Efficient Feature Transformations for Discriminative and Generative Continual Learning

Vinay Kumar Verma[1], Kevin J Liang[1,2], Nikhil Mehta[1], Piyush Rai[3], Lawrence Carin[1]
[1]Duke University [2]Facebook AI [3]IIT Kanpur
vinaykumar.verma@duke.edu

Abstract

As neural networks are increasingly being applied to real-world applications, mechanisms to address distributional shift and sequential task learning without forgetting are critical. Methods incorporating network expansion have shown promise by naturally adding model capacity for learning new tasks while simultaneously avoiding catastrophic forgetting. However, the growth in the number of additional parameters of many of these types of methods can be computationally expensive at larger scales, at times prohibitively so. Instead, we propose a simple task-specific feature map transformation strategy for continual learning, which we call Efficient Feature Transformations (EFTs). *These EFTs provide powerful flexibility for learning new tasks, achieved with minimal parameters added to the base architecture. We further propose a feature distance maximization strategy, which significantly improves task prediction in class incremental settings, without needing expensive generative models. We demonstrate the efficacy and efficiency of our method with an extensive set of experiments in discriminative (CIFAR-100 and ImageNet-1K) and generative (LSUN, CUB-200, Cats) sequences of tasks. Even with low single-digit parameter growth rates, EFTs can outperform many other continual learning methods in a wide range of settings.*

1. Introduction

While deep learning has led to impressive advances in many fields, neural networks still tend to struggle in sequential learning settings, largely due to catastrophic forgetting [31, 39]: when the training distribution of a model shifts over time, neural networks overwrite previously learned knowledge if not repeatedly revisited during training. Pragmatically, this typically means that data collection must be completed before training a neural network, which can be problematic in settings like reinforcement learning [34] or the real world [44], which is constantly evolving. Otherwise, the model must constantly be re-trained as new data arrives. This limitation significantly hampers building and deploying intelligent systems in changing environments.

A variety of continual learning methods [37] have been proposed to address this shortcoming. Regularization-based methods [18, 62, 36, 1, 41] prevent forgetting by constraining model parameters from drifting too far away from previous solutions, but they can also restrict the model's ability to adapt to new tasks, often resulting in sub-optimal solutions. Additionally, regularization methods commonly make the assumption that each weight's importance for a task is independent, which may explain why they have difficulty scaling to more complex networks and tasks. Replay methods [27, 46, 36, 42] retain knowledge by rehearsing on data saved from previous tasks. While effective at preventing forgetting, the performance of replay-based approaches is highly dependent on the size and contents of the memory buffer, and in certain strict settings, saving any data at all may not be an option. The nature of this replay buffer also tends to highly bias the model toward recently learned tasks. As the number of tasks grows, performance degrades quickly, especially in large-scale settings [38, 40].

As an alternative to regularization or replay, expansion methods [43, 59, 58, 32] combat forgetting by growing the model with each task. Expansion alleviates catastrophic forgetting by design, and additional necessary capacity can be easily added to accommodate new knowledge needed for new tasks. This ability to scale arbitrarily without needing to save any data gives expansion methods the best chance to succeed in large-scale settings. However, added capacity to model future tasks needs to be carefully balanced against the number of parameters added, especially since the number of tasks the model must learn is often unknown ahead of time. Too inefficient of an approach can easily exceed computational resources even after a moderate number of tasks. Moreover, many previous expansion methods

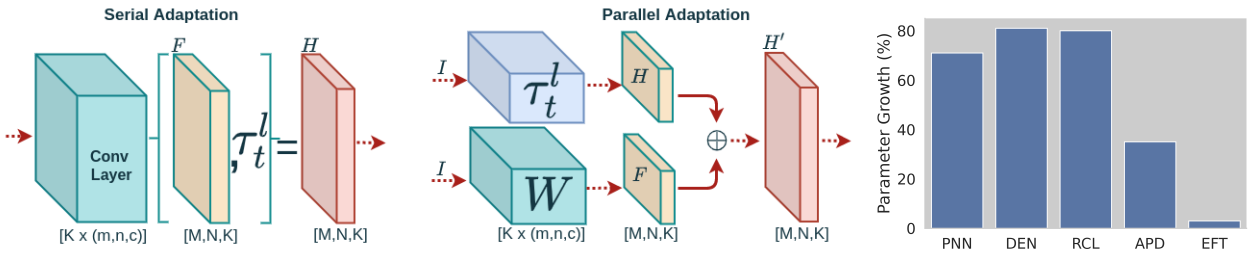

Figure 1. **Left:** EFT transforms global feature map F to a task-specific feature map H with parameters τ_t. **Right:** Parameter growth for the 10 task on the LeNet architecture, EFT shows significantly lower growth than other expansion models.

are either computationally inefficient or inflate model size by a significant amount.

To overcome these limitations, we propose a compact, task-specific feature map transformation for large-scale continual learning, which we call *Efficient Feature Transformation (EFT)*. In particular, we partition the model into *global* parameters (θ) and task-specific *local* parameters (τ_t), with the pair (θ, τ_t) as the optimal parameters for a particular task t (see Figure 1). In constructing these local transforms, we leverage efficient convolution operations [14, 49, 56], maintaining expressivity, while keeping model growth small. We also minimize the impact on the global base architecture, allowing us to use pre-existing architectures, which can be critical for achieving strong performance in large-scale settings. This compact nature of the added transformations also makes EFTs faster to train than comparable methods because we have to update only task-specific parameters. Finally, we propose a strategy for maximizing feature distance to improve task prediction, a critical component for continual learning methods operating in class incremental settings.

To show the efficacy and efficiency of the proposed approach, we extensively evaluate our model on a variety of datasets and architectures. In class incremental and task incremental sequential classification settings, EFTs achieve significant performance gains on CIFAR-100 [19] and ImageNet [6] with only a minor growth in parameter count and computation. We also evaluate our approach for continual generative modeling, demonstrating a 22.7% relative improvement in FID [12] on the LSUN [60], CUB-200 [52], and ImageNet [6] cat datasets compared to recent state-of-the-art models.

2. Methods

2.1. Efficient Feature Transforms

Commonly used modern vision and language architectures can be quite large, sometimes having tens to hundreds of millions of parameters [11, 7]. This can make them incompatible with many previous continual learning methods: if one must add entire parallel columns [43] or regularize the entirety of a model's weights [18] per task t, it can very easily exceed system capacity. Additionally, many of large-scale networks have been carefully engineered through years of manual architecture search, resulting in structures with specific hyperparameter settings or inductive biases that make them especially effective [3]. Modifications to these design may lead to unintentional degradation of the model's overall effectiveness. Many previous continual learning methods significantly alter the base network in a way that may be detrimental to overall performance.

With these considerations in mind, we propose partitioning the network into a global base network parameterized by θ and task-specific transformations τ_t. During a task t, only the task-specific parameters τ_t are trained; previous local parameters $\tau_{<t}$ and global parameters θ remain unchanged. Under this set-up, the global network can be any architecture, preferably an effective one for the problem at hand. Of particular note, this means that pre-trained weights can also be employed, if available and appropriate. Because attempting to transform parameters θ in its entirety can be expensive, we instead propose task-specific local transformations of the *features* at various layers within the network. We aim to keep the task-specific transformations minimal. In particular, using efficient operations, our approach can keep the number of parameters in τ_t very small without degrading performance. To ensure network compatibility, we also ensure that the dimensionality of each transformed feature tensor remains the same as the original, meaning that this operation can be inserted into any architecture without any changes. We outline here how this can be done for 2D convolutional and fully connected layers. We focus on these two operations as they comprise the backbone of many deep architectures, but these concepts can be generalized to other types of layers as well.

2.1.1 2D Convolutional Layer

Convolutional neural networks (CNNs) [22] play a major role in many modern computer vision algorithms, with

the 2D convolutional layer being one of the primary building blocks. Let I be the input to a convolutional layer. In the typical formulation, each convolution layer is composed of K convolutional filters $\mathcal{W}$; each filter $\mathcal{W}_k$ in $\mathcal{W}$ has size $(m \times n \times c)$, with m and n being the filter's spatial dimensions and c the number of channels in I. Each filter $\mathcal{W}_k$ convolved with I produces an output feature map $F_k \in \mathbb{R}^{M \times N}$. The whole operation can be summarized as

$$F = \mathcal{W} * I \tag{1}$$

with $*$ being the 2D convolution operator and $F \in \mathbb{R}^{M \times N \times K}$ being all feature maps F_k stacked into a tensor. Generally, this operation includes an additive bias, which we omit here for notational convenience.

Rather than adjusting $\mathcal{W}$ directly, we instead propose appending a small convolutional transformation to change the features F. As this operation needs to be done for each task t, we seek to do so efficiently, more so than existing expansion-based methods. We do this by leveraging groupwise [20, 56] and depthwise [14] convolutions, producing new features for a specific task. We use two types of convolutional kernels: $\omega^s \in \mathbb{R}^{3 \times 3 \times a}$ for capturing spatial features within groups of channels and $\omega^d \in \mathbb{R}^{1 \times 1 \times b}$ for capturing features across channels at every pixel in F, where a and b are hyperparameters defining the group size. For the former case, we perform a groupwise convolution with cardinality a using depth-a convolutions. In other words, we split convolutional feature maps F into K/a groups, and for each spatially convolve a unique set of a filters ω_i^s over the group of feature maps. The resulting K/a groups of a feature maps H_i^s, are all concatenated into a single tensor $H^s \in \mathbb{R}^{M \times N \times K}$. Importantly, this is the same dimensions as the original feature map F, keeping with our goal of keeping the architecture unchanged after inserting this transformation. A similar operation is done for filters ω^d, but with cardinality b and depth b, to constitute feature maps $H^d \in \mathbb{R}^{M \times N \times K}$. The construction of both feature maps can be expressed as:

$$H^s_{ai:(ai+a-1)} = [\omega^s_{i,1} * F_{ai:(ai+a-1)} \mid \ldots \mid \omega^s_{i,a} * F_{ai:(ai+a-1)}], \quad i \in \{0, \ldots, \frac{K}{a} - 1\} \tag{2}$$

$$H^d_{bi:(bi+b-1)} = [\omega^d_{i,1} * F_{bi:(bi+b-1)} \mid \ldots \mid \omega^d_{i,b} * F_{bi:(bi+b-1)}], \quad i \in \{0, \ldots, \frac{K}{b} - 1\} \tag{3}$$

where $\mid$ is the concatenation operation, $H^s_{ai:(ai+a-1)} \in \mathbb{R}^{M \times N \times a}$, and $H^d_{bi:(bi+b-1)} \in \mathbb{R}^{M \times N \times b}$. Generally, $a \ll K$ and $b \ll K$, so this operation is far more parameter efficient than simply using another convolutional layer with K filters. The latter requires at least as many parameters as the original network itself for each task, the same as learning separate models per task.

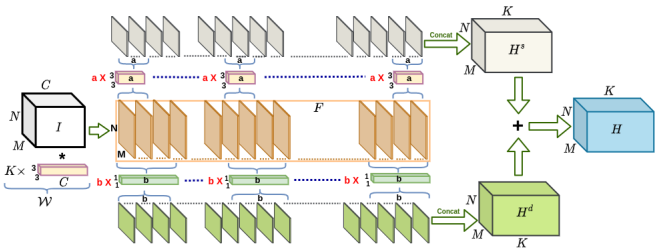

Figure 2. Illustration of the proposed parameter-efficient transformation of convolutional features.

We combine the features from the spatial and depth convolutions H^s and H^d additively:

$$H = H^s + \gamma H^d \tag{4}$$

where $\gamma \in \{0, 1\}$ is an indicator indicating if the pointwise convolutions ω^d are employed. Note, if $\gamma = 0$, we do not perform convolutions with ω^d; this sacrifices some expressivity, but further reduces the number of added parameters per task. For example, the extreme case of $a = 1$ and $\gamma = 0$ results in a small 0.17% increase in parameters per task in ResNet-18 while still achieving state-of-the-art performance on the ImageNet-1K/10 continual learning task.

2.1.2 Fully Connected Layer

Fully connected layers are common in many architectures, typically to project from one dimensionality to another. In convolutional neural networks, they are frequently used to project to the number of output classes to produce the final prediction. A fully connected layer is implemented as a matrix multiply between a weight matrix W and an input vector v, producing an output feature vector $f = Wv$; the bias vector is again omitted for convenience.

As with the convolutional layer, where we transform the convolutional features F to a task-specific feature H with additional convolutional operations, we also transform output vector f with another fully connected layer (parameterized by E) to a task-specific feature vector h. Like in the convolutional case, we must put restrictions on the form of E to prevent this operation from being overly costly. In particular, we constrain E to be diagonal, which significantly reduces the number of parameters. This operation can be expressed as $h = Ef$. In practice, since E is diagonal, this operation can be implemented as a Hadamard product.

2.1.3 Parallel Adaption

We have thus far introduced our approach as an efficient, sequential feature transformation, adapting the features to each task *after* its computation in the base network. An alternative parameterization is to compute these feature calibrations in parallel (Figure 1, center), making the transformation additive rather than compositional:

$$H' = (\mathcal{W} * I) \oplus (\tau_t^l * I) \tag{5}$$
$$= F \oplus H \tag{6}$$

where $\oplus$ is element-wise addition and τ_t^l is the task specific parameter at layer l. Empirically we find that both sequential and parallel models achieve similar performance. Unless otherwise mentioned, we report our results with sequential transformations. A comparison of the two types is shown in the supplementary material.

2.2. Task Prediction

Knowing which task t an inference-time input corresponds to is a common implicit assumption in continual learning, but in some settings this information is unavailable, necessitating predicting the task t alongside the class; such a setting has been referred to as class incremental learning (CIL) [15]. Predicting the task ID can be challenging, especially when data from previous tasks is not saved. We choose to predict t by selecting the task-specific set of parameters τ_t that produces the maximum confidence prediction for the input. However, a naive, direct measurement of the post-softmax prediction entropy often performs poorly, as the typical cross-entropy training objective tends to produce high confidence in deterministic neural networks, even for out-of-distribution (OoD) samples [10]. To remedy this, we propose feature distance maximization, a simple regularization to increase task prediction ability.

Without intervention, the pre-output layer feature distributions learned by the parameters τ_t for each task t may show overlap, which may hinder attempts to discriminate task features. To mitigate this, we add a regularizer to create a margin between the features produced by each task's τ_t for a given task's data:

$$\mathcal{L}_{\mathcal{M}} = \sum_{i=1}^{t-1} \max(\Delta - \mathrm{KL}(P_t || Q_i), 0) \tag{7}$$

where $P_t = \mathcal{N}(\mu_t, \Sigma_t)$ and $Q_i = \mathcal{N}(\mu_i, \Sigma_i)$ are the distributions of the current task t and earlier task $i < t$. This regularizer helps the model learn representations for τ_t such that the current task data ($\mathcal{D}_t$) has at least Δ separability in the feature space from the features encoded by previous tasks' parameters $\tau_{<t}$.

2.3. Summary

The joint loss for Efficient Feature Transforms (EFTs) is defined as:

$$\mathcal{L}_{\mathrm{EFT}} = \mathcal{L}(\hat{y}, y) + \lambda \mathcal{L}_{\mathcal{M}} \tag{8}$$

where $\mathcal{L}(\hat{y}, y)$ is the standard cross-entropy loss between the predicted class $\hat{y}$ and ground truth y, and λ is a weighting hyperparameter. At test time, for a given input x, we measure the output entropy with each τ_t and predict the task ID as $\arg\max_t(x|\theta, \tau_t)$. Once we have the task ID, we can choose the task-specific parameter τ_t to predict a class.

In summary, we recognize that if one adjusts the original model parameters in a given layer (what we call "global" model parameters), this leads to an adjustment in the features output from the layer. However, the number of layer-dependent global parameters may be massive, and adjustment of them may cause loss of information accrued from prior datasets and tasks. Since layer-wise adjustment of global parameters simply adjusts the output feature map, we leave the global layer-dependent features unchanged, and introduce a new set of lightweight task-specific parameters, that directly refine the feature map itself. These task-specific parameters may adjust to new tasks, while maintaining the knowledge represented in well-training global model parameters. We have developed methods to apply this concept to convolutional and fully-connected layers. Additionally, we make the model more amenable to task ID prediction by maximizing a margin between task-specific feature distributions.

3. Related Work

The continual learning literature is vast [5, 37]. Broadly, continual learning (CL) methods can be grouped into three categories, by strategy: replay, regularization, and expansion. We focus here primarily on expansion-based methods, as they are the most closely related to our work.

A number of previous works [66, 59, 43, 57, 8, 45, 29, 30, 51, 58] have proposed methods for expanding a neural network to learn a sequence of tasks. For example, Progressive Neural Networks (PNNs) [43] adds a new neural network column for each new task; weights for previous tasks are frozen in place, and lateral connections are added for forward knowledge transfer. Side-tuning [66] takes an approach similar to but simpler than PNNs: they propose learning small task-specific networks whose outputs are fused to the larger base network. Dynamically Expandable Network (DEN) [59] optimizes three sub-problems of selective training, network expansion, and network duplication, while Reinforced Continual Learning RCL [57] using reinforcement

Table 1. Average accuracy on CIFAR-100 in class incremental learning setting when trained on 10 tasks sequentially.

Dataset / #Tasks	Methods	1	2	3	4	5	6	7	8	9	Final
CIFAR-100/10	Finetune	88.5	47.1	32.1	24.9	20.3	17.5	15.4	13.5	12.5	11.4
	FixedRep	88.5	45.9	30.1	22.4	17.7	15.2	12.3	11.1	9.8	8.8
	LwF [25]	88.5	70.1	54.8	45.7	39.4	36.3	31.4	28.9	25.5	23.9
	EWC [18]	88.5	52.4	48.6	38.4	31.1	26.4	21.6	19.9	18.8	16.4
	EWC+SDC [61]	88.5	78.8	75.8	73.1	71.5	60.7	53.9	43.5	29.5	19.3
	SI [62]	88.5	52.9	40.7	33.6	31.8	29.4	27.5	25.6	24.7	23.3
	MAS [1]	88.5	42.1	36.4	35.1	32.5	25.7	21.0	19.2	17.7	15.4
	RWalk [2]	88.5	55.1	40.7	32.1	29.2	25.8	23.0	20.7	19.5	17.9
	DMC [65]	88.5	76.3	67.5	62.4	57.3	52.7	48.7	43.9	40.1	36.2
	EFT-a_4b_0 (+1.7%)	90.2	75.7	68.6	62.6	58.1	53.9	51.6	48.7	46.7	**44.4**
	EFT-a_4b_8 (+2.0%)	90.1	75.5	68.9	63.6	58.7	54.6	52.4	49.8	47.3	**45.1**
	EFT-a_8b_{16} (+3.9%)	90.2	76.2	70.1	63.1	57.9	53.6	52.1	49.6	47.6	**45.5**

learning to determine the growth of the architecture. Alternatively, some recent works leverage Bayesian non-parametrics to let the data dictate expansion [32, 21, 23], but the benchmarks considered in Bayesian methods have been limited to MNIST and CIFAR-10.

An iterative training and pruning strategy [29, 28, 16] has also been proposed for expansion-based continual learning, with the pruning incorporated to reduce model growth; however, PackNet [29] still requires saving masks to recover networks of previous models, which can take up storage space as the number of tasks grows. Other mask-based approaches have proven popular in recent years. HAN [45] proposes hard attention masks for the each task. TFM [30] applies ternary masks to the feature maps, which results in less memory per mask, as the feature maps are often smaller in size than the number of weights in the model. Additive Parameter Decomposition (APD) [58] uses masks to decompose the model parameters into task-shared and task-specific parameters; however, the significant changes made to architecture makes it harder to scale and precludes the use of pre-trained weights. SupSup [54] proposes supermasks [67] for each task, storing the supermask in a fixed size Hopfield network network [13]. Masking approaches work well for task-incremental settings, but the task prediction required for class incremental learning to select the appropriate mask can be challenging and costly; replay is sometimes relied upon [29]. By contrast, our proposed method shows promising results in both scenarios, without relying on replay.

4. Experiments

We demonstrate our approach's performance for multiple base architectures (ResNet [11], VGG [47], AlexNet [20]) and datasets (ImageNet-1K [6], CIFAR-100 [19], Tiny-ImageNet) in class and task incremental learning scenarios.[1] We use the nomenclature `[DATASET]`-$\mathcal{C}/T$ throughout this section to denote a continual learning set-up with $\mathcal{C}$ total classes evenly divided into T tasks, meaning each task involves learning $\frac{\mathcal{C}}{T}$ new classes. We also explore generative continual learning on the StackGAN-v2 [64] architecture for the four sequential task (cats, birds, church and tower), and run several ablation studies. In all scenarios, we achieve a significant improvement compared to the base model.

4.1. CIFAR-100

CIFAR-100 [19] is commonly among the most challenging datasets considered by many previous continual learning methods. We break down the 100 classes into three different class sequence splits: CIFAR-100/5, CIFAR-100/10 and CIFAR-100/20. With more classes per tasks, CIFAR-100/5 requires the model to learn a harder problem for each task, while CIFAR-100/20 increases the length of the task sequence, testing a continual learning method's retention. We run our CIFAR experiments in the class incremental setting, which means task information is unknown at test time, thus requiring task prediction. In our CIFAR experiments, we utilize the ResNet-18 [11] architecture for CIFAR datasets as our base architecture. We report the average top-1 accuracy of all previously seen tasks up to t, for each t, averaged over 5 random permutations of task order. We compare EFT with several other popular continual learning methods for CIFAR-100/10 in Table 1, and plot performance over time (tasks seen) for CIFAR-100/5 and CIFAR-100/20 in Figure 3.

If simply training on each task dataset in sequence, finetuning the model on each incoming dataset without any continual learning measures, we observe severe catastrophic forgetting. Because of the diversity and complexity of the classes in each task, previous tasks are forgotten almost entirely in order to specialize the model for the current task. While other methods show improvements over the finetuning baseline, all still show

[1] https://github.com/vkverma01/EFT

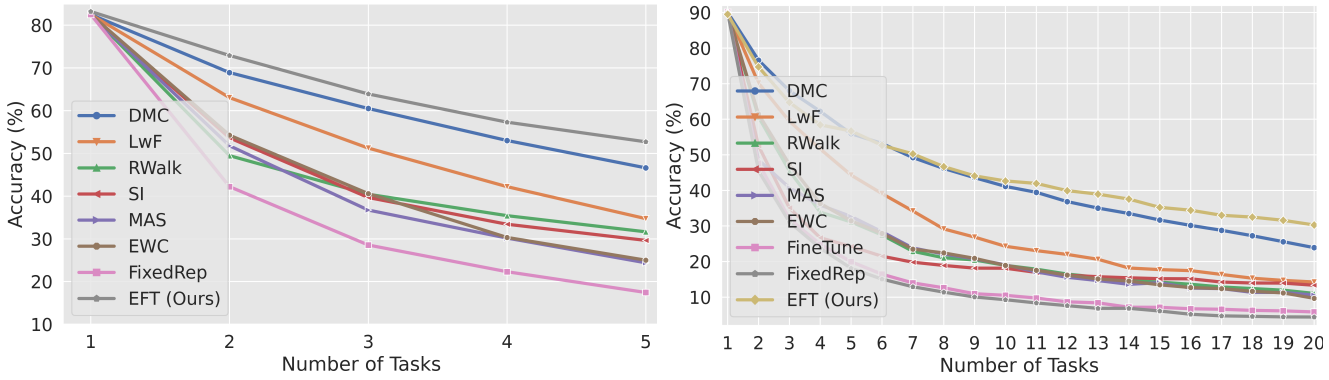

Figure 3. CIFAR-100/5 and CIFAR-100/20 result in the class incremental learning setup.

Table 2. Comparison with the state-of-the-art. ImageNet-1K/10 on AlexNet and Tiny-ImageNet-200/10 on VGG-16 from scratch. Accuracy of each task after learning all tasks.

Dataset / #Tasks	Methods	1	2	3	4	5	6	7	8	9	10	Avg
Tiny ImageNet/10 (VGG-16)	Finetuning	38.1	36.0	43.2	44.1	45.5	54.5	50.3	50.5	51.0	61.2	47.4
	Freezing	51.7	36.4	39.5	41.7	42.9	46.2	45.7	41.1	41.2	40.9	42.7
	LfL [17]	32.4	35.4	43.4	44.1	45.0	55.9	49.4	51.1	58.6	61.4	47.7
	LwF [25]	45.1	45.5	53.5	57.6	56.2	65.7	63.5	58.4	59.6	58.5	56.4
	IMM-mode [24]	50.6	38.5	44.7	49.2	47.5	51.9	53.7	47.7	50.0	48.7	48.3
	EWC [18]	33.9	35.4	43.6	46.7	49.5	52.5	47.8	50.2	56.6	61.4	47.8
	HAT [45]	46.8	49.1	55.8	58.0	53.7	61.0	58.7	54.0	54.6	50.3	54.2
	PackNet [29]	52.5	49.7	56.5	59.8	55.0	64.7	61.7	55.9	55.2	52.5	56.4
	TFM [30]	48.2	47.7	56.7	58.2	54.8	62.2	61.5	57.3	58.5	54.8	56.0
	EFT-a_8b_{16} (3.1%)	67.2	62.5	69.4	62.6	68.3	69.6	59.0	67.8	71.5	70.1	**66.8**
ImageNet-1K/10 (AlexNet)	Finetuning	25.8	32.2	31.4	37.8	39.1	43.7	46.0	50.0	53.4	63.7	42.3
	Freezing	68.8	53.5	52.0	51.2	51.3	53.9	52.2	53.9	51.7	51.2	54.0
	LwF [25]	27.6	37.2	42.0	44.4	50.5	56.6	57.9	61.2	62.0	62.7	50.2
	IMM-mode [24]	68.5	53.6	52.1	51.7	52.5	55.5	54.7	53.5	54.2	51.8	54.8
	EWC [18]	21.8	26.5	29.5	32.9	35.6	40.4	40.0	44.7	47.8	61.1	38.0
	PackNet [29]	67.5	65.8	62.2	58.4	58.6	58.7	56.0	56.5	54.1	53.6	59.1
	EFT-$a_{16}b_{64}$ (0.6%)	69.0	63.2	60.1	62.5	53.6	57.2	55.1	52.8	55.7	62.5	**59.4**

major signs of forgetting. We show results for EFTs with different convolutional group sizes $\{a, b\}$, using the notation EFT-$a_\alpha b_\beta$ to indicate EFT with $a = \alpha$ and $b = \beta$. For CIFAR-100/10, EFT-a_4b_0, EFT-a_4b_8, and EFT-a_8b_{16} result in small 1.7%, 2.0%, and 3.9% increases in parameters per task respectively, leading to 8.2%, 8.9% and 9.3% absolute gain in final average accuracy in comparison to recent regularization or expansion-based models. We observe a similar pattern for CIFAR-100/20 and CIFAR-100/5: we see consistently better results throughout the task sequence when there is a larger number of tasks (CIFAR-100/20) and when individual tasks are harder (CIFAR-100/5). More details about the experimental setup and hyperparameters can be found in the supplementary material, as well as additional comparisons with SupSup [54] and CCLL [48].

4.2. ImageNet

The ImageNet-1K [6] classification dataset contains 1000 classes based on the WordNet [33] hierarchy. Tiny-ImageNet is a smaller subset of the ImageNet dataset, containing 200 classes downsampled to $64 \times 64 \times 3$ resolution. For many years, ImageNet-1K served as a measuring stick for deep computer vision progress, and it still remains a very difficult and rarely tested setting for continual learning. Our ImageNet experiments are conducted in the task-incremental learning scenario, assuming the task-id at test time to choose τ_t for that task. We use VGG-16 [47] and AlexNet [20] CNN architectures for Tiny-ImageNet-200/10 and ImageNet-1K/10, respectively. The results are shown in the Table 2. We report the average top-1 accuracy across all encountered tasks for three random task orders. Compared with the baselines, we see significant performance improvements

Table 3. Final test accuracies after heterogeneous dataset sequence with VGG16. ↓ follow the SVHN → CIFAR-10 → CIFAR-100 task order, while ↑ corresponds to CIFAR-100 → CIFAR-10 → SVHN.

	L2T		PB [28]		PNN [43]		APD [58]		EFT-$a_{16}b_{16}$	
Task order	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑
SVHN	10.7	88.4	96.8	96.4	96.8	96.2	96.8	96.8	96.8	95.5
CIFAR-10	41.4	35.8	83.6	90.8	85.8	87.7	90.1	91.0	89.2	90.4
CIFAR-100	29.6	12.2	41.2	67.2	41.6	67.2	61.1	67.2	64.6	71.5
Avg	27.2	45.5	73.9	84.8	74.7	83.7	83.0	85.0	**83.5**	**85.8**

Table 4. Expansion cost and CIFAR-100/10 average accuracy with the convolutional architecture in [58].

Methods	Cost	Acc.
PNN [43]	1.71x	54.90
DEN [59]	1.81x	57.38
RCL [57]	1.80x	55.26
APD [58]	1.35x	60.74
EFT	1.04x	**64.17**

Table 5. Feature distance maximization ablation for CIFAR-100/10 with ResNet-18: CIL and task prediction (TP) accuracies after 10 tasks.

Models	CIL	TP
a_8b_{16}	44.4	44.9
$a_8b_{16} + \mathcal{L}_{\mathcal{M}}$	45.5	46.1
a_4b_8	44.0	44.3
$a_4b_8 + \mathcal{L}_{\mathcal{M}}$	45.1	45.7

with EFTs. Notably, with VGG-16 on Tiny-ImageNet-200/10, we observe a 10.4% absolute gain on the 10-task sequence. Please refer to the supplementary material for more details about the model architecture and hyperparameter settings.

4.3. Heterogeneous Datasets

While the classes of ImageNet-1K and CIFAR-100 cover a diverse set of categories, tasks drawn solely from CIFAR-100/10 or ImageNet-1K/10 ultimately share many similarities: for example, they mostly consist of natural images, and the total number of classes per task are the same. It can also be of interest to evaluate if continual learning methods can adapt to shifts in domain or label space cardinality. To test this, we evaluate our method on Street View House Numbers (SVHN) [35], CIFAR-10, and CIFAR-100, using VGG-16 [47]. Between SVHN and CIFAR-10, the model must adapt from digits to animals and vehicles, and between CIFAR-10 and CIFAR-100, the model must adapt between a 10-class and a 100-class problem.

Results training on this sequence of tasks are shown in Table 3, for both directions. Once again, compared to the baselines, we observe that our approach does a significantly better job retaining performance on old tasks while also maintaining plasticity for learning new ones. Additionally, previous work [58] has highlighted the potential importance of task-order robustness in continual learning, which can play a role in model fairness. We observe that our method has low order sensitivity as well, with similar accuracies on each dataset regardless of the order the datasets are learned.

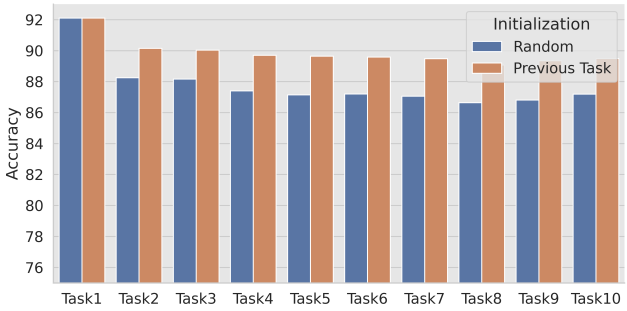

Figure 4. Accuracy after task t, with and without forward transfer.

Table 6. Computation for CIFAR-100/10 with ResNet-18.

Models	Paramater (↑)	FLOPs (↑)	Accuracy
Base	11.17M	1.11G	–
a_8b_{16}	11.60M (3.87%)	1.21G (8.78%)	45.5
a_4b_8	11.39M (1.97%)	1.16G (4.44%)	45.1
a_4b_0	11.35M (1.59%)	1.15G (3.60%)	44.4
a_0b_{64}	11.48M (2.79%)	1.18G (6.36%)	43.2
a_0b_{32}	11.33M (1.42%)	1.15G (3.20%)	42.3

4.4. Forward Transfer

Positive forward knowledge transfer is an essential ability in continual learning systems. Successful transfer leverages previously learned knowledge, reducing the amount of new data and training time necessary to learn future tasks. EFTs achieve forward transfer by initializing the task-specific parameters τ_t with the previous task local parameters τ_{t-1}. We evaluate the forward transfer of EFTs by comparing this forward transfer approach with a random initialization strategy for τ_t, plotting the performance of the model on CIFAR-100/10 using ResNet-18 (a_8b_{16}) in Figure 4. Empirically, we observe that initializing τ_t from τ_{t-1} results in consistently better performance, verifying that our approach results in positive forward transfer.

4.5. Expansion and Computation Cost

While expansion-based methods are able to effectively scale to an arbitrary number of tasks, the number of added parameters needed for each task is an important consideration. At worst, the model growth per task should not exceed the size of the original architecture, as at that point, it is no more efficient than learning a set of independent full models and then ensembling. We compare our parameter-efficient formulation with other expansion-based methods using the convolutional architecture of [58], showing both final total parameter expansion factor and average accuracy at the end of the CIFAR-100/10 task sequence in Table 4. For these experiments, we set $a = 5$ and b to the full channel depth (20 and 50 for the first and second layers, re-

Table 7. GAN FIDs evaluated after training on each dataset, sequentially

Task t	Cats 1	Cats 2	Cats 3	Cats 4	Birds 1	Birds 2	Birds 3	Birds 4	Churches 1	Churches 2	Churches 3	Churches 4	Towers 1	Towers 2	Towers 3	Towers 4	Final Average
Finetune	29.0	156.9	189.6	182.8	-	21.2	174.5	161.5	-	-	11.4	48.0	-	-	-	12.7	101.2
EWC [18]	29.0	147.3	190.7	186.2	-	65.9	165.4	155.9	-	-	38.2	48.2	-	-	-	33.8	106.1
MeRGAN-RA [55]	29.0	56.4	58.2	61.3	-	50.9	53.7	65.9	-	-	23.2	28.3	-	-	-	15.7	42.8
EFT-$a_{32}b_{16}$ (+4.8%)	29.0	29.0	29.0	29.0	-	44.1	44.1	44.1	-	-	32.3	32.3	-	-	-	27.2	**33.1**

spectively). Notably, our proposed method requires the least amount of expansion by a large margin (only 1.04× more parameters than the base network), while also achieving the highest overall accuracy.

While the number of parameters is representative of the amount of memory necessary for the model, memory is not the only computational consideration; the number of floating point operations (FLOPs) is also an important consideration. While the two are often correlated, they are not necessarily one-to-one: for example, convolutional layer weights take up only 10% of the total parameters in VGG-16, but they represent 90% of the total FLOPs. Thus, we also calculate the increase in FLOPs resulting from adding EFTs with various values of $\{a, b\}$ to a ResNet-18 architecture on CIFAR-100/10 in Table 6. In terms of wall clock time, we also observe minimal impact form EFTs. For example, inference time per task (1K samples) requires 1.05 and 0.87 seconds for EFTs and the original model respectively, for task incremental learning (TIL).

4.6. Task Inference Regularization

We proposed max-margin task-specific feature distribution regularization in Section 2.2 to improve task prediction. We conduct an ablation study of the loss term $\mathcal{L}_{\mathcal{M}}$ to demonstrate its effectiveness. We use 10% of the training data as a validation set to tune hyperparameters, finding $\lambda_1 = 0.05$ and $\Delta = 1$ to work well. Our empirical study (Table 5) shows that the regularization term $\mathcal{L}_{\mathcal{M}}$ boosts task prediction accuracy.

4.7. Generative Modeling

In addition to the discriminative models shown above, we also apply our continual learning approach to deep generative models. As in classification, if a generative adversarial network (GAN) [9] is trained on a sequence of different data distributions, it will experience catastrophic forgetting [26, 63, 4, 50], tending to forget how to produce samples from previous distributions. We train a StackGAN-v2 [64] on a sequence of different image distributions: cats (ImageNet [6]), birds (CUB-200 [52]), churches and towers (LSUN [60]).

After completing training on a dataset, we report Fréchet Inception Distance (FID) [12] to quantify GAN performance for each previously seen dataset (Table 7).

Figure 5. Generated samples from the cat, bird, church, and tower distributions after training on each dataset. While naive finetuning (top) forgets previously seen distributions, EFT (bottom) effectively retains them.

A finetuning approach commits the entirety of the network to learning each new dataset, without any restrictions or regard for previous tasks, and is thus able to attain very low FID scores on each dataset directly after training on them. However, as expected, we also observe that finetuning directly results in severe catastrophic forgetting, leading to the generator completely forgetting previous data distributions (see Figure 5, top); for the finetune approach, each supercolumn in Table 7 shows excellent performance for a task when first trained on, followed by immediate degradation. Applying the popular regularization approach EWC [18] does not mitigate this issue; we still see severe forgetting. On the other hand, we observe that EFT is able to effectively learn multiple data distributions sequentially without forgetting (see Figure 5, bottom), beating MeRGAN-RA [55] in final average FID.

5. Conclusions

We propose Efficient Feature Transforms, balancing the need for expressivity to model incoming tasks with parameter efficiency, allowing the model to expand its knowledge without ballooning its size and computation. We demonstrate the superiority of EFTs with a thorough slate of varied experiments, including on ImageNet-1K and CIFAR-100, demonstrating performance gains while also being efficient, without having to resort to saving data samples for replay. We also demonstrate the success on complex generative modeling tasks, demonstrating significant catastrophic forgetting mitigation. By demonstrating success and scalability in challenging settings, we hope to bring continual learning to more practical applications.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning What (not) to Forget. *European Conference on Computer Vision*, 2018. 1, 5, 14

[2] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. *European Conference on Computer Vision*, 2018. 5, 14

[3] Nadav Cohen and Amnon Shashua. Inductive Bias of Deep Convolutional Networks through Pooling Geometry. *International Conference on Learning Representations*, 2017. 2

[4] Yulai Cong, Miaoyun Zhao, Jianqiao Li, Sijia Wang, and Lawrence Carin. Gan memory with no forgetting. *Advances in Neural Information Processing Systems*, 33, 2020. 8

[5] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2019. 4

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. ImageNet: A Large-Scale Hierarchical Image Database. *Computer Vision and Pattern Recognition*, 2009. 2, 5, 6, 8, 12

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *North American Chapter of the Association for Computational Linguistics*, 2019. 2

[8] Qiang Gao, Zhipeng Luo, and Diego Klabjan. Efficient architecture search for continual learning. *arXiv preprint arXiv:2006.04027*, 2020. 4

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. *Neural information processing systems*, 2014. 8

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017. 4

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Computer Vision and Pattern Recognition*, 2016. 2, 5, 12, 13, 14

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Neural Information Processing Systems*, 2017. 2, 8

[13] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, pages 2554–2558, 1982. 5

[14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 3

[15] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines, 2019. 4

[16] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, Picking and Growing for Unforgetting Continual Learning. *Neural Information Processing Systems*, 2019. 5

[17] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016. 6

[18] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, 2017. 1, 2, 5, 6, 8, 14

[19] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009. 2, 5, 12

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 3, 5, 6, 12, 14

[21] Abhishek Kumar, Sunabha Chatterjee, and Piyush Rai. Nonparametric Bayesian Structure Adaptation for Continual Learning. *arXiv preprint arXiv:1912.03624*, 2019. 5

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[23] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A Neural Dirichlet Process Mixture Model for Task-Free Continual Learning. *International Conference on Learning Representations*, 2020. 5

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

[25] Zhizhong Li and Derek Hoiem. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 5, 6, 14

[26] Kevin J Liang, Chunyuan Li, Guoyin Wang, and Lawrence Carin. Generative Adversarial Network Training is a Continual Learning Problem. *arXiv preprint arXiv:1811.11083*, 2018. 8

[27] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. *Neural Information Processing Systems*, 2017. 1

[28] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018. 5, 7

[29] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018. 4, 5, 6

[30] Marc Masana, Tinne Tuytelaars, and Joost van de Weijer. Ternary feature masks: continual learning without any forgetting. *arXiv preprint arXiv:2001.08714*, 2020. 4, 5, 6, 12

[31] Michael McCloskey and Neal J Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *The Psychology of Learning and Motivation*, 1989. 1

[32] Nikhil Mehta, Kevin J Liang, Vinay Kumar Verma, and Lawrence Carin. Continual Learning using a Bayesian Nonparametric Dictionary of Weight Factors. *Artificial Intelligence and Statistics*, 2021. 1, 5

[33] George A Miller. *WordNet: An Electronic Lexical Database*. MIT press, 1998. 6, 12

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 1

[35] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 7

[36] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational Continual Learning. *International Conference on Learning Representations*, 2018. 1

[37] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual Lifelong Learning with Neural Networks: A Review. *Neural Networks*, 2019. 1, 4

[38] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. *arXiv preprint arXiv:2003.11652*, 2020. 1

[39] Roger Ratcliff. Connectionist Models of Recognition Memory: Constraints Imposed by Learning and Forgetting Functions. *Psychology Review*, 1990. 1

[40] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental Classifier and Representation Learning. *Computer Vision and Pattern Recognition*, 2017. 1

[41] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting. *Neural Information Processing Systems*, 2018. 1

[42] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience Replay for Continual Learning. *Neural Information Processing Systems*, 2019. 1

[43] 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. 1, 2, 4, 7

[44] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden Technical Debt in Machine Learning Systems. *Neural Information Processing Systems*, 2015. 1

[45] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557, 2018. 4, 5, 6

[46] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. *Neural Information Processing Systems*, 2017. 1

[47] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5, 6, 7, 12, 14

[48] Pravendra Singh, Vinay Kumar Verma, Pratik Mazumder, Lawrence Carin, and Piyush Rai. Calibrating cnns for lifelong learning. *Advances in Neural Information Processing Systems*, 33, 2020. 6, 13

[49] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P Namboodiri. Hetconv: Heterogeneous Kernel-Based Convolutions for Deep CNNs. *Computer Vision and Pattern Recognition*, 2019. 2

[50] Sakshi Varshney, Vinay Kumar Verma, Lawrence Carin, and Piyush Rai. Efficient continual adaptation for generative adversarial networks. *arXiv preprint arXiv:2103.04032*, 2021. 8

[51] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020. 4, 13

[52] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 2, 8

[53] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *ICLR*, 2020. 13, 14

[54] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33, 2020. 5, 6, 13, 14

[55] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, Bogdan Raducanu, et al. Memory Replay GANs:

Learning to Generate New Categories without forgetting. *Neural Information Processing Systems*, pages 5962–5972, 2018. 8

[56] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 2, 3

[57] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *Advances in Neural Information Processing Systems*, pages 899–908, 2018. 4, 7

[58] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and Order-robust Continual Learning with Additive Parameter Decomposition. *International Conference on Learning Representations*, abs/1902.09432, 2020. 1, 4, 5, 7

[59] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong Learning with Dynamically Expandable Networks. *International Conference on Learning Representations*, 2018. 1, 4, 7

[60] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *arXiv preprint arXiv:1506.03365*, 2015. 2, 8

[61] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6982–6991, 2020. 5, 14

[62] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. *International Conference on Machine Learning*, 2017. 1, 5, 13, 14

[63] Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. Lifelong GAN: Continual Learning for Conditional Image Generation. *International Conference on Computer Vision*, 2019. 8

[64] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks. *Transactions on Pattern Analysis and Machine Intelligence*, 2018. 5, 8, 13

[65] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-Incremental Learning via Deep Model Consolidation. *Winter Conference on Applications of Computer Vision*, 2020. 5, 14

[66] Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-tuning: Network adaptation via additive side networks. *arXiv preprint arXiv:1912.13503*, 2019. 4

[67] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing Tickets: Zeros, Signs, and the Supermask. *Advances in Neural Information Processing Systems*, 2019. 5

A. Implementation Details

We describe here the architectures, hyperparameters, and other implementation details used in our experiments.

A.1. Class Incremental Learning

CIFAR-100 [19] is a medium-scale dataset widely used for supervised classification. It contains 100 diverse classes that can grouped into 20 superclasses. We leverage the CIFAR-100 dataset for the class incremental learning setup. We perform experiments in three settings: CIFAR-100/10 which contains 10 task of 10 classes each, CIFAR-100/20 which contains 20 task of 5 classes each and CIFAR-100/5 which contains 5 task of 20 class each. CIFAR-100/20 tests the model's ability to resist catastrophic forgetting for a large number of task, while CIFAR-100/5 tests the model's ability when each task contains more classes. Interestingly, we observe that many models from the literature showing good results for a small number of tasks perform badly for a larger number of tasks. Thus, we find a large number of tasks to be a more reliable setting to test a model's continual learning ability.

We use the ResNet-18 [11] architecture for all experiments on the CIFAR dataset. ResNet-18 contains 11.69M parameters across 18 layers. The proposed *Efficient Feature Transformation* (EFT) module can be applied in two ways: *i*) Serial adaptation and *ii*) parallel adaptation (see Section 2.1.3). The added serial adapter on ResNet-18 architecture is shown in the Figure 6 (top).

We use the SGD optimizer for all experiments with a weight decay of 0.005. The ResNet-18 architecture is trained for 250 epochs for the first task and 200 epochs for subsequent tasks, as forward transfer from the first task leads to quicker convergence. We train the first task with an initial learning rate of 0.01, with $0.1\times$ learning rate decay at epochs 100, 150, and 200. For the second task onward, the same initial learning and decay rate are used, with decay steps occurring at epochs 70, 120 and 150. In all our experiments (CIFAR-100/10, CIFAR-100/20 and CIFAR-100/5), we set $\lambda = 0.05$. For hyperparameter tuning, 10% data was held out as validation data; for final training, we merge the training data and validation data. In Equation 4, $\gamma = 0$ indicates that the 1×1 filters are not used for the EFT transformation. We show a setting with such a case in Tables 6 and 8. In the class incremental learning scenario, task IDs are available during training, but they must be inferred at test time to determine the corresponding τ_t. During training, only samples of task t are present, while at test time, test samples may come from any of tasks 1 to t.

A.2. Task Incremental Learning

We perform experiments in the task incremental learning scenario on medium- and large-scale datasets with two standard architectures: VGG-16 [47] and AlexNet [20].

ImageNet-1K The ImageNet-1K [6] classification dataset contains 1000 classes based on the WordNet [33] hierarchy. In the continual learning setup, 1000 classes are divided into 10 tasks of 100 classes each. We use the SGD optimizer to optimize an AlexNet architecture. The model is trained for 90 epochs for ImageNet with a weight decay of 0.0005 for the first task and a weight decay of 0.00005 for subsequent tasks. An initial learning rate of 0.0001 is used with a step decay of $0.5\times$ at epochs 40, 60 and 80. EFT layers are appended after each layer.

Tiny-ImageNet-200 The Tiny-ImageNet-200 dataset is a subset of the ImageNet dataset and contains 200 classes at downsampled resolution. For continual learning, these 200 classes are divided into 10 tasks with 20 classes each. Since Tiny ImageNet has a lower resolution, the last maxpool layer and the last three convolutional layers from the feature extractor are removed from the standard VGG-16 architecture [30]. The VGG-16 model is trained for 160 epochs with an initial learning rate of 0.01 and a weight decay of 0.0005, for all tasks. The learning rate is decayed by a factor of $0.1\times$ at epochs 70, 100, and 120. EFT layers are appended as shown in Figure 6.

A.3. VGG-16 for Heterogeneous Datasets

Many previous approaches evaluate the model's performance in homogeneous settings: tasks strongly resemble the preceding ones. We evaluate the model on heterogeneous datasets, where later tasks are very different from previous ones, and the number of classes may change between tasks. In this challenging setup, we use the VGG-16 architecture with batch-norm to learn CIFAR-10, SVHN, and CIFAR-100 in sequence. We show results for two different task orders: CIFAR-100→CIFAR-10→SVHN and SVHN→CIFAR-10→CIFAR-100. We report the results in Table 3 as the final performance of the model on each of the datasets at the conclusion of the task sequence. The VGG-16 architecture is trained for 200 epochs with an initial learning rate of 0.01 and a weight decay of 0.0005. A learning rate decay of $0.1\times$ is used at steps 100, 150, and 170.

A.4. Parallel Adaptation

An alternative parallel adaptation strategy is proposed in Section 2.1.3. We show results for the parallel

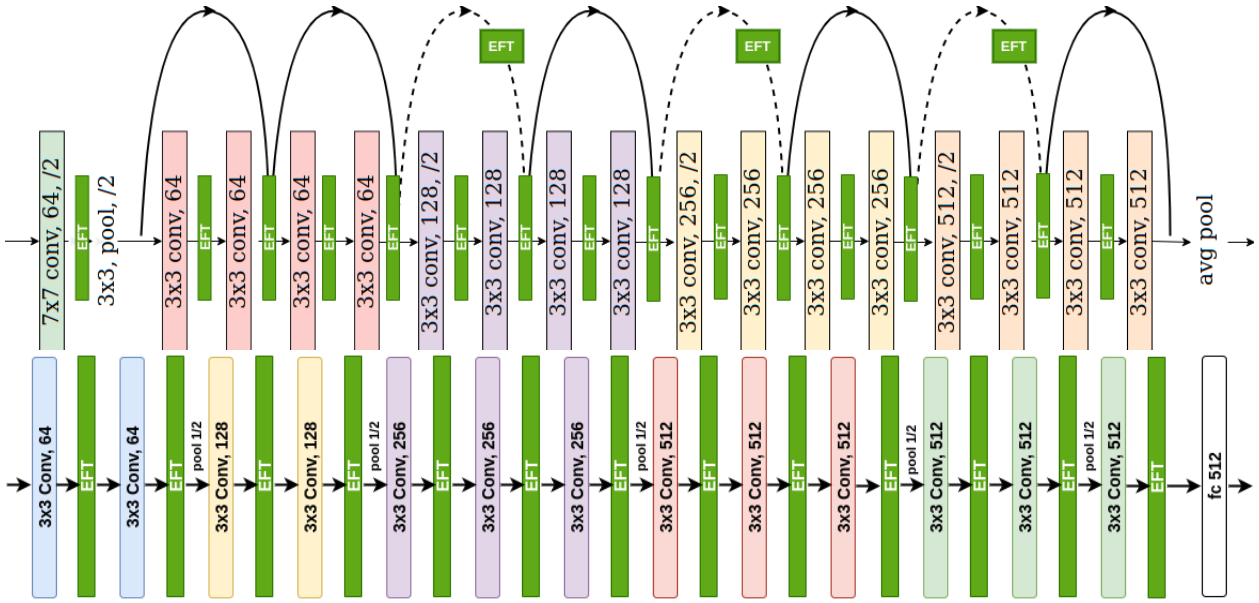

Figure 6. Overview of inserted transformations for ResNet-18. The added EFT layers in each architecture used for continual learning are shown in green. Except for the EFTs, all parameters remain unchanged between tasks.

adaptation strategy in Table 8. We observe that parallel adaptation with EFTs results in slightly inferior performance compared to serial adaptation. Note that for both parallel and serial adaptation, we use the same optimizer, learning rate, learning rate decay, and weight decay. Please refer to Section A.1 for more details.

A.5. StackGAN-v2 for Generative Modeling

Continual learning in generative models (*e.g.*, GANs) is a challenging problem rarely explored by recent literature. Existing methods tend to rely on replay-based approaches. In contrast, we use an expansion-based method to learn to generate successive datasets in a continual fashion. Specifically, we use StackGAN-v2 [64] as the base network and use EFT layers to continually expand the architecture for each novel task. We append EFT layers after each convolutional layer (serial adaptation) in both the generator and the discriminator, before batchnorm and ReLU activations are applied, resulting in 4.8% extra parameters per dataset. We use the default StackGAN-v2 hyperparameter values for learning rate and the number of epochs.

B. Others Comparisons

The model architectures used for continual learning tend to vary from paper to paper, so we did our best to compare our proposed approach with recent baselines in a fair, consistent setting. However, other papers occasionally report results with other setups. For example, continual learning performance on CIFAR is occasionally reported with CIFAR-10 being the base dataset,

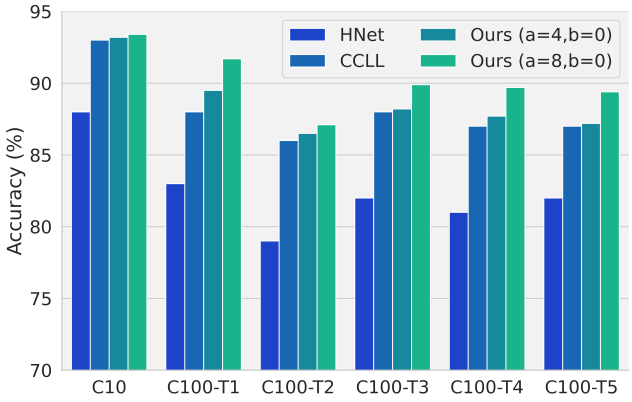

Figure 7. Comparison of the proposed model on the ResNet-32 architecture on the CIFAR10-CIFAR100 dataset.

with 5 subsequent tasks of 10 classes drawn from the CIFAR-100 dataset. For this setting, we follow prior work in using ResNet-32 [11], an architecture with a smaller number of parameters and FLOPs commonly used for CIFAR. We compare our proposed approach with the recent works HNet [51] and CCLL [48] in this setting. The results are shown in Figure 7. We use the same training procedure as discussed we did with the ResNet-18 architecture. We see stronger results with our EFT approach.

ResNet-18/3 [62], a standard ResNet-18 but with a third of the filters per layer, is another widely used architecture for continual learning. We compare our EFTs with the recently proposed SupSup [54] and BatchE [53] on CIFAR-100/20 in the task incremental setting, showing the results in Table 9. Again, we observe the pro-

Table 8. Average accuracy on CIFAR-100 in class incremental learning setting when trained on 10 tasks sequentially.

Dataset / #Tasks	Methods	1	2	3	4	5	6	7	8	9	Final
CIFAR-100/10	Finetune	88.5	47.1	32.1	24.9	20.3	17.5	15.4	13.5	12.5	11.4
	FixedRep	88.5	45.9	30.1	22.4	17.7	15.2	12.3	11.1	9.8	8.8
	LwF [25]	88.5	70.1	54.8	45.7	39.4	36.3	31.4	28.9	25.5	23.9
	EWC [18]	88.5	52.4	48.6	38.4	31.1	26.4	21.6	19.9	18.8	16.4
	EWC+SDC [61]	88.5	78.8	75.8	73.1	71.5	60.7	53.9	43.5	29.5	19.3
	SI [62]	88.5	52.9	40.7	33.6	31.8	29.4	27.5	25.6	24.7	23.3
	MAS [1]	88.5	42.1	36.4	35.1	32.5	25.7	21.0	19.2	17.7	15.4
	RWalk [2]	88.5	55.1	40.7	32.1	29.2	25.8	23.0	20.7	19.5	17.9
	DMC [65]	88.5	76.3	67.5	62.4	57.3	52.7	48.7	43.9	40.1	36.2
	EFT-a_4b_0 (+1.7%)	90.3	73.9	65.9	59.0	54.6	50.6	48.4	45.6	43.3	**41.2**
	EFT-a_4b_8 (+2.0%)	90.3	73.9	65.9	59.0	54.6	50.6	48.4	45.6	43.3	**41.5**
	EFT-a_8b_{16} (+3.9%)	90.3	74.2	66.5	60.4	55.8	51.5	49.5	46.7	44.7	**42.7**

posed model shows significant improvements compared to the baseline model.

Entry	Avg Acc@1
Upper Bound	91.62 ± 0.89
SupSup (GG) [54]	86.45 ± 0.61
SupSup (GG) Transfer [54]	88.52 ± 0.85
BatchE (GG) [53]	79.75 ± 1.00
Separate Heads	70.60 ± 1.40
EFT a_4b_0(+5.2%)	89.25 ± 0.31
EFT a_5b_0(+6.7%)	90.17 ± 0.47

Table 9. Task incremental learning accuracy on CIFAR-100/20 with the ResNet-18/3 [62] architecture.

C. Selection of Hyperparameter a and b

The choice of the hyperparameters a and b control both the model's representational power per task, as well as the growth in parameters and FLOPs. This leads to a trade-off: increasing the value of a and b will increase the model's performance at the cost of higher computation and memory (refer to Table-6 for ablation). Optimal values depend on the situation and can vary based on architecture. We observe that for ResNet [11] and VGG [47], we can use lower values a and b, leading to a parameter growth of 2-4%, while for the AlexNet [20] architecture, higher values of a and b only lead to 0.6% growth, as a significant proportion of the model parameters are in the fully connected layer. Using lower hyperparameter values for AlexNet results in a very small number of parameters per task, making it difficult for the model to adapt to novel tasks. Additionally, the proposed EFT for fully connected layers only adds a diagonal matrix for continual adaption, which adds only a negligible number of parameters. We observe that architectures with more parameters in fully connected layers (especially AlexNet) have a harder time adapting features with just diagonal matrices. Therefore, we find it advantageous to increase the number of convolutional layer parameters. We can observe this in Table 2 for the AlexNet architecture.