MIND: Multi-Task Incremental Network Distillation

Jacopo Bonato[*1], **Francesco Pelosin**[*1 2], **Luigi Sabetta**[*1], **Alessandro Nicolosi** [1]

[1]Leonardo Labs, Rome, Italy
[2]Covision Lab, Brixen - South-Tyrol, Italy
jacopo.bonato.ext@leonardo.com, francesco.pelosin@covisionlab.com, luigi.sabetta.ext@leonardo.com,
alessandro.nicolosi@leonardo.com

Abstract

The recent surge of pervasive devices that generate dynamic data streams has underscored the necessity for learning systems to adapt continually to data distributional shifts. To tackle this challenge, the research community has put forth a spectrum of methodologies, including the demanding pursuit of class-incremental learning without replay data. In this study, we present MIND, a parameter isolation method that aims to significantly enhance the performance of replay-free solutions and achieve state-of-the-art results on several widely studied datasets. Our approach introduces two main contributions: two alternative distillation procedures that significantly improve the efficiency of MIND increasing the accumulated knowledge of each sub-network, and the optimization of the BachNorm layers across tasks inside the sub-networks. Overall, MIND outperforms all the state-of-the-art methods for rehearsal-free Class-Incremental learning (with an increment in classification accuracy of approx. $+6\%$ on CIFAR-100/10 and $+10\%$ on TinyImageNet/10) reaching up to approx. $+40\%$ accuracy in Domain-Incremental scenarios. Moreover, we ablated each contribution to demonstrate its impact on performance improvement. Our results showcase the superior performance of MIND indicating its potential for addressing the challenges posed by Class-incremental and Domain-Incremental learning in resource-constrained environments.

Introduction

Despite the remarkable achievements witnessed in deep learning in recent years, a fundamental challenge that remains unresolved pertains to enabling lifelong learning in deep neural networks. Learning continually would unlock the ability of artificial networks to learn new tasks sequentially by adapting to distributional shifts and overcoming the so-called catastrophic forgetting. This issue arises because the network's parameters adapted to the incoming task become ill-suited for the old data, resulting in performance degradation over time. Mitigating catastrophic forgetting typically involves retraining the system from scratch using both old and new data (Zhou et al. 2023; Masana et al. 2023), which is not only expensive but also fails to adapt

to future scenarios automatically. In response, numerous approaches have been proposed. Some methods employ replay buffers to approximate the old data while training on new data (Prabhu, Torr, and Dokania 2020; Rolnick et al. 2019; Aljundi et al. 2019). In contrast, others instantiate new parameters as new tasks emerge (Rusu et al. 2016; Douillard et al. 2022). Additional approaches leverage regularization techniques in the parameter space to address this issue (Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017).

Recently, the deep learning research community has emphasized the importance of exploring compositional approaches in learning systems. The compositional nature of networks has been identified as a crucial aspect of intelligent systems, wherein each sub-module can be selectively accessed to solve specific tasks. For example, the research on LLMs demonstrates through multimodal pipelines how such approaches are effective and efficient: sub-modules are tailored for a particular task characterized by different modalities (i.e. LLAVA (Liu et al. 2023) LENS (Berrios et al. 2023) and Flamingo (Alayrac et al. 2022)). For this reason, systems that exhibit compositionality offer the advantage of being compact, enabling them to address multiple tasks within a single architecture while minimizing memory requirements. In particular, this modular structure of intelligent systems is aligned with some neuroscientific theories like the complementary learning system (CLS) theory (McClelland 1995) which describes how the brain employs two distinct and specialized systems for learning and memory. Among this research stream, the continual learning community proposed several architectures characterized by a slow- and a fast-learner coupled to tackle incremental learning tasks(Arani, Sarfraz, and Zonooz 2022).

Along this research line, we propose a new method called MIND that belongs to the category of parameter isolation approaches (Masana et al. 2023), where sub-regions of the network, called sub-networks, are allocated for tackling individual tasks. However, these sub-regions are not completely disjoint between each other and share a fraction of parameters facilitating the transfer of previously acquired knowledge for future task-solving. In this context, MIND exploits a distillation procedure (Hinton, Vinyals, and Dean 2015) to encapsulate and compress the knowledge from a new model trained for each new task into a sub-network fragment. Furthermore, we propose a vari-

---

*These authors contributed equally.

ation of the optimization procedure of MIND that works under memory limitation, involving a self-distillation procedure where the new model is substituted by MIND itself. Altogether MIND significantly enhances the performance of standard parameter isolation approaches like Pack-Net (Mallya and Lazebnik 2018) and demonstrates superior effectiveness in learning new data while retaining past information compared to the current state-of-the-art methods. We make code and experiments available at `https://github.com/Lsabetta/MIND`.

In summary, the contributions of our work can be outlined as follows:

- We develop a novel parameter isolation approach equipped with a distillation mechanism. This optimization procedure makes use of the knowledge acquired by a new model trained for each new task and transfers it into a sub-network fragment of MIND by matching output probability distributions of the new model and a sub-network of MIND. Importantly, starting from this procedure we propose a different distillation mechanism where MIND self-distills its knowledge about a task inside a single sub-network. Hence, this approach allows MIND with self-distillation to work under memory limitation.

- We propose different policies to select sub-networks for each task. In particular, when using a new model we select randomly the sub-network weights of MIND whereas when performing self-distillation we can select the weights with the highest absolute value.

- As an integrated part of our method, we introduced a gating mechanism to be applied in our backbone. The gating mechanism guides the gradient flow during backpropagation and approximates more correctly its computation.

- We provide a broader and solid experimental framework by testing MIND in 4 different datasets in Class Incremental (CI) scenario (i.e. new classes are presented for each new task). MIND shows optimal performance and outperforms the state-of-the-art methods. Moreover, our results are confirmed when MIND is tested in Domain Incremental (DI) scenario (i.e. the same classes are presented for each new task but the context of the inputs changes).

Related Works

Continual learning has gained significant attention in recent years and various approaches have been proposed to address the problem of catastrophic forgetting. This section provides a small overview of the most prominent works, however, more detailed descriptions of the field have been proposed in reviews such as (Masana et al. 2023) and (Zhou et al. 2023). These categorizations are not mutually exclusive, and many approaches may incorporate techniques from multiple categories.

Architectural-Based Architectural-based approaches aim to modify the model architecture to alleviate catastrophic forgetting. The first works falling in this category are Progressive Neural Networks (PNN) (Rusu et al. 2016), where the network is augmented with new connections spanning both height-wise and width-wise, and Dynamically Expandable Networks (DEN) (Yoon et al. 2018) where they cope with new tasks by splitting/duplicating units and timestamping them. Finally, PackNet, proposed by (Mallya and Lazebnik 2018) compresses several datasets into a single network and works as a multi-task architecture.

Regularization-Based Regularization-based approaches focus on modifying the learning objective or introducing regularization terms to preserve knowledge from previous tasks. Perhaps the most widely used method in this category is Learning without Forgetting (LwF) (Li and Hoiem 2018), which employs distillation to transfer knowledge from an old model to a new model that faces a new task. On the same line of work, where distillation is the major forgetting-preventing mechanism, Learning without Memorizing (LwM) (Dhar et al. 2019) proposes to distillate the attention heatmaps (obtained through Grad-CAM (Selvaraju et al. 2017)) to preserve previous spatial awareness of the model. Another common approach is to use a penalty term for each weight, as in Synaptic Intelligence (SI) (Zenke, Poole, and Ganguli 2017) and Memory Aware Synapses (MAS) (Aljundi et al. 2018). Another seminal work is Elastic Weight Consolidation (EWC) (Kirkpatrick et al. 2017), which penalizes weight changes that could disrupt previously learned knowledge through the computation of the fisher information at the end of each task. Finally, RWalk (Chaudhry et al. 2018) builds upon EWC and introduces a KL-divergence-based Regularization on top of the standard methodology. Finally, PASS (Zhu et al. 2021), uses a prototype vector for each class to reduce catastrophic forgetting combined with a self-supervised learning technique to reduce task-level overfitting.

Rehearsal-Based Rehearsal-based approaches tackle catastrophic forgetting by explicitly storing and replaying past experiences during training. The first method proposed on this line is Experience Replay (ER) (Rolnick et al. 2019), where replay patterns of old tasks are randomly selected to be replayed in future tasks. Shortly after, a plethora of other methodologies have been proposed such as GDumb (Prabhu, Torr, and Dokania 2020), iCarl (Rebuffi et al. 2017) etc. Among the rehearsal-based approaches, we can devise a subsection called pseudo-rehearsal where replay data is generated through generative networks, towards this direction (Shin et al. 2017) constitutes the first work.

Method

We consider a class incremental (CI) scenario, where data streams are split into separate N tasks $\mathcal{T}_i$ with $i = 0, ..., N-1$. Each new task $\mathcal{T}_i$ is characterized by a set of data $\mathcal{X}_i$ and their respective labels $\mathcal{Y}_i$. The number of classes presented at each $\mathcal{T}_i$ is $\frac{M}{N}$ where M is the total number of classes and these are not shared between the other tasks (i.e. $\mathcal{Y}_i \cap \mathcal{Y}_j = \varnothing$ if $i \neq j$). We focused on a replay-free CI scenario, where during the learning phase of $\mathcal{T}_i$ the access to samples of previous tasks is negated. During test time data stream $\mathcal{X}_{test}$ contains images from all the M classes presented during N tasks.
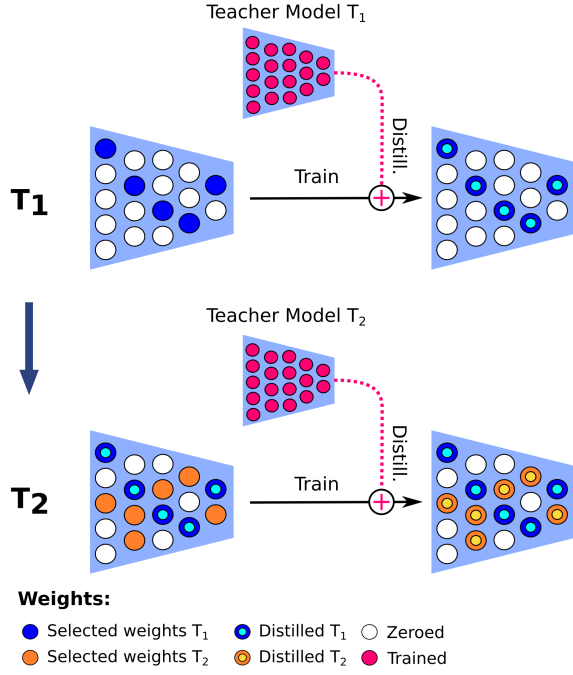
Figure 1: *MIND training pipeline during Task 1 and Task 2.* After optimizing the new model (i.e. teacher model T_1), weights are selected randomly (i.e. blue circles) from MIND for distilling the knowledge from the teacher model (i.e. blue plus cyan circles).

MIND belongs to the category of parameter isolation approaches ((Mallya and Lazebnik 2018), (Serrà et al. 2018)), which comprise sub-networks optimized for each specific task. To enhance the performance of those methods, we employ a distillation technique during the sub-network per-task finetuning process. This methodology eliminates the need for accessing past data, as all the knowledge about previously encountered classes is effectively retained within the MIND sub-networks.

In the subsequent sections, we will detail the various optimization procedures employed in MIND and demonstrate their application in a class incremental (CI) learning scenario.

Sub-network Optimization

A commonly employed technique for optimizing sub-networks is illustrated in PackNet (Mallya and Lazebnik 2018). In this method, the conventional network denoted as f undergoes an iterative process of network pruning. This process selects specific learnable parameters that can effectively accommodate new tasks. This approach enables continual learning without requiring an increase in network capacity, while also minimizing the impact of performance degradation. The fundamental procedure involves three key steps: training the network f on a specific task, pruning a certain fraction of its weights (i.e., setting them to zero), and subsequently retraining the network pruned ($\hat{f}$) to restore accuracy by accounting for the changes in network connectiv-

ity. The value of the free parameters related to the current task (i.e., those that are not set to zero) will be frozen in time once this finetuning step is over.

Once a new task is introduced to train f, all the weights are utilized during the forward pass for output computation. However, only the weights associated with the current task, excluding those about previous tasks, are optimized. This selective optimization process is followed by applying the aforementioned pruning and retraining strategies to finetune the network for the second task. This entire process is iteratively repeated until all the tasks have been completed. Importantly, all the knowledge acquired during previous tasks and stored in the frozen weights of each sub-network is always used as initialization knowledge for the new tasks (i.e. task 3 sub-network uses task 1 and 2 sub-networks weights when performing the forward pass).

In order to optimize f and $\hat{f}$, Cross-Entropy loss $\mathcal{L}_{CE}$ eq. 1 is applied in both training and re-training.

$$\mathcal{L}_{CE} = \sum_{i=1}^{C} t_i log(p_i) \qquad (1)$$

where C is the number of classes in the current task, t_i the true label and p_i is the softmax probability of the ith class.

MIND

A crucial step for methods involving the sub-network optimization Sec. implies re-training the pruned network $\hat{f}$ resulting in a reduced network capacity and consequent performance degradation. Hence, MIND solves this issue by incorporating a distillation mechanism into the optimization procedure (Fig. 1). During each new task $\mathcal{T}_i$, a new network g is initialized and trained from scratch on the new incoming task data $(\mathcal{X}_i; \mathcal{Y}_i)$. Once g is trained, it is used as the teacher model during the $\mathcal{T}_i$ distillation phase whereas we consider the network f utilized in MIND as the student model. f is iteratively pruned using a *random policy* (RP) for weights selection. The RP involves randomly selecting a fraction of the available weights from the network f, where available weights refer to those weights that have not been chosen and optimized during the training of previous tasks. The knowledge from the freshly trained network g is then distilled into the sub-network of f corresponding to task $\mathcal{T}_i$.

To optimize the pruned network $\hat{f}$ during task $\mathcal{T}_i$ through distillation, we employ the Jensen-Shannon loss, denoted as $\mathcal{L}_{SD}$ in Eq. 2. At task $\mathcal{T}_i$, given the new network g parameterized by weights ψ and the pruned network $\hat{f}$ parameterized by weights ϕ_i, we can define the distillation loss as follows:

$$\mathcal{L}_{SD} = \sum_{x_i \in X} \frac{1}{2} D_{KL}(p(z|x,\psi) \parallel p(z_i|x_i,\phi_i)) + \\ \frac{1}{2} D_{KL}(p(z|x,\phi_i) \parallel p(z_i|x_i,\psi)) \qquad (2)$$

where D_{KL} is the Kullback-Leibler divergence and p represents the softmax output of the logits z_i of f_ϕ or g_ψ given a batch input X. Importantly, during the back-propagation only the subset of the weights ϕ_i selected for the task $\mathcal{T}_i$

are updated while the rest is frozen to retain past acquired knowledge.

This distillation loss is combined with cross-entropy loss Eq. 1 using a hyperparameter β as in Eq. 3.

$$\mathcal{L} = \mathcal{L}_{CE} + \beta\mathcal{L}_{SD} \qquad (3)$$

Gating Mechanism   We introduced a binary gating mask acting as a learning routing mechanism to guide the back-propagation procedure. This contribution redirects the flow of the gradient towards the active units of MIND (see Supp. Information Fig. 7).

Weights are defined as active (i.e. mask set to 1) if the parameters have been assigned to any sub-network, or as inactive if the parameters are not assigned yet (i.e. masks set to 0). Among active weights, old sub-network ones are frozen whereas current sub-network ones are updated during backpropagation. By setting previous sub-networks weights to active, the current sub-network is learned by exploiting previously acquired knowledge (i.e. the forward computation takes into account also old sub-networks). This active/inactive masking procedure is also employed by each sub-network during the inference forward passes.

With this gating mechanism, the gradient computation is more precise and avoids discarding some of its magnitude that would be otherwise flowed towards inactive weights unlocking more fast and proficient learning.

Batch Norm   To enhance the adaptability of MIND in CI and DI learning scenarios, we train the Batch-Norm layers (Ioffe and Szegedy 2015) in each task and save the learned parameters corresponding to each sub-network. During the inference phase, we utilize the fitted Batch-Norm parameters that correspond to the selected sub-network. This contribution has been tested and proven to be highly effective in handling distributional shifts and achieving superior performance in our particular scenario, as demonstrated through ablation studies (Sec. Ablations). This solution allows MIND to leverage task-specific Batch-Norm parameters, ensuring a better adaptation to each task and overcoming the limitations observed when Batch-Norm parameters are trained only during the initial task.

MIND with Self-Distillation

Our base distillation approach relies on initializing a new model for each new incoming task. However, in certain real-use case scenarios, hardware limitations such as in applications with low-power devices, impose constraints on available memory resources. For this reason, we explored a self-distillation procedure (Fig. 2) which reduces the amount of memory used.

Instead of using a new network reinitialized at each task, the free weights (zeroed in Fig. 2) of MIND are directly trained on task $\mathcal{T}_i$. Then, we proceed with the pruning step and select the *most important parameters* (MIP policy) trained on $\mathcal{T}_i$ which will be the target for our distillation (student sub-network). The distillation loss is the same as in Eq. 2, using MIND before pruning instead of the new model g.

The MIP policy selects a fraction of weights with the highest absolute values for each layer. We assigned the same

fraction of weights per task using all the available weights of the network (i.e., 10 % of weights in the scenario with 10 tasks). A depiction of the self-distillation procedure is presented in Fig.2

Inference

During the inference phase (Fig. 3), each input image x is fed through all the sub-networks of MIND, and the corresponding logits $\mathbf{z}_i$ vectors are collected (i.e. for each $\mathcal{T}_i$ there is a corresponding logits vector $\mathbf{z}_i$). During inference, these sub-networks are retrieved through the binary active/inactive masking mechanism applied to the network weights described in Sec. . After processing the input image through all the sub-networks, we compute the probability distributions $\mathbf{p}_i$ from the softmax of the logits vectors $\mathbf{z}_i$ scaled by a temperature τ (Eq. 4).

$$\mathbf{p}_i = \texttt{softmax}(\mathbf{z}_i/\tau) \qquad (4)$$

From the distributions of probability $\mathbf{p}_i$ with $i = 0, ..., N-1$ where N is the number of tasks, we select as the predicted class the one with the highest likelihood. Through the `softmax` and temperature scaling (Guo et al. 2017), the logits vectors across sub-networks are respectively standardized and calibrated, obtaining comparable probability distributions of predictions.
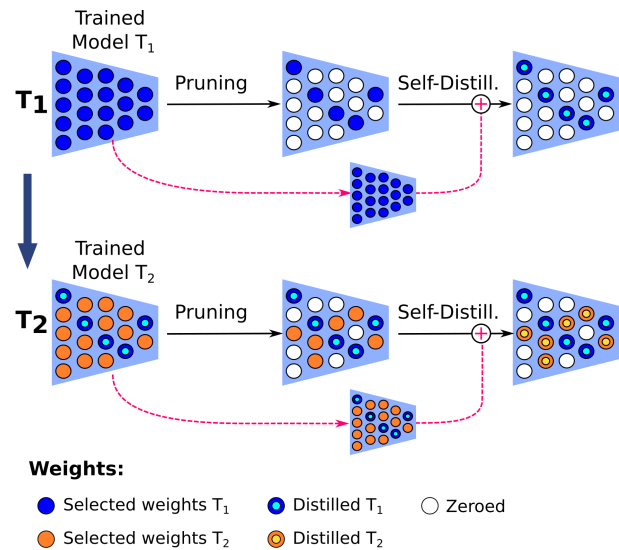

Weights:
- Selected weights T_1
- Distilled T_1
- Zeroed
- Selected weights T_2
- Distilled T_2

Figure 2: *MIND training with self-distillation during task 1 and task 2*. After the optimization of all available weights of the model during task $\mathcal{T}_i$, the most important weights policy is employed for selecting weights with the highest activation values (i.e. blue circles for $\mathcal{T}_1$) and pruning the remaining weights (white circles). These pruned weights serve as targets for distilling knowledge from the non-pruned network. Consequently, the distilled weights (i.e. blue and cyan circles for $\mathcal{T}_1$) are kept unchanged for the new incoming tasks.
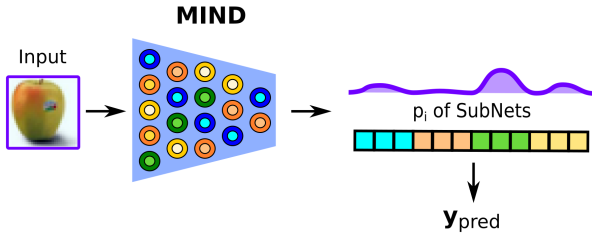
MIND

Figure 3: *MIND inference overview*. For a given input image, MIND collects logit vectors from all sub-networks. During post hoc selection, the class with the highest probability computed from the logits vectors (Eq. 4), is selected.

Experiments

For our experiments, we consider 4 datasets in the standard class-incremental (CI) learning scenario with all classes equally split among 10 tasks. More in detail we used: CIFAR100/10 (Krizhevsky, Hinton et al. 2009) composed of $32 \times 32 \times 3$ images of 100 different classes split into 10 classes per task. TinyImageNet/10 (Chaudhry et al. 2019) composed of $64 \times 64 \times 3$ images with a total of 200 classes split into 20 classes per task. Core50/10 (Lomonaco and Maltoni 2017) composed of $64 \times 64 \times 3$ images of 50 domestic objects split into 5 classes per task. Synbols/10 (Lacoste et al. 2020) composed of $64 \times 64 \times 3$ images of 200 ideograms of the Japanese alphabet split into 20 classes per task (see Supp. Information for details). We opted to use Synbols to facilitate future research to manipulate the latent space of the input distribution, unlocking a deeper understanding of the pros and cons of MIND.

We set the backbone of MIND to `gresnet32` a variation of a `resnet32` comprising the gating mechanism described in Sec. . The dimension of the embeddings is set to $D = 64$ as all the competitors. The training hyperparameters were optimized for each dataset: in brevity, we performed a grid search for each hyper-parameter on a subset of values empirically observed while training. Final numbers and more specs are reported in the Supp. Information.

We report the task agnostic (no task-label) accuracy over all the classes of the dataset after training the last task: $ACC_{TAG} = \frac{1}{C} \sum_{c=1}^{C} a_n$ where C is the total number of classes in the dataset and a_c the accuracy of the single class c. We also report the task aware setting, where at inference time we have access to the task label, unlocking the ability of MIND to query the correct sub-network: $ACC_{TAW} = \frac{1}{T} \sum_{t=1}^{T} a_t$ where T is the total number of tasks and a_t the accuracy on task t.

We run the experiments on a machine equipped with: GPU NVIDIA GeForce RTX 3080, 11th Gen Intel(R) Core(TM) i9-11950H @ 2.60GHz processor, and 32 GB of RAM.

Class-Incremental and Domain-Incremental settings

The thorough assessment of MIND across diverse datasets encompassing CI and DI learning scenarios unveils con-

sistent and reliable performances in both ACC_{TAW} and ACC_{TAG}.

As it is evident from Tab. 1 A, our approach consistently outperforms all other methods in various benchmark settings. Notably, in the CIFAR100/10 dataset, MIND demonstrates a remarkable superiority, achieving approx. +6% increase in ACC_{TAG} and +10% increase in ACC_{TAW} compared to the best existing memory-free technique documented in the literature. Particularly striking is our method's performance on the challenging TinyImageNet dataset, where it significantly outperforms all counterparts by a substantial margin in both ACC_{TAG} and ACC_{TAW} (approx. +6% and +6% respectively).

Analyzing Figure 4 B, it becomes evident that ACC_{TAG} on the observed classes, while experiencing an initial decline, remains relatively steady thereafter. This observation underscores the effectiveness of accurate task identification after the initial tasks have been encountered (when the number of observed classes reaches or exceeds 80). This suggests a judicious balance between the plasticity and stability of MIND.

Importantly, these results are confirmed in Core50/10 and Synbols/10 datasets (Table 1 A and Supp. Information Fig. 8). Collectively, these findings emphasize the significant progress made in the memory-free class-incremental scenario through the utilization of a multi-sub-networks paradigm with distillation, as exemplified by MIND.

To verify the efficacy of MIND compared to another parameter isolation method like PackNet, we conducted experiments on the 4 datasets described above for CI scenarios. The results obtained show that MIND consistently outperforms PackNet across different datasets (CIFAR100/10, TinyImageNet/10, Core50/10, Synbols/10) in ACC_{TAW} and ACC_{TAG}. The improvements in ACC_{TAW} highlight how our contributions increase the adaptability of each sub-network to novel tasks. Furthermore, the enhancements in ACC_{TAG} underscore the superior task detection capabilities of MIND compared to PackNet across various sub-networks. This, in turn, leads to a more refined representation of the output probability distributions denoted as $\mathbf{p}_i$ with $i = 0, ..., N - 1$, where N represents the number of tasks.

For our experiments in Domain-Incremental (DI) scenarios, we consider the Core50 (Lomonaco and Maltoni 2017) dataset (for details see the Supp. Informations). We set the backbone of MIND to `gresnet18` with the dimension of the embeddings set to $D = 512$. We consider a DI learning scenario with 11 tasks where the same 50 classes are presented with a different background for each new task. From the results reported in Tab. 1 B, MIND more than doubles the DI learning results obtained by LwF, EWC, and SI. This result demonstrates how well our method copes with clear distribution shifts in input images thanks to which the proper sub-network configuration can be chosen easily during inference.

Self-Distillation

Self-distillation in MIND represents a fundamental contribution that allows us to make use of the distillation mecha-
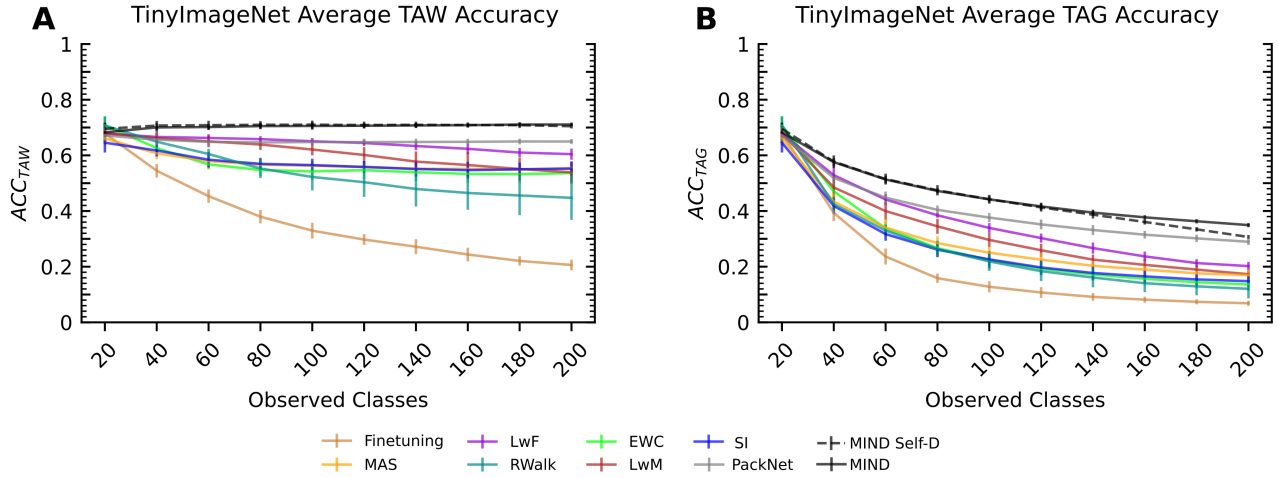
Figure 4: *MIND outperforms all the state-of-the-art algorithms for CI learning*. **A-B**) Comparison between MIND and state-of-the-art CI learning algorithms ACC_{TAG} on TinyImageNet/10 (A) and on Core50/10 (B). Results are reported as mean $\pm$ std across 10 runs obtained from 10 different seeds.

A

		# Params.	CIFAR100/10	TinyImageNet	Core50 (CI)	Synbols
Task-Aware	Finetuning	0.47M	$38.3^{\ddagger}$	20.6 ± 1.9	38.5 ± 7.0	42.8 ± 5.2
	LwF	0.47M	$76.6^{\ddagger}$	60.4 ± 2.0	79.0 ± 5.1	93.8 ± 1.7
	EWC	0.47M	$56.7^{\ddagger}$	53.6 ± 3.1	52.7 ± 6.7	83.8 ± 3.6
	SI	0.47M	$53.1^{\ddagger}$	55.2 ± 2.4	44.8 ± 8.2	81.5 ± 6.1
	MAS	0.47M	$58.6^{\ddagger}$	55.4 ± 1.9	66.5 ± 3.5	84.7 ± 2.7
	RWalk	0.47M	$49.3^{\ddagger}$	44.7 ± 7.9	40.4 ± 8.2	67.8 ± 8.3
	LwM	0.47M	$70.4^{\ddagger}$	53.9 ± 4.0	67.2 ± 4.9	93.0 ± 1.8
	PackNet	0.47M	72.4 ± 1.4	65.0 ± 0.9	95.7 ± 1.2	95.7 ± 1.1
	MIND (Self-D)	0.47M	$\mathbf{82.2 \pm 0.5}$	$\mathbf{70.7 \pm 0.6}$	$\mathbf{99.8 \pm 0.04}$	$\mathbf{98.7 \pm 0.2}$
	MIND	0.94M	$\mathbf{82.3 \pm 0.6}$	$\mathbf{71.1 \pm 0.7}$	$\mathbf{99.7 \pm 0.08}$	$\mathbf{98.4 \pm 0.2}$
Task-Agnostic	Joint	0.47M	$75.39^{\dagger}$	$59.38^{\dagger}$	94.8 ± 0.18	99.4 ± 0.11
	Finetuning	0.47M	$10.1^{\ddagger}$	6.8 ± 0.9	6.6 ± 2.8	11.5 ± 4.0
	LwF	0.47M	$30.2^{\ddagger}$	20.2 ± 1.5	15.0 ± 2.1	47.3 ± 6.0
	EWC	0.47M	$13.1^{\ddagger}$	13.6 ± 2.2	7.6 ± 2.3	34.3 ± 4.7
	SI	0.47M	$13.6^{\ddagger}$	14.5 ± 2.0	7.7 ± 1.2	34.4 ± 6.9
	MAS	0.47M	$13.9^{\ddagger}$	16.9 ± 1.9	11.0 ± 1.9	29.7 ± 1.2
	RWalk	0.47M	$14.0^{\ddagger}$	12.0 ± 3.4	7.1 ± 2.1	23.2 ± 6.0
	LwM	0.47M	$21.9^{\ddagger}$	17.3 ± 1.5	14.8 ± 1.5	47.0 ± 4.6
	PASS	11.2M	$33.76^{\dagger}$	$24.23^{\dagger}$	-	-
	PackNet	0.47M	28.5 ± 2.2	29.0 ± 1.2	40.0 ± 4.3	60.4 ± 4.7
	MIND (Self-D)	0.47M	$\mathbf{35.7 \pm 0.7}$	$\mathbf{30.7 \pm 0.7}$	$\mathbf{55.9 \pm 2.3}$	$\mathbf{76.9 \pm 1.0}$
	MIND	0.94M	$\mathbf{39.9 \pm 0.9}$	$\mathbf{35.0 \pm 0.8}$	$\mathbf{57.9 \pm 2.1}$	$\mathbf{76.5 \pm 2.6}$

B

	Core50 (DI)
LwF	31.38 ± 0.02
EWC	27.91 ± 0.01
SI	25.5 ± 0.01
MIND	$\mathbf{79.28 \pm 2.63}$

Table 1: **A**) Comparison on CIFAR100/10, TinyImageNet/10, Core50/10, and Synbols/10 in Class-Incremental CI scenario with 10 tasks. All methods use `resnet32`. "Joint" here represents the case when the model is trained with all the classes available at once. **B**) Comparison on Core50 Domain-Incremental (DI). ACC_{TAW} and ACC_{TAG} are reported as mean$\pm$ std across 10 runs obtained from 10 different seeds using Avalanche (Lomonaco et al. 2021) framework and FACIL framework (Masana et al. 2023) respectively. When std is not present the results are reported from literature. In particular, results marked by $\ddagger$ are taken from the survey of (Masana et al. 2023) while results marked by $\dagger$ are taken from (Cotogni et al. 2022).

nism and at the same time be compliant with common real use-case hardware limitations. Notably, our self-distillation approach yields task-aware results that are on par with those

achieved using standard MIND. This alignment underscores the efficacy of our method in retaining task-specific information. Interestingly, the self-distillation technique maintains a
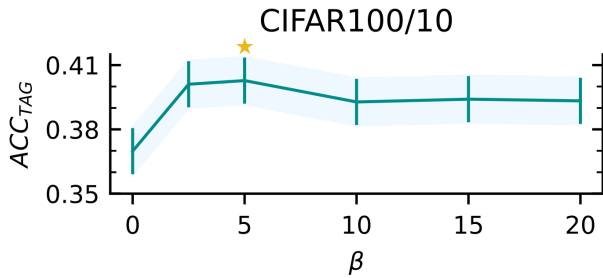
CIFAR100/10

Figure 5: ACC_{TAG} as a function of β for the CIFAR100/10 dataset. Results are reported as mean $\pm$ std across 3 different runs from 3 different seeds. The star represents the final selected value.

Ablation	ACC_{TAG}
✗ Weight Sharing	37.2 ± 0.8
✗ Distillation	37.2 ± 1.2
✗ Batch-Norm	32.6 ± 0.9

Table 2: Ablation studies ACC_{TAG} results for CIFAR100/10. In each row of the table, we removed a component of MIND and reported the final ACC_{TAG} result.

remarkably low loss in ACC_{TAG} if compared to standard MIND (Tab. 1 A). This loss is significant only during the very last tasks (Fig. 4 B) since the self-distillation becomes less efficient given the fact that the trainable parameters of the teacher model decrease as the number of tasks increases. Despite that, the self-distillation approach exhibits competitive performances over existing state-of-the-art methods in both ACC_{TAW} and ACC_{TAG} in all the datasets.

Overall, our self-distillation technique stands as an optimal choice for hardware limitations contexts, offering a harmonious blend of efficient resource utilization and noteworthy performance outcomes. This makes it an attractive solution for systems with limited computational capabilities.

Ablation Studies

Through the following ablation studies, we investigate the effects and contributions of the different components of MIND. All the results are reported for the CIFAR100/10 dataset, obtained through 5 different seeds for each experiment.

Weight Sharing   To evaluate the effectiveness of encapsulating a set of sub-networks into a single model, we conducted an ablation study by removing the weights-sharing between sub-networks. This experiment was crucial in identifying the advantages of using a cohesive set of sub-networks that incrementally share their weights, as opposed to an ensemble of independent sub-networks. The results of this experiment (Tab. 2) showed a decrease of ACC_{TAG} of -2.9%, demonstrating how the knowledge acquired during previous tasks is available for the new sub-networks and can also be used for increasing the current task knowledge.

Distillation   To assess the influence of the distillation loss, we conducted an ablation study by varying the parameter β across the range [0-20]. The outcomes of this ablation investigation are graphically represented in Fig. 5. We observe a decrease of -2.7% when distillation is omitted ($\beta = 0$), as compared to the accuracy reported in Tab. 2. The distillation procedure plays a crucial role in effectively compressing the knowledge from new models and transferring it to sub-networks within MIND. Removing this component affects the overall performance, highlighting the importance of the distillation loss in achieving better accuracy and adaptation in the continual learning setup. To ensure coherence, we opted for a value of 5 for all other experiments, as it delivers the best performance when evaluated on CIFAR100/10.

Batch-normalization   A fundamental aspect to investigate in MIND is the effect of task-specific Batch-Norm parameters on the adaptability of the sub-networks on the CI learning scenario tasks. For this reason, we trained the Batch-Norm-layers only during the first task in both the new model and the sub-network $\mathcal{T}_1$ training and fixed them for the new incoming tasks. We observed a decrease in accuracy of 7.3% (Tab. 2) which suggests how task-specific Batch-Norm parameters are highly effective in handling distributional shifts and achieving superior performances.

Conclusion

In this work, we introduced MIND, a rehearsal-free continual learning method. In particular, we proposed a new parameter isolation method that creates sub-networks tailored for each incremental task. MIND uses a distillation procedure to condense a new model (trained from scratch on each new task) in a sub-network of MIND, that will be exploited as compressed inter-knowledge thereafter. We also introduced a gating mechanism that optimizes the learning, guiding the gradient flow and selecting only the correct units that contributed during learning. This unlocks a more precise and reliable computation of the gradient providing more fast and proficient learning.

Moreover, we proposed an alternative distillation procedure that can be used on systems with memory resource limitations. This alternative approach, called self-distillation, substitutes the role of the teacher (new model) during the distillation procedure with MIND itself.

Finally, we validated our results by running a wide batch of experiments encompassing 5 different benchmarks. Moreover, we ablated several architectural components of MIND and provided a sensitivity analysis of the distillation loss hyperparameter. Results show that MIND can be considered the new state-of-the-art method for class incremental rehearsal-free continual learning.

Acknowledgments

References

Alayrac, J.-B.; Donahue, J.; Luc, P.; Miech, A.; Barr, I.; Hasson, Y.; Lenc, K.; Mensch, A.; Millican, K.; Reynolds, M.; Ring, R.; Rutherford, E.; Cabi, S.; Han, T.; Gong, Z.; Samangooei, S.; Monteiro, M.; Menick, J.; Borgeaud, S.; Brock, A.; Nematzadeh, A.; Sharifzadeh, S.; Binkowski, M.; Barreira, R.; Vinyals, O.; Zisserman, A.; and Simonyan, K. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. arXiv:2204.14198.

Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; and Tuytelaars, T. 2018. Memory aware synapses: Learning what (not) to forget. In *IEEE Proceedings of the European Conference on Computer Vision (ECCV)*.

Aljundi, R.; Belilovsky, E.; Tuytelaars, T.; Charlin, L.; Caccia, M.; Lin, M.; and Page-Caccia, L. 2019. Online Continual Learning with Maximal Interfered Retrieval. In *Advances in Neural Information Processing Systems, (NeurIPS)*.

Arani, E.; Sarfraz, F.; and Zonooz, B. 2022. Learning Fast, Learning Slow: A General Continual Learning Method based on Complementary Learning System. In *International Conference on Learning Representations, (ICLR)*.

Berrios, W.; Mittal, G.; Thrush, T.; Kiela, D.; and Singh, A. 2023. Towards Language Models That Can See: Computer Vision Through the LENS of Natural Language. arXiv:2306.16410.

Chaudhry, A.; Dokania, P. K.; Ajanthan, T.; and Torr, P. H. S. 2018. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In Ferrari, V.; Hebert, M.; Sminchisescu, C.; and Weiss, Y., eds., *European Conference on Computer Vision, (ECCV)*.

Chaudhry, A.; Rohrbach, M.; Elhoseiny, M.; Ajanthan, T.; Dokania, P. K.; Torr, P. H. S.; and Ranzato, M. 2019. On Tiny Episodic Memories in Continual Learning.

Cotogni, M.; Yang, F.; Cusano, C.; Bagdanov, A. D.; and van de Weijer, J. 2022. Gated Class-Attention with Cascaded Feature Drift Compensation for Exemplar-free Continual Learning of Vision Transformers. *arXiv preprint arxiv:2211.12292*.

Dhar, P.; Singh, R. V.; Peng, K.-C.; Wu, Z.; and Chellappa, R. 2019. Learning Without Memorizing. In *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Douillard, A.; Ramé, A.; Couairon, G.; and Cord, M. 2022. DyTox: Transformers for Continual Learning with DYnamic TOken eXpansion. In *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On Calibration of Modern Neural Networks. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, 1321–1330. PMLR.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Bach, F.; and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 448–456. Lille, France: PMLR.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N. C.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Kumaran, D.; and Hadsell, R. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Lacoste, A.; López, P. R.; Branchaud-Charron, F.; Atighehchian, P.; Caccia, M.; Laradji, I. H.; Drouin, A.; Craddock, M.; Charlin, L.; and Vázquez, D. 2020. Synbols: Probing Learning Algorithms with Synthetic Datasets. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems, NeurIPS 2020*.

Li, Z.; and Hoiem, D. 2018. Learning without Forgetting. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023. Visual Instruction Tuning. 2304.08485.

Lomonaco, V.; and Maltoni, D. 2017. CORe50: a New Dataset and Benchmark for Continuous Object Recognition. In *Annual Conference on Robot Learning, CoRL*. PMLR.

Lomonaco, V.; Pellegrini, L.; Cossu, A.; Carta, A.; Graffieti, G.; Hayes, T. L.; Lange, M. D.; Masana, M.; Pomponi, J.; van de Ven, G.; Mundt, M.; She, Q.; Cooper, K.; Forest, J.; Belouadah, E.; Calderara, S.; Parisi, G. I.; Cuzzolin, F.; Tolias, A.; Scardapane, S.; Antiga, L.; Amhad, S.; Popescu, A.; Kanan, C.; van de Weijer, J.; Tuytelaars, T.; Bacciu, D.; and Maltoni, D. 2021. Avalanche: an End-to-End Library for Continual Learning. In *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Mallya, A.; and Lazebnik, S. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Masana, M.; Liu, X.; Twardowski, B.; Menta, M.; Bagdanov, A. D.; and van de Weijer, J. 2023. Class-Incremental Learning: Survey and Performance Evaluation on Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence, (TPAMI)*.

McClelland, O. R., McNaughton. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory.

Prabhu, A.; Torr, P. H. S.; and Dokania, P. K. 2020. GDumb: A Simple Approach that Questions Our Progress in Continual Learning. In Vedaldi, A.; Bischof, H.; Brox, T.; and Frahm, J., eds., *European Conference on Computer Vision (ECCV)*.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Rolnick, D.; Ahuja, A.; Schwarz, J.; Lillicrap, T. P.; and Wayne, G. 2019. Experience Replay for Continual Learning. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems, (NeurIPS)*.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *IEEE International Conference on Computer Vision (ICCV)*.

Serrà, J.; Surís, D.; Miron, M.; and Karatzoglou, A. 2018. Overcoming catastrophic forgetting with hard attention to the task. arXiv:1801.01423.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual Learning with Deep Generative Replay. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems (NeurIPS)*.

Yoon, J.; Yang, E.; Lee, J.; and Hwang, S. J. 2018. Lifelong Learning with Dynamically Expandable Networks. In *International Conference on Learning Representations, (ICLR)*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual Learning Through Synaptic Intelligence. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research.

Zhou, D.-W.; Wang, Q.-W.; Qi, Z.-H.; Ye, H.-J.; Zhan, D.-C.; and Liu, Z. 2023. Deep Class-Incremental Learning: A Survey. arXiv:2302.03648.

Zhu, F.; Zhang, X.-Y.; Wang, C.; Yin, F.; and Liu, C.-L. 2021. Prototype Augmentation and Self-Supervision for Incremental Learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.