Supplementary Material for Bring Evanescent Representations to Life in Lifelong Class Incremental Learning

Marco Toldo [1,2*] Mete Ozay[1]
[1]Samsung Research UK [2]University of Padova
marco.toldo@dei.unipd.it m.ozay@samsung.com

In this document, we present supporting material for the paper "Bring Evanescent Representations to Life in Lifelong Class Incremental Learning". In Section 1, we explicate the probabilistic interpretation underlying our approach. Then, in section 2 we introduce implementation details of the models and optimisation methods. Finally, we report some additional results and ablation studies in Section 3.

1. Theoretical Motivation

We explore dynamics of incrementally learned classifiers using a probabilistic approach. To elucidate the dynamics of models used for CIL, we factorise the probability $p(C \in \mathcal{C}|F \in \mathcal{F})$ as follows

$$p(C \in \mathcal{C}|F \in \mathcal{F}) \propto \frac{\mathbf{P}_A - \mathbf{P}_B}{\mathbf{P}_C}, \tag{1}$$

where:

- $F \in \mathcal{F} = \mathcal{F}_t \cup \mathcal{F}_{old}$ is the random variable taking values from the set of feature representations $\mathcal{F}_t$ learned at step t on the available dataset $\mathcal{D}_t$ and from $\mathcal{F}_{old} = \{\mathcal{F}_{t'}\}_{t'=0}^{t-1}$
- $C \in \mathcal{C} = \mathcal{C}_t \cup \mathcal{C}_{old}$ is a random variable of semantic (class) representations, where $\mathcal{C}_{old} = \{\mathcal{C}_{t'}\}_{t'=0}^{t-1}$

In Table 1 we explicate each individual term within the factorisation in 1, *i.e.*, $\mathbf{P}_C$, $\mathbf{P}_A := \mathbf{P}_1 + \mathbf{P}_2 + \mathbf{P}_3 + \mathbf{P}_4$, and $\mathbf{P}_B := \mathbf{P}_{12} + \mathbf{P}_{13} + \mathbf{P}_{14} + \mathbf{P}_{24} + \mathbf{P}_{34}$, and provide an interpretation for each of their constituents.

2. Implementation Details

2.1. Design of Representation Drift Models

2.1.1 Modeling Feature Drift

GM: We use a simple and lightweight multilayer perceptron to implement $\Gamma_{\gamma_t^n} : \mathcal{F}_t^0 \rightarrow \mathcal{F}_t^n$, composed of two fully-connected (FC) layers and a ReLU activation between them (Table 2). Input and output variables of $\Gamma_{\gamma_t^n}$ correspond to sets of B feature vectors of dimension D (number of feature channels at the output of the feature extractor, which is set to 512 in all experiments), both arranged in a $B \times D$ matrix. In addition, the number of output channels of the first FC layer is set to $2 * D$ and B is set equal to the cardinality of $\mathcal{D}_t$ (*i.e.*, the available training set at step t).
VM: We use a lightweight conditional variational autoencoder [15] to implement $\Gamma_{\gamma_t^n}$. The task of the cVAE is to learn a generative function of feature representations of training samples at stage n ($\mathcal{F}_t^n$), conditioned on the representations of the same samples at the beginning of the current incremental step ($\mathcal{F}_t^0$). The encoder and decoders are composed of two FC layers each. We refer to Table 3 for a more detailed description of the employed architectures. We perform conditioning in input and latent spaces by concatenation along the channel dimension. Input, output and conditioning variables of $\Gamma_{\gamma_t^n}$ correspond to sets of B feature vectors of dimension D, all arranged in $B \times D$ matrices. We set B equal to the cardinality of $\mathcal{D}_t$.

2.1.2 Modeling Semantic Drift

GM: We use a simple and lightweight multilayer perceptron to implement $\Psi_{\psi_t^n} : \mathcal{F}_0^n \rightarrow \Pi_{old}^{t,0}$, composed of two FC layers and a ReLU activation between them (Table 2). Input and output variables correspond, respectively, to sets of B and C_{old}^t feature vectors of dimension D, arranged in $D \times B$ and $D \times C_{old}^t$ matrices, where C_{old}^t denotes the number of past classes present at the current incremental step $t > 0$ and D the number of feature channels at the output of the feature extractor. The number of output channels of the first FC layer is set to $2 * B$ and B is set equal to the cardinality of $\mathcal{D}_t$.
VM: We use a lightweight conditional variational autoencoder to implement $\Psi_{\psi_t^n}$. The task of the cVAE is to learn a generative function of feature representations of old classes $\mathcal{F}_{old}^0$ (whose distribution is approximated as $p(\mathcal{F}_{old}^0; \Pi_{old}^{t,0}) \sim \mathcal{N}(\pi_c, \sigma_c)$), conditioned on those of new classes $\mathcal{F}_t^0$ (which can be extracted from the available train-

*Researched during internship at Samsung Research UK

Table 1: Factorisation of $p(\mathcal{C}|\mathcal{F})$ in CIL, where $\cdot$ is used to indicate multiplication of probability functions.

Term	Distribution	Interpretation	Model
$\mathbf{P}_C$	$p(\mathcal{F}_{old}, \mathcal{F}_t)$	Distribution of features shared among old and new classes.	$p(\mathcal{F}_{old}\|\mathcal{F}_t) \cdot p(\mathcal{F}_t)$
$\mathbf{P}_1$	$p(\mathcal{C}_t, \mathcal{F}_t)$	Distribution of features for new classes.	$p(\mathcal{C}_t\|\mathcal{F}_t) \cdot p(\mathcal{F}_t)$
$\mathbf{P}_2$	$p(\mathcal{C}_t, \mathcal{F}_{old})$	Distribution of features of old classes for new classes.	$p(\mathcal{C}_t\|\mathcal{F}_{old}) \cdot (\mathcal{F}_{old})$
$\mathbf{P}_3$	$p(\mathcal{C}_{old}, \mathcal{F}_t)$	Distribution of features of new classes for old classes.	$p(\mathcal{C}_{old}\|\mathcal{F}_t) \cdot p(\mathcal{F}_t)$
$\mathbf{P}_4$	$p(\mathcal{C}_{old}, \mathcal{F}_{old})$	Distribution of features for old classes.	$p(\mathcal{C}_{old}\|\mathcal{F}_{old}) \cdot p(\mathcal{F}_{old})$
$\mathbf{P}_{12}$	$p(\mathcal{C}_t, \mathcal{F}_t, \mathcal{F}_{old})$	Distribution of shared features for new classes.	$p(\mathcal{C}_t\|\mathcal{F}_t, \mathcal{F}_{old}) \cdot p(\mathcal{F}_t\|\mathcal{F}_{old}) \cdot p(\mathcal{F}_{old})$
$\mathbf{P}_{13}$	$p(\mathcal{C}_t, \mathcal{C}_{old}, \mathcal{F}_t)$	Distribution of features of new classes for new and old classes.	$p(\mathcal{C}_t\|\mathcal{C}_{old}, \mathcal{F}_t) \cdot p(\mathcal{C}_{old}\|\mathcal{F}_t) \cdot p(\mathcal{F}_t)$
$\mathbf{P}_{14}$	$p(\mathcal{C}_{old}, \mathcal{F}_t, \mathcal{F}_{old})$	Distribution of shared features for old classes.	$p(\mathcal{C}_{old}\|\mathcal{F}_t, \mathcal{F}_{old}) \cdot p(\mathcal{F}_t\|\mathcal{F}_{old}) \cdot p(\mathcal{F}_{old})$
$\mathbf{P}_{24}$	$p(\mathcal{C}_t, \mathcal{C}_{old}, \mathcal{F}_{old})$	Distribution of features of old classes for new and old classes.	$p(\mathcal{C}_t\|\mathcal{C}_{old}, \mathcal{F}_{old}) \cdot p(\mathcal{C}_{old}\|\mathcal{F}_{old}) \cdot p(\mathcal{F}_{old})$
$\mathbf{P}_{34}$	$p(\mathcal{C}_{old}, \mathcal{F}_{old}, \mathcal{F}_t)$	Distribution of shared features for old classes.	$p(\mathcal{C}_{old}\|\mathcal{F}_t, \mathcal{F}_{old}) \cdot p(\mathcal{F}_t\|\mathcal{F}_{old}) \cdot p(\mathcal{F}_{old})$

Table 2: Architecture of GM identified by MLP.

Feature Drift			**Semantic Drift**		
Input	Operator	Output	Input	Operator	Output
$B \times D$	FC layer	$B \times 2D$	$D \times B$	FC layer	$D \times 2B$
$B \times 2D$	FC layer	$B \times D$	$D \times 2B$	FC layer	$D \times C$

We set $D = 512$ and $B = |\mathcal{D}_t|$

Table 3: Architecture of VM identified by conditional VAE.

	Feature Drift			**Semantic Drift**		
	Input	Operator	Output	Input	Operator	Output
Encoder	$B \times 2D$	FC layer	$B \times 4D$	$D \times (C+B)$	FC layer	$D \times 2(C+B)$
	$B \times 4D$	FC layer	$B \times 2D$	$D \times 2(C+B)$	FC layer	$D \times 2$
Decoder	$B \times 2D$	FC layer	$B \times 4D$	$D \times (1+B)$	FC layer	$D \times 2(1+B)$
	$B \times 4D$	FC layer	$B \times 2D$	$D \times 2(1+B)$	FC layer	$D \times C$

We set $D = 512$, $B = |\mathcal{D}_t|$ and $C = |\mathcal{C}^t_{old}|$

ing set $\mathcal{D}_t$). Each encoder and decoder is composed of two FC layers. Once more, we refer to Table 3 for further details about the employed architectures. We perform conditioning in input and latent spaces by concatenation along the channel dimension. Input and output variables of $\Gamma_{\gamma^n_t}$ correspond to sets C^t_{old} of feature vectors of dimension D, arranged in $D \times C^t_{old}$ matrices. The conditioning variable is a set of B feature vectors of dimension D, arranged in $D \times B$ matrices. We set B equal to the cardinality of $\mathcal{D}_t$.

2.2. Training Details

2.2.1 Modeling Feature Drift

GM: We learn $\Gamma_{\gamma^n_t}$ by minimising the mean squared error between $\Gamma_{\gamma^n_t}(\mathcal{F}^0_t)$ and $\mathcal{F}^n_t$, *i.e.*, $L^t_f = ||\Gamma_{\gamma^n_t}(\mathcal{F}^0_t) - \mathcal{F}^n_t||^2_2$.
VM: We optimise the variational model (conditional VAE) following the objective proposed in [20]. The learning objective is composed of a reconstruction constraint and a regularisation term measuring the KL divergence between the posterior distribution modeled by the encoder and the standard normal prior, plus an additional term to maximise the mutual information between input and latent variables. Thus, the objective is of the form $L^t_f = \beta L^t_{rec,f} + (1-\alpha)L^t_{kl,f} + (\alpha - \lambda_{info} - 1)L^t_{info,f}$[1], where $L^t_{rec,f}$ is the reconstruction loss, $L^t_{kl,f}$ is the KL divergence loss and $L^t_{info,f}$ is the loss of the InfoVAE. We set $\beta = 1\text{e}1$ in all experiments, $\alpha = -1\text{e}1$ and $\lambda_{info} = 1\text{e}1$ on CIFAR100, and $\alpha = -1\text{e}2$ and $\lambda_{info} = 1\text{e}2$ on TinyImageNet and CUB200.

2.2.2 Modeling Semantic Drift

GM: We learn $\Psi_{\psi^n_t}$ by minimising the mean squared error between $\Psi_{\psi^n_t}(\mathcal{F}^0_t)$ and $\Pi^{t,0}_{old}$, *i.e.*, $L^t_s = ||\Psi_{\psi^n_t}(\mathcal{F}^0_t) - \Pi^{t,0}_{old}||^2_2$.
VM: We optimise the variational model (conditional VAE) by maximizing the ELBO (Evidence Lower BOund) [7]. The learning objective is thus composed of a reconstruction constraint and a regularisation term measuring the KL divergence between the posterior distribution modeled by the encoder and the standard normal prior, *i.e.*, $L^t_s = L^t_{rec,s} + \lambda_{kld,s}L^t_{kd,s}$. We set $\lambda_{kd,s} = 1$ in all experiments.

2.2.3 Model Fusion

We experimentally finetuned the values of λ_{fus} and λ_{corr} for each drift model configuration, dataset and incremental setup. In particular, we perform gridsearch such that $\lambda_{fus}, \lambda_{corr} \in \{1\text{e}2, 1\text{e}1, 1\text{e}0, 1\text{e}-1, 1\text{e}-2, 1\text{e}-3, 1\text{e}-4, 1\text{e}-5\}$ and select the best value combination.

In all the aforementioned setups, we employ the Adam optimiser [6] with fixed learning rate η, and train until convergence by performing early-stopping, that is, the model is

[1] We base our code on the implementation of InfoVAE provided on https://github.com/AntixK/PyTorch-VAE

trained until the loss function does not change for a predefined constant number of steps $\tau = 25$. We experimentally finetuned the value of learning rate $\eta \in \{1e-3, 1e-4, 1e-5\}$ for each drift model configuration, dataset and incremental setup.

Finally, we apply weight normalisation [14] to $\Psi_{\psi_t^n}$ (both for GM and VM implementations) and spectral normalisation [11] to weights of $\Gamma_{\gamma_t^n}$ (both for GM and VM implementations), since we observed an improvement in robustness of training convergence.

2.3. Competitors

We compare our approach with several CIL methods storing exemplars of old classes (EEIL [1], iCarl [13], UCIR [5]) and other SotA exemplar-free methods (EWC [8], LwF [10], LwM [4], PASS [21], SDC [19]). All methods are evaluated with the ResNet18 image classification model and batch size of 64 [21]. We perform gridsearch over key hyperparemeters of [4, 1, 13, 5, 8, 10] and report the best results[2]. As for exemplar-based methods [4, 1, 13, 5], we store 20 samples with *herd selection* [5, 13]. In addition, we use the original code of PASS [21]. Finally, we evaluate the SDC [19] method by employing the prototype drift compensation proposed in [19] to update prototypes of past classes. In particular, we employ the original code of [19] to evaluate and compensate for the feature drift of old-class prototypes (*i.e.*, in place of the proposed semantic and feature representation drift models), and we use the estimated up-to-date representations $\Pi_{old}^{t,n}$ by SDC [19] to approximate feature distribution of old classes with a parametrized Gaussian model, *i.e.*, $p(F \in \mathcal{F}_{old}^n; \pi_c \in \Pi_{old}^{t,n}) \sim \mathcal{N}(\pi_c, \sigma_c)$. Computation of σ_c was explained in the main text.

3. Experimental Evaluation

3.1. Additional Comparison with State-of-the-Art

Evaluation Metrics: We evaluate the performance of different methods using the standard top-1 accuracy. Firstly, we resort to a per-step metric [21] (Table 4 and Fig 1), defined as the average top-1 classification accuracy over all classes observed up to the current incremental step k

$$\bar{a}^k = \frac{1}{|\mathcal{C}_{0:k}|} \sum_{c \in \mathcal{C}_{0:k}} a_c^k, \quad \mathcal{C}_{0:k} = \bigcup_{t=0}^{k} \mathcal{C}_t, \tag{2}$$

where a_c^k denotes the accuracy for class c attained at step k. We additionally make use of the step-average incremental accuracy measure proposed in [13] (Table 4), defined as

$$\hat{a}^k = \frac{1}{k+1} \sum_{k'=0}^{k} \bar{a}^{k'} \tag{3}$$

where we take into account the evolution of the per-step accuracy, as computed in Eq. 2, up to the current step k.

Finally, we report a measure of forgetting [2] computed for each past class c at step $t > 0$ by

$$f_t^c = \max_{k<t}(a_k^c - a_t^c), \ \forall c \in \bigcup_{k<t} \mathcal{C}_k \tag{4}$$

where a_k^c denotes the top-1 accuracy for class c attained at the step $k < t$. We then compute the class-wise average of forgetting measures over all past classes at each step (Fig. 2), as well as the task-wise average at the end of incremental training (Fig. 3) (*i.e.*, we compute averages of forgetting measures at the final step over classes belonging to the same task).

Datasets. We evaluate our approach on multiple standard CIL benchmarks, that is, CIFAR100 [9], TinyImageNet [12] and CUB200-2011 [17] datasets. In addition, we provide some results on the large-scale ImageNet-Subset benchmark [3]. We devise 3 class-incremental protocols; first, the training is performed on half of the available semantic classes (except for one setup on CIFAR100, where only 40 classes are selected as the first task); then, the remaining class set is evenly divided into respectively 5, 10 or 20 incremental steps, each with an equal number of new classes being introduced. Class order is selected randomly and then fixed at every class split.

Comparisons: We first compare the proposed approach using the per-step top-1 accuracy (Eq. 2) with the vanilla and state-of-the-art (SotA) CIL methods (Fig. 1). We observe how the proposed approach provides improved classification accuracy w.r.t. some exemplar-based methods [4, 1, 13, 5] and some exemplar-free approaches [8, 10, 19, 21], attaining SotA performance on CIL without stored exemplars.

To validate robustness of our approach w.r.t. evaluation metric, in Table 4 we report additional results in the form of the step-average incremental accuracy proposed in [13] (Eq. 3). We notice that we still outperform most of the competitors. This holds also for the large-scale ImageNet-Subset benchmark, where our approach attains better performance than SotA exemplar-free methods and some exemplar-based frameworks, demonstrating resilience to change of data settings. In Table 4 we further report evaluation results on Cifar100 when a modified 32-layers ResNet [13] is employed as classification network. Once more, we observe that we surpass PASS [21] and SDC [19] exemplar-free SotA competitors by a large margin, especially when 10 incremental steps are performed. This indicates robustness to change of classification backbone.

Furthermore, we evaluate the CIL methods considered by computing a measure of forgetting (Eq. 4). In Fig. 2, we analyse how the class-wise average forgetting evolves throughout incremental training when different methods are employed. We observe that the proposed CIL approach

[2] We used the code from https://github.com/mmasana/FACIL to perform gridsearch.

Table 4: *acc1:acc2* top-1 accuracy (%) where *acc1* is the class-wise average accuracy (Eq. 2) and *acc2* is the class- and step-wise average accuracy (Eq. 3). Both measures are computed at the end of the last incremental step.

Method	CIFAR100-Res18			CIFAR100-ResNet32 [13]		Imagenet-Subset	
	5 Steps	10 Steps	20 Steps	5 Steps	10 Steps	10 Steps	20 Steps
iCarl [13]	54.1:64.8	51.1:62.3	41.2:56.3	-:-	-:-	54.7:68.6	48.7:63.9
UCIR [5]	51.1:61.4	46.0:57.5	38.3:51.3	-:63.4*	-:60.2*	57.5:66.6	45.5:57.3
PASS [21]	56.5:65.1	47.6:60.8	47.3:58.7	51.1:58.9	44.4:53.2	58.1:68.2	47.2:61.4
SDC [19]	57.6:66.2	52.3:62.7	48.8:59.2	51.4:59.4	47.0:54.4	58.6:68.6	47.1:61.0
DER [18]*	-:73.2	-:72.8	-:-	-:68.5	-:67.1	74.9:78.2	-:-
Feat. Drift (GM-MLP)	57.9:66.3	54.5:63.7	50.6:61.2	53.1:60.2	50.9:58.3	59.1:68.1	50.6:62.6
Sem. Drift (GM-MLP)	58.3:66.0	54.2:63.4	50.9:60.8	53.3:60.3	51.2:58.4	59.3:69.2	50.4:63.4
Fusion (GM-MLP)	59.4:66.9	56.0:64.8	51.9:61.5	53.9:60.9	52.0:58.5	59.7:69.3	51.5:63.6
Feat. Drift (VM-VAE)	57.0:65.8	53.7:63.0	51.1:60.9	53.8:60.2	50.9:58.3	59.3:68.8	50.9:62.8
Sem. Drift (VM-VAE)	58.2:66.7	55.4:63.3	51.7:61.6	54.1:60.9	51.3:58.9	59.5:69.3	51.0:63.1
Fusion (VM-VAE)	58.7:66.8	56.9:65.1	51.8:61.6	54.2:61.3	52.1:59.1	60.2:70.0	51.6:63.7

* Numerical values were directly taken from [18].

based on representation drift modelling mitigates forgetting more efficiently than the majority of the competitors (only iCARL [13] shows superior performance). Nonetheless, we remark that iCARL [13] leverages replay data from the past to address forgetting. Furthermore, iCARL [13] yields a lower or comparable classification accuracy with respect to our approach, suggesting that a focus of iCARL [13] on preserving past knowledge (*i.e.*, *stability*) is accompanied by a less efficient learning of novel classes (*i.e.*, *plasticity*).

Finally, we propose an analysis using task forgetting computed at the end of incremental training (Fig. 3). We notice how our CIL method causes overall less forgetting than most of the competing approaches, providing an improved performance equally shared among all tasks. This is especially noticeable when evaluation is performed on the CUB200 dataset, where even the SotA PASS and SDC methods induce almost total forgetting of the oldest tasks.

3.2. Additional Ablation Studies

3.2.1 Analysis of Semantic Drift Models

We investigate how our method proposed to model representation drifts captures and preserves semantic relationships between feature representations of old and new classes. For this purpose, we compute prototypical representations of novel classes on the training data available at an incremental step t, and estimate the *revived* prototypes of old categories by modeling semantic and feature drifts (employed individually or fused). Then, we express inter-class relationships in the form of Euclidean distance between prototypes of past and new classes, and observe the evolution of such distance throughout an incremental step t. In particular, we compute distance values between pairs of class prototypes at the beginning and at the end of the same incremental step t, and measure their change across the optimisation interval (*i.e.*, in the form of absolute value of the difference between the estimates performed at the beginning and end of step t).

In Fig. 4, we report the results of the analysis carried out when performing 20 incremental training steps on the CIFAR100, TinyImageNet and CUB200 datasets. We focus on analyzing the change of semantic relationship during the first incremental step (*i.e.*, $t = 1$). We notice how prototypes estimated by leveraging the modeled semantic representation drift tend to more effectively preserve inter-class relationships with respect to novel classes. By utilising feature drift alone, in fact, we notice that class representations tend to modify their interconnections. Nonetheless, the proposed model fusion allows to better identify and retain inter-class relationships, whereas keeping prototypes fixed (as in PASS) causes a greater impairment of inter-class relationships as new representations are learned.

3.2.2 Analysis of Feature Drift Models

We analyse the efficacy of the proposed model in estimating the feature drift undergone by evanescent representations of old classes when novel classes are learned. To this end, we compute the Euclidean and cosine distances between estimated (*revived*) prototypes of old classes and their reference (*i.e.*, evanescent) representations at each incremental step (Fig. 5). We provide per-step class-wise average distance values (main curves), as well as the maximum and minimum class values that identify, respectively, the upper and lower bound of the shaded regions for each setup. Feature prototypes of old classes are estimated by computing class wise averages of feature representations over training data when they are available at an incremental step, which can then be simply fixed for the rest of the training (as done in PASS [21]), or can be updated by SDC [19] or by the proposed drift models. Evanescent prototypical representations of the same old classes are instead computed over the test set (unavailable during training).

We replicate the analysis on the CIFAR100, TinyImageNet and CUB200 datasets to provide a robust evaluation. The results show that our proposed methods can track

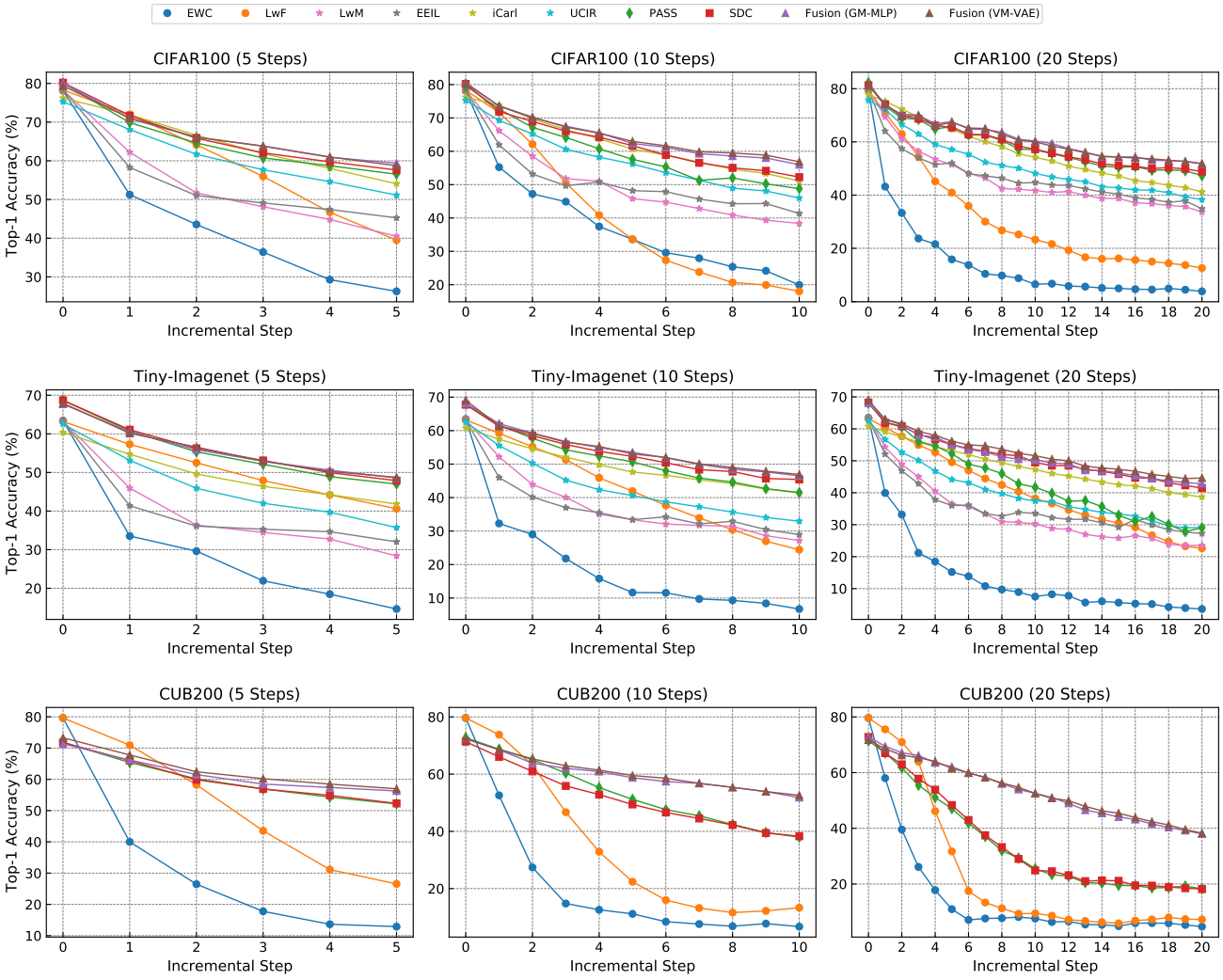


Figure 1: Per step average top-1 accuracy (%) on the CIFAR100, CUB200-2011 and TinyImageNet datasets.

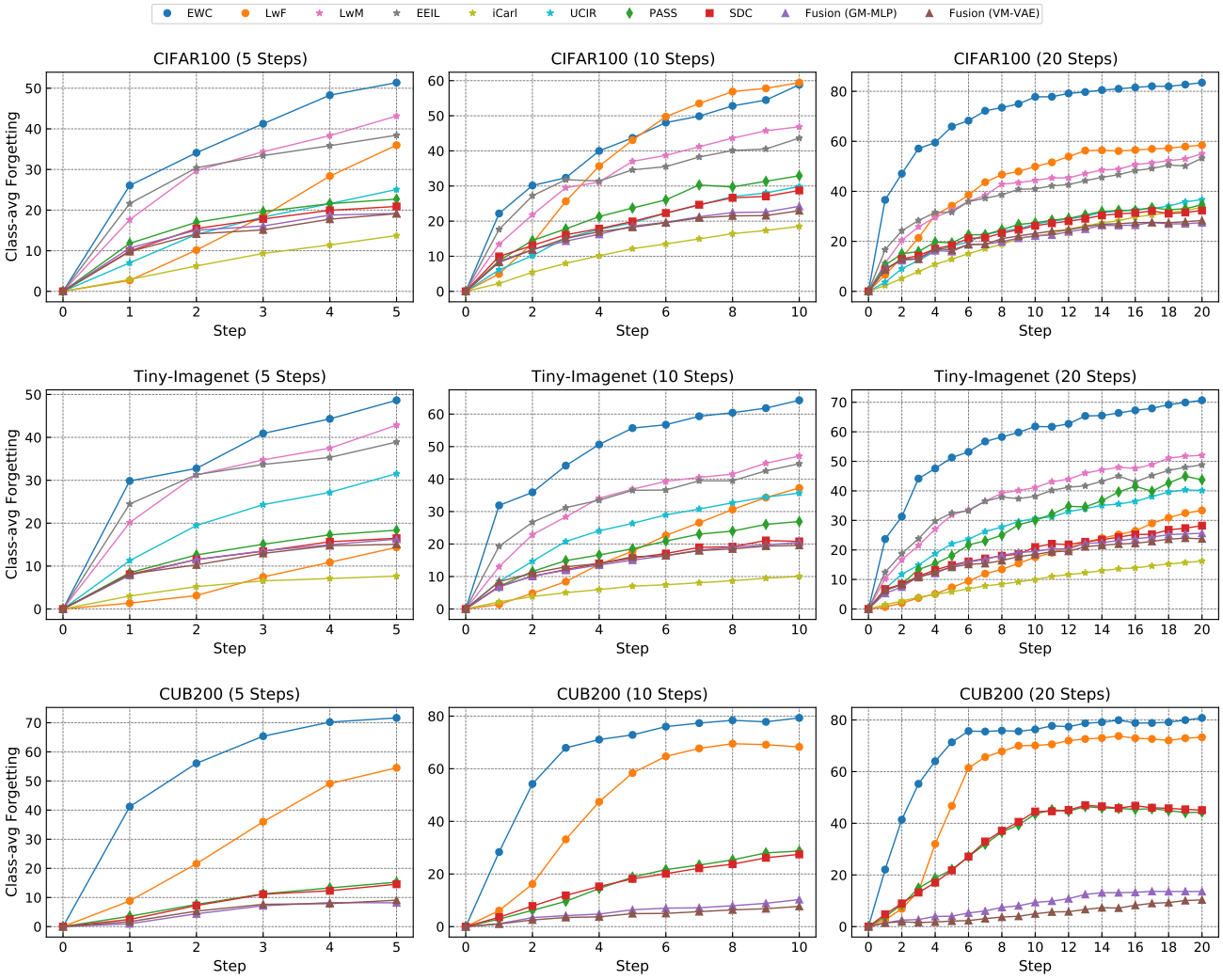


Figure 2: Per step average class forgetting (%) on the CIFAR100, CUB200-2011 and TinyImageNet datasets.

the trajectory of evanescent prototypes more efficiently (in terms of geometric distances) compared to the SotA PASS and SDC methods, by modeling the evolution of the representations (*i.e.*, feature drift). Furthermore, we observe that the improved accuracy with respect to the SotA is shared among the three datasets chosen for evaluation. In particular, we remark the noticeable improvement experienced on the CUB200 dataset. Our method, in fact, provides much lower Euclidean distance between revived and evanescent representations, whose average value is kept almost constant as incremental training progresses and new classes are introduced. The same trend can be observed for the cosine similarity of the estimated and evanescent representations, whose value tends to be steady through the incremental training and much closer to the upper bound when our method is adopted.

In Fig. 6, we compare the Euclidean distance between the estimated and evanescent prototypes for different number of total incremental steps, on the CIFAR100, TinyImageNet and CUB200 datasets, when employing model fusion with both GM and VM. We observe that our method always outperforms PASS, in all the evaluation setups. In addition, we notice that by employing VM to identify representation drifts, we reach the largest improvement over PASS, which translates into the revived evanescent representations closer to their unknown reference version. We also highlight once more the noticeable accuracy provided by our model for the 20-step setup on the CUB200, where our approach jointly shows the largest improvement over the SotA accuracy by $\sim$20%.

Finally, we visualise 2D embeddings of feature vectors of the first four observed classes as computed by the feature extractor on the same set of input samples at different incremental steps. To project high-dimensional feature vectors to a 2D space, we use Isomap [16]. Results are reported in Fig. 8 for the CIFAR100, TinyImageNet and CUB200 datasets. We observe that our proposed method allows to estimate revived prototypical representations that tend to be projected closer to their evanescent versions compared to PASS (especially on the CIFAR100 and CUB200 benchmarks). This shows that we can approximate the trajectory of evanescent representations of old classes, without having access to training data of such categories, by modeling representation drift.

3.2.3 Analysing How Learned Evanescent Representations Affect Classification Accuracy

In this section, we investigate the relationship tying the accuracy of the classification model and the *normalised* distance between the revived and evanescent prototypes of old classes. Normalisation is performed by dividing individual distances computed for single past classes by the average distance among all the past classes. We then provide the average of normalised distance values at each incremental step. We evaluate the accuracy of the proposed method when fusing semantic and feature drift models and adopting GM to identify representation drift, alongside with that of the SDC and PASS. Results are reported in Fig. 7.

We observe that classification accuracy measured at each incremental step and distance between the estimated and evanescent prototypes are negatively correlated, with similar trends shared by the different methods being analysed, across the CIFAR100, TinyImageNet and CUB200 benchmarks. It is worth noting that our method and SDC display very similar correlation patterns, whilst the latter reaching lower accuracy and higher distance values at the final incremental step. Thereby, we argue that our proposed method yields superior performance compared to the SotA PASS and SDC by more accurately tracking and modeling evanescent old class prototypes.

3.2.4 Statistical Analyses of Representations

We explore how well the estimated revived prototypes of old classes exemplify feature representations of samples of such categories (which are unavailable during training at incremental steps). To this end, we first compute probability distributions over the set of old classes based on the Euclidean distance between feature representation and class prototypes by

$$p_F(c) = \exp(-||F-\pi_c)||_2/\zeta) / \sum_j \exp(-||F-\pi_j)||_2/\zeta), \tag{5}$$

where $F \in \mathcal{F}_{old}$ is the feature representation of a test sample of $\mathcal{C}_{old}$ as produced by the current feature extractor, $\{\pi_j\}_j$ are revived prototypes of $\mathcal{C}_{old}$ and ζ is set to 0.1.

We analyse the change of entropy (H) and cross-entropy (CE) of p_F across incremental steps in Fig. 9. The cross-entropy is computed w.r.t. to the one-hotted ground-truth distribution of the labeled input sample corresponding to $F \in \mathcal{F}_{old}$. In addition, for each old class $c \in \mathcal{C}_{old}$, we compute the mean H and CE of p_F, *i.e.*, we average over all F corresponding to the same c.
We observe that our method provides higher H and smaller CE compared to PASS, and this trend is shared across CIFAR100, TinyImageNet and CUB200. This result suggests that information capacity of representations learned by our methods increases along with classification accuracy more rapidly compared to the SotA as models are incrementally trained.

To further validate this claim, for each old class c, we compute the probability distribution over feature represen-

tations of input samples of *all* past categories, defined by

$$p_c(F_i) = \exp(-||F_i - \pi_c)||_2/\tau) / \sum_j \exp(-||F_j - \pi_c)||_2/\tau), \quad (6)$$

where $F_j \in \mathcal{F}_{old}$ are feature representations of test samples and $\{\pi_j\}_j$ are revived old-class prototypes. In addition, τ is set to 0.1. In Fig. 9, we report entropy values of p_c across different incremental steps. Once more, we observe that our method causes entropy of p_c to reach higher values compared to PASS, indicating that prototypical representations revived by our method are more informative than their fixed counterparts, while still being representative of the corresponding class, as suggested by the lower CE of p_F.

3.2.5 Analysis of Learning Curves for Training Representation Drift Models

We analyse the learning curves of semantic and feature drift models. In particular, in Fig. 10, we report the convergence values achieved by the total loss function used to train drift models at different epochs across multiple incremental steps (for the CUB200 dataset and a setup employing 5 incremental steps). For semantic drift models, we show the results for the total loss (denoted by *Fusion (VAE/MLP)*) comprising L_s and L_{fus}. As for feature drift models, we present the individual behaviour of the loss L_f denoted by *Only Feat Drift (VAE/MLP)* (without aggregating the loss L_{fus}), along with combination of the loss functions L_f and L_{fus} denoted by *Fusion (VAE/MLP)*.

In the results, we observe that the convergence point stabilises in all the analysed setups as learning progresses during each incremental step. This suggests that, as the classification model learns representations of new classes throughout an incremental step, drift models converge to a stable configuration, in which they can provide a robust estimate of representation drift. Moreover, we observe that the VMs (VAE) converge faster than GMs (MLP).

References

[1] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 3

[2] Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 3

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 3

[4] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[5] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 4

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 2

[7] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 2

[8] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. 3

[9] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 3

[10] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 3

[11] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 3

[12] Hadi Pouransari and Saman Ghili. Tiny imagenet visual recognition challenge. In *CS231N course, Stanford Univ., Stanford, CA, USA*, 2014. 3

[13] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3, 4

[14] Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 3

[15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 1

[16] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 7

[17] Catherine Wah, Peter Welinder Steve Branso and, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 3

[18] Shipeng Yan, Jiangwei Xie, and Xuming He. DER: dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 4

[19] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 4

[20] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *CoRR*, abs/1706.02262, 2017. 2

[21] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3, 4

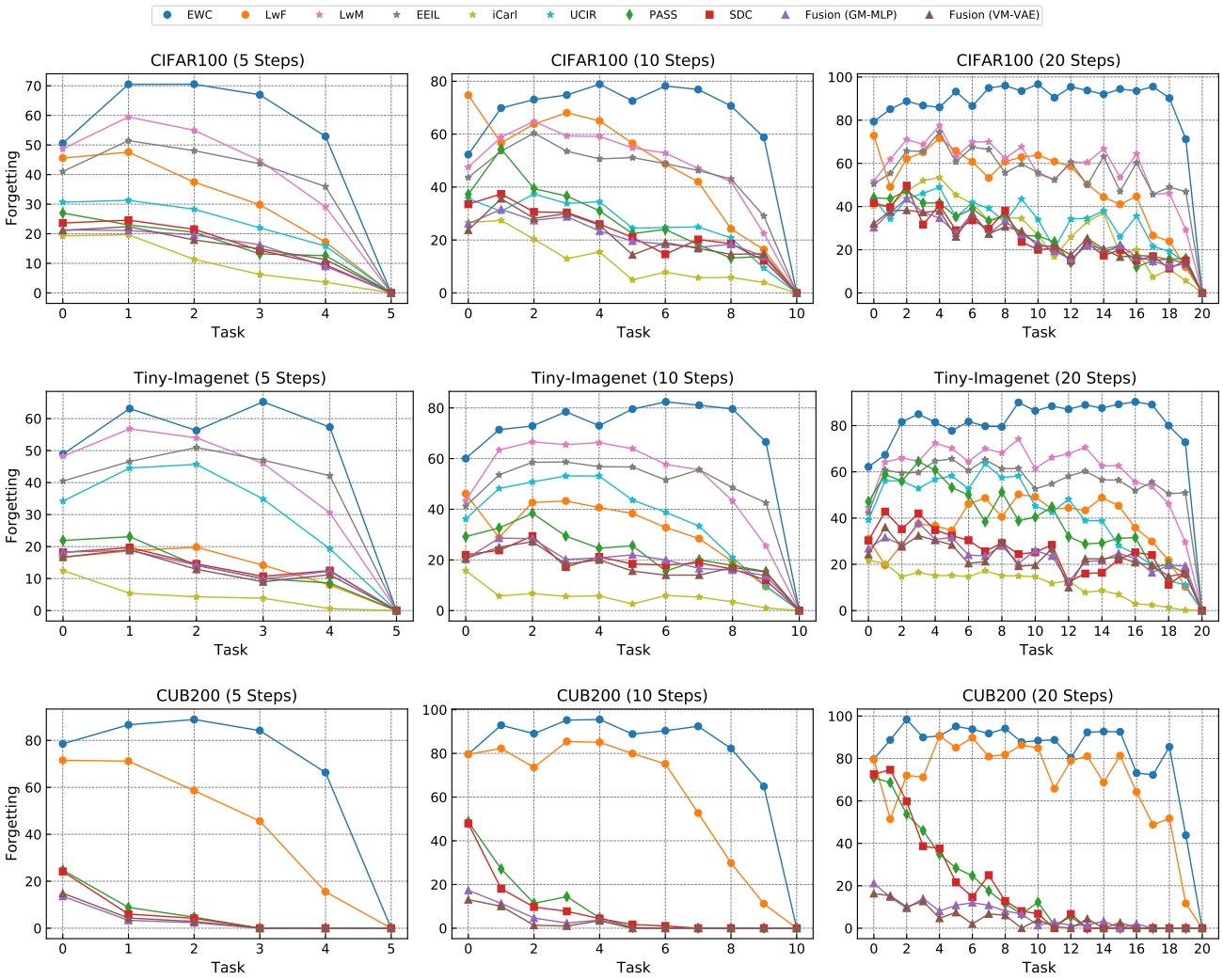
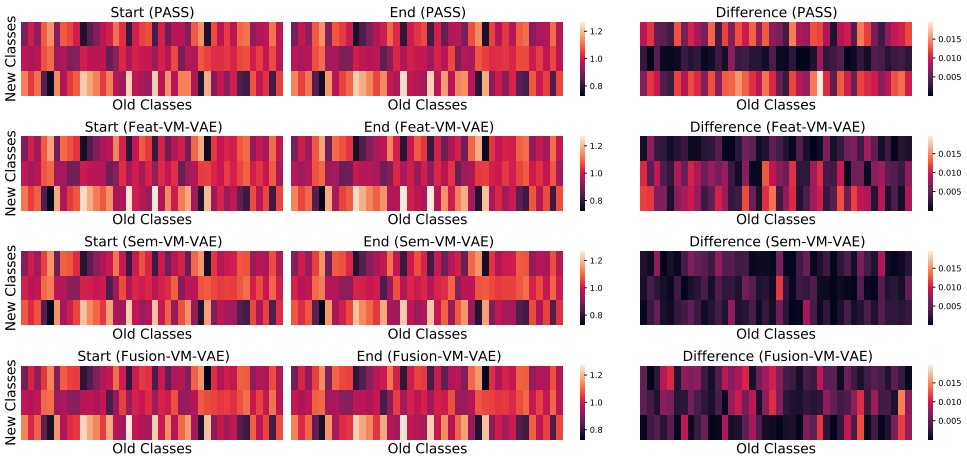
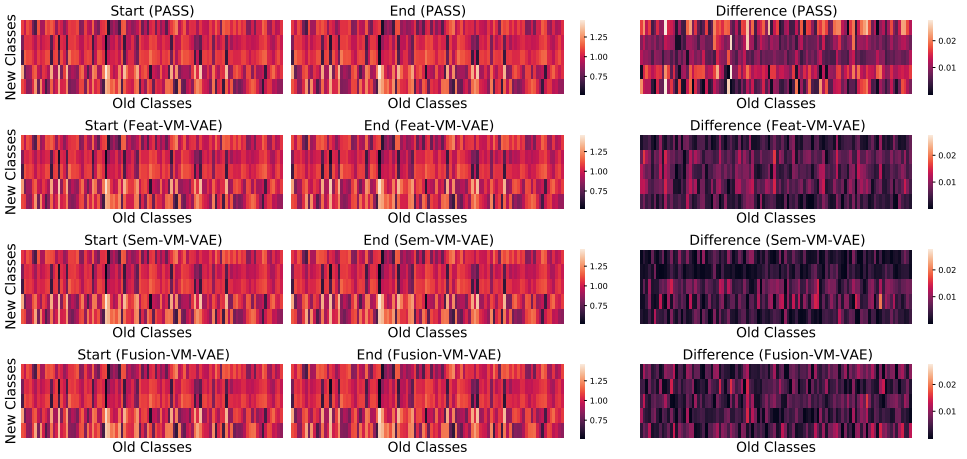


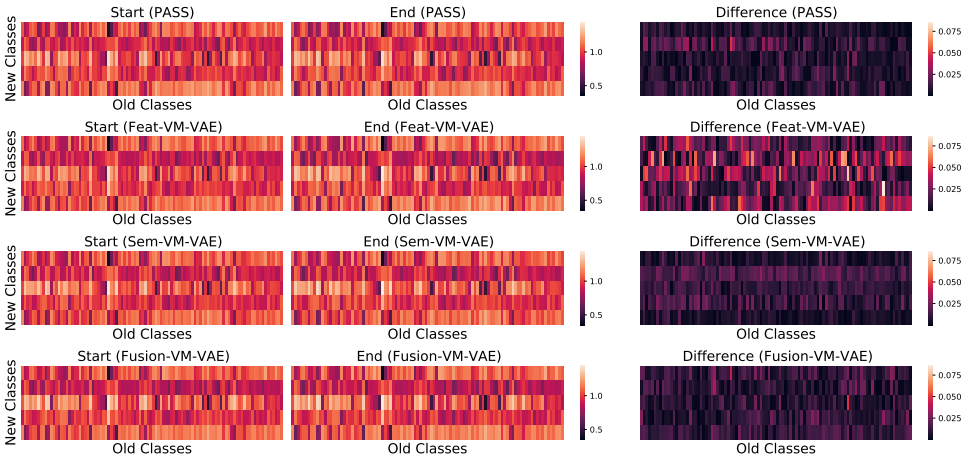
Figure 3: Per task average forgetting (%) computed at the end of incremental training on the CIFAR100, CUB200-2011 and TinyImageNet datasets.



(a) CIFAR100



(b) TinyImageNet



(c) CUB200

Figure 4: Analysis of normalised distance between estimated prototypes of classes seen at steps $t = 0$ and $t = 1$, captured at the beginning (left) and end (mid) of step $t = 1$. We report the absolute value of the difference of the two measures (right). We replicate the analysis for the 20-step incremental setup, over the CIFAR100 (top), TinyImageNet (middle) and CUB200 (bottom) datasets.

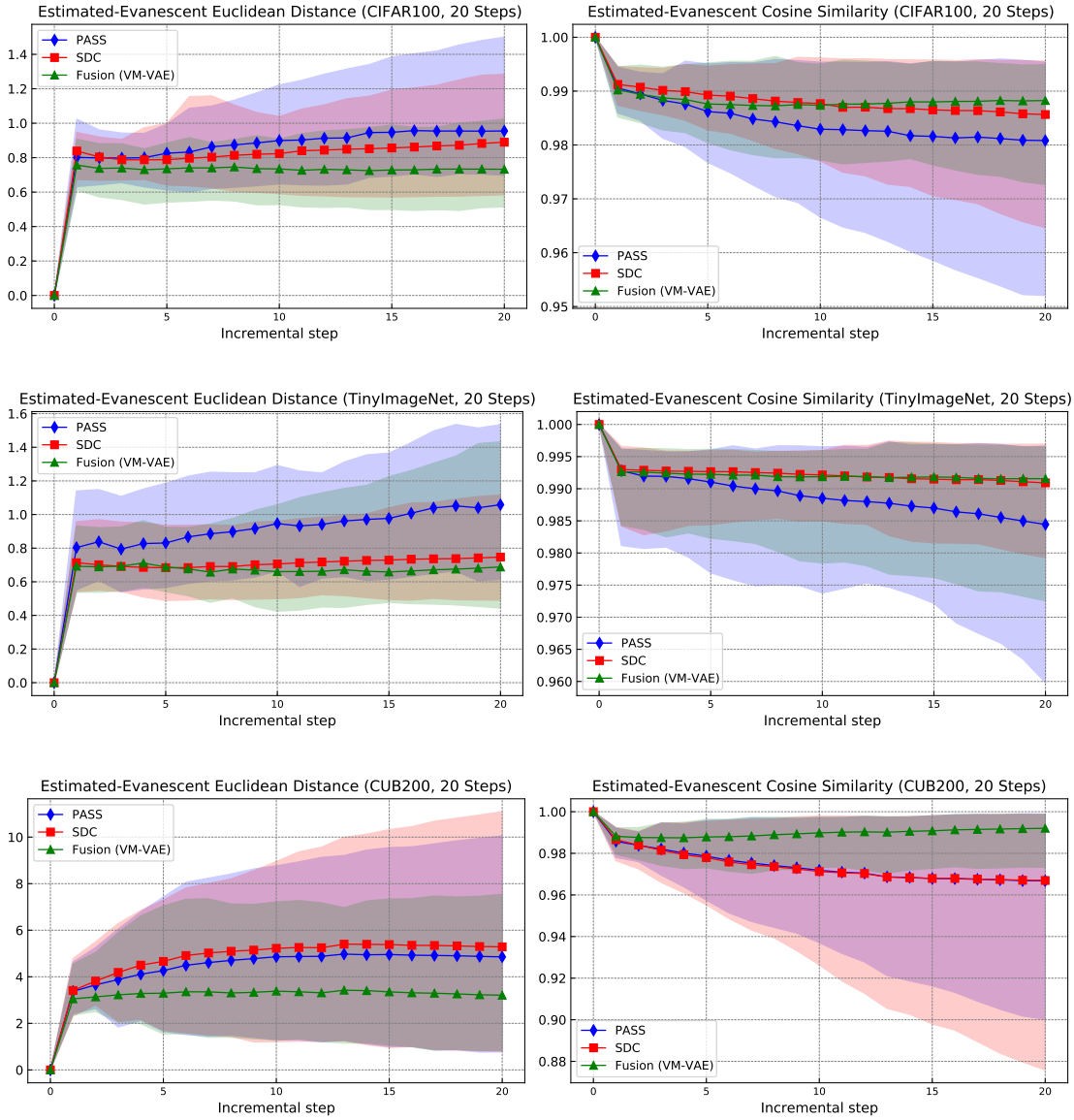


Figure 5: Analysis of average distance between estimated (*revived*) and evanescent prototypes of old classes.

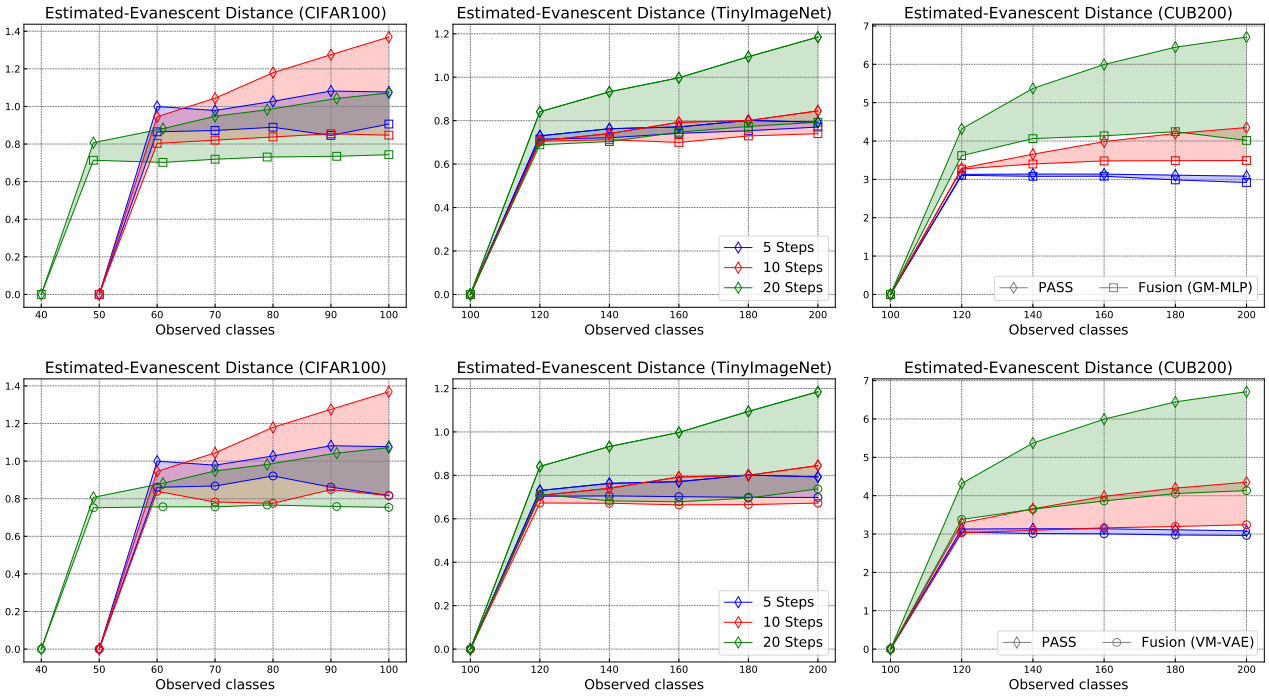


Figure 6: Average Euclidean distance between the estimated (*revived*) and evanescent prototypes of old classes.

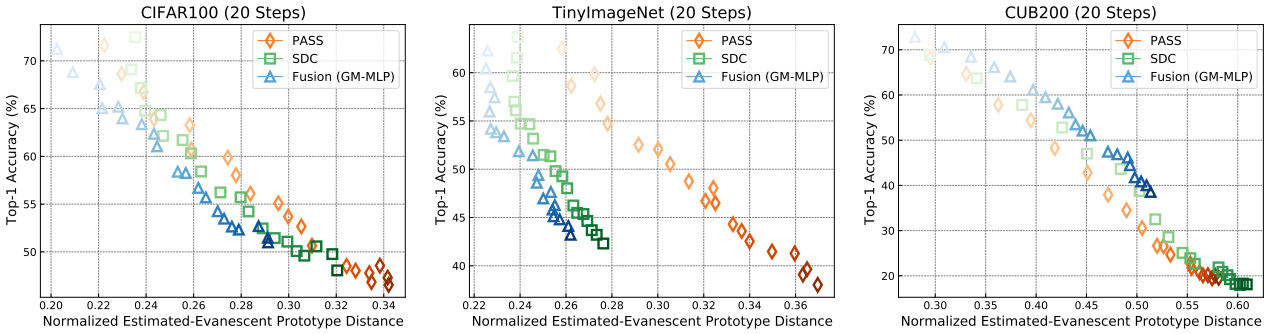
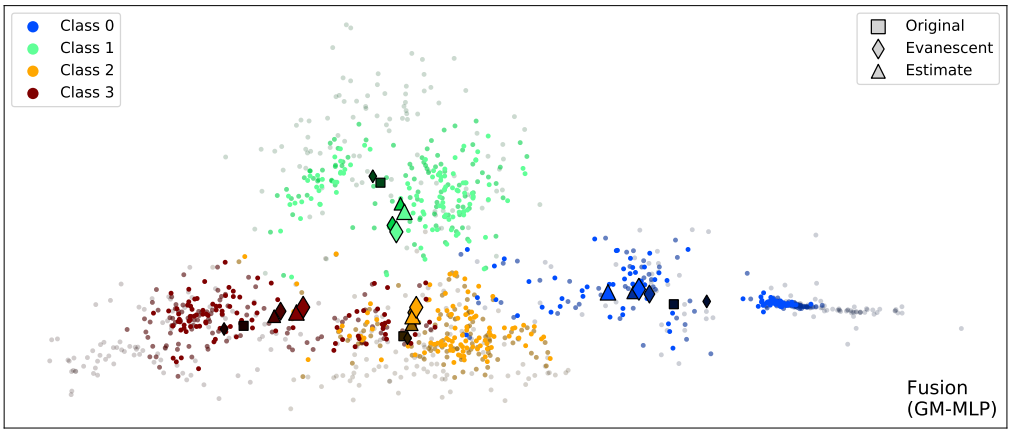
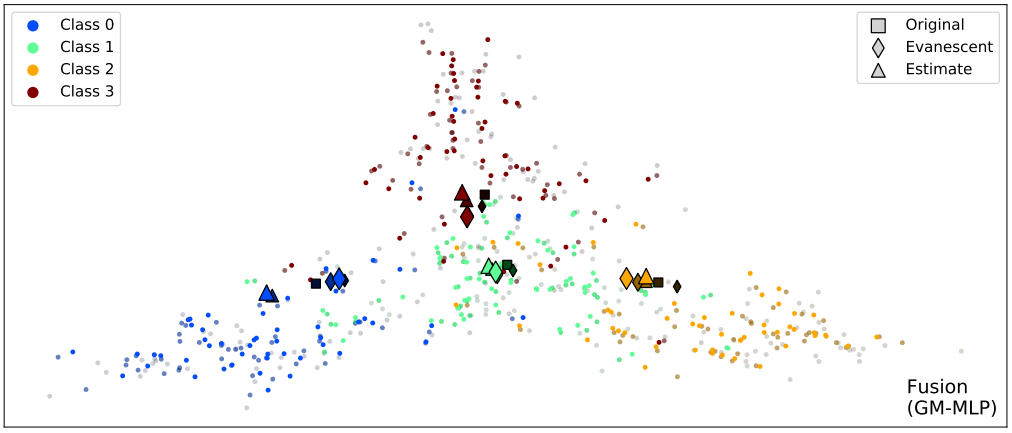


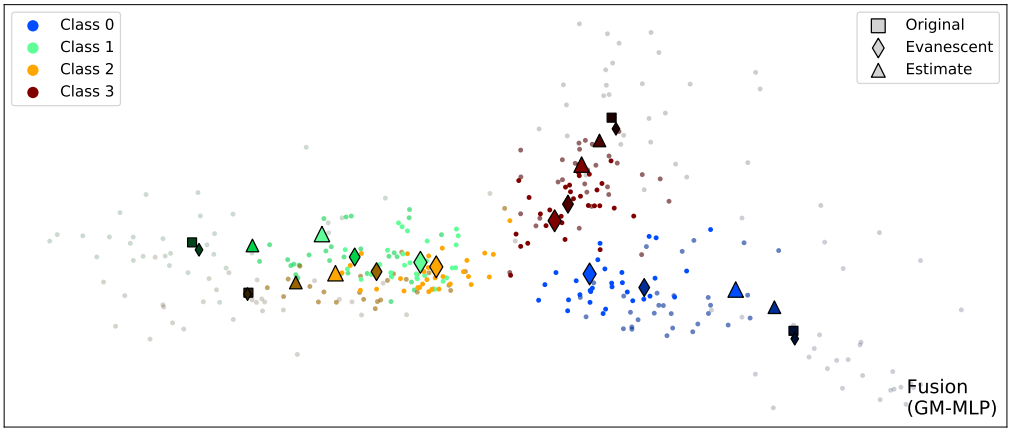
Figure 7: Relationship between top-1 accuracy (%) and *normalised* Euclidean distance between estimated and evanescent old-class prototypes. Each point depicts a single training phase, and the decrease in transparency indicates progressively increasing incremental steps. For each step, accuracy values have been averaged over all classes observed so far, and distances are averaged over all past classes.



(a) CIFAR100

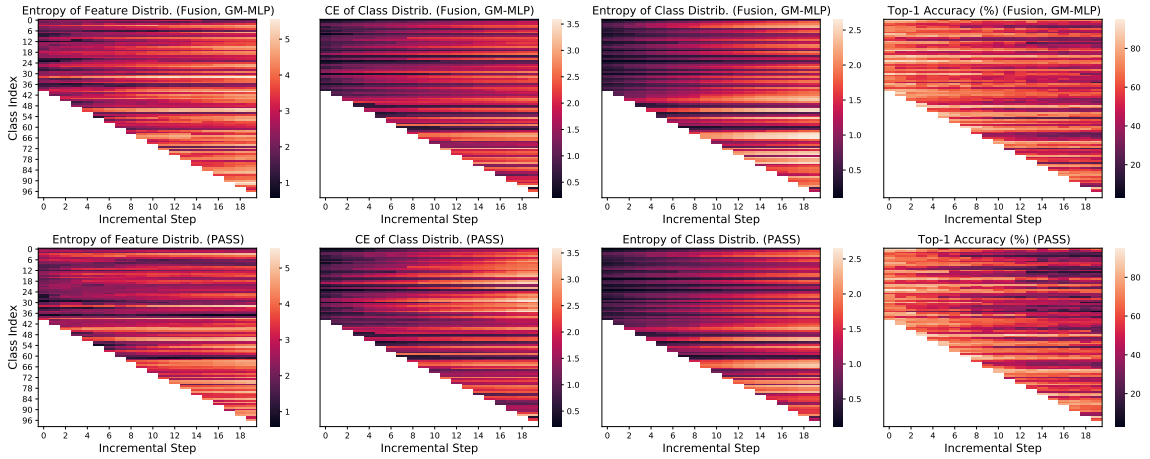


(b) TinyImageNet

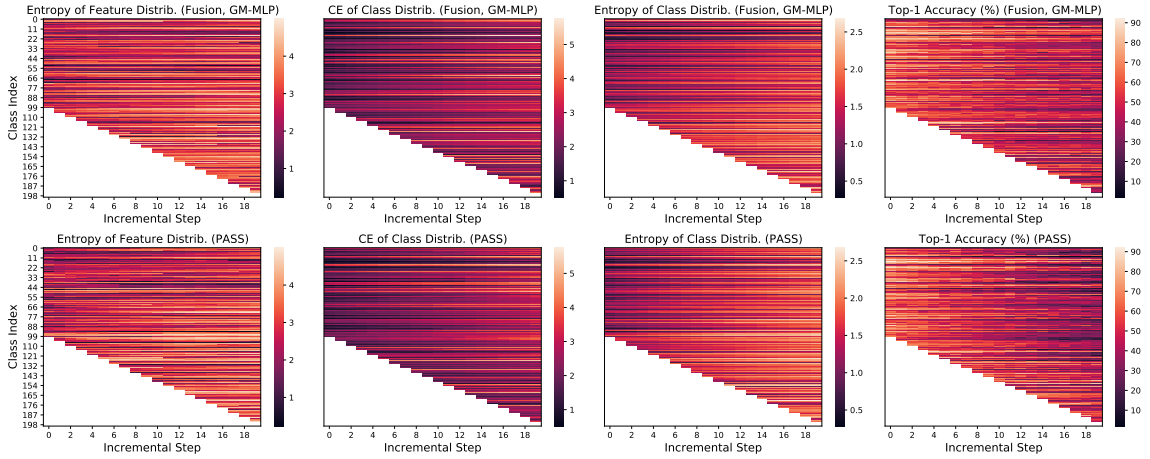


(c) CUB200

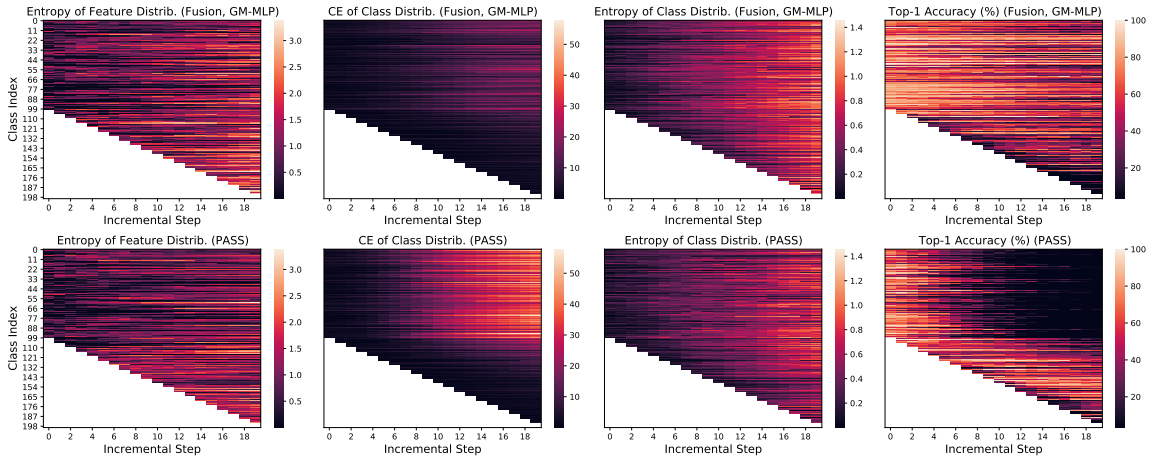
Figure 8: Feature representations of the first four learned classes (CIFAR100 (top), TinyImageNet (middle) and CUB200 (bottom), 20 incremental steps) extracted from samples of the test set (dots), along with their prototypes computed over the available training data (squares), over the test data (diamonds) and the estimated prototypes (diamonds). Decrease in transparency and increase in brightness indicate that representations are extracted at progressively increasing incremental steps (*i.e.*, at steps 0, 10 and 20).



(a) CIFAR100



(b) TinyImageNet



(c) CUB200

Figure 9: Left: average entropy of $p_c(F_i) = \exp(-||F_i - \pi_c)||_2/\zeta) / \sum_j \exp(-||F_j - \pi_c)||_2/\zeta)$. Mid-left and mid-right: entropy and cross-entropy w.r.t. GT of $p_F(c) = \exp(-||F - \pi_c)||_2/\tau) / \sum_j \exp(-||F - \pi_j)||_2/\tau)$. Each feature F_j is extracted from a test image belonging to an old class, and π_j are the estimated prototypes of old classes. Right: Top-1 accuracy for the CIFAR100 (top), TinyImagenet (middle) and CUB200 (bottom) (all models were evaluated for 20 incremental steps).

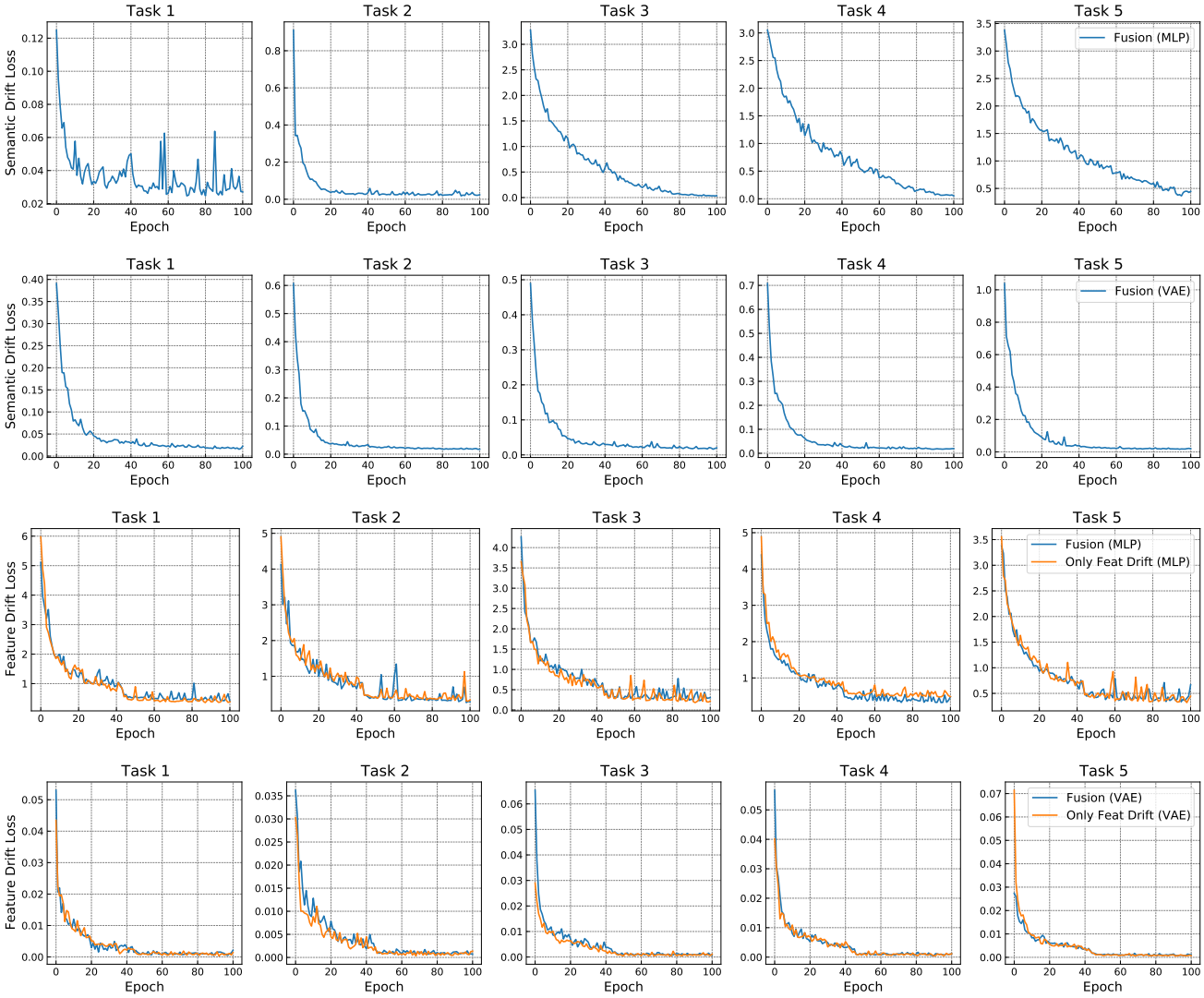


Figure 10: Total loss of semantic and feature drift models measured throughout incremental training (CUB200, 5 incremental steps).