MEMORY HEAD FOR PRE-TRAINED BACKBONES IN CONTINUAL LEARNING

Matteo Tiezzi
DIISM, University of Siena
Siena, Italy
matteo.tiezzi@unisi.it

Federico Becattini
DIISM, University of Siena
Siena, Italy
federico.becattini@unisi.it

Simone Marullo
DINFO, University of Florence
Florence, Italy
simone.marullo@unifi.it

Stefano Melacci
DIISM, University of Siena
Siena, Italy
stefano.melacci@unisi.it

ABSTRACT

This paper is focused on the role of classification heads for pre-trained backbones in the context of continual learning. A novel neuron model is proposed as basic constituent of what we refer to as Memory Head, which naturally includes self-organized memorization capabilities, going beyond the ones of classic neurons and specifically designed for continual learning purposes. Memory Heads are based on memory units which are indexed depending on the input, with a mechanism which resembles attention models. Such a computational structure allows the head to adapt to different regions of the input space without altering its behavior in other regions, that might be possibly associated to previously acquired knowledge. The neuron model is generic, as it does not exploit any supervisory information and it does not require any experience replay strategies. When stacked on top of pre-trained backbones, the proposed head allows the network to adapt to new knowledge and to memorize the properties of the temporally streamed data. The experimental activity of the paper covers both the case of frozen and fine-tuned backbones, showing that Memory Heads overcome recent state-of-the art competitors which work in the same setting. Moreover, continual online learning is explored in the class-domain incremental setting, being it a more challenging scenario which is less frequently analyzed in the literature. We demonstrate that Memory Heads are more flexible with respect to "vanilla" heads, and more effective than several experience-replay-based approaches.

1 INTRODUCTION

The recent success attained by exploiting the representational capabilities of large-scale pre-trained networks, often referred to as foundation models (Bommasani et al., 2021), has led to significant improvements in vision (Alayrac et al., 2022), language modeling (Brown et al., 2020; Devlin et al., 2018), time series forecasting (Rasul et al., 2023) and others. Representations learned from large pre-trained models have been proven to be robust, transferable and, to a certain extent, task-agnostic (Radford et al., 2021), leading towards a change in training paradigm that departs from the classic end-to-end learning. These attractive properties have recently been leveraged even outside the domains on which such large models have been designed. Indeed, these architectures are usually learned under the assumption that data is independent and identically distributed (i.i.d). Unfortunately, the i.i.d. assumption does not often hold in realistic scenarios, where samples might not be all accessible at the same time and could be provided as a continuous stream with shifting distribution (Gunasekara et al., 2023). When this happens, the learning process is likely to fail, as neural networks are subject to catastrophic forgetting, meaning that the network over-adapts to the most recently observed samples, at the expense of the earlier ones (Parisi et al., 2019). Continual Learning (CL) strategies (Delange et al., 2021; Mai et al., 2022b) do not require the i.i.d. assumption in order to converge to meaningful solutions, yet they need additional mechanisms that are based on data statistics (Li & Hoiem, 2017) and often require to store significant amounts of training samples (Rolnick et al., 2019) or data prototypes (De Lange & Tuytelaars, 2021b). Such information is frequently stored into memory buffers, to periodically approximate an i.i.d. scenario by rehearsing a carefully selected subset of previously observed data (Delange et al., 2021).

Whilst the standard approach in CL is to train models from *scratch*, the aforementioned attractive properties of large pre-trained models led to a paradigm shift even in the non-i.i.d. scenario tackled by CL (Galashov et al., 2023). Recent findings showed that pre-training weights on large data-collections fosters their robustness to catastrophic forgetting and favours knowledge transfer for downstream CL tasks related to the domains in which the model was pre-trained (Zhang et al., 2023; Galashov et al., 2023). Additionally, this effect tends to be more significant as the scale of pre-training increases (Ostapenko et al., 2022). The standard approach is to leverage the knowledge of representations extracted by a pre-trained *backbone model*. Classification *heads* acting on the representations are then learned on sequences of incremental tasks. There are two major trends for exploiting pre-trained knowledge: (i) keep the backbone fixed while solely learning the heads or other few additional parameters (e.g. prompts or prompt pools (Wang et al., 2022c;a)), or (ii) adapt the learnable parameters of both the backbone and the head. The former approach has the advantage of being less computationally demanding, given that updates are restricted to a small amount of parameters belonging to the heads. Recently proposed prompt-based approaches (Wang et al., 2022c;a), where just small prompts/prompt pools have to be learned, proved to be superior even to the latter approach of fine-tuning both the backbone and the classification head. However, a recent work by Zhang et al. (2023) questioned the current progress of pre-trained-based CL suggesting that simply updating the backbone model at a slower pace with respect to the head, without reverting to any replay buffer or statistics, is already sufficient to reach a good trade off between task specificity and generalizability in continually finetuning the backbone.

In this paper, we follow the promising direction of training classification heads for pre-trained backbones by proposing a novel head model, hereinafter referred to as Memory Head (MH), equipped with memorization capabilities and specifically tailored for continual learning. In the proposed MH, each neuron can route the computation across a learnable key-value mechanism, exhibiting a dynamic behaviour depending on its own input. This allows the model to perform parameter isolation and use separate sets of weights depending on the properties of the current input. This solution goes beyond explicit memorization and replay schemes that are common in CL, since the neuron autonomously develops memorization capabilities by design, without the need for rehearsal steps to avoid forgetting. Moreover, the neuron model is general, since it does not require knowledge of class-labels to build its internal representations. This implies that it can be paired with other existing CL techniques.[1]

Differently from vanilla neural heads, MHs allow the model to handle time-changing data distributions (or new classes) with sets of weights that are *computed* on the fly, as soon as the new information is presented to the network. This prevents forgetting in a way that resembles parameter isolation methods, since previously specialized weights, trained to classify earlier sets of samples, are (softly) isolated from the ones that are being updated to learn the newly provided input information. At the same time, a soft-attention mechanism still allows to transfer information across the different sets of parameters. Fig. 1 illustrates the potentiality of the neurons that compose memory heads in an online stream of 2D data. Such neurons include learnable keys that, while learning from the data stream, end up covering meaningful regions of the feature space (Fig. 1-bottom). For each region, a different set of weights is available, which is softly-blended with the ones in the neighboring regions. As a result, we have different class-related decision boundaries in different areas of the input space. On the contrary, classic neurons (Fig. 1-top) adapt all their weights based on the current input, thus adapting to the most recently observed data distribution (catastrophic forgetting).

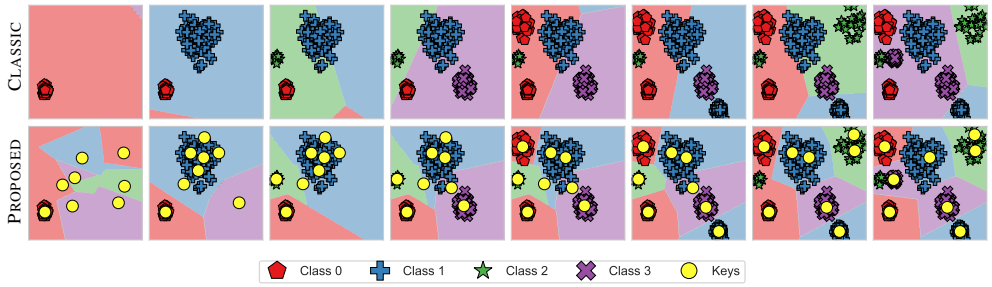

Figure 1: Decision boundaries over time (left-to-right) in a toy 2D dataset with head-only models: data belongs to 4 mutually exclusive categories, where each class is distributed as a bi-modal gaussian distribution, thus we have a composition of 8 gaussian distributions. Such gaussian distributions are considered sequentially to generate samples, which are fed to the network one-by-one. Each picture refers to the instant in which the last sample from a gaussian distribution is processed. We compare the proposed Memory Head (bottom) with a head of classic neurons (top).

[1] MH could be trained with experience replay. However, this goes beyond the scope of this paper, which is focused on comparing the proposed head with classic, replay-based models, showing that it can overcome them in several settings.

MHs are based on neurons that, in principle, could be used to train a deep neural network from scratch. However, the information routing process is strongly sensitive to the neuron inputs, which also contribute in selecting the values of the weights to use for evaluating the neuron output. As a result, this might introduce instabilities during the training stage in deep nets, where the inputs of some hidden layers might significantly and abruptly change over time when the same pattern is provided to the network at different time instants.[2] Differently, MHs are useful both in the setting in which the backbone is frozen, thus where the MH takes care of handling the latent representations with an appropriate set of weights, and in a more dynamic setting in which both the backbone and the MH are updated over time. In fact, slow changes to the backbone can be efficiently handled by the MHs. Moreover, MHs are fully agnostic to the selected backbone architectures, being applicable on top of both convolutional models or Transformers. This is a significant difference from prompt based approaches, such as (Wang et al., 2022b), that are specifically designed for transformers. We show that the versatility of MHs extends beyond the common class-incremental settings, as they can effectively be used in more challenging class-and-domain incremental scenarios, i.e., when both inter-class and intra-class samples are not i.i.d.. Additionally, we evaluate MHs in a realistic setting of online continual learning (Mai et al., 2022b), assuming to continuously update the model on an on-the-edge device with limited computational capabilities, processing data in a single-pass, thus making the learning process more challenging. MHs allow to reach performances that are comparable or that overcome experience replay-based models and other models that store significantly larger pieces of information about the data stream.

To summarize, in this paper (i) we propose a classification head with self-developed memorization skills, referred to as Memory Head (MH), that can be trained to tackle continual learning problems, exploiting frozen or slowly changing pre-trained backbones. MHs are agnostic to the backbone architectures. (ii) MHs use an internal key-value architecture that alters the behavior of the neurons, selecting and blending different sets of weights based on the inputs, thus achieving effective parameter isolation and promoting memorization capabilities. (iii) MHs can address both shifts in class distributions as well as domain changes, without using buffers of stored examples and replay strategies. We also validate MHs in a continual online learning setting which is both class and domain incremental, showing that it overcomes related state-of-the art competitors, including those that replay experiences.

2 Related Works

Whilst CL techniques usually focus on training deep architectures from scratch (Parisi et al., 2019; Delange et al., 2021; Mai et al., 2022b), a recent branch of research activities is focused on exploring the benefits of leveraging pre-trained models in CL (Ostapenko et al., 2022). This approach has been proven to facilitate knowledge transfer and increase robustness to forgetting (Mehta et al., 2021; Cao et al., 2023; Marullo et al., 2023). There are two primary strategies for leveraging knowledge encoded in pre-trained weights. In the first one, a pre-trained backbone is kept unchanged, while all the learning efforts are dedicated solely to the development of classification heads or of specifically added parameters (Houlsby et al., 2019). A recent trend of works specifically consider the role of prompts in transformers (prompt-based methods), such as L2P (Wang et al., 2022b) and dual prompt (Wang et al., 2022a). Indeed, in the case of vision, prompt-based methods prepend a set of learnable parameters to the patch-features processed by pre-trained vision transformers (ViTs) (Dosovitskiy et al., 2021), enabling the model to encode task-specific information into the prompts. These methods are limited, by design, to be exploited in the framework of transformers, while here we propose architecture-agnostic heads with a learnable key-value mechanism. The second strategy for leveraging pre-trained models is to adapt/finetune the learnable parameters of both the backbone and the head, generally also replaying training samples (Ostapenko et al., 2022) or exploiting data statistics on the current task to better align the classification heads or the whole model (Zhang et al., 2023). MHs are fully compatible with these settings, even if they do not explicitly require any replays.

The recently proposed SLCA (Zhang et al., 2023) challenges previous efforts, showing that simply updating the backbone model at a slower pace with respect to the head, thus without replay buffers or other statistics, facilitates a gradual adjustment of the representations and a rapid adaptation of the classification head. However, SLCA also exploits a memory-demanding procedure based on feature covariance matrices of each class and the assumption of a gaussian distribution in the latent space. This poses a significant issue when such a single-mode-gaussian assumption is not well-supported, such as in domain incremental scenarios. Conversely, MHs are plastic enough to adapt to new properties of the input stream, still preserving what was learned in previously considered regions of the input space, thus supporting multi-modal distributions, even in a domain incremental setting. MHs do not perform replay and are capable by design of isolating computations.

The computational model of MHs involves (softly) isolated computations, thus they share relationships with parameter isolation methods. Such methods bypass task interference by allocating different parameters to each task (Aljundi

[2]This consideration is supported by preliminary experiments, and deepening this aspect goes beyond the scope of this work.

et al., 2019b; Mai et al., 2022b) or exploiting learned gating mechanisms to route computations toward different experts (Aljundi et al., 2017), generally leveraging task-related information (Delange et al., 2021). Similarly, in the context of conditional computation (Shazeer et al., 2017; Lin et al., 2019; Abati et al., 2020) parameters of the neural network are function of the input. However, clear task distinctions are needed in order to gain benefit from such an architectural design. Conversely, our approach is general and completely agnostic to the knowledge of the task boundaries, building a natural parameter isolation scheme by leveraging a learnable query-key-value mechanism.

The role of architecture components for CL has been investigated in several works by Mirzadeh et al. (2022a;b). In this paper, we rethink the low-level operating mechanisms of a neural layer, by introducing a novel neuron model explicitly designed to learn from information coming from a non-stationary distribution. MHs differ from models that build supervised prototypes of the input data, which strongly depend on class-label information (Ayub & Wagner, 2020; Ren et al., 2021). The neurons composing MHs share some intuitions with Sparse Mixture of Expert (MoEs) (Riquelme et al., 2021; Puigcerver et al., 2023) and winner-take-all strategies (or on top-k sparsity (Bricken et al., 2023)) for continual learning (Aljundi et al., 2018; Iyer et al., 2022). While the latter methods require to condition the top-k selection on an external signal (e.g., task-ID), the sparse routing mechanism of MHs is implicitly implemented by a query-key-value mechanism completely task-agnostic.

3 Memory Head

Backbones and Heads in CL. Let us consider a neural network f that processes an input pattern $x \in \mathbb{R}^d$, $d \geq 1$ (integer). We indicate with θ the learnable parameters of the model, and $f(x|\theta)$ returns the values computed by the c output units of the network. We decompose f into two parts, $f := h \circ b$, consisting of a backbone b with learnable parameters θ_b, that projects input patterns into a latent representation, and a classification head h with c neurons that maps such representations onto the output space, whose parameters are stored in θ_h, thus $\theta = (\theta_b, \theta_h)$. We indicate with $\phi \in \mathbb{R}^u$, $u \geq 1$ (integer), the representation yielded by the backbone, i.e.,

$$\phi = b(x|\theta_b), \quad o = h(\phi|\theta_h).$$

In this paper, we focus on the case in which the backbone b was pre-trained on a large-scale dataset, being it generic enough to be transferred to other related learning problems, as it is typical of modern foundation models (Yuan, 2023). This setting was recently studied in the context of continual learning, focusing on several aspects including the scale of the problem and the relationships between the pre-training domains and the target one (Ostapenko et al., 2022; Galashov et al., 2023; Zhang et al., 2023). On the one hand, the head h is still subject to forgetting when trained in a lifelong fashion. On the other hand, the already acquired representational skills of the backbone b yield stability in the process, reducing the variability in the input space of the head, thus favouring an easier development of h. Depending on the application setting in which the network f is instantiated, it could be useful to keep θ_b frozen to speed up the learning process and reduce resource usages, since no gradients are backpropagated though b, which is usually a very deep network. However, when possible, it has been recently investigated how slowly updating θ_b helps learn powerful classification heads, even if they are implemented with a single vanilla layer of neurons (Zhang et al., 2023). The latter setting has been shown to be enough to overcome replay methods (Ostapenko et al., 2022), and while the role of popular benchmarks is still controversial (Galashov et al., 2023), it clearly demonstrates that a pre-trained backbone can be effectively exploited in continual learning, even if h is simple.

Vanilla Head. We consider the case of h composed of a dense layer of neurons, discarding, to keep the notation simple, the role of the activation function and of the biases,

$$h(\phi|W) = W'\phi, \quad W \in \mathbb{R}^{u,c}, \tag{1}$$

being W' the transpose of the learnable parameters W. The i-th column of W defines the way the i-th neuron responds to its input, i.e., its behaviour, and it is progressively developed during the learning process, modeling a form of "memory" attached to the neuron and, when considering the whole W, to the neural layer.

Memory Heads. A Memory Head (MH) is a classification head composed of novel models of neurons, designed to augment the computational mechanism of Eq. 1 with increased memorization capabilities. MHs are devised to address continual learning problems, where learning from the currently streamed data should not affect what was learned in the past (i.e., avoid strong forgetting). In fact, in a continual learning setting, every ϕ which is provided to the head contributes in altering W, yielding catastrophic forgetting or suffering from limited plasticity. MHs go beyond this limit, introducing a computational mechanism that explicitly computes W based on the layer input ϕ. As a consequence, the neurons belonging to the MH dynamically determine which weights to consider, depending on the properties of what is provided to the network. MH introduces multiple learnable *memory units*, that are dynamically combined in order to yield the matrix W for a given ϕ. Such computational mechanism is designed to depend only

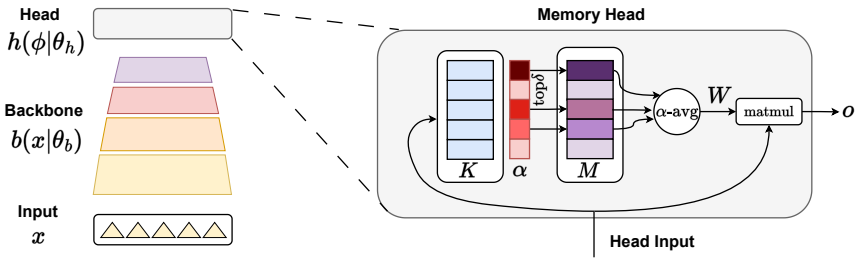

Figure 2: The proposed Memory Head. Given the representations ϕ extracted from a pre-trained backbone, the Memory Head is composed by a novel kind of neuron, composed by learnable keys K and memory units M, returning W as the outcome of blending (weighted average–avg) multiple memories in function of the their input ϕ.

on a subset of the available memory units, thus updating the resulting W will not alter those units that are out of such a subset, favouring an intrinsic form of parameter isolation. As a result, MHs can also react in significantly different manners to input stimuli distributed in different areas of the input space, due to the multiple input-dependent ways of combining the memory units to get W.

3.1 Model

MHs are based on a set of $m \geq 1$ learnable memory units, each in $\mathbb{R}^{u,c}$ (i.e., of the same size of the weight matrix W), and paired with a learnable u-sized key. An attention mechanism compares the input ϕ with the m keys, returning m attention scores, and memory units are blended according to such scores to yield the final W which is used to compute the layer output as in Eq. 1. Memory units associated to keys that got higher attention will have a more significant contribution to the final weight matrix W, while units with zero attention will not be used nor updated (Fig. 2). In order to keep the description more compact, we refer to the whole set of m memory units and keys as two matrices of learnable parameters, M and K, respectively. Matrix $M \in \mathbb{R}^{m,cu}$ is obtained by flattening the units and storing them row-wise, while matrix $K \in \mathbb{R}^{m,u}$ collects (row-wise) the m keys. MHs introduce a novel function ξ, which is responsible of evaluating the current input ϕ, comparing it with the keys in K, and blending the memory units collected in M with the goal of generating the weight matrix to be used. The head model of Eq. 1 becomes

$$h(\phi|K, M) = \xi(\phi, K, M)'\phi, \tag{2}$$

where ξ returns a weight matrix $\in \mathbb{R}^{u,c}$. The learnable parameters of h are K and M. Notice that ξ depends on the current input ϕ, keys K and memories M, and it is formally defined as

$$\xi(\phi, K, M) = \pi(M'\alpha(\phi, K, \delta)), \tag{3}$$

where the function α returns a vector with m scores, which determines the relevance of each memory unit according to the similarity between ϕ and the keys in K (δ will be described shortly) and π is a reshaping function to convert a flat vector of size cu into a $u \times c$ matrix. Notice the two-fold dependence on the layer input ϕ in Eq. 2.

The m scores returned by α are a distribution of attention scores, thus they are positive and sum to one, and such a function is characterized by three main properties suitable for continual learning purposes. $(i.)$ Attention scores are constrained to be sparse, with the parameter $\delta \in [1, m)$ defining the maximum number of non-zero entries. Sparsity is a crucial feature to implement a natural form of parameter isolation, that is typical of a large category of continual learning methods (Delange et al., 2021). In fact, it guarantees that the excluded $m - \delta$ memories, as they do not contribute to the output, will not be altered by the learning process, preserving stored information and mitigating catastrophic forgetting. $(ii.)$ Moreover, α is expected to return high values for keys "close" to ϕ, in the sense inducted by the way such similarity function is implemented. This means that similar layer inputs will trigger the same memory units and yield similar attention scores, while significantly different inputs will trigger different memories. More distant inputs might instead trigger sets of memory units that intersect, giving the same or different emphasis to the shared units, thus favoring a transfer of information between nearby regions of the input space, which is another important property in continual learning. $(iii.)$ Ideally, α should be computationally simple in practical implementations. Referring to the three listed requirements $i.$, $ii.$, and $iii.$, we define α by means of $\mathrm{softmax}^{\delta}$, which is the softmax function restricted to the top-δ logits, setting to 0 all the excluded components in the output of $\mathrm{softmax}^{\delta}$, thus ensuring the δ-sparsity requirement $(i.)$,

$$\alpha(\phi, K, \delta) = \mathrm{softmax}^{\delta}(\gamma \cdot \mathrm{sim}(\phi, K)), \tag{4}$$

where $\gamma > 0$ is a temperature parameter that tunes its sharpness and $\mathrm{sim}(\cdot)$ is a similarity function ($ii.$) that compares ϕ with the keys in K, and returning m similarity scores. There exist several different ways of implementing sim, such as the dot product (scaled by the square root of the key size), the cosine similarity, RBF kernels $\left[\exp\left(-\frac{1}{2\bar{\sigma}}\|\phi - K_i\|^2\right)\right]_{i=1}^{m}$, being $\|\cdot\|$ the L_2 norm, and others. In our experience, we used the RBF implementation in 2-dim cases, while the cosine similarity in all the other experiments of this paper. The neuron model in MH shares several intuitions that are largely used in the context of transformers, mixture of experts, kernel machines, and in the case of piece-wise activation functions. Moreover, it is easy to verify that the neurons in MHs formally extend vanilla neurons with augmented memory.

Proposition 1. *Classic neurons in neural networks are equivalent to the MH neurons with a single memory unit.*

Proof. When the number m of memory units is 1, α necessarily returns a single scalar equal to 1 for all ϕ and K, due to the constraint of returning m positive values that sum to 1. This makes ξ independent on ϕ and K. Moreover, since $m = 1$, M is a single-row matrix ($1 \times cu$), thus we get $\xi(\phi, K, M) = \pi(M)$, that, after reshaping by π, corresponds to W in Eq. 1, making Eq. 1 and Eq. 2 equivalent. □

3.2 Lifelong Learning with Memory Heads

We consider a data stream yielding at each time instant t a sample $x^{(t)}$ and a label $y^{(t)}$. In the running examples anticipated in Fig. 1, where we have 2D data and no backbones, it turns out that $\phi^{(t)} = x^{(t)}$, while the MH progressively processes the input information, with $c = 4$ output neurons (Fig. 1-bottom). Keys ($m = 8$) are represented by yellow dots, and randomly initialized, as well as memory units. The problem is class and domain incremental, and, as long as time passes, keys move over the centers of the modes in the data distribution without forgetting the properties of modes streamed in the earlier stages, while the neurons learn to exploit the blended memory units to make different predictions in different areas of the input space. The decision boundaries change over time according to the newly observed classes, but the background colors (representing class predictions) are preserved for previously processed data. In the case of classic neurons (Fig. 1-top), W is affected by catastrophic forgetting, so that the decision boundaries and the predictions strongly change over time, influenced by the last streamed class. To train MHs, in principle, gradient-based learning is immediately possible once we replace the classic neuron model with the one of Eq. 2. We refer to this gradient-based variant of MHs as MH_g. In MHs up to δ memory units are blended to generate the weight matrix, and a small δ allows a hard parameter isolation, for the way we implemented the function α. It might be also important, in some problems, to keep δ to a value larger than 1, to favor information transfer across units. However, this might lead to catastrophic forgetting since gradient steps alter all the δ involved units, with unforeseeable impact on previously learned information. Moreover, limiting the computations to the top δ memory units (Eq. 3) could end up emphasizing just a few common keys, and not using the other ones in K.

Conservative Updates. To overcome the first issue, we conservatively update only the memory unit (and key) associated to the largest attention score (independently of the value of δ), in a Winner-Take-All (WTA) fashion. In this way, MHs still benefit from information transfer during inference (where all the δ units are involved), while they specialize only the "winning" memory unit (key) during training. The second issue suggests rethinking the procedure to update K, not only in order to avoid degenerate cases (where a single key is used) but also to bias the update dynamics in function of the requirements of continual learning. Keys can be interpreted as "representatives" of the data distribution, that should be located in different regions of the layer input space. In this way, MH can select different memory units depending on where the sample lies in such space. We propose to update the keys in K with an online WTA procedure that is inspired by online K-Means (Zhong, 2005). Consistently with the selected implementation of the similarity function in Eq. 4, if $K_\dagger$ is the winning key for the current ϕ (i.e., the $\dagger$-th row of K), then

$$K_\dagger = K_\dagger + \beta \nabla \mathrm{sim}(\phi, K_\dagger), \tag{5}$$

where $\beta > 0$ tunes the strength of the update operation. Eq. 5 is equivalent to a gradient step in the direction that maximizes the similarity between ϕ and the winning key, so that the winning key $K_\dagger$ is slightly moved toward the current ϕ. Attention scores are refreshed according to the updated K. We denote this online K-Means based variant with MH_{ok}.

Maximizing Resource Usage. To favor the development of all the available keys, a MH_{ok} computes basic statistics regarding how keys are exploited. Namely, we consider the *absolute usage* μ and the *absolute age* η, that are vectors with m components (one-per-key). The former counts how many times each key was a winning key (usage), while the latter counts the number of time steps since each key won (age). We use these statistics to suggest possible replacements. While it is typical of many K-Means like algorithms (Zhong, 2005) to re-init unused representatives based on "usage", here we also consider their "age", given its importance in a continual learning setting. A young key might not be used much due to its recent introduction, while a largely used key might not be updated for a very long time because of strong and permanent changes in the data distribution. In detail, the i-th key is marked as not used

enough if $\mu_i < \tau^\mu$, while it is too old if $\eta_i \geq \tau^\eta$, given two custom positive thresholds τ^μ and τ^η. We label as *weak* a key that fulfills both the conditions, and we replace it with a re-initialized key/memory-unit to favour the usage of all the available resources (more details in Appendix A).

4 EXPERIMENTS

MHs are agnostic to the specific type of continual learning problem, differently from many approaches and CL algorithms in existing literature (task, class, domain incremental, etc. (van de Ven et al., 2022; Delange et al., 2021)). The code for MHs can be found at https://github.com/sailab-code/memory_head.

4.1 CLASS-INCREMENTAL LEARNING

We begin our investigation in the popular class-incremental setting, following the experimental setup in Zhang et al. (2023), where both the backbone (a ViT) and the head are finetuned for multiple epochs.

Datasets & Metrics. We follow Zhang et al. (2023), making comparisons in CIFAR100 (Krizhevsky, 2009) (100 classes, 500 samples per class), CUB200 (Wah et al., 2011) (200 classes, 60 samples per class – 30/30 training/test), Imagenet-R (Hendrycks et al., 2021) (200 classes, 24k training, 6k test), Cars196 (Krause et al., 2013) (196 classes, 8k training, 8k test). Each dataset is randomly split into $\mathcal{T} = 10$ disjoint tasks having the same amount of classes. Imagenet-R is acknowledged to have a large domain gap with respect to ImageNet, containing data of different styles, such as cartoon, graffiti and origami with a significant intra-class diversity. Following Zhang et al. (2023), we report the average accuracy after learning the last task $\mathcal{T}$, referred to as $\mathcal{A}_\mathcal{T}$, and the average performance $\mathcal{A}_{\text{avg}} = \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} \mathcal{A}_t$.

Compared Models & Training Details. We compare the performance of our model with the most representative regularization-based approaches (EWC (Kirkpatrick et al., 2017), LwF (Li & Hoiem, 2017)), prompt based approaches (L2P (Wang et al., 2022c), DualPrompt (Wang et al., 2022a)), the Slow Learner (SL) proposed by Zhang et al. (2023) both in the vanilla variant and when empowered by the memory-based class alignment technique (SLCA), and replay-based methods (BiC (Wu et al., 2019), DER++ (Buzzega et al., 2020), GDumb (Prabhu et al., 2020)) exploiting a memory buffer of 1000 samples. We report the case of joint training as upper bound performance, as in Zhang et al. (2023). We adopt a Vision Transformer (ViT) (Dosovitskiy et al., 2021) pre-trained on ImageNet21K (Russakovsky et al., 2015) (ViT-B/16-IN21K) as the backbone $b(x|\theta_b)$ for all the baselines, as commonly done in pretrained-CL literature (Zhang et al., 2023). Following SLCA, we finetune the whole deep model f using a smaller learning rate ρ for the backbone ($\rho_{\theta_b} = 10^{-4}$) with respect to the one exploited for the classification head, $\rho_{\theta_h} = 10^{-2}$. SGD optimizer is used for all the baselines, except for the prompt-based methods that are trained via Adam (Zhang et al., 2023). Learning happens for multiple epochs $\mathcal{E}$ for each task ($\mathcal{E} = 20$ for CIFAR100 and $\mathcal{E} = 50$ for the other datasets), with 128-sized batches. When learning each incremental task, only the head-parameters corresponding to the classes of the current task are trained together with the backbone. Following Zhang et al. (2023), for a fair comparison, all the baseline approaches are trained with the Slow Learner approach, except the prompt-based methods, which keep the backbone fixed. In Zhang et al. (2023), a new head is added for each new task. We perform the same to establish a fair comparison. In our case, we set the number of memories to $m = 10$ for each task. See Appendix E for details on hyper-parameter tuning.

Results. We report in Tab. 1 the performances achieved by MHs in two variants (i.e., the main proposed one, indicated with MH_{ok}, and when keys are learned by gradient descent, MH_g), with methods that, as well as MHs, do not leverage rehearsal buffers. Both the proposed MH variants outperform the competitors in almost all the benchmarks (3/4 in the case of MH_g, 4/4 for MH_{ok}), both when considering the accuracy after learning the last task, $\mathcal{A}_\mathcal{T}$, and the average incremental accuracy $\mathcal{A}_{\text{avg}}$. We remark that the proposed online K-Means based MH-variant, MH_{ok}, is indeed capable to increase the performances attained by MHs trained with standard gradient-descent. In datasets characterized by a large domain gap with respect to the pre-train setting (Imagenet-R, CUB200), the MHs are capable to effectively steer the backbone knowledge and foster its generalizability, by routing computations and isolating specific keys. Additionally, we report in Fig. 3 the performance of the proposed MH against replay-buffer methods. Even though competitors leverage an explicit buffer of 1000 samples that are replayed at each training step, MH achieves similar or higher results (CIFAR100, ImageNet-R, CUB200) without storing any exemplar. The key-memory mechanism characterizing MHs learns keys capable to route the computations towards appropriate memory banks for the current pattern. Notice that MHs use 10 memories per task ($|\mathcal{T}| = 10$), that is one order of magnitude less than the 1000-exemplar buffers. The novel neuron model composing MHs is general and can be paired with other existing CL techniques. As a proof-of-concept, in Tab. 2 we report the results obtained when combining MHs with the Class-Alignment (CA) procedure proposed by Zhang et al. (2023). CA performs 5 additional epochs at the end of each task, on which it aligns the statistics of previously-learned classes in a post-hoc fashion, by assuming a gaussian distribution

Table 1: Class-Incremental setting, main results, comparing with models in Zhang et al. (2023) (same setting). Average and stds over three runs.

Method	CIFAR-100		ImageNet-R		CUB200		Cars196	
	$\mathcal{A}_T$	$\mathcal{A}_{\text{avg}}$	$\mathcal{A}_T$	$\mathcal{A}_{\text{avg}}$	$\mathcal{A}_T$	$\mathcal{A}_{\text{avg}}$	$\mathcal{A}_T$	$\mathcal{A}_{\text{avg}}$
Joint-Training	93.22±0.16	-	79.60±0.87	-	88.00±0.34	-	80.31±0.13	-
L2P	82.76±1.17	88.48±0.83	66.49±0.40	72.83±0.56	62.21±1.92	73.83±1.67	38.18±2.33	51.79±4.19
DualPrompt	85.56±0.33	90.33±0.33	68.50±0.52	72.59±0.24	66.00±0.57	77.92±0.50	40.14±2.36	56.74±1.78
EWC	89.30±0.23	92.31±1.66	70.27±1.99	76.27±2.13	68.32±2.64	79.95±2.28	52.50±3.18	64.01±3.25
LwF	87.99±0.05	92.13±1.16	67.29±1.67	74.47±1.48	69.75±1.37	80.45±2.08	49.94±3.24	63.28±1.11
SL	88.86±0.83	92.01±1.71	71.80±1.45	76.84±1.26	68.07±1.09	79.04±1.69	49.74±1.25	62.83±2.16
MH_g	89.00±0.13	**93.24**±0.06	75.48±0.26	81.08±0.94	72.19±1.27	81.53±0.81	51.93±2.97	63.38±0.65
MH_{ok}	**89.56**±0.83	92.64±0.91	**77.10**±0.22	**81.69**±1.31	**78.17**±1.52	**85.99**±0.72	**56.75**±1.79	**66.88**±0.67

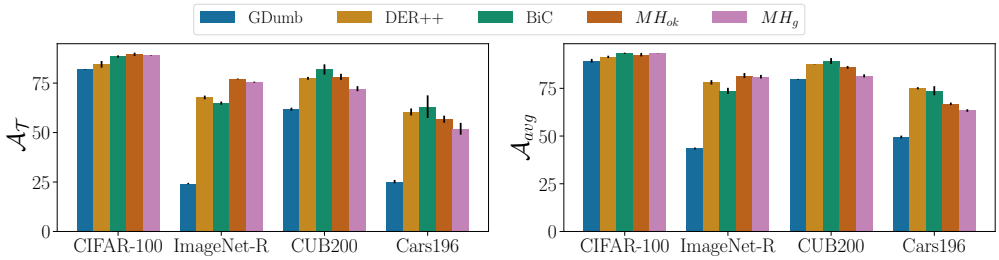

Figure 3: Class-Incremental setting, performances against replay-based competitors.

in the latent space and by storing the mean features and covariance matrix of each single class. Tab. 2 shows that also MHs benefit from CA (when compared with Tab. 1), even if it is not designed for MHs and it is memory demanding.

4.2 Online Class-domain Incremental Learning

Several CL settings are built upon some assumptions made in the learning phase that might be computational or memory demanding (i.e. storing past exemplars (Rolnick et al., 2019), fine-tuning a large backbone, large mini-batches, multiple passes (Zhang et al., 2023), known task boundaries (Shazeer et al., 2017; Lin et al., 2019)). However, in realistic scenarios such as training directly on the edge on devices with limited capabilities, these assumptions may not hold or might be unfeasible to pursue. To meet these stricter requirements, we assume that data samples are presented only once to the network and that updates are made one sample at a time (i.e., batch size 1). To simulate a computationally constrained scenario we leverage a relatively small pre-trained backbone, namely a ResNet50 ($\approx$ 25 million parameters vs the $\approx$ 86 million parameters of the ViT used in Sec. 4.1), and we keep it fixed, by only finetuning the memory head. The data stream, indexed with time instants in $[1, T]$, is partitioned into N non-overlapping intervals, $[1, T] = \cup_{j=1}^{N} I_j$, with $I_j \cap I_r = \emptyset$, $\forall j, r$, featuring data sampled from different distributions. We simulate the class domain incremental setting, exploiting data from c classes, each sampled from q different distributions: after the first pass on all the classes, the process is repeated, showing the classes in the same order, yet sampled from different distributions, for a total of $N = cq$ distributions.

Datasets & Metrics. First, we investigate this learning framework using a 2D dataset in line with Fig. 1, referred to as MODES, with 1000 samples generated from $c = 4$ mutually exclusive bi-modal distributed ($q = 2$) categories. We just use MH without any backbone, processing raw data coordinates. Then, moving to a more realistic setting we perform experiments on the challenging NonStationary-MiniImagenet (NS-IMAGENET) dataset (Mai et al., 2022b). The dataset was proposed for continual online learning and designed for the class and domain incremental setting. It is composed of images belonging to $c = 100$ classes, at the resolution of 84×84. We consider $q = 3$ domains, sampling data from the original distribution plus two perturbed distributions adding gaussian noise (see Fig. 5). In total, we obtain 180K images, 20% of which are used for testing and 10% compose a validation set. For all experiments, performance is measured using common metrics, as defined in (Mai et al., 2022b): average accuracy at time T (indicated with $\mathcal{A}_T$ for coherence with the previous experiments), average forgetting at time T, as well as forward transfer (backward transfer was not significant in our comparisons) – see Appendix C.

Compared Models & Training Details. We compared MH against models with memory buffers, used either for continuous replay purposes, such as Experience Replay (ER) (Rolnick et al., 2019), using Random Sampling (RND) or the more advanced Reservoir (RES) Sampling (Chrysakis & Moens, 2020), or to set up the GDUMB approach (Prabhu et al., 2020), that retrains the network only with class-wise balanced data from the buffer. We also compared to state-of-the-art continual learning approaches such as MIR (Aljundi et al., 2019a), which features specifically designed

Table 2: Performances obtained with Class Alignment (Zhang et al., 2023), *not designed for MHs.*

Method	CIFAR-100		ImageNet-R		CUB200		Cars196	
	$\mathcal{A}_T$	$\mathcal{A}_{\text{avg}}$	$\mathcal{A}_T$	$\mathcal{A}_{\text{avg}}$	$\mathcal{A}_T$	$\mathcal{A}_{\text{avg}}$	$\mathcal{A}_T$	$\mathcal{A}_{\text{avg}}$
SLCA	**91.53**±0.28	**94.09**±0.87	77.00±0.33	81.17±0.64	84.71±0.40	90.94±0.68	**67.73**±0.85	**76.93**±1.21
MH_g + CA	90.77±0.46	93.80±0.97	78.01±0.36	82.62±0.92	83.94±0.36	90.06 ±1.15	65.72±1.60	73.89±1.06
MH_ok + CA	91.24±0.44	94.02±1.15	**78.72**±0.30	**83.01**±0.81	**84.93**±0.38	**91.12**±1.05	67.01±1.78	75.77±0.94

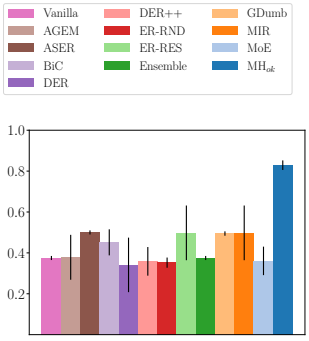

Figure 4: Accuracy $\mathcal{A}_T$ in MODES, online class-domain incremental.

Table 3: NS-IMAGENET, online class-domain incr.

Method	NS-ImageNet		
	$\mathcal{A}_T \uparrow$	FORGET ↓	FWD. T. ↑
Vanilla	6.8±0.6	13.2 ± 0.6	1.6 ± 0.1
AGEM	6.4±0.3	93.9 ± 0.3	2.0 ± 0.1
ASER	54.0±0.2	32.3 ± 0.4	21.0 ± 0.3
BIC-kd	13.0±0.6	1.1 ± 0.1	4.6 ± 0.3
DER	46.5±4.9	53.5 ± 5.0	15.6 ± 2.1
DER++	55.9±1.0	43.4 ± 1.1	25.0 ± 0.4
ER-Random	22.4±0.1	76.8 ± 0.3	9.0 ± 0.1
ER-Reservoir	49.9±0.3	49.4 ± 0.3	21.5 ± 0.0
Ensemble	22.9±2.2	35.2 ± 1.5	4.1 ± 0.4
Gdumb	6.0±0.6	61.4 ± 1.3	4.7 ± 0.0
MIR	53.7±0.3	45.8 ± 0.3	24.4 ± 0.2
MoE	1.7±0.7	86.3 ± 1.7	1.2 ± 0.0
MH_{ok}	**57.8**±0.1	24.3 ± 0.1	24.2 ± 0.1

retrieval operations on the buffer, and A-GEM (Chaudhry et al., 2019), with gradient-based memories, ASER (Shim et al., 2021), BIC-inspired Knowledge Distillation (Mai et al., 2022a), DER and DER++ (Buzzega et al., 2020). These methods represent the SOTA in online continual learning without explicit task boundaries (Mai et al., 2022a; Wang et al., 2023). We also considered ENSEMBLE models (Zhou, 2012), where the outputs of 10 networks are combined either by averaging them or learning a gating function (Mixture of Experts (MOE) (Shazeer et al., 2017)), and VANILLA networks with classic neurons. We tune hyper-parameters on a validation set after half of the length of the whole stream (Cai et al., 2021), differently from approaches that identify the best models on test data (Lopez-Paz & Ranzato, 2017; De Lange & Tuytelaars, 2021a). In Appendix D we formally analyze and compare the memory and computational burden of MHs vs. classic-nets, to set up a fair comparison (Zhou et al., 2023). In MODES we set $m = 8$ memory units, and we equipped competitors with a buffer of size 8. Notice that even when replaying only a sample per step, the computational burden of the competitors is always larger than MH. In NS-IMAGENET we considered $m \in \{10, 25, 50, 100\}$ (see Appendix E for details on hyper-parameters), while competitors leverage a 1000-exemplars buffer sampled with a batch size of 10 samples, following the experience of Mai et al. (2022b).

Results. Results in MODES are shown in Fig. 4, reporting the accuracy at time T, i.e., at the end of the data stream. Our MH, thanks to its pool of weights, can effectively separate the data, achieving an average accuracy that doubles the performance of most competitors. It can be noted how the vanilla networks are not able to capture the non-stationary nature of the data distributions, while some competitors equipped with memory buffers can almost double their performance. The high accuracy of our model confirms the capability of MHs to incrementally memorize the properties of the data distributions effectively. Moving to NS-IMAGENET, we report the results in Tab. 3. It can be seen that the backbone, equipped with our MH, achieves the highest accuracy across all competitors. Interestingly, only our model and MIR, DER++, ASER, and ER-RESERVOIR can provide appreciable forward transfer capabilities. This confirms that the weights learned by our method can indeed be shared across distributions and effectively used throughout training. More importantly, MH is the only strategy that does not dramatically suffer from forgetting issues, while also retaining a high accuracy. In fact, VANILLA and BIC-KD report almost no forgetting, which can be traced to the fact that such models never obtain sufficiently high accuracy in any task. MHs allow the model to achieve a certain degree of forward transfer, comparable to DER++ and MIR.

Ablations. We perform an ablation study regarding key/memory updates. In Fig. 6 (a-b) MH_a is a global update of memories corresponding to *a*ll the top-δ keys instead of the best-scoring one (as in the WTA approach). Similarly, MH_g is the already defined *g*radient-based update of the keys. The MH_a variant favors forgetting since multiple sets of memories are updated at the same time, leading to lower accuracies. Gradient-based key updates yield a bigger accuracy drop with more forgetting. We attribute this to a less predictable behavior of keys when using gradient-based updates as they are modulated by the quality of the predictions since gradients scale with the loss. This can lead to unwanted behaviors with respect to Eq. 5, such as too big updates or no update at all, depending on the fitting of each sample. As one would expect, combining both MH_g and MH_a does not help. In Fig. 6 (c-d) we investigated the role

Figure 5: Samples from NS-IMAGENET (Mai et al., 2022b). Original samples (1st) are perturbed to create different distributions with Gaussian noise with noise factor $\texttt{nf}= 0.25$ and standard deviation $\sigma = 0.1$ (2nd) and $\sigma = 0.2$ (3rd).

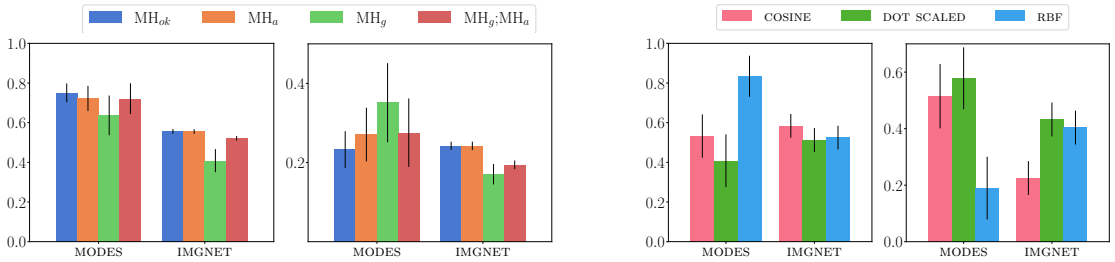

Figure 6: Ablation studies. (a-b): modified versions of MH. MH$_a$ is a global update (instead of winner-take-all) that involves the *M*emories corresponding to *a*ll the top-δ keys; MH$_g$ indicates the *g*radient-based update of the *K*ey. (c-d): similarity functions. Average accuracy at T (a-c, *higher* is better) and forgetting at T (b-d, *lower* is better).

of function sim (Eq. 4). In MODES the RBF kernel provides the best performance, as expected. On NS-IMAGENET instead the cosine similarity helps with the high dimensionality of the data extracted by the backbone.

Limitations. The ability of MHs to isolate computations strongly depends on the number of keys/memory units and on the measure of similarity used for comparisons, that requires model selection. If too few keys are chosen, the model does not isolate computations and may be subject to forgetting. On the contrary, if too many keys are selected, the computational burden increases (linearly), and it might yield an excessive isolation, not favoring the transfer of information between different memory units. On one hand, as it is typical in machine learning algorithms, including continual learning approaches, the optimal trade-off depends on the considered problem and on the available resources. On the other hand, in future work we plan to design a dynamic mechanism that expands the model according to needs, to try to mitigate this issue. Another limitation of MHs is due to the dynamics of the key update mechanism. If the data changes at a fast time scale, or if the backbone is not pretrained (for example), then representations will significantly change at a fast time scale, making it more challenging to find a stable configuration for the keys.

5 Conclusions and Future Work

This paper proposed Memory Head, a dynamic classification head to be exploited with pre-trained backbones in continual learning. MHs are built upon memory units indexed in an input-dependent manner akin to attention models, and exhibit self-organized memorization capabilities that surpass traditional neurons. The neurons composing MHs can generate their weights based on the current input, aiming for plasticity but less subject to catastrophic forgetting issues. We evaluated MHs with both frozen and fine-tuned backbones, achieving competing performances against recent state-of-the-art competitors. We tested MHs in class-incremental learning and in the less explored and demanding scenario of continual online learning and class-domain incremental setting. MHs do not introduce any restriction to the domains that can be tackled and can be paired with any CL techniques, an exploration that we leave for future works, as long as the development of architectures entirely composed by neurons having memorization capacity.

References

Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3931–3940, 2020.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3366–3375, 2017.

Rahaf Aljundi, Marcus Rohrbach, and Tinne Tuytelaars. Selfless sequential learning. *arXiv preprint arXiv:1806.05421*, 2018.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019b.

Ali Ayub and Alan R Wagner. Cognitively-inspired model for incremental learning using a few examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 222–223, 2020.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Trenton Bricken, Xander Davies, Deepak Singh, Dmitry Krotov, and Gabriel Kreiman. Sparse distributed memory is a continual learner. *arXiv preprint arXiv:2303.11934*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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*, 33:15920–15930, 2020.

Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8281–8290, 2021.

Boxi Cao, Qiaoyu Tang, Hongyu Lin, Xianpei Han, Jiawei Chen, Tianshu Wang, and Le Sun. Retentive or forgetful? diving into the knowledge memorizing mechanism of language models. *arXiv preprint arXiv:2305.09144*, 2023.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2019.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9640–9649, October 2021.

Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International Conference on Machine Learning*, pp. 1952–1961. PMLR, 2020.

Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8250–8259, October 2021a.

Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8250–8259, 2021b.

M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. doi: 10.1109/TPAMI.2021.3057446.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2021.

Alexandre Galashov, Jovana Mitrovic, Dhruva Tirumala, Yee Whye Teh, Timothy Nguyen, Arslan Chaudhry, and Razvan Pascanu. Continually learning representations at scale. In *Conference on Lifelong Learning Agents*, pp. 534–547. PMLR, 2023.

Nuwan Gunasekara, Bernhard Pfahringer, Heitor Murilo Gomes, and Albert Bifet. Survey on online streaming continual learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 6628–6637, 2023.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Abhiram Iyer, Karan Grewal, Akash Velu, Lucas Oliveira Souza, Jeremy Forest, and Subutai Ahmad. Avoiding catastrophe: Active dendrites enable multi-task learning in dynamic environments. *Frontiers in neurorobotics*, 16: 846219, 2022.

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. *Proc. of the National Academy of Sciences*, 2017.

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*, pp. 554–561, 2013.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Min Lin, Jie Fu, and Yoshua Bengio. Conditional computation for continual learning. *arXiv preprint arXiv:1906.06635*, 2019.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022a. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.10.021. URL https://www.sciencedirect.com/science/article/pii/S0925231221014995.

Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022b.

Simone Marullo, Matteo Tiezzi, Marco Gori, Stefano Melacci, and Tinne Tuytelaars. Continual learning with pre-trained backbones by tuning in the input space. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, 2023. doi: 10.1109/IJCNN54540.2023.10191069.

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.

Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Huiyi Hu, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Wide neural networks forget less catastrophically. In *International Conference on Machine Learning*, pp. 15699–15717. PMLR, 2022a.

Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning. *arXiv preprint arXiv:2202.00275*, 2022b.

Oleksiy Ostapenko, Timothee Lesort, Pau Rodriguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay. In *Conference on Lifelong Learning Agents*, pp. 60–91. PMLR, 2022.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 524–540. Springer, 2020.

Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts. *arXiv preprint arXiv:2308.00951*, 2023.

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 *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Vincent Hassen, Anderson Schneider, et al. Lag-llama: Towards foundation models for time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

Mengye Ren, Tyler R Scott, Michael L Iuzzolino, Michael C Mozer, and Richard Zemel. Online unsupervised learning of visual representations and categories. *arXiv preprint arXiv:2109.05675*, 2021.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Intl. Journal of Computer Vision (IJCV)*, 2015.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9630–9638, 2021.

Gido M van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, pp. 1–13, 2022.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9 (11), 2008.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*, 2023.

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 *European Conference on Computer Vision*, pp. 631–648. Springer, 2022a.

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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022b.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 139–149, 2022c.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 374–382, 2019.

Yang Yuan. On the power of foundation models. In *International Conference on Machine Learning*, pp. 40519–40530. PMLR, 2023.

G. Zhang, L. Wang, G. Kang, L. Chen, and Y. Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. pp. 19091–19101, oct 2023. doi: 10.1109/ICCV51070.2023.01754. URL https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.01754.

Shi Zhong. Efficient online spherical k-means clustering. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 5, pp. 3180–3185. IEEE, 2005.

Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=S07feAlQHgM.

Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

A Replacing Weak Keys

We consider the vector μ to be initialized with zeros, while the entries in η are set to τ^η, so that, at the beginning, all keys are weak. Before updating keys with Eq. 5, if there exists at least a weak key and ϕ is distant from $K_\dagger$, i.e., the maximum similarity score returned by sim is smaller than τ^α (indicating that ϕ is somewhat different from all the entries of K), the MH replaces the weak key with ϕ itself, zeroing its usage counter.[3] Moreover, instead of randomly re-initializing the memory unit associated to the replaced key, the MH initializes it to $M_\dagger$, to favour a warm-start of the memory values. Algorithm 1 summarizes continual learning in a MH (see Appendix B for the case of mini-batches). As already previously introduced, given a matrix Z the notation Z_i indicates the i-th of a matrix. For example, in the case of memory units, M_i is the i-th memory unit (which is stored, by definition, into the t-th row of matrix M).

Algorithm 1 Learning with a Memory Head: WTA updates of memory units and keys.

Require: Stream $\mathcal{S}$; generic loss function $\mathrm{loss}(\dots)$; learning rates ρ, β; $K \leftarrow \mathrm{rand}$, $M \leftarrow \mathrm{rand}$, $\mu \leftarrow 0$'s, $\eta \leftarrow \tau^\eta$'s.

while $true$ **do**
- $\phi \leftarrow \mathrm{next\_mh\_input}(\mathcal{S})$
- $a, s \leftarrow \alpha(\phi, K, \delta)$ ▷ Attn. scores (Eq. 4) and similarities
- $\dagger \leftarrow \arg\max_{j \in \{1,\dots,m\}} \{a_j\}$ ▷ Index of the winning key
- $K, M, \dagger, \mu \leftarrow$
  - $\textsc{Scramble}(\phi, K, M, s, \dagger, \mu, \eta)$ ▷ Possibly replace a key
- $K_\dagger \leftarrow K_\dagger + \beta \nabla \mathrm{sim}(\phi, K_\dagger)$ ▷ Upd. winning key, Eq. 5
- $a \leftarrow \textsc{Refresh}(\phi, s, K_\dagger, \dagger)$ ▷ Refresh a (dummy)
- $\mu_\dagger \leftarrow \mu_\dagger + 1, \ \eta_\dagger \leftarrow 0$ ▷ Increase $K_\dagger$ usage, reset its age
- $\eta = \eta + 1$ ▷ Increase *all* ages
- $W \leftarrow \pi(M'a)$ ▷ Generate weights, Eq. 3
- $o \leftarrow W'\phi$ ▷ Compute output, Eq. 2
- $M_\dagger \leftarrow M_\dagger - \rho \nabla_{M_\dagger} \mathrm{loss(o)}$ ▷ Upd. winning memory unit

end while

function $\textsc{Scramble}(\phi, K, M, s, \dagger, \mu, \eta)$
- $\mathcal{W} \leftarrow \{z : \mu_z < \tau^\mu \wedge \eta_z \geq \tau^\eta\}$ ▷ Indices of weak keys
- **if** $s_\dagger < \tau^\alpha \wedge \mathcal{W} \neq \emptyset$ **then** ▷ If the current match is loose
  - $j \leftarrow \arg\max_{z \in \mathcal{W}} \{\eta_z\}$ ▷ The weakest key is the oldest
  - $K_j \leftarrow \phi$ ▷ Replace weakest key
  - $M_j \leftarrow M_\dagger$ ▷ Warm-start for replaced memory unit
  - $\mu_j \leftarrow 0$ ▷ Reset usage counter
  - $\dagger \leftarrow j$ ▷ Update winning key index
- **end if**

return $K, M, \dagger, \mu$

end function

B Batched Computations

In Algorithm 2 we report the same operations of Algorithm 1 when a mini-batch of size b is provided by the stream at each time instant. The case of batched computations is pretty straightforward, with a major difference in the key

[3] There might be multiple weak keys. We select the oldest one.

scrambling routine. As a matter of fact, scrambling would require to serialize the processing of the element in the mini-batch, that is not a desirable feature. For this reason, we restrict to 1 the number of weak keys that can be potentially replaced for each mini-batch, selecting the one associated to the batch element that is returning the smallest similarity score with respect to K.

Algorithm 2 Learning with a Memory Head when a batch of b samples is provided at each time instant by the stream. Notice that Φ is the mini-batch matrix ($b \times u$), function α is intended to compute attention scores for each element of the batch, $\dagger$ and o are now arrays of length b. Scrambling involves up to 1 key for each mini-batch (function SCRAMBLEONE).

Require: Stream $\mathcal{S}$; generic loss function $\text{loss}(\dots)$, learning rates ρ, β; $K \leftarrow \text{rand}$, $M \leftarrow \text{rand}$, $\mu \leftarrow 0$'s, $\eta \leftarrow \tau^\eta$'s.
while $true$ **do**
- $\Phi \leftarrow \text{next\_mh\_inputs}(\mathcal{S})$ ▷ Φ is a $b \times u$ matrix
- $A, S \leftarrow \alpha(\Phi, K, \delta)$ ▷ A and S are $b \times m$ matrices
- $\dagger_h \leftarrow \arg\max_{j \in \{1,\dots,m\}} \{A_{hj}\},\ h = 1, \dots, b$ ▷ Indices of the winning keys
- $K, M, \dagger, \mu \leftarrow \text{SCRAMBLEONE}(\Phi, K, M, S, \dagger, \mu, \eta)$ ▷ Possibly replace a weak key
- $K_{\dagger_h} \leftarrow K_{\dagger_h} + \beta \nabla \text{sim}(\Phi_h, K_{\dagger_h}),\ h = 1, \dots, b,$ ▷ Upd. winning keys, Eq. 5
- $A \leftarrow \alpha(\Phi, K, \delta)$ ▷ Refresh attention (from scratch)
- $\mu_{\dagger_h} \leftarrow \mu_{\dagger_h} + 1, \quad \eta_{\dagger_h} \leftarrow 0, \quad h = 1, \dots, b$ ▷ Increase winning keys usages, reset ages
- $\eta = \eta + b$ ▷ Increase *all* ages
- $W \leftarrow AM$ ▷ Generate weights, Eq. 3, W is a $b \times cu$ matrix
- $o_h \leftarrow \pi(W_h)\Phi'_h, \quad h = 1, \dots, b$ ▷ Compute output, Eq. 2
- $M_\dagger \leftarrow M_\dagger - \rho \nabla_{M_\dagger} \text{loss(o)}$ ▷ Upd. winning memory unit

end while

function SCRAMBLEONE($\Phi, K, M, S, \dagger, \mu, \eta$)
- $\mathcal{W} \leftarrow \{z : \mu_z < \tau^\mu \wedge \eta_z \geq \tau^\eta\}$ ▷ Set of weak keys (if any)
- $k \leftarrow \arg\min_{h \in \{1,\dots,b\}} \{S_{h_{\dagger_h}}\}$ ▷ Idx of the sample with lowest similarity to its winning key
- **if** $S_{k\dagger_k} < \tau^\alpha \wedge \mathcal{W} \neq \emptyset$ **then** ▷ If the current match is loose
  - $j \leftarrow \arg\max_{z \in \mathcal{W}} \{\eta_z\}$ ▷ The weakest key is the oldest
  - $K_j \leftarrow \Phi_k$ ▷ Replace weakest key with data sample
  - $M_j \leftarrow M_\dagger$ ▷ Warm-start for replaced memory unit
  - $\mu_j \leftarrow 0$ ▷ Reset usage counter
  - $\dagger_k \leftarrow j$ ▷ Update winning key index
- **end if**

return $K, M, \dagger, \mu$
end function

C METRICS

Following the notation of Section 4.2, we are given a data stream $\mathcal{S}$ partitioned into non-overlapping time intervals, and we indicate with t_j the last time instant of the j-th interval I_j, with $t_N = T$. In the following description, we assume class indices to be ordered with respect to the time in which they become available, to keep the notation simple. We indicate with $p(x|I_i)$ the data distribution in the i-th interval, with θ_j is the model developed up to t_j, while D_i is an held out test set with data sampled from $p(x|I_i)$. Then, we indicate with

$$\text{acc}_i^{\theta_j} = \text{accuracy}(\mathcal{D}_i, \theta_j)$$

the accuracy on data sampled from $p(x|I_i)$ computed using the model parameters at t_j, i.e., θ_j. We collect the following matrix of accuracies during the learning procedure,

Model/Test Data	$\mathcal{D}_1$	$\mathcal{D}_2$	$\dots$	$\mathcal{D}_j$	$\dots$	$\mathcal{D}_N$
θ_1	$\text{acc}_1^{\theta_1}$	$\text{acc}_2^{\theta_1}$	$\dots$	$\text{acc}_h^{\theta_1}$	$\dots$	$\text{acc}_N^{\theta_1}$
θ_2	$\text{acc}_1^{\theta_2}$	$\text{acc}_2^{\theta_2}$	$\dots$	$\text{acc}_j^{\theta_2}$	$\dots$	$\text{acc}_N^{\theta_2}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
θ_j	$\text{acc}_1^{\theta_j}$	$\text{acc}_2^{\theta_j}$	$\dots$	$\text{acc}_j^{\theta_j}$	$\dots$	$\text{acc}_N^{\theta_j}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
θ_N	$\text{acc}_1^{\theta_N}$	$\text{acc}_2^{\theta_N}$	$\dots$	$\text{acc}_j^{\theta_N}$	$\dots$	$\text{acc}_N^{\theta_N}$

that we indicate as continual confusion matrix (CCM, being CCM_j the matrix up to t_j), and we exploit it to compute the following measures. Notice that it is a square matrix.

- The average accuracy at t_z is defined as the average of the z-th row of the CCM, up to the z-th column (included),

$$\text{avg\_accuracy}(\text{CCM}_z) = \frac{1}{z} \sum_{i=1}^{z} \text{acc}_i^{\theta_z},$$

and we commonly measure the average accuracy at the end of training, $t_z = t_N$ ($\mathcal{A}_T$ of Section 4).

- The average forgetting at t_z can be defined as

$$\text{avg\_forgetting}(\text{CCM}_z) = \frac{1}{z-1}\sum_{i=1}^{z-1}\left(\text{acc}_i^\star - \text{acc}_i^{\theta_z}\right),$$

where $\text{acc}_i^\star$ is the best accuracy obtained on data $\mathcal{D}_i$ so far, i.e., $\max_{\theta_k \in \{\theta_1,\ldots,\theta_{z-1}\}} \text{acc}_i^{\theta_k}$ (maximum of the i-th column up to row $z-1$). We commonly measure the average forgetting at the end of training, $t_z = t_N$.

- Forward transfer measures how learning at checkpoint t_z (positively) influences predictions on data introduced in future intervals,

$$\text{forward}(\text{CCM}_z) = \frac{2}{z(z-1)}\sum_{i=1}^{z-1}\sum_{j=i+1}^{z}\text{acc}_i^{\theta_j},$$

i.e., it is the average of the upper-triangular portion of the CCM (excluding the diagonal). Similarly to the previous cases, we commonly measure it at $t_z = t_N$.

D Computational Cost

A classic neuron model in a neural network requires u products to compute the output score, being u the size of the input, Eq. 1. We compare this cost in terms of the operations performed by MHs, still using products as a reference. In this section, comparing to the main paper, we consider also an additional part of the model that we did not exploit in the main paper, but that might help in some problems. In particular, given the MH input, ϕ, we denote with $\psi(\phi)$ a newly introduced simplified representation of ϕ. The function ψ is a fixed transformation that maps ϕ toward a customizable space in which it might be easier to compute similarities. It is desirable to select ψ so that it will not be particularly sensitive to small variations of ϕ, to promote stability in the matching process, but nothing prevents ψ to be the identity function. That is what we considered in the main paper. There is room for investigating the way ψ could be defined—beyond the scope of this paper. For example, if ϕ is an image/feature map, $\psi(\phi)$ could be a down-scaling operation. Hence, computing the output of a MH involves three main operations: (1) evaluating $\psi(\phi)$, whose cost is $C(\psi)$, that transforms the neuron input in a customizable manner, being $\tilde{u}$ the size of the ψ-output; (2) computing the attention scores by α, Eq. 4, with cost $m\tilde{u}$ plus the cost of the softmax^δ operation, that is δ; (3) blending memories, δu products, due to the sparsity of the attention scores; (4) computing the usual output function as in a classic neuron, u products. In total, we have $C(\psi) + m\tilde{u} + \delta + \delta u + u$. The cost of a layer of n classic neurons trivially becomes un, while the cost of a Memory Head that share the same K is

$$C(\psi) + m\tilde{u} + \delta + \delta un + un, \tag{6}$$

where only the last two terms depends on n, since the first three ones are about operations that are performed only once, being K shared. In order to reasonable relate the cost of classic and MH neurons, some basic considerations must be introduced. First of all, the cost $C(\psi)$ is expected to be way smaller than the cost of the whole layer. For example, when ψ is just limited to the L_2 normalization of ϕ. Moreover, depending on the considered problem, there could be room for designing ψ such that $\tilde{u}$ is smaller than u. Of course, this does not always hold. It is reasonable to assume the term δ in Eq. 6 to be way smaller than the other ones (being it a strong sparsity index, always $< m$), thus we discard it. As a result, we can compute the ratio R_C between the cost of a MH and the corresponding classic layer,

$$R_C \quad = \quad \frac{m\tilde{u} + (\delta+1)un}{un} = \frac{m\tilde{u}}{nu} + \delta + 1. \tag{7}$$

In terms of memory consumption, a MH with n neurons stores matrix K and n matrices of memory units (M), that is a total of $m\tilde{u} + mun$ floating point numbers, while in a classic layer only the weight matrix is stored (un floating point values).

A candidate way to compare MH-based net with models that replay data from memory buffers, is to use the exact same network architecture, using classic neurons in place of MH neurons. Then, we allow replay-based methods to sample $R_C - 1$ examples from the buffer at each time step. In fact, these buffer-based models make a prediction on a mini-batch of buffer data in addition to the currently streamed sample, according to the continual online learning setting experimented in this paper. Of course, when comparing with models that have more layers that the MH-net, it is harder to keep a perfect balance in term of computational cost, so we allowed competitors to have a cost that is slightly larger than the one of the MH-net, making the comparison more challenging. Moreover, we recall that the replay-based methods learn by exploiting the label-related information they store on the replay-buffer, while no-label-information is stored by the MH (this the comparison becomes unfair when using very large replay buffers).

E HYPER-PARAMETERS & CODE

We evaluated multiple combinations of values for the main hyper-parameters of MHs and competitors, that we summarize in the following, in addition to the already described parameter values of the main paper.

Class-Incremental Results. As described in the main text, we selected $m = 10$ memory units per task. We considered $\delta \in \{1, 2\}$, $\beta \in \{10^{-4}, 10^{-3}, 10^{-2}\}$, $\tau_\alpha \in \{0.7, 0.95\}$, $\gamma \in \{1, 5\}$. In the case of MH$_{ok}$, we considered $\tau_\mu \in \{5k, 15k\}$, $\tau_\eta \in \{5k, 15k\}$. For our model and competitors, we considered a learning rate for the head $\rho_{\theta_h} \in \{10^{-4}, 10^{-3}, 10^{-2}\}$ (the backbone learning rate is selected as $\rho_{\theta_b} = 10^{-2} \cdot \rho_{\theta_h}$), and trained with fixed-step-size gradient descent. The best performing learning rate was the same one reported by Zhang et al. (2023), $\rho_{\theta_h} = 10^{-2}$ both for our model and competitors. We selected the optimal values of the hyper-parameters from the grids by maximizing the average accuracy $\mathcal{A}_\mathcal{T}$ on CIFAR100 and test the winning configuration on all the other datasets.

Online Class-Domain Incremental. In the case of MHs-based nets, in MODES we selected $m = 8$ memory units with $\delta = 2$, while we tested $\beta \in \{10^{-4}, 10^{-3}, 10^{-2}, 1\}$, $\tau_\mu \in \{50, 200\}$, $\tau_\eta \in \{50, 200\}$, $\tau_\alpha \in \{0.85, 0.95\}$, $\gamma \in \{1, 5, 25\}$. In NS-IMAGENET we considered $m \in \{10, 25, 50, 100\}$, $\delta \in \{2, 5\}$, $\beta \in \{10^{-4}, 10^{-3}, 10^{-2}\}$, $\tau_\mu \in \{50, 500, 5000\}$, $\tau_\eta \in \{50, 500, 5000\}$, $\tau_\alpha \in \{0.7, 0.85, 0.95\}$, $\gamma \in \{1, 5, 25\}$. For competitors, we also evaluated architectures equipped with one hidden layer with size $h \in \{5, 25\}$ for 2D data, while in $h \in \{50, 100\}$ for NS-IMAGENET. In all the models, we considered a learning rate for the head $\rho_{\theta_h} \in \{10^{-4}, 10^{-3}, 10^{-2}, 1\}$, and trained with fixed-step-size gradient descent. We also evaluated the case of Adam, which yielded lower results on average. Indeed, adopting optimizers with memory such as Adam may be tricky: at every step, the model might select a different set of weights to be updated, making the statistics of the optimizer invalid. We leave the investigation about the effect of such optimizers for future work, restricting our analysis to memoryless optimizers, which do not suffer from this issue. We also considered a weight decay factor DECAY for the optimizers $\in \{10^{-4}, 10^{-3}, 0\}$. We trained GDumb for 10 epochs on the buffer data. Other minor internal parameters of the competitors were set to the values suggested in the respective papers. In all the experiments in the online domain-incremental setting (Section 4.2), we selected the optimal values of the hyper-parameters by maximizing the average accuracy on a held-out validation set after having processed $N/2$ intervals–coherently with recent work (Cai et al., 2021) and differently from approaches that identify the best models on test data (Lopez-Paz & Ranzato, 2017; De Lange & Tuytelaars, 2021a).

Regarding the η_μ and η_τ parameters, their choice/ranges mainly depends on the nature of the tackled task. For instance, in the Class-Incremental experiments, multiple epochs are performed over data belonging to the same task: this means that data is seen multiple times and we expect keys to become outdated or less used (i.e., weak keys) in longer periods w.r.t. the Online Class-Domain Incremental experiments, where data is experienced only once (and we expect that keys become weak more rapidly).

The results reported in the main paper are averaged over three runs with different seeds. For all the experiments in this work we used PyTorch, running on a Linux machine–NVIDIA GeForce RTX 3090 GPU (24 GB). The code for MHs can be found at https://github.com/sailab-code/memory_head.

E.1 OPTIMAL VALUE OF THE HYPER-PARAMETERS

We report in Tab. 4 the best selected hyperparameters for the proposed MH in all the considered datasets and settings described in the main paper. The best configuration (MH$_{ok}$) on CIFAR100 was exploited for all the other class-incremental datasets (Section 4.1). We carried out a standard grid search over the reported grids. Each training on NS-IMAGENET took approximately 30-40 minutes with the reported hardware specifications. We were able to run 4 training procedures on a single GPU, at the same time, which took 1 hour in total.

E.2 HYPERPARAMETER SENSITIVITY

We analyzed the model sensitivity to variations to hyperparameter values. We report in Table 5 and Table 6 the performance attained by the best selected configuration when varying the relevant parameters δ, m, and β, in the case of the NS-IMAGENET dataset (Section 4.2). As a general comment, the influence of the number of memories (m) and δ mostly depends on the task that is being tackled. Limiting updates to the closest memory unit ($m = 1$) helps in better isolating the computations. Indeed, in that case, if there is a sufficient number of memories, the information transfer across tasks is minimal. As expected, when the number of memories (m) is lower than the number of classes (NS-IMAGENET is composed by 100 classes), the memories are shared among different classes and activating more than one memory for the computations ($\delta > 1$) improves the performance, since information transfer is beneficial. Conversely, when the amount of memories is too big, the behavior is less predictable. As it is typical in ML algorithms, including the other CL approaches, the optimal trade-off depends on the considered problems and on the available resources.

Table 4: Optimal parameters. The best selected hyperparameters for the proposed MH, drawn from the grids described in the text, for the datasets. See the code for further details.

Parameters	CIFAR100	MODES	NS-IMAGENET
δ	1	2	5
β	10^{-3}	10^{-2}	10^{-4}
ρ	10^{-2}	10^{-1}	10^{-4}
γ	1	25	1
m	10	8	25
τ_α	0.95	0.95	0.7
τ_η	15k	50	5000
τ_μ	500	50	500
DECAY	0.	0.	10^{-3}

Table 5: Sensitivity to hyperparameters (δ and m). We compare the performances obtained by the best selected hyperparams configuration (the corresponding row is denoted with *main paper*) with other configurations obtained by varying only one hyperparameter at the time, keeping all the others fixed.

Method	NS-IMAGENET		
	$\mathcal{A}_T \uparrow$	FORGET $\downarrow$	FWD. T. $\uparrow$
$\delta = 1$	42.2±0.2	57.2±0.3	16.3±0.1
$\delta = 2$	50.3±0.4	42.9±0.5	20.6±0.2
$\delta = 5$ (main paper)	57.8±0.1	24.3 ± 0.1	24.2 ± 0.1
$\delta = 10$	59.3±0.5	19.1±0.4	23.8±0.3
$m = 10$	54.3±0.6	35.3±0.3	23.6±0.3
$m = 25$ (main paper)	57.8±0.1	24.3 ± 0.1	24.2 ± 0.1
$m = 50$	52.1±0.1	37.8±0.4	23.3±0.2
$m = 100$	43.7±1.8	22.1±0.4	18.1±0.5

In the results reported in Table 5 we increased/decreased δ and m around the best configuration we found in our grids (following the validation procedure described in the paper).

Interestingly, we found that increasing the range δ would have been beneficial (see Appendix E for the range of values we consider in our experiments), but, again, our goal is to keep the model selection procedure realistic (as a side note, we briefly tried to increment it more, with no further improvements).

In Table 6, we analyze the MH_{ok} sensitivity with respect to variations on the key step size β. We remark that this component behaves as a standard learning rate for the keys, guiding the development of the keys for each step. We investigated both lower and higher values resulting in a mild performance decrease. Please refer to the main paper for results obtained with the MH_g variant, which does not consider the online update, as well as the avoidance of the WTA scheme (Fig 6).

E.3 TYPES OF BACKBONES

Our proposal was inspired by the recent interest in adapting pretrained representations/backbones/foundation models in the continual learning scenario, as we mention when introducing our work. MHs take advantage of the fact that the pretrained backbones are characterized by stable representations that slowly change over time, and the learning mechanism characterizing MHs is thus capable of adapting the internal key/value routing mechanism to the change of the backbone representation. Of course, when faced with a highly varying representation (e.g., caused by data that changes at a fast time-scale or by representations that changes at a fast time-scale, i.e., when the backbone was not pretrained in advance) the proposed method could face some difficulties in identifying proper keys – i.e. , also the keys might have difficulties in stabilizing on "right" regions of the input space. However, the plasticity of the keys can be easily adapted by tuning the β parameter: choosing a greater β could foster the model ability to be plastic, while a lower value would result in a more stable solution. Regarding the nature of the pretrained backbone, we tested the model in some preliminary investigations with both weights pretrained in supervised tasks and (more briefly) in the case of self-supervised techniques (MoCo v3 by Chen et al. (2021)), leading to similar conclusion. MHs are designed without having in mind any specific backbones and/or pretraining schemes.

Table 6: Sensitivity to key step size β. We compare the performances obtained by the best selected hyperparams configuration (the corresponding row is denoted with *main paper*) with other configurations obtained by varying only β and keeping fixed all the other hyperparamters.

Method	NS-IMAGENET		
	$\mathcal{A}_T \uparrow$	FORGET $\downarrow$	FWD. T. $\uparrow$
$\beta = 10^{-3}$	51.1±0.4	33.4±0.1	21.3±0.2
$\beta = 10^{-4}$ (main paper)	57.8±0.1	24.3 ± 0.1	24.2 ± 0.1
$\beta = 10^{-5}$	52.0±0.2	32.9±0.3	22.7±0.4

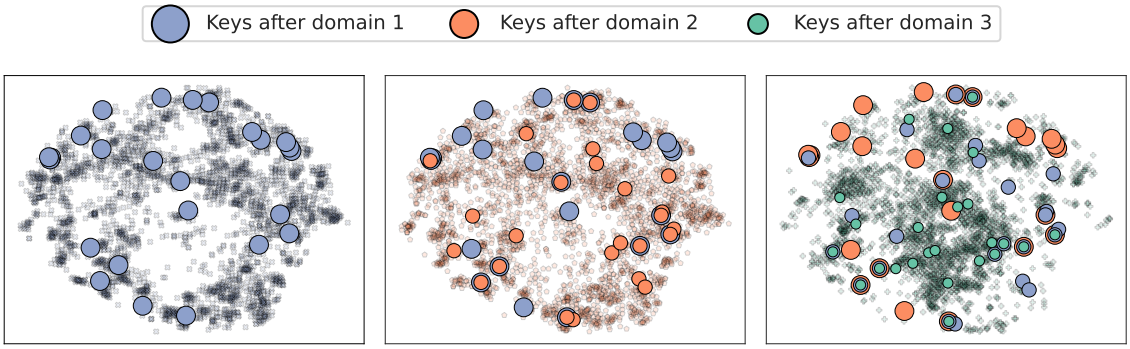

Figure 7: NS-IMAGENET. The data is organized in $q = 3$ domains, processed by the agent one after the other. The agent is based on $m = 25$ keys/memory units. When the data of each domain has been fully processed, such data and the locations of the keys are shown. The agent learns to cover (with the keys) the space regions in which the domain data is distributed. Left-to-right: the data and the keys after having processed each of the $q = 3$ domains. In each plot, we also report (as a reference) the locations of the keys at the end of the previous domain(s).

F DYNAMICS OF THE KEYS

We performed an additional investigation in the real-world NS-IMAGENET dataset (100 classes), by visualizing the temporal evolution of the backbone-provided features (in the three different data domains) and the keys developed by the MH, leveraging the t-SNE technique (Van der Maaten & Hinton, 2008), see Figure 7. The data is organized into $q = 3$ domains, as described in the main text (including noise as well). We report three t-SNE plots: each plot reports the representations of the samples belonging to each one of the $q = 3$ data domains, as well as the 25 keys (i.e., m=25) developed by the model after having finished processing data of each domain (for this reason, we have 3 plots). As a reference, in each plot we also report the keys that were developed at the end of the previous domain(s) (with different colors – see the legend), to highlight how the keys are evolving during the learning process. Some keys develop representations that are stable across domains, while others are plastic enough to model the changes in the data distribution over time.