3EF: CLASS-INCREMENTAL LEARNING VIA EFFICIENT ENERGY-BASED EXPANSION AND FUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks suffer from catastrophic forgetting when sequentially learning tasks phase-by-phase, making them inapplicable in dynamically renewal systems. Class-incremental learning (CIL) aims to enable neural networks to learn different categories at multi-stages. Recently, dynamic-structure-based methods achieves remarkable performance. However, these methods train all modules in a coupled manner and do not consider possible conflicts among modules, resulting in increasing training costs and spoilage of eventual predictions. In this work, we propose a unifying energy-based theory and framework called **E**fficient **E**nergy-Based **E**xpansion and **F**usion (3EF) to analyze and achieve the goal of CIL. We demonstrate the possibility of training independent modules in a decoupled manner while achieving bi-directional compatibility among modules through two additionally allocated prototypes, and then integrating them into a unifying classifier with minimal cost. Furthermore, 3EF extends the exemplar-set to a more challenging setting, where exemplars are randomly selected and imbalanced, where 3EF maintains its performance when prior methods fail dramatically. Extensive experiments on three widely used benchmarks: CIFAR-100, ImageNet-100, and ImageNet-1000 demonstrate that 3EF achieves state-of-the-art performance in both the ordinary and challenging CIL settings.

1 INTRODUCTION

The ability to learn new knowledge continuously is necessary in this ever-changing world, which is seen as an important symbol of human intelligence. Applicable AI systems are expected to learn novel concepts in a stream form while preserving knowledge for prior ones. However, deep-neural-network-based systems, which, though, have achieved tremendous success under the closed-world assumption (Minker, 1982), encounter the notorious problem known as catastrophic forgetting (French, 1999; Golab & Özsu, 2003) that they abruptly forget the prior learned knowledge when they are directly fine-tuned on new tasks. To address this challenge, the class-incremental learning (CIL) field aims to design learning paradigms that enable deep neural networks to learn novel categories in multi-stages while maintaining discrimination abilities for prior ones (Rebuffi et al., 2017; Masana et al., 2020b; Delange et al., 2021).

So far, many approaches have been proposed to achieve the goal of CIL. Generally speaking, typical CIL methods can be categorized into two groups: regularization-based methods and dynamic-structure-based methods. Regularization-based methods (Kirkpatrick et al., 2017; Aljundi et al., 2018; Li & Hoiem, 2017; Rebuffi et al., 2017) add constraints (*e.g.*, parameter drift penalty) when updating, thus forcing the model to maintain crucial information for old categories. However, this kind of method typically falls into the stability-plasticity dilemma, without enough capacity to handle all categories simultaneously. Dynamic-structure-based methods (Yan et al., 2021; Li et al., 2021) expand new modules at each learning stage to enhance the model's capacity and learn the task-specific knowledge through the new module, achieving remarkable performance. Whereas, these methods face several critical challenges. First, with an increasing number of modules coupled together, the model becomes rather bloated, resulting in increasing training costs. Second, they directly retain all learned modules without considering conflicts among modules, thus corrupting the joint feature representation and misleading the ultimate predictions. For example, old modules may mislead final prediction for new classes, since they are directly retained and have limited knowledge about new tasks. These defects make them unsuitable for long-term incremental tasks.

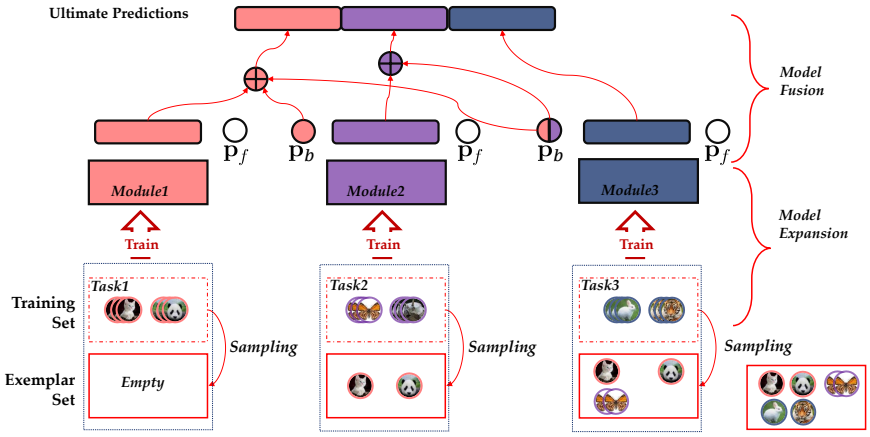


Figure 1: **The conceptual illustration of 3EF.** The training consists of two phases: expansion and fusion. At the expansion phase, we independently train the new module for the current task, while classifying all the samples from prior tasks into $\mathbf{p}_b$ and classifying all samples generated from the built-in energy-based model into $\mathbf{p}_f$. At the fusion phase, the output of $\mathbf{p}_b$ is equally added to the output of prior modules to mitigate the task-bias and form a unifying classifier.

Coupled training brings prohibitive training costs, so is it possible to train each module independently in a *decoupled* manner? That is, all the old modules are not involved in the new training process. Hence, the training cost is always equivalent to that of tuning a single module and will not increase in incremental stages. To solve the conflicts among different modules, we propose to achieve *bi-directional compatibility*, namely backward compatibility and forward compatibility. Here, *backward compatibility* is committed to making the discrimination ability of old modules unaffected by new ones. On the contrary, *forward compatibility* aims to make old modules sensitively capture input distribution shift and automatically reduce their impact on the ultimate predictions when new categories emerge. By achieving bi-directional compatibility, given a sample from a specific task, the module responsible for the task will dominate the ultimate predictions.

Motivated by this, we propose a novel yet unifying perspective from the energy-based model (EBM) (LeCun et al., 2006), dubbed as **E**fficient **E**nergy-Based **E**xpansion and **F**usion (3EF). Fig. 1 displays our 3EF training framework, which is made of two phases (Model Expansion and Model Fusion). At the expansion phase, we create a new module and train it isolatedly without the involvement of the other modules. Then, we propose an efficient fusion strategy to combine all the modules and form a unifying classifier for all classes with minimal cost. To achieve bi-directional compatibility, we introduce two additional prototypes: forward prototype $\mathbf{p}_f$ and backward prototype $\mathbf{p}_b$ when tuning a single module. The backward prototype measures the confidence of the current module for old classes. By setting $\mathbf{p}_b$ as the prototype for all old classes, it forces the new module to learn a task boundary between current classes and prior ones and effectively reduces the risk of overfitting caused by the limited access to old samples. The forward prototype aims to measure uncertainty in the open world. Specifically, it helps to transform the open world problem into a closed world form and introduces an adversarial training process. Combining the forward prototype and the theory of EBMs, we can model the input distribution synchronously while learning the discrimination ability, which makes the model representation robust and sensitive to the input distribution shift. We theoretically prove that the overall training phase is based on a unifying energy modeling process. Besides, each module is inherently an energy-based model measuring the energy for given samples, where energy represents their open world uncertainty. When evaluating on the test set, since all categories in the test set are known, we additionally apply energy alignment to align the input samples to sub-spaces with lower energies, making the model more suitable for in-distribution discrimination and boosting performance. Vast experiments on three widely-used benchmarks show that our method achieves state-of-the-art performance. Besides, prevalent CIL methods all require a well-selected balanced exemplar-set for rehearsal, which might be impractical due to data privacy issues (Delange et al., 2021; Ji et al., 2014) and computation cost when choosing exemplars. Our method pushes it to a harder setting. With only randomly sampled data from prior tasks, 3EF maintains its effectiveness while the performance of other methods declines drastically.

2 Related Work

Incremental learning. Most recent studies on incremental learning are either task-based or class-based. The crucial difference between them is whether task-id is known at the evaluation phase (Van de

Ven & Tolias, 2019). Our work is a class-based method, which is typically called class-incremental learning (CIL). Prevalent CIL methods can be categorized into two classes: regularization-based, and dynamic-structure-based. *Regularization-based* methods impose constraints when learning tasks. Kirkpatrick et al. (2017); Aljundi et al. (2018) penalize the parameter drift. Li & Hoiem (2017); Rebuffi et al. (2017); Wu et al. (2019); Zhao et al. (2020) utilize knowledge distillation (Hinton et al., 2015) to constrain the model's output. Douillard et al. (2020) propose a novel spatial knowledge distillation. Zhou et al. (2022) propose the concept of *forward compatible* (Gheorghioiu et al., 2003; Shen et al., 2020) and squeeze the space for known categories, thereby reserving feature space for future categories. *Dynamic-structure-based* methods create new modules to enhance the capacity for learning new tasks. Yan et al. (2021); Li et al. (2021) combine all modules together to form a unifying classifier, but it leads to an increasing training cost. Douillard et al. (2021) applies transformer (Dosovitskiy et al., 2020; Touvron et al., 2021) to CIL and dynamically expands task tokens when learning new tasks. Wang et al. (2022) proposes to dynamically expand and compress the model based on gradient boosting (Mason et al., 1999), thereby adaptively learning new tasks. Besides, prevalent CIL approaches usually require a well-selected class-balanced exemplar-set for rehearsal (Rebuffi et al., 2017), which has an evident impact on their performance (Masana et al., 2020a) as we verify experimentally. (Liu et al., 2021) proposes a novel dynamic memory management strategy that is optimized for the incremental phases. 3EF not only achieves state-of-the-art performance but shows strong robustness to the choice of exemplar-set.

Energy-based learning. EBMs define probability distributions with density proportional to $\exp(-E)$, where E is the energy function (LeCun et al., 2006). So far, the theory and implementation of EBMs have been well studied. Xie et al. (2016) show that the generative random field model can be derived from the discriminative ConvNet. Xie et al. (2018) well study the cooperative training of two generative models for image modeling and synthesis. Additionally, Xie et al. (2022) study cooperative learning of two generative flow models. Nijkamp et al. (2019) propose to treat the non-convergent short-run MCMC as a learned generator model or a flow model and show that the model is capable of generating realistic samples. Xiao et al. (2021) propose to learn a VAE to initialize the finite-step MCMC for efficient amortized sampling of the EBM. Xiao et al. (2021) propose a symbiotic composition of a VAE and an EBM that can generate high-quality images while achieving fast traversal of the data manifold. Zhao et al. (2021) propose a multistage coarse-to-fine expanding and sampling strategy, which starts with learning a coarse-level EBM from images at low resolution and then gradually transits to learn a finer-level EBM from images at higher resolution. Besides, EBMs have been successfully applied in many fields such as data generation (Zhai et al., 2016; Zhao et al., 2016; Deng et al., 2020; Du & Mordatch, 2019), out-of-distribution detection (OOD) (Hendrycks & Gimpel, 2016; Bai et al., 2021; Liu et al., 2020; Lee et al., 2020; Lin et al., 2021), and density estimation (Silverman, 2018; Zhao et al., 2016), *etc*. Wang et al. (2021) model the open world uncertainty as an extra dimension in the classifier, achieving better calibration in OOD datasets. Grathwohl et al. (2019) propose to model the joint distribution $\mathbb{P}(\mathbf{x}, y)$ for the classification problem. Bian et al. (2022) apply energy-based learning for cooperative games and derive new player valuation methods. There have been several attempts to apply EBMs to incremental learning. Li et al. (2020) propose a novel energy-based classification loss and network structure for continual learning. Joseph et al. (2022) build an energy-based latent aligner that recovers the corrupted latent representation.

3 METHOD

In this section, we give a description of 3EF and how we apply energy-based model to CIL to efficiently learn a unifying classifier while achieving bi-directional compatibility. In Sec. 3.1, we first introduce some basic knowledge of CIL. In Sec. 3.2, we present the definition of energy and optimization objective at the expansion phase. Then, to avoid the intractability of normalizing constant, we prove a gradient equivalent objective and explain why it helps to achieve the bi-directional ability. After that, we propose an efficient yet effective fusion strategy in Sec. 3.3. Finally, we apply energy alignment to further boost the performance in Sec. 3.4. Additionally, we extend 3EF to Stable-3EF by allocating multiple forward prototypes and backward prototypes experimentally and theoretically. Experiments show that it helps to stabilize the training and boost performance. The detailed illustration and proof is provided in Appendix B.

3.1 PRELIMINARIES

CIL aims to learn a unifying classifier from a sequence of data divided into several incremental sessions with different class groups. At the t^{th} incremental session, the model receives the training dataset $\mathcal{D}_t = \{(\mathbf{x}_i^t, y_i^t)\}_{t=1}^n$, where $\mathbf{x}_i^t \in \mathcal{X}_t$ is an input sample and $y_i^t \in \mathcal{Y}_t$ is the corresponding label,

which is not accessible at latter sessions. Only a small amount of exemplars of previous categories are retained in a size-limited exemplar-set $\mathcal{V}_t \subseteq \cup_{i=1}^{t-1}\mathcal{D}_i$. The model is expected to train on $\mathcal{D}_t \cup \mathcal{V}_t$ and be evaluated on the test set of all known categories. At the following discussions, we focus on the details of the t^{th} incremental session without loss of generality. Particularly, we denote the label spaces of all known classes and novel classes as $\mathcal{Y}_o = \cup_{i=1}^{t-1}\mathcal{Y}_i$ and $\mathcal{Y}_n = \mathcal{Y}_t$, respectively. $|\mathcal{Y}_n| = K$ and $|\mathcal{Y}_o| = M$, representing the number of new categories and that of old ones.

3.2 Energy-basesd Model Expansion

Let $h_\theta : \mathcal{X} \longrightarrow \Delta^{K+1}$ be the newly created module, where $\mathcal{X} = \cup_{i=1}^{t}\mathcal{X}_i$ and Δ^{K+1} is a $K+1$-standard simplex. Therefore, we can further decompose h_θ as $\mathcal{S} \circ \mathcal{F} \circ \Phi$, where $\Phi : \mathcal{X} \longrightarrow \mathbb{R}^d$ is the non-linear feature extractor, $\mathcal{F} : \mathbb{R}^d \longrightarrow \mathbb{R}^{K+2}$ is a linear classifier transforming the feature into the $K+2$ - dimensional logits, and $\mathcal{S}$ denotes the non-linear activation function softmax which constrains the final output on the $K+1$- standard simplex. $h_\theta(\mathbf{x})[k]$ represents the $k+1^{th}$ element of the final output.

First, given an input-label pair $(\mathbf{x}, y) \in \mathcal{X} \times (\mathcal{Y}_o \cup \mathcal{Y}_n)$, we define the energy $E_\theta(\mathbf{x}, y)$ as

$$E_\theta(\mathbf{x}, y) = \begin{cases} -\log h_\theta(\mathbf{x})[\sigma(y)], & y \in \mathcal{Y}_n \\ -\log\left(h_\theta(\mathbf{x})[0]/M\right), & y \in \mathcal{Y}_o \end{cases}, \tag{1}$$

where $\sigma : \mathcal{Y}_n \longrightarrow 1, 2, \ldots, K$ is a bijection function mapping a given label to its corresponding class index. The energy $E(\mathbf{x}, y)$ measures the uncertainty of predicting $\mathbf{x}$'s label as y. Hence, the definition of the energy is compatible with traditional classification definitions, since we typically use $h_\theta(\mathbf{x})[\sigma(y)]$, which is the negative exponent of the energy, to indicate the confidence of predicting $\mathbf{x}$'s label as y. Moreover, we use $h_\theta(\mathbf{x})[0]$ to represent the overall confidence of $\mathbf{x}$'s label belonging to $\mathcal{Y}_o$ and do not expect the new module to distinguish between the old categories. Hence, $E(\mathbf{x}, y)$ for any $y \in \mathcal{Y}_o$ is represented as $-\log\left(h_\theta(\mathbf{x})[0]/M\right)$. The denominator M makes the energy larger and indicates that the current module has a larger uncertainty about old categories due to the limited supervision for old categories from $\mathcal{V}_t$.

Since $\mathbb{P}_\theta(y|\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x},y))}{\sum_{y'}\exp(-E_\theta(\mathbf{x},y'))}$ and $\mathbb{P}_\theta(\mathbf{x}) = \frac{\sum_{y'}\exp(-E_\theta(\mathbf{x},y'))}{\sum_{\mathbf{x}'}\sum_{y'}\exp(-E_\theta(\mathbf{x}',y'))}$, the conditional probability density and marginal probability density can be formulated as

$$\mathbb{P}_\theta(y|\mathbf{x}) = \begin{cases} \frac{h_\theta(\mathbf{x})[\sigma(y)]}{\sum_{k=0}^{K} h_\theta(\mathbf{x})[k]}, & y \in \mathcal{Y}_n \\ \frac{h_\theta(\mathbf{x})[0]}{M\sum_{k=0}^{K} h_\theta(\mathbf{x})[k]}, & y \in \mathcal{Y}_o \end{cases}, \qquad \mathbb{P}_\theta(\mathbf{x}) = \frac{\sum_{k=0}^{K} h_\theta(\mathbf{x})[k]}{\sum_{\mathbf{x}'}\sum_{k=0}^{K} h_\theta(\mathbf{x}')[k]}. \tag{2}$$

Note that $\mathbb{P}_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{\sum_{\mathbf{x}'}\exp(-E_\theta(\mathbf{x}'))}$, then we get that the energy function in $\mathbb{P}_\theta(\mathbf{x})$ is formulated as

$$E_\theta(\mathbf{x}) = -\log\sum_{k=0}^{K} h_\theta(\mathbf{x})[k]\,. \tag{3}$$

With energy functions defined above, we give proof of the derivation of our optimization objective when training a new module and demonstrate how it works to achieve bi-directional compatibility. Instead of simply learning a discriminator $\mathbb{P}_\theta(y|\mathbf{x})$, which usually causes overconfident predicts even when receiving samples from unseen distributions, we estimate the joint distribution $\mathbb{P}_\theta(\mathbf{x}, y)$

$$\arg\min_\theta \quad \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log\mathbb{P}_\theta(\mathbf{x}, y)\right] \tag{4}$$

$$= \arg\min_\theta \quad \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log\mathbb{P}_\theta(\mathbf{x})\right] + \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log\mathbb{P}_\theta(y|\mathbf{x})\right]. \tag{5}$$

The estimation of the joint distribution not only encourages the model to learn to distinguish all known categories but also helps model the input distribution, thus making the model sensitive to the input distribution drift. Therefore, when unseen categories emerge, it helps those modules alleviate overconfident predictions and reduce their impact on the ultimate predictions. However, due to the intractability of the normalizing constant $\sum_{\mathbf{x}'}\sum_{y'}\exp(-E_\theta(\mathbf{x}', y'))$, we optimize the gradient equivalent objective of Eq. 4.

Theorem 3.1 (Marginal Distribution Maximum Likelihood Estimation)**.** *Defining* $E'_\theta(\mathbf{x}) = -\log h_\theta(\mathbf{x})[K+1]$ *and its corresponding marginal distribution as* $\mathbb{P}'_\theta(\mathbf{x})$*, the optimization of* $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log\mathbb{P}_\theta(\mathbf{x})\right]$ *is equivalent to that of* $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log\sum_{k=0}^{K} h_\theta(\mathbf{x})[k]\right] +$

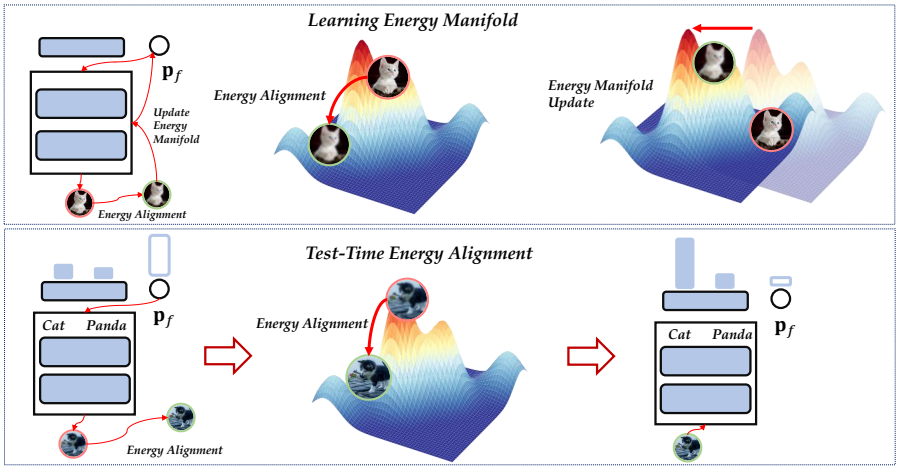


Figure 2: **Learning energy manifold and test-time energy alignment.** Upper: the learning process of the energy manifold. Lower: test-time energy alignment through the learned energy manifold.

$\lambda_{\bar{\theta}}\mathbb{E}_{\mathbb{P}'_{\bar{\theta}}(\mathbf{x})}\left[-\log h_\theta(\mathbf{x})[K+1]\right]$ *when gradient descend is applied, where* $\lambda_{\bar{\theta}}$ *is the ratio of the normalizing constants determined by* $E'_\theta(\mathbf{x})$ *and* $E_\theta(\mathbf{x})$*, and* $\bar{\theta}$ *means that parameters of* θ *is frozen* (i.e., *instances sampled from* $\mathbb{P}'_{\bar{\theta}}(\mathbf{x})$ *are detached).*

Theorem 3.2 (Conditional Distribution Maximum Likelihood Estimation)**.** *With preliminaries from Thm. 3.1, the optimization of* $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log \mathbb{P}_\theta(y|\mathbf{x})\right]$ *is equivalent to that of* $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log h_\theta(\mathbf{x})[\sigma'(y)]\right] + \mu_{\bar{\theta}}\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log h_\theta(\mathbf{x})[K+1]\right]$ *when gradient descend is applied, where* $\mu_{\bar{\theta}} = \frac{h_{\bar{\theta}}(\mathbf{x})[K+1]}{\sum_{k=0}^{K} h_{\bar{\theta}}(\mathbf{x})[k]}$, $\sigma'(y) = \begin{cases}\sigma(y), & y \in \mathcal{Y}_n \\ 0, & y \in \mathcal{Y}_o\end{cases}$.

Due to the space limit, detailed proofs for Thm. 3.1 and Thm. 3.2 are deferred to Appendix A. Combining Thm. 3.1 and Thm. 3.2, the ultimate optimization objective can be formulated as

$$\begin{aligned}\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log\sum\nolimits_{k=0}^{K} h_\theta(\mathbf{x})[k]\right] + \lambda_{\bar{\theta}}\mathbb{E}_{\mathbb{P}'_{\bar{\theta}}(\mathbf{x})}\left[-\log h_\theta(\mathbf{x})[K+1]\right] + \\ \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log h_\theta(\mathbf{x})[\sigma'(y)]\right] + \mu_{\bar{\theta}}\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log h_\theta(\mathbf{x})[K+1]\right],\end{aligned} \tag{6}$$

which is upper bounded by

$$\begin{aligned}&2\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log h_\theta(\mathbf{x})[\sigma'(y)]\right] + \mu_{\bar{\theta}}\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log h_\theta(\mathbf{x})[K+1]\right] + \\ \text{(Objective)}\qquad &\lambda_{\bar{\theta}}\mathbb{E}_{\mathbb{P}'_{\bar{\theta}}(\mathbf{x})}\left[-\log h_\theta(\mathbf{x})[K+1]\right].\end{aligned} \tag{7}$$

We take Eq. 7 as the eventual training objective. Here, we explain the roles of different components. $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log h_\theta(\mathbf{x})[\sigma'(y)]\right]$ prompts the new module to accurately discriminate all categories from current task as well as build explicit decision boundaries between current task and prior ones. By setting $\mathbf{p}_b$ as the special prototype for all old categories, we can better exploit the shared structure of all old categories and reduce the risk of over-fitting, which typically results from inadequate training samples on old categories. Given a sample from prior tasks, the new module perceives this task boundary and reduces the confidence of its own task, so that the old module dominates the ultimate prediction. Hence, we achieve better backward compatibility for old categories than naively tuning the new module on all categories. $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log h_\theta(\mathbf{x})[K+1]\right]$ encourages the module to reserve a certain degree of confidence for virtual class $\mathbf{p}_f$, thus measuring the out-of-distribution uncertainty for given samples and mitigating overconfident predictions. As shown in Fig. 2, $\mathbb{E}_{\mathbb{P}'_{\bar{\theta}}(\mathbf{x})}\left[-\log h_\theta(\mathbf{x})[K+1]\right]$ introduces an adversarial learning process, where we iteratively generate instances believed to have lower energies from $\mathbb{P}'_\theta(\mathbf{x})$ and then update the energy manifold to increase the energy of generated samples and decrease that of real samples. This process effectively enhances the modeling of the known input distribution, making in-distribution samples have low energy and out-of-distribution data have high energy. Therefore, for a sample from unseen distributions, the module will produce predictions with large uncertainty ($h_\theta(\mathbf{x})[K+1]$) and low confidence due to the fact that the confidence must be lower than $1 - h_\theta(\mathbf{x})[K+1]$. Then, modules created in the future to handle these unknown distributions will dominate the final prediction. Hence, we achieve the forward compatibility for the future unseen categories.

3.3 Energy-based Model Fusion

After training the new module, we aim to fuse it with the prior ones to form a unifying classifier for all seen categories. Assuming that we have trained a unifying model h_{θ_o} for all the old tasks and σ_o maps the label to the output index of h_{θ_o}, a vanilla approach to combining the h_{θ_o} and h_θ is to redefine the energy function as

$$E_{\{\theta_o,\theta\}}(\mathbf{x},y)=\begin{cases}-\log h_{\theta_o}(\mathbf{x})[\sigma_o(y)], & y\in\mathcal{Y}_o\\ -\log h_\theta(\mathbf{x})[\sigma(y)], & y\in\mathcal{Y}_n\end{cases}. \quad (8)$$

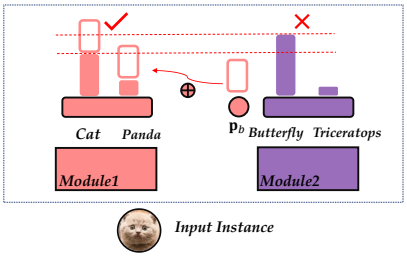


Figure 3: $\mathbf{p}_b$ acts as a soft task-discriminator. It alleviates the task-bias and determines the dominant module for the ultimate prediction.

Then we have

$$\mathbb{P}_{\{\theta,\theta_o\}}(y|\mathbf{x})=\begin{cases}\frac{h_{\theta_o}(\mathbf{x})[\sigma_o(y)]}{\sum_{m=1}^{M}h_{\theta_o}(\mathbf{x})[m]+\sum_{k=1}^{K}h_\theta(\mathbf{x})[k]}, & y\in\mathcal{Y}_o\\ \frac{h_\theta(\mathbf{x})[\sigma(y)]}{\sum_{m=1}^{M}h_{\theta_o}(\mathbf{x})[m]+\sum_{k=1}^{K}h_\theta(\mathbf{x})[k]}, & y\in\mathcal{Y}_n\end{cases}. \quad (9)$$

However, this might cause task bias. Different modules may produce predictions with different entropies, the combined model has a tendency to modules with larger entropies. As shown in Fig. 3, Simply combing the modules as Eq. 9 leads to misclassification due to the larger entropy in module2. Considering that $\mathbf{p}_b$ measures the confidence for old categories, we redefine $E_{\{\theta_o,\theta\}}$ as

$$\begin{cases}-\log\left\{h_{\theta_o}(\mathbf{x})[\sigma_o(y)]+\alpha h_\theta(\mathbf{x})[0]+\beta\right\}, & y\in\mathcal{Y}_o\\ -\log h_\theta(\mathbf{x})[\sigma(y)], & y\in\mathcal{Y}_n\end{cases}. \quad (10)$$

Then we have

$$\mathbb{P}_{\{\theta,\theta_o\}}(y|\mathbf{x})=\begin{cases}\frac{h_{\theta_o}(\mathbf{x})[\sigma(y)]+\alpha h_\theta(\mathbf{x})[0]+\beta}{\sum_{m=1}^{M}[h_{\theta_o}(\mathbf{x})[m]+\alpha h_\theta(\mathbf{x})[0]+\beta]+\sum_{k=1}^{K}h_\theta(\mathbf{x})[k]}, & y\in\mathcal{Y}_o\\ \frac{h_\theta(\mathbf{x})[\sigma(y)]}{\sum_{m=1}^{M}[h_{\theta_o}(\mathbf{x})[m]+\alpha h_\theta(\mathbf{x})[0]+\beta]+\sum_{k=1}^{K}h_\theta(\mathbf{x})[k]}, & y\in\mathcal{Y}_n\end{cases}. \quad (11)$$

We finetune α,β to minimize the negative log-likelihood on a tiny sub-dataset (exemplar-set), thus mitigating the task bias, namly

$$\alpha^*,\beta^*=\arg\min_{\alpha,\beta}\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log\mathbb{P}_{\{\theta,\theta_o\}}(y|\mathbf{x})\right]. \quad (12)$$

3.4 Energy-based model alignment

Equipped with an energy-based nature, 3EF allows for a pleasant by-product: test-time algnment. During the evaluation phase, we make energy alignment to the input. Specifically, we apply SGLD (Welling & Teh, 2011) to decrease energy of given samples. That is,

$$\mathbf{x}_{i+1}=\mathbf{x}_i-\frac{\tau}{2}\nabla_{\mathbf{x}}E_\theta(\mathbf{x})+\sqrt{\tau}\omega_i,\omega_i\sim\mathcal{N}(0,\mathbf{I}), \quad (13)$$

where τ is the step size and ω is the random noise sampled from standard normal distribution. Because all samples are from known classes at test time, aligning given samples to a manifold with lower energy can make the model more suitable for testing in a closed world. As shown in Fig. 2, through energy alignment, some useful characteristics shared by training samples are transferred into the test sample to reduce its energy, making it more explicit and producing more convincing predictions.

3.5 Summary of 3EF

To conclude, 3EF provides a unifying energy-based framework for CIL. In order to achieve decoupled training and reduce training overhead, we propose a two-stage training approach: expansion and fusion. At the expansion phase, we utilize two additionally allocated special prototypes to achieve the bi-directional compatibility. Then, we propose a novel fusion strategy to mitigate the task-bias and form a unifying classifier. At the evaluation phase, by advantage of the inherently built energy-based model, we additionally apply energy alignment to further boost the performance.

3.6 3EF-Distill

Note that even 3EF has kept the training cost nearly the same for all the incremental sessions, which is a great advantage over classical dynamic-architecture-based methods. However, 3EF still suffer from the increasing number of model parameters, which is a great disadvantage for almost all dynamic-structure-based methods and might violate the memory usage limitation in class-incremental learning, and therefore it is necessary to compress the increasingly expanded modules. Note that

FOSTER (Wang et al., 2022) proposed the feature compression phase to compress the expanded dual branch model into a single skeleton, and we show that it is also possible to compress the expanded 3EF model into a single skeleton with the same strategy in FOSTER. Specifically, after each training stage, we apply the balanced knowledge distillation to compress the model, that is

$$\mathcal{L}_{\text{BKD}} = \text{KL}\left(\boldsymbol{w} \otimes \mathbb{P}_{\{\theta,\theta_o\}}(\cdot \mid \mathbf{x}) \| \mathbb{P}_{\theta}(\cdot \mid \mathbf{x})\right) , \tag{14}$$

where the $\otimes$ means the tensor product, $\boldsymbol{w}$ is the weighted vector obtained as effective number (Cui et al., 2019). We will show that even though the distillation strategy in FOSTER is not specially designed for 3EF, 3EF-Distill still achieves competitive performance.

4 Empirical Studies

4.1 Experimental Settings

Datasets. We validate our methods on widely used benchmarks of class-incremental learning CIFAR-100 (Krizhevsky et al., 2009) and ImageNet100/1000 (Deng et al., 2009). **CIFAR-100**: CIFAR-100 consists of 50,000 training images with 500 images per class, and 10,000 test images with 100 images per class. **ImageNet-1000**: ImageNet-1000 is a large scale dataset composed of about 1.28 million images for training and 50,000 for validation with 500 images per class. **ImageNet-100**: ImageNet-100 is composed of 100 classes randomly chosen from the original ImageNet-1000 dataset.

CIAFR-100 Protocol. For benchmark CIFAR-100, we evaluate two widely recognized protocols: **CIFAR-100 B0**: All the 100 classes are averagely divided into 5, 10, and 20 groups, respectively, *i.e.*, we should train all the 100 classes gradually with 20, 10, 5 classes per incremental session. In addition, models are allowed to save an exemplar-set to store no more than 2000 exemplars throughout all sessions. **CIFAR-100 B50**: We first train half of 100 classes at the base learning stage. Then, the rest 50 classes are averagely divided into 5, 10, and 25 groups, respectively, *i.e.*, we should train the rest 50 classes gradually with 10, 5, and 2 classes per incremental session. Slightly different from the first protocol, models are allowed to store no more than 20 exemplars for each class. Therefore, after training all the 100 classes, there are also no more than 2,000 exemplars.

ImageNet Protocol. For benchmarking ImageNet-100, we evaluate the performance on two different incremental tasks. In the first task, we split the 100 classes averagely into 10 sequential incremental sessions, and up to 2,000 exemplars are allowed to be stored in the exemplar-set. In the second task, models are first trained on 50 base classes and then sequentially trained on the 10 classes at the following incremental sessions (*i.e.*, 5 incremental sessions totally). The same as the protocol CIFAR-100 B50, models are allowed to store no more than 20 exemplars for each class. For benchmark ImageNet-1000, we train all the 1000 classes with 100 classes per step (10 steps in total) with an exemplar-set storing no more than 20,000 exemplars.

Compared methods. We compare 3EF to strong regularization-based methods: iCaRL (Rebuffi et al., 2017), BiC (Wu et al., 2019), WA (Zhao et al., 2020), and PodNet (Douillard et al., 2020). Beside, we compare to the dynamic-structure-based method RPSNet (Rajasegaran et al., 2019), DER (Yan et al., 2021), Dytox (Douillard et al., 2021), FOSTER (Wang et al., 2022). Apart from the above methods, we also compare with RMM (Liu et al., 2021), which adjusts the memory partition strategy for new and old data. Among the compared methods, Dytox applies stronger neural architecture (Convit (d'Ascoli et al., 2021)) and additional data augmentation; FOSTER (Wang et al., 2022) additionally uses the AutoAugmentation (Cubuk et al., 2019) to enhance the sample efficiency and improve classification accuracy. RMM (Liu et al., 2021) achieves a better memory management strategy. With the same training memory, RMM chooses some new samples to train and discards the rest to allow restoring more old exemplars. Note that all these are orthogonal to the method itself, and therefore we combine the augmentation and memory management strategy with 3EF for fair comparison with these methods, which we regard as 3EF-Distill++. We show that 3EF-Distill achieves state-of-the-art performance.

4.2 Results and Analysis

Comparison with SOTAs. CIFAR-100: Table 2 summarizes the experimental results on CIFAR-100 benchmark. Significant performance improvements can be seen under both B50 and B0 protocols. Particularly, we achieve 1.16, 2.00, and 1.35 increase of average accuracy under the protocol B0 with 5, 10, and 20 incremental sessions. As shown in Fig. 4, 3EF consistently surpass other methods at each incremental sessions. Besides, we achieve more evident performance improvements under B50 protocols. Specifically, we improve average accuracy by 3.12, 3.62, 5.45 under protocol B50

Methods	ImageNet-100 10 steps				ImageNet-1000 10 steps				ImageNet-100 B50 5 steps			
	top-1		top-5		top-1		top-5		top-1		top-5	
	Avg	Last	Avg	Last	Avg	Last	Avg	Last	Avg	Last	Avg	Last
Bound	-	81.50	-	95.10	89.27	-	79.89	-	81.20	81.50	-	95.10
Replay	59.21	41.00	81.67	68.44	-	-	-	-	55.73	43.38	79.17	71.08
iCaRL	67.11	50.98	84.08	71.52	38.4	22.7	63.7	44.0	62.56	53.69	81.75	73.58
BiC	65.13	42.40	84.04	64.14	-	-	84.0	73.2	66.36	49.9	83.59	70.42
WA	68.60	55.04	89.53	80.32	65.67	55.60	86.60	81.10	65.81	56.64	84.97	79.36
PodNet	64.03	45.40	84.06	68.58	-	-	-	-	73.84	62.94	89.51	83.52
DER	77.08	66.84	92.49	88.64	66.87	58.83	88.01	81.59	77.57	71.10	93.37	91.3
DyTox	71.85	57.94	90.72	83.52	68.14	59.75	87.03	82.93	-	-	-	-
RPSNet	-	-	87.90	74.00	-	-	-	-	-	-	-	-
RMM	-	-	-	-	-	-	-	-	79.52	-	-	-
FOSTER	78.71	70.14	-	-	68.34	58.53	-	-	80.22	75.52	-	-
3EF	**77.62**	**68.78**	**93.66**	**89.32**	**67.89**	**59.34**	**88.39**	**82.31**	77.38	**71.63**	**93.71**	**91.76**
3EF-Distill++												

Table 1: Performance on ImageNet. We report both average and last accuracy of top-1 and top-5.

Methods	CIFAR-100 B0						CIFAR-100 B50					
	5 steps		10 steps		20 steps		5 steps		10 steps		25 steps	
	Avg	Last	Avg	Last	Avg	Last	Avg	Last	Avg	Last	Avg	Last
Bound	80.40	-	80.41	-	81.49	-	79.89	-	79.91	-	80.37	-
Replay	60.63	43.08	59.38	41.01	58.20	38.69	52.70	41.26	43.43	36.16	41.09	37.50
iCaRL	67.60	54.23	64.64	49.52	63.51	45.12	61.79	52.04	52.69	44.64	52.10	45.57
BiC	67.63	56.22	65.38	50.79	62.38	43.08	61.68	49.19	57.04	43.82	53.61	40.38
WA	69.11	57.97	67.15	52.30	64.65	48.46	64.65	55.85	53.87	46.72	52.51	44.90
PodNet	63.35	49.08	56.41	37.68	47.88	27.99	64.79	55.21	63.55	53.94	60.59	49.94
DER	71.15	62.4	69.94	58.59	67.98	53.95	68.58	61.94	66.40	58.85	60.66	49.30
RPSNet	70.5	-	68.6	-	-	-	-	-	-	-	-	-
DyTox	-	-	71.50	57.76	68.86	51.47	-	-	-	-	-	-
RMM	-	-	-	-	-	-	68.86	-	67.61	-	**66.21**	-
FOSTER	72.54	64.55	72.81	**62.54**	70.65	56.28	70.10	64.01	67.94	60.44	63.83	54.31
3EF	72.31	**62.58**	71.94	60.98	69.84	56.71	**71.70**	**65.24**	**70.71**	**63.51**	**66.11**	**54.36**
3EF-Distill++	**73.05**	62.48	**72.93**	61.45	**71.69**	**57.06**	**71.58**	**64.54**	**71.70**	61.19	64.32	**54.81**

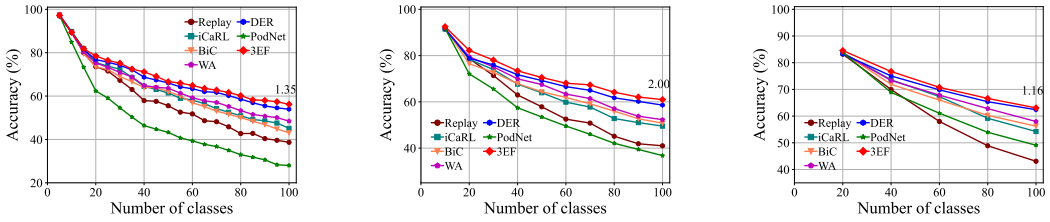
Table 2: Performance on CIFAR-100. We report both the top-1 average and last accuracy.

with 5, 10, 25 incremental sessions. This shows that the effectiveness of 3EF in both short-term and long-term incremental learning tasks. The detailed performance in Fig. 7 demonstrates that 3EF consistently outperforms other methods after each incremental session. **ImageNet-100/1000**: Tabel 1 summarizes the experimental results on both ImageNet-100 and ImageNet-1000 benchmarks. It is observed that both the top-1 and top-5 accuracy of 3EF still surpass other methods in most protocols, indicating the robustness of 3EF. The detailed performance on ImageNet-100 are shown in Fig. E.1 We achieves 1.96 and 0.53 increase of top-1 last accuracy under ImageNet-100 B0 and B50 protocols with incremental step 10. 3EF improves the top-1 and top-5 last accuracy by 0.51 and 0.72 on ImageNet-1000 with 10 incremental sessions, demonstrating that the efficacy of 3EF under large-scale incremental tasks.

Comparison under imbalanced exemplar-set. Fig. 4 displays the average accuracy changes of different methods after each incremental session on CIFAR-100 B50 with 5 steps. Fig. 5(c) illustrates the performance when exemplars are randomly sampled from all the available old instances. Though the exemplar-set is statistically balanced, prior methods encounters a performance drop and the gap between 3EF and prior methods is enlarged. Fig. 5(b) and Fig. 5(a) illustrate the performance changes under extreme class imbalance and sample from half classes missing, respectively. Although the performance of prior methods declines dramatically, 3EF maintains its effectiveness under these two imbalanced protocols. 3EF achieves more than 10% performance gain under these challenging settings. In addition, since some classes have no exemplars stored in the exemplar-set to calculate the class center, iCaRL based on NCM-classifier (Mensink et al., 2013) fails in the base training phase.

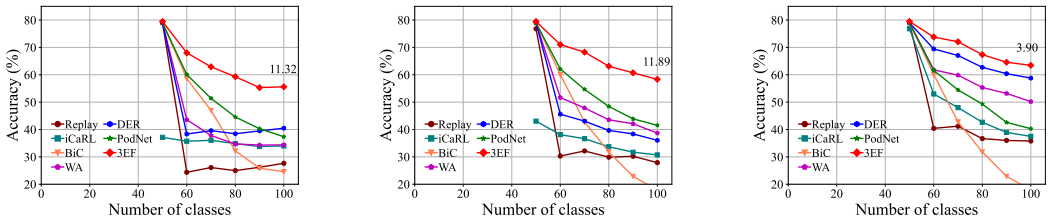
4.3 Ablation Studies

Ablations of key components in 3EF. To verify the effectiveness of the components in 3EF, we conduct ablation studies on CIFAR-100 B50 with 5 incremental sessions. As shown in Table 3, the average and last accuracy gradually increase as we add more components. It is notable that after the fusion strategy, the unifying model get a large increase. Besides, by learning energy manifolds with the forward prototype $\mathbf{p}_f$ and energy alignment, we can further improve the performance.



(a) CIFAR-100 B0 20 steps (b) CIFAR-100 B0 10 steps (c) CIFAR-100 B0 5 steps

Figure 4: Performance on standard CIFAR-100 B0 protocols.



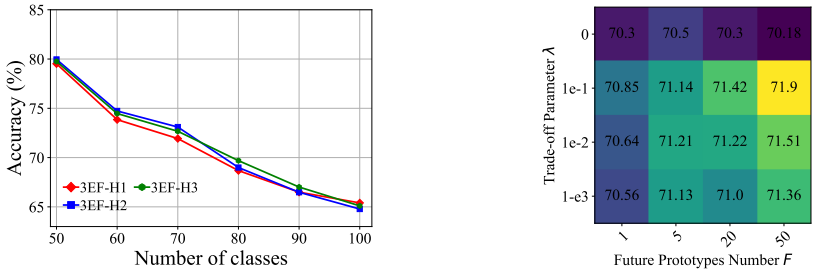
(a) exp: $\gamma = 0.8$ (b) half-half: $\gamma = 1$ (c) random

Figure 5: Performance on imbalanced protocols.

Expansion		Fusion			
backward compatible	forward compatible	task discriminator	energy alignment	**Avg**	**Last**
✓				63.60	55.16
✓	✓			64.50	56.50
✓		✓		70.51	64.43
✓	✓	✓		70.92	64.79
✓	✓	✓	✓	71.75	65.24

Table 3: Ablations of key components in 3EF. We report the average and last accuracy on CIFAR-100 B50 with 5 incremental sessions.

Sensitive study of hyper-parameters. There are three hyper-parameters in 3EF when modeling the energy manifold: hidden layer where energy modeling, the trade-off coefficient λ, and the number of forward prototypes F in Stable-3EF. As shown in Fig. 6(a) we conduct experiments on three different hidden layers for modeling the energy manifold, and the experimental results show 3EF the robustness to the choice of different hidden layer. We also change the trade-off coefficients λ from $\{0, 1e-1, 1e-2, 1e-3\}$ and the number of forward prototypes F from $\{1, 5, 20, 50\}$, and the average accuracies on CIFAR-100 B50 with 5 incremental sessions are shown in Fig. 6(b). We can see that with the increase of F and λ, the accuracy has an upward trend.



(a) Influence of hidden layers (b) Impact of λ, F

Figure 6: Sensitive studies of hyper-parameters.

CONCLUSION. In this work we presented 3EF for achieving efficient class-incremental learning. Under this framework, we efficiently train a specific module for the current task while achieving bi-directional compatibility and fuse it with the prior model under minimal effort. 3EF is equipped with a theoretical analysis showing that its training process is inherently the modeling of an energy-based model. This energy-based nature enables test-time alignment to further improve the performance.

REFERENCES

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pp. 139–154, 2018.

Haoyue Bai, Rui Sun, Lanqing Hong, Fengwei Zhou, Nanyang Ye, Han-Jia Ye, S-H Gary Chan, and Zhenguo Li. Decaug: Out-of-distribution generalization via decomposed feature representation and semantic augmentation. In *AAAI*, volume 35, pp. 6705–6713, 2021.

Yatao Bian, Yu Rong, Tingyang Xu, Jiaxiang Wu, Andreas Krause, and Junzhou Huang. Energy-based learning for cooperative games, with applications to valuation problems in machine learning. In *ICLR*, 2022. URL https://openreview.net/forum?id=xLfAgCroImw.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, pp. 113–123, 2019.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, pp. 9268–9277, 2019.

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

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

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.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, pp. 86–102. Springer, 2020.

Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. *arXiv preprint arXiv:2111.11326*, 2021.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *NeurIPS*, 32, 2019.

Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pp. 2286–2296. PMLR, 2021.

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

Ovidiu Gheorghioiu, Alexandru Salcianu, and Martin Rinard. Interprocedural compatibility analysis for static object preallocation. In *POPL*, pp. 273–284, 2003.

Lukasz Golab and M Tamer Özsu. Issues in data stream management. *ACM Sigmod Record*, 32(2): 5–14, 2003.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Zhanglong Ji, Zachary C Lipton, and Charles Elkan. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*, 2014.

KJ Joseph, Salman Khan, Fahad Shahbaz Khan, Rao Muhammad Anwer, and Vineeth N Balasubramanian. Energy-based latent aligner for incremental learning. In *CVPR*, pp. 7452–7461, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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. *PNAS*, 114(13):3521–3526, 2017.

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

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

Kyungmin Lee, Hunmin Yang, and Se-Yoon Oh. Adversarial training on joint energy based model for robust classification and out-of-distribution detection. In *ICCAS*, pp. 17–21. IEEE, 2020.

Shuang Li, Yilun Du, Gido M van de Ven, and Igor Mordatch. Energy-based models for continual learning. *arXiv preprint arXiv:2011.12216*, 2020.

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

Zhuoyun Li, Changhong Zhong, Sijia Liu, Ruixuan Wang, and Wei-Shi Zheng. Preserving earlier knowledge in continual learning with the help of all previous feature extractors. *arXiv preprint arXiv:2104.13614*, 2021.

Ziqian Lin, Sreya Dutta Roy, and Yixuan Li. Mood: Multi-level out-of-distribution detection. In *CVPR*, pp. 15313–15323, 2021.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *NeurIPS*, 33:21464–21475, 2020.

Yaoyao Liu, Bernt Schiele, and Qianru Sun. Rmm: Reinforced memory management for class-incremental learning. *NeurIPs*, 34:3478–3490, 2021.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *arXiv preprint arXiv:2010.15277*, 2020a.

Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *arXiv preprint arXiv:2010.15277*, 2020b.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. *NeurIPS*, 12, 1999.

Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2624–2637, 2013.

Jack Minker. On indefinite databases and the closed world assumption. In *CADE*, pp. 292–308. Springer, 1982.

Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. *NeurIPs*, 32, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, Ling Shao, and Ming-Hsuan Yang. An adaptive random path selection approach for incremental learning. *arXiv preprint arXiv:1906.01120*, 2019.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pp. 2001–2010, 2017.

Yantao Shen, Yuanjun Xiong, Wei Xia, and Stefano Soatto. Towards backward-compatible representation learning. In *CVPR*, pp. 6368–6377, 2020.

Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *ICCV*, pp. 32–42, 2021.

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

Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. *arXiv preprint arXiv:2204.04662*, 2022.

Yezhen Wang, Bo Li, Tong Che, Kaiyang Zhou, Ziwei Liu, and Dongsheng Li. Energy-based open-world uncertainty modeling for confidence calibration. In *ICCV*, pp. 9302–9311, 2021.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, pp. 681–688. Citeseer, 2011.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pp. 374–382, 2019.

Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. VAEBM: A symbiosis between variational autoencoders and energy-based models. In *ICLR*, 2021. URL https://openreview.net/forum?id=5m3SEczOV8L.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *ICML*, pp. 2635–2644. PMLR, 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.

Jianwen Xie, Yaxuan Zhu, Jun Li, and Ping Li. A tale of two flows: Cooperative learning of langevin flow and normalizing flow toward energy-based model. In *ICLR*. OpenReview.net, 2022. URL https://openreview.net/forum?id=31d5RLCUuXC.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, pp. 3014–3023, 2021.

Shuangfei Zhai, Yu Cheng, Rogerio Feris, and Zhongfei Zhang. Generative adversarial networks as variational training of energy based models. *arXiv preprint arXiv:1611.01799*, 2016.

Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *CVPR*, pp. 13208–13217, 2020.

Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

Yang Zhao, Jianwen Xie, and Ping Li. Learning energy-based generative models via coarse-to-fine expanding and sampling. In *ICLR*. OpenReview.net, 2021. URL https://openreview.net/forum?id=aD1_5zowqV.

Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. Forward compatible few-shot class-incremental learning. In *CVPR*, pp. 9046–9056, 2022.

Appendix of 3EF

CONTENTS

A PROOF FOR THM. 3.1 AND THM. 3.2

Here we provide the detailed proofs for Thm. 3.1 and Thm. 3.2.

Theorem 3.1 (Marginal Distribution Maximum Likelihood Estimation)**.** *Defining* $E'_\theta(\mathbf{x}) = -\log h_\theta(\mathbf{x})[K+1]$ *and its corresponding marginal distribution as* $\mathbb{P}'_\theta(\mathbf{x})$*, the optimization of* $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \mathbb{P}_\theta(\mathbf{x})\right]$ *is equivalent to that of* $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \sum_{k=0}^{K} h_\theta(\mathbf{x})[k]\right] + \lambda_{\bar{\theta}}\mathbb{E}_{\mathbb{P}'_{\bar{\theta}}(\mathbf{x})}\left[-\log h_\theta(\mathbf{x})[K+1]\right]$ *when gradient descend is applied, where* $\lambda_{\bar{\theta}}$ *is the ratio of the normalizing constants determined by* $E'_\theta(\mathbf{x})$ *and* $E_\theta(\mathbf{x})$*, and* $\bar{\theta}$ *means that parameters of* θ *is frozen (*i.e.*, instances sampled from* $\mathbb{P}'_{\bar{\theta}}(\mathbf{x})$ *are detached).*

Proof. Since $\mathbb{P}_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{\sum_{\mathbf{x}'}\exp(-E_\theta(\mathbf{x}))}$, we have

$$\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \mathbb{P}_\theta(\mathbf{x})\right] = \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[E_\theta(\mathbf{x})\right] + \log \sum_{\mathbf{x}'} \exp(-E_\theta(\mathbf{x}'))\,. \tag{15}$$

Take the gradient of the right part in Eq. 15, and we have

$$
\begin{aligned}
&\nabla_\theta \log \sum_{\mathbf{x}'} \exp(-E_\theta(\mathbf{x}')) \\
=&\sum_{\mathbf{x}} \frac{\exp(-E_\theta(\mathbf{x}))}{\sum_{\mathbf{x}'} \exp(-E_\theta(\mathbf{x}'))} \nabla_\theta E_\theta(\mathbf{x}) \\
=&\sum_{\mathbf{x}} \frac{\sum_{k=0}^{K} h_\theta(\mathbf{x})[k]}{\sum_{\mathbf{x}'} \exp(-E_\theta(\mathbf{x}'))} \cdot \frac{\nabla_\theta \sum_{k=0}^{K} h_\theta(\mathbf{x})[k]}{\sum_{k=0}^{K} h_\theta(\mathbf{x})[k]} \\
=&\sum_{\mathbf{x}} \frac{\nabla_\theta (1 - h_\theta(\mathbf{x})[K+1])}{\sum_{\mathbf{x}'} \exp(-E_\theta(\mathbf{x}'))} \\
=&-\sum_{\mathbf{x}} \frac{\nabla_\theta h_\theta(\mathbf{x})[K+1]}{\sum_{\mathbf{x}'} \exp(-E_\theta(\mathbf{x}'))} \\
=&-\sum_{\mathbf{x}} \frac{\sum_{\mathbf{x}'} \exp(-E'_\theta(\mathbf{x}'))}{\sum_{\mathbf{x}'} \exp(-E_\theta(\mathbf{x}'))} \cdot \frac{\exp(-E'_\theta(\mathbf{x}))}{\sum_{\mathbf{x}'} \exp(-E'_\theta(\mathbf{x}'))} \cdot \frac{\nabla h_\theta(\mathbf{x})[K+1]}{\exp(-E'_\theta(\mathbf{x}))} \\
=&-\sum_{\mathbf{x}} \frac{\sum_{\mathbf{x}'} \exp(-E'_\theta(\mathbf{x}'))}{\sum_{\mathbf{x}'} \exp(-E_\theta(\mathbf{x}'))} \cdot \frac{\exp(-E'_\theta(\mathbf{x}))}{\sum_{\mathbf{x}'} \exp(-E'_\theta(\mathbf{x}'))} \cdot \frac{\nabla h_\theta(\mathbf{x})[K+1]}{h_\theta(\mathbf{x})[K+1]} \\
=&-\sum_{\mathbf{x}} \frac{Z'_\theta}{Z_\theta} \cdot \frac{\exp(-E'_\theta(\mathbf{x}))}{\sum_{\mathbf{x}'} \exp(-E'_\theta(\mathbf{x}'))} \cdot \nabla_\theta \log h_\theta(\mathbf{x})[K+1] \\
=&-\frac{Z'_\theta}{Z_\theta} \mathbb{E}_{\mathbb{P}'_\theta(\mathbf{x})} \left[\nabla_\theta \log h_\theta(\mathbf{x})[K+1]\right] \\
=&-\nabla_\theta \left\{ \frac{Z'_{\bar{\theta}}}{Z_{\bar{\theta}}} \mathbb{E}_{\mathbb{P}'_{\bar{\theta}}(\mathbf{x})} \left[\log h_\theta(\mathbf{x})[K+1]\right] \right\}.
\end{aligned}
\tag{16}
$$

where $\bar{\theta}$ represents that θ is frozen (*i.e.*, gradients are not computed), $Z'_{\bar{\theta}} = \sum_{\mathbf{x}'} \exp(-E'_\theta(\mathbf{x}'))$ is the normalizing constant for energy $E'_\theta(\mathbf{x})$, and Z_θ is the normalizing constant for energy $E_\theta(\mathbf{x})$. Hence, the objective $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})} [-\log \mathbb{P}_\theta(\mathbf{x})]$ is equivalent to

$$
\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})} \left[-\log \sum_{k=0}^{K} h_\theta(\mathbf{x})[k] \right] + \lambda_{\bar{\theta}} \mathbb{E}_{\mathbb{P}_{\bar{\theta}}(\mathbf{x}')} \left[-\log h_\theta(\mathbf{x})[K+1] \right], \tag{17}
$$

where $\lambda_{\bar{\theta}} = \frac{Z'_{\bar{\theta}}}{Z_{\bar{\theta}}}$. □

Theorem 3.2 (Conditional Distribution Maximum Likelihood Estimation)**.** *With preliminaries from Thm. 3.1, the optimization of* $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)} [-\log \mathbb{P}_\theta(y|\mathbf{x})]$ *is equivalent to that of* $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)} [-\log h_\theta(\mathbf{x})[\sigma'(y)]] + \mu_{\bar{\theta}} \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})} [-\log h_\theta(\mathbf{x})[K+1]]$ *when gradient descend is applied, where* $\mu_{\bar{\theta}} = \frac{h_{\bar{\theta}}(\mathbf{x})[K+1]}{\sum_{k=0}^{K} h_{\bar{\theta}}(\mathbf{x})[k]}$, $\sigma'(y) = \begin{cases} \sigma(y), & y \in \mathcal{Y}_n \\ 0, & y \in \mathcal{Y}_o \end{cases}$.

Proof. Considering the definition of $\mathbb{P}_\theta(y|\mathbf{x})$ in Eq. 2, we have

$$
\begin{aligned}
&\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)} [-\log \mathbb{P}_\theta(y \mid \mathbf{x})] \\
=&\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)} \left[E_\theta(\mathbf{x}, y) + \log \sum_{y' \in \mathcal{Y}_n \cup \mathcal{Y}_o} \exp(-E_\theta(\mathbf{x}, y')) \right] \\
=&\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y) \cap y \in \mathcal{Y}_o} \left[-\log \frac{h_\theta(\mathbf{x})[0]}{M} \right] + \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y) \cap y \in \mathcal{Y}_n} [-\log h_\theta(\mathbf{x})[\sigma(y)]] + \\
&\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})} \left[\log \sum_{k=0}^{K} h_\theta(\mathbf{x})[k] \right]
\end{aligned}
\tag{18}
$$

The gradient of $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)\cap y\in\mathcal{Y}_o}\left[-\log\frac{h_\theta(\mathbf{x})[0]}{M}\right]$ with respect to θ is equal to that of $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)\cap y\in\mathcal{Y}_o}\left[-\log h_\theta(\mathbf{x})[0]\right]$. Therefore, define $\sigma'(y)=\begin{cases}\sigma(y), & y\in\mathcal{Y}_n\\ 0, & y\in\mathcal{Y}_o\end{cases}$, and the first two components in Eq. 18 can be written in a unifying form, that is

$$\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)\cap y\in\mathcal{Y}_o\cup\mathcal{Y}_n}\left[-\log h_\theta(\mathbf{x})[\sigma'(y)]\right]. \tag{19}$$

For the last component in Eq. 18, we have

$$\begin{aligned}
&\nabla_\theta\log\sum_{k=0}^{K}h_\theta(\mathbf{x})[k]\\
=&\frac{\nabla_\theta(1-h_\theta(\mathbf{x})[K+1])}{\sum_{k=0}^{K}h_\theta(\mathbf{x})[k]}\\
=&-\frac{\nabla_\theta h_\theta(\mathbf{x})[K+1]}{\sum_{k=0}^{K}h_\theta(\mathbf{x})[k]}\\
=&-\frac{h_\theta(\mathbf{x})[K+1]}{\sum_{k=0}^{K}h_\theta(\mathbf{x})[k]}\cdot\frac{\nabla_\theta h_\theta(\mathbf{x})[K+1]}{h_\theta(\mathbf{x})[K+1]}\\
=&-\frac{h_\theta(\mathbf{x})[K+1]}{\sum_{k=0}^{K}h_\theta(\mathbf{x})[k]}\cdot\nabla_\theta\log h_\theta(\mathbf{x})[K+1]\\
=&\nabla_\theta\left\{-\frac{h_{\bar\theta}(\mathbf{x})[K+1]}{\sum_{k=0}^{K}h_{\bar\theta}(\mathbf{x})[k]}\cdot\log h_\theta(\mathbf{x})[K+1]\right\}
\end{aligned} \tag{20}$$

Therefore, the Objective $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log\mathbb{P}_\theta(y\mid\mathbf{x})\right]$ is also equivalent to

$$\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log h_\theta(\mathbf{x})[\sigma'(y)]\right]+\mu_{\bar\theta}\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log h_\theta(\mathbf{x})[K+1]\right], \tag{21}$$

where $\mu_{\bar\theta}=\frac{h_{\bar\theta}(\mathbf{x})[K+1]}{\sum_{k=0}^{K}h_{\bar\theta}(\mathbf{x})[k]}$. □

B PROOFS FOR STABLE-3EF

As we have claimed in Sec. 3 and Sec. 4, we expand the original 3EF to have multiple backward and forward prototypes (dubbed as Stable-3EF), which stabilizes the training process and improves the performance. Here we give a formal illustration of the expandable form and provide proof of it.

First, at the t^{th} incremental session, instead of creating one backward prototype $\mathbf{p}_b$ to measure the confidence for all the old tasks, we create a backward prototype for each old task. This expansion prompts the new module to learn a discriminator for each old task, thereby improving the performance and stability of training especially when there are clear domain shifts among old tasks. Therefore, there are $t-1$ backward prototypes for measuring the confidence of old tasks. Furthermore, in Stable-3EF, we create multiple forward prototypes during the training. Concretely, for samples generated through the marginal distribution defined by the energy function, we assign the most-likely pseudo labels to them. We set the number of forward prototypes to a constant number F. After introducing these multiple prototypes, *i.e.*, $\mathbf{p}_b$s and $\mathbf{p}_f$s, we should redefine $h_\theta:\mathcal{X}\longrightarrow\Delta^{K+F+(t-2)}$.

Given an input-label pair $(\mathbf{x},y)\in\cup_{i=1}^{t}\mathcal{X}_i\times\cup_{i=1}^{t}\mathcal{Y}_i$, we define the energy $E_\theta(\mathbf{x},y)$ as

$$E_\theta(\mathbf{x},y)=\begin{cases}-\log h_\theta(\mathbf{x})[\sigma(y)], & y\in\mathcal{Y}_t\\ -\log\left(h_\theta(\mathbf{x})[\sigma(y)]/|\mathcal{Y}_i|\right), & y\in\mathcal{Y}_i,\quad i=1,2,\ldots,t-1\end{cases}, \tag{22}$$

where $\sigma:\cup_{i=1}^{t}\mathcal{Y}_i\longrightarrow K+F+(t-1)$ maps new labels to their K class-corresponding prototypes and maps old labels to their $t-1$ task-corresponding backward prototypes.

Therefore, similar to Eq. 2, the conditional probability density and marginal probability density for Stable-3EF can be re-formulated as :

$$\mathbb{P}_\theta(y|\mathbf{x})=\begin{cases}\frac{h_\theta(\mathbf{x})[\sigma(y)]}{\sum_{k=0}^{K+t-2}h_\theta(\mathbf{x})[k]}, & y\in\mathcal{Y}_t\\ \frac{h_\theta(\mathbf{x})[\sigma(y)]}{|\mathcal{Y}_i|\sum_{k=0}^{K+t-2}h_\theta(\mathbf{x})[k]}, & y\in\mathcal{Y}_i, i=1,\ldots,t-1\end{cases},\quad \mathbb{P}_\theta(\mathbf{x})=\frac{\sum_{k=0}^{K+t-2}h_\theta(\mathbf{x})[k]}{\sum_{\mathbf{x}'}\sum_{k=0}^{K+t-2}h_\theta(\mathbf{x}')[k]}. \tag{23}$$

And then we can induce that the energy function in $\mathbf{P}_\theta(\mathbf{x})$ is formulated as

$$E_\theta(\mathbf{x}) = -\log \sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]\,. \tag{24}$$

Like Eq. 4, Stable-3EF also estimates the joint distribution $\mathbb{P}_\theta(\mathbf{x}, y)$, that is

$$\arg\min_\theta \quad \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \mathbb{P}_\theta(\mathbf{x})\right] + \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log \mathbb{P}_\theta(y|\mathbf{x})\right]\,. \tag{25}$$

And same to the original 3EF, Stable-3EF also aims to find its gradient equivalent optimization objective to avoid the intractable normalizing constant in the joint distribution defined by the energy $\mathbf{E}_\theta(\mathbf{x}, y)$.

Theorem B.1 (Marginal Distribution Maximum Likelihood Estimation for Stable-3EF)**.** *Defining $E'_\theta(\mathbf{x}) = -\log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]$ and its corresponding marginal distribution as $\mathbb{P}'_\theta(\mathbf{x})$, the optimization of $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \mathbb{P}_\theta(\mathbf{x})\right]$ is equivalent to that of $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]\right] + \lambda_{\bar\theta}\mathbb{E}_{\mathbb{P}'_{\bar\theta}(\mathbf{x})}\left[-\log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]\right]$ when gradient descend is applied, where $\lambda_{\bar\theta}$ is the ratio of the normalizing constants determined by $E'_\theta(\mathbf{x})$ and $E_\theta(\mathbf{x})$, and $\bar\theta$ means that parameters of θ is frozen (*i.e.*, instances sampled from $\mathbb{P}'_{\bar\theta}(\mathbf{x})$ are detached).*

Proof. Since $\mathbb{P}_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{\sum_{\mathbf{x}'}\exp(-E_\theta(\mathbf{x}))}$, we have

$$\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \mathbb{P}_\theta(\mathbf{x})\right] = \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[E_\theta(\mathbf{x})\right] + \log \sum_{\mathbf{x}'} \exp(-E_\theta(\mathbf{x}'))\,. \tag{26}$$

Take the gradient of the right part in Eq. 26, and we have

$$\begin{aligned}
&\nabla_\theta \log \sum_{\mathbf{x}'} \exp(-E_\theta(\mathbf{x}')) \\
=&\sum_{\mathbf{x}} \frac{\exp(-E_\theta(\mathbf{x}))}{\sum_{\mathbf{x}'}\exp(-E_\theta(\mathbf{x}'))}\nabla_\theta E_\theta(\mathbf{x}) \\
=&\sum_{\mathbf{x}} \frac{\sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]}{\sum_{\mathbf{x}'}\exp(-E_\theta(\mathbf{x}'))} \cdot \frac{\nabla_\theta \sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]}{\sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]} \\
=&\sum_{\mathbf{x}} \frac{\nabla_\theta(1 - \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k])}{\sum_{\mathbf{x}'}\exp(-E_\theta(\mathbf{x}'))} \\
=&-\sum_{\mathbf{x}} \frac{\nabla_\theta \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]}{\sum_{\mathbf{x}'}\exp(-E_\theta(\mathbf{x}'))} \\
=&-\sum_{\mathbf{x}} \frac{\sum_{\mathbf{x}'}\exp(-E'_\theta(\mathbf{x}'))}{\sum_{\mathbf{x}'}\exp(-E_\theta(\mathbf{x}'))} \cdot \frac{\exp(-E'_\theta(\mathbf{x}))}{\sum_{\mathbf{x}'}\exp(-E'_\theta(\mathbf{x}'))} \cdot \frac{\nabla \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]}{\exp(-E'_\theta(\mathbf{x}))} \\
=&-\sum_{\mathbf{x}} \frac{\sum_{\mathbf{x}'}\exp(-E'_\theta(\mathbf{x}'))}{\sum_{\mathbf{x}'}\exp(-E_\theta(\mathbf{x}'))} \cdot \frac{\exp(-E'_\theta(\mathbf{x}))}{\sum_{\mathbf{x}'}\exp(-E'_\theta(\mathbf{x}'))} \cdot \frac{\nabla \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]}{\sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]} \\
=&-\sum_{\mathbf{x}} \frac{Z'_\theta}{Z_\theta} \cdot \frac{\exp(-E'_\theta(\mathbf{x}))}{\sum_{\mathbf{x}'}\exp(-E'_\theta(\mathbf{x}'))} \cdot \nabla_\theta \log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k] \\
=&-\frac{Z'_\theta}{Z_\theta}\mathbb{E}_{\mathbb{P}'_\theta(\mathbf{x})}\left[\nabla_\theta \log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]\right] \\
=&-\nabla_\theta\left\{\frac{Z'_{\bar\theta}}{Z_{\bar\theta}}\mathbb{E}_{\mathbb{P}'_{\bar\theta}(\mathbf{x})}\left[\log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]\right]\right\}\,.
\end{aligned} \tag{27}$$

where $\bar{\theta}$ represents that θ is frozen (*i.e.*, gradients are not computed), $Z'_{\bar{\theta}} = \sum_{\mathbf{x}'} \exp(-E'_\theta(\mathbf{x}'))$ is the normalizing constant for energy $E'_\theta(\mathbf{x})$, and Z_θ is the normalizing constant for energy $E_\theta(\mathbf{x})$. Hence, the objective $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \mathbb{P}_\theta(\mathbf{x})\right]$ is equivalent to

$$\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]\right] + \lambda_{\bar{\theta}} \mathbb{E}_{\mathbb{P}_{\bar{\theta}}(\mathbf{x}')}\left[-\log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[K+1]\right], \tag{28}$$

where $\lambda_{\bar{\theta}} = \frac{Z'_{\bar{\theta}}}{Z_{\bar{\theta}}}$. □

Theorem B.2 (Conditional Distribution Maximum Likelihood Estimation for Stable-3EF)**.** *With preliminaries from Thm. B.1, the optimization of* $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log \mathbb{P}_\theta(y|\mathbf{x})\right]$ *is equivalent to that of* $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log h_\theta(\mathbf{x})[\sigma(y)]\right] + \mu_{\bar{\theta}} \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]\right]$ *when gradient descend is applied, where* $\mu_{\bar{\theta}} = \frac{\sum_{k=K+t-1}^{K+t-2+F} h_{\bar{\theta}}(\mathbf{x})[K+1]}{\sum_{k=0}^{K+t-2} h_{\bar{\theta}}(\mathbf{x})[k]}$.

Proof. Considering the definition of $\mathbb{P}_\theta(y|\mathbf{x})$ in Eq. 2, we have

$$\begin{aligned}
&\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log \mathbb{P}_\theta(y \mid \mathbf{x})\right] \\
=&\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[E_\theta(\mathbf{x},y) + \log \sum_{y' \in \cup_{i=1}^{t} \mathcal{Y}_i} \exp(-E_\theta(\mathbf{x},y'))\right] \\
=&\sum_{i=1}^{t-1} \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y) \cap y \in \mathcal{Y}_i}\left[-\log \frac{h_\theta(\mathbf{x})[\sigma(y)]}{|\mathcal{Y}_i|}\right] + \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y) \cap y \in \mathcal{Y}_t}\left[-\log h_\theta(\mathbf{x})[\sigma(y)]\right] + \\
&\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[\log \sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]\right]
\end{aligned} \tag{29}$$

The gradient of $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y) \cap y \in \mathcal{Y}_i}\left[-\log \frac{h_\theta(\mathbf{x})[\sigma(y)]}{|\mathcal{Y}_i|}\right]$ with respect to θ is equal to that of $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y) \cap y \in \mathcal{Y}_i}\left[-\log h_\theta(\mathbf{x})[\sigma(y)]\right]$ $(i = 1, 2, \ldots, t-1)$ no matter whatever $|\mathcal{Y}_i|$ is. Therefore, the first t components in Eq. 29 can be written in a unifying form, that is

$$\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y) \cap y \in \cup_{i=1}^{t} \mathcal{Y}_i}\left[-\log h_\theta(\mathbf{x})[\sigma(y)]\right], \tag{30}$$

where For the last component in Eq. 18, we have

$$\begin{aligned}
&\nabla_\theta \log \sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k] \\
=&\frac{\nabla_\theta(1 - \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k])}{\sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]} \\
=&-\frac{\nabla_\theta \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]}{\sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]} \\
=&-\frac{\sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]}{\sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]} \cdot \frac{\nabla_\theta \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]}{\sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]} \\
=&-\frac{\sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]}{\sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]} \cdot \nabla_\theta \log h_\theta(\mathbf{x})[K+1] \\
=&\nabla_\theta\left\{-\frac{\sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]}{\sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]} \cdot \log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]\right\}
\end{aligned} \tag{31}$$

Therefore, the Objective $\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log \mathbb{P}_\theta(y \mid \mathbf{x})\right]$ is also equivalent to

$$\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log h_\theta(\mathbf{x})[\sigma(y)]\right] + \mu_{\bar{\theta}} \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]\right], \tag{32}$$

where $\mu_{\bar{\theta}} = \frac{\sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]}{\sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]}$. □

Combining Thm. 28 and Thm. 32, we get the final optimization objective

$$\begin{aligned}\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]\right] + \lambda_{\bar{\theta}}\mathbb{E}_{\mathbb{P}_{\bar{\theta}}(\mathbf{x}')}\left[-\log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]\right] + \\ \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log h_\theta(\mathbf{x})[\sigma(y)]\right] + \mu_{\bar{\theta}}\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]\right]\end{aligned} \tag{33}$$

Note that $h_\theta(\mathbf{x})[i] \geq 0$ for all $i = 1, 2, \ldots, K+t-2+F$, and then we get

$$\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \sum_{k=0}^{K+t-2} h_\theta(\mathbf{x})[k]\right] \leq \mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log h_\theta(\mathbf{x})[\sigma(y)]\right], \tag{34}$$

$$\mathbb{E}\left[-\log \sum_{k=K+t-1}^{K+t-2+F} h_\theta(\mathbf{x})[k]\right] \leq \mathbb{E}\left[-\log \max_{k\in\{K+t-1,\ldots,K+t-2+F\}} h_\theta(\mathbf{x})[k]\right]. \tag{35}$$

Hence, Eq. 33 is upper bounded by

$$\begin{aligned}2\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x},y)}\left[-\log h_\theta(\mathbf{x})[\sigma(y)]\right] + \lambda_{\bar{\theta}}\mathbb{E}_{\mathbb{P}_{\bar{\theta}}(\mathbf{x}')}\left[-\log \max_{k\in\{K+t-1,\ldots,K+t-2+F\}} h_\theta(\mathbf{x})[k]\right] + \\ \mu_{\bar{\theta}}\mathbb{E}_{\mathbb{P}_{real}(\mathbf{x})}\left[-\log \max_{k\in\{K+t-1,\ldots,K+t-2+F\}} h_\theta(\mathbf{x})[k]\right].\end{aligned} \tag{36}$$

Taking Eq. 36 as the optimization objective, we expand the expansion phase in original 3EF into a more stable form with multiple backward and forward prototypes, namely the expansion phase in Stable-3EF.

Here we further discuss how we achieve the expanded fusion phase in Stable-3EF. We also assume that we have trained a unifying model h_{θ_o} for all the prior tasks and there is a σ_o maps labels to output index of h_{θ_o}. Considering that those $t-1$ backward prototypes measures the confidence for $t-1$ prior tasks, similar to Eq. 10, we define

$$E_{\{\theta_o,\theta\}}(\mathbf{x}, y) = \begin{cases} -\log\left\{h_{\theta_o}(\mathbf{x})[\sigma(y)] + \alpha_i h_\theta(\mathbf{x})[\sigma(y)] + \beta_i\right\}, & y \in \mathcal{Y}_i, i = 1, 2, \ldots, t-1 \\ -\log h_\theta(\mathbf{x})[\sigma(y)], & y \in \mathcal{Y}_t \end{cases}. \tag{37}$$

Then we have

$$\mathbb{P}_{\{\theta,\theta_o\}}(y|\mathbf{x}) = \begin{cases} \frac{h_{\theta_o}(\mathbf{x})[\sigma(y)] + \alpha h_\theta(\mathbf{x})[0] + \beta}{\sum_{m=1}^{M}[h_{\theta_o}(\mathbf{x})[m] + \alpha h_\theta(\mathbf{x})[0] + \beta] + \sum_{k=1}^{K} h_\theta(\mathbf{x})[k]}, & y \in \mathcal{Y}_o \\ \frac{h_\theta(\mathbf{x})[\sigma(y)]}{\sum_{m=1}^{M}[h_{\theta_o}(\mathbf{x})[m] + \alpha h_\theta(\mathbf{x})[0] + \beta] + \sum_{k=1}^{K} h_\theta(\mathbf{x})[k]}, & y \in \mathcal{Y}_n \end{cases}. \tag{38}$$

α_i and β_i are obtained through the minimization of the negative log-likelihood on the exemplar-set.

C More Experimental Settings

C.1 Exemplar selection.

As we claimed, all the prior methods assume that all the old samples are available when selecting exemplars. Typically, they apply the exemplar selection strategy proposed in Rebuffi et al. (2017), where exemplars are carefully selected by greedily minimizing the derivation of the feature center between the selected exemplars and all the old samples. Besides, they assume all the old categories are equal and store the same number of exemplars for each old class. This violates the truth in many application scenarios, where available exemplars are usually imbalanced and even some instances for old categories become unavailable at the following sessions. Therefore, the robustness to the

imbalance or lack of some categories is important for CIL methods. Assuming there are k old classes and we want to store m exemplars for each old class. We design three protocols for the imbalance of exemplar-set: (1) **half-half**: in this protocol, half of k classes are allowed to store more than m exemplars and the other half of k classes are only allowed to store less than m exemplars. Specifically, defining the balance factor $\gamma \in [0, 1]$, half of k classes store $(1 + \gamma)m$ exemplars while the other store $(1 - \gamma)m$ exemplars. (2) **exp**: in this protocol, we use a negative exponential sequence of length k as the weight for each category. Specifically, the i^{th} class has the weight $\exp(-\gamma i)$ and its number of exemplars is defined by $\left\lfloor \frac{\exp(-\gamma i)}{\sum_{j=1}^{k} \exp(-\gamma j)} \cdot km \right\rfloor$. (3) **random**: we uniformly sample km exemplars from all available old instances. This protocol is the most similar to the original setting since the expectation of the number of exemplars for each category is m statistically.

C.2 Implementation details.

Our method and all baselines are implemented with Pytorch (Paszke et al., 2017). For ImageNet, we adopt the standard ResNet-18 (He et al., 2016) as our feature extractor and set the batch size as 256. The learning rate starts from 0.1 and gradually decays to zero with a cosine annealing scheduler (Loshchilov & Hutter, 2016) (170 epochs in total). For CIFAR-100, we use ResNet-32 (He et al., 2016) as our feature extractor and set the batch size to 128. The learning rate also starts from 0.1 and gradually decays to zero with a cosine annealing scheduler (170 epochs in total). For both ImageNet and CIFAR-100, we use SGD with the momentum of 0.9 and the weight decay of 5e-4 at the expansion phase. At the fusion phase, we use SGD with a momentum of 0.9 and set the weight decay to 0, and train the fused model on the exemplar-set for 30 epochs. For data augmentation, we apply the same data augmentation as Rebuffi et al. (2017). For generating samples from $\mathbb{P}'_\theta(\mathbf{x})$ when learning the energy manifold, we follow Stochastic Gradient Langevin Dynamics (SGLD) (Welling & Teh, 2011). However, to facilitate the sampling process, we use representations of real instances as the starting point of SGLD at the hidden layer, which can be seen as a disturbance to the original hidden representation of the real samples. When applying energy alignment at the evaluation phase, we also use the SGLD at the hidden layer. For the SGLD in sampling process, we use Adam (Kingma & Ba, 2014) to consider the accumulated gradient and set the learning rate and the derivation to be $1e-1$, and $1e-3$, respectively. And the number of updates is set to be 5. Therefore, we are able to generate new samples rather efficiently when learning the energy manifold. When applying energy alignment at the evaluation phase, we use a constant step size of 2 and a standard deviation of 1e-3. The number of updates in each SGLD round is set to be 30. To further stabilize the training process and improve the performance, we expand the original 3EF into Stable-3EF with multiple backward and forward prototypes. The number of backward prototypes is set to be the number of old tasks, and the number of forward prototypes is set to a constant number F. The detailed illustration and proof are provided in Appendix B. Besides, to simplify the training procedure and avoid the intractability of normalizing constants, we use a trade-off coefficient λ to replace $\lambda_{\bar{\theta}}$ and $\mu_{\bar{\theta}}$ in our implementation. By default, we set λ to be $1e-1$ and F to be 50.

D More Discussions

D.1 Contributions of 3EF

Our contribution is three-fold: 1) We propose a novel training paradigm for CIL: We efficiently train a specific module for the current task while achieving bi-directional compatibility and then fuse it with the prior model under minimal effort. 2) We provide a theoretical proof showing that the training cost is inherently the modeling of an energy-based model and apply the energy model to align the test samples and reduce their energy (*i.e.*, open world uncertainty) to further boost the performance. 3) We achieve state-of-the-art performance with lower training cost and maintain the performance when only randomly sampled old data is available while other methods fail dramatically.

D.2 Difference and connections with prior methods

Here we discuss the crucial difference and connections between our method and prior works. Though Wang et al. (2021) has proposed to use an energy-based extra dimension to model the open world

uncertainty in OOD detection, it is not suitable for CIL where old categories should be considered to alleviate forgetting. Besides, we take the joint distribution $\mathbb{P}(\mathbf{x}, y) = \mathbb{P}(\mathbf{x})\mathbb{P}(y|\mathbf{x})$ into consideration, which forces the model to learn discrimination ($\mathbb{P}(y|\mathbf{x})$) and be sensitive to the input distribution shift ($\mathbb{P}(\mathbf{x})$) and thus modules that do not match the input distribution will capture the distribution shift and weaken their influence on the ultimate predictions. Similar to Joseph et al. (2022), we also utilize the energy-based model to achieve energy alignment at the evaluation phase. While our energy-based model is inherently built at the training phase, Joseph et al. (2022) need to train an extra energy aligner after the training phase, introducing additional training costs. Apart from that, despite they assume that the task-ids of given samples are known in their implementation, we do not require that. Zhou et al. (2022) utilize multiple virtual prototypes to reserve feature space for future unseen categories, thus achieving forward compatibility. Our method can be seen as a bi-directional compatible method, with one prototype $\mathbf{p}_b$ measuring the confidence for old categories and the other prototype $\mathbf{p}_f$ modeling the out-of-distribution probability.

D.3 WHY 3EF IS ROBUST TO THE IMBALANCED EXEMPLAR-SET

Here we give an explanation about why 3EF demonstrates strong robustness to the imbalance of exemplar-set. Previous methods, whether based on dynamic-structure or regularization, rely on a unifying feature representation and classifier. When the exemplar-set is unbalanced, that is, when the number of samples in some categories is very small or does not exist, the feature representation and classifier of the class are damaged. Although our method still needs to fuse all modules into a unified classifier, each module is decoupled from each other and is responsible for different tasks. Thus, when training new tasks, even if samples of some old classes are not available, the old modules responsible for these classes are not affected. In addition, when training new modules, we classify all old classes into a special backward prototype. This special backward prototype aims to learn the characteristics shared by categories in old tasks, which is insensitive to the imbalance of old samples. In the model fusion phase, the confidence of backward prototype is uniformly added to the output of old modules, thus acting as a soft task discriminator without affecting the predictions of old modules among old tasks.

D.4 WHY 3EF IS EFFICIENT

Here we explain why 3EF is efficient. Previous methods, whether based on knowledge distillation or dynamic structure, require the forward propagation of old modules when learning new tasks, to achieve a unifying classifier. Besides, with the increasing number of modules retained, dynamic-structure-based methods suffer from increasing training costs. As we claimed, the training of 3EF consists of two phases, expansion and fusion. At the expansion phase, we only need to train the newly created module independently without the involvement of prior modules, so that the training cost at the expansion phase is equivalent to tuning a single module and will not increase at incremental stages. Though we follow Stochastic Gradient Langevin Dynamics (SGLD) (Welling & Teh, 2011) to generate samples from $\mathbb{P}'_\theta(\mathbf{x})$ when learning the energy manifold. However, instead of initializing samples randomly as the starting point, we use representations of real instances as the starting point of SGLD at the hidden layer, which can be seen as a disturbance to the original hidden representation of the real samples and greatly accelerate the generation process. Besides, we use the Adam optimizer to automatically adjust the step size and consider the cumulative gradient to further improve the generation efficiency. We run only 5 updates to get the generated samples. At the fusion phase, we only need to learn two factors including scale factor α and bias factor β through training on the randomly selected exemplar-set for about 30 epochs, whose training cost is minimal compared to the expansion phase.

Particularly, we compare the training cost of 3EF with the methods including the classical dynamic-structure-based method DER Yan et al. (2021) and regularization-based methods like iCaRL (Rebuffi et al., 2017) and WA (Wu et al., 2019). Without loss of generality, we take the t^{th} incremental stage as an example. Although all the old modules are frozen in DER, the training process still requires the forward propagation of all $t-1$ old modules and the forward propagation of the new module and then applies backpropagation to update the new module. Apart from that, DER also needs to be the final classifier with a balanced reserved dataset. Therefore, the training process of DER at t^{th} incremental stage is: $t \times$ forward propagation, 1 x backpropagation, $1 \times$ finetune classifier. Similarly, due to the requirements of knowledge distillation in regularization-based methods like iCaRL, the

training cost is: $2 \times$ forward propagation, $1 \times$ backpropagation. In contrast, in 3EF, we first train the new module in a decoupled manner, which only requires: $1 \times$ forward propagation, $1 \times$ backward propagation. Then, the fusion phase only needs to tune two parameters with a small subset, whose cost is minimal and is even more efficient than the process of simply tuning the final classifier in DER. Therefore, the training cost of 3EF consists of: $1 \times$ forward propagation, $1 \times$ backward propagation, $1 \times$ finetune α and β.

Specifically, we compare the training time of BiC (regularization-based), DER (dynamic-structure-based) and 3EF at each incremental session on CIFAR-100 B0 10 steps protocol, the results are shown in Table. D.4. We can observe that the average training time of 3EF is lower than both the regularization-based method BiC and the dynamic-structure-based method DER.

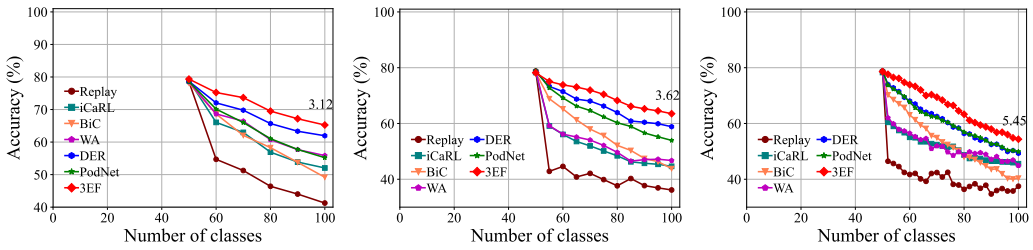
	Training time cost in each session (min)										
Methods	1	2	3	4	5	6	7	8	9	10	**Average**
BiC	25	42	43	44	42	43	45	43	46	48	42.1
DER	25	37	43	49	53	58	66	74	78	91	57.4
3EF	30	31	33	40	38	41	43	42	45	46	38.9

Table 4: Performance of 3EF with backbones in various sizes.

E More experimental results

E.1 Detailed performance on standard CIFAR-100 B50 protocols.

We report the detailed accuracy on standard CIFAR-100 B50 protocols with 5, 10, 25 incremental sessions in Fig. 7.



(a) CIFAR-100 B50 5 steps (b) CIFAR-100 B50 10 steps (c) CIFAR-100 B50 25 steps

Figure 7: Performance on standard CIFAR-100 B50 protocols.

E.2 Performance of 3EF-Distill with backbones in various sizes.

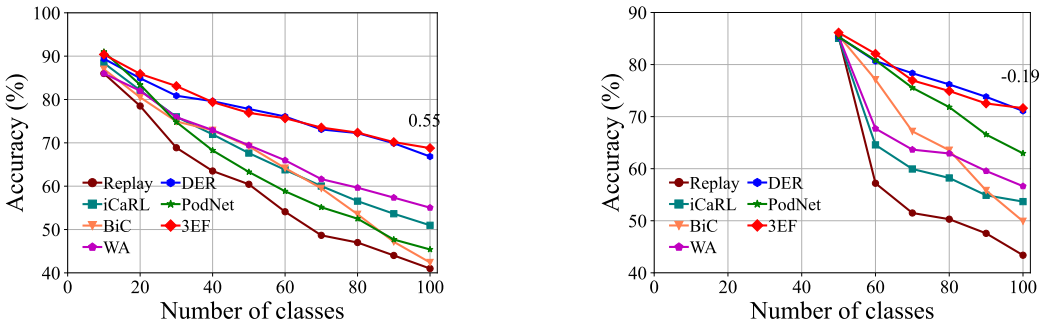
We report the detailed accuracy of 3EF-Distill with different backbones, including ResNet 20, ResNet 32, ResNet 44, ResNet 56, and ResNet 110 in Table E.2. We can see that the training strategy of 3EF is consistently effective in backbones of various sizes. The accuracy of each incremental session has an upward trend as the model becomes larger.

	Accuracy in each session (%)						
Backbones	1	2	3	4	5	6	**Average**
ResNet 20	77.18	74.15	70.96	65.45	61.28	58.22	67.87
ResNet 32	80.9	77.73	73.69	68.22	64.61	61.76	71.15
ResNet 44	81.54	77.87	74.10	69.42	65.67	62.06	71.78
ResNet 56	82.44	78.45	75.20	70.05	65.16	62.77	72.34
ResNet 110	83.6	79.72	75.47	70.44	65.89	63.52	73.11

Table 5: Performance of 3EF with backbones in various sizes.

E.3 DETAILED PERFORMANCE ON STANDARD IMAGENET-100 PROTOCOLS.

We report the detailed accuracy on standard ImageNet-100 Fig. E.1.

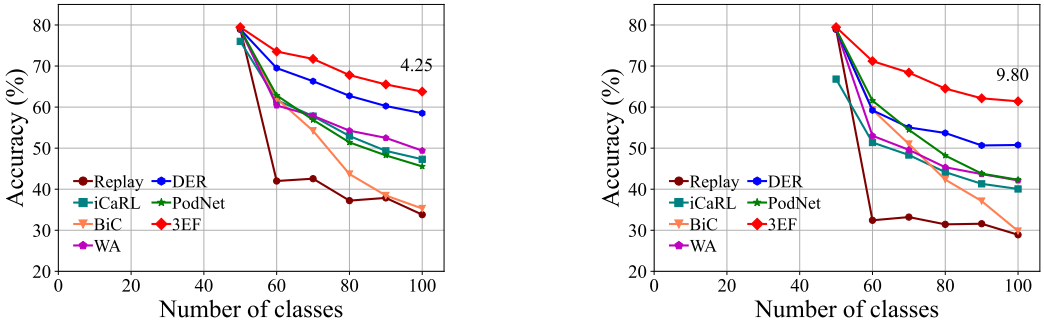


(a) ImageNet-100 B0 10 steps (b) ImageNet-100 B50 5 steps

Figure 8: Performance on ImageNet-100 protocols.

E.4 MORE RESULTS ON IMBALANCED PROTOCOLS.

We report more results on imbalanced protocols in Fig. 9.



(a) half-half: $\gamma = 0.5$ (b) exp: $\gamma = 0.9$

Figure 9: Performance on imbalanced protocols.