Continual Learning via Bit-Level Information Preserving

Yujun Shi[1]   Li Yuan[1]   Yunpeng Chen[2]   Jiashi Feng[1]

[1]National University of Singapore   [2] YITU Technology

{shi.yujun,yuanli}@u.nus.edu   yunpeng.chen@yitu-inc.com   elefjia@nus.edu.sg

Abstract

Continual learning tackles the setting of learning different tasks sequentially. Despite the lots of previous solutions, most of them still suffer significant forgetting or expensive memory cost. In this work, targeted at these problems, we first study the continual learning process through the lens of information theory and observe that forgetting of a model stems from the loss of information gain *on its parameters from the previous tasks when learning a new task. From this viewpoint, we then propose a novel continual learning approach called Bit-Level Information Preserving (BLIP) that preserves the information gain on model parameters through updating the parameters at the bit level, which can be conveniently implemented with parameter quantization. More specifically, BLIP first trains a neural network with weight quantization on the new incoming task and then estimates information gain on each parameter provided by the task data to determine the bits to be frozen to prevent forgetting. We conduct extensive experiments ranging from classification tasks to reinforcement learning tasks, and the results show that our method produces better or on par results comparing to previous state-of-the-arts. Indeed, BLIP achieves close to zero forgetting while only requiring constant memory overheads throughout continual learning[1].*

1. Introduction

Continual learning tackles the setting where an agent learns different tasks sequentially. It is challenging since the agent is usually not allowed to refer to the previously learned tasks when learning a new one. Current artificial neural networks generally fail as they tend to suffer severe performance degradation on previously learned tasks after learning new ones, which is known as *catastrophic forgetting* [23, 11]. A commonly acknowledged reason for this problem is that model parameters drift when fitting the new incoming task data.

Two types of methods have been developed to address this hazard. The first type of methods, which are based on model pruning technique [10, 29, 22, 21], first identify model parameters important for a task and then store a task-specific mask to mark these parameters and prohibit them from changing in subsequent learning to prevent forgetting. During the evaluation stage, only the parameters marked by the task-specific mask are applied on the given task. While these methods usually enjoy relatively low forgetting rate, they suffer from the drawback of linearly growing memory overheads resulted from task-specific masks as continual learning proceeds. The other type, known as "regularization-based methods" [14, 12, 36, 26, 16], impose model regularization terms when training on the new task to prevent model parameters from deviating away from previously learned ones and thus alleviate forgetting issue. They do not involve any task-specific masks but usually suffer more severe forgetting comparing to pruning-based ones. The hard trade-off between forgetting prevention and memory efficiency for these two types of methods is because they only consider preventing forgetting on the parameter-level.

In this paper, we dive into the finer granularity of bit-level instead of parameter-level to investigate and address the forgetting problem in continual learning. To start with, we study continual learning from information theory perspective, which is developed from the Bayesian inference framework that interprets supervised learning as an inference process over the model parameters from the training data. For a continual learning (inference) process, where data seen by model accumulate as the model experiences more tasks, the inference on each model parameter should become increasingly certain. To quantify this expected increase of certainty on model parameters given the streaming data, we exploit *information gain*, which corresponds to the reduction of entropy on parameters estimated after experiencing new task data. In this way, we consider continual learning as a recursive process of gaining information on model parameters, driven by the sequentially incoming task data. From this viewpoint, forgetting can be understood as loss of information provided by previous task data on model parameters.

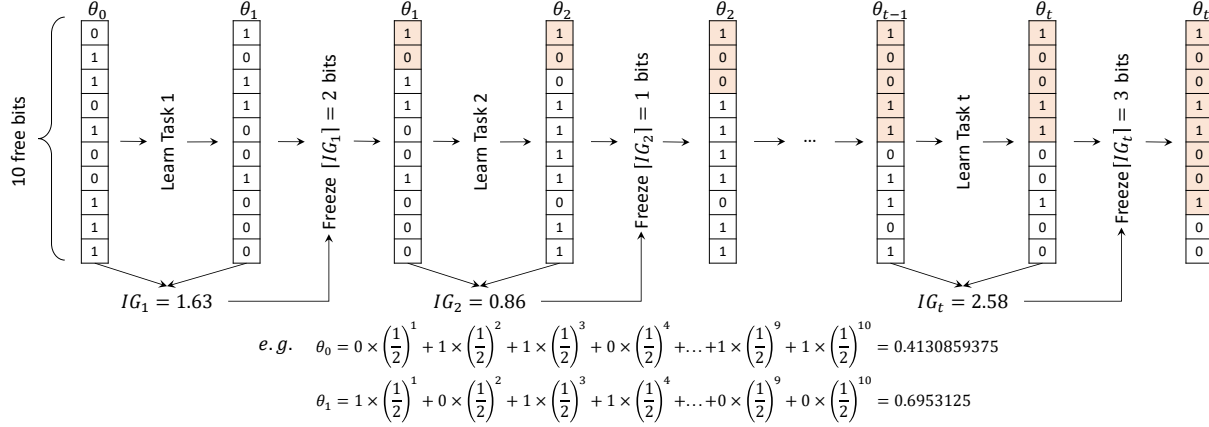[1]Code: https://github.com/Yujun-Shi/BLIP

Figure 1. **Workflow of BLIP. Best viewed in color.** We consider a simple scenario with one single parameter θ quantized to 10 bits to illustrate our method. θ_t denotes the parameter after learning on task 1 to t, and θ_0 is a randomly initialized value before training on any task. IG_t denotes information gain on θ after learning the task t. Bit representation of θ after learning each task is shown below. From the higher bit positions to lower ones is more significant bits to less significant ones. Frozen bits are filled with color and the rest bits are free bits. After learning each task, the information gain is calculated and then $\lceil IG_t \rceil$ bits are to be frozen in the bit representation. By repeating this process, the information on previous tasks can be preserved, enabling continual learning for neural networks.

To more intuitively study information gain in continual learning context, we quantize model parameters (*e.g.* to 20 bits) and view them in their bit representation, where values are represented by series of bits. Initially, before learning on any tasks, a model parameter is free to update, which corresponds to all bits of the parameter being free-to-flip. Shannon entropy, which is equivalent to the number of free-to-flip bits, is 20 at this point. Next, after training on the first incoming task, information gain brought by the first task data, which is the reduction of Shannon entropy after learning the first task, thus corresponds to how many bits are now becoming certain.[2] Leaving these bits to continue to flip in the subsequent learning process means discarding the information provided by this task, and forgetting thus happens. This motivates us to freeze these certain bits and prohibiting them from flipping in subsequent task iterations to preserve information gain provided by the first task. By applying the same information gain-guided bit freezing after learning each subsequent task, information provided by each task can be preserved and forgetting can thus be prevented in continual learning.

We accordingly develop a Bit-Level Information Preserving (BLIP) method to tackle the forgetting issue in continual learning. Given an incoming task, BLIP first trains a weight quantized neural network. Then, BLIP estimates the information gain on the parameters brought by this task's data, which suggests how many bits to freeze. The frozen bits are not allowed to flip in subsequent learning, preserving the information provided by this task. This process is applied recursively with each new task to enable continual

learning without forgetting. We provide a simple overview of its workflow in Fig. 1.

Unlike previous pruning-based methods, the memory overheads of our method is constant as BLIP only keeps track of how many bits to freeze in total for each parameter, without requiring any task-specific masks. On the other hand, our method is a more precise and stronger way of regularizing parameters compared to previous regularization-based methods, which effectively prevents forgetting. We validate our method across diverse settings of classification tasks and reinforcement learning tasks, and well prove that it performs on par with or better than previous state-of-the-arts.

This work makes following contributions:

1) We study continual learning through the lens of information theory and provide a new perspective on forgetting from information gain.

2) We propose a novel approach called Bit-Level Information Preserving (BLIP), which quantizes model parameters and directly preserves information gain on model parameters brought by previously learned tasks via bit freezing. Comparing to prior works, our method enjoys advantages of both low forgetting rate and reasonable memory overheads.

3) We conduct extensive experiments ranging from classification tasks to reinforcement learning tasks to demonstrate the effectiveness of our method.

4) To the best of our knowledge, our work is the first to explore weight quantization for continual learning.

---

[2]As will be explained in Sec. 4.4, bits becoming certain from more significant bits (with higher bit positions) to less significant ones (with lower bit positions).

2. Related Work

Pruning Based Continual Learning These methods rely on network pruning to preserve the previously learned knowledge [10, 29, 22, 21, 7]. In the training stage, these methods divide their model parameters into two sets: frozen set and free set. Only free parameters can change to adapt to learn the incoming task while frozen parameters stay unchanged in order to protect the previously learned knowledge. After learning on a task, pruning is applied to free parameters to identify parameters that are important for this task. A task-specific mask on model parameters is then saved to mark the parameters frozen after learning this task. During the evaluation stage, choosing which parameter to use is conditioned on the given task and task-specific masks.

Regularization Based Continual Learning These methods are mostly based on a Bayesian Inference framework [14, 36, 26, 16, 7, 1]. Under this framework, posterior distribution on model parameters after learning all previous tasks is viewed as prior when learning a new task. From this point of view, knowledge preserving can be achieved by regularizing parameter posterior of the new task to avoid deviating drastically from the prior (which is the posterior given previous tasks). This is normally achieved by adding penalty terms in the optimization objective.

Replay Based Continual Learning These methods save representative examples of previous tasks in a replay buffer of limited size [19, 5, 13, 27, 37, 9] or train a generative model to generate samples of previous tasks [30, 31, 25, 32]. During training of a new incoming task, data in replay buffer or produced by the trained generative model are then used to constrain the model to perform consistently on previous tasks.

Dynamic Architecture Based Continual Learning These methods [2, 35, 17] normally enlarge the model dynamically to adapt new incoming tasks. During inference, different model components are applied conditioned on the given or inferred task identity.

Parameter Quantization The parameter quantization technique has been widely applied to accelerating the deep neural networks [3, 38]. These works usually apply aggressive quantization on model parameters (*e.g.* 1/2/4 bits) for efficiency consideration, which causes performance degradation on models. However, in our work, we quantize parameters simply to view them in their bit representations and thus develop our method. Therefore, we only slightly quantized the parameter (*e.g.* to 20 bits).

3. Preliminaries

Problem Setup and Assumptions We consider learning a total of T tasks sequentially with a deep neural network model. Data of these T tasks are denoted by $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_T\}$ with $\mathcal{D}_t = \{\mathcal{X}_t, \mathcal{Y}_t\}$ for $t \in \{1, 2, ..., T\}$.

Here $\mathcal{X}_t$ denotes the set of raw data and $\mathcal{Y}_t$ denotes the corresponding label set. Throughout the continual learning process, when the model is learning task t, it is not allowed to refer to $\mathcal{D}_1, \ldots, \mathcal{D}_{t-1}$. In order to estimate the information gain on model parameters provided by each task's data, we adopt the Bayesian inference framework, which interprets model parameters as variables to be inferred given the training data.

We assume posteriors on all model parameters are mutually independent Gaussian distributed similar to previous works [14, 12]. Therefore, without loss of generality, we study a single scalar model parameter θ in the following to illustrate our method.

Notations and Definitions We denote the posterior on the model parameter θ after sequentially learning on task 1 to t as $p(\theta|\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_t)$ with the shorthand being $p(\theta_{0:t})$. The value of θ after learning on tasks 1 to t is denoted as $\theta_{0:t}^*$. Loss on $\mathcal{D}_t$ when learning on task t is denoted as $l(D_t, \theta)$. The model prediction output on a data sample x given the model parameter θ is denoted as $p_\theta(x)$, and $p_\theta(y|x)$ denotes the prediction probability of x's label being y. Since we rely on Fisher information of θ to estimate information gain and develop our method, we denote Fisher information estimated over $\mathcal{D}_t$ as F_t and the one over $\mathcal{D}_1 \ldots \mathcal{D}_t$ as $F_{0:t}$[3]. We denote the quantization function by Q. The result of quantizing θ to N bits is denoted by $Q(\theta, N)$. This means $Q(\theta, N)$ has a total of N bits in its bit representation. Correspondingly, the Shannon entropy on $Q(\theta, N)$ is defined as

$$H(Q(\theta, N)) = -\sum_{i=1}^{2^N} P(Q(\theta, N) = q_i) \log_2 P(Q(\theta, N) = q_i),$$
(1)

where q_i denotes $Q(\theta, N)$'s i-th possible value.

4. Bit-Level Information Preserving

4.1. Motivation and Method Overview

Continual learning can be regarded as a continual inference process over model parameters given streaming task data under Bayesian inference framework. As data seen by the model accumulates, the inference certainty on model parameters shall gradually increase. In order to quantify such increase of certainty, we introduce the concept of *information gain*, which amounts to the reduction of entropy.

To better understand the intuition of information gain in the context of continual learning, we study a model parameter θ quantized to N bits via Q, and view it in its bit representation, whose value is represented by a series of binary bits. Before learning any tasks, θ is free to update, which is

---

[3]F_0 corresponds to prior of Fisher information before learning any task, which is a hyper-parameter in our method.

equivalent to all N bits of $Q(\theta, N)$ being free-to-flip. Approximately speaking, each free-to-flip bit in $Q(\theta, N)$ flips with an equal probability, and the Shannon entropy can thus correspond to the total number of freely flipped bits, which is N in this case.

After learning an incoming task, the posterior on θ becomes more peaky and concentrated, which corresponds to some bits of $Q(\theta, N)$ becoming certain and being not able to flip. As will be elaborated, information provided by this task's data on $Q(\theta, N)$ immediately indicates how many bits of $Q(\theta, N)$ become certain from more significant bits to less significant bits. To preserve information provided by this task and thus prevent forgetting, a natural solution is to freeze these certain bits and not allow them to flip.

Motivated by this intuition of information gain, we develop our Bit-Level Information Preserving (BLIP) method, which iteratively conducts information gain-guided bit freezing to prevent forgetting in the continual learning. Specifically, without loss of generality, we consider learning the t-th task to illustrate our method. We denote n_t as the number of bits frozen after learning the t-th task and $n_{0:t} = \sum_{i=1}^{i=t} n_i$ is the total number of bits frozen after all previous t tasks. Before learning on task t, $\theta_{0:t-1}^*$ and $n_{0:t-1}$ are saved to do bit freezing, and $F_{0:t-1}$ is saved to help estimate information gain provided by task t data.

To learn task t, the proposed method quantizes the neural network weights and trains it on $\mathcal{D}_t$. Throughout the training process, the first $n_{0:t-1}$ bits of $Q(\theta, N)$ are frozen by BLIP to prevent forgetting on previous tasks and the rest $N - n_{0:t-1}$ bits can flip to adapt on $\mathcal{D}_t$. After the training is completed, BLIP estimates information gain provided by $\mathcal{D}_t$, which implies how many additional bits become certain. Finally, bits that become certain after learning $\mathcal{D}_t$ are frozen in the subsequent learning to prevent forgetting on task t.

We further elaborate on the details of above three steps in the following subsections.

4.2. Training on Task t

To start with, we describe our model parameter quantization scheme and explain how to optimize θ on $\mathcal{D}_t$ while keeping $Q(\theta, N)$'s first $n_{0:t-1}$ bits frozen.

Our quantization function Q is defined as follows:

$$Q(\theta, N) = \frac{\lfloor (2^N \min(\max(\theta, -1 + \frac{1}{2^{N+1}}), 1 - \frac{1}{2^{N+1}})) \rceil}{2^N},$$
(2)

where $\lfloor \cdot \rceil$ rounds the number to its nearest integer. For the quantized model, we rely on Straight Through Estimator (STE) [3], which approximates the gradient of the quantization function Q by $\frac{\partial Q(\theta, N)}{\partial \theta} = 1$, to optimize the model parameter θ. Therefore, θ is directly updated with the gra-

dient descent by

$$\theta := \theta - \alpha \frac{\partial l(\mathcal{D}_t, \theta)}{\partial Q(\theta, N)},$$
(3)

where α is the learning rate.

Next, to keep the first $n_{0:t-1}$ bits of $Q(\theta, N)$ unchanged, we clip the value of θ after the above gradient descent updates. Specifically, based on the definition of Q in Eqn. (2), keeping $Q(\theta, N)$'s first $n_{0:t-1}$ bits frozen is equivalent to restricting θ to be within an interval centered around $c_{t-1} = Q(\theta_{0:t-1}^*, n_{0:t-1})$ with the radius of $\frac{1}{2^{n_{0:t-1}}}$.

Therefore, the first $n_{0:t-1}$ can be kept frozen by the following clipping operation:

$$\theta := \min(\max(\theta, c_{t-1} - \frac{1}{2^{n_{0:t-1}}}), c_{t-1} + \frac{1}{2^{n_{0:t-1}}}).$$
(4)

The above gradient descent and clipping steps are applied repeatedly until the training loss l converges on $\mathcal{D}_t$.

4.3. Estimating Information Gain

After training on $\mathcal{D}_t$, we estimate the Information Gain (IG) on $Q(\theta, N)$ provided by $\mathcal{D}_t$, which is the reduction of the Shannon entropy on $Q(\theta, N)$ after training on $\mathcal{D}_t$:

$$IG(Q(\theta, N), \mathcal{D}_t) = H(Q(\theta_{0:t-1}, N)) - H(Q(\theta_{0:t}, N)).$$
(5)

According to this definition and the definition of the Shannon entropy Eqn. (1), estimating information gain requires the knowledge of $p(\theta_{0:t-1})$ and $p(\theta_{0:t})$, which are the posterior on θ before and after learning on $\mathcal{D}_t$ respectively.

Unfortunately, the posterior on θ is intractable and has to be approximated by Laplace's approximation [20]. This approximation states that given a dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, $p(\theta|\mathcal{D})$ can be approximated by $\mathcal{N}(\theta_\mathcal{D}^*, (mF_\mathcal{D}(\theta_\mathcal{D}^*))^{-\frac{1}{2}})$, where $\theta_\mathcal{D}^*$ is θ's value after learning on D, m is the number of samples used to estimate Fisher information, and $F_\mathcal{D}(\theta)$ is *Fisher information* estimated over $\mathcal{D}$:

$$F_\mathcal{D}(\theta) = \mathbb{E}_{x \sim \mathcal{X}, y \sim p_\theta(x)} \left[\left(\frac{\partial \ln p_\theta(y|x)}{\partial \theta} \right)^2 \right].$$
(6)

By assuming we use the same number of samples to estimate Fisher information for each task, $p(\theta_{0:t-1})$ and $p(\theta_{0:t})$ can be approximated by $\mathcal{N}(\theta_{0:t-1}^*, (mtF_{0:t-1})^{-\frac{1}{2}})$ and $\mathcal{N}(\theta_{0:t}^*, (m(t+1)F_{0:t})^{-\frac{1}{2}})$ respectively. $F_{0:t-1}$, as mentioned before, is previously saved after learning task 1 to t, and only $F_{0:t}$ is unknown. To obtain $F_{0:t}$, we first estimate F_t over $\mathcal{D}_t$ by Eqn. (6). Then, with $F_{0:t-1}$ and F_t, $F_{0:t}$ can be obtained by

$$F_{0:t} = \frac{tF_{0:t-1} + F_t}{t+1}.$$
(7)

With the posterior approximation introduced above and the definition of information gain Eqn. (5), the information gain provided by $\mathcal{D}_t$ can be estimated by

$$IG(Q(\theta, N), \mathcal{D}_t) \approx \frac{1}{2} \log_2 m(t+1)F_{0:t} - \frac{1}{2} \log_2 mtF_{0:t-1}$$
$$= \frac{1}{2} \log_2 \frac{tF_{0:t-1} + F_t}{tF_{0:t-1}}. \tag{8}$$

Refer to appendix for detailed elaboration on Eqn. (8).

4.4. Information Gain-Guided Bit Freezing

With the estimated information gain, we now investigate which bits shall be frozen to prevent forgetting on task t in the subsequent learning. Approximately speaking, each free bit of $Q(\theta, N)$ flip with equal probability, and the Shannon entropy on $Q(\theta, N)$ is thus the number of bits free-to-flip in total. Information gain, which is the reduction of the Shannon entropy after learning $\mathcal{D}_t$, is thus approximately equivalent to how many more bits of $Q(\theta, N)$ become certain throughout the process. Leaving these certain bits to continue to flip means discarding the information gain provided, which directly motivates us to freeze these certain bits. Therefore, the number of additional bits to freeze n_t can be obtained by

$$n_t = \lceil IG(Q(\theta, N), \mathcal{D}_t) \rceil. \tag{9}$$

In addition, since the posterior on θ is a Gaussian distribution and only becomes peaky and concentrated locally, bits in $Q(\theta, N)$ become certain starting from more significant bits (with higher bit positions) to less significant bits (with lower bit positions). In practice, we also clip n_t between 0 and $N - n_{0:t-1}$ so that the number of bits frozen in total does not surpass N. Therefore, by freezing the first n_t bits of the remaining $N - n_{0:t-1}$ bits of $Q(\theta, N)$, our method can specifically preserve the information gain provided by $\mathcal{D}_t$ and thus prevent forgetting on task t.

In the end, the number of bits frozen in total $n_{0:t}$, current Fisher information $F_{0:t}$, as well as current parameter value $\theta_{0:t}^*$ are then saved for the next task iteration. By recursively preserving information gain provided by each task, our method can mitigate forgetting in continual learning. A detailed summary for our algorithm is shown in Alg. 1.

5. Experiments

In this section, we report experiment results of our method. In Sec. 5.1, we introduce experimental setups. Next, we demonstrate the effectiveness of our method in continual image classification settings of various scales and continual reinforcement learning in Sec. 5.2 and Sec. 5.3 respectively. Finally, in Sec. 5.4, based on our RL agent, we visualize the bit freezing process throughout continual learning to further see how BLIP works.

Algorithm 1 Bit-Level Information Preserving (BLIP)

1: $\theta \leftarrow$ randomly init parameters
2: $N \leftarrow$ pre-defined quantization bits
3: $c \leftarrow 0$, $S \leftarrow 0$, $F \leftarrow$ hyper parameter F_0
4: $L \leftarrow$ loss function, $\alpha \leftarrow$ learning rate
5: $Q \leftarrow$ quantization function
6: **for** $t \leftarrow 1$ to T **do**
7:   obtain task t dataset $D_t = \{\mathcal{X}, \mathcal{Y}\}$
8:   $\theta \leftarrow \text{TRAIN}(\theta, c, S, D, L)$
9:   $IG, F_{post} \leftarrow \text{ESTIMATEINFOGAIN}(\theta, \mathcal{X}, F, t)$
10:   $n_t \leftarrow \min\left(\max(\lceil IG \rceil, 0), N - S\right)$
11:   $c \leftarrow Q(\theta, S)$, $S \leftarrow S + n_t$, $F \leftarrow F_{post}$
12: **end for**
13: **function** $\text{TRAIN}(\theta, c, S, D, L, Q, \alpha)$
14:   **while** loss L not converged on D **do**
15:    $\theta \leftarrow \theta - \alpha \nabla_\theta L(D, Q(\theta))$
16:    $\theta \leftarrow \min(\max(\theta, c - \frac{1}{2^S}), c + \frac{1}{2^S})$
17:   **end while**
18:   return θ
19: **end function**
20: **function** $\text{ESTIMATEINFOGAIN}(\theta, \mathcal{X}, F, t)$
21:   $F_t \leftarrow 0$
22:   **for** each data point x in $\mathcal{X}$ **do**
23:    sample $y \sim p_\theta(x)$
24:    $F_t \leftarrow F_t + \left(\frac{\partial \log p_\theta(y|x)}{\partial \theta}\right)^2$
25:   **end for**
26:   $F_t \leftarrow \frac{F_t}{|\mathcal{X}|}$
27:   $F_{post} \leftarrow tF + F_t$
28:   $IG \leftarrow \frac{1}{2} \log_2 \frac{F_{post}}{tF}$
29:   return IG, F_{post}
30: **end function**

5.1. Experimental Setups

Evaluation Metrics We evaluate all methods with two widely used metrics: average accuracy (ACC) and backward transfer (BWT) [19]. ACC is the average accuracy over all learned tasks at the end of the continual learning process, while BWT quantifies how learning on new tasks affects model performance on previously learned tasks. Formally, let the total number of learned tasks be T, and $A_{i,j}$ denote the accuracy on the j-th task after sequentially learning the first i tasks. ACC is defined as

$$ACC = \frac{1}{T} \sum_{i=1}^{i=T} A_{T,i}, \tag{10}$$

and BWT is defined as

$$BWT = \frac{1}{T-1} \sum_{i=1}^{i=T} (A_{T,i} - A_{i,i}). \tag{11}$$

Negative BWT implies forgetting on previous tasks while positive means learning on new tasks can even help with the performance on previous tasks. For each of the two metrics, higher values indicate better performance.

Benchmarks and Models We target the setting of continual learning with disjoint tasks, where task identity is given during both training and evaluation. The following continual image classification benchmarks are used to evaluate our method: MNIST-5 [7, 8], PMNIST [14, 29], Alternating Cifar10/100 [29], 20-Split mini-ImageNet [37, 6], Sequence of 5 tasks [8]. Reinforcement Learning agents are evaluated over a sequence of six Atari environments [24]. For MNIST-based settings, we adopt a two layer perceptron with 1,200 hidden units as [7]. For other classification settings, we use AlexNet [15] for all methods. For the reinforcement learning setting, we use a PPO agent [28] with three convolution layers and one fully connected layer.

Baselines For image classification tasks, we compare our method with naive baselines of direct sequential fine-tuning (FT), fine-tuning classifier with backbone fixed after the first task (FT-FIX), and previous methods including EWC [14, 12], VCL [26] IMM [16], LWF [18], A-GEM [5], HAT [29], UCB [7], and ACL [8]. Among these methods, HAT and ACL incur linearly growing memory overheads due to task specific parameters. A variant of ACL, which is ACL with replay buffer (ACL-R), using replay buffer to further boost forgetting prevention. In addition, UCB and VCL are based on Bayesian neural networks [4], which have much higher computation cost than normal networks during training as several times of Monte-Carlo sampling are needed over model parameters in one iteration of gradient descent. For IMM, we use its "mode" version [16]. For EWC, we use its online variant proposed in [12].

Hyperparameters For classification tasks, our code is based on released implementations of [29]. Plain Stochastic Gradient Descent with batch size 32 and initial learning rate 0.05 is used for optimization. Learning rate decays by a factor of 3 if validation loss plateaus for 5 consecutive epochs. Training stops when learning rate is below 1×10^{-4} or we have iterated over 200 epochs. For reinforcement learning, the PPO agent is trained with initial learning rate 2.5×10^{-4}, entropy regularization coefficient 0.01. We sample totally 10 millions steps in each environment to train our agents. Model parameters are quantized to 20 bits for BLIP. In addition, we ablate the choice for F_0 in appendix.

5.2. Continual Learning for Image Classification

5.2.1 MNIST-5 and P-MNIST

We start with a relatively simple setting of MNIST-5 where the ten MNIST classes are equally split into five separate tasks for the model to learn sequentially. In addition, we evaluate our method on another MNIST-based continual learning setting, Permuted MNIST. In this setting, the first

Table 1. Results on MNIST-5. MO is memory overheads complexity. Results denoted by (*) are provided by [7].

Methods	BWT(%)	ACC(%)	MO
VCL*	-0.56	98.20	$O(1)$
IMM*	-11.20	88.54	$O(1)$
EWC*	-4.20	95.78	$O(1)$
HAT*	0.00	99.59	$O(T)$
UCB*	0.00	99.63	$O(1)$
BLIP	0.01	99.64	$O(1)$

Table 2. Results on PMNIST. Results denoted by (*) are provided by [7].

Methods	BWT(%)	ACC(%)	MO
LWF*	-31.17	65.65	$O(1)$
IMM*	-7.14	90.51	$O(1)$
HAT*	0.03	97.34	$O(T)$
UCB*	0.03	97.42	$O(1)$
BLIP	-0.21	97.31	$O(1)$

Table 3. Results on Alternating Cifar10/100.

Methods	BWT(%)	ACC(%)	MO
FT	-14.23	67.94	0
FT-FIX	0.00	43.78	0
LWF	-40.88	44.27	$O(1)$
IMM	-16.47	66.43	$O(1)$
EWC	-5.5	72.11	$O(1)$
HAT	-0.02	80.22	$O(T)$
BLIP	-0.43	74.70	$O(1)$

task is the standard MNIST classification while the following nine tasks are created by permuting pixels of MNIST images with nine different schemes. Results on MNIST-5 and P-MNIST are shown in Tab. 1 and Tab. 2 respectively. From these results, it can be seen that our method achieves on par performance with state-of-the-art methods such as UCB, HAT, and is significantly better than other methods in terms of ACC and BWT. However, as mentioned, UCB and HAT achieve these results with either much higher computation cost or undesired growing memory overheads during training, while our method is free from these hazards.

5.2.2 Alternating Cifar10/100

Next, we evaluate our method on a more challenging setting of Alternating Cifar10/100. Cifar10 and Cifar100 datasets are both divided equally into five different tasks respectively, with two classes in each Cifar10 task and twenty classes in each Cifar100 task. The model is required to learn sequentially the combined ten tasks. Our results are presented in Tab. 3. In this setting, one can observe that the gap in terms of forgetting prevention between pruning-based method (HAT) and regularization-based methods (LWF, IMM, EWC) is very large. Our method BLIP, which suffers

no memory overheads during evaluation, greatly surpasses the compared regularization-based methods and makes a significant step towards continual learning being free from forgetting and growing memory overheads.

5.2.3 20-Split mini-ImageNet

To further show the effectiveness of our method in larger scale scenario, we conduct experiments in the setting of 20-Split mini-ImageNet, where the 100 classes of mini-ImageNet are equally split into 20 tasks. Results are shown in Tab. 4. Our splitting scheme is exactly the same as [8]. Model size of each method is kept similar for fair comparison. From the results, we can see that BLIP achieves the highest ACC. HAT has better BWT than BLIP by using task-specific parameters while ACL-R boosts BWT with replay buffer.

Table 4. Experiment Results on 20-Split mini-ImageNet. RB is size of replay buffer. Results are averaged over 5 random seeds; mean $\pm$ std is reported. Results denoted by ($\dagger$) are provided by [8].

Methods	BWT (%)	ACC (%)	RB (MB)
LWF	-45.93 $\pm$ 1.05	29.30 $\pm$ 0.64	-
A-GEM†	-15.23 $\pm$ 1.45	52.43 $\pm$ 3.10	110.1
HAT†	-0.04 $\pm$ 0.03	59.45 $\pm$ 0.05	-
ACL†	-3.71 $\pm$ 1.31	57.66 $\pm$ 1.44	-
ACL-R†	0.00 $\pm$ 0.00	62.07 $\pm$ 0.51	8.5
BLIP	-1.05 $\pm$ 0.42	65.69 $\pm$ 0.87	-

Table 5. Experiment Results on Sequence of 5 tasks. MS is Model Size. Results are averaged over 5 random seeds; mean $\pm$ std is reported. Results denoted by ($\dagger$) are provided by [8].

Methods	BWT (%)	ACC (%)	MS (MB)
FT	-34.16 $\pm$ 9.00	65.31 $\pm$ 7.18	16.97
FT-FIX	0.00 $\pm$ 0.00	71.57 $\pm$ 4.55	16.97
LWF	-61.14 $\pm$ 5.92	42.93 $\pm$ 4.59	16.97
IMM	-21.56 $\pm$ 5.46	73.39 $\pm$ 4.20	16.97
UCB†	-1.34 $\pm$ 0.04	76.34 $\pm$ 0.12	16.4
ACL†	-0.01 $\pm$ 0.15	78.55 $\pm$ 0.29	16.5
BLIP	-0.13 $\pm$ 0.324	82.87 $\pm$ 1.43	16.97

5.2.4 Sequence of 5 tasks

To further demonstrate our method's ability to learn sequentially on more diverse domains, we evaluate it by learning on Cifar10, notMNIST, MNIST, SVHN, FashionMNIST. Our results are averaged over 5 random seeds as [8]. From the results, one can observe that our method considerably surpasses previous state-of-the-arts such as UCB and ACL in terms of ACC. Because ACL relies on given task information and saved task-specific modules, it can achieve better BWT than BLIP. When doing a fair comparison with other methods, BLIP can achieve much better BWT.

5.3. Continual Reinforcement Learning

Our continual reinforcement learning benchmark requires the agent to sequentially learn to play the following six Atari games: kung fu master, boxing, james bond, krull, river raid, space invaders.

We first demonstrate our results on each task separately in Fig. 3, where we plot rewards of trained agents on each task from the point they start to train on the task to the ending of the whole continual learning process. From this figure, we can see that the agent trained by FT suffers from catastrophic forgetting as its reward on one task drastically decreases after shifting to learning on the following tasks. For the agent trained with EWC, although forgetting is alleviated and performance can be retained on the last four tasks, its performance on the first two tasks can not sustain through the continual learning process and decreases dramatically. Different from these two, our method can remember to play all tasks in this continual learning setting.

Next, to compare the performance of each method in an overall manner, we plot the accumulated reward, which is the sum of rewards on all learned tasks at each environment step, in Fig. 4. We normalize rewards on each task to the scale of 0 to 1. Therefore, if an agent can both learn and remember all six tasks well, its accumulated normalized reward shall steadily increase from 0 to 6 in the continual learning process. On the contrary, the accumulated normalized reward should oscillate around 1 after training the first task if an agent constantly forgets to perform on previously learned tasks, which is the case for FT in the figure. As illustrated, our method BLIP outperforms EWC in terms of the accumulated normalized reward in the whole continual learning process.

5.4. Visualizing Bit Freezing and Related Discussion

To further understand the underlying working mechanism of BLIP through the whole continual learning process, we visualize the corresponding bit freezing process in Fig. 2. In this figure, we show how bits are gradually frozen for parameters of our RL agent after learning each task. As can be observed, more and more bits of model parameters are frozen to anchor the learned knowledge as continual learning proceeds. In addition, we can see less bits are frozen after learning the last two tasks comparing to the first four tasks, which means not many bits of the model parameters have been effectively used to adapt to the last two tasks. This corresponds to the phenomenon that BLIP does not adapt well on the last two tasks in Fig. 3. We leave solving this problem for future work.

Previous neurobiology evidences suggest that human brain may mitigate forgetting by synaptic consolidation[34, 33], which is the process of strengthening synapses that are crucial for certain knowledge or skills. In the context of artificial neural networks, where synapses (connections
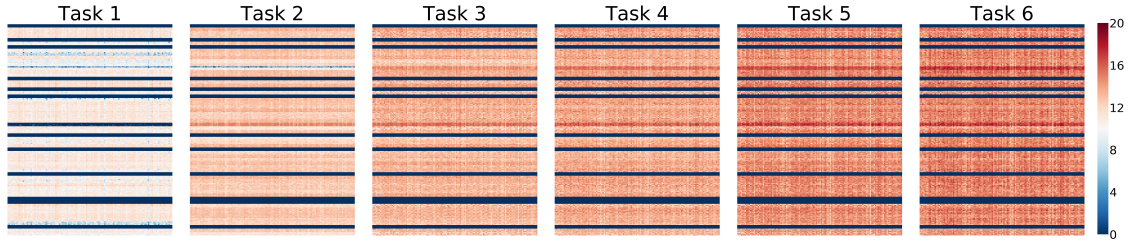
Figure 2. **Bit Freezing Visualization** We visualize bit freezing process of our RL agent's third convolution layer parameters during continual learning. Each pixel in a heat map represents the number of frozen bits of the corresponding entry (parameter) in weight matrix of the convolution layer. Each parameter has a total of 20 bits. From darker blue to darker red denotes more bits being frozen. Visualization of other layers are shown in appendix.
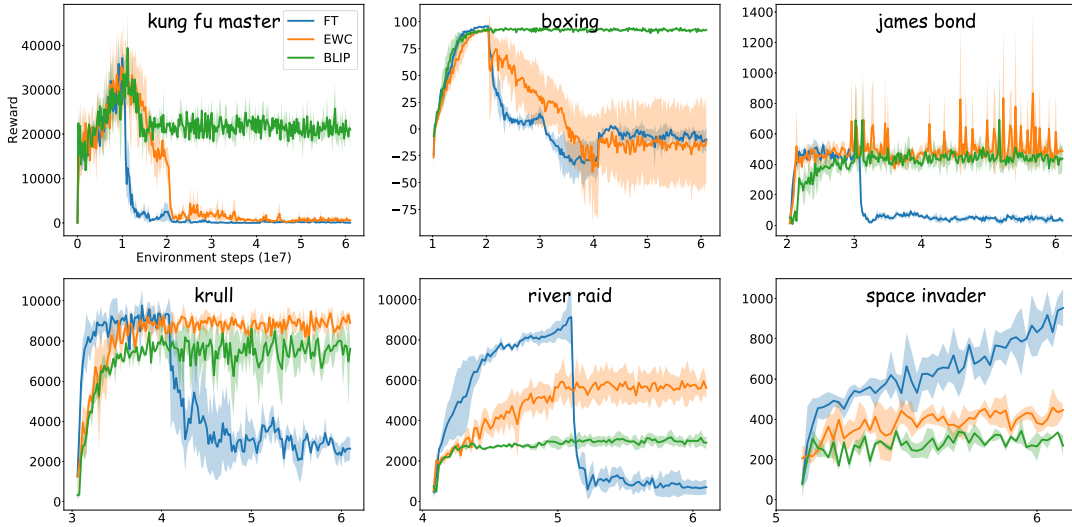

Figure 3. **Rewards on each task**. Each task is trained for 10 million environment steps, and 60 million environment steps are thus used in total. Tasks are learned left to right and top to bottom. Results are averaged over 3 random seeds.
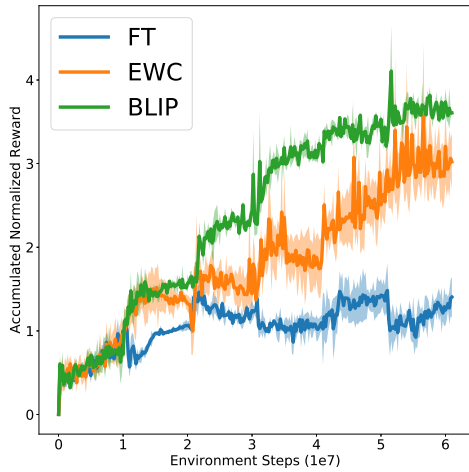

Figure 4. **Accumulated Normalized Rewards**.

between neurons) are implemented as weight parameters, BLIP mimics the process of synaptic consolidation via bit freezing. Concretely, as continual learning proceeds, more

bits are gradually frozen as shown in Fig. 2, which corresponds to the process of synapses being strengthened as more skills and knowledge are learned.

6. Conclusion

In this work, we study continual learning through lens of information theory and provide a new perspective on forgetting from information gain. Based on this perspective, we propose a novel approach called Bit-Level Information Preserving, which dives into bit-level and directly preserves information gain on model parameters provided by previously learned tasks via bit freezing. We conduct extensive experiments to show that our method can successfully adapt to continual classification settings of various scales as well as continual reinforcement learning settings.

Acknowledgment

References

[1] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. *arXiv preprint arXiv:1905.11614*, 2019. 3

[2] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375, 2017. 3

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3, 4

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015. 6

[5] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018. 3, 6

[6] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019. 6

[7] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. In *International Conference on Learning Representations*, 2019. 3, 6

[8] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. In *The European Conference on Computer Vision (ECCV)*, 2020. 6, 7

[9] Sayna Ebrahimi, Suzanne Petryk, Akash Gokul, William Gan, Joseph E Gonzalez, Marcus Rohrbach, and Trevor Darrell. Remembering for the right reasons: Explanations reduce catastrophic forgetting. *arXiv preprint arXiv:2010.01528*, 2020. 3

[10] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017. 1, 3

[11] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 1

[12] Ferenc Huszár. On quadratic penalties in elastic weight consolidation. *arXiv preprint arXiv:1712.03847*, 2017. 1, 3, 6

[13] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. *arXiv preprint arXiv:1802.10269*, 2018. 3

[14] 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*, 114(13):3521–3526, 2017. 1, 3, 6

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 6

[16] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in neural information processing systems*, pages 4652–4662, 2017. 1, 3, 6

[17] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pages 3925–3934. PMLR, 2019. 3

[18] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 6

[19] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, pages 6467–6476, 2017. 3, 5

[20] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003. 4

[21] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018. 1, 3

[22] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018. 1, 3

[23] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 1

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 6

[25] Martin Mundt, Sagnik Majumder, Iuliia Pliushch, and Visvanathan Ramesh. Unified probabilistic deep continual learning through generative replay and open set recognition. *arXiv preprint arXiv:1905.12019*, 2019. 3

[26] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017. 1, 3, 6

[27] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pages 350–360, 2019. 3

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 6

[29] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423*, 2018. 1, 3, 6

[30] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017. 3

[31] Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018. 3

[32] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6619–6628, 2019. 3

[33] Guang Yang, Cora Sau Wan Lai, Joseph Cichon, Lei Ma, Wei Li, and Wen-Biao Gan. Sleep promotes branch-specific formation of dendritic spines after learning. *Science*, 344(6188):1173–1178, 2014. 7

[34] Guang Yang, Feng Pan, and Wen-Biao Gan. Stably maintained dendritic spines are associated with lifelong memories. *Nature*, 462(7275):920–924, 2009. 7

[35] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017. 3

[36] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *Proceedings of machine learning research*, 70:3987, 2017. 1, 3

[37] Mengmi Zhang, Tao Wang, Joo Hwee Lim, Gabriel Kreiman, and Jiashi Feng. Variational prototype replays for continual learning. *arXiv preprint arXiv:1905.09447*, 2019. 3, 6

[38] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 3