

Weighted Ensemble Models Are Strong Continual Learners

Imad Eddine Marouf¹

Subhankar Roy²

Enzo Tartaglione¹

Stéphane Lathuilière¹

¹LTCI, Télécom-Paris, Institut Polytechnique de Paris ²University of Aberdeen

imad.marouf@ip-paris.fr

Abstract

In this work we study the problem of continual learning (CL) where the goal is to learn a model on a sequence of tasks, such that the data from the previous tasks becomes unavailable while learning on the current task data. CL is essentially a balancing act between being able to learn on the new task (i.e plasticity) and maintaining the performance on the previously learned concepts (i.e stability). With an aim to address the stability-plasticity trade-off we propose to perform weight-ensembling of the model parameters of the previous and current task. This weight-enssembled model, which we call Continual Model Averaging (or CoMA), attains high accuracy on the current task by leveraging plasticity, while not deviating too far from the previous weight configuration, ensuring stability. We also propose an improved variant of CoMA, named Continual Fisher-weighted Model Averaging (or CoFiMA), that selectively weighs each parameter in the weight ensemble by leveraging the Fisher information of the weights of the model. Both the variants are conceptually simple, easy to implement and effective in attaining state-of-the-art performance on several standard CL benchmarks.¹

1. Introduction

Deep learning models have achieved impressive results in many tasks such as image classification [11, 20], object detection [64] and semantic segmentation [27]. These advances are largely due to training with extensive offline datasets [8, 54] for a finite set of categories. Contrary to these controlled conditions, real-world applications often involve continuous learning from data streams that introduce novel categories [17, 19]. This process typically results in *catastrophic forgetting* (CF) [17], where the acquisition of new information leads to the erosion of the previously learned knowledge. Continual Learning (CL)

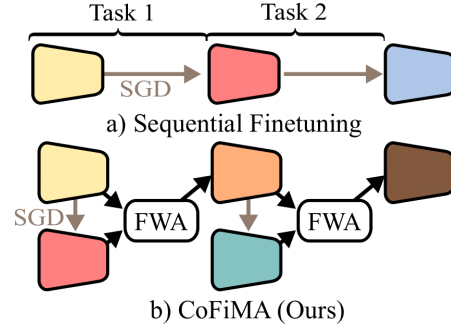


Figure 1. Comparison of CoFiMA with sequential fine-tuning in Continual Learning. In sequential finetuning, the model from the previous task serves as the starting point for the current task. Conversely, CoFiMA employs an averaging strategy: the model from the previous task $t-1$ is merged with the one fine-tuned on the current task t , using a Fisher-weighted averaging (FWA) technique.

emerges as a solution, allowing models to assimilate new classes from the temporal data stores while retaining classification capabilities for the previously learned classes using a unified classifier [37].

In prior work on CL, the common approach is to train the networks (usually ResNets [20]) from scratch on the current task while regularizing them to counteract forgetting [37, 70]. The emergence of pre-trained models (PTMs) [11, 13, 50] (i.e Vision Transformer (ViT) [11, 36]), trained on large datasets such as ImageNet [54], has provided an effective alternative for CL tasks [66, 71–73, 84]. PTM-based CL methods [72, 73, 82] enhance data efficiency and consistently excel over traditional CL methods. In particular, Zhang *et al.* [82] showed that a simple solution such as sequential fine-tuning using PTMs yields competitive results, outperforming modern CL approaches involving prompting [72, 73] and the traditional ones using regularization [28, 33]. This is further validated in recent works [12, 40], where it has been shown that PTMs, owing to their strong and diverse pre-training, are inherently less susceptible to catastrophic forgetting.

Despite the significant advancements in CL with PTMs, there remains a challenge in achieving an optimal balance

¹Code is available: <https://github.com/IemProg/CoFiMA>

between retaining knowledge from previous tasks and excelling in current tasks, referred to as *stability-plasticity* dilemma [1, 26, 42]. Current methods often face trade-offs between these objectives, leading to either partial forgetting of old tasks or having sub-optimal performance on new ones. In search of finding this optimal balance, we look at CL through the lens of model averaging [51, 74, 75, 79]. Specifically, WiSE-FT [75], a weight-space ensembling approach, that linearly interpolates between the weights of the PTM (e.g., zero-shot CLIP) and the PTM fine-tuned on the target distribution. As a consequence, WiSE-FT achieves high accuracy on the target distribution, while maintaining good performance on datasets with distributional shifts (a setting where zero-shot CLIP would normally excel).

Motivated by the findings of WiSE-FT, we cast CL as a transfer learning approach where we aim to linearly combine the weights from a previous task with the model weights from the current task (see Fig. 1). This endows us with improved plasticity on the current task, while maintaining stability on past tasks. The underpinning idea of our proposed method, **Continual Model Averaging (CoMA)**, also resonates with the work by Mizraheh *et al.* [41], that compares minima resulting from multitask learning and continual learning, establishing that minima from continual learning are linearly connected to optimal sequential multi-task minima but not to each other, leading to forgetting.

The model averaging in CoMA assumes that all the weights of the network have same importance for a given task. While effective, putting equal importance to all weights could result in a weight-ensembled model that lies in an error basin with high loss [74]. To mitigate this problem, we aim to selectively ensemble the weights based on the importance of each weight parameter for a given task. Inspired by elastic weight consolidation (EWC) [28], we leverage the Fisher information [10, 59] to weigh the model parameters during model averaging. Fisher information inherently captures the importance of each weight parameter on the dataset (or task) the model has been trained on. We call this variant of CoMA as **Continual Fisher-weighted Model Averaging (or CoFiMA)**. While this variant requires storing the Fisher information matrix after training on each task, it is computationally lightweight as it requires a single forward and backward pass on the current task data, and yields state-of-the-art performance surpassing both CoMA and existing PTM-based CL solutions.

Our **contributions** are summarised as follows. (i) We propose CoMA, a weight-ensemble inspired CL approach, that addresses the challenging task of stability-plasticity trade-off in CL. (ii) We extend CoMA by employing Fisher information to weight the averaging of parameters (i.e CoFiMA). Fisher information weighting enables us to determine which parameters contain important “information” to previous tasks, and which can be changed freely. (iii) We

run extensive experiments on CL benchmarks and demonstrate that CoFiMA despite being simple consistently outperforms PTM-based CL solutions.

2. Related Work

Continual Learning with PTM. Until now, the predominant focus in continual learning has been on the sequential training of deep neural networks from scratch, aiming to proficiently acquire new tasks while mitigating significant forgetting of preceding tasks. Noteworthy strategies encompass regularization-based approaches [2, 9, 28, 34, 80], which maintain the initial model and selectively stabilize parameter or prediction alterations; replay-based approaches [4, 49, 69, 77], that seek to approximate and regenerate previously learned data distributions; and architecture-based approaches [55, 56, 78], which allocate discrete parameter sub-spaces for each incrementally introduced task.

The recent trajectory of research has further probed into the advantages of pre-training and using PTMs within the CL framework [12, 72, 73]. Representations derived from supervised pre-training have demonstrated the capacity to facilitate not only knowledge transfer but also resilience against catastrophic forgetting during downstream continual learning [39, 52]. Moreover, learning a substantial base of classes during the initial training phase permits CL with minimal adaptations [76]. L2P [73] leveraged techniques inspired by pre-trained knowledge utilization in NLP, employing an additional set of learnable parameters, termed “prompts”, which dynamically guide a pre-trained representation layer in learning incremental tasks. Dual-Prompt [72] elaborated on this concept by attaching supplementary prompts to the pre-trained representation layer to facilitate the learning of both task-invariant and task-specific instructions. Though prompt-based approaches have been documented to significantly outperform conventional continual learning baselines, they introduce an extra inference cost. Recently, Zhang *et al.* [82] show that sequential fine-tuning with small learning rate using PTMs outperforms traditional CL approaches.

Unlike prior approaches in CL that enhance PTMs using prompts [72, 73] or replay-buffers [4, 49, 69, 77], CoFiMA employs a different strategy. It reduces forgetting during training by averaging the weights of parameters from previous models. In contrast to EWC [28], which uses the Fisher Information Matrix (FIM) as a regularization term for L2-transfer between tasks, CoFiMA applies the FIM as a weighting factor to assess the significance of weights for each task.

Output-space/weight-space ensembles. Traditional ensemble methods, or output-space ensembles, combine multiple classifiers’ predictions, often outperforming single models and providing more calibrated uncertainty estimates under distribution shifts [18, 32, 46, 61]. These output-

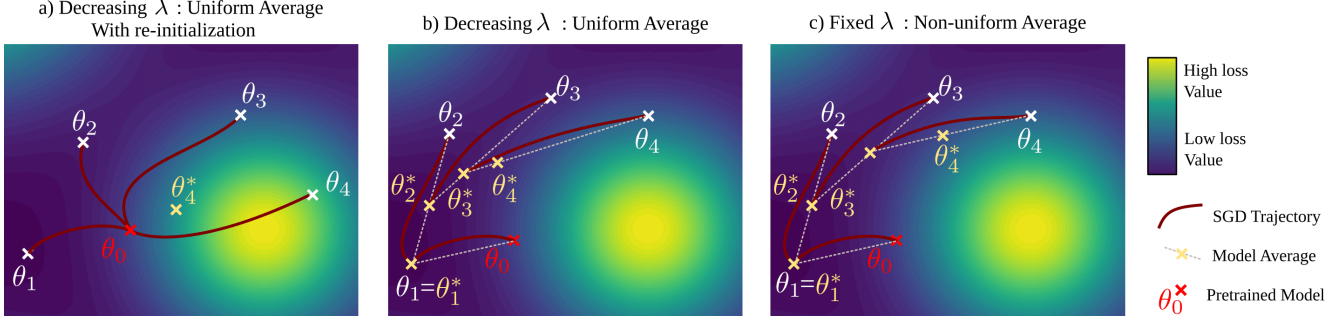


Figure 2. Illustration of the parameter trajectory and model averaging in the loss landscape. (a) Shows the trajectory where models are re-initialized at the start of each task leading to disparate solutions. (b) Depicts decreasing λ_t : Uniform Averaging without re-initialization, showing the convergence of model parameters towards a solution that balances between tasks. (c) Recent tasks are given more weight, resulting in a solution that remains close to the latest task’s model while considering previous tasks. The color gradient indicates the loss value, with darker shades representing lower loss.

space ensembles, however, demand substantial computational resources at inference. Weight-space ensembles offer a computationally efficient alternative by interpolating between model weights [25, 44, 63, 74, 75]. Wortsman *et al.* [75] achieved this by interpolating between zero-shot CLIP and fine-tuned model weights, resulting in performance gains on both the fine-tuning task and under-distribution shifts. Matena *et al.* [38] average models with the same initialization that are fine-tuned on different text classification tasks. We show that this merging approach does not produce good performance in CL.

Our approach differs from [38, 74, 75] as it focuses on sequential finetuning with weight-averaging to be accustomed to the CL setting, we iteratively perform our procedure once per task, using the averaged model at each task as the initialization for the next task.

Linear mode connectivity. Interpolation of neural network weights has been shown to maintain high accuracy across various scenarios, along a shared optimization trajectory [7, 14, 16, 25, 74, 75]. The concept of linear mode connectivity, introduced by Frankle *et al.* [16], describes a state where accuracy is retained along the linear interpolation between two networks’ weights. Analogously, Neyshabur *et al.* [43] demonstrates that exists a connection between minima obtained by pre-trained models versus freshly initialized ones. They note that there is no performance barrier between solutions coming from pre-trained models, but there can be a barrier between solutions of different randomly initialized models. Mizraheh *et al.* [41] investigated linear connectivity in the context of multi-task learning and CL, they show that it exists a linear path solution between two models: one excelling in task A, and the other fine-tuned on both tasks A and B.

These works offer a solid foundation of both theoretical and empirical evidence supporting the efficacy of linear interpolation in model performance. Building on these findings, our work presents a novel solution tailored to CL, an area yet to be explored from this angle.

3. Method

3.1. Problem formulation and overview

Continual learning with pre-trained models. We consider a classification model $M_\theta(\cdot) = h(f(\cdot))$, where $f(\cdot)$ is a feature extractor and $h(\cdot)$ is a classification head, both parameterized by a unified set of parameters θ . The model M_θ is initialized with parameters θ_0 of a pre-trained model and then trained sequentially on a series of incremental tasks, each represented by the corresponding training set \mathcal{D}_t , for $t \in \{1, \dots, T\}$. The primary objective is to achieve robust performance across the test sets associated with these tasks. Specifically, for each task t , the dataset \mathcal{D}_t is defined as $\mathcal{D}_t = \bigcup_{c \in \mathcal{C}_t} \mathcal{D}_t^c$ where $\mathcal{D}_t^c = \{(x_n^c, y_n^c)\}_{n=1}^{N_c}$, and \mathcal{C}_t denotes the set of novel classes introduced in task t . Here, N_c represents the number of training instances for each class c , with (x_n^c, y_n^c) denoting the n -th training instance and its corresponding label. In Class-Incremental Learning (CIL), evaluation is conducted across all observed classes without the need for task-index labels [65].

This problem poses two primary challenges: (i) the necessity to adapt the knowledge acquired from the pre-trained model to new tasks; and (ii) the imperative to maintain the model’s comprehensive learning capabilities to avoid forgetting previously acquired knowledge while assimilating new tasks.

Overview. Our method, named *Continual Model Averaging (CoMA)*, tackles the CIL problem. Our approach is based on the idea that averaging the old and updated model weights proves to be an efficient method of balancing the old and new knowledge. In summary, upon completing each task, we use the obtained model as a starting point for the next task. This starting model is then fine-tuned on the data from the novel task. The task concludes with a weighted average of the initial and updated model parameters, yielding a model that performs effectively on both the past and current tasks. The averaging procedure is detailed in Sec. 3.2.

Furthermore, we introduce *Continual Fisher-weighted*

Model Averaging (CoFiMA), wherein each parameter is set to its weighted average, derived from both the preceding model and the fine-tuned variant. Unlike CoMA, the weighting for each parameter is determined by its Fisher information [10, 59] to provide control over the importance assigned to each weight of the model. This procedure is detailed in Sec. 3.3, and the corresponding algorithm is described in Alg. 1.

3.2. Continual Model Averaging

Temporarily disregarding the typical memory limitations of CIL, a practical approach through model ensembling is proposed as follows: individual models are trained for each task sequentially, starting with the pre-trained model, and the model is saved after each task. Suppose we have trained on a sequence of T tasks, then this method will result in T distinct neural networks, each with its own set of parameters, denoted as $\theta_1, \dots, \theta_T$. The objective is to create a composite neural network with parameters θ ensuring effective performance across all the tasks.

We model the posterior over the composite parameters θ conditioned on the task-specific parameters θ_t as $p(\theta|\theta_t)$. The parameters are assumed to be distributed according to an isotropic Gaussian distribution $\theta \sim \mathcal{N}(\theta_t, I)$, where I is the identity matrix [38, 75]. By maximizing the average of the task-specific log-likelihoods of these posterior distributions for all T models, we obtain the following optimization problem:

$$\theta^* = \arg \max_{\theta} \frac{1}{T} \sum_{t=1}^T \log p(\theta|\theta_t). \quad (1)$$

The solution to this optimization problem has closed-form and is the simple average of the model parameters [38]. With a slight abuse in notation, where we use the sum operator to denote the element-wise summation across sets, it can be written:

$$\theta^* = \frac{1}{T} \sum_{t=1}^T \theta_t. \quad (2)$$

Below, we detail how we adapt this model-averaging method to the constraints of CIL where the task data arrives sequentially, and without storing previous data.

First, one of the goals of our approach is to prevent the number of parameters from growing linearly with T . Therefore, we shift from simultaneous averaging all models $\theta_1, \dots, \theta_t$ to the iterative computation of the average θ^* :

$$\theta_t^* = \lambda \theta_t + (1 - \lambda) \theta_{t-1}^*, \quad (3)$$

where $\lambda \in [0, 1]$. We add model-level weighting λ as an additional hyper-parameter to set the relative importance of each model. In terms of memory, during each training phase, the storage overhead of our approach compared to naive sequential fine-tuning over all the tasks is limited to

the size of a single model. However, when transitioning to subsequent tasks, only θ_t^* requires storage. To initialize this recursion, we start at $t = 1$ with θ_1^* which is set to the model parameters obtained at the end of the first task. Moreover, the rationale for Eq. 3 is reinforced by the findings of Mirzadeh *et al.* [41]. This work indicates that, given two models each trained on a separate task, a model proficient in both tasks typically resides at the linear interpolation of their respective parameter spaces.

Secondly, initiating the training of each task from the same pre-trained model θ_0 can result in convergence to distinct regions within the parameter space [16, 53]. This case is illustrated in Fig. 2 (a). However, if the learned parameters are too distant, the Gaussian assumption of the posterior distribution is no longer valid, and averaging the network could lead to high-loss regions and poor performance of the aggregated model. Therefore, for each task t , we initiate finetuning from θ_{t-1}^* rather than the initial pre-trained model θ_0 . This change limits the risk of reaching distant parameter regions.

Third, we observe that the uniform averaging of the models may still lead to a region that is far from every θ_t as illustrated in Fig. 2 (b). Therefore, we propose to perform the non-uniform averaging, giving higher importance to the latest tasks as shown in Fig. 2 (c). This is obtained by using a constant weight parameter $\lambda = 0.5$. The motivation for giving more importance to the latest tasks is that early models are trained only on the first few tasks, while *the last model has encoded the knowledge from both old and recent tasks via sequential fine-tuning*. In this way, the final model θ_T^* is unlikely to fall in the region with the high value of the loss since it remains close to the model from the last task.

Handling Classifier Parameters. In each novel task t , there are unique parameters (specifically, new head parameters associated with new classes) not present in the preceding models that are subject to averaging. To accommodate this, *we restrict the averaging process (as in Eq. 3) exclusively to the parameters that are common across models* (this includes both backbone parameters and the shared portions of head parameters), while excluding the new head weights (pertaining to new classes) from the averaging.

3.3. Continual Fisher-weighted Model Averaging (CoFiMA)

Uniform weight-averaging operates under the implicit assumption that all weights of the model have the same importance for the training task t . This assumption could potentially compromise model performance since different parameters can have various impacts on the network output [28]. To enhance the averaging process, we propose a more refined model averaging for CL. Specifically, we employ the Fisher information matrix [10, 59], which encapsulates the quantity of information that observed data x

Algorithm 1 CoFiMA

Inputs: Training dataset \mathcal{D}_t for task $t = 1, \dots, T$; network $M_\theta(\cdot)$ with pre-trained parameters θ_0^* ;

Initialization: Random Initialization of the classifiers.

- 1: *# sequential tasks.*
 - 2: **for** task $t = 1, \dots, T$ **do**
 - 3: - Train M_{θ_t} with cross-entropy loss on \mathcal{D}_t .
 - 4: - Calculate Fisher Diagonal information F_t for θ_t .
 - 5: - Calculate new weights θ_t^* according to Eq. 7.
 - 6: - Calculate Fisher Diagonal information F_t^* for θ_t^* .
 - 7: **end for**
-

provides about the network parameters θ_t . The Fisher information matrix F_{θ_t} of a neural network $p_{\theta_t}(y|x)$ trained to predict an output y from input data x is computed as in [3]:

$$F_{\theta_t} = \mathbb{E}_{x \sim \mathcal{D}_t} \left\{ \mathbb{E}_{y \sim p(y|x, \theta_t)} [\nabla_{\theta_t} \log p(y|x, \theta_t) \nabla_{\theta_t} \log p(y|x, \theta_t)^T] \right\}. \quad (4)$$

Given the computational cost of storing the full Fisher matrix, this study adopts the diagonal of the Fisher matrix for practicality [28, 47, 59]. The computation of the diagonal Fisher is feasible within the same order of complexity as the standard back-propagation training process, as it necessitates only a single gradient computation per data point. Through Monte Carlo sampling [59, 60], the diagonal Fisher matrix can be estimated as follows:

$$\hat{F}_{\theta_t} = \frac{1}{N} \sum_{i=1}^N \mathbb{E}_{y \sim p(y|x_i, \theta_t)} \left\{ [\nabla_{\theta_t} \log p(y|x_i, \theta_t)]^2 \right\}, \quad (5)$$

where the expectation over y is estimated from the data samples N . The estimation of this Fisher information matrix F_θ is based on the assumption that the statistical properties of the data are locally similar around the parameter values [35, 58]. This assumption aligns with the Gaussian assumption previously utilized in deriving CoMA. However, as discussed in the work of Chizat *et al.* [6], the use of PTMs limits the risk of facing significant changes in parameter distributions which would violate the *locality assumption* [10].

We now consider that the posterior of each task model $p(\theta|\theta_t)$ is defined as $\theta \sim \mathcal{N}(\theta_t, F_t^{-1})$. As in Eq. 2, maximization of the average of the task-specific log-likelihoods brings us to the following closed-form solution:

$$\theta^* = \frac{\sum_{t=1}^T F_t \theta_t}{\sum_{t=1}^T F_t} \quad (6)$$

with a slight abuse in notation, where we use the division operator to denote the element-wise division. As in Sec. 3.2, we adapt this formulation making it iterative and increasing

the importance of the latest task in the parameter update. This brings us to the following update rule for θ_t^* :

$$\theta_t^* = \frac{\lambda F_t \theta_t + (1 - \lambda) F_{t-1}^* \theta_{t-1}^*}{\lambda F_t + (1 - \lambda) F_{t-1}^*}, \quad (7)$$

where F_{t-1}^* is the Fisher matrix estimated with θ_{t-1}^* on the data \mathcal{D}_{t-1} . In terms of memory requirements, this solution requires the storage of only the model parameters θ_t^* and the Fisher matrix θ_t^* in between two consecutive tasks. Similarly to CoMA, the recursion is initialized at the end of the first task with $\theta_1^* = \theta_1$ and estimating F_1^* with θ_1 on \mathcal{D}_1 .

4. Experiments

In this section, we first briefly describe the experimental setups and then present the experimental results.

4.1. Experimental Setups

Datasets. Our method was tested on five continuous learning benchmarks. We split each into 10 separate tasks, as done in SLCA [82]. The CIFAR-100 dataset [31] has 100 classes of natural images, each with 500 training images. The ImageNet-R dataset [22] includes images from 200 classes, divided into 24,000 for training and 6,000 for testing. These images, although related to ImageNet-21K, are considered challenging because they are either hard examples from ImageNet or new images in different styles, making them tough for the pre-trained model to adapt. The CUB-200 dataset [68] consists of images from 200 bird classes, with about 60 images per class, half for training and half for testing. The Cars-196 dataset [29] is made up of 196 types of car images, split into 8,144 for training and 8,040 for testing, maintaining a similar class ratio. Lastly, VTAB-50 [81] dataset contains diverse classes from multiple *complex realms*. We sample 5 datasets from VTAB, each containing 10 classes, to construct the cross-domain CIL setting. The first two datasets are broader in classification, similar to the ones used in [72, 73], while the last three focus on more specific classifications. More details are reported in the Supp.

We compute the average classification accuracy (denoted as Inc-Acc (%)) of all the classes ever seen after learning each incremental task and present the accuracy after learning the last task (denoted as Last-Acc (%)).

Baselines and Competitors: We first compare to the state-of-the-art PTM-based CIL methods L2P [73], Dual-Prompt [72] and SLCA [82]. We also utilize the same PTM as the initialization for classical CIL methods GDumb [49], LwF [34], DER [4], BIC [77], EWC [28]. Additionally, we report the baselines, sequentially fine-tuning of the model, denoted as Seq FT, and Prototype-classifier is a cosine similarity classifier on the extracted features of the pre-trained

Method	Memory -Free	Split CIFAR-100		Split ImageNet-R		Split CUB-200		Split Cars-196	
		Last-Acc	Inc-Acc	Last-Acc	Inc-Acc	Last-Acc	Inc-Acc	Last-Acc	Inc-Acc
Joint-Training	-	93.22 \pm 0.16	-	79.60 \pm 0.87	-	88.00 \pm 0.34	-	80.31 \pm 0.13	-
GDumb [49]		81.92 \pm 0.15	89.46 \pm 0.94	24.23 \pm 0.35	43.48 \pm 0.49	61.80 \pm 0.77	79.76 \pm 0.18	25.20 \pm 0.84	49.48 \pm 0.74
DER++ [4]		84.50 \pm 1.67	91.49 \pm 0.61	67.75 \pm 0.93	78.13 \pm 1.14	77.42 \pm 0.71	87.61 \pm 0.09	60.41 \pm 1.76	75.04 \pm 0.57
BiC [77]		88.45 \pm 0.57	93.37 \pm 0.32	64.89 \pm 0.80	73.66 \pm 1.61	81.91 \pm 2.59	89.29 \pm 1.57	63.10 \pm 5.71	73.75 \pm 2.37
L2P [73]	✓	82.76 \pm 1.17	88.48 \pm 0.83	66.49 \pm 0.40	72.83 \pm 0.56	62.21 \pm 1.92	73.83 \pm 1.67	38.18 \pm 2.33	51.79 \pm 4.19
DualPrompt [72]	✓	85.56 \pm 0.33	90.33 \pm 0.33	68.50 \pm 0.52	72.59 \pm 0.24	66.00 \pm 0.57	77.92 \pm 0.50	40.14 \pm 2.36	56.74 \pm 1.78
EWC [28]	✓	89.30 \pm 0.23	92.31 \pm 1.66	70.27 \pm 1.99	76.27 \pm 2.13	68.32 \pm 2.64	79.95 \pm 2.28	52.50 \pm 3.18	64.01 \pm 3.25
LwF [34]	✓	87.99 \pm 0.05	92.13 \pm 1.16	67.29 \pm 1.67	74.47 \pm 1.48	69.75 \pm 1.37	80.45 \pm 2.08	49.94 \pm 3.24	63.28 \pm 1.11
Prototype-classifier	✓	60.29 \pm 0.00	69.18 \pm 0.00	38.45 \pm 0.00	45.59 \pm 0.00	80.66 \pm 0.00	88.95 \pm 0.00	28.58 \pm 0.00	39.83 \pm 0.00
Seq FT	✓	88.86 \pm 0.83	92.01 \pm 1.71	71.80 \pm 1.45	76.84 \pm 1.26	68.07 \pm 1.09	79.04 \pm 1.69	49.74 \pm 1.25	62.83 \pm 2.16
SLCA [82]	✓	91.53 \pm 0.28	94.09 \pm 0.87	77.00 \pm 0.33	81.17 \pm 0.64	84.71 \pm 0.40	90.94 \pm 0.68	67.73 \pm 0.85	76.93 \pm 1.21
CoFiMA (Ours)	✓	92.16 \pm 0.24	94.66 \pm 0.94	79.48 \pm 0.26	83.28 \pm 0.56	87.02 \pm 0.56	92.03 \pm 0.69	73.46 \pm 0.64	79.35 \pm 0.96

Table 1. Experimental results for continual learning on Split CIFAR-100, Split ImageNet-R, Split CUB-200 and Split Cars-196 using ViT-B/16 [11] supervised pre-training on ImageNet-21K [54].

Method	Memory -Free	Split CIFAR-100		Split ImageNet-R		Split CUB-200		Split Cars-196	
		Last-Acc	Inc-Acc	Last-Acc	Inc-Acc	Last-Acc	Inc-Acc	Last-Acc	Inc-Acc
Joint-Training	-	89.11 \pm 0.06	-	72.80 \pm 0.23	-	79.55 \pm 0.04	-	74.52 \pm 0.09	-
GDumb [49]		69.72 \pm 0.20	80.95 \pm 1.19	28.24 \pm 0.58	43.64 \pm 1.05	45.29 \pm 0.97	66.86 \pm 0.63	20.95 \pm 0.42	45.40 \pm 0.66
DER++ [4]		63.64 \pm 1.30	79.55 \pm 0.87	53.11 \pm 0.44	65.10 \pm 0.91	61.47 \pm 0.32	77.15 \pm 0.61	50.64 \pm 0.70	67.64 \pm 0.45
BiC [77]		80.57 \pm 0.86	89.39 \pm 0.33	57.36 \pm 2.68	68.07 \pm 0.22	74.39 \pm 1.12	82.13 \pm 0.33	65.57 \pm 0.93	73.95 \pm 0.29
EWC [28]	✓	81.62 \pm 0.34	87.56 \pm 0.97	64.50 \pm 0.36	70.37 \pm 0.41	61.36 \pm 1.43	72.84 \pm 2.18	53.16 \pm 1.45	63.61 \pm 1.06
LwF [34]	✓	77.94 \pm 1.00	86.90 \pm 0.90	60.74 \pm 0.30	68.55 \pm 0.65	61.66 \pm 1.95	73.90 \pm 1.91	52.45 \pm 0.48	63.87 \pm 0.31
Seq FT	✓	81.47 \pm 0.55	87.55 \pm 0.95	64.43 \pm 0.44	70.48 \pm 0.54	61.67 \pm 1.37	73.25 \pm 1.83	52.91 \pm 1.61	63.32 \pm 1.31
SLCA [82]	✓	85.27 \pm 0.08	89.51 \pm 1.04	68.07 \pm 0.21	73.04 \pm 0.56	73.01 \pm 0.16	82.13 \pm 0.34	66.04 \pm 0.08	72.59 \pm 0.04
CoFiMA (Ours)	✓	86.59 \pm 0.47	89.66 \pm 0.53	68.15 \pm 0.31	73.66 \pm 0.78	75.86 \pm 0.18	83.37 \pm 0.16	63.93 \pm 0.16	70.83 \pm 0.83

Table 2. Experimental results on Split CIFAR-100, Split ImageNet-R, Split CUB-200 and Split Cars-196 using MoCo v3 [5] self-supervised pre-training on ImageNet-1K.

model. Joint-Training is an upper bound, where the model has been trained on all the tasks at the same time.

Implementation. We adopt a pre-trained ViT-B/16 backbone for all the baselines unless mentioned otherwise. We follow the implementation of SLCA [82] that adapts a small learning rate of 0.0001 for the representation layer and 0.01 for the classification layer with Class-Alignment strategy (CA) that improves the classification layer by modeling the class-wise distributions and aligning the classification layers. We use a batch size of 128 with our hyperparameter $\lambda=0.5$ in all our experiments.

4.2. Experimental Results

Performance Analysis. This section analyzes the performance of CoFiMA across various continual learning benchmarks including Split CIFAR-100, Split ImageNet-R, Split CUB-200, and Split Cars-196, as detailed in Tabs 1 and 2.

In supervised pre-training (Tab. 1), CoFiMA achieves a Last-Acc of 92.16% and an Inc-Acc of 94.66% on Split

CIFAR-100. This performance surpasses SLCA’s performance, demonstrating a gain of **0.63%** and **0.57%** in Last-Acc and Inc-Acc, respectively. Similarly, on Split ImageNet-R, CoFiMA records a Last-Acc of 79.48%, which is a **2.48%** improvement over SLCA’s 77.00%, and an Inc-Acc of 83.28%, outperforming SLCA’s 81.17%. This trend of outperforming SLCA is also observed in Split Cars-196 with a **5.73%** improvement.

In self-supervised pre-training scenarios (Tab. 2), CoFiMA maintains its lead. On Split ImageNet-R, CoFiMA’s Last-Acc of 68.15% and Inc-Acc of 73.66% continue to show an advantage over SLCA. CoFiMA also leads in Split CUB-200 and Split Cars-196, showcasing its consistent performance across different datasets.

Notably, CoFiMA’s performance is not far from the Joint-Training baselines. For instance, in Split CIFAR-100 (supervised), CoFiMA’s Last-Acc is only **1.06%** lower than the Joint-Training baseline of 93.22%. This gap narrows further in other datasets, indicating CoFiMA’s effective-

Variant	Init.	λ	Split CIFAR-100		Split Imagenet-R		Split CUB-200		Split Cars-196		Split VTAB-50	
			Last-Acc	Inc-Acc	Last-Acc	Inc-Acc	Last-Acc	Inc-Acc	Last-Acc	Inc-Acc	Last-Acc	Inc-Acc
Seq FT	-	-	88.86 \pm 0.83	92.01 \pm 1.71	71.80 \pm 1.45	76.84 \pm 1.26	68.07 \pm 1.09	79.04 \pm 1.69	49.74 \pm 1.25	62.83 \pm 2.16	68.48 \pm 0.63	84.37 \pm 0.35
SLCA	-	-	91.53 \pm 0.28	94.09 \pm 0.87	77.00 \pm 0.33	81.17 \pm 0.64	84.71 \pm 0.40	90.94 \pm 0.68	67.73 \pm 0.85	76.93 \pm 1.21	92.52 \pm 0.61	93.54 \pm 1.08
Weight-Ens.	θ_0^*	$1/t$	61.68 \pm 0.14	70.24 \pm 0.46	44.90 \pm 0.14	52.03 \pm 0.46	82.49 \pm 0.27	87.20 \pm 0.56	36.20 \pm 0.27	45.31 \pm 0.31	89.12 \pm 0.35	89.17 \pm 0.14
Weight-Ens.	θ_{t-1}^*	$1/t$	91.69 \pm 0.23	94.52 \pm 0.95	75.85 \pm 0.73	81.51 \pm 0.60	84.28 \pm 0.47	90.07 \pm 0.22	71.82 \pm 0.47	78.85 \pm 0.31	91.26 \pm 0.31	91.72 \pm 0.18
EMA	-	-	91.40 \pm 0.12	93.89 \pm 0.59	79.35 \pm 0.83	83.07 \pm 0.94	84.99 \pm 0.49	90.84 \pm 0.74	65.41 \pm 0.18	73.68 \pm 0.46	93.35 \pm 0.71	92.87 \pm 0.46
CoMA	θ_{t-1}^*	0.5	92.00 \pm 0.13	94.12 \pm 0.63	79.32 \pm 0.05	82.32 \pm 0.17	86.95 \pm 0.29	91.75 \pm 0.39	73.35 \pm 0.59	78.55 \pm 0.42	92.23 \pm 0.52	92.30 \pm 0.43
CoFiMA (Full)	θ_{t-1}^*	$0.5F_t$	92.16 \pm 0.24	94.66 \pm 0.94	79.48 \pm 0.26	83.28 \pm 0.56	87.02 \pm 0.56	92.03 \pm 0.69	73.46 \pm 0.64	79.35 \pm 0.96	93.67 \pm 0.23	94.96 \pm 0.11

Table 3. Experimental results comparing our methods (CoMA, and CoFiMA) to weight-averaging baselines using ViT-B-16 [11].

ness in approaching the upper bounds of CL performance. CoFiMA’s approach, which efficiently balances the retention of old knowledge with the acquisition of new information, contributes to its strong performance in both supervised and self-supervised settings.

Model Averaging. This section evaluates our continual model-averaging approach against two baselines: (i) *Weight-Ensemble*, which uniformly averages model weights (e.g. $\lambda = 1/t$), initializing each model M_t from either the pre-trained PTM (θ_0) or the previous task’s parameters (θ_{t-1}); (ii) *Exponential Moving Average (EMA)* [62], a technique for a running average of model parameters. We use debiased EMA such that for each iteration $m \in 1, \dots, M$, the model is updated as follows: $\theta_m = \beta\theta_m + (1 - \beta)\theta_{m-1}$ with $\beta = 0.999$ (see details in Supp.). Tab. 3 presents the results.

CoFiMA shows superior performance across all datasets. The *Weight-Ensemble* with θ_{t-1} initialization yields competitive results in Split CIFAR-100 (Last-Acc: 91.69%) but underperforms in datasets like Split Cars-196 (Last-Acc: 71.82%). The *Weight-Ensemble* starting from θ_0 shows lower performance. This indicates the limitations of re-initialization to θ_0 , as models are trained on different tasks it leads to different optima, as depicted in Fig. 2(a).

However, *Weight-Ensemble* with θ_{t-1} initialization outperforms the variant starting from θ_0 , emphasizing the importance of initialization. Initializing θ_t from θ_{t-1}^* improves performance on both task t and $t-1$, as done in [24, 75]. This gain in performance indicates that averaging successive models leads to good performance on the current task t , while preserving previous knowledge.

Averaging all models trained up to task t is suboptimal, as weights at task t likely differ significantly from those at task $t=1$. Such averaging shifts the θ_t^* values toward a less optimal minima, leading to decreased performance (refer to Fig. 2(b)). In contrast, CoFiMA avoids this by averaging only with the preceding parameters θ_{t-1} leading to better performance, in line with findings that parameters of over-parametrized pre-trained models undergo minimal change during finetuning [6].

The EMA method, though superior to both *Weight-Ensemble* variants in most scenarios, falls short of

CoFiMA’s performance. This discrepancy may stem from excessive averaging in EMA, where θ_t^* is modified multiple times within the same task, potentially leading to suboptimal outcomes. Differently, our method applies weight averaging only after completing each task, enhancing computational efficiency by selectively focusing on pertinent model weights, while retaining previous knowledge.

4.3. Ablation Study

Effect of PTMs. In this section, we evaluate the performance of CoFiMA approach across a variety of backbone architectures, including self-supervised (MAE [21], MoCoV3 [5], and DINOv2 [45]) and supervised (ViT-Tiny [11], and ViT-B/16-SAM [15]) models. This comprehensive analysis aimed to ascertain the adaptability and performance consistency of CoFiMA in diverse training paradigms. Results are visualized in Fig. 3.

Our results indicate that CoFiMA consistently enhances performance across all tested backbone architectures relative to the baseline SLCA method. For instance, with the ViT-Tiny backbone, CoFiMA improves the Top-1 Average Accuracy from 80.25% to 82.96%. This trend is similarly observed with ViT-Large, where CoFiMA achieves an accuracy of 86.81%, surpassing SLCA’s 85.93%.

In the context of self-supervised learning models, CoFiMA demonstrates its efficacy by outperforming the SLCA on the ViT-B/16-DINOv2, ViT-B/16-MAE, and ViT-B/16-MoCoV3 backbones. Importantly, the ViT-B/16-SAM backbone achieves the best performance among the evaluated models. This can be attributed to the effective generalization features ingrained in the SAM backbone, a result of being trained with the SAM optimizer [15]. This optimizer is known for its capability to enhance model generalizability, which is reflected in the superior performance metrics observed in our experiments, also mentioned in Mehta *et al.* [40] work. We also notice that self-supervised pre-training usually results in larger performance gaps between continual learning baselines and joint training.

When compared to Joint-Training, a standard upper-bound in CL setting, CoFiMA demonstrates competitive performance. Although Joint-Training occasionally leads, as seen with ViT-Large (94.45%) and ViT-B/16-SAM

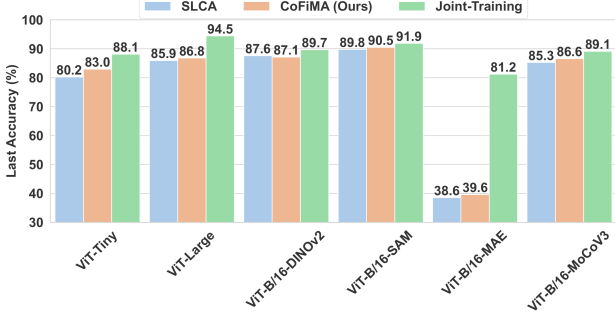


Figure 3. Assessing performance of CoFiMA using various PTMs on CIFAR-100 dataset. CoFiMA enhances the results of SLCA. Additionally, it is noted that CoFiMA’s performance is contingent on the characteristics of the chosen PTM backbone.

(91.87%), CoFiMA remains close, particularly with ViT-B/16-SAM, where it achieves 90.48%.

These results indicate that CoFiMA is a versatile approach, effective in enhancing the performance of various vision transformer backbones in both supervised and self-supervised learning contexts. However, the performance boost from our approach varies with the choice of backbone (size), and its pertaining paradigm. The consistent performance improvements across different models highlight its potential effectiveness in CL.

Balancing the Information of an Old and a New Task.

In CL, a primary objective is to balance knowledge retention from previous tasks with the acquisition of new information from current tasks. This study examines the impact of the λ hyperparameter used to balance the weight of the models when averaging.

In addition to our method, we also include the aggregation scheme of Wise-FT [75] for comparison. More specifically, we introduce the following additional baselines: (i) **WiSE-FT** (θ_0) employs an averaging strategy between the model fine-tuned on the current task (θ_t) and the initial pre-trained model (θ_0), following the formula $\theta_t = \lambda\theta_t + (1 - \lambda)\theta_0$. This method aligns with the WiSE-FT [75] and *concentrates exclusively on the initial pre-training and the current task without integrating knowledge from preceding tasks*. (ii) **WiSE-FT** (θ_{t-1}) adapts Wise-FT to the CL context by incorporating sequential fine-tuning model with the initial pre-trained model θ_0 . and (iii) **CoMA (Ours)**.

According to Fig. 4, CoMA demonstrates superior performance compared to both versions of Wise-FT across various λ settings. This improvement suggests that our method is more effective at incorporating new knowledge while preserving old information. The best performance is achieved with $\lambda = 0.3$, giving the best trade-off between learning new task t (i.e plasticity), while preserving knowledge from previous tasks (i.e stability). For CoMA, the extreme values for λ present scenarios similar to Seq FT, we continue fine-tuning the previous/current model if $\lambda = 0/\lambda = 1$.

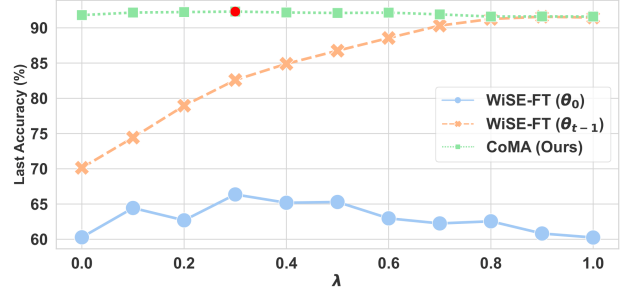


Figure 4. Ablation study on the effect of λ on CIFAR-100 dataset. The red marker point represents the best performance.

Furthermore, the performance of WiSE-FT (θ_{t-1}) surpasses that of WiSE-FT (θ_0), highlighting the benefit of incorporating sequential learning from previous tasks. As λ increases, there is an observed enhancement in the performance of WiSE-FT (θ_{t-1}). However, it remains less effective compared to CoMA, this underperformance is attributed to the impact of averaging with the model θ_0 , which possesses significantly different parameter values due to fine-tuning. Consequently, this averaging process results in suboptimal parameter values as explained in Sec. 3.

In summary, our method, CoMA, effectively maintains a balance between retaining old task information and adapting to new task data. This is achieved by leveraging weight averaging between successive models. The key benefit here is that the parameter values of models at tasks t and $t - 1$ do not diverge significantly, ensuring that the averaged parameters remain effective for both tasks [6, 24].

5. Conclusion

In this work, we have introduced a novel approach, CoFiMA, designed to address the challenge of catastrophic forgetting in the CL setting. This approach builds its grounding on two pillars. First, it leverages linear connectivity, providing a balanced mechanism for retaining prior knowledge while accommodating new information. Second, Fisher information is incorporated to intelligently weigh the averaging of parameters. This refinement allows for the adjustment of each parameter’s value based on its importance, as determined by its Fisher information, thereby effectively reducing catastrophic forgetting.

Our benchmarking on diverse datasets with various PTMs backbones demonstrates that CoFiMA consistently outperforms state-of-the-art CIL methods. Our findings underscore the efficacy of CoFiMA in mitigating forgetting and highlight its versatility across different PTM backbones and benchmark datasets.

Acknowledgements. This paper has been supported by Hi!PARIS Center on Data Analytics and Artificial Intelligence. This work was granted access to the HPC resources of IDRIS under the allocation AD011013860 made by GENCI.

References

- [1] Wickliffe C Abraham and Anthony Robins. Memory retention—the synaptic stability versus plasticity dilemma. *Trends in neurosciences*, 28(2):73–78, 2005. 2
- [2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018. 2
- [3] Shun-ichi Amari. Neural learning in structured parameter spaces - natural riemannian gradient. In *Advances in Neural Information Processing Systems*. MIT Press, 1996. 5
- [4] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33, 2020. 2, 5, 6
- [5] Xinlei Chen*, Saining Xie*, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021. 6, 7, 2
- [6] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming, 2020. 5, 7, 8
- [7] Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. Fusing finetuned models for better pretraining, 2022. 3
- [8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1
- [9] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyang Wu, and Rama Chellappa. Learning without memorizing. In *CVPR*, 2019. 2
- [10] Mine Dogucu, Alicia Johnson, and Miles Ott. *bayesrules: Datasets and Supplemental Functions from Bayes Rules! Book*, 2021. R package version 0.0.2.9000. 2, 4, 5
- [11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 1, 6, 7, 2
- [12] Ethan Dyer, Aitor Lewkowycz, and Vinay Ramasesh. Effect of scale on catastrophic forgetting in neural networks. In *ICLR*, 2022. 1, 2
- [13] Stéphane d’Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *ICML*. PMLR, 2021. 1
- [14] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations (ICLR)*, 2022. 3
- [15] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. 7, 2
- [16] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, 2020. 3, 4
- [17] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4), 1999. 1
- [18] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*. Springer series in statistics New York, 2001. 2
- [19] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *CSUR*, 50(2), 2017. 1
- [20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2015. 1
- [21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. 7, 2
- [22] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 2021. 5, 3
- [23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2
- [24] Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights, 2022. 7, 8
- [25] Pavel Izmailov, Dmitrii Podoprikin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018. 3
- [26] Dongwan Kim and Bohyung Han. On the stability-plasticity dilemma of class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20196–20204, 2023. 2
- [27] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 1
- [28] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13), 2017. 1, 2, 4, 5, 6
- [29] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. 5
- [30] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. 3
- [31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009. 5, 3
- [32] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2

- [33] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*. Springer, 2016. 1
- [34] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 40(12), 2017. 2, 5, 6
- [35] Zhibin Liao, Tom Drummond, Ian Reid, and Gustavo Carneiro. Approximate fisher information matrix to characterize the training of deep neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1): 15–26, 2020. 5
- [36] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. 1
- [37] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1
- [38] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022. 3, 4
- [39] Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning. *arXiv preprint arXiv:2112.09153*, 2021. 2
- [40] Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning, 2023. 1, 7
- [41] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning, 2020. 2, 3, 4
- [42] Kengo Murata, Seiya Ito, and Kouzou Ohara. Learning and transforming general representations to break down stability-plasticity dilemma. In *Proceedings of the Asian Conference on Computer Vision*, pages 3994–4010, 2022. 2
- [43] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning?, 2021. 3
- [44] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 3
- [45] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 7, 2
- [46] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2
- [47] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks, 2014. 5
- [48] Quang Pham, Chenghao Liu, and Steven Hoi. Continual normalization: Rethinking batch normalization for online continual learning, 2022. 2
- [49] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2020. 2, 5, 6
- [50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*. PMLR, 2021. 1
- [51] Adrian E Raftery, David Madigan, and Jennifer A Hoeting. Bayesian model averaging for linear regression models. *Journal of the American Statistical Association*, 92(437): 179–191, 1997. 2
- [52] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *Proceedings of the International Conference on Learning Representations*, 2021. 2
- [53] Alexandre Ramé, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization, 2023. 4
- [54] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses, 2021. 1, 6
- [55] 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. 2
- [56] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*. PMLR, 2018. 2
- [57] Christian Simon, Masoud Faraki, Yi-Hsuan Tsai, Xiang Yu, Samuel Schuster, Yumin Suh, Mehrtash Harandi, and Manmohan Chandraker. On generalizing beyond domains in cross-domain continual learning. *arXiv preprint arXiv:2203.03970*, 2022. 2
- [58] Alexander Soen and Ke Sun. On the variance of the fisher information for deep learning. *Advances in Neural Information Processing Systems*, 34:5708–5719, 2021. 5
- [59] James C Spall. Monte carlo computation of the fisher information matrix in nonstandard settings. *Journal of Computational and Graphical Statistics*, 14(4):889–909, 2005. 2, 4, 5
- [60] James C. Spall. Improved methods for monte carlo estimation of the fisher information matrix. In *2008 American Control Conference*, pages 2395–2400, 2008. 5
- [61] Asa Cooper Stickland and Iain Murray. Diverse ensembles improve calibration. In *International Conference on Machine Learning (ICML) Workshop on Uncertainty and Robustness in Deep Learning*, 2020. 2
- [62] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015. 7, 2
- [63] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 3

- [64] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, 2020. 1
- [65] Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning, 2019. 3
- [66] Andrés Villa, Juan León Alcázar, Motasem Alfarra, Kumail Alhamoud, Julio Hurtado, Fabian Caba Heilbron, Alvaro Soto, and Bernard Ghanem. Pivot: Prompting for video continual learning. *arXiv preprint arXiv:2212.04842*, 2022. 1
- [67] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 3
- [68] Catherine Wah, Steve Branson, Peter Welinder, et al. The caltech-ucsd birds-200-2011 dataset. 2011. 5
- [69] Liyuan Wang, Kuo Yang, Chongxuan Li, Lanqing Hong, Zhenguo Li, and Jun Zhu. Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning. In *CVPR*, 2021. 2
- [70] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2023. 1
- [71] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An ocam’s razor for domain incremental learning. *arXiv preprint arXiv:2207.12819*, 2022. 1
- [72] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*. Springer, 2022. 1, 2, 5, 6, 3
- [73] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *CVPR*, 2022. 1, 2, 5, 6
- [74] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, 2022. 2, 3, 1
- [75] Mitchell Wortsman, Gabriel Ilharco, Mike Li, Jong Wook Kim, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 4, 7, 8
- [76] Tz-Ying Wu, Gurumurthy Swaminathan, Zhizhong Li, Avinash Ravichandran, Nuno Vasconcelos, Rahul Bhotika, and Stefano Soatto. Class-incremental learning with strong pre-trained models. In *CVPR*, 2022. 2
- [77] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, 2019. 2, 5, 6
- [78] Binbin Yang, Xincheng Deng, Han Shi, Changlin Li, Gengwei Zhang, Hang Xu, Shen Zhao, Liang Lin, and Xiaodan Liang. Continual object detection via prototypical task correlation guided gating mechanism. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9255–9264, 2022. 2
- [79] Guandao Yang, Tianyi Zhang, Polina Kirichenko, Junwen Bai, Andrew Gordon Wilson, and Chris De Sa. Swalp: Stochastic weight averaging in low precision training. In *International Conference on Machine Learning*, pages 7015–7024. PMLR, 2019. 2
- [80] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017. 2
- [81] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019. 5, 3
- [82] Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model, 2023. 1, 2, 5, 6
- [83] Jie Zhang, Junting Zhang, Shalini Ghosh, Dawei Li, Jingwen Zhu, Heming Zhang, and Yalin Wang. Regularize, expand and compress: Nonexpansive continual learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020. 2
- [84] Da-Wei Zhou, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need, 2023. 1, 3

Weighted Ensemble Models Are Strong Continual Learners

Supplementary Material

In this supplementary material, we provide more details about the experimental results mentioned in the main paper, as well as additional empirical evaluations and discussions. The supplementary material is organized as follows: In Section A, we provide the detailed algorithm for CoFiMA. Section B provides additional details on weight-averaging methods. In Section C, we report extra experiments about CoFiMA with different PTM backbones. Finally, in Section D, we detail the baselines, and datasets used in our experiments.

A. Detailed Algorithm

In this section, we provide the detailed implementation of CoFiMA algorithm (see Alg. 2). We utilize the same notations as in SLCA [82] to describe CoFiMA training process.

CoFiMA algorithm begins with inputs that include a pre-training dataset D_{pt} , training datasets D_t for tasks $t = 1, \dots, T$, and a network $M_\theta(\cdot)$ with parameters $\theta = \{\theta_{rps}, \theta_{cls}\}$. The parameters θ_{rps} are initialized by pre-training on D_{pt} , and θ_{cls} are randomly initialized. For each task, the network is trained using cross-entropy loss on D_t , applying different learning rates α and β to θ_{rps} and θ_{cls} , respectively, until convergence.

After training, Fisher Diagonal information F_t is calculated for the current parameters θ_t , followed by an update to new weights θ_t^* . The Fisher information is recalculated for these updated weights as F_t^* . This process includes collecting and saving mean μ_c and covariance Σ_c for features F_c for each class c in C_t .

In the final stage, the algorithm performs classifier alignment [82], where features \hat{F}_c are sampled from a normal distribution defined by μ_c and Σ_c . The classifier $h_{\theta_{cls}}$ is then trained with normalized logits computed from these features until convergence, completing the task-specific training cycle.

Implementation Specifics. A pre-trained ViT-B/16 backbone is utilized for our method as in [82]. For approaches that maintain fixed backbones and employ prompting, an Adam optimizer is used [72, 73] as in the original papers; conversely, for baselines that involve updating the entire model, an SGD optimizer is employed, both using a uniform batch size of 128. Our approach employs a learning rate of 0.0001 for the PTM and 0.01 for the classification layer. Empirically, Zhang *et al.* [82] found that supervised pre-training tends to reach convergence more rapidly than self-supervised pre-training. Thus, for supervised pre-training, we train all baselines for 20 epochs on Split CIFAR-100 and 50 epochs on other datasets. For self-

Algorithm 2 CoFiMA

Input: Pre-training dataset D_{pt} ; training dataset D_t for task $t = 1, \dots, T$; model $M_\theta(\cdot) = h_{\theta_{cls}}(f_{\theta_{rps}}(\cdot))$ with parameters $\theta = \{\theta_{rps}, \theta_{cls}\}$; learning rates α and β for θ_{rps} and θ_{cls} respectively.

Initialization: Pre-train θ_{rps} on D_{pt} ; Initialize θ_{cls} randomly.

```
1: for each task  $t \in \{1, \dots, T\}$  do
2:   Train network  $M_\theta$  on  $D_t$  until convergence:
3:   while not converged do
4:     Update  $M_\theta$  using cross-entropy loss.
5:   end while
6:   Weight update process:
7:   Compute Fisher Information  $F_t$  for  $\theta_t$ .
8:   Update weights  $\theta_t^*$  as per Eq.7.
9:   Calculate Fisher Information  $F_t^*$  for  $\theta_t^*$ .
10:  Feature collection:
11:  for each class  $c \in C_t$  do
12:    Collect feature set  $F_c = [r_{c,1}, \dots, r_{c,N_c}]$ .
13:    Compute mean  $\mu_c$  and covariance  $\Sigma_c$  of  $F_c$ .
14:  end for
15: end for
16: Classifier alignment:
17: for each class  $c \in C_{1:T}$  do
18:   Sample  $\hat{F}_c$  from  $\mathcal{N}(\mu_c, \Sigma_c)$ .
19: end for
20: Train classifier  $h_{\theta_{cls}}$ :
21: while not converged do
22:   Compute and normalize logits  $H_{1:T}$ .
23:   Update  $h_{\theta_{cls}}$  using normalized logits.
24: end while
```

supervised pre-training, a training duration of 90 epochs is applied across all benchmarks.

B. More Details about Baselines

We provide an in-depth analysis of existing weight-averaging methods, including a comparative discussion.

- **Weight-Ensemble** [74]: The key idea of this work is that by averaging the weights of multiple models that have been fine-tuned with different hyperparameter configurations, it is possible to improve both the accuracy and robustness of the resulting model. This method is referred to as "model soups". One of the main benefits is that, unlike conventional ensemble methods, averaging many models in this way does not incur additional inference or

memory costs.

CoFiMA differs from the Model Soup method by integrating Fisher information into the model averaging process, assigning weights to parameters based on their estimated importance to mitigate catastrophic forgetting in continual learning tasks. In contrast, Model Soup [74] averages weights of multiple fine-tuned models to improve accuracy without regard to the sequential nature of data or the forgetting of previous tasks.

- **WiSE-FT** [75] introduces a method for enhancing robustness by ensembling the weights of the zero-shot and fine-tuned models. This approach aims to balance the need for high accuracy on target distributions while preserving the model’s ability to perform well on varied and unforeseen data distributions as follows: $\theta_{\text{WiSE-FT}} = \alpha\theta_{\text{zero-shot}} + (1 - \alpha)\theta_{\text{fine-tuned}}$.

CoFiMA utilizes Fisher information to dynamically weight parameter averaging across tasks, thereby mitigating catastrophic forgetting. In contrast, WiSE-FT focuses on robust fine-tuning of zero-shot models by interpolating between zero-shot and fine-tuned model weights to enhance robustness against distribution shifts without additional computational overhead.

- **Exponential Moving Average (EMA)** [62]: EMA in deep learning models is a technique used to smooth out data by giving more weight to recent observations. It is often used in training neural networks to stabilize and improve the learning process by averaging the parameters over time: $\theta_m = \beta\theta_m + (1 - \beta)\theta_{m-1}$. In our setting, we found that the best performance is achieved by using $\beta = 0.999$ on the evaluated datasets.

Simon *et al.* [57] have already used EMA in the setting of continual learning in the context of knowledge distillation. Other works such as [48, 83] used EMA to stabilize training and reduce variance.

CoFiMA differentiates itself from papers using EMA by leveraging Fisher information to adjust the averaging process, prioritizing parameters based on their relevance to preserve knowledge from previous tasks in CL. Furthermore, CoFiMA’s averaging process is done after learning completely the task, not after some epochs. In contrast, EMA applies a decay factor to average model weights over time, primarily to stabilize training and improve convergence, without a task-specific focus on mitigating forgetting.

C. Ablation Backbones

In this section, we evaluate the performance of CoFiMA approach across a variety of backbone architectures on another dataset CUB-196. Backbones evaluated are self-supervised (MAE [21], MoCoV3 [5], and DINOv2 [45]) and supervised (ViT-Tiny [11], and ViT-B/16-SAM [15]) models. This analysis aims to evaluate the adaptability and perfor-

mance consistency of CoFiMA with different architectures. Results are visualized in Fig. 5.

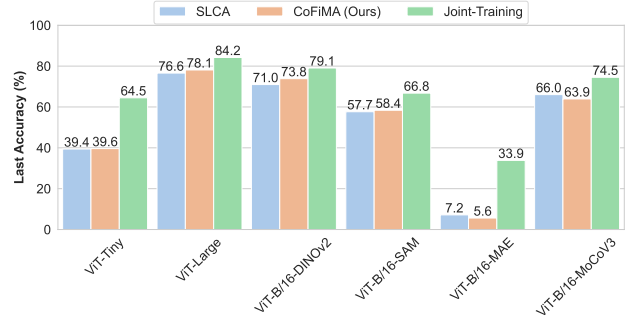


Figure 5. Assessing performance of CoFiMA using various PTMs on CUB-200 dataset. CoFiMA enhances the results of SLCA.

Our results on CUB-200 dataset indicate that CoFiMA consistently enhances performance across all tested backbone architectures relative to the baseline SLCA method. Furthermore, the performance gain varies with the choice and the size of backbone, and its pertaining paradigm.

D. More experimental details

We outline the methods against which our approach is evaluated, and the datasets used in evaluation:

D.1. Baselines

- **Finetune**: Incrementally trains on new datasets, inducing catastrophic forgetting as a consequence.
- **DER++** [4]: mitigates catastrophic forgetting by using a replay buffer of past network inputs and predictions, combined with regularizing current predicted logits, effectively blending rehearsal, knowledge distillation, and regularization.
- **LwF** [34]: Employs knowledge distillation [23] as a regularizer to mitigate forgetting, relying on the legacy model for soft target generation.
- **L2P** [73]: A leading PTM-based CIL method that maintains a frozen pre-trained model while optimizing a prompt pool. It incorporates a ‘key-value’ pairing mechanism for prompt selection and leverages an auxiliary pre-trained model for prompt retrieval.
- **DualPrompt** [72]: An extension of L2P that utilizes two categories of prompts—general and expert—for enhanced performance. It also uses an additional pre-trained model for prompt retrieval.
- **SLCA** [82]: improves the classification layer by modeling the class-wise distributions and aligning the classification layers in a post-hoc fashion.

Table 4. Introduction about benchmark datasets. ObjectNet, OmniBenchmark, and VTAB contain massive classes, and we sample a subset from them to construct the incremental learning task.

Dataset	# training instances	# testing instances	# Classes	Link
CIFAR100	50,000	10,000	100	Link
CUB200	9,430	2,358	200	Link
ImageNet-R	24,000	6,000	200	Link
Cars196	8,144	8,044	196	Link
VTAB	1,796	8,619	50	Link

D.2. Datasets

- **CIFAR100** [31]: Comprises 100 classes, 60,000 images 50,000 for training and 10,000 for testing.
- **CUB200** [67]: Focuses on fine-grained visual categorization, containing 11,788 bird images across 200 subcategories, with 9,430 for training and 2,358 for testing.
- **ImageNet-R** [22]: Extended for CIL by [72], includes various styles and hard instances, totaling 24,000 training and 6,000 testing instances.
- **Cars-196** [30]: is a collection of over 16,000 images of 196 classes of cars, categorized by make, model, and year. It is commonly used for fine-grained image recognition and classification tasks in machine learning.
- **VTAB** [81]: Encompasses 19 tasks across three categories—Natural, Specialized, and Structured. We select five datasets to construct a cross-domain CIL setting. Similar to ADAM [84], we select 5 to construct a cross-domain class-incremental learning setting, i.e., Resisc45, DTD, Pets, EuroSAT, and Flowers.