsal* Robins (1995). GeppNet Gepperth & Karaoguz (2016) and its variant GeppNet-STM were introduced in this category for class incremental learning settings. They use a self-organizing map (SOM), a linear regression layer, to re-structure the input data topographically. Once the SOM is trained, it will be only updated when a an input belong to a new class arrive. This example is stored by a short-term memory buffer to prevent forgetting it in the future. iCaRL Rebuffi et al. uses this idea to perform class incremental learning through nearest-mean-of-exemplars using current data and stored examples of previously seen classes. It was shown to be capable of incrementally learning large scale datasets such as ImageNet. However, this method can not be used in learning multiple sequential tasks from different datasets. Gradient episodic memory (GEM) Lopez-Paz et al. (2017) was another work for class incremental learning where data is stored at the end of each episode to be used later to restrict gradient updates from deviating away from their previous values. GEM is not only able to mitigate forgetting, but also allows for positive backward knowledge transfer which is referred to improvement on previously learned tasks. Another value of this work was that it was the first method capable of learning with seeing each training example only once. Since storing examples can be costly and sometimes unpractical, a generative model was employed to encode/decode pseudo information Draelos et al. (2017); Shin et al. (2017); Nguyen et al. (2017). However, in practice generative models are difficult to train specially in sequential learning scenarios.

Regularization methods: These approaches do not require storing data while retaining the whole network. Catastrophic forgetting is alleviated via preventing significant changes to representation learned for previous tasks. This can be performed through regularizing the objective function, or directly applied on weight parameters such that important weights to one task are not allowed to change in the future. Various methods have been explored in both weight *importance* measurements as well as regularization techniques. Elastic weight consolidation (EWC) method Kirkpatrick et al. (2017) imposes a quadratic penalty on changes in weights that are found to be important to prior tasks. However, the importance is measured through computing the diagonal of the Fisher information metric over parameters of the old task resulting in inefficiency on high-dimensional output spaces. Synaptic Intelligence (SI) suggested an online way of computing importance which had comparable results to EWC while being more efficient. Similar to SI, Memory-aware Synapses (MAS) Aljundi et al. (2017) proposed an online way of computing importance adaptive to the test set using the change in the model outputs w.r.t the inputs. PackNet Mallya & Lazebnik (2017) used iterative pruning to fully restrict gradient updates on important weights via saving binary masks. While applying this binary mask at inference results in zero forgetting of the initial performance, it requires to know which task we are performing to use the appropriate mask. PackNet also ranks the weight importance by their magnitude which is not guaranteed to be a proper importance indicative. A follow-up work to PackNet Mallya & Lazebnik (2018) suggested learning these binary masks end-to-end using a meta-learner.

3 Background: Variational Bayes-by-Backprop

In this section we review the Bayesian framework we use to be able to learn sequential tasks by remembering task-specific parameters. The Bayesian formulation used here was first introduced by Blundell et al. (2015) that learns the probability distribution over network parameters. In the original paper, it was shown that this novel back-propagation-compatible algorithm acts as a regularizer and yields comparable performance to dropout on MNIST dataset. In the next section (Section 4) we show how to use the derived weight uncertainty to measure the importance in performing the task in hand and use the rest of the network to learn later tasks with.

3.1 Variational inference in neural networks

In Bayesian models, latent variables are drawn from a prior density $p(\mathbf{w})$ which are related to the observations through the likelihood $p(x|\mathbf{w})$. During the inference, the posterior distribution $p(\mathbf{w}|\mathbf{x})$ is computed conditioned on the given input data. However, in practice this probability distribution is intractable and is often estimated through approximate inference. Markov chain Monte Carlo (MCMC) sampling (Hastings, 1970) has been widely used and explored for this purpose (see Robert & Casella (2013) for different methods under this category). However, MCMC algorithms, despite providing guarantees in finding asymptotically exact samples from the target distribution, they are not suitable for large datasets and/or large models as they are bounded by speed and scalability issues. Alternatively, Variational inference provides a faster solution to the same problem in which the posterior is approximated using optimization rather than being sampled from a chain Peterson (1987); Hinton & Van Camp (1993); Jaakkola & Jordan (1996; 1997). Similar to any other approximation algorithms, there will be trade-offs between the precision of the results and the efficiency of the optimization. However, variational inference methods always take advantage of fast optimization techniques such as citestochastic methods, distributed methods, which allows them to explore data models faster to find the best approximation. See Blei et al. (2017) for a complete review of the theory.

3.2 Bayes by Backprop

Let $\mathbf{x} \in \mathbb{R}^n$ be a set of observed variables and $\mathbf{w}$ be a set of latent variables parametrized by θ sharing a joint density of $p(\mathbf{w}, \mathbf{x})$. Given the observations $\mathbf{x}$ and by using the set of weight parameters $\mathbf{w}$, a neural network, as a probabilistic model, $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$ can output a probability associated with a possible output $\mathbf{y} \in \mathcal{Y}$. Variational inference aims to calculate this conditional probability distribution over the latent variables by finding the closest proxy to the exact posterior through solving an optimization problem.

We first propose a family $\mathcal{D}$ of probability densities over the latent variables ($q(\mathbf{w}|\theta) \in \mathcal{D}$). We then find the closest member of this family to the true conditional probability of interest $p(\mathbf{w}|\mathbf{x})$ by minimizing the Kullback-Leibler (KL) divergence between q and p:

$$q^*(\mathbf{w}|\theta) = \underset{q(\mathbf{w}|x)\in\mathcal{D}}{\arg\min}\ \mathrm{KL}\big(q(\mathbf{w}|\theta)\|p(\mathbf{w}|\mathbf{x})\big) \tag{1}$$

Once solved, $q(\cdot)$ would be the closest approximation to variational posterior parametrized by θ^* to construct the parameters of the network. We can unpack Eq. 1 and call it a loss function commonly known as variational free energy or expected lower bound :

$$\mathcal{L}_{BBB}(\mathbf{w},\theta) = \mathrm{KL}\big[q(\mathbf{w}|\theta)\|p(\mathbf{w})\big] - \mathbb{E}_{q(\mathbf{w}|\theta)}\big[\log(p(\mathbf{x}|\mathbf{w})\big] \tag{2}$$

Blundell et al. (2015) showed that Eq. 2 can be approximated using i Monte Carlo samples from the variational posterior:

$$\mathcal{L}_{BBB}(\mathbf{w},\theta) \approx \sum_{i=1}^{n} \log q(\mathbf{w}^{(\mathbf{i})}|\theta) - \log \mathbf{p}(\mathbf{w}^{(\mathbf{i})}) - \log p(\mathbf{x}|\mathbf{w}^{(i)}) \tag{3}$$

We assume $q(\mathbf{w}|\theta)$ to have a Gaussian pdf with diagonal covariance. θ is parametrized point-wise with a mean μ and standard deviation of $\sigma = \log(1+\exp(\rho))$. Variational posterior parameters will have a form of $\mu + \log(1+\exp(\rho)) \circ \epsilon$ where ϵ is a sample drawn from posited q. For the prior, as it was suggested in Blundell et al. (2015), a scale mixture of two Gaussian pdfs are chosen which are zero-centered while having difference variances of σ_1^2 and σ_2^2. The uncertainty obtained for every weight parameter, as opposed to hidden units Rezende et al. (2014); Kingma & Welling (2013) has been successfully used in model compression Blundell et al. (2015), uncertainty-based exploration in reinforcement learning application Blundell et al. (2015); Lipton et al. (2016). In this work we want to use this framework to learn sequential tasks without forgetting using per-weight derived uncertainties.

4 Lifelong learning using Bayes by Backprop (B-LLL)

Unlike regular deep neural networks, in BBB model, weight parameters are represented by probability distributions parametrized by their mean and standard deviation. In order to take into account both mean and variance we use the quantity of signal-to-noise ratio (SNR) for each parameter defined as

$$\mathrm{SNR} = \frac{|\mu|}{\sigma} \tag{4}$$

SNR is a commonly used measure in signal processing to distinguish between "useful" information from unwanted noise contained in a signal. In the context of neural models, SNR can be thought as an indicative of parameter importance; the higher the SNR, the more effective or important the parameter is to the model predictions for a given task.

In our method called, Bayesian Lifelong Learning (B-LLL), we have considered two common scenarios of lifelong learning: 1) sequential learning of multiple tasks; 2) class-incrementally learning of a single dataset. We perform continual learning using weight pruning and weight regularization based upon our newly defined pruning criterion and parameter importance measurement leveraged from Bayes by backprop algorithm.

4.1 Weight pruning

Weight pruning is a long standing solution to reduce inference computation or network compression Han et al. (2016); Liu et al. (2017); Molchanov et al. (2016). Moreover, it has been recently used to perform lifelong learning Mallya & Lazebnik (2017), where the goal is to gradually using a single network's capacity to learn multiple tasks with. This is done by freeing up parameters deemed to be unimportant to the current task. Forgetting is prevented in pruning by saving a task-specific binary mask of important vs. unimportant parameters. However, parameter importance measurement in Mallya & Lazebnik (2017) is performed based on parameters' magnitude only. Hence, our work is significantly different from Mallya & Lazebnik (2017) in the different criterion we propose to

Algorithm 1 Bayesian Lifelong Learning with Pruning (B-LLL Prune)

Require: hyper parameters for BBB only: σ, σ_1, σ_2, ρ, μ, π, α, n
Require: hyper parameter for additional pruning step: p ▷ p := pruning portion per layer

$\epsilon \sim \mathcal{N}(0, I)$
while not done **do**
  $l_1 \leftarrow \sum_{i=1}^{n} \log \mathcal{N}(\mathbf{w}_i|\mu, \sigma^2)$ ▷ log-posterior
  $l_2 \leftarrow \sum_{i=1}^{n} \log \left(\pi\mathcal{N}(\mathbf{w}_i \mid 0, \sigma_1^2) + (1-\pi)\mathcal{N}(\mathbf{w}_i \mid 0, \sigma_2^2)\right)$ ▷ log prior
  $l_3 \leftarrow \sum_{i=1}^{n} \log(p(\mathbf{x}|\mathbf{w}))$ ▷ log-likelihood of the data
  $\mathcal{L}_{BBB} \leftarrow \frac{1}{M}(l_1 - l_2) - l_3$ ▷ M := number of minibatches
  Compute $\nabla\mathcal{L}_\mu$ and $\nabla\mathcal{L}_\rho$
  $\text{SNR}_i \leftarrow \frac{|\mu_i|}{\sigma_i}$ ▷ signal-to-noise ratio for every weight
  for all layers **do**
    for all parameters in layer **do**
      if $\text{SNR}_i \in$ Top p portion of SNR values in layer **then**
        $\nabla\mathcal{L}_\mu = 0$ ▷ No parameter update
        $\nabla\mathcal{L}_\rho = 0$ ▷ No parameter update
      else
        $\mu' \leftarrow \mu - \alpha\nabla\mathcal{L}_\mu$ ▷ Update parameter using gradient descent
        $\rho' \leftarrow \rho - \alpha\nabla\mathcal{L}_\rho$ ▷ Update parameter using gradient descent

use for importance measurement; the statistically-grounded uncertainty defined in Bayesian neural networks.

BBB-pruning is performed as follows: for every layer, convolutional or fully-connected, the parameters are ordered by their SNR value and those with lowest importance are pruned (set to zero). The pruned parameters, are "saved" using a binary mask so that they can be used later in learning new tasks whereas the important parameters remain fixed throughout training on future tasks. Once a task is learned, an associated binary mask is saved which will be used at inference to recover key parameters to the desired task. The overhead memory caused by saving the binary mask (less than 20MB for ResNet18), is negligible given the fact it completely eliminates the forgetting. To make a fair comparison with Mallya & Lazebnik (2017) we only used weight parameters (not biases) to be candidates for removal.

4.2 Weight regularization

This method can be thought as a *softer* version of weight pruning and has been previously used in continual learning settings Kirkpatrick et al. (2017); Zenke et al. (2017); Aljundi et al. (2017). The key idea in regularization is to preserve the representation learned for one task by minimizing the change in important weight parameters, instead of freezing them as previously done in pruning. Therefore, in regularization, gradient updates are allowed for all parameters, however the goal is to condition the updates on the weights importance. As a result, detecting important parameters plays an important role in better regularization. We propose to use the per weight uncertainty afforded in BBB again to regulate the changes in parameters using their importance. Regularization, in comparison to pruning, does not require memory for saving the binary parameter selection mask. However, forgetting is possible to occur because all parameters are allowed to change.

Considering neural network training from a probabilistic perspective, we aim at finding their most probable values given the data $\mathcal{D}$ with $p(\theta|\mathcal{D})$. In life-long learning where $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots \mathcal{D}_T\}$, $\mathcal{D}_{1:T-1}$ represents the tasks learnt till the last time step, and $\mathcal{D}_T$ is the task being trained at the current time step T. Hence, we can compute the conditional probability $p(\theta|\mathcal{D})$ from the prior probability of the parameters $p(\theta|\mathcal{D}_{1:T-1})$ and the probability of the data $p(\mathcal{D}_T|\theta)$ by using Bayes' rule

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}_T|\theta) + \log p(\theta|\mathcal{D}_{1:T}) - \log p(\mathcal{D}_{1:T-1}) \propto \log p(\mathcal{D}_T|\theta) + \log p(\theta|\mathcal{D}_{1:T}) \quad (5)$$

where $\log p(\mathcal{D}_T|\theta)$ is simply the negative of the loss function for the current task and $\log p(\theta|\mathcal{D}_{1:T})$ is the prior over the parameters that represents *forgetting*.

Algorithm 2 Bayesian Lifelong Learning with Regularization (B-LLL Reg)

Require: hyper parameters for BBB-only: σ, σ_1, σ_2, ρ, μ, π, α, n, λ
Require: hyper parameters for additional regularization: λ

$\epsilon \sim \mathcal{N}(0, I)$
$\sigma = \log(1 + \exp(\rho))$ ▷ Ensures σ is always positive
$\mathbf{w} = t(\theta, \epsilon) = \mu + \sigma \circ \epsilon$ ▷ $\mathbf{w}$:= a posterior sample of weights
while not done **do**
  $l_1 \leftarrow \sum_{i=1}^{n} \log \mathcal{N}(\mathbf{w}_i | \mu, \sigma^2)$ ▷ log-posterior
  $l_2 \leftarrow \sum_{i=1}^{n} \log \left(\pi \mathcal{N}(\mathbf{w}_i \mid 0, \sigma_1^2) + (1-\pi)\mathcal{N}(\mathbf{w}_i \mid 0, \sigma_2^2)\right)$ ▷ log prior
  $l_3 \leftarrow \sum_{i=1}^{n} \log(p(x|\mathbf{w}))$ ▷ log-likelihood of the data
  $\mathcal{L}_{BBB} \leftarrow \frac{1}{M}(l_1 - l_2) - l_3$ ▷ M := number of minibatches
  $\mathcal{L}_{\text{forgetting}} \leftarrow \lambda \sum_i \frac{1}{\sigma_i}(\mu_i - \mu_i^*)^2 + \lambda \sum_i \frac{1}{\sigma_i}(\rho_i - \rho_i^*)^2$
  $\mathcal{L} \leftarrow \mathcal{L}_{BBB} + \mathcal{L}_{\text{forgetting}}$
  $\mu' \leftarrow \mu - \alpha \nabla \mathcal{L}_\mu$ ▷ Update all parameters using gradient descent
  $\rho' \leftarrow \rho - \alpha \nabla \mathcal{L}_\rho$ ▷ Update all parameters using gradient descent

Modeling Forgetting using BBB ($p(\theta|\mathcal{D}_{1:T})$) For tractability, we approximate the posterior as a Gaussian distribution with mean given by the parameters $\theta^*_{1:T-1}$ and a diagonal precision given by the diagonal of that represent the standard deviation of that distribution. Typically, this measure is engineered to represent the importance of each parameter, the higher the standard deviation, the less important the parameter is. In MAS Aljundi et al. (2017), parameters with highest importance are those with highest sensitivity to the prediction of the neural network. In EWC Kirkpatrick et al. (2017), important parameters are those to have the highest in terms if the fisher information metric. In Int.Synapses, this parameter importance notion is engineered to correlate with the the loss function, parameters that contributes more to the loss are more important. In contrast since our approach is Beysian, the training procedures directly provides the uncertainty of each parameter which directly model $p(\theta|\mathcal{D}_{1:T})$. Concretely, for continual learning using BBB with regularization, we start by training a regular BBB network in the task at hand. Once a task is finished, per-weights' $\frac{1}{\sigma}$ values are computed. Hence, the total objective function for Bayesian Lifelong Learning with regularization is as follows:

$$\mathcal{L}_{B-LLL}(\mathbf{w}, \theta) = \mathcal{L}_{\text{BBB}} + \lambda \sum_i \frac{1}{\sigma_i}(\mu_i - \mu_i^*)^2 + \lambda \sum_i \frac{1}{\sigma_i}(\rho_i - \rho_i^*)^2 \tag{6}$$

where $\mathcal{L}_{\text{BBB}} = \log p(\mathcal{D}_T|\theta)$ is the bayesian learning loss of the current task, λ is the regularization parameter, and $\sum_i \frac{1}{\sigma_i}(\mu_i - \mu_i^*)^2 = p(\theta|\mathcal{D}_{1:T}) + \sum_i \frac{1}{\sigma_i}(\rho_i - \rho_i^*)^2$. Note that, we included in our prior not to deviate from the learnt uncertrainty, encouraging the certainty not to decrease over time. In all our experiments λ was considered to be 10^{-5}.

Complete algorithm for BBB with regularization is shown in Algorithm 3.

5 Experimental Setup

Datasets: We evaluate our approach B-LLL on sequential learning of multiple datasets as well as class incrementally learning of single datasets. We have chosen five commonly used fine-grained classification vision datasets (see Table 1) to assess both techniques.

Training details: It is important to note that in all our experiments, **no pre-trained ImageNet model** is used. All trainings start from random initialization for the first task including all results computed for baselines under this condition. Hence, they might differ from the originally reported values due to this reason. ImageNet dataset Deng et al. (2009) consists of 1000 object categories and includes the main objects of our utilized fine-grained datasets (flowers, birds, cars, aircrafts) which in principal lowers the forgetting effect between tasks because of the existing overlaps between them.

In all the experiments we resized images to 224×224. We used random horizontal flipping for data augmentation. Each dataset is also normalized using channel-wise mean and standard deviation of its own. B-LLL-related hyperparameters including are all computed per dataset using a validation

Table 1: Utilized datasets descriptions with B-LLL-related hyper parameters

Name	Training	Test	# Classes	B-LLL Hyperparameters				
				ρ	σ_1	σ_2	π	n
Oxford Flowers Nilsback & Zisserman (2008)	2040	6149	102	-3	0	6	0.25	1
MIT Indoors Scenes Quattoni & Torralba (2009)	5360	1340	67	-3	0	6	0.75	1
Caltech-UCSD Birds 2011 Wah et al. (2011)	5994	5794	200	-3	0	6	0.75	1
Stanford Cars Krause et al. (2013)	8144	8041	196	-3	1	8	0.75	2
Stanford Aircrafts Maji et al. (2013)	3334	3333	70	-3	0	6	0.75	1

set of 0.1 size of its training set using a grid search on originally given values in Blundell et al. (2015) and are given in Table 1.

The network architecture used in all experiments is ResNet18. However, because there exists double number of parameters in a BBB network compared to its equivalent regular neural net, we ensured fair comparison by reducing the number of filters or neurons in the Bayesian network by half. Due to the existing regularization nature of BBB algorithm, we have not used dropout in our Bayesian ResNet18.

We used gradient descent in all experiments with learning rate of 0.01 and momentum factor of 0.9 decaying by a factor of 0.5 upon being plateau for 15 epochs. Batch size of 10 was consistently used throughout all experiments.

5.1 Compared methods

Naive Finetuning Our simplest baseline is consecutively training 5 datasets by finetuning all layers after finishing a task.

Individual networks This baseline trains a separate randomly initialized network for each task resulting in the highest possible performance for that task. This baseline serves as an upper bound for performance and a lower bound in forgetting.

PackNet: This serves as the closest baseline to access B-LLL-pruning. We have used the original implementation of this method Mallya & Lazebnik (2017) but the network is changed to ResNet18 with no pretrained imagenet model. The accordingly altered hyperparameters are all given in the appendix.

We measure backward transfer, BWT, (inverse forgetting) and evaluate accuracy, ACC, in class incremental learning as follows:

$$\text{BWT} = \frac{1}{T-1}\sum_{i=1}^{T-1} R_{i,T} - R_{i,i}, \quad \text{ACC} = \frac{1}{T}\sum_{i=1}^{T} R_{i,T} \tag{7}$$

where $R_{i,j}$ is the test classification accuracy on task i after finishing training on task j.

6 Results

Table 2 shows the results for sequential learning of multiple datasets using B-LLL-pruning in comparison to fine-tuning and PackNet Mallya & Lazebnik (2017). B-LLL-Prune and PackNet are are pruned by fractions that causes performance drops of less than 1%. Note that it is always possible to re-train after pruning if performance drop is significant. Here we did not find it necessary as the focus is on detecting important task-specific parameters in a model and learn more with the rest of them. We have also shown the maximum achievable performance for each task using a randomly initialized individual network for both PackNet and B-LLL-Prune. B-LLL-Prune consistently outperforms PackNet across all tasks achieving 18% better average classification accuracy of 45.84% and 46.29%, before and after pruning, respectively. We believe this is due to the correct selection of parameter importance using uncertainty instead of magnitude.

We used the same datasets presented in Table 3 and incrementally learned them by dividing each dataset into multiple tasks ranging from 10 to 49 (see Table 1 for number of classes per dataset). Averaged accuracy of the final model evaluated on all previous tasks and backward transfer (Eq. 7)

Table 2: Sequential learning of 5 fine-grained classification datasets using B-LLL-pruning compared against PackNet Mallya & Lazebnik (2017). Note that all networks are trained from scratch. * indicates that performance can only be achieved when saving a network for each task, breaking the scalability. Bold indicates the best performance for a single network.

		T1: Flowers	T2: Scenes	T3: CUBS	T4: Cars	T5: Aircrafts	Mean
PackNet	Prune %	60%	40%	40%	40%	40%	
	Pre-prune *	69.04	33.73	36.42	38.93	39.51	43.53
	Post-prune	68.51	33.66	36.43	38.90	39.51	37.40
B-LLL-Prune (Ours)	Prune %	60%	40%	40%	40%	40%	
	Pre-prune *	73.38	37.91	38.39	40.43	41.34	46.29
	Post-prune	**72.48**	**37.99**	**37.97**	**39.87**	**40.89**	**45.84**
Individual regular networks *		69.04	48.78	49.34	69.76	70.89	61.56
Individual Bayesian networks *		73.38	50.08	49.39	68.20	73.27	62.87

Table 3: Class incremental learning of five fine-grained classification benchmarks using B-LLL-Reg with two importance measurement criteria compared with naive fine-tuning in BBB

		Flowers	Scenes	CUBS	Cars	Aircrafts
Method	Performance	17 Tasks	13 Tasks	20 Tasks	49 Tasks	10 Tasks
BBB finetune	ACC	84.75	63.34	72.32	73.20	39.43
(Blundell et al., 2015)	BWT	-39.06	-26.43	-25.44	-16.70	-28.98
B-LLL-Reg (ours)	ACC	**85.75**	63.54	75.65	74.44	42.54
with $\lvert\mu\rvert/\sigma$	BWT	-36.98	-24.22	-24.12	-15.23	-23.45
B-LLL-Reg (ours)	ACC	85.16	**67.76**	**78.44**	**78.57**	**43.81**
with $1/\sigma$	BWT	**-36.77**	**-21.33**	**-20.43**	**-11.32**	**-22.86**

are presented to measure performance and forgetting. We have also considered using SNR, which our pruning criterion was based on, to measure the importance in the regularization setting. B-LLL-Reg method compared to naive finetuning performs consistently better across all datasets, suggesting that restricting the parameters that were deemed to be important, successively preserved the learned representation for earlier tasks while allowing the parameters to change in favor of learning new classes within the same distribution. Results for different measurements of importance confirm our hypothesis on. Table 3 shows that using $1/\sigma$ as a notion of importance outperforms using $SNR = \mu/\sigma$ due to two reasons. First, $1/\sigma$ directly respects the Bayesian distribution until $T-1$ as a prior where the uncertainty of each parameter directly interprets its importance. Second, having μ is close to zero implies low importance according to SNR and hence these weights will be allowed to change under our B-LLL-reg approach. In pruning, changing these weights does not effect prior tasks since these weights are masked out for earlier tasks (i.e., older task predictions are not affected). However, in B-LLL-reg, changing close to zero weights would affect the predictions of earlier tasks especially those with high certainty (i.e., low σ).

7 Conclusion

In this work we propose a lifelong learning formulating for Bayesian networks, called B-LLL, that uses the Bayesian-embedded uncertainty predictions as a guidance to perform lifelong learning; i.e. to identify task-specific important parameters that can be either fully preserved (through a saved binary mask) or allowed to change conditioned on their importance (as a regularizer) for learning new tasks. We demonstrated how the probabilistic uncertainty distributions per weights are helpful in two lifelong learning approaches: weight pruning and weight regularization applied on multiple fine-grained classification datasets learned sequentially and class incrementally.

REFERENCES

Alireza Aghasi, Afshin Abdi, Nam Nguyen, and Justin Romberg. Net-trim: Convex pruning of deep neural networks with performance guarantee. In *Advances in Neural Information Processing Systems*, pp. 3180–3189, 2017.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. *arXiv preprint arXiv:1711.09601*, 2017.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. Ieee, 2009.

Timothy J Draelos, Nadine E Miner, Christopher C Lamb, Jonathan A Cox, Craig M Vineyard, Kristofor D Carlson, William M Severa, Conrad D James, and James B Aimone. Neurogenesis deep learning: Extending deep networks to accommodate new classes. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pp. 526–533. IEEE, 2017.

Alexander Gepperth and Cem Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 8(5):924–934, 2016.

Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. *arXiv preprint arXiv:1607.04381*, 2016.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13. ACM, 1993.

Tommi Jaakkola and Michael Jordan. A variational approach to bayesian logistic regression models and their extensions. In *Sixth International Workshop on Artificial Intelligence and Statistics*, volume 82, pp. 4, 1997.

Tommi S Jaakkola and Michael I Jordan. Computing upper and lower bounds on likelihoods in intractable networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pp. 340–348. Morgan Kaufmann Publishers Inc., 1996.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, pp. 201611835, 2017.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081*, 2016.

Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 806–814, 2015.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 2755–2763. IEEE, 2017.

David Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. *arXiv preprint arXiv:1711.05769*, 1(2):3, 2017.

Arun Mallya and Svetlana Lazebnik. Piggyback: Adding multiple tasks to a single, fixed network by learning to mask. *arXiv preprint arXiv:1801.06519*, 2018.

James L McClelland, Bruce L McNaughton, and Randall C O'reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *CoRR, abs/1611.06440*, 2016.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, pp. 722–729. IEEE, 2008.

Carsten Peterson. A mean field theory learning algorithm for neural networks. *Complex systems*, 1:995–1019, 1987.

Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–420. IEEE, 2009.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

Deboleena Roy, Priyadarshini Panda, and Kaushik Roy. Tree-cnn: A deep convolutional neural network for lifelong learning. *arXiv preprint arXiv:1802.05800*, 2018.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pp. 2990–2999, 2017.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks, 2018.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *arXiv preprint arXiv:1703.04200*, 2017.