ENERGY-BASED MODELS FOR CONTINUAL LEARNING

Shuang Li
MIT CSAIL
lishuang@mit.edu

Yilun Du
MIT CSAIL
yilundu@mit.edu

Gido M.van de Ven
Baylor College of Medicine
ven@bcm.edu

Igor Mordatch
Google Brain
imordatch@google.com

ABSTRACT

We motivate Energy-Based Models (EBMs) as a promising model class for continual learning problems. Instead of tackling continual learning via the use of external memory, growing models, or regularization, EBMs have a natural way to support a dynamically-growing number of tasks or classes that causes less interference with previously learned information. Our proposed version of EBMs for continual learning is simple, efficient and outperforms baseline methods by a large margin on several benchmarks. Moreover, our proposed contrastive divergence based training objective can be applied to other continual learning methods, resulting in substantial boosts in their performance. We also show that EBMs are adaptable to a more general continual learning setting where the data distribution changes without the notion of explicitly delineated tasks. These observations point towards EBMs as a class of models naturally inclined towards the continual learning regime.

1 INTRODUCTION

Humans are able to rapidly learn new skills and continuously integrate them with prior knowledge. The field of Continual Learning (CL) seeks to build artificial agents with the same capabilities (Parisi et al., 2019). In recent years, continual learning has seen increased attention, particularly in the context of classification problems. Continual learning requires models to remember prior skills as well as incrementally learn new skills, without necessarily having a notion of an explicit task identity. Standard neural networks (He et al., 2016; Simonyan & Zisserman, 2014) experience the catastrophic forgetting problem and perform poorly in this setting. Different approaches have been proposed to mitigate catastrophic forgetting, but many rely on the usage of external memory (Li & Hoiem, 2017), additional models (Shin et al., 2017), or auxiliary objectives and regularization (Maltoni & Lomonaco, 2019), which can constrain the wide applicability of these methods.

In this work, we propose a new approach towards continual learning on classification tasks. Most existing CL approaches tackle these tasks by utilizing normalized probability distribution (i.e., softmax output layer) and trained with a cross-entropy objective. In this paper, we argue that by viewing classification from the lens of training an un-normalized probability distribution, we can significantly improve continual learning performance in classification problems. In particular, we interpret classification as learning an Energy-Based Model (EBM) across classes. Training becomes a wake-sleep process, where the energy of an input data at its ground truth label is decreased while the energy of the input at (an)other selected class(es) is increased. An important advantage is that this framework offers freedom to choose what classes to update in the continual learning process. By contrast, the cross entropy objective reduces the likelihood of *all* negative classes when given a new input, creating updates that lead to catastrophic forgetting.

The energy function, which maps an input-label pair to a scalar energy, also provides a way for the model to select and filter portions of the input that are relevant towards the classification on hand. We show that this enables EBMs training updates for new data to interfere less with previous data. In particular, our formulation of the energy function allows us to compute the energy of an input by learning a conditional gain based on the class label, which serves as an attention filter to select the most relevant information. In the event of a new class, a new conditional gain can be learned.

These unique properties benefit EBMs in addressing two important open challenges in continual learning. 1) First, we show that EBMs are promising for class-incremental learning, which is one of the most challenging settings for continual learning (van de Ven & Tolias, 2019). Generally, successful existing methods for class-incremental learning either store data or use generative replay, which has disadvantages in terms of memory and/or computational efficiency. We show that EBMs perform well in class-incremental learning without using replay and without relying on stored data. 2) The second open challenge that EBMs can address is continual learning without task boundaries. Typically, a continual learning problem is set up as a sequence of distinct tasks with clear boundaries that are known to the model (the *boundary-aware* setting). Most existing continual learning methods rely on these known boundaries for performing certain consolidation steps (e.g., calculating parameter importance, updating a stored copy of the model). However, assuming such clear boundaries is not always realistic, and often a more natural scenario is the *boundary-agnostic* setting (Zeno et al., 2018; Rajasegaran et al., 2020), in which data distributions gradually change without a clear notion of task boundaries. While many common CL methods cannot be used without clear task boundaries, we show that EBMs can be naturally applied to this more challenging setting.

Our work has three main contributions: First, we introduce energy-based models for classification CL problems. We show that EBMs can naturally deal with challenging problems in CL, including the boundary-agnostic setting and class-incremental learning without using replay. Secondly, we propose an energy-based training objective that is *simple* and broadly applicable to different types of models, with significant boosts on their performance. This contrastive divergence based training objective can naturally handle the dynamically growing number of classes and significantly reduces catastrophic forgetting. Lastly, we show that the proposed EBMs perform strongly on four standard CL benchmarks. These observations point towards EBMs as a class of models naturally inclined towards the CL regime and as an important new baseline upon which to build further developments. The code are made public to facilitate further research [1].

2 Continual Learning with softmax-based classifiers

The most common way to do classification with deep neural networks is to use a softmax output layer in combination with a cross-entropy loss. In CL, virtually all existing methods for classification are based on the softmax-based classifier (SBC) (Li & Hoiem, 2017; Zenke et al., 2017).

Given an input $\mathbf{x} \in \mathbb{R}^D$ and a discrete set $\mathcal{Y} = \{1, \ldots, N\}$ of N possible class labels, a traditional softmax-based classifier defines the conditional probabilities of those labels as $p_{\boldsymbol{\theta}}(y|\mathbf{x}) = \exp([f_{\boldsymbol{\theta}}(\mathbf{x})]_y) / \sum_{i \in \mathcal{Y}} \exp([f_{\boldsymbol{\theta}}(\mathbf{x})]_i), \quad$ for all $y \in \mathcal{Y}$, where $f_{\boldsymbol{\theta}}(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^N$ is a feed-forward neural network, parameterized by $\boldsymbol{\theta}$, that maps an input $\mathbf{x}$ to a N-dimensional vector of logits. $[\cdot]_i$ indicates the i^{th} element of a vector. A schema of SBC is shown in Figure 1 left.

Training. A softmax-based classifier is typically trained by optimizing the cross-entropy loss function. For a given input $\mathbf{x}$ and corresponding ground truth label y^+, the cross-entropy loss is $\mathcal{L}_{CE}(\boldsymbol{\theta}; \mathbf{x}, y^+) = -\log(p_{\boldsymbol{\theta}}(y^+|\mathbf{x}))$.

Inference. Given an input $\mathbf{x}$, the class label predicted by the softmax-based classifier is the class with the largest conditional probability $\hat{y} = \arg\max_{y \in \mathcal{Y}} p_{\boldsymbol{\theta}}(y|\mathbf{x})$.

2.1 Softmax-based classifiers for continual learning

When used for continual learning, and in particular when used for class-incremental learning, softmax-based classifiers face several challenges. One important issue is that softmax-based classifiers compute the cross-entropy loss over all classes (or sometimes over all classes that have been seen so far). As a result, when training on a new task, the likelihood of the currently observed classes is increased, but the likelihood of old classes is too heavily suppressed since they are not encountered in the new task. The softmax operation introduces competitive, winner-take-all dynamics that make the classifier catastrophically forget past tasks. We show such phenomenon of SBC in Section 4.1.3.

[1] Code and documentation are available at `https://energy-based-model.github.io/Energy-Based-Models-for-Continual-Learning`

3 Continual Learning with Energy-Based Models

In this section, we propose a simple but efficient energy-based training objective that can successfully mitigate the catastrophically forgetting in continual learning.

3.1 Energy-based models

EBMs (LeCun et al., 2006) are a class of maximum likelihood models that define the likelihood of a data point $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$ using the Boltzmann distribution:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{\exp(-E_{\boldsymbol{\theta}}(\mathbf{x}))}{Z(\theta)}, \quad Z(\boldsymbol{\theta}) = \int_{\mathbf{x} \in \mathcal{X}} \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) \tag{1}$$

where $E_{\boldsymbol{\theta}}(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$, known as the energy function, maps each data point $\mathbf{x}$ to a scalar energy value, and $Z(\boldsymbol{\theta})$ is the partition function. In deep learning applications, the energy function $E_{\boldsymbol{\theta}}$ is a neural network parameterized by $\boldsymbol{\theta}$.

EBMs are powerful models that have been applied to different domains, such as structured prediction Belanger & McCallum (2016); Gygli et al. (2017); Rooshenas et al. (2019); Tu & Gimpel (2019), machine translation Tu et al. (2020), text generation Deng et al. (2020), reinforcement learning Haarnoja et al. (2017), image generation Salakhutdinov & Hinton (2009); Du et al. (2020a;b; 2019); Xie et al. (2016; 2018); Nijkamp et al. (2019), memory modeling Bartunov et al. (2019), classification Grathwohl et al. (2019), and biologically-plausible training Scellier & Bengio (2017). As far as we are aware, EBMs for CL has so far remained unexplored.

3.2 Energy-based models for classification

To solve the classification tasks, we adapt the above general formulation of an EBM as follows. Given inputs $\mathbf{x} \in \mathbb{R}^D$ and a discrete set $\mathcal{Y}$ of possible class labels, we propose to use the Boltzmann distribution to define the conditional likelihood of label y given $\mathbf{x}$:

$$p_{\boldsymbol{\theta}}(y|\mathbf{x}) = \frac{\exp(-E_{\boldsymbol{\theta}}(\mathbf{x}, y))}{Z(\boldsymbol{\theta}; \mathbf{x})}, Z(\boldsymbol{\theta}; \mathbf{x}) = \sum_{y' \in \mathcal{Y}} \exp(-E_{\boldsymbol{\theta}}(\mathbf{x}, y')) \tag{2}$$

where $E_{\boldsymbol{\theta}}(\mathbf{x}, y) : (\mathbb{R}^D, \mathbb{N}) \to \mathbb{R}$ is the energy function that maps an input-label pair $(\mathbf{x}, y)$ to a scalar energy value, and $Z(\boldsymbol{\theta}; \mathbf{x})$ is the partition function for normalization.

Training. We want the distribution defined by $E_{\boldsymbol{\theta}}$ to model the data distribution p_D, which we do by minimizing the negative log likelihood of the data

$$\mathcal{L}_{\text{ML}}(\boldsymbol{\theta}) = \mathbb{E}_{(x,y) \sim p_D}[-\log p_{\boldsymbol{\theta}}(y|\mathbf{x})] \tag{3}$$

with the expanded form:

$$\mathcal{L}_{\text{ML}}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},y) \sim p_D}\left[E_{\boldsymbol{\theta}}(\mathbf{x}, y) + \log(\sum_{y' \in \mathcal{Y}} e^{-E_{\boldsymbol{\theta}}(\mathbf{x}, y')}) \right]. \tag{4}$$

Equation 4 minimizes the energy of $\mathbf{x}$ at the ground truth label y and minimizes the overall partition function by increasing the energy of $\mathbf{x}$ at other labels y'.

Inference. Given an input $\mathbf{x}$, the class label predicted by our EBMs is the class with the smallest energy at $\mathbf{x}$: $\hat{y} = \arg\min_{y' \in \mathcal{Y}} E_\theta(\mathbf{x}, y')$.

3.3 Energy-based models for continual learning

3.3.1 EBM Training Objective

We notice that in Equation 4, the energy over all class labels y' given data $\mathbf{x}$ are maximized. Directly maximizing energy across all labels raises the same problem as the softmax-based classifier models that the old classes are suppressed when training a model on new classes and thus cause the catastrophic forgetting. Inspired by (Hinton, 2002; Du & Mordatch, 2019; Xie et al., 2016), we find that the contrastive divergence approximation of Equation 4 can mitigate this problem and lead to a simpler equation. To do so, we define the following contrastive divergence loss:

$$\mathcal{L}_{\text{CD}}(\boldsymbol{\theta}; \mathbf{x}, y) = \mathbb{E}_{(\mathbf{x},y) \sim p_D}\left[E_{\boldsymbol{\theta}}(\mathbf{x}, y) - E_{\boldsymbol{\theta}}(\mathbf{x}, y^-) \right], \tag{5}$$

where y is the ground truth label of data $\mathbf{x}$ and y^- is a negative class label randomly sampled from the set of class labels in the current training batch $\mathcal{Y}_B$.

Different from the SBC, EBMs maximize likelihood not by normalizing over all classes but instead by contrastively increasing the energy difference between the ground truth label and another negative label for a given data point. This operation causes less interference with previous classes and enables EBMs to suffer less from catastrophic forgetting.

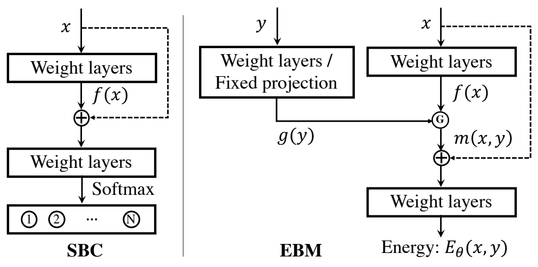

Figure 1: Schematic of the model architectures of the softmax-based classifier (SBC) and energy-based models (EBM). SBC takes an image $\mathbf{x}$ as input and outputs a fixed pre-defined N-dimensional vector to represent the probabilities of N different classes. EBM takes a data $\mathbf{x}$ and a class y as input and outputs their energy value. The dash lines are optional skip connections.

Importantly, such a sampling strategy also allows our EBMs to be naturally applied, without any modification, to different CL settings described in Section E.1. Since we select the negative sample(s) from the current batch, our EBMs do not require knowledge of the task on hand. This allows application of EBMs in the *boundary-agnostic* setting, in which the task boundaries are not given.

We find that the proposed EBM training objective is efficient enough to achieve good performance on different CL datasets (Table 1 and Table 4). We note however that it is possible to use other strategies for choosing the negative classes in the partition function in Equation 4. In Table 3, we explore alternative strategies: 1) using all classes in the current training batch $\mathcal{Y}_\mathcal{B}$ as negative classes, and 2) using all classes seen so far as negative classes. The usage of negative samples in the EBM training objective provides freedom for models to choose which classes to train on which is important for preventing catastrophic forgetting in continual learning.

EBMs are not limited to modeling the conditional distribution between $\mathbf{x}$ and y as shown in Equation 2. Another way to use the Boltzmann distribution is to define the joint likelihood of $\mathbf{x}$ and y. In Supplement Section A, we show that a different training objective can further improve the results.

3.3.2 Energy network

Another important difference from softmax-based classifiers is that the choice of model architectures becomes more flexible in EBMs. Traditional classification models only feed in $\mathbf{x}$ as input. In contrast, EBMs have many different ways to combine $\mathbf{x}$ and y in the energy function with the only requirement that $E_\theta(\mathbf{x}, y) : (\mathbb{R}^D, \mathbb{N}) \to \mathbb{R}$. In EBMs, we can treat y as an attention filter or gate to select the most relevant information between $\mathbf{x}$ and y.

To compute the energy of any data $\mathbf{x}$ and class y pair, we use y to influence a conditional gain on $\mathbf{x}$, which serves as an attention filter (Xu et al., 2015) to select the most relevant information between $\mathbf{x}$ and y. In Figure 1 (right), we first send $\mathbf{x}$ into a small network to generate the feature $f(\mathbf{x})$. The label y is mapped into a same dimension feature $g(y)$ using a small learned network or a random projection. We use the gating block G to select the most relevant information between $\mathbf{x}$ and y: $m(\mathbf{x}, y) = G(f(\mathbf{x}), g(y))$. The output is finally sent to weight layers to generate the energy value $E_\theta(\mathbf{x}, y)$. See Supplement Section D for more details about our model architectures.

Our EBMs allow any number of classes in new batches by simply training or defining a new conditional gain $g(\mathbf{y})$ for the new classes and generating its energy value with data point $\mathbf{x}$.

3.3.3 Inference

Table 1: Evaluation of *Class-IL* on the *boundary-aware* setting on four datasets. Note our comparison is restricted to methods that do not replay stored or generated data.

Method	**splitMNIST**	**permMNIST**	**CIFAR-10**	**CIFAR-100**
SBC	19.90 ± 0.02	17.26 ± 0.19	19.06 ± 0.05	8.18 ± 0.10
EWC	20.01 ± 0.06	25.04 ± 0.50	18.99 ± 0.03	8.20 ± 0.09
Online EWC	19.96 ± 0.07	33.88 ± 0.49	19.07 ± 0.13	8.38 ± 0.15
SI	19.99 ± 0.06	29.31 ± 0.62	19.14 ± 0.12	9.24 ± 0.22
LwF	23.85 ± 0.44	22.64 ± 0.23	19.20 ± 0.30	10.71 ± 0.11
MAS	19.50 ± 0.30	-	20.25 ± 1.54	8.44 ± 0.27
BGD	19.64 ± 0.03	84.78 ± 1.30	-	-
EBM	**53.12 ± 0.04**	**87.58 ± 0.50**	**38.84 ± 1.08**	**30.28 ± 0.28**

During inference, because we evaluate according to the class-incremental learning scenario (van de Ven & Tolias, 2019; Tao et al., 2020), the model must predict a class label by choosing from all classes seen so far. Let $\mathbf{x}_k$ be one data point from a batch $\mathcal{B}_k$ with an associated discrete label $y \in \mathcal{Y}_k$, where $\mathcal{Y}_k$ contains the class labels in $\mathcal{B}_k$. Then there are $\mathcal{Y} = \bigcup_{k=1}^{K} \mathcal{Y}_k$ different class labels in total after seeing all the batches. The MAP estimate is $\hat{y} = \arg\min_y E_{\theta_K}(\mathbf{x}_k, y)$, where $y \in \bigcup \mathcal{Y}_k$, and $E_{\theta_K}(\mathbf{x}_k, y)$ is the energy function with parameters θ_K resulting from training on the batches $\{\mathcal{B}_1, \cdots, \mathcal{B}_K\}$. The energy function can compute an energy for any discrete class input, including unseen classes, which avoids needing to predefine the number of classes in advance.

4 Experiments

In this section, we want to answer the following questions: How do the proposed EBMs perform on different CL settings? Can we apply the EBM training objective to other methods? And can we qualitatively understand the differences between EBMs and baselines? We report experiments on the *boundary-aware* setting in Section 4.1 and the *boundary-agnostic* setting in Section 4.2.

4.1 Experiments on Boundary-Aware Setting

4.1.1 Datasets and evalution protocols

We evaluate the proposed EBMs on the split MNIST (Zenke et al., 2017), permuted MNIST (Kirkpatrick et al., 2017), CIFAR-10 (Krizhevsky et al., 2009), and CIFAR-100 (Krizhevsky et al., 2009) datasets. The split MNIST dataset is obtained by splitting the original MNIST dataset (LeCun et al., 1998) into 5 tasks with each task having 2 classes. This dataset has 60,000 training images and 10,000 test images. The permuted MNIST protocol has 10 tasks, each task with 10 classes. For each task, the original images pixels are randomly permuted to generate 32×32 images. We separate CIFAR-10 into 5 tasks, each task with 2 classes. Similarly, CIFAR-100 is split into 10 tasks with each task having 10 classes. These last two datasets each have 50,000 training images and 10,000 test images.

As noted by (van de Ven & Tolias, 2019), the above task protocols could be evaluated according either the task-incremental, domain-incremental, or class-incremental learning scenario. Most CL approaches perform well on the first two simpler scenarios, but fail when asked to do class-incremental learning, which is considered as the most natural and also the hardest setting for CL (Tao et al., 2020; He et al., 2018). In this paper, we perform all experiments according to the class-incremental learning scenario.

4.1.2 Comparisons with existing methods

The most successful existing methods for *Class-IL* either rely on an external quota of memory (Rebuffi et al., 2017; Hayes et al., 2020) or on using generative replay (Shin et al., 2017; van de Ven et al., 2020). One of their disadvantages is that they are relatively expensive in terms of memory and/or computation. In this paper, we focus on CL without using replay and without storing data.

Table 2: Continual learning approaches using our training objective and their original one.

	split MNIST		CIFAR-10	
	Original	Ours	Original	Ours
SBC	19.90 ± 0.02	44.98 ± 0.05	19.06 ± 0.05	19.22 ± 1.12
EWC	20.01 ± 0.06	50.68 ± 0.04	18.99 ± 0.03	36.51 ± 1.20
Online EWC	19.96 ± 0.07	50.99 ± 0.03	19.07 ± 0.13	36.16 ± 1.02
SI	19.99 ± 0.06	49.44 ± 0.03	19.14 ± 0.12	35.12 ± 1.70
EBM	-	$\mathbf{53.12 \pm 0.04}$	-	$\mathbf{38.84 \pm 1.08}$

Comparisons with baselines. We compare our proposed EBM method with available baseline models that do not use replay, including the standard softmax-based classifier (SBC), EWC (Kirkpatrick et al., 2017), Online EWC (Schwarz et al., 2018), SI (Zenke et al., 2017), LwF (Li & Hoiem, 2017), MAS (Aljundi et al., 2019), and BGD (Zeno et al., 2018). The *Class-IL* results on four datasets are shown in Table 1.

Table 3: Performance of EBM on CIFAR-100 with different strategies for selecting the negative samples.

Dataset	CIFAR-100
All Neg Seen	8.07 ± 0.10
All Neg Batch	29.03 ± 0.53
1 Neg Batch	$\mathbf{30.28 \pm 0.28}$

All the baselines and EBMs use similar model architectures with similar number of model parameters for fair comparison. For split MNIST and permuted MNIST, we use several fully-connected layers. For CIFAR-10 and CIFAR-100, we use a convolutional network (Supplement Section D for details). For CIFAR-100, all compared models used convolutional layers that were pre-trained on CIFAR-10. Similar training regimes were used for the EBMs and baselines. On split MNIST, permuted MNIST, and CIFAR-10, we trained for 2000 iterations per task. On CIFAR-100, we trained for 5000 iterations per task. All experiments used the Adam optimizer with learning rate $1e^{-4}$. Each experiment was performed at least 10 times with different random seeds, with results reported as the mean $\pm$ SEM in Table 1. EBMs have a significant improvement over the baseline methods on all the datasets, showing that EBMs forget less when updating models for new tasks.

EBM training objective on existing approaches. Our proposed energy-based training objective is simple and can also be directly applied to existing CL approaches. We test this by modifying

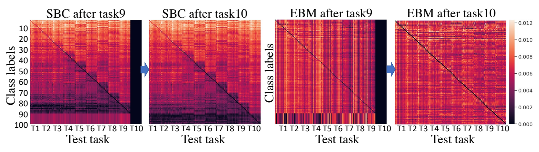

Figure 2: Energy landmaps of SBC and EBMs after training on task T_9 and T_{10} on permuted MNIST. The darker the diagonal is, the better the model is in preventing forgetting previous tasks.

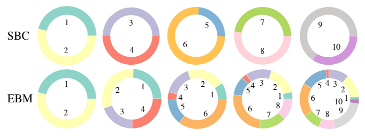

Figure 3: Predicted label distribution after learning each task on the split MNIST dataset. The SBC only predicts classes from the current task, while our EBM predicts classes for all seen classes.

the training objective of baseline models to that of our proposed energy objective, which computes the softmax normalization only over class labels in the current training batch. In Table 2, we find that our proposed objective significantly improves the performance of different CL methods. This is because the new training objective does not suppress the probability of old classes when improving the probability of new classes. Our training objective provides an orthogonal direction to tackle the CL problem and is simple to implement on existing CL approaches.

Effect of energy training objective. We conduct an experiment on the CIFAR-100 dataset to investigate how different EBM training objectives influence the CL results. We compare three strategies for selecting the negative samples as described in Section 3.3.1. The first strategy uses all seen classes so far as negative labels (**All Neg Seen**). The second one takes all the classes in the current batch as negative labels (**All Neg Batch**). The last one randomly selects one class from the current batch as the negative as described in Equation 5 (**1 Neg Batch**). In Table 3, we find using only one negative sample generates the best result, and using negatives sampled from classes in the current batch is better than from all seen classes. Since our EBM training objective aims at improving the energy of negative samples while decreasing the energy of positive ones, sampling negatives from the current batch has less interference with previous classes than sampling from all seen classes.

4.1.3 Qualitative analysis

Most existing CL methods are based on the softmax-based classifiers (Pellegrini et al., 2019; Zenke et al., 2017). To better understand why EBMs suffer less from catastrophic forgetting, we qualitatively compare our EBMs and the standard SBC model shown in Figure 1 in this part.

Energy landscape. In Figure 2, we show the energy landscapes after training on task 9 and task 10 of the permuted MNIST dataset. For SBC, the energy is given by the negative of the predicted probability. Each datapoint has 100 energy values (EBM) or probabilities (SBC) corresponding to the 100 labels in the dataset. Dark elements on the diagonal indicate correct predictions. After training on task T_9, SBC assigns high probabilities to classes from T_9 (80-90) for almost all the data from T_1 to T_9. After learning T_{10}, the highest probabilities shift to classes from T_{10} (90-100). SBC tends to assign high probabilities to new classes for both old and new data, indicating forgetting. In contrast, EBM has low energies across the diagonal, which means that after training on new tasks, EBM still assigns low energies to the true labels of data from previous tasks. This shows that EBM is better than SBC at learning new tasks without catastrophically forgetting of old tasks.

Predicted class distribution. In Figure 3, for the split MNIST dataset, we plot the proportional distribution of predicted classes. Only data from the tasks seen so far was used for this figure. Taking the second panel in the first row as an example, it shows the distribution of predicted labels on test data from the first two tasks after finishing training on the second task. Since the number of test images from each class are similar, the ground truth proportional distribution should be uniform over those four classes. After training on the first task, the predictions of SBC are roughly uniformly distributed over the first two classes (first panel). However, after learning new tasks, SBC only predicts classes from the most recent task and fails to correctly memorize previous classes. In contrast, the predictions of EBM are substantially more uniformly distributed over all seen classes.

we provide further comparisons between SBC and EBMs in Supplement Section B and Section C.

4.2 Experiments on Boundary-Agnostic Setting

When applying continual learning in real life, boundaries are not usually well defined between different tasks. However, most existing CL methods rely on the presence of sharp boundaries between tasks to determine when to consolidate the knowledge. We show that EBMs are able to flexibly perform CL across different setups, and perform well on the *boundary-agnostic* setting as well.

4.2.1 Datasets and evalution protocols

For the *boundary-agnostic* setting, we use the same datasets as the *boundary-aware* setting in Section 4.1.1. We use the code of "continuous task-agnostic learning" proposed by (Zeno et al., 2018) to generate a continually changing data stream. See supplement Section E.1 for details. All experiments are performed according to the *Class-IL* scenario.

4.2.2 Comparison with existing methods

We focus on the investigation of the performance of different model architectures with similar model size and memory footprint, and thus do no compare with replay-based methods. Since there is no knowledge on the number of tasks, many methods of continual learning that rely on task boundaries are generally inapplicable. One trivial adaptation is to take the core action after every batch step instead of every task. However, doing such adaptation is impractical for most algorithms, such as EWC, because of the large computational complexity. However, we managed to run the Online EWC, SI, and BGD baselines in this way. All compared methods used similar model architectures as in the *boundary-aware* setting. Each experiment was performed 5 times with different random seeds, the results are reported as the mean ± SEM over these runs. The results are shown in Table 4. We observe that EBMs have a significant improvement on all the datasets. The experiments show that EBMs have good generalization ability for different continual learning problems as EBMs can naturally handle data streams with and without task boundaries.

Table 4: Evaluation of class-incremental learning performance on the *boundary-agnostic* setting.

Method	**splitMNIST**	**permMNIST**	**CIFAR-10**	**CIFAR-100**
SBC	24.03 ± 0.59	21.42 ± 0.11	23.30 ± 0.81	9.85 ± 0.02
Online EWC	39.62 ± 0.14	41.37 ± 0.04	22.53 ± 0.41	9.57 ± 0.02
SI	28.79 ± 0.24	35.71 ± 0.11	26.26 ± 0.72	10.42 ± 0.01
BGD	21.65 ± 1.15	26.15 ± 0.22	17.03 ± 0.82	8.50 ± 0.02
EBM	$\mathbf{81.78 \pm 1.22}$	$\mathbf{92.35 \pm 0.11}$	$\mathbf{49.47 \pm 1.25}$	$\mathbf{34.39 \pm 0.24}$

5 Conclusion

In this paper, we show that energy-based models are a promising class of models in a variety of different continual learning settings. We demonstrate that EBMs exhibit many desirable characteristics to prevent catastrophic forgetting in continual learning, and we experimentally show that EBMs obtain strong performance on the challenging class-incremental learning scenario on multiple benchmarks, both on the boundary-aware and boundary-agnostic settings.

One drawback of the current EBM method is that it computes the energy of a data point and all class labels during inference. One way to speed up is to make hierarchical decisions, e.g., first decide whether it is a cat or a dog, then decide the specific breed. Study this case could be a good direction for future research.

References

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11254–11263, 2019.

Sergey Bartunov, Jack W Rae, Simon Osindero, and Timothy P Lillicrap. Meta-learning deep energy-based memory models. *arXiv preprint arXiv:1910.02720*, 2019.

David Belanger and Andrew McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, pp. 983–992, 2016.

Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. Initial classifier weights replay for memoryless class incremental learning. *arXiv preprint arXiv:2008.13710*, 2020.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual energy-based models for text generation. *arXiv preprint arXiv:2004.11714*, 2020.

Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. 2019.

Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation and inference with energy based models. *arXiv preprint arXiv:2004.06030*, 2020a.

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020b.

Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.

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.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

Michael Gygli, Mohammad Norouzi, and Anelia Angelova. Deep value networks learn to evaluate and iteratively refine structured outputs. *arXiv preprint arXiv:1703.04363*, 2017.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

Tyler L Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 220–221, 2020.

Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pp. 466–483. Springer, 2020.

Chen He, Ruiping Wang, Shiguang Shan, and Xilin Chen. Exemplar-supported generative reproduction for class incremental learning. In *BMVC*, pp. 98, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 831–839, 2019.

Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *International Conference on Learning Representations*, 2019.

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.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34, 1998.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative models from the perspective of continual learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019.

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

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019.

Nicolas Y Masse, Gregory D Grant, and David J Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018.

Martin Mundt, Yong Won Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *arXiv preprint arXiv:2009.01797*, 2020.

Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. *arXiv preprint arXiv:1903.12370*, 2019.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. *arXiv preprint arXiv:1912.01100*, 2019.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. 2020.

Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. In *Advances in Neural Information Processing Systems*, pp. 12669–12679, 2019.

Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13588–13597, 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

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

Amirmohammad Rooshenas, Dongxu Zhang, Gopal Sharma, and Andrew McCallum. Search-guided, lightly-supervised training of structured prediction energy networks. In *Advances in Neural Information Processing Systems*, pp. 13522–13532, 2019.

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.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In David van Dyk and Max Welling (eds.), *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR. URL `http://proceedings.mlr.press/v5/salakhutdinov09a.html`.

Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

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.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423*, 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.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. 2020.

Lifu Tu and Kevin Gimpel. Benchmarking approximate inference methods for neural structured prediction. *arXiv preprint arXiv:1904.01138*, 2019.

Lifu Tu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. Engine: Energy-based inference networks for non-autoregressive machine translation. *arXiv preprint arXiv:2005.00850*, 2020.

Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11:4069, 2020.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 374–382, 2019.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pp. 2635–2644, 2016.

Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of descriptor and generator networks. *IEEE transactions on pattern analysis and machine intelligence*, 42(1):27–45, 2018.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pp. 2048–2057, 2015.

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

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3987–3995. JMLR. org, 2017.

Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123*, 2018.

Appendices

In Section A, we provide more details about the alternative EBM training objective mentioned in the main paper Section 3.3.1. Section B and Section C show more comparisons and analysis of EBMs and baselines. In Section D, we provide the model architecture details of the proposed EBMs and baselines on difference continual learning datasets. Section E show some related works of different types of continual learning settings and approaches.

A ALTERNATIVE EBM TRAINING OBJECTIVE

A.1 ENERGY-BASED MODELS FOR CLASSIFICATION

In the main paper Section 3.3.1, we mentioned an alternative EBM training objective can further improve the continual learning results. Here we provide more details about this training objective. We propose to use the Boltzmann distribution to define the joint likelihood of image $\mathbf{x}$ and label y:

$$\begin{aligned} p_{\boldsymbol{\theta}}(\mathbf{x}, y) &= \frac{\exp(-E_{\boldsymbol{\theta}}(\mathbf{x}, y))}{Z(\boldsymbol{\theta})}, \\ Z(\boldsymbol{\theta}) &= \sum_{\mathbf{x}' \in \mathcal{X}, y' \in \mathcal{Y}} \exp(-E_{\boldsymbol{\theta}}(\mathbf{x}', y')) \end{aligned} \tag{6}$$

where $E_{\boldsymbol{\theta}}(\mathbf{x}, y) : (\mathbb{R}^D, \mathbb{N}) \to \mathbb{R}$ is the energy function that maps an input-label pair $(\mathbf{x}, y)$ to a scalar energy value, and $Z(\boldsymbol{\theta})$ is the partition function for normalization.

Training. We want the distribution defined by $E_{\boldsymbol{\theta}}$ to model the joint data distribution p_D, which we do by minimizing the negative log likelihood of the data

$$\mathcal{L}_{\text{ML}}(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim p_D}[-\log p_{\boldsymbol{\theta}}(\mathbf{x}, y)]. \tag{7}$$

with the expanded form:

$$\mathcal{L}_{\text{ML}}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},y)\sim p_D}\left[E_{\boldsymbol{\theta}}(\mathbf{x}, y) + \log\Big(\sum_{\mathbf{x}' \in \mathcal{X}, y' \in \mathcal{Y}} e^{-E_{\boldsymbol{\theta}}(\mathbf{x}', y')}\Big)\right]. \tag{8}$$

Equation 8 minimizes the energy of $\mathbf{x}$ at the ground truth label y and minimizes the overall partition function by increasing the energy of any other randomly paired $\mathbf{x}'$ and y'.

Inference. Given an input $\mathbf{x}$, the class label predicted by our EBMs is the class with the smallest energy at $\mathbf{x}$:

$$\hat{y} = \arg\min_{y' \in \mathcal{Y}} E_\theta(\mathbf{x}, y'), \tag{9}$$

A.2 ENERGY-BASED MODELS FOR CONTINUAL LEARNING

As described in the main paper Section 3.3.1, directly maximizing energy across all labels of a data point $\mathbf{x}$ raises the same problem as the softmax-based classifier models that the old classes are suppressed when training a model on new classes and thus cause catastrophic forgetting.

Inspired by Hinton (2002); Du & Mordatch (2019); Xie et al. (2016), we find that the contrastive divergence approximation of Equation 8 can mitigate this problem and lead to a simpler equation. We approximate Equation 8 by sampling a random pair of image $\mathbf{x}'$ and label y' from the current training batch to approximate the partition function. Our training objective is given by:

$$\mathcal{L}_{\text{CD}}(\boldsymbol{\theta}; \mathbf{x}, y) = \mathbb{E}_{(\mathbf{x},y)\sim p_D}\left[E_{\boldsymbol{\theta}}(\mathbf{x}, y) - E_{\boldsymbol{\theta}}(\mathbf{x}', y')\right], \tag{10}$$

where y is the ground truth label of data $\mathbf{x}$.

This training objective is reminiscent of the contrastive divergence training objective used to train EBMs in the main paper Equation 5. The major difference is that we utilize both images and labels from the current batch as our contrastive samples instead of just labels used in the main paper Equation 5. We show in the experiments that using the proposed contrastive training objective in this supplement Equation 10 can further improve the continual learning performance.

Table 5: Evaluation of class-incremental learning on the *boundary-aware* setting on the split MNIST and permuted datasets. Each experiment is performed at least 10 times with different random seeds, the results are reported as the mean $\pm$ SEM over these runs. Note our comparison is restricted to methods that do not replay stored or generated data.

Method	**splitMNIST**	**permMNIST**
SBC	19.90 ± 0.02	17.26 ± 0.19
EWC	20.01 ± 0.06	25.04 ± 0.50
Online EWC	19.96 ± 0.07	33.88 ± 0.49
SI	19.99 ± 0.06	29.31 ± 0.62
LwF	23.85 ± 0.44	22.64 ± 0.23
MAS	19.50 ± 0.30	-
BGD	19.64 ± 0.03	84.78 ± 1.30
EBM	53.12 ± 0.04	87.58 ± 0.50
EBM Alt CD	$\mathbf{60.14 \pm 1.66}$	$\mathbf{89.15 \pm 0.89}$

A.3 Inference

We use the same inference methods as described in the main paper Section 3.3.3 to perform the *Class-IL* evaluation on the continual learning datasets. The model predicts the class label $\hat{y}$ of a data point $\mathbf{x}_k$ from all class labels, where $\mathbf{x}_k$ is one data point from a batch $\mathcal{B}_k$ with an associated discrete label $y \in \mathcal{Y}_k$ and $\mathcal{Y}_k$ contains the class labels in $\mathcal{B}_k$. Then there are $\mathcal{Y} = \bigcup_{k=1}^{K} \mathcal{Y}_k$ different class labels in total after seeing all the batches. The MAP estimate is

$$\hat{y} = \arg\min_{y} E_{\theta_K}(\mathbf{x}_k, y), \quad y \in \bigcup \mathcal{Y}_k, \tag{11}$$

where $E_{\theta_K}(\mathbf{x}_k, y)$ is the energy function with parameters θ_K resulting from training on the batches $\{\mathcal{B}_1, \cdots, \mathcal{B}_K\}$.

A.4 Comparisons with existing methods

We follow the experiments performed in the main paper Section 4.1.2 and evaluate the *Class-IL* on the split MNIST (Zenke et al., 2017) and permuted MNIST (Kirkpatrick et al., 2017) datasets on the *Boundary-Aware* setting.

We compare EBMs using different training objectives and the baseline approaches in this supplement Table 5. All the baselines and EBMs use similar model architectures with similar number of model parameters for fair comparison. "EBM" means the results of the training objective used in the main paper Equation 5. "EBM Alt CD" represents the alternative training objective described in this supplement Equation 10. EBMs have a significant improvement over the baseline methods on all the datasets, showing that EBMs forget less when updating models for new tasks. "EBM Alt CD" can further improve the continual learning performance.

B Additional experiments

B.1 Is the strong performance of EBMs due to the Energy Training Objective or due to the Label Conditioning?

In the main paper Section 4.1.2, we investigate the effect of **energy training objective**. Here we conduct more experiments to investigate the effect of the label conditioning architecture in EBMs.

Effect of label conditioning. We test whether the label conditioning in our EBMs is important for their performance in the main paper Table 2. As mentioned in Section 4.1.2, we modify baseline models using our training objective. EBMs still outperform the modified baselines, implying that the label conditioning architecture also contributes to why EBMs suffer less from catastrophic forgetting.

We further show the testing accuracy of each task as the training progresses in Figure 4. We compare the standard classifier (SBC), classifier using our training objective (SBC*), and our EBMs on two different datasets. The left figures show results on the split MNIST dataset while the right figures show results on the permuted MNIST dataset. We find that the accuracy of old tasks in SBC drop sharply when learning new tasks, while the EBM training objective used in SBC* is better. The curve on EBMs drops even slower than SBC*, implying our EBMs can mitigate the forgetting problem.

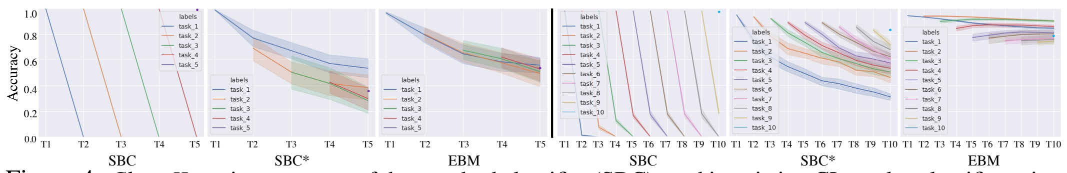

Figure 4: *Class-IL* testing accuracy of the standard classifier (SBC) used in existing CL works, classifier using our training objective (SBC*), and EBMs on each task on the split MNIST dataset (left) and permuted MNIST dataset (right).

Table 6: Performance of EBM on CIFAR-10 with different label conditioning architectures.

Model architectures		**Normalization types**	
Beginning (V1)	13.69 ± 1.12	End Fix (V4)	34.30 ± 1.03
Middle(V2)	20.16 ± 1.05	End Fix Norm2 (V4)	33.91 ± 1.13
Middle(V3)	18.36 ± 0.97	End Fix Softmax (V4)	35.97 ± 1.09
End (V4)	38.13 ± 0.59	End Norm2 (V4)	37.23 ± 1.20
		End Softmax (V4)	**38.84 ± 1.08**

To summarize, we show that the strong performance of our EBMs is due to both the EBM training objective and the label conditioning architecture. Moreover, these results indicate that surprisingly, and counterintuitively, directly optimizing the cross-entropy loss used by existing approaches may not be the best way to approach continual learning.

B.2 Comparison of different architectures

EBMs allow flexibility in integrating data information and label information in the energy function. To investigate where and how to combine the information from data $\mathbf{x}$ and label y, we conduct a series of experiments on CIFAR-10. Table 6 shows four model architectures (V1-V4) that combine $\mathbf{x}$ and y in the early, middle, and late stages, respectively (see Supplement Section D for more details). We find combining $\mathbf{x}$ and y in the late stage (V4) performs the best. We note that instead of learning a feature embedding of label y, we can use a fixed projection matrix which is randomly sampled from the uniform distribution $\mathcal{U}(0, 1)$. Even using this fixed random projection can already generates better results than most baselines in Table 1. Note further that the number of trainable parameters in the "Fix" setting is much lower than that of the baselines. Using a learned feature embedding of y can further improve the result. We may also apply different normalization methods over the feature channel of y. We find that Softmax (**End Softmax (V4)**) is better than the L2 normalization (**End Norm2 (V4)**) and no normalization (**End (V4)**).

B.3 Effect of different numbers of tasks

To test the generality of our proposed EBMs, in Table 7 we repeat the boundary-aware experiments on CIFAR-100 for different number of classes per task. In Table 1, the CIFAR-100 dataset was split up into 10 tasks, resulting in 10 classes per task. Here we additionally split CIFAR-100 up into 5 tasks (i.e., 20 classes per task), 20 tasks (i.e., 5 classes per task) and 50 tasks (i.e., 2 classes per task). Our EBM substantially outperforms the baselines on all settings.

B.4 Can EBMs use Replay?

Although EBMs already achieve good performance on *Class-IL* without using any replay, we found that EBMs are flexible enough to be combined with replay-based approaches to further improve the

Table 7: Comparison of our EBM with baselines on different variants of the split CIFAR-100 protocol. Results of the *Class-IL* performance on the *boundary-aware* setting are reported.

Method	**CIFAR-100 split up into:**			
	5 tasks	**10 tasks**	**20 tasks**	**50 tasks**
SBC	14.74 ± 0.20	8.18 ± 0.10	4.46 ± 0.03	1.91 ± 0.02
EWC	14.78 ± 0.21	8.20 ± 0.09	4.46 ± 0.03	1.91 ± 0.02
SI	14.07 ± 0.24	9.24 ± 0.22	4.37 ± 0.04	1.88 ± 0.03
LwF	25.75 ± 0.14	10.71 ± 0.11	12.18 ± 0.16	7.68 ± 0.16
EBM	**34.88 ± 0.14**	**30.28 ± 0.28**	**25.04 ± 0.33**	**13.60 ± 0.50**

Table 8: Comparisons of the softmax-based classifier and EBMs using exact replay on four datasets. Results of *Class-IL* on the *boundary-aware* setting are reported. k is the memory budget size.

Method	**splitMNIST**	**permMNIST**	**CIFAR-10**	**CIFAR-100**
	k=1000	k=1000	k=1000	k=2000
SBC ER	90.65 ± 0.45	93.70 ± 0.09	42.07 ± 0.64	28.57 ± 0.35
EBM ER	91.13 ± 0.35	94.59 ± 0.09	44.76 ± 0.73	34.07 ± 0.55

performance. In Table 8, we show the results of SBC and EBM using exact replay. When training a new task, we mix the new data with data sampled from a memory buffer that stores examples of previously learned tasks to train the models. The examples stored in the buffer are randomly selected from the classes encountered so far, and the available memory budget k is equally divided over all classes encountered so far. On the split MNIST, permuted MNIST, and CIFAR-10 datasets, we use a memory budget of $k = 1000$. On the CIFAR-100 datasets, we use a memory budget of $k = 2000$.

Taking the split MNIST dataset as an example, after we finishing training the first task, we randomly select 1000 data-label pairs from the first task and save them in the replay buffer. Then we train the model on the second task. In each training batch, we randomly sample a set of data-label pairs from the second task, and randomly sample a set of data-label pairs from the replay buffer. We compute the final loss by adding the loss of data from the current task $\mathcal{L}_{\text{CD current}}(\boldsymbol{\theta}; \mathbf{x}, y)$ and the loss of data from the replay buffer $\mathcal{L}_{\text{CD replay}}(\boldsymbol{\theta}; \mathbf{x}, y)$ using the following equation:

$$\mathcal{L}_{\text{CD}}(\boldsymbol{\theta}; \mathbf{x}, y) = \mathcal{L}_{\text{CD current}}(\boldsymbol{\theta}; \mathbf{x}, y) + \mathcal{L}_{\text{CD replay}}(\boldsymbol{\theta}; \mathbf{x}, y), \tag{12}$$

After we finishing training the second task, we randomly select 500 data-label pairs from the replay buffer and randomly sample 500 data-label pairs from the second task and update the replay buffer using the new sampled data-label pairs. Note we always keep the number of data-label pairs in the replay buffer at 1000. Then we train the model on the third task. In each training batch, we randomly sample a set of data-label pairs from the third task, and randomly sample a set of data-label pairs from the replay buffer. We compute the final loss using Equation 12. We use such a strategy to train the models until the final task.

The baseline model, *i.e*. "SBC ER", in Table 8 uses replay in the same way. The only difference from our EBM is that "SBC ER" uses the cross-entropy loss as described in the main paper Section 2 but "EBM ER" uses the proposed EBM training objective. Each experiment was performed 5 times with different random seeds, with results reported as the mean ± SEM. We use the similar training regimes for EBMs and SBC. On split MNIST, permuted MNIST, and CIFAR-10, we trained for 2000 iterations per task. On CIFAR-100, we trained for 5000 iterations per task. In each training batch, we sampled 128 data-label pairs from the current task and 128 data-label pairs from the replay buffer (start from the second task) to train the model. All experiments used the Adam optimizer with learning rate $1e^{-4}$.

We report the results on 4 datasets. Note these numbers might be slightly different from results reported in some existing works because of the usage of different model architectures, different memory sizes, and different ways to split the datasets. In our experiments, we control the baselines and EBMs to have similar model architectures with similar number of model parameters and the same buffer sizes on each dataset. After using extra memory, both SBC and EBM have improvements. EBMs still outperform SBC, especially on more challenging datasets, e.g., CIFAR-10 and CIFAR-100. Interestingly, on CIFAR-100, we find that EBMs without using replay (30.28%; Table 1) perform better than SBC with using replay (28.57%; Table 8).

Again our focus is try to address the CL problems without using replay or stored data. Our results show that the proposed EBM formulation provides an orthogonal direction to tackle the CL problems. This approach can be further combined with existing CL methods.

B.5 Class confusion matrix at the end of learning

We show confusion matrices for EBM and SBC. A confusion matrix illustrates the relationship between the ground truth labels and the predicted labels. Figure 5 in this supplement shows the confusion matrices after training on all the tasks on the split MNIST dataset and permuted MNIST dataset. The standard classifier tends to only predict the classes from the last task (class 8, 9 for split MNIST and classes 90-100 for permuted MNIST). The EBMs on the other hand have high values

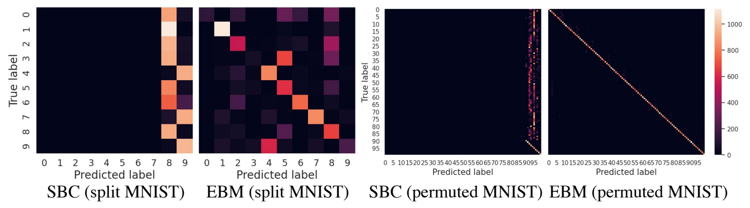

Figure 5: Confusion matrices between ground truth labels and predicted labels at the end of learning on split MNIST (left) and permuted MNIST (right). The lighter the diagonal is, the more accurate the predictions are.

Table 9: The model architectures used for the model capacity analysis. h are 512, 1024, and 4096 for the small, medium and large network, respectively.

(a) The architecture of EBMs.

x = FC(32 × 32 × 3, h) (x)
x = ReLU (x)
y = Embedding (y)
x = x * y
x = ReLU (x)
out = FC(h, 1) (x)

(b) The architecture of the standard classifier.

x = FC(32 × 32 × 3, h) (x)
x = ReLU (x)
x = FC(h, h) (x)
x = ReLU (x)
out = FC(h, 10) (x)

along the diagonal, which indicates that the predicted results match the ground truth labels for all the sequentially learned tasks.

C Additional Analyses

Extending the results presented in the main paper Section 4.1, here we further compare EBMs with the baseline models by providing additional quantitative analyses of their performance. We show the model capacity comparisons in Section C.1 and parameter importance measurement in Section C.2.

C.1 Model capacity

Another hypothesized reason for why EBMs suffer less from catastrophic forgetting than standard classifiers is potentially their larger effective capacity. To analyze effective capacity of our models, we test the model capacity of the standard classifier and EBMs on both the generated images and natural images.

Model capacity on generated images. We generate a large randomized dataset of 32×32 images with each pixel value uniformly sampled from -1 to 1. Each image is then assigned a random class label between 0 and 10. We measure the model capacity by evaluating to what extent the model can fit a such dataset. For both the standard classifier and the EBM, we evaluate three different sizes of models (small, medium, and large). For a fair comparison, we control the EBM and classifier have similar number of parameters. The Small EBM and SBC have $2,348,545$ and $2,349,032$ parameters respectively. The medium models have $5,221,377$ (EBM) and $5,221,352$ (SBC) parameters while the large models have $33,468,417$ (EBM) and $33,465,320$ (SBC) parameters. We use the model architectures in this supplement Table 9 for EBMs and classifiers.

The resulting training accuracies are shown in this supplement Figure 6 with the number of data ranges from one to five millions. Given any number of datapoints, EBM obtains higher accuracy than the classifier, demonstrating that indeed EBM has larger capacity to memorize data given a similar number of parameters. The gap between EBM and SBC increases when the models become larger. The larger capacity of EBM potentially enables it to memorize more data and mitigate the forgetting problem.

Model capacity on natural images. We also compare classifiers and EBMs on natural images from CIFAR-10. Each image is assigned a random class label between 0 and 10. We use the same network architecture as in Table 9 but with a hidden unit size of $h = 256$. Since there are only $50,000$ images on CIFAR-10, we use a small classifier and EBM and train them on the full dataset.

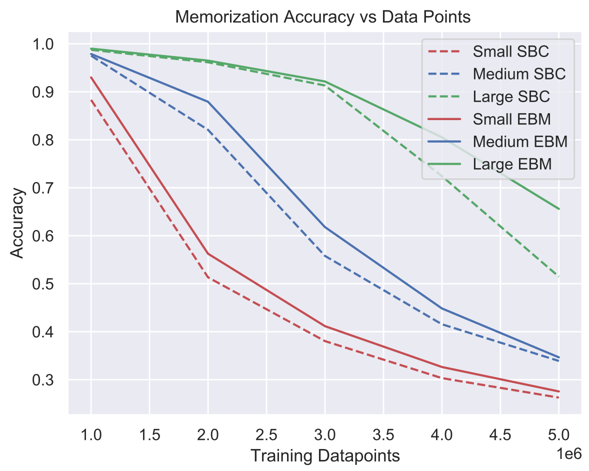

Figure 6: Model capacity of the standard classifier (SBC) and EBM using different model sizes.

After training 100000 iterations, the EBM obtains a top-1 prediction accuracy of 82.81, while the classifier is 42.19. We obtain the same conclusion that EBM has larger capacity to memorize data given a similar number of parameters.

C.2 Parameter importance

To further understand why EBMs suffer less from catastrophic forgetting than standard classifiers, we design an experiment to test the importance of model parameters on past data. Inspired by the elastic weight consolidation (EWC) (Kirkpatrick et al., 2017), we estimate the importance of parameters for each tasks using the diagonal elements of the Fisher information matrix (FIM) F. Let θ_i be the model parameters after training on task T_i. Given one of previous tasks $T_j, j < i$, we evaluate how important each parameter is for tasks T_j. The k^{th} diagonal of F is defined as the gradient on the EBM loss

$$F_{i,k} = \mathbb{E}_{\mathbf{x}\sim T_j}\left[\left(\nabla_{\theta_{i,k}}\left(E_{\theta_i}(\mathbf{x}, y) - E_{\theta_i}(\mathbf{x}, y^-)\right)\right)^2\right], \tag{13}$$

where $\mathbf{x}$ is sampled from tasks T_j and $E(\mathbf{x}, y)$ is the energy value of the input data $\mathbf{x}$ and ground truth label y. The class label $y^- \in \mathcal{Y}_j$ are randomly selected from the current batch. Here we use a single negative class. The above equation assigns high values to parameters crucial to task T_j as their gradients with respect to the loss are larger. Since the diagonal elements of the fisher information matrix measure the importance of each parameter to a given task, the density of diagonal elements represents the proportion of important parameters over all parameters. More density means more parameters are important for the given task and less parameters can be recruited for new tasks. Ideally, we expect these values to be sparse.

In this supplement Figure 7, we show the diagonal elements of the standard classifier (SBC), classifier using our training objective (SBC*), and our EBMs on the split MNIST dataset. For SBC and SBC*, we follow (Kirkpatrick et al., 2017) to compute their fisher information matrices. For comparisons across multiple models, we normalize the FIM diagonal elements of each method to be between 0 and 1 and report the normalized results in Figure 7. For example, "Fisher 5 on data 1" shows the diagonal elements of the Fisher information matrix obtained by Equation 13 using the model parameters θ_5 (after training on task T_5) and data $\mathbf{x}, y, y^-$ from task T_1. The distribution of EBMs is sparser than SBC and SBC* indicating that EBMs have fewer important parameters for previous data. Updating parameters for the new task will have less negative impact on old tasks. In addition, more parameters can be used for learning new tasks if the distribution is sparse. This may provide another explanation for why EBMs can mitigate catastrophic forgetting.

D Model architectures

In this section, we provide details of the model architectures used on the different datasets.

Images from the split MNIST and permuted MNIST datasets are grey-scale images. The baseline models for these datasets, similar as in (van de Ven & Tolias, 2019), consist of several fully-

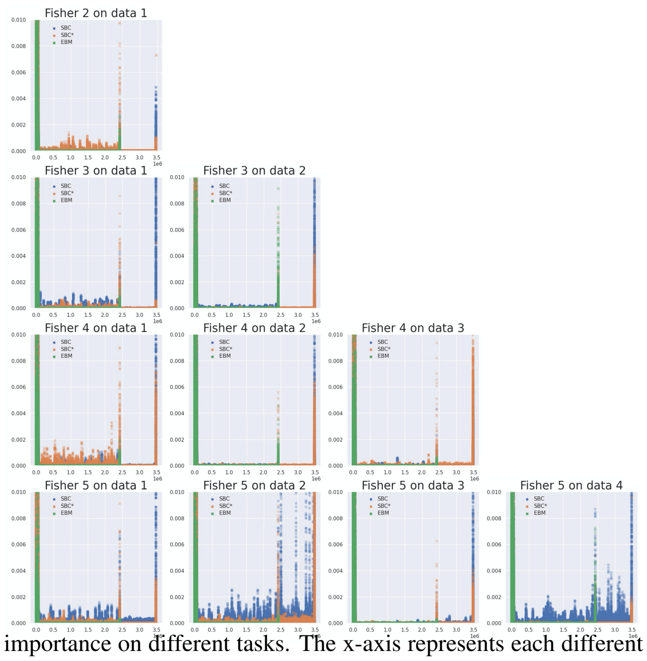

Figure 7: Parameter importance on different tasks. The x-axis represents each different parameter and y-axis is the FIM value in Equation 13. The sparser the parameters are, the fewer important parameters there are for previous data. The sparsity of parameter importance in EBMs may explain why EBMs have less influence on previous tasks after training on new data. "Fisher 5 on data 1" means the diagonal elements of the fisher information matrix obtained by Equation 13 using the model parameters θ_5 (after training on task T_5) and data from task T_1.

Table 10: The model architectures used on split MNIST. $h = 400$.

(a) The architecture of the EBMs.

x = FC(784, h) (x)
x = ReLU (x)
y = Embedding (y)
x = x * Norm2 (y) + x
x = ReLU (x)
out = FC(h, 1) (x)

(b) The architecture of the baseline models.

x = FC(784, h) (x)
x = ReLU (x)
x = FC(h, h) (x)
x = ReLU (x)
out = FC(h, 10) (x)

connected layers. For the EBMs we use similar number of parameters. The model architectures of EBMs on the split MNIST dataset and permuted MNIST dataset are the same, but have different input and output dimensions and hidden sizes. The model architectures of EBMs and baseline models on the split MNIST dataset are shown in this supplement Table 10. The model architectures of EBMs and baseline models on the permuted MNIST dataset are shown in Table 11.

Images from the CIFAR-10 and CIFAR-100 datasets are RGB images. For CIFAR-10, we use a small convolutional network for both the baseline models and the EBMs. The model architectures of EBMs and baseline models on the CIFAR-10 dataset are shown in Table 12. We investigate different architectures to search for the effective label conditioning on EBMs training as described in the main paper Section 4.1.2. The model architectures used on the CIFAR-100 dataset are detailed in Table 13.

E Related work

E.1 Continual learning settings

Boundary-aware versus boundary-agnostic. In most existing CL studies, models are trained in a "boundary-aware" setting, in which a sequence of distinct tasks with clear task boundaries is given (e.g., Kirkpatrick et al., 2017; Zenke et al., 2017; Shin et al., 2017). There are no overlaps between

Table 11: The model architectures used on permuted MNIST. $h = 1000$.

(a) The architecture of the EBMs.

x = FC(1024, h) (x)
x = ReLU (x)
y = Embedding (y)
x = x * Norm2 (y) + x
x = ReLU (x)
out = FC(h, 1) (x)

(b) The architecture of the baseline models.

x = FC(1024, h) (x)
x = ReLU (x)
x = FC(h, h) (x)
x = ReLU (x)
out = FC(h, 100) (x)

any two tasks; for example task 1 has data class labels "1,2" and task 2 has data with class labels "3,4". Models are first trained on the first task and then move to the second one. Moreover, models are typically told when there is a transition from one task to the next. However, it could be argued that it is more realistic for tasks to change gradually and for models to not be explicitly informed about the task boundaries. Such a boundary-agnostic setting has been explored in (Zeno et al., 2018; Rajasegaran et al., 2020; Aljundi et al., 2019). In this setting, models learn in a streaming fashion and the data distributions gradually change over time. In (Zeno et al., 2018), the percentage of "1s" gradually decrease while the percentage of "2s" increases during training. Importantly, most existing CL approaches are not applicable to this setting as they require the task boundaries to decide when to perform certain consolidation steps. In this paper, we will show that our proposed approach can naturally handle both the boundary-aware and boundary-agnostic settings.

Task-incremental versus class-incremental learning. Another important distinction in CL is between task-incremental learning (*Task-IL*) and class-incremental learning (*Class-IL*) (van de Ven & Tolias, 2019; Prabhu et al., 2020). In *Task-IL*, also referred to as the multi-head setting (Farquhar & Gal, 2018), models predict the label of an input data by choosing only from the labels in the task where the data come from. In *Class-IL*, also referred to as the single-head setting, models chose between the classes from all tasks so far when asked to predict the label of an input data. *Class-IL* is more challenging than *Task-IL* as it requires models to select the correct labels from the mixture of new and old classes. Generally, to perform well on *Class-IL*, existing methods need to store data, use replay, or pretrain models from another large dataset (Rebuffi et al., 2017; Rajasegaran et al., 2019; Belouadah et al., 2020; Maltoni & Lomonaco, 2019; Hayes & Kanan, 2020).

E.2 Continual learning approaches

Numerous methods have been proposed for continual learning. Here we broadly partition them into three categories: task-specific, regularization, and replay-based approaches.

Task-specific methods. One way to reduce interference between tasks is by using different parts of a neural network for different tasks. For a fixed-size network, such specialization could be achieved by learning a separate mask for each task (Fernando et al., 2017; Serra et al., 2018), by *a priori* defining a different, random mask for every task (Masse et al., 2018), or by using a different set of parameters for each task (Zeng et al., 2019; Hu et al., 2019). Other methods let models grow or recruit new resources when learning new tasks, such as progressive neural networks (Rusu et al., 2016) and dynamically expandable networks (Yoon et al., 2017). Although these methods are generally successful in reducing catastrophic forgetting, a key disadvantage is that they require knowledge of task identities during training and testing. They are therefore not suitable for *Class-IL*.

Regularization-based methods. Regularization is used in CL to encourage the stability of those aspects of the network that are important for previous tasks. A popular strategy is to add a regularization loss to penalise changes of important parameters. EWC (Kirkpatrick et al., 2017)) and online EWC (Schwarz et al., 2018) evaluate the importance of parameters using the diagonal elements in the fisher information matrices, while SI (Zenke et al., 2017) estimates the parameters' importance online. LwF (Li & Hoiem, 2017) regularizes the network at the functional level. Although regularization-based methods can be computationally efficient, a disadvantage is that typically they gradually reduce the model's capacity for learning new tasks. Moreover, while in theory these methods can be used for *Class-IL*, in practice they have been shown to fail on such problems (Farquhar & Gal, 2018; van de Ven & Tolias, 2019).

Replay methods. To preserve knowledge, replay methods periodically rehearse previous information during training (Robins, 1995). Exact or experience replay based methods store data from

Table 12: The model architectures used on the CIFAR-10 dataset.

(a) EBM: **Beginning (V1)**

Input: x, y
y = Embedding(N, 3) (y)
y = Softmax(dim=-1) (y)
y = y * y.shape[-1]
x = x * y
x = Conv2d(3×3, 3, 32) (x)
x = ReLU (x)
x = Conv2d(3×3, 32, 32) (x)
x = ReLU (x)
x = Maxpool(2, 2) (x)
x = Conv2d(3×3, 32, 64) (x)
x = ReLU (x)
x = Conv2d(3×3, 64, 64) (x)
x = ReLU (x)
x = Maxpool(2, 2) (x)
x = FC(2304, 1024) (x)
x = ReLU(x)
out = FC(1024, 1) (x)

(b) EBM: **Middle (V2)**

Input: x, y
x = Conv2d(3×3, 3, 32) (x)
x = ReLU (x)
x = Conv2d(3×3, 32, 32) (x)
x = ReLU (x)
y = Embedding(N, 32) (y)
y = Softmax(dim=-1) (y)
y = y * y.shape[-1]
x = x * y
x = Maxpool(2, 2) (x)
x = Conv2d(3×3, 32, 64) (x)
x = ReLU (x)
x = Conv2d(3×3, 64, 64) (x)
x = ReLU (x)
x = Maxpool(2, 2) (x)
x = FC(2304, 1024) (x)
x = ReLU(x)
out = FC(1024, 1) (x)

(c) EBM: **Middle (V3)**

Input: x, y
x = Conv2d(3×3, 3, 32) (x)
x = ReLU (x)
x = Conv2d(3×3, 32, 32) (x)
x = ReLU (x)
x = Maxpool(2, 2) (x)
x = Conv2d(3×3, 32, 64) (x)
x = ReLU (x)
x = Conv2d(3×3, 64, 64) (x)
x = ReLU (x)
y = Embedding(N, 64) (y)
y = Softmax(dim=-1) (y)
y = y * y.shape[-1]
x = x * y
x = Maxpool(2, 2) (x)
x = FC(2304, 1024) (x)
x = ReLU(x)
out = FC(1024, 1) (x)

(d) EBM: **End Fix Softmax (V4)**

Input: x, y
x = Conv2d(3×3, 3, 32) (x)
x = ReLU (x)
x = Conv2d(3×3, 32, 32) (x)
x = ReLU (x)
x = Maxpool(2, 2) (x)
x = Conv2d(3×3, 32, 64) (x)
x = ReLU (x)
x = Conv2d(3×3, 64, 64) (x)
x = ReLU (x)
x = Maxpool(2, 2) (x)
x = FC(2304, 1024) (x)
y = Random Projection (y)
y = Softmax(dim=-1) (y)
y = y * y.shape[-1]
x = x * y
out = FC(1024, 1) (x)

(e) EBM: **End Softmax (V4)**

Input: x, y
x = Conv2d(3×3, 3, 32) (x)
x = ReLU (x)
x = Conv2d(3×3, 32, 32) (x)
x = ReLU (x)
x = Maxpool(2, 2) (x)
x = Conv2d(3×3, 32, 64) (x)
x = ReLU (x)
x = Conv2d(3×3, 64, 64) (x)
x = ReLU (x)
x = Maxpool(2, 2) (x)
x = FC(2304, 1024) (x)
y = Embedding(N, 1024) (y)
y = Softmax(dim=-1) (y)
y = y * y.shape[-1]
x = x * y
out = FC(1024, 1) (x)

(f) Baseline models

Input: x
x = Conv2d(3×3, 3, 32) (x)
x = ReLU (x)
x = Conv2d(3×3, 32, 32) (x)
x = ReLU (x)
x = Maxpool(2, 2) (x)
x = Conv2d(3×3, 32, 64) (x)
x = ReLU (x)
x = Conv2d(3×3, 64, 64) (x)
x = ReLU (x)
x = Maxpool(2, 2) (x)
x = FC(2304, 1024) (x)
x = ReLU (x)
x = FC(1024, 1024) (x)
x = ReLU (x)
out = FC(1024, 10) (x)

previous tasks and revisit them when training on new tasks. Although straightforward, such methods face critical non-trivial questions, such as how to select the data to be stored and how to use them (Lopez-Paz & Ranzato, 2017; Hou et al., 2019; Wu et al., 2019; Mundt et al., 2020). An alternative

Table 13: The model architectures used on the CIFAR-100 dataset. Following van de Ven et al. (2020), for all models the convolutational layers were pre-trained on CIFAR-10. The 'BinaryMask'-operation fully gates a randomly selected subset of $X\%$ of the nodes, with a different subset for each y. Hyperparameter X was set using a gridsearch.

(a) The architecture of the EBMs.

Input: x, y
x = Conv2d(3×3, 3, 16) (x)
x = BatchNorm (x)
x = ReLU (x)
x = Conv2d(3×3, 16, 32) (x)
x = BatchNorm (x)
x = ReLU (x)
x = Conv2d(3×3, 32, 64) (x)
x = BatchNorm (x)
x = ReLU (x)
x = Conv2d(3×3, 64, 128) (x)
x = BatchNorm (x)
x = ReLU (x)
x = Conv2d(3×3, 128, 256) (x)
x = FC(1024, 2000) (x)
x = ReLU (x)
x = BinaryMask (x, y)
x = FC(2000, 2000) (x)
x = ReLU (x)
x = BinaryMask (x, y)
out = FC(2000, 1) (x)

(b) The architecture of the baseline models.

Input: x
x = Conv2d(3×3, 3, 16) (x)
x = BatchNorm (x)
x = ReLU (x)
x = Conv2d(3×3, 16, 32) (x)
x = BatchNorm (x)
x = ReLU (x)
x = Conv2d(3×3, 32, 64) (x)
x = BatchNorm (x)
x = ReLU (x)
x = Conv2d(3×3, 64, 128) (x)
x = BatchNorm (x)
x = ReLU (x)
x = Conv2d(3×3, 128, 256) (x)
x = FC(1024, 2000) (x)
x = ReLU (x)
x = FC(2000, 2000) (x)
x = ReLU (x)
out = FC(2000, 100) (x)

is to generate the replayed data. In (Shin et al., 2017), a generative model is sequentially trained to generate samples of previous tasks. While both types of replay can mitigate forgetting, an important disadvantage is that they are computationally relatively expensive. Additionally, storing data might not always be possible while incrementally training a generative model is a challenging problem in itself (Lesort et al., 2019; van de Ven et al., 2020).

In contrast, we propose EBMs for continual learning that reduce catastrophic forgetting without requiring knowledge of task-identity, without gradually restricting the model's learning capabilities, and without using stored data.