Uncertainty-Guided Continual Learning in Bayesian Neural Networks – Extended Abstract

Sayna Ebrahimi  
UC Berkeley  
sayna@eecs.berkeley.edu

Mohamed Elhoseiny  
Facebook AI Research  
mohamed.elhoseiny@gmail.com

Trevor Darrell  
UC Berkeley  
trevor@eecs.berkeley.edu

Marcus Rohrbach  
Facebook AI Research  
maroffm@gmail.com

Abstract

Continual learning aims to learn new tasks without forgetting previously learned ones. This is especially challenging when one cannot access data from previous tasks and when the model has a fixed capacity. Current regularization-based continual learning algorithms need an external representation and extra computation to measure the parameters’ importance. In contrast, we propose Bayesian Continual Learning (BCL), where the learning rate adapts according to the uncertainty defined in the probability distribution of the weights in networks. We evaluate our BCL approach extensively on diverse object classification datasets with short and long sequences of tasks and report superior or on-par performance compared to existing approaches. Additionally we show that our model can be task-independent at test time, i.e. it does not presume knowledge of which task a sample belongs to.

1. Introduction

Humans can easily accumulate and maintain can knowledge gained from previously observed tasks, and continuously learn to solve new problems or tasks. Artificial learning systems typically forget prior tasks when they cannot access all training data at once but are presented with task data in sequence. Overcoming these challenges is the focus of continual learning.

Catastrophic forgetting [12] 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 the initial task change in favor of learning new objectives. Given a network of limited capacity, one way to address this problem is to identify the importance of each parameter and penalize further changes to those parameters that were deemed to be important for the previous tasks [6, 1, 19].

Bayesian neural networks [2] propose an intrinsic importance model based on weight uncertainty. These networks represent each parameter with a distribution defined by a mean and variance over possible values drawn from a shared latent probability distribution. Variational inference can approximate posterior distributions using Monte Carlo sampling for gradient estimation. These networks act like ensemble methods in that they reduce the prediction variance but only use twice the number of parameters present in a regular neural network.

We propose Bayesian Networks for continual learning, and develop a new method which exploits the inherent measure of uncertainty therein to adapt the learning rate of individual parameters. Second, we present a hard-threshold variant of our method that decides which parameters to freeze. Third, we validate our approach experimentally, comparing it to prior art both on single datasets split into different tasks, as well as for the more difficult scenario of learning a sequence of different datasets. Forth, in contrast to most prior work, our approach does not rely on knowledge about task boundaries at inference time, which humans do not need and might not be always available. We show that our approach naturally supports this scenario, sometimes also referred to as a “single head” scenario for all tasks. We refer to evaluation metric of a “single head” model without task information at test time as “generalized accuracy”.

1.1. Background: Variational Bayes-by-Backprop

Let $x \in \mathbb{R}^n$ be a set of observed variables and $w$ be a set of latent variables. A neural network, as a probabilistic model $p(y|x,w)$, given a set of training examples $D = (x,y)$ can output $y$ which belongs to a set of classes by using the set of weight parameters $w$. We first assume
a family of probability densities over the latent variables $w$ parametrized by $\theta$, i.e., $q(w|\theta)$. We then find the closest member of this family to the true conditional probability of interest $p(w|x,y)$ by minimizing the Kullback-Leibler (KL) divergence between $q$ and $p$:

$$q^*(w|\theta) = \arg\min_{q} \text{KL}(q(w|\theta)||p(w|x))$$ (1)

Eq. 1 is commonly known as variational free energy or expected lower bound:

$$\mathcal{L}(\theta, D) = \text{KL} [q(w|\theta)||p(w)] - \mathbb{E}_{q(w|\theta)} \left[ \log(p(y|x,w)) \right]$$ (2)

[2] showed that Eq. 2 can be approximated using $N$ Monte Carlo samples from the variational posterior:

$$\mathcal{L}(\theta, D) \approx \sum_{i=1}^{N} \log q(w^{(i)}|\theta) - \log p(y^{(i)}|x,w^{(i)})$$ (3)

We assume $q(w|\theta)$ to have a Gaussian pdf with diagonal covariance and parametrized by $\theta = (\mu, \rho)$. A sample weight of the variational posterior can be obtained by sampling from a unit Gaussian and reparametrized by $\mu + \sigma \odot \epsilon$ where $\odot$ is a pointwise multiplication. Standard deviation is parametrized as $\sigma = \log(1 + \exp(\rho))$ and thus is always non-negative.

2. Bayesian Continual Learning Neural Networks

A common strategy to perform continual learning is to reduce forgetting by regularizing further changes in the model representation based on parameters’ importance. In this section we introduce our Bayesian Continual Learning approach (BCL), which exploits estimated uncertainty of the parameters’ posterior distribution to regulate the change in certain/uncertain parameters. BCL regulates the change of certain/important parameters in a soft way such that the learning rate of each parameter and hence its gradient update becomes a function of its importance.

**Uncertainty-defined importance** We use the well-defined uncertainty in parameters distribution, i.e., standard deviation, as a notion of importance. In particular we scale the learning rate for each parameter proportional to its uncertainty to reduce changes in certain parameters while allowing uncertain parameters to alter in favor of learning new tasks.

**Parameter regularization in BCL.** Learning rate regularization means that the learning rate is adapted per parameter according to an importance measure $\Omega$ as shown in the following equations

$$\mu' = \mu - \Omega_{\mu} \alpha \nabla \mathcal{L}_{B B B_{\mu}}$$

$$\rho' = \rho - \Omega_{\rho} \alpha \nabla \mathcal{L}_{B B B_{\rho}}$$ (4, 5)

where $\alpha$ is the learning rate and $\nabla \mathcal{L}_{B B B_{\mu}}$ and $\nabla \mathcal{L}_{B B B_{\rho}}$ are the gradients of Eq. 3 w.r.t $\mu$ and $\rho$, respectively. We empirically find that $\Omega_{\mu} = \sigma$ and $\Omega_{\rho} = 1$, i.e. no change in the learning for $\rho$ works best.

2.1. BCL using weight pruning (BCL-P)

A variant of our method, BCL-P, is related to recent efforts in weight pruning in the context of reducing inference computation and network compression [9]. Forgetting is prevented in pruning by saving a task-specific binary mask of important vs. unimportant parameters. Here, we adapt pruning to the Bayesian neural networks and propose to use the statistically-grounded uncertainty defined in Bayesian neural networks as the pruning criterion. We use the signal-to-noise ratio (SNR) [2] for each parameter defined as $\text{SNR} = \frac{\sigma}{\sigma^2}$ as a hard threshold for importance.

3. Experimental Setup and Results

**Datasets and sequence of tasks:** We evaluate our approach in two common scenarios for continual learning: 1) class-incremental learning of a single or two randomly alternating datasets, where each task covers only a subset of the classes in a dataset, and 2) continual learning of multiple datasets, where each task is a dataset. We use MNIST split, permuted MNIST, and CIFAR10/100 for class incremental learning with similar experimental settings as used in [16, 10]. We also evaluate our approach on continually learning a sequence of datasets which have different distributions using the identical 8 task sequence as in [16], which includes FaceScrub [13], MNIST, C1FAR100, NotMNIST [3], SVHN, CIFAR10, TrafficSigns [17], and FashionMNIST [18].

**Baselines:** Within the Bayesian framework, we compare to three models which do not incorporate the importance of parameters, namely fine-tuning, feature extraction, and joint training. In fine-tuning (BBB-FIT), training continues upon arrival of new tasks without any forgetting avoidance strategy. Feature extraction, denoted as (BBB-FE) in our experiments, refers to freezing all layers in the network after training the first task and training only the last layer for the remaining tasks. In joint training (BBB-JT) we learn all the tasks jointly in a multitask learning fashion which serves as the upper bound for average accuracy on all tasks, as it does not adhere to the continual learning scenario. From prior work, we compare with state-of-the-art approaches including Elastic Weight Consolidation (EWC) [6], Incremental Moment Matching (IMM) [7], Learning Without Forgetting (LWF) [8], Less-Forgetting Learning (LFL) [5], PathNet [4], Progressive neural networks (PNNs) [15], and Hard Attention Mask (HAT) [16] using implementations provided by [16].

**Performance measurement:** Let $n$ be the total number of tasks. Once all are learned, we evaluate our model on all
n tasks. ACC is the average test classification accuracy across all tasks. To measure forgetting we report backward transfer, BWT, which indicates how much learning new tasks has influenced the performance on previous tasks. While BWT < 0 directly reports catastrophic forgetting, BWT > 0 indicates that learning new tasks has helped with the preceding tasks. Formally, BWT and ACC are defined as follows:

\[
\text{BWT} = \frac{1}{n} \sum_{i=1}^{n} R_{i,n} - R_{i,i}, \quad \text{ACC} = \frac{1}{n} \sum_{i=1}^{n} R_{i,n}
\]

where \( R_{i,n} \) is the test classification accuracy on task \( i \) after sequentially finishing learning the \( n \)th task. Note that in BCL-P, \( R_{i,i} \) refers the test accuracy on task \( i \) before pruning and \( R_{i,n} \) after pruning which is equivalent to the end of sequence performance.

Table 1 and 2 show ACC and BWT of BCL and BCL-P in comparison to state-of-the-art models. BCL performs better than all baselines reaching ACC of 97.7%, 97.4%, 70.4%, and 84.0% in Split MNIST, permuted MNIST, CIFAR10/100, and 8 subsequent tasks, respectively.

**Single Head and Generalized Accuracy.** BCL can be used even if the task information is not given at test time. For this purpose, at training time, instead of using a separate fully connected classification head for each task, we use a single head with total number of outputs for all tasks. Table 3 presents our results for BCL and BBB-FT trained with a single head against having a multi-head architecture, in columns 4-7. Interestingly, we found a small performance degrade for BCL from training with multi head to single head. We evaluated BCL and BBB-FT with a more challenging metric where the prediction space covers the classes across all the tasks. Hence, confusion of similar class labels across tasks can be measured. Performance for this condition is reported as Generalized ACC in Table 3 in columns 2-3. We also observed a small performance reduction in going from ACC to Generalized ACC, suggesting non-significant confusion caused by presence of higher number of classes at test time.

Table 3: Single Head vs. Multi Head architecture and Generalized vs. Standard Accuracy. Generalized accuracy means that Test time task independent task information is not available at test time. SM, PM, CF, and 8T denote the Split MNIST, Permuted MNIST, Alternating CIFAR10/100, and sequence of 8 tasks, respectively.

4. Conclusion

In this work, we proposed a continual learning formulation based on Bayesian neural networks, called BCL, that uses uncertainty predictions to perform continual learning: important parameters can be either fully preserved through a saved binary mask (BCL-P) or allowed to change condi-
tioned on their uncertainty for learning new tasks (BCL). We show that BCL performs superior or on par with state-of-the-art models such as HAT [16] across all the experiments. BCL can also be used in a single head setting where the right subset of classes belonging to the task is not known during inference.

References


