Learning Representations on the Unit Sphere: Investigating Angular Gaussian and von Mises-Fisher Distributions for Online Continual Learning

Nicolas Michel, Giovanni Chierchia, Romain Negrel, Jean-François Bercher

Learning Representations on the Unit Sphere: Investigating Angular Gaussian and von Mises-Fisher Distributions for Online Continual Learning

Nicolas Michel*
Univ Gustave Eiffel, CNRS, LIGM
F-77454 Marne-la-Vallée, France
nicolas.michel@esiee.fr

Giovanni Chierchia
Univ Gustave Eiffel, CNRS, LIGM
F-77454 Marne-la-Vallée, France
giovanni.chierchia@esiee.fr

Romain Negrel
Univ Gustave Eiffel, CNRS, LIGM
F-77454 Marne-la-Vallée, France
romain.negrel@esiee.fr

Jean-François Bercher
Univ Gustave Eiffel, CNRS, LIGM
F-77454 Marne-la-Vallée, France
jf.bercher@esiee.fr

Abstract

We use the maximum a posteriori estimation principle for learning representations distributed on the unit sphere. We propose to use the angular Gaussian distribution, which corresponds to a Gaussian projected on the unit-sphere and derive the associated loss function. We also consider the von Mises-Fisher distribution, which is the conditional of a Gaussian in the unit-sphere. The learned representations are pushed toward fixed directions, which are the prior means of the Gaussians; allowing for a learning strategy that is resilient to data drift. This makes it suitable for online continual learning, which is the problem of training neural networks on a continuous data stream, where multiple classification tasks are presented sequentially so that data from past tasks are no longer accessible, and data from the current task can be seen only once. To address this challenging scenario, we propose a memory-based representation learning technique equipped with our new loss functions. Our approach does not require negative data or knowledge of task boundaries and performs well with smaller batch sizes while being computationally efficient. We demonstrate with extensive experiments that the proposed method outperforms the current state-of-the-art methods on both standard evaluation scenarios and realistic scenarios with blurry task boundaries. For reproducibility, we use the same training pipeline for every compared method and share the code at https://t.ly/SQTj.

1 Introduction

Deep neural networks can achieve very impressive performances when trained on independent and identically distributed data sampled from a fixed set of classes. In a real-world scenario, however, it may be desirable to train a model on a continuous data stream, where multiple classification tasks are presented sequentially so that the data from the old tasks are no longer accessible when learning new ones and data from the current task can be seen only once. This scenario is known as *online Continual*

*This work has received support from Agence Nationale de la Recherche (ANR) for the project APY, with reference ANR-20-CE38-0011-02. This work was granted access to the HPC resources of IDRIS under the allocation 2022-AD011012603 made by GENCI

Preprint. Under review.

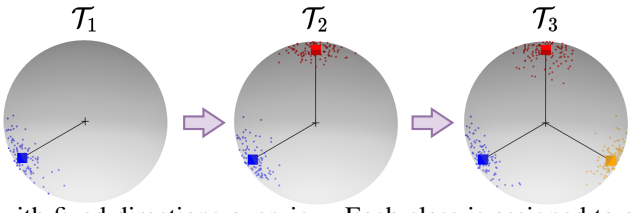

Figure 1: Training with fixed directions overview. Each class is assigned to a fixed vector of the standard basis. When changing task $\mathcal{T}$, new classes are encountered and mapped to remaining standard basis vectors. Best viewed in color.

Learning (CL) and poses a challenge for standard learning algorithms because the distribution of data changes over time (continual setting), and data are not accessible more than once (online setting). If such factors are not adequately taken into account, a trained model may suffer from Catastrophic Forgetting (CF), which is the loss of previously learned knowledge when learning new tasks or from new data. Online CL has seen growing interest in recent years [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], and several variations have been proposed [11, 8]. This paper focuses on class-incremental CL.

Among the different approaches for online CL [8], memory-based or replay-based methods have shown the best performances for the online setting [7, 12, 1, 8, 9]. In these approaches, a subset of past data is stored while training. When encountering a new batch from the stream, another batch is retrieved from memory and combined with the current batch for training. This mitigates forgetting by seeing past data along with current data. Recently, representation learning techniques combined with replay strategies have shown impressive performances for unsupervised CL [13, 14, 15] and supervised CL [9, 1]. However, contrastive learning-based methods [9] often require large batch sizes to benefit from negative samples [5], and distillation-based methods [1] require knowledge of task boundaries.

In this work, we propose an algorithm for memory-based representation learning based on a new loss function. Our loss function is devised from the principle of *maximum a posteriori* estimation under the hypothesis that the latent representations are distributed on the unit sphere. Such a hypothesis explicitly takes into account the fact that normalizing the latent vectors is a standard practice in contrastive learning. Specifically, we investigate the angular Gaussian distribution and the Mises-Fisher distribution, both designed for modeling antipodal symmetric directional data. The peculiarity of the resulting loss function is that the learned representations are pushed toward fixed directions, allowing for a learning strategy that is resilient to data drift and thus suitable for online continual learning. In summary, the contributions of this work are as follows.

- We devise a new loss function for representation learning based on the principle of maximum a posteriori estimation. We investigate both the angular Gaussian distribution and the von Mises-Fisher distribution for modeling representations that are restricted on the unit sphere.
- The key idea is to essentially assign pre-determined, mutually separated class means in the hidden space (in this case chosen to be the one-hot vectors) and training inputs from each class are coerced to increase overlap with its own class mean. The proposed loss function is resilient to data drift and does not require negative data or knowledge of task boundaries and performs well with smaller batch sizes while being computationally efficient.
- We show experimentally on benchmark datasets for online continual learning that the proposed approach outperforms state-of-the-art methods in most scenarios, and is robust to blurry task boundaries.

The paper is organized as follows. Section 2 describes related work and formally defines the problem addressed. Section 3 explains the mechanisms of our method. Section 4 presents our experimental results. Section 5 analyses the behaviour of the proposed approach, and section 6 concludes the paper.

2 Related work

Representation Learning In representation learning, it is common to work with latent vectors projected onto the sphere [16, 9, 17, 18, 19, 20]. Contrastive learning is a popular family of approaches

for representation learning and has been applied to online CL in previous work, producing state-of-the-art results [8, 1, 21, 4]. However, contrastive losses require large batch size to sample enough negatives. In this work, we introduce a loss adapted to CL which does not need negative samples.

Class incremental learning (CIL) One of the most popular continual learning scenario is CIL [11], which refers to learning from a sequence of tasks, where each task is composed of non-overlapping classes. Formally, consider $\{\mathcal{T}_1, \cdots, \mathcal{T}_K\}$ a learning sequence of K tasks, with $\{\mathcal{D}_1, \cdots, \mathcal{D}_K\}$ the corresponding dataset sequence with $\mathcal{D}_k = (X_k, Y_k)$ the data-label pairs. In CIL, it is assumed that $\forall k, j \in \{1, \cdots, K\}$ if $k \neq j$ then $Y_k \cap Y_{k_2} = \emptyset$ and the number of classes in each task is the same. In this study, we also refer to this setup as *clear* boundaries, meaning that task boundaries are clearly defined as no overlap between tasks exists.

Online Continual Learning In online CL, a new constraint is added by restricting the model to seeing the data only once. This problem has been demonstrated to be significantly harder than its offline counterpart and has been the main focus of various recent works [2, 7, 1, 5, 12, 22]. Notably, replay-based methods have shown the best performances.

Replay-based methods In recent years, several methods using fixed memory for replaying past data have addressed online CL. Experience Replay [7] introduces the use of a Reservoir sampling strategy [23] to replay past data while training on the current task. A-GEM [24] leverages memory data to constrain the current optimization step. DER++ [12] improves ER by adding knowledge distillation between tasks. SCR [9] also capitalizes on replaying past data but uses a supervised contrastive loss [25]. OCM [1] takes advantage of memory data in online CL with knowledge distillation and maximizes mutual information between previous and current representation with infoNCE [26]. Likewise, DVC [5] combines rehearsal strategies and information maximization. Other strategies using no-memory data usually perform poorly in an online context. In this work, we also focus on online CL and leverage memory data with reservoir sampling. However, we introduce a new loss based on Maximum a Posteriori estimation, defined in section 3.

Fixed Directions Recall that classification using cross entropy often ends with selecting components of the logit, which corresponds to a scalar product of the logit with basis vectors. In this sense, the usual practice uses a fixed classifier at the end of the network. This has been made more precise and generalized in previous works [27, 28]. However, our theoretical motivations lead to a more general framework from which multiple loss functions can be derived. Additionally, contrary to our approach, proposed losses are computed on unnormalized vectors (not projected on the hypersphere) while in this work, we take into consideration normalized representations and adapt the expression of the proposed loss to the hyperspherical topology.

Learning on the unit sphere Hyperspherical loss functions have been proposed in previous studies [29, 30] leveraging von Mises-Fisher distributions. In this work, we propose a more general framework and introduce new loss functions based on Saw distributions [31].

3 Proposed approach

In this section, we define the proposed approach by introducing new losses for online CL.

3.1 Representation learning with maximum a posteriori estimation

We are interested in estimating a function $\chi : \mathbb{R}^D \to [\![1, L]\!]$ that maps an input data to its corresponding class label, with $D \in \mathbb{N}$ the dimensionality of the input data, and $L \in \mathbb{N}$ the number of classes. Let us consider $\mathbf{x} \in \mathbb{R}^D$. For a given class $c \in [\![1, L]\!]$, we are interested in the posterior probability

$$\mathcal{P}(Y = c|X = \mathbf{x}) = \frac{\mathcal{P}(X = \mathbf{x}|Y = c)\mathcal{P}(Y = c)}{\mathcal{P}(X = \mathbf{x})}, \tag{1}$$

where X and Y are the random variables corresponding to the input and label. In terms of probability densities, we have the posterior density

$$p(Y = c|X = \mathbf{x}) = \frac{g_c(\mathbf{x})\pi_c}{\sum_{\ell=1}^{L} g_\ell(\mathbf{x})\pi_\ell} \tag{2}$$

with g_c the conditional p.d.f. of X given $Y = c$ and $\pi_c = \mathcal{P}(Y = c)$ the prior probability for class c.

Let us consider a latent variable $\mathbf{z} \in \mathbb{R}^d$ produced by an encoder $\Phi_\theta(.)$ parameterized by θ such that $\mathbf{z} = \Phi_\theta(\mathbf{x})$, with $d \in \mathbb{N}$ the dimension of the latent space. The posterior from Equation (2) can be written according to the random variable Z:

$$p(Y = c|Z = \mathbf{z}) = \frac{f_c(\mathbf{z})\pi_c}{\sum_{\ell=1}^{L} f_\ell(\mathbf{z})\pi_\ell} \tag{3}$$

where f_c is the conditional distribution of Z given $Y = c$. The objective is now to find the best mapping from X to Z to maximize the posterior distribution. Namely, we aim to find the parameters $\theta^\star$ such that $\theta^\star = \arg\max_\theta p(Y|Z)$. For a set of b independent observations $(\mathbf{z}_i, y_i)_{1\leq i\leq b}$, this amounts to maximizing $p(y_1 \cdots y_b|\mathbf{z}_1 \cdots \mathbf{z}_b) = \prod_{c=1}^{L} \prod_{i\in I_c} p(Y = c|\mathbf{z}_i)$ with $I_c = \{i \in [\![1, b]\!] \mid y_i = c\}$. The posterior distribution in Equation (3) can be thus expressed as

$$p(y_1 \cdots y_b|\mathbf{z}_1 \cdots \mathbf{z}_b) = \prod_{c=1}^{L} \prod_{i\in I_c} \frac{f_c(\mathbf{z}_i)\pi_c}{\sum_{\ell=1}^{L} f_\ell(\mathbf{z}_i)\pi_\ell}. \tag{4}$$

Eventually, we express the resulting loss in a batch-by-batch manner. For an incoming batch $\mathcal{B} = (\mathbf{x}_i, y_i)_{1\leq i\leq b}$ of size b we minimize Equation (5) with respect to parameters θ with $C_\mathcal{B}$ the classes in batch $\mathcal{B}$. We take $\pi_l = 0$ for classes that are not represented in the current batch.

$$\mathcal{L}_{MAP}(\mathcal{B}, \theta) = -\prod_{c\in C_\mathcal{B}} \prod_{i\in I_c} \frac{f_c(\Phi_\theta(\mathbf{x}_i))\pi_c}{\sum_{\ell\in\mathcal{C}_\mathcal{B}} f_\ell(\Phi_\theta(\mathbf{x}_i))\pi_\ell}. \tag{5}$$

3.2 Saw distributions on the unit sphere

The objective defined in Equation (5) requires us to express the conditional density functions explicitly. In representation learning, it is common to work with normalized vectors in the latent space [16, 9, 17, 18, 19, 20], making it natural to consider distributions on the unit sphere. In a seminal paper [31], Saw presented a large class of distributions on the sphere parameterized by a mean direction $\boldsymbol{\mu}_c$ (with $\|\boldsymbol{\mu}_c\| = 1$) and a concentration $\kappa \geq 0$. These distributions depend on a point $\mathbf{z}$ on the sphere (with $\|\mathbf{z}\| = 1$) only through the scalar product $t = \mathbf{z}^\top \boldsymbol{\mu}_c$, leading to the general form

$$f_c(\mathbf{z}) = a_\kappa \, g_\kappa(\mathbf{z}^\top \boldsymbol{\mu}_c). \tag{6}$$

Here above, a_κ is a normalization constant, the scalar product corresponds to the cosine similarity and $g_\kappa(t)$ is a non-negative increasing function that must verify a normalizing condition derived from the tangent-normal decomposition of the sphere [31].

A recent study [32] suggests that the representations learned by a neural network have a tendency to follow a Gaussian mixture model, which supports the assumption that the density on the sphere in Equation (6) shall be derived from such Gaussian mixture. In representation learning, we usually project the representation onto the unit sphere, which has a virtue in stabilizing training. Hence, we shall use the probability distribution of a projected Gaussian distribution onto the unit sphere.

This distribution, in the isotropic case, is the Angular Gaussian (AGD) distribution, which is a Saw distribution [31] (of dimensions d) defined by

$$g_\kappa^{AGD}(t) = e^{-\kappa^2} \sum_{n=0}^{\infty} \frac{(2\kappa t)^n \, \Gamma\left(\frac{d}{2} + \frac{n}{2}\right)}{n! \, \Gamma\left(\frac{d}{2}\right)}. \tag{7}$$

This expression of the AGD is given without proof in [31]. We prove it and give different expressions of the probability density for a general Gaussian vector $\mathcal{N}(\mu, \Sigma)$ projected onto the unit-sphere are proved in Appendix. One of these expressions reduces to (7) in the isotropic case $\Sigma = \sigma^2 I$, with $\sigma^2 = ||\boldsymbol{\mu}||^2/(2\kappa^2)$.

Alternatively, starting from a normal distribution with isotropic covariance $\kappa^{-1}\mathrm{I}$ and mean $\boldsymbol{\mu}_c$, we can condition on $\|\mathbf{z}\| = 1$ to obtain the Von Mises–Fisher (vMF) distribution, which is a Saw distribution with

$$g_\kappa^{\mathrm{vMF}}(t) = \exp(\kappa \, t). \tag{8}$$

3.3 Fixed directions for continual learning

Working with the Saw distributions in Section 3.2 requires knowledge of the mean directions of the different classes. With L equiprobable classes, it is natural to use a one-hot encoding of these classes; or, in other words, to assign them to the vertexes of the standard L-simplex. Alternatively, one could estimate these directions as a parameter of the network [29], or estimate them on the fly. The latter is cumbersome, however, as it requires large batches for the normalized mean estimation to be accurate, and the estimation is strongly biased at the start of each task when new classes are encountered.

Formally for a class c, we set $\boldsymbol{\mu}_c = \mathbf{e}_c = [0, 0, \dots, 1, 0, \dots 0]$, a vector where every component is 0 except the c-th component. With this strategy, the directions have the same distance of $\sqrt{2}$ from one another. Fixing directions also implies that they are independent of batch size or training step, which brings training stability. Such stability is crucial in CL, where new classes can easily conflict with older ones in the latent space [22]. Finally, fixed directions are obtained at no computational cost. We emphasize that in online CL, maintaining low computational overhead is also an important aspect, as new batches can come in fast succession while storage is limited. An overview of fixing mean direction onto the unit sphere is given in Figure 1.

3.4 Loss expression

Taking the logarithm of the objective defined in Equation (5) we obtained the general loss expression:

$$\mathcal{L}^{\log}_{\mathrm{MAP}}(\mathcal{B}, \theta) = -\sum_{c \in C_{\mathcal{B}}} \sum_{i \in I_c} \log \frac{g_\kappa\big(\mathbf{e}_c^\top \Phi_\theta(\mathbf{x}_i)\big)\pi_c}{\sum_{\ell \in C_{\mathcal{B}}} g_\kappa\big(\mathbf{e}_\ell^\top \Phi_\theta(\mathbf{x}_i)\big)\pi_\ell} \tag{9}$$

with $C_{\mathcal{B}}$ the classes in batch $\mathcal{B}$, θ the parameters of the model, $I_c = \{i \in [\![1, b]\!] \mid y_i = c\}$ the data indexes for class c, and $\mathbf{e}_c$ the c-th vector of the standard basis. Note that the above general expression includes the losses proposed in [10, 29] based on the von Mises-Fisher distribution.

The Angular Gaussian loss with Fixed Directions (AGD-FD) is obtained by plugging (7) into (9), and it is denoted by $\mathcal{L}_{\text{AGD-FD}}(\mathcal{B}, \theta)$. Likewise, the expression of the von-Mises-Fischer loss with Fixed Directions (vMF-FD) is obtained by plugging (8) into (9). In all our experiments, we work under the assumption that every class has the same prior.

3.5 Implementation details

Multi-view batch In online CL, the model has to overcome not only a changing data distribution but also the fact that data outside of memory are seen only once. To improve leveraging information from the incoming batch, each image is augmented several times to artificially increase the current batch size and show many 'views' of current data simultaneously. Specifically, for an incoming batch $\mathcal{B}$ and a random augmentation procedure $Aug(.)$, the model is trained on $\mathcal{B}_I = \mathcal{B} \bigcup_{i=1}^{n} Aug(\mathcal{B})$ with n the number of views. We show in section 5 leveraging a multi-view batch helps improve performances.

Guillotine regularization Similar to recent Representation Learning techniques, we apply Guillotine Regularization [33], to our model. Specifically, we express our model as $\Phi_\theta(.) = (\psi_{\theta_p} \circ \phi_{\theta_r})(.)$ with $\theta = \{\theta_p, \theta_r\}$. ψ_{θ_p} is referred to as the projection layer and ϕ_{θ_r} the representation layer. The projection layer usually is a simple multilayered perceptron, while the representation layer is a full neural network (e.g. a ResNet). During training, latent variables $\mathbf{z} = (\psi_{\theta_p} \circ \phi_{\theta_r})(\mathbf{x})$ are used for computing the loss from Equation (9). For inference, the projection layer is dropped, and latent variables $h = \phi_{\theta_r}(\mathbf{x})$ are used for the downstream task.

3.6 Training procedure

Since our proposed approach leads to learning representation, an extra step is needed in order to obtain our final classifier. For fair comparison, we consider that only images stored in memory are available at the end of training. Similar to SCR, when evaluating, the entire memory is used for training an intermediate classifier $\mathcal{C}_w$, with parameters w, on top of the frozen representations from ϕ_{θ_r}. During the evaluation step $\mathcal{C}_w \circ \phi_{\theta_r}(.)$ is used. A detailed procedure of our method is presented in algorithm 1.

Algorithm 1 Proposed Training Method

Input: Data stream $\mathcal{S}$; Memory $\mathcal{M}$; Augmentation procedure $Aug(.)$; Representation Learning Model $\Phi_\theta(.) = (\psi_{\theta_p} \circ \phi_{\theta_r})(.)$; Intermediate classifier $\mathcal{C}_w(.)$; Number of augmentations n;
Output: End-to-end classifier $\mathcal{C}_w \circ \phi_{\theta_r}(.)$; Memory $\mathcal{M}$;
`Training Phase:`
$\mathcal{M} \leftarrow \{\}$
for $\mathcal{B}_\mathcal{S} \in \mathcal{S}$ **do**
  $\mathcal{B}_\mathcal{M} \leftarrow Retrieve(\mathcal{M})$ ▷ Random retrieval
  $\mathcal{B}_C \leftarrow \mathcal{B}_\mathcal{S} \cup \mathcal{B}_\mathcal{M}$
  $(X_I, Y_I) \leftarrow \mathcal{B}_C \bigcup_{i=1}^{n} Aug(\mathcal{B}_C)$
  $\mathcal{B} \leftarrow (\Phi_\theta(X_I), Y_I)$
  $\theta \leftarrow Adam(\mathcal{L}^{log}_{\text{MAP}}(\mathcal{B}, \theta))$ ▷ Losses from Section 3.4
  $\mathcal{M} \leftarrow MemoryUpdate((X_\mathcal{S}, Y_\mathcal{S}), \mathcal{M})$ ▷ Reservoir Sampling
`Testing Phase:`
$(X_\mathcal{M}, Y_\mathcal{M}) \leftarrow \mathcal{M}$ ▷ Get all stored memory data
$H_\mathcal{M} \leftarrow \phi_{\theta_r}(X_\mathcal{M})$ ▷ Encode all memory data
$w \leftarrow Train(H_\mathcal{M}, Y_\mathcal{M}, \mathcal{C}_w)$ ▷ Train from frozen representations
return: θ_r; w; $\mathcal{M}$

4 Experiments

In this section, we describe our experimental setup and analyse obtained results of our method when compared to the current state-of-the-art.

4.1 Towards real-world scenarios

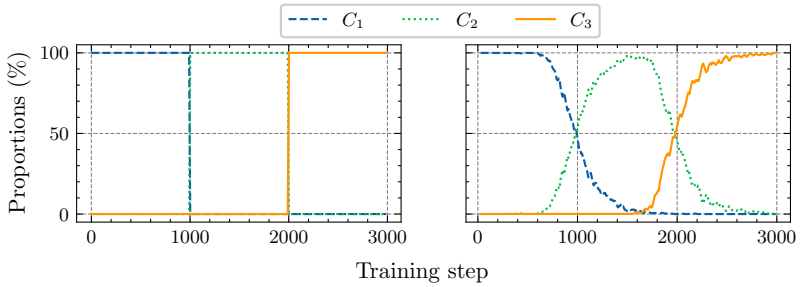

Figure 2: Visualisation of class proportions in the incoming batch during training. The left side shows data drift with *clear* boundaries while the right side shows data drift with *blurry* boundaries for $\sigma = 1500$ with 3 tasks, 10,000 images per task. C_i corresponds to the classes of task $\mathcal{T}_i$ with $i \in [1, 3]$.

Blurry task boundaries As introduced in section 2, CIL setups assume clear task boundaries. As an effect, several methods rely on knowing when the task change occurs to use techniques such as distillation [12, 1]. However, in a real-world scenario, there is no guarantee that task boundaries are clearly defined. Therefore simulating such an environment is crucial for testing models robustness. In that sense, we construct datasets with blurry task boundaries.

Algorithm 2 Blurry task boundaries shuffling

Input: Stream sequence with clear boundaries $\mathcal{S}_c$; Scale σ;
Output: Stream sequence with blurry boundaries $\mathcal{S}_b$
$\mathcal{S}_b \leftarrow \{\}$
while $|\mathcal{S}_c| \geq 0$ **do**
  $i \sim \mathcal{HN}(\sigma)$ ▷ Sample from a Half-Normal p.d.f.
  $\mathcal{S}_b \leftarrow \mathcal{S}_b \cup \mathcal{S}_c[i]$ ▷ Add i-th element of $\mathcal{S}_c$ to $\mathcal{S}_b$
  $\mathcal{S}_c \leftarrow \mathcal{S}_c \backslash \{\mathcal{S}_c[i]\}$ ▷ Drop i-th element of $\mathcal{S}_c$
return: $\mathcal{S}_b$

To create such a dataset we start from a dataset with clear boundaries and shuffle it using algorithm 2. We use a Half-Normal distribution with p.d.f $f_{HN}(y, \sigma) = \frac{\sqrt{2}}{\sigma\sqrt{\pi}} e^{-\frac{y^2}{2\sigma^2}}$ where $y \geq 0$ and σ is the scale parameter. The data shift occurring with the obtained dataset can be visualized in Figure 2.

Random label order In previous work, experiments are often concluded with the same label order for every run [1, 12]. However, studies show that label order is important in CL [34]. For fair comparison, we experimented for several runs and for each run, the order of the labels is randomly changed. This ensures more reproducibility and generalization of the proposed results.

4.2 Evaluation protocol

Subsequent is an exhaustive description of the datasets and baselines we considered, as well as details regarding the implementation decisions of our experiment.

Datasets To build continual learning environments, variations of standard image classification datasets [35, 36] are used. As introduced in section 4.1, we distinguish two variations of original datasets, one with clear task boundaries (*clear* variants) and the other with blurry task boundaries (*blurry* variants). For *clear* variants, each task is composed of non-overlapping classes while in *blurry* variants we introduce some overlap with the procedure previously described. More details are given in Appendix.

Baselines Several state-of-the-art approaches for online CL are used for comparison. **ER** [7] is a memory-based approach using a reservoir sampling [23] with a cross-entropy loss. **SCR** [9] is a memory-based approach trained using the SupCon loss [25] and a reservoir sampling. **GDumb** [6] is a method that stores data from stream in memory, ensuring a balanced class selection. The model is trained offline on memory data at inference time. **AGEM** [37] ensures that the average loss of past task does not increase by constraining the gradient using memory data. **DER++** [12] leverages knowledge distillation and reservoir sampling. **DVC** [5] maximizes information from different images views. **ER-ACE** [22] leverages an asymmetric cross-entropy loss along with reservoir sampling. **OCM** [1] maximizes mutual information with infoNCE [26] and uses reservoir sampling. **PFC** [27] which combined experience replay and Pre-Fixed Classifiers.

	CIFAR10		CIFAR100			Tiny ImageNet		
Method	M=500	M=1k	M=1k	M=2k	M=5k	M=2k	M=5k	M=10k
AGEM	16.88±1.42	16.86±1.24	4.3±0.7	4.28±0.66	4.24±0.66	0.71±0.12	0.73±0.09	0.74±0.13
DER++	47.01±5.76	53.18±5.8	21.74±1.72	28.42±2.45	34.92±1.52	6.65±1.12	13.58±1.47	14.82±5.21
DVC	55.8±4.67	61.42±2.68	19.79±2.63	23.19±3.6	27.43±3.26	2.45±1.27	1.72±0.74	2.22±1.39
ER-ACE	54.15±1.89	61.36±1.99	27.71±0.82	32.95±1.12	39.66±1.15	15.27±1.07	22.69±1.53	27.49±1.42
ER	52.51±6.27	59.02±3.27	23.04±0.9	29.65±1.33	35.52±1.43	12.49±0.5	20.55±0.96	24.06±1.01
GDUMB	34.06±1.81	41.42±1.25	11.43±0.69	15.74±0.61	25.53±0.44	7.07±0.38	13.79±0.5	21.72±0.4
PFC	57.21±0.84	62.90±0.92	24.1±0.90	31.1±1.60	38.6±0.90	11.73±0.73	19.15±2.2	23.51±2.15
SCR	60.63±1.19	68.17±0.97	30.31±0.64	36.64±0.62	40.6±0.76	19.44±0.34	23.21±0.76	24.43±0.7
OCM	**68.47±1.07**	**72.6±1.98**	29.09±1.41	36.67±1.01	42.49±1.45	19.38±0.61	27.52±0.8	32.3±1.34
vMF-FD	60.17±2.09	69.86±1.02	32.98±0.83	41.04±0.81	50.39±0.75	19.85±0.68	28.8±0.62	34.21±0.69
AGD-FD	61.29±1.54	70.06±1.11	**33.77±0.84**	**41.85±0.85**	**50.54±0.67**	**20.46±0.71**	**29.56±0.68**	**34.77±0.52**

Table 1: Final average accuracy (%) for all methods on datasets CIFAR10 split into 5 tasks, CIFAR100 split into 10 tasks, and TinyIN split into 100 tasks for varying memory sizes M. Tasks boundaries are *clear* in this setting. Results are computed over 10 runs, and the means and standard deviations are displayed. Best results are in bold. Second are underlined.

Metrics For evaluation, we use the average accuracy across all tasks at the end of training. This is also known as the final average accuracy [38, 11, 8].

Implementation details For memory-based models, except GDumb, we use random retrieval and reservoir sampling for memory management. DVC, SCR, and OCM employ a two-layer MLP with 512 neurons for intermediate layers (ReLU activation) and 128 neurons for the projection layer. Since our model requires more dimensions than classes, the projection layer output size remains fixed at 512, but additional dimensions can be added as new classes appear. For all methods, we use a

full untrained ResNet18. Stream batch size ($|X_S|$) is 10 and memory batch size ($|X_\mathcal{M}|$) is 64 for all methods. We use a Nearest Class Mean (NCM) classifier for intermediary classification. Other intermediate classifiers can be used but we observed little impact on the accuracy.

Hyperparameter search We performed a hyper-parameter search on CIFAR100 with a memory size $M = 5k$ and 10 tasks. This search includes data augmentation. Best parameters were kept and used for training on every dataset. For fair comparison, we applied this strategy to all approaches, including ours. For OCM we used the parameters from the original paper. Details regarding parameter selection can be found in Appendix.

Data augmentation For DER++, ER-ACE and GDumb we use random crop and random horizontal flip as augmentation. For every other method, we use the same data augmentation procedure composed of random crop, random horizontal flip, color jitter, and random grayscale. For our method, we use a number of views $n = 5$. OCM comes with additional data augmentation, which we did not change.

Adaptation to blurry boundaries For blurry boundaries, we adapted methods that required knowing task boundaries for this setup. Namely, we detected task changes with simple rules. (1) If new classes appear in the stream batch, a new task starts (2) Every task must be at least 100 batches long. This strategy helped to adapt OCM and DER++ but is limited as it can detect more tasks than desired.

4.3 Experimental results

In the following, we discuss the performances obtained by our method.

Clear boundaries The proposed approach has shown to outperform every considered baseline on CIFAR-100 and Tiny datasets with clear boundaries, as displayed in Table 1. This margin becomes even more significant for larger memory sizes on CIFAR-100 up to 8.05% with $M = 5k$, which exhibits better scaling with memory size than compared methods. Moreover, our method outperforms every considered baselines except OCM on CIFAR-10. However, experiments on *blurry* variants show evidence that OCM performances highly rely on task boundaries.

Blurry boundaries As shown in Table 2, our method outperforms every other method in this scenario. Notably, OCM, which requires precise task change for distillation, suffers from a consequent drop in performance in the blurry scenario while our approach gains performance instead. This demonstrates that our method is more suited to realistic scenarios than current state-of-the-art approaches.

	CIFAR10		CIFAR100			Tiny ImageNet		
Method	M=500	M=1k	M=1k	M=2k	M=5k	M=2k	M=5k	M=10k
AGEM	12.62±1.92	12.28±2.44	2.36±0.33	2.51±0.27	2.42±0.33	1.15±0.24	1.18±0.25	1.16±0.3
DER++	49.49±5.18	55.17±4.15	26.25±2.08	28.52±10.05	33.48±4.75	11.07±2.06	18.47±2.84	22.88±4.38
DVC	58.26±2.29	62.38±2.89	24.5±2.02	26.3±5.74	33.16±2.57	11.6±2.02	17.78±2.5	18.16±4.05
GDUMB	34.06±1.81	41.42±1.25	11.43±0.69	15.74±0.61	25.53±0.44	7.08±0.39	13.79±0.76	22.35±0.23
ER	54.55±2.04	61.7±2.41	23.68±0.95	29.84±1.83	36.27±1.52	11.33±2.03	19.14±1.46	24.51±1.63
ER-ACE	59.93±3.12	65.3±1.52	29.54±1.0	34.73±0.71	41.16±1.57	20.85±0.85	26.79±0.97	31.6±1.01
SCR	61.27±1.34	68.31±1.61	30.81±0.54	36.42±0.42	40.19±0.57	19.39±0.55	23.08±0.58	24.26±0.63
OCM	47.4±3.11	51.98±6.03	26.81±1.54	34.77±0.82	40.34±1.47	18.11±0.85	25.37±1.0	29.88±0.67
vMF-FD	63.87±1.72	71.04±1.29	34.68±0.76	42.0±0.68	50.71±0.60	21.71±0.54	30.21±0.51	35.16±0.49
AGD-FD	**64.38±2.0**	**71.59±0.99**	**36.32±0.68**	**43.51±0.35**	**50.84±0.72**	**23.57±0.4**	**31.75±0.42**	**35.96±0.47**

Table 2: Final average accuracy (%) for all methods on *blurry* variants with CIFAR10 split into 5 tasks, CIFAR100 split into 10 tasks and TinyIN split into 100 tasks for varying memory sizes M. Dataset boundaries are blurred with a scale $\sigma = 1500$. Results are computed over 10 runs and the means and standard deviations are displayed. Best results are in bold. Second are underlined.

5 Model Analysis

In subsequent we study the impact of several hyper-parameters for our method. We also apply some of our methods components such as multi-view batch and guillotine regularization to other methods for a fair comparison.

Impact of not fixing μ_c In section 3.3, we discussed the choice of $\boldsymbol{\mu}_c$, the mean for class c. In the following we compare the effect of fixing $\boldsymbol{\mu}_c$ as (i) $\boldsymbol{\mu}_c = \mathbf{e}_c$ where $\mathbf{e}_c$ is the c-th vector of the standard basis; to its estimation as (ii) $\boldsymbol{\mu}_c = \hat{\boldsymbol{\mu}}_c / ||\hat{\boldsymbol{\mu}}_c||$, the *spherical mean*, where $\hat{\mu}_c$ is the arithmetic mean of class c computed with current batch representations. Given that the mean estimation depends on the batch size, we also explore the effect of increasing the number of images retrieved from memory, denoted as $|X_{\mathcal{M}}|$, to facilitate more accurate estimation for larger batch sizes. Results in Table 3 demonstrate that fixing the mean values during training leads to a substantial improvement in the overall accuracy, even when using larger batch sizes.

$\mu_c \setminus \lvert X_{\mathcal{M}} \rvert$	16	32	64	128	256	512
$\hat{\mu}_c / \lvert\lvert\hat{\mu}_c\rvert\rvert$	16.42±0.25	19.23±0.5	25.54±1.48	35.82±0.65	44.54±0.82	45.79±0.79
e_c	**36.91±1.08**	**45.42±0.7**	**50.36±0.58**	**50.98±0.38**	**49.98±0.72**	**49.62±0.49**

Table 3: Comparison of the final AA on CIFAR100 with 10 tasks and M=5k for fixed mean $\boldsymbol{\mu}_c = \mathbf{e}_c$ and spherical mean estimates $\boldsymbol{\mu}_c = \hat{\boldsymbol{\mu}}_c / ||\hat{\boldsymbol{\mu}}_c||$, and different values of $|X_{\mathcal{M}}|$, the number of images retrieved from memory. Means and standard deviations over 5 runs are displayed.

Impact of concentration parameter κ Another hyper-parameter to choose for our method is the concentration parameter κ. On the one hand, the concentration must be low enough for the problem to be feasible but on the other hand, when the concentration falls below a certain value, Gaussians will overlap, which can lead to lower classification accuracy. To illustrate this effect, Figure 3 depicts the impact of different values of κ on the classification accuracy, where optimal values occur at $\kappa = 7$ for vMF and $\kappa = 0.1$ for AGD.

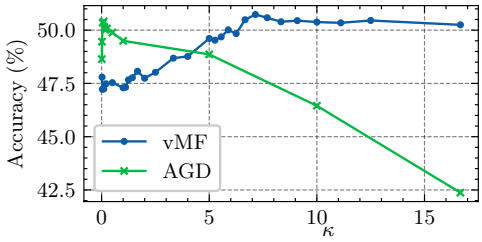

Figure 3: Final average accuracy (%) for $\kappa \in [0.02, 20]$ on CIFAR-100 with $M = 5k$.

n	3	4	5	6	7	8	9	10
SCR	43.9±0.6	44.4±0.4	44.9±1.0	45.5±0.4	45.3±0.6	45.5±0.8	46.5±0.6	46.3±0.5
ER	42.6±1.0	45.0±2.0	44.2±1.5	44.6±1.1	44.3±1.2	44.1±1.9	43.7±1.0	44.3±1.7
ER-ACE	42.2±0.9	42.1±1.4	42.8±0.7	42.7±0.8	41.8±1.7	41.1±1.8	40.9±1.3	-
AGD-FD	49.4±0.4	**50.0±0.4**	50.1±0.7	**51.1±0.4**	**51.1±0.3**	**50.9±0.6**	**51.7±0.3**	**51.2±0.4**
vMF-FD	**49.5±0.6**	50.0±0.7	**50.7±0.5**	50.9±0.7	50.9±0.4	50.4±0.7	50.9±0.3	50.8±0.3

Table 4: Accuracy on CIFAR100 with 10 tasks, $M = 5k$ for SCR, ER, ER-ACE, vMF-FD, AGD-FD and the number of views $n \in [3, 10]$. Means and standard deviations over 5 runs are displayed.

Impact of the number of views n Due to the online setting, increasing the number of views has a notable impact on the performances, as shown in Table 4. To ensure reasonable training time, we use $n = 5$ in our experiments, but increasing n could be considered for better performances. To discern the individual impacts of the multi-augmentation and the proposed training loss, we apply the former to SCR and record the resulting performances in Table 4. Notably, using a multi-view batch also boosts SCR performance, but our method still outperforms SCR even with identical numbers of augmentations. This observation reinforces the efficacy of the proposed loss function.

6 Conclusion

This paper proposes a new approach to deal with image classification, adapted to Continual Learning. This approach provides a framework that allows to outperform current state-of-the-art methods. Our learning strategy is to search for the neural network's representations that maximize the a posteriori probability density, a known and consistent approach.

The obtained performance likely results from the choice of data distribution. Since the data used are projected onto the hypersphere, a standard practice in representation learning, we consider distributions on the sphere: the von Mises Fisher distribution and the angular Gaussian distribution. The angular Gaussian distribution, which effectively corresponds to the projection of Gaussian data on the sphere, is the one that shows the best performance, probably because it is the model that

most closely approximates the nature of the data. A third ingredient that explains the performance is the use of fixed mean directions, corresponding to each of the classes. This both simplifies the implementation and makes the method robust to data-drift. It is particularly noticeable in the case of blurry boundaries between tasks, where our approach further widens the gap with other approaches. Finally, the results are based on careful implementations and a relevant choice of hyperparameters. Comparisons have been obtained both on standard evaluation scenarios and on more realistic datasets with blurry task boundaries. All methods were carefully re-implemented, often performing better than advertised in the original papers. Beyond pure performance, our approach is computationally efficient and does not require large batch sizes or negative data.

Here we used distributions with an i.i.d. assumption on the components. It is possible that the use of a non-diagonal covariance matrix would further improve the results. Moreover, other probability distributions, such as a multivariate Student-t on the sphere, can be considered. Finally, the application of our new losses to non-online batch learning should also be considered.

References

[1] Yiduo Guo, Bing Liu, and Dongyan Zhao, "Online Continual Learning through Mutual Information Maximization," in *Proceedings of the 39th International Conference on Machine Learning*, June 2022, pp. 8109–8126.

[2] 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*, 2019, vol. 32.

[3] Jiangpeng He and Fengqing Zhu, "Online continual learning via candidates voting," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 3154–3163.

[4] Nicolas Michel, Romain Negrel, Giovanni Chierchia, and Jean-François Bercher, "Contrastive Learning for Online Semi-Supervised General Continual Learning," July 2022, arXiv:2207.05615 [cs].

[5] Yanan Gu, Xu Yang, Kun Wei, and Cheng Deng, "Not Just Selection, but Exploration: Online Class-Incremental Continual Learning via Dual View Consistency," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 7432–7441.

[6] 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, Proceedings, Part II 16*, 2020, pp. 524–540.

[7] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne, "Experience Replay for Continual Learning," in *Advances in Neural Information Processing Systems*, 2019, vol. 32.

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

[9] Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner, "Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3589–3599.

[10] Huiwei Lin, Baoquan Zhang, Shanshan Feng, Xutao Li, and Yunming Ye, "PCR: Proxy-based Contrastive Replay for Online Class-Incremental Continual Learning," Apr. 2023, arXiv:2304.04408 [cs].

[11] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira, "Re-evaluating continual learning scenarios: A categorization and case for strong baselines," *arXiv preprint arXiv:1810.12488*, 2018.

[12] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara, "Dark experience for general continual learning: a strong, simple baseline," in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 15920–15930.

[13] Enrico Fini, Victor G. Turrisi Da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal, "Self-Supervised Models are Continual Learners," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA, June 2022, pp. 9611–9620, IEEE.

[14] Divyam Madaan, Jaehong Yoon, Yuanchun Li, Yunxin Liu, and Sung Ju Hwang, "Representational Continuity for Unsupervised Continual Learning," Apr. 2022, arXiv:2110.06976 [cs].

[15] MohammadReza Davari, Nader Asadi, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky, "Probing Representation Forgetting in Supervised and Unsupervised Continual Learning," *arXiv:2203.13381 [cs]*, Mar. 2022, arXiv: 2203.13381.

[16] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," *arXiv:2002.05709 [cs, stat]*, June 2020, arXiv: 2002.05709 version: 3.

[17] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko, "Bootstrap your own latent: A new approach to self-supervised Learning," *arXiv:2006.07733 [cs, stat]*, Sept. 2020, arXiv: 2006.07733.

[18] Xinlei Chen and Kaiming He, "Exploring Simple Siamese Representation Learning," *arXiv:2011.10566 [cs]*, Nov. 2020, arXiv: 2011.10566.

[19] Tongzhou Wang and Phillip Isola, "Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere," Aug. 2022, arXiv:2005.10242 [cs, stat].

[20] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny, "Barlow Twins: Self-Supervised Learning via Redundancy Reduction," June 2021, arXiv:2103.03230 [cs, q-bio].

[21] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin, "Co2l: Contrastive continual learning," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9516–9525, 2021.

[22] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky, "New insights on reducing abrupt representation change in online continual learning," 2022.

[23] Jeffrey S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software*, vol. 11, no. 1, pp. 37–57, Mar. 1985.

[24] David Lopez-Paz and Marc'Aurelio Ranzato, "Gradient Episodic Memory for Continual Learning," *arXiv:1706.08840 [cs]*, Nov. 2017, arXiv: 1706.08840.

[25] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan, "Supervised contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18661–18673, 2020.

[26] Aaron van den Oord, Yazhe Li, and Oriol Vinyals, "Representation Learning with Contrastive Predictive Coding," *arXiv:1807.03748 [cs, stat]*, Jan. 2019, arXiv: 1807.03748.

[27] Federico Pernici, Matteo Bruni, Claudio Baecchi, Francesco Turchini, and Alberto Del Bimbo, "Class-incremental learning with pre-allocated fixed classifiers," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 6259–6266.

[28] Piotr Bojanowski and Armand Joulin, "Unsupervised learning by predicting noise," in *International Conference on Machine Learning*. PMLR, 2017, pp. 517–526.

[29] Md Abul Hasnat, Julien Bohné, Jonathan Milgram, Stéphane Gentric, and Liming Chen, "von Mises-Fisher Mixture Model-based Deep learning: Application to Face Verification," Dec. 2017, arXiv:1706.04264 [cs].

[30] Pascal Mettes, Elise Van der Pol, and Cees Snoek, "Hyperspherical prototype networks," *Advances in neural information processing systems*, vol. 32, 2019.

[31] John G. Saw, "A family of distributions on the m-sphere and some hypothesis tests," *Biometrika*, vol. 65, no. 1, pp. 69–73, 1978.

[32] Yasuhiko Asao, Ryotaro Sakamoto, and Shiro Takagi, "Convergence of neural networks to gaussian mixture distribution," *arXiv:2204.12100*, 2022.

[33] Florian Bordes, Randall Balestriero, Quentin Garrido, Adrien Bardes, and Pascal Vincent, "Guillotine regularization: Improving deep networks generalization by removing their head," *arXiv:2206.13378*, 2022, version: 1.

[34] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang, "Scalable and Order-robust Continual Learning with Additive Parameter Decomposition," Feb. 2020, arXiv:1902.09432 [cs, stat].

[35] Alex Krizhevsky et al., "Learning multiple layers of features from tiny images," *University of Toronto*, 2009.

[36] Ya Le and Xuan Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, pp. 3, 2015.

[37] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny, "Efficient Lifelong Learning with A-GEM," *arXiv:1812.00420 [cs, stat]*, Jan. 2019, arXiv: 1812.00420.

[38] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al., "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[39] P.E. Jupp and K.V. Mardia, *Directional Statistics*, Wiley Series in Probability and Statistics. Wiley, 2009.

[40] Tarmo M. Pukkila and C. Radhakrishna Rao, "Pattern recognition based on scale invariant discriminant functions," *Information Sciences*, vol. 45, no. 3, pp. 379–389, 1988.

[41] P. J. Paine, S. P. Preston, M. Tsagris, and Andrew T. A. Wood, "An elliptically symmetric angular gaussian distribution," *Statistics and Computing*, vol. 28, no. 3, pp. 689–697, 05 2018.

[42] John G. Saw, "Jacobians of singular transformations with applications to statistical distribution theory," *Communications in Statistics*, vol. 1, no. 1, pp. 81–91, 1973.

[43] Daniel Zwillinger, Victor Moll, I.S. Gradshteyn, and I.M. Ryzhik, Eds., *Table of Integrals, Series, and Products (Eighth Edition)*, Academic Press, Boston, eighth edition edition, 2014.

[44] Alex Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[45] Vincenzo Lomonaco and Davide Maltoni, "Core50: a new dataset and benchmark for continuous object recognition," in *Conference on robot learning*. PMLR, 2017, pp. 17–26.

Appendices

A Projected-normal or Angular Gaussian Distribution

Let x be a random vector of $\mathbb{R}^d$ with a Gaussian distribution of mean μ and covariance matrix Σ:

$$f_X(x) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{10}$$

and define by

$$u = \frac{x}{||x||} = \frac{x}{(x^Tx)^{\frac{1}{2}}} = \frac{x}{r} \tag{11}$$

the projected vector onto the unit sphere $S_{d-1} = \{y \in \mathbb{R}^d : y^Ty = 1\}$. The marginal of x on S_{d-1} is called *projected-normal* in [39] or *angular Gaussian* in [40]. It seems to be little known and utilized [41], especially in the case $d > 3$.

We give here several expressions for the density $f_U(u)$ of the normalized vector, recalling the result of [40] in terms of a recursively computable integral, proving a result which has been stated in [31] without direct proof, and extending it to the general case. Finally, we provide a closed-form expression in terms of a special function. Let $r = (x^Tx)^{\frac{1}{2}}$. The Jacobian of the transformation

$x \to (r,u)$ is r^{d-1} [42], so that the density of (r,u) with respect to the surface element $\mathrm{d}\omega_{d-1}$ on the unit sphere, is given by

$$f_{R,U}(r,u) = \frac{r^{d-1}}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(ru-\mu)^T\Sigma^{-1}(ru-\mu)\right) \tag{12}$$

$$= \frac{r^{d-1}}{(2\pi)^{\frac{d}{2}}|\Sigma|} \exp\left(-\frac{1}{2}\mu^T\Sigma^{-1}\mu\right) \exp\left(-\frac{1}{2}r^2u^T\Sigma^{-1}u + ru^T\Sigma^{-1}\mu\right). \tag{13}$$

The density for $f_U(u)$ is obtained by marginalizing $f_{R,U}(r,u)$ over r: $f_U(u) = \int_0^\infty f_{R,U}(r,u)\mathrm{d}r$. Let $r' = r(u^T\Sigma^{-1}u)^{\frac{1}{2}}$; then

$$f_U(u) = \frac{(u^T\Sigma^{-1}u)^{-\frac{d}{2}}}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\mu^T\Sigma^{-1}\mu\right) \int_0^\infty r'^{d-1} \exp\left(-\frac{1}{2}r'^2 + r'\frac{u^T\Sigma^{-1}\mu}{u^T\Sigma^{-1}u}\right)\mathrm{d}r' \tag{14}$$

Denoting $\lambda = (\mu^T\Sigma^{-1}\mu)^{\frac{1}{2}}$, $\bar{u} = \frac{u}{(u^T\Sigma^{-1}u)^{\frac{1}{2}}}$ and $\bar{\mu} = \frac{\mu}{(\mu^T\Sigma^{-1}\mu)^{\frac{1}{2}}}$, (14) becomes

$$f_U(u) = \frac{(u^T\Sigma^{-1}u)^{-\frac{d}{2}}}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\lambda^2\right) \int_0^\infty r'^{d-1} \exp\left(-\frac{1}{2}r'^2 + \lambda r'\,\bar{u}^T\Sigma^{-1}\bar{\mu}\right)\mathrm{d}r' \tag{15}$$

Remark: With $\mu = 0$ and $\Sigma = \sigma^2 1$, which means that x is distributed as a centered isotropic Gaussian, (15) reduces to

$$f_U(u) = \frac{1}{(2\pi)^{\frac{d}{2}}} \int_0^\infty r'^{d-1} \exp\left(-\frac{1}{2}r'^2\right)\mathrm{d}r' = \frac{\Gamma\left(\frac{d}{2}\right)}{2\pi^{\frac{d}{2}}} = \frac{1}{\omega_{d-1}} \tag{16}$$

where we used $u^Tu = 1$ and the known property

$$\int_0^\infty r^{d-1} \exp\left(-\frac{1}{2}r^2\right)\mathrm{d}r = 2^{\frac{d}{2}-1}\Gamma\left(\frac{d}{2}\right). \tag{17}$$

Equation (16) shows that $f_U(u)$ is the uniform distribution on the unit-sphere, where ω_{d-1} is the surface of the unit-sphere.

Starting with (14), we can now state the first result, which is due to [40].

Proposition 1. *With $\lambda = (\mu^T\Sigma^{-1}\mu)^{\frac{1}{2}}$ and $\alpha = \frac{u^T\Sigma^{-1}\mu}{u^T\Sigma^{-1}u}$, the probability density of the normalized Gaussian vector is*

$$f_U(u) = \frac{(u^T\Sigma^{-1}u)^{-\frac{d}{2}}}{(2\pi)^{\frac{d}{2}-1}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\lambda^2 - \alpha^2\right) I_d(\alpha) \tag{18}$$

with

$$I_d(\alpha) = \frac{1}{\sqrt{2\pi}} \int_0^\infty r^{d-1} \exp\left(-\frac{1}{2}(r-\alpha)^2\right)\mathrm{d}r \tag{19}$$

and can be computed as

$$I_d(\alpha) = \alpha I_{d-1}(\alpha) + (d-2)I_{d-2}(\alpha),$$

with $I_1 = \Phi(\alpha)$ and $I_2 = \phi(\alpha) + \alpha\Phi(\alpha)$, where $\phi(.)$ and $\Phi(.)$ are respectively the standard normal probability density function and cumulative distribution function.

Proof. Completing the square in the argument of the exponential under the integral in (14) gives (18), with the definition of I_d in (19). Integration by part of I_d yields the recurrence equation. Finally, the initial values follow by direct calculation. □

The downside of (18) is of course that it depends on an integral form, even if this integral can be easily evaluated by recurrence. From (15), it is possible to obtain the density as a series. We give here this result in the general case and recover the result stated in [31] without direct proof.

Proposition 2. *With $\lambda = (\mu^T\Sigma^{-1}\mu)^{\frac{1}{2}}$, $\bar{u} = \frac{u}{(u^T\Sigma^{-1}u)^{\frac{1}{2}}}$ and $\bar{\mu} = \frac{\mu}{(\mu^T\Sigma^{-1}\mu)^{\frac{1}{2}}}$, the probability density of the normalized Gaussian vector is*

$$f_U(u) = \frac{\Gamma\left(\frac{d}{2}\right)}{2\pi^{\frac{d}{2}}} \frac{(u^T\Sigma^{-1}u)^{-\frac{d}{2}}}{|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}\lambda^2} \sum_{k=0}^\infty \left(\lambda\bar{u}^T\Sigma^{-1}\bar{\mu}\right)^k \frac{\Gamma\left(\frac{d+k}{2}\right)}{k!\,\Gamma\left(\frac{d}{2}\right)} \tag{20}$$

Proof. In the integral in (15), we can expand the exponential $\exp\left(\lambda r\ \bar{u}^T\Sigma^{-1}\bar{\mu}\right)$ in Taylor series, so that

$$\int_0^\infty r^{d-1}\exp\left(-\frac{1}{2}r^2+\lambda r\ \bar{u}^T\Sigma^{-1}\bar{\mu}\right)\mathrm{d}r \tag{21}$$

$$=\int_0^\infty r^{d-1}\exp\left(-\frac{1}{2}r^2\right)\sum_{k=0}^{\infty}\frac{1}{k!}\left(\lambda r\ \bar{u}^T\Sigma^{-1}\bar{\mu}\right)^k\mathrm{d}r \tag{22}$$

$$=\sum_{k=0}^{\infty}\frac{1}{k!}\left(\lambda\bar{u}^T\Sigma^{-1}\bar{\mu}\right)^k\int_0^\infty r^{d-1+k}\exp\left(-\frac{1}{2}r^2\right) \tag{23}$$

$$=2^{\frac{d}{2}-1}\sum_{k=0}^{\infty}\frac{1}{k!}\left(\lambda\bar{u}^T\Sigma^{-1}\bar{\mu}\right)^k\Gamma\left(\frac{d+k}{2}\right) \tag{24}$$

where the last line follows from the identity (17). Plugging this in (15) and simplifying yield (20). □

Note that the first term in (20) is the inverse of the unit-sphere's surface ω_{d-1}. In the isotropic case, that is $\Sigma=\sigma^2 1$, (20) reduces to

$$f_U(u)=\frac{\Gamma\left(\frac{d}{2}\right)}{2\pi^{\frac{d}{2}}}e^{-\frac{1}{2}\lambda^2}\sum_{k=0}^{\infty}\left(\lambda u^T\bar{\mu}\right)^k\frac{\Gamma\left(\frac{d+k}{2}\right)}{k!\,\Gamma\left(\frac{d}{2}\right)} \tag{25}$$

where we used the fact that $u^Tu=1$ and where $\bar{\mu}$ is now $\bar{\mu}=\frac{\mu}{(\mu^T\mu)^{\frac{1}{2}}}$. This is the formula given in [31], up to minor notations differences. Finally, for $\mu=0$, (25) reduces to the uniform distribution on the unit-sphere $f_U(u)=1/\omega_{d-1}$.

Finally, it is possible to obtain a closed form in terms of a special function.

Proposition 3. *With $\lambda=(\mu^T\Sigma^{-1}\mu)^{\frac{1}{2}}$ and $\gamma=\frac{u^T\Sigma^{-1}\mu}{(u^T\Sigma^{-1}u)^{\frac{1}{2}}}$, the probability density of the normalized Gaussian vector is*

$$f_U(u)=\frac{(u^T\Sigma^{-1}u)^{-\frac{d}{2}}}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}}e^{-\frac{1}{2}\lambda^2-\frac{1}{8}\gamma^2}\Gamma(d)D_{-d}\left(\sqrt{2}\gamma\right), \tag{26}$$

where D_{-d} is a Parabolic cylinder function.

Proof. A result in the celebrated Tables of integrals, Series and Products of Gradshteyn and Ryzhik states, [43, eq. 3.462], that

$$\int_0^\infty x^{\nu-1}e^{-\beta x^2-\gamma x}\mathrm{d}x=(2\beta)^{-\nu/2}\Gamma(\nu)e^{-\frac{\gamma^2}{8\beta}}D_{-\nu}\left(\frac{\gamma}{\sqrt{2\beta}}\right)\text{ for }\beta>0,\nu>0 \tag{27}$$

where D_ν is a parabolic cylinder function, [43, eq. 9.240]. We see that the integral in (15) has precisely this form, with $\nu=d$, $\beta=1/2$, and $\gamma=\lambda\bar{u}^T\Sigma^{-1}\bar{\mu}$. Plugging this in (15) and rearranging yield (26). □

B Hyperparameter Search

This appendix describes which hyper-parameter values have been tested for every compared method. As described in the main paper, the search has been conducted on one setup, namely CIFAR-100 with M=5k, and the resulting hyper-parameters have been used for all remaining scenarios. This strategy has been applied to every compared method for fair comparison.

B.1 Augmentation strategy

Some methods presented prove to gain from simple augmentations rather than more complex augmentations. To obtain the best performances possible for every compared method, we considered two augmentations strategies, which we named *partial* and *full*, respectively.

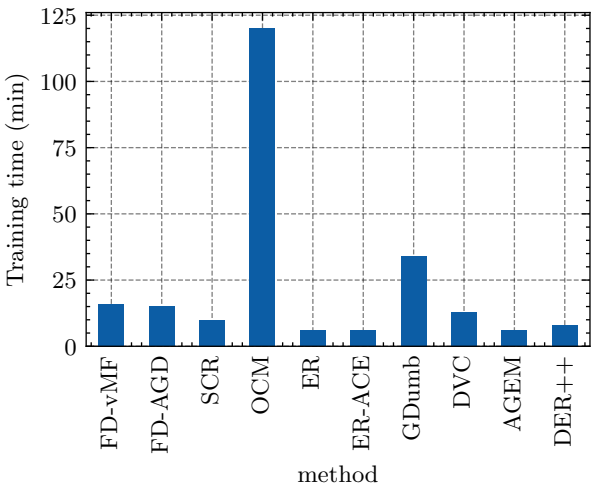

Figure 4: Time consumption in minutes, for every trained methods, on CIFAR100, M=5k, and 10 tasks.

***Partial* augmentation strategy.** The partial augmentation strategy is, as the name implies, composed of only a subpart of the augmentations used in the *full* strategy. Precisely, it is only a sequence of a random crop and a random horizontal flip, with $p = 0.5$.

***Full* augmentation strategy.** The full augmentation strategy is composed a more augmentations. Namely, it is a sequence of random crop, horizontal flip, color jitter and random gray scale. Color jitter parameters are set to $(0.4, 0.4, 0.4, 0.1)$ and $p = 0.8$. The probability of applying random gray scale is set to 0.2.

B.2 Hyper-parameters table

Table 5 is an exhaustive list of the hyper-parameters values tried during grid search. Note that for OCM we used the values given in their original work. Similarly, for GDumb, we experimented only with the augmentations and kept the other parameters as given in their original work.

C Hardware and computation

For compared methods we trained on 2 RTX A5000 GPUs. Figure 4 references the training time of each method on CIFAR100 M=5k. Our method can achieve best performance while having a low computational overhead. Notably, the time consumption difference between SCR and FD-AGD/FD-vMF is due to the number of augmentations used for training.

D Additional Experiments

In this section we gives more detail dataset used in the main paper.

D.1 Datasets

As described in the main paper, for the CIL case we experimented on CIFAR10, CIFAR100[44] and Tiny ImageNet [36] with blurry and clear task boundaries. In the Appendix, we included partial experiments on CORe50 [45].

CIFAR10 contains 50,000 32x32 train images as well as 10,000 test images and is split into 5 tasks containing 2 classes each for a total of 10 distinct classes.

CIFAR100 contains 50,000 32x32 train images as well as 10,000 test images and is split into 10 tasks containing 10 classes each for a total of 100 distinct classes.

Method	Parameter	Values
ER	optim	[SGD, Adam]
	weight decay	[0, 1e-4]
	lr	[0.0001, 0.001, 0.01, 0.1]
	momentum	[0, 0.9]
	aug. strat.	[full, partial]
ER-ACE	optim	[SGD, Adam]
	weight decay	[0, 1e-4]
	lr	[0.0001, 0.001, 0.01, 0.1]
	momentum	[0, 0.9]
	aug. strat.	[full, partial]
A-GEM	optim	[SGD, Adam]
	weight decay	[0, 1e-4]
	lr	[0.0001, 0.001, 0.01, 0.1]
	momentum	[0, 0.9]
	aug. strat.	[full, partial]
DER++	optim	[SGD, Adam]
	weight decay	[0, 1e-4]
	lr	[0.0001, 0.001, 0.01, 0.03]
	momentum	[0, 0.9]
	aug. strat.	[full, partial]
	alpha	[0.1, 0.2, 0.5, 1.0]
	beta	[0.5, 1.0]
DVC	optim	[SGD, Adam]
	weight decay	[0, 1e-4]
	lr	[0.0001, 0.001, 0.01, 0.1]
	momentum	[0, 0.9]
	aug. strat.	[full, partial]
SCR	optim	[SGD, Adam]
	weight decay	[0, 1e-4]
	lr	[0.0001, 0.001, 0.01, 0.1]
	momentum	[0, 0.9]
	aug. strat.	[full, partial]
GDumb	aug. strat.	[full, partial]
FD-AGD	optim	[SGD, Adam]
	weight decay	[0, 1e-4]
	lr	[0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1]
	momentum	[0, 0.9]
	aug. strat.	[full, partial]
	var	[0.05, 0.5, 1, 2, 3, 4, 5, 10]
FD-vMF	optim	[SGD, Adam]
	weight decay	[0, 1e-4]
	lr	[0.0001, 0.001, 0.01, 0.1]
	momentum	[0, 0.9]
	aug. strat.	[full]
PFC	optim	[SGD, Adam]
	weight decay	[0, 1e-4]
	lr	[0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1]
	momentum	[0, 0.9]
	aug. strat.	[full, partial]
	var	[0.05, 0.5, 1, 2, 3, 4, 5, 10]

Table 5: Hyper-parameters tested for every method on CIFAR100, M=5k, 10 tasks.

Tiny ImageNet is a subset of the ILSVRC-2012 classification dataset and contains 100,000 64x64 train images as well as 10,000 test images and is split into 100 tasks containing 2 classes each for a total of 200 distinct classes.

CORe50 is a domain incremental dataset designed for continuous object recognition. It is composed of $164,866$ 128×128 RGB images. We experimented in the New Instances setting and used sessions 3,7 and 10 for testing.

D.2 Experiments in Domain Incremental Learning (DIL) scenario

To further experiments on various Continual Learning scenarios, we included experiments on CORe50 [45] for *ER*, *SCR* and *AGD-FD* and various memory sizes.

Domain Incremental Learning (DIL) is another popular Continual Learning scenario where the distribution of the input data shift while keeping the same available classes between tasks. For example, the lighting conditions can changes from one task to the other. One popular dataset for evaluation in DIL is CORe50 [45].

Results We report the final average accuracy for considered methods on Table 6. It can be observed that our approach is on par with state-of-the art for every memory size. Additionally, the standard deviation of AGD-FD (ours) decreases for larger memory sizes, demonstrating superior stability for large memory size compared with other considered methods.

Method	M=1k	M=2k	M=5k
ER	32.0±2.6	36.6±3.0	39.3±2.2
SCR	42.1±2.0	**46.5±3.2**	48.6±2.3
AGD-FD	**42.6±2.3**	46.0±1.6	**50.0±0.9**

Table 6: Final average accuracy on CORe50, new instances setting, for varying memory sizes. Mean and standard deviation over 5 runs are reported.