Leveraging low rank filters for efficient and knowledge-preserving lifelong learning

Muhammad Tayyab[1] and Abhijit Mahalanobis[2]

[1] University of Central Florida, Orlando FL 32816, USA
[2] Department of ECE, University of Arizona, Tucson, AZ

Abstract. We propose a low rank filter approximation based continual learning approach which decomposes convolution filters into compact basis filters and remixing coefficients. For lifelong learning, we keep the same basis filters to allow knowledge sharing, but add separate coefficients for each new task. Task specific feature maps are computed by a sequence of convolutions, first with shared basis filters and followed by the task specific coefficients. This method enables the model to preserve the previously learned knowledge, thus avoiding the problem of catastrophic forgetting. Additionally, choosing compact basis lets us get away with using a small number of basis filters which enables reduction in FLOPs and number of parameters in the model. To demonstrate efficiency of the proposed approach, we evaluate our model on a variety of datasets and network architectures. With Resnet18 based architecture, we report performance improvement on CIFAR100 with significantly low FLOPs and parameters as compared to other methods. For ImageNet our method achieves comparable performance to other recent methods with reduced FLOPs.

Keywords: Neural network compression · Continual Learning · Convolutional neural networks · Image classification

1 INTRODUCTION

Recent progress in machine learning research has led to impressive advances and has enabled many practical applications of the deep learning models [1, 2]. These conventional methods however assume that all training data is available at once which is hardly the case in real world applications. For instance, imagine a warehouse robot tasked with visually scanning items. If traditional training pipeline is used, every time a set of new items classes are added to the inventory the model will need to be trained again from scratch, with all previous and newly available data. This process is not only time consuming but also needlessly computationally expensive. Additionally, previous training data may not be available now. Naively training the model on new dataset alone makes the model encounter *catastrophic forgetting*, where the model forgets the previously learned representations while adapting to new data only. This limitation is one of the major barriers hindering the widespread adoption of deep learning methods.

This problem (referred to as continual learning or lifelong learning in the literature) is an active area of research. Lifelong learning aims to train a model incrementally, as

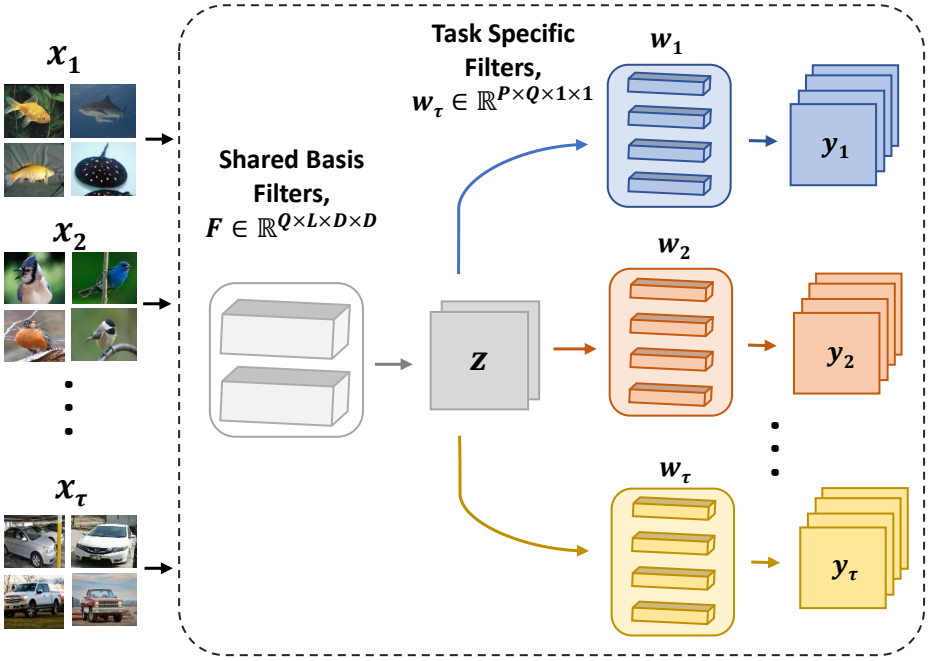


Fig. 1. Our Efficient Lifelong Learning (ELL) method. We substitute the original filters $\mathbf{H}$ with shared basis filters $\mathbf{F}$ and a set of task specific coefficients $\mathbf{w_t}$. Combination of these two lets us compute task specific features throughout the network.

new data becomes available while also preserving the previously learned representations. To solve this problem several methods have been introduced among them, network expansion based methods work have shown potential. These methods add dedicated parameters to the model for each new task which can be used to calculate the task specific feature maps, thus avoiding forgetting. For example Rusu et al. [3], Yoon et al. [4] and Jerfel et al., 2019 [5] proposed incrementally adding parameters to the model for each new task. Vinay et. al. [6] suggested adding several task specific feature map transformation layers to the model which add a small number of additional parameters. Similarly, Miao et. al. [7] proposed a low rank sub-space based approach which decompose original filters into a set of task specific atoms and shared coefficients. Task specific filters are obtained at inference time by multiplying the shared coefficients with corresponding filter atoms.

To address the problem of Life Long Learning, we propose a filter decomposition based approach. Our method enables knowledge sharing between tasks using shared basis filters while task specific coefficients enable the model to compute task specific feature maps. We leverage the low rank approximation of convolution filters $\mathbf{H}$ to decompose them into compact basis filters $\mathbf{F}$ and coefficients $\mathbf{w}$. We share the filters $\mathbf{F}$ among all tasks while new coefficients ($\mathbf{w}$) are added to the model for each addi-

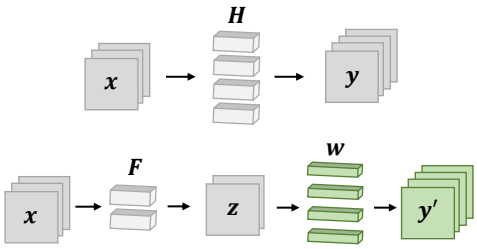


Fig. 2. Illustration of low rank filter decomposition. On top is the 2D convolution with filters $\mathbf{H}$, which can be decomposed into basis filters $\mathbf{F}$ and coefficients $\mathbf{w}$ as shown below. A convolution with $\mathbf{F}$ followed by $\mathbf{w}$ yields $\mathbf{y}'$ that approximates the original output $\mathbf{y}$.

tional task. In contrast to Miao et al. [7], task specific feature maps are computed by a sequence of convolutions with $\mathbf{F}$ followed by $\mathbf{w_t}$, as as depicted in Fig. 1. Finally choosing compact basis to represent $\mathbf{H}$ lets us get away with using a small number of basis filters. This enables significant reduction in FLOPs and number of parameters in the model.

To demonstrate efficiency of the proposed approach, we evaluate our model on a variety of datasets and network architectures. With Resnet18 [1] based architecture, we report performance improvement on CIFAR100 [8] with significantly low FLOPs and parameters as compared to other methods. While for ImageNet [9] our method achieves comparable performance to the recent methods.

2 RELATED WORK

2.1 LIFELONG LEARNING

A lot of research has been done on lifelong learning in recent years. We have followed Delange et al. [10] to divide these methods into following groups based on the way they tackle forgetting of the previously learned knowledge.

Network Expansion methods are most relevant to our proposed approach. These methods prevent catastrophic forgetting by adding dedicated parameters for each new task to calculate the task specific feature maps. However unconstrained parameter growth can very quickly overwhelm the memory resources, so the rate of parameter growth is a matter of concern here.

Rusu et al. [3], Yoon et al. [4] and Jerfel et al., 2019 [5] proposed incrementally adding parameters to the model. While Mallaya et al. [11] prunes the previous tasks parameters before introducing new task. Similarly, Wortsman et al., 2020 [12] presented a masking mechanism to train separate subnetwork for each task. Recently, Vinay et. al. [6] suggested adding several task specific feature map transformation layers to the model which add a small number of additional parameters. Miao et. al. [7] proposed a low rank sub-space based approach which decompose original filters into a set of task specific atoms and shared coefficients. Task specific filters are obtained at inference time by multiplying the shared coefficients with corresponding filter atoms.

Algorithm 1 Training procedure of the proposed method

TRAIN($\mathbf{H}$, k, $\mathcal{D}_1$ ... $\mathcal{D}_\mathcal{T}$)
- Train model using $\mathcal{D}_1$, acc. to Eq. 3
- Decompose $\mathbf{H}$ in each Conv2D layer into $\mathbf{F}$ and $\mathbf{w}$
- Truncate $k\%$ filters in $\mathbf{F}$ and update $\mathbf{w}$ accordingly
- **for** (task: $t = 1$ to $\mathcal{T}$) **do**
  - **if** ($t == 1$)
    - $\mathbf{w}_1 \leftarrow \mathbf{w}$
    - Finetune $\mathbf{F}$ and $\mathbf{w}_1$ using $\mathcal{D}_1$, acc. to Eq. 4
  - **else**
    - $\mathbf{w}_t \leftarrow \mathbf{w_1}$
    - Train $\mathbf{w}_t$ using $\mathcal{D}_t$, acc. to Eq. 5
  - **end if**
- **end for**
- **return** ($\mathbf{F}$, $\mathbf{w}_1$ .. $\mathbf{w}_\mathcal{T}$)

Replay Methods assign a small memory to store a subset of previous task samples or train a generator model to synthesise pseudo-samples. These samples are then used to train the model along with new task data to make sure that previously learned knowledge is retained. Storing samples from previous tasks however raises privacy concerns which is a drawback of these methods.

Shin et al. [13] proposed training a generative model to produce samples for previous tasks. Rebuffi et al. [14] stores a subset of exemplars per class, selected to best approximate class means in the learned feature space. Rolnick et al. [15] suggest a sampling strategy to limit size of the memory buffer. Yan et al. [16] proposed combining the network expansion with a memory buffer to store previous task samples.

Regularization-based methods avoid storing any samples and instead add a regularization term to the loss function meant to prevent the drift in previous task's loss landscape. These methods need to carefully balance the plasticity vs stability of the model to make sure the new information is ingested properly while also preventing catastrophic forgetting.

Li et al. [17] propose a knowledge distillation based technique, where previous task outputs as the soft labels to mitigate forgetting and transfer knowledge. Kirkpatrick et al. EWC [18] estimate the Fisher information matrix which is used to identify the important parameters for previous tasks. Training algorithm then selectively penalizes changes to these parameters. Similarly, Aljundi et al. MAS [19] suggest unsupervised importance estimation using gradient magnitude. Finally, Titsias et al. [20] introduced Bayesian functional approach which avoids forgetting by constructing an approximate posterior belief of previous tasks.

2.2 LOW RANK APPROXIMATION

Our work takes inspiration from class of techniques that rely on low-rank approximations to represent the convolution filters. Jaderberg et al. [21] replace the pretrained 4D filter tensor by a set of separable rank-1 filters and proposed to approximate the response of the original filters by minimizing a L2 norm. In Denton et al. [22], the authors

form clusters in the higher filter layers, and then approximate the clusters using a sum of the outer-products of separable 1D filters. Zhang et al. [23] have employed a similar linear combination architecture to ours, but they approximate the response of the filters at each layer. Finally Qiu et al. [24] also proposes an convolutional filter decomposition as a truncated expansion with prefixed basis filters.

3 METHOD

Our lifelong learning method has been inspired from low rank filter decomposition-based methods [21,22]. We show that this formulation can not only be used to compress the pretrained convolution layer but also naturally extends to adapt the model for lifelong learning problem setting. It has been observed by [22] that task parameters lie in low dimensional sub space of the convolutional filters. We have exploited this fact for CNN compression by substituting the trained convolutional filters, of the first task, with compact filter basis $\mathbf{F}$ and coefficients $\mathbf{w}$, as shown in Fig. 2. For lifelong learning problem setting the basis filters are shared (and are kept fixed) while separate coefficients are added and trained for each new task. The combination of shared basis filters and task specific coefficients enables the model to preserve previously learned knowledge and avoid forgetting altogether.

3.1 LOW RANK APPROXIMATIONS OF FILTERS

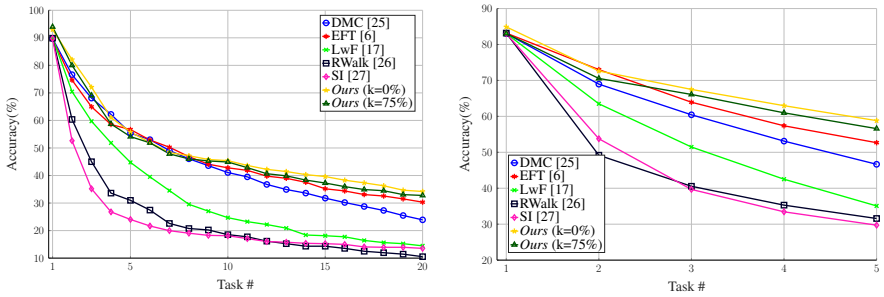
To find these compact basis filters and coefficients, consider the fundamental 2D convolution operation in a CNN. Assume that an input tensor x is convolved with a set of filters $\mathbf{H} \in \mathbb{R}^{P \times L \times D \times D}$, where *P* is the number of filters in H and with each filter of size $L \times D \times D$. The output y, of this convolution operation is expressed as

$$y = x * \mathbf{H} \tag{1}$$

We know that eigen decomposition results in a compact basis that minimizes the reconstruction error achieved by a linear combination of basis functions. We therefore choose $\mathbf{F}$ as the eigen filters that represent the sub-space in which the original filters $\mathbf{H}$ lie. The method for obtaining these is also well-known and straightforward. To approximate the compact sub-space these filters are flattened and arranged as columns of the matrix $\tilde{\mathbf{H}} \in \mathbb{R}^{P \times A}$, where $A = LD^2$. We then compute its singular value decomposition as $\tilde{\mathbf{H}} = \mathbf{USV}^T$ and initialize $\tilde{\mathbf{F}}$ with the columns of $\mathbf{V}^T$ corresponding to non zero eigen values. This leads to $\tilde{\mathbf{w}} = \tilde{\mathbf{F}}^T \tilde{\mathbf{H}}$ and finally appropriate reshaping of $\tilde{\mathbf{F}}$ and $\tilde{\mathbf{w}}$ lets us rewrite the convolution operation defined in Eq. 1 as a sequence of of two successive convolutions:

$$y' = (x * \mathbf{F}) * \mathbf{w} \tag{2}$$

where $\mathbf{F} \in \mathbb{R}^{P \times L \times D \times D}$ and $\mathbf{w} \in \mathbb{R}^{P \times P \times 1 \times 1}$. We refer to the construct introduced in Eq. 2 as *decomposed convolution*. This representation is typically used to compress the model in terms of FLOPs and number of parameters however in its current form it actually increases FLOPs and parameters. To compress the model we discard $k\%$ the basis filters in $\mathbf{F}$ corresponding to the smallest eigen values. This results in



(a) *20-split* CIFAR100 (b) *5-split* CIFAR100

Fig. 3. Comparison of CIL accuracy with Resnet18 architecture on CIFAR100 dataset.

Table 1. Comparison of CIL accuracy with Resnet18 architecture on *10-split* CIFAR100 dataset.

Method / Task ID	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	Avg.
LwF [17]	88.5	70.1	54.8	45.7	39.4	36.3	31.4	28.9	25.5	23.9	44.5
EWC [18]	88.5	52.4	48.6	38.4	31.1	26.4	21.6	19.9	18.8	16.4	36.2
SDC [28]	88.5	78.8	75.8	73.1	71.5	60.7	53.9	43.5	29.5	19.3	59.5
SI [27]	88.5	52.9	40.7	33.6	31.8	29.4	27.5	25.6	24.7	23.3	37.8
MAS [19]	88.5	42.1	36.4	35.1	32.5	25.7	21.0	19.2	17.7	15.4	33.4
RWalk [26]	88.5	55.1	40.7	32.1	29.2	25.8	23.0	20.7	19.5	17.9	35.3
DMC [25]	88.5	76.3	67.5	62.4	57.3	52.7	48.7	43.9	40.1	36.2	57.4
EFT [6]	90.2	76.2	70.1	63.1	57.9	53.6	52.1	49.6	47.6	45.5	60.6
Ours (k=0%)	92.4	76.3	67.1	63.6	62.0	58.6	56.4	54.3	52.1	49.8	**63.3**
Ours (k=75%)	89.1	74.7	65.1	61.6	59.8	55.6	53.7	51.4	49.4	47.3	60.8

$\mathbf{F} \in \mathbb{R}^{Q\times L\times D\times D}$ and correspondingly $\mathbf{w} \in \mathbb{R}^{P\times Q\times 1\times 1}$, where $Q = \lceil P - kP/100 \rceil$. Analysis of FLOPs and parameters required for implementing Eq. 2 is presented in Sec. 3.3.

3.2 EFFICIENT LIFELONG LEARNING

The objective of lifelong learning algorithm is to incrementally train the model on a set of disjoint classes with some data availability constraints. Formally, the model observes data $(\mathcal{X}_t, \mathcal{Y}_t)$ randomly drawn from distribution $\mathcal{D}_t$, corresponding to task t. Where $\mathcal{X}_t$ and $\mathcal{Y}_t$ represents the input images and corresponding ground truth labels respectively. The goal of a continual learning algorithm is to train model on $\mathcal{D}_t$ while also preserving the previously learned knowledge. However, this data availability constraint introduces the problem of catastrophic forgetting [17,18] where model *forgets* the previous knowledge and ends up performing poorly on previous tasks.

We seek to solve this catastrophic forgetting problem by using the *decomposed convolution* structure. We initially train first task on a vanilla neural network with convolution filters $\mathbf{H}$, given the data samples $(\mathcal{X}_1, \mathcal{Y}_1)$.

$$\underset{\mathbf{H}}{\operatorname{argmin}} \sum_{\mathcal{D}_1} \ell(f(\mathcal{X}_1; \mathbf{H}), \mathcal{Y}_1) \tag{3}$$

Table 2. Comparison of TIL accuracy with AlexNet architecture on *10-split* ImageNet.

Method / Task ID	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	Avg.
LwF [17]	27.6	37.2	42.0	44.4	50.5	56.6	57.9	61.2	62.0	62.7	50.2
IMM [29]	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 [11]	67.5	65.8	62.2	58.4	58.6	58.7	56.0	56.5	54.1	53.6	59.1
EFT [6]	69.0	63.2	60.1	62.5	53.6	57.2	55.1	52.8	55.7	62.5	59.4
Ours (k=0%)	65.6	63.3	60.7	63.9	56.5	57.4	55.0	52.8	55.7	64.1	59.5
Ours (k=25%)	65.0	63.0	59.7	62.8	55.5	56.3	54.5	52.4	55.0	63.5	58.8

Table 3. Comparison of TIL accuracy with VGG16 architecture on *10-split* TinyImageNet.

Method / Task ID	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	Avg.
LfL [30]	32.4	35.4	43.4	44.1	45.0	55.9	49.4	51.1	58.6	61.4	47.7
LwF [17]	45.1	45.5	53.5	57.6	56.2	65.7	63.5	58.4	59.6	58.5	56.4
IMM [29]	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 [31]	46.8	49.1	55.8	58.0	53.7	61.0	58.7	54.0	54.6	50.3	54.2
PackNet [11]	52.5	49.7	56.5	59.8	55.0	64.7	61.7	55.9	55.2	52.5	56.4
TFM [32]	48.2	47.7	56.7	58.2	54.8	62.2	61.5	57.3	58.5	54.8	56.0
EFT [6]	67.2	62.5	69.4	62.6	68.3	69.6	59.0	67.8	71.5	70.1	66.8
Ours (k=0%)	64.3	64.6	70.8	65.9	68.0	70.6	60.2	69.4	71.2	69.0	67.4
Ours (k=50%)	64.2	64.6	69.1	64.0	68.4	68.7	60.4	67.5	72.1	69.7	66.9

Where $f(:)$ represents model function and l is the loss function. Once the model is trained we decompose and compress the trained filters as described in Sec. 3.1. Compact nature of basis filters allows us to discard up to 25% of the total filters corresponding to the smallest eigen values, without an adverse affect on model's performance. However with even higher levels of compression model encounters performance degradation. To counter that we fine tune $\mathbf{F}$ and $\mathbf{w_1}$ to minimize the same loss as before.

$$\underset{\mathbf{F},\mathbf{w_1}}{\operatorname{argmin}} \sum_{\mathcal{D}_1} \ell(f(\mathcal{X}_1; \mathbf{F}, \mathbf{w_1}), \mathcal{Y}_1) \tag{4}$$

It is important to note that initial training and fine tuning is done using $\mathcal{D}_1$ only. For each new task $t > 1$, we add additional task specific coefficients $\mathbf{w_t}$ and initialize them with $\mathbf{w_1}$. This is followed by training the model using $\mathcal{D}_t$ as follows.

$$\underset{w_t}{\operatorname{argmin}} \sum_{\mathcal{D}_t} \ell(f(\mathcal{X}_t; w_t), \mathcal{Y}_t) \tag{5}$$

To compute the task specific feature maps, we first convolve the input tensor with the shared basis filters followed by a convolution with corresponding task specific coefficients, as depicted in Fig. 1. Since shared basis filters are only trained once, this formulation allows us to preserve the learned task specific representations perfectly through out the model and thus avoids the catastrophic forgetting problem entirely.

Table 4. Average CIL with Resnet32 architecture on non uniform *6-split* and *11-split* CIFAR100. Dataset is divided into tasks such that first task contains 50 classes and remaining are equally divided into 5 or 10 sets.

Method	*6-split*	*11-split*
LwF [17]	57.03	56.82
EWC [18]	56.28	55.41
iCaRL [14]	57.17	52.57
SDC [28]	57.10	56.80
BiC [33]	59.36	54.20
Rebalancing [34]	63.12	60.14
FAS-a [7]	60.23	55.54
FAS-b [7]	65.44	62.48
Ours (k=0%)	66.65	62.54
Ours (k=25%)	62.73	59.23

Table 5. Average TIL accuracy and task prediction accuracy for Resnet32, trained on non-uniform split CIFAR100.

Method	*6-split*	*11-split*
TIL (K=0%)	88.6	93.4
TIL (K=25%)	87.5	93.0
Task Prediction (K=0%)	79.1	74.5
Task Prediction (K=25%)	76.7	73.2

3.3 FLOPS AND PARAMETER GROWTH

In this section we present a theoretical analysis of the impact of our method on FLOPs and parameter growth in the model. Depending on the values of P and Q, the proposed compression scheme lead to substantial reduction in the number of multiplication operations. If the size of the filters is $D \times D \times L$, and the size of the input data is $M \times N \times L$, It is easy to show that $O = LD^2(M - D + 1)(N - D + 1)$. Therefore, multiplications required in Eq. (1) is

$$A = PO = PLD^2(M - D + 1)(N - D + 1) \tag{6}$$

while multiplications required to obtain output of Eq (2) is

$$\begin{aligned} B &= QLD^2(M - D + 1)(N - D + 1) + \\ &\quad PQ(M - D + 1)(N - D + 1) \\ &= Q[L^2 + P](M - D + 1)(N - D + 1) \end{aligned} \tag{7}$$

We see that the ratio of the two is

$$\begin{aligned} \frac{A}{B} &= \frac{PLD^2(M - D + 1)(N - D + 1)}{Q[LD^2 + P](M - D + 1)(N - D + 1)} \\ &= \frac{PLD^2}{Q[LD^2 + P]} \end{aligned} \tag{8}$$

Thus, as long as $LD^2 >> P$, the number of multiplications will be reduced by a factor close to P/Q (i.e. the ratio of the the original number of filters and the number of basis filters used).

New recall that number of parameters in the original filters is LD^2. Since there are P such filters, the total number of original parameters is PLD^2. However, the total number of parameters for *decomposed convolution* is $(QLD^2 + QP)$ (depicted in Fig.

1 as a Q basis filters with LD^2 parameters and P one-dimensional filters of length Q). Therefore, the reduction in the number of parameters is $PLD^2/Q(LD^2 + P)$. If $LD^2 >> P$, the number of parameters is also reduced by a factor of P/Q. Finally since each new task adds additional coefficients $\mathbf{w_t}$, so parameters growth is proportional to PQ.

4 EXPERIMENTS

We evaluated our method on two lifelong learning scenarios; Task Incremental Learning (TIL) and Class Incremental Learning (CIL). These two scenarios differ in the manner in with new task is treated at inference time. In TIL it is assumed that the task ID is known at inference time which can be used to select the corresponding $\mathbf{w_t}$ to calculate task specific representations. In CIL task ID is not provided and has be to predicted at inference time. We adopted a simple entropy based strategy to predict the task ID, where task ID of an unknown sample is chosen to be the ID of classification head with least entropy.

We also evaluated our method for compression (in Sec. 3.3) by calculating the FLOPs and parameters required for a given model. We opted FLOPs over wall clock time as a measure of model's efficiency, since FLOPs are machine and implementation independent and can be easily estimated in Pytorch.

To train the models, we used SGD optimizer with momentum 0.9, weight decay of 5E-4 and starting learning rate of 0.1. We divided the learning rate by 10 at 100, 150 and 200 epoch mark and train the model for 250 epochs with a batch size of 64.

4.1 CLASS INCREMENTAL LEARNING (CIL)

Resnet18: To evaluate our method on CIL we first adopted Resnet18 network architecture and train it on three uniform splits of CIFAR100. Specifically we divide the 100 classes into 5, 10 and 20 sets of tasks and refer to them as *5-split*, *10-split* and *20-split respectively*. These splits cover a wide range of problem difficulty as *20-split* CIFAR100 tests models ability to adapt to large number of new tasks while 5-split setting tests model's ability to train on a larger number of classes per task. For each one of these splits we trained Resenet18 at two compression levels with value of k set to 0% and 75% (k is the percentage of basis filters discarded). Results of these experiments are summarised in Tab. 1 and Fig.3. We notice that our method outperform others even with very high levels of compression. When compared with EFT [6] our Resnet18 with $k = 75\%$ performs slightly better with 60.8% average CIL accuracy as compared to EFT's 60.6%. Additionally truncation of large number of basis filters significantly reduces the FLOPs and number of parameters in the model (detailed comparison in Sec. 4.5).

Resnet32: For evaluating Resnet32 on CIFAR100 where followed the FAS's [7] non uniform split. This split is different form the previous ones because here the first tasks always contains 50 classes while the remaining 50 are divided into sets of 5 or 10. This experiment evaluates a significantly different and harder problem as now we have very uneven number of classes per task. As we can see in Tab. 4 our method with $k = 0$

Table 6. Comparison of average test accuracy and forgetting for *20-split* CIFAR100 and *20-split* miniImageNet datasets using ResNet18 architecture.

Method	*20-split* CIFAR100		*20-split* miniImageNet	
	Accuracy	Forgetting	Accuracy	Forgetting
EWC [18]	43.2	26	34.8	24
ICARL [14]	46.4	16	44.2	24.7
AGEM [35]	60.3	11.0	42.3	17.0
ER-Ring [36]	59.6	0.1	49.8	12.0
Ortho sub [37]	63.4	8.4	51.4	10.0
Adam-NSCL [38]	74.3	9.5	57.9	13.4
IBP-WF [39]	68.3	0	55.8	0
ITLIR [40]	68.5	0	59.3	0
Ours (k=0%)	**91.3**	**0**	**85.4**	**0**
Ours (k=95%)	87.4	0	79.8	0
Parallel full-rank	92.7	0	94.5	0

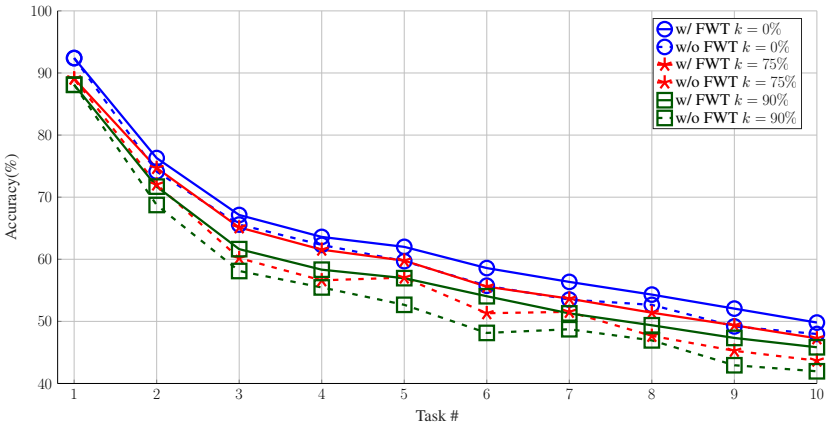


Fig. 4. Evaluation of Forward Knowledge Transfer (FWT). We conducted two experiments for three values of k (0%, 75% and 90%). In one set of experiments we initialize task specific coefficients w_i randomly (w/o FWT) while for the other one we initialized it with w_{i-1} (w/ FWT). As we can see, models using w_{i-1} initialization perform better then the ones using random initialization

out performs other methods. However due to non-uniformity of the splits our method does not perform equally well with high compression. This performance degradation mainly comes from the failure of task prediction mechanism as shown in Tab. 5.

4.2 TASK INCREMENTAL LEARNING (TIL)

We tested our method on TIL with AlexNet and VGG16 architectures. We trained these models on uniform *10-split* ImageNet [9] and TinyImageNet datasets respectively. ImageNet is a classification dataset containing 1000 classes, while TinyImageNet is a

Table 7. Ablation on k for CIL with Resnet18 architecture on *10-split* CIFAR100

k / Task ID	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	Avg.
0%	92.4	76.3	67.1	63.6	62.0	58.6	56.4	54.3	52.1	49.8	63.3
25%	90.4	76.3	65.7	61.75	61.0	58.2	55.9	53.3	50.9	48.9	62.2
75%	89.1	74.7	65.1	61.6	59.8	55.6	53.7	51.4	49.4	47.3	60.8
90%	88.1	71.6	61.6	58.3	56.9	54.0	51.2	49.3	47.3	45.8	58.4

smaller subset of ImageNet dataset containing 200 classes, downsampled to 64×64 spatial resolution. The results of these experiments are shown in the Tab. 2 and Tab. 3. We can see that, for both ImageNet and TinyImageNet our uncompressed models ($k = 0\%$) achieves better average accuracy compared to the other methods. While the compressed variants attain results comparable to the previous SOTA.

For TIL we also evaluated our method on Resnet18 architecture with *20-split* CIFAR100 and miniImageNet datasets. For this set of experiments, along with the average accuracy we also report the model's *forgetting*. Forgetting is defined in literature [37] as follows. Let $a_{t,j}$ be the test accuracy of task $j < t$ after the model has finished learning task $t \in \{1, ..., T\}$ in a incremental manner. Forgetting F_t is the decrease in the accuracy of a task after its training, and after one or several tasks are learned incrementally and is defined as $F_t = \frac{1}{t-1} \sum_{j=1}^{t-1} (a_{j,j} - a_{t,j})$. Table 6 summarizes the results of these experiments. Our method demonstrates a substantial improvement in comparison to compared approaches. Of particular interest is [40], which employs a filter decomposition structure similar to ours. However, our method diverges by compressing the model subsequent to training on the first task, rather than truncating randomly initialized filters. As a result, our shared basis filters learn more meaningful representations for continual learning, thereby improving the overall model's accuracy.

4.3 FORWARD TRANSFER (FWT)

Forward transfer (FWT) is an important metric of the quality of representations learned by a lifelong learning system. It measures the ability of the model to positively influence a future task's performance based on the existing representations. FWT is formally defined in Equation 4 by Lopez-Paz et. al [41]. However we followed the experimental setup proposed in EFT [6] to informally present this metric. We argue that our model enables FWT by two mechanisms, first by sharing basis filters among all tasks and second, by initializing w_i with previously trained w_{i-1}. To empirically show this, we conducted two experiments for three values of k (0%, 75% and 90%). In one set of experiments we initialize task specific coefficients w_i randomly (w/o FWT) while for the other one we initialized it with w_{i-1} (w/ FWT). As we can see in the Figure 4, models using w_{i-1} initialization perform better then the ones using random initialization. Secondly the gap between these curves increase for larger values of k. Form these experiments we deduce that our model should get better at FWT with increasing compression.

4.4 SELECTION OF k

Value of k impact the number of parameters in the model which consequently impacts the model's accuracy. For all experiments we have reported results for an uncompressed model (with $k = 0\%$) and we empirically select another value of k which is competitive with other methods in terms of accuracy. Table 7 presents additional results with Resnet18 for different values of k.

4.5 ANALYSIS OF FLOPS AND PARAMETERS

In this section we have compared the FLOPs and parameters of various network architectures used in our experiments. For our method, number of total tasks does not impact the FLOPs of a single input image. As can be deduced from Eq. 2, output at each layer is obtained by convolving the input tensor with the shared basis filters followed by the convolution with a single task specific $\mathbf{w_t}$. However as more tasks are added to the model, additional coefficients and classification heads increase the number of parameters in the model drastically. For our comparisons we picked the Resnet18, VGG16 and ALexNet to contain 10 task specific $\mathbf{w_t}$'s in each layer. Tab. 8 show the FLOPs and parameters for 4 levels of compression.

We can see that our Resnet18 trained for CIL, with $k = 75\%$ needs 0.32 Billion FLOPs and 7.24 Million parameters. In comparison EFT [6] performs equally well but needs 1.21 Billion FLOPs and 11.60 Million parameters. Similarly our Resnet32 with $k = 0\%$ outperforms FAS-b [7] on non-uniform split CIFAR100 and needs only 0.15 Billion FLOPs as compared to 0.28 Billion required by FAS-b [7].

Table 8. Comparison of FLOPs and parameters for various models used in our experiments.

k	**FLOPs (Billions)**				**Params (Millions)**			
	Resnet18	*Resnet32*	*VGG16*	*AlexNet*	*Resnet18*	*Resnet32*	*VGG16*	*AlexNet**
Baseline	1.11	0.14	2.50	1.74	11.16	0.46	14.76	2.88
0%	1.25	0.15	2.84	2.00	26.84	0.80	33.24	9.77
25%	0.95	0.12	2.10	1.50	20.61	0.60	25.05	8.36
50%	0.64	0.08	1.43	1.00	14.38	0.41	16.86	6.94
75%	0.32	0.04	0.72	0.48	7.24	0.21	8.66	5.52

5 CONCLUSION

We propose a method for continual learning which leverages ideas from neural network compression and applies them to incrementally learn new tasks. Our method takes inspiration from filter decomposition-based methods and we show that this formulation naturally extends to the continual learning domain. It is well known that task parameters lie in low dimensional sub space of the convolutional filters, this fact can be exploited to represent the convolutional filters in a compact form by using the basis filters and

remixing coefficients. For continual learning problem setting the basis filters are shared while separate coefficients are added for each new task. This enables the model to perfectly preserve the previously learned knowledge, thus avoiding the catastrophic forgetting entirely. We applied this method to several image classification based continual learning problems and show that our method obtains performance gains with uncompressed model while achieving competitive results when a large number of basis filters are discarded.

References

1. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016, IEEE Computer Society.
2. Joseph Redmon and Ali Farhadi, "Yolov3: An incremental improvement," *https://arxiv.org/abs/1804.02767*, 2018.
3. 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.
4. Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang, "Lifelong learning with dynamically expandable networks," in *International Conference on Learning Representations*, 2018.
5. Ghassen Jerfel, Erin Grant, Tom Griffiths, and Katherine A Heller, "Reconciling meta-learning and continual learning with online mixtures of tasks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
6. Vinay Kumar Verma, Kevin J. Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin, "Efficient feature transformations for discriminative and generative continual learning," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 13860–13870, 3 2021.
7. Zichen Miao, Ze Wang, Wei Chen, and Qiang Qiu, "Continual learning with filter atom swapping," in *International Conference on Learning Representations*, 2022.
8. Alex Krizhevsky and G Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.
9. Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.
10. 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*, pp. 1–1, 2021.
11. Arun Mallya and Svetlana Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018.
12. Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi, "Supermasks in superposition," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15173–15184, 2020.
13. Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim, "Continual learning with deep generative replay," *Advances in neural information processing systems*, vol. 30, 2017.

14. Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, G. Sperl, and Christoph H. Lampert, "icarl: Incremental classifier and representation learning," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5533–5542, 2017.
15. David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne, "Experience replay for continual learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
16. Shipeng Yan, Jiangwei Xie, and Xuming He, "Der: Dynamically expandable representation for class incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3014–3023.
17. Zhizhong Li and Derek Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 2935–2947, 2018.
18. James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al., "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
19. Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *The European Conference on Computer Vision (ECCV)*, September 2018.
20. Michalis K. Titsias, Jonathan Schwarz, Alexander G. de G. Matthews, Razvan Pascanu, and Yee Whye Teh, "Functional regularisation for continual learning with gaussian processes," in *International Conference on Learning Representations*, 2020.
21. Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *BMVC 2014 - Proceedings of the British Machine Vision Conference 2014*. 2014, British Machine Vision Association, BMVA.
22. Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," *Advances in Neural Information Processing Systems*, 2014.
23. Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.
24. Qiang Qiu, Xiuyuan Cheng, Robert Calderbank, and Guillermo Sapiro, "Dcfnet: Deep neural network with decomposed convolutional filters," *35th International Conference on Machine Learning, ICML 2018*, 2018.
25. 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," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1131–1140.
26. Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 532–547.
27. Friedemann Zenke, Ben Poole, and Surya Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3987–3995.
28. 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," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6980–6989, 2020.
29. Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang, "Overcoming catastrophic forgetting by incremental moment matching," *Advances in neural information processing systems*, vol. 30, 2017.

30. Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim, "Less-forgetting learning in deep neural networks," *ArXiv*, vol. abs/1607.00122, 2016.
31. Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4548–4557.
32. Marc Masana, Tinne Tuytelaars, and Joost van de Weijer, "Ternary feature masks: continual learning without any forgetting," *arXiv preprint arXiv:2001.08714*, vol. 4, no. 5, pp. 6, 2020.
33. Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu, "Large scale incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 374–382.
34. Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin, "Learning a unified classifier incrementally via rebalancing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 831–839.
35. Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny, "Efficient lifelong learning with a-gem," in *International Conference on Learning Representations*, 2019.
36. Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato, "On tiny episodic memories in continual learning," *arXiv preprint arXiv:1902.10486*, 2019.
37. Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr, "Continual learning in low-rank orthogonal subspaces," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9900–9911, 2020.
38. Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu, "Training networks in null space of feature covariance for continual learning," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2021, pp. 184–193.
39. Nikhil Mehta, Kevin Liang, Vinay Kumar Verma, and Lawrence Carin, "Continual learning using a bayesian nonparametric dictionary of weight factors," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 100–108.
40. Rakib Hyder, Ken Shao, Boyu Hou, Panos Markopoulos, Ashley Prater-Bennette, and M Salman Asif, "Incremental task learning with incremental rank updates," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII*. Springer, 2022, pp. 566–582.
41. David Lopez-Paz and Marc'Aurelio Ranzato, "Gradient episodic memory for continual learning," *Advances in neural information processing systems*, vol. 30, 2017.