Insights from the Future for Continual Learning

Arthur Douillard[1,2], Eduardo Valle[3], Charles Ollion[2,4], Thomas Robert[2], Matthieu Cord[1,5]

[1]Sorbonne Université, [2]Heuritech, [3]Univerty of Campinas, [4]CMAP École Polytechnique, [5]valeo.ai

Abstract

Continual learning aims to learn tasks sequentially, with (often severe) constraints on the storage of old learning samples, without suffering from catastrophic forgetting. In this work, we propose prescient continual learning, a novel experimental setting, to incorporate existing information about the classes, prior to any training data. Usually, each task in a traditional continual learning setting evaluates the model on present and past classes, the latter with a limited number of training samples. Our setting adds future classes, with no training samples at all. We introduce Ghost Model, a representation-learning-based model for continual learning using ideas from zero-shot learning. A generative model of the representation space in concert with a careful adjustment of the losses allows us to exploit insights from future classes to constraint the spatial arrangement of the past and current classes. Quantitative results on the AwA2 and aP&Y datasets and detailed visualizations showcase the interest of this new setting and the method we propose to address it.[1]

1. Introduction

Continual learning models contrast with traditional models by approaching a sequence of tasks incrementally. With limitations (often severe) on the training data they can retain, those models must avoid catastrophic forgetting of past tasks [39, 14], while remaining receptive to new tasks. Many approaches exist to counteract forgetting: keeping a limited amount of training data from previous tasks [38, 6]; learning to generate the training data [24, 41]; extending the architecture for new tasks [49, 29]; using a sub-network for each task [13, 16, 22]; and constraining the model divergence as it evolves [25, 32, 3, 31, 38, 6, 11]. Those approaches are often complementary.

New Setting: We propose a challenging new setting, *prescient continual learning*, in which the model must perform well not only for present and past tasks, but also for *future* ones, both avoiding catastrophic forgetting (using a limited

number of training samples for past classes), and giving the best possible estimates for the future classes (using no training samples at all). To make the setting possible, the model must know the classes and have some prior information about them. Indeed, Aljundi et al. [4] remark that the ability to make room for future classes is a key limitation of current continual learning models, and propose a regularization loss to make the model more "selfless", explicitly leaving capacity for future classes in the representation. Han et al. [19] proposed a setting where the training samples from all classes are present from the beginning, but the labels become available incrementally. In a way, our setting is the inverse: we know which labels we are going to encounter, but the training data for those labels arrive incrementally.

Real-World Applications: This setting has several actual applications, where the classes to be predicted (and some information about them) are known from start, but training data about those classes only arrives incrementally. The starting definition of the classes will vary according to the application and opportunity: e.g., a zoological expedition to catalog animals could employ encoded attributes (e.g., 'brown-fur', 'can swim'), position in a phylogenetic tree (e.g., iNaturalist [1]), or even free-form textual description (e.g., Wikipedia article about the animal, encoded with Word2vec [34]). That latter option extends the setting to many applications where explicit attributes are missing. The example of animal classification may be adapted to cover many other topics such as fashion classification with meta-attributes of clothing, movies classification with actors, genre, and synopsis, etc. Such setting allows training the model in different phases without having to wait for the completion of the data collection to have an usable model. Furthermore it enables exploiting privileged metadata [44], which are generaly available in industrial projects.

Zero-Shot Learning: To address the proposed setting, we take inspiration from zero-shot learning [27, 46], which allows classifying examples from unseen classes by combining a vision model with an embedding possessing knowledge about the classes (e.g. a word embedding [34, 36] or an attributes matrix). Although several approaches exist for zero-shot learning, we will focus on generating a representation for the future classes [5, 26, 47]. The framework of

---

[1]Code is available at:
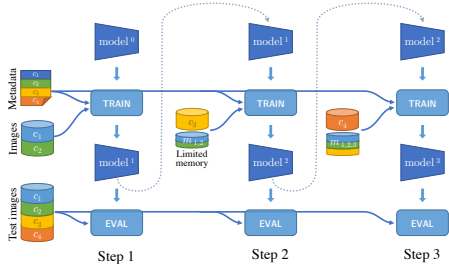github.com/arthurdouillard/incremental_learning.pytorch

Figure 1: The enriched continual learning setting proposed in this work. At each training task, we learn a new set of classes, but the model is evaluated on *all classes* — past, present, and future. The model has to avoid catastrophic forgetting of past classes (using a limited number of rehearsal training samples), as well as make a good guess for future classes (using no training samples at all).

representation learning will allow us to combine continual and zero-shot learning seamlessly, as we advance through the tasks, and future classes become present classes, and then past classes. Moreover, we will be able to use *ghost features*, predicted features for the future classes, to make room in the representation space for future classes. All those goals are integrated into a simple, streamlined model due to a careful construction of the losses. While Chaudhry et al. [8] proposed a basic addition of zeroshot to continual learning that only improves performance on future classes, our model aims to improve the representation of *all* classes with insights from the future, resulting in a *generalization* of continual learning. Furthermore, Chaudhry et al. evaluate each task separately, using a task descriptor, while our evaluation set comprises the entire set of classes, a more realist and challenging setting. The work of Wang et al. [45], released simultaneous to ours, proposed an integration of zeroshot into continual learning limited to two tasks, severely curtailing the opportunity to exploit insights from the future. Furthermore it has been shown that continual training longer than a few tasks follows a more complex evolution [10]. Skorokhodov et al. [42], a concurrent work, combine zeroshot with continual learning but don't consider the regularization effect of selflessness that we investigate. Pernici et al. [37], another concurrent work, investigate the notion of future/unseen classes but use negative samples to train the classifier nodes of those classes while we use positive samples through ghost features.

Contributions: The contributions of this work are two-fold: (1) we propose a new challenging setting, *prescient* continual learning, where the model must perform well on past, present, and *future* classes; (2) we propose our *ghost model* to address that setting, integrating continual and zeroshot learning into a coherent whole. We evaluate those contributions in comprehensive experiments, showcasing both

the intuitive appeal of our model, and its performance.

2. Setting: prescient continual learning

In continual learning, a classifier is trained in multiple steps called *tasks*. Each task $t \in 1 : T$ comprises a set of new classes C^t. The model is evaluated after each task t, traditionally, on all classes seen so far $C^{1:t}$. Our *rehearsal memory* severely limits the training samples we are allowed to keep from the past classes: following [21, 11], we allow a small constant number of samples s per past class. We must take the maximum advantage from those limited rehearsal data to avoid catastrophic forgetting.

We propose an enriched experimental setting, prescient continual learning, in which each task is evaluated on *all classes* $C^{1:T}$: past ($C^{1:t-1}$), present (C^t), and future ($C^{t+1:T}$). In that challenging new setting, we must not only avoid the catastrophic forgetting of past classes (using the limited rehearsal training samples), but also give our best estimates for future classes (using no training samples at all). That will only be possible if we have some prior information about the classes, e.g., their hierarchy in a semantic network (like WordNet), an associated word embedding (like Word2vec), or an attribute matrix. Such setting is illustrated in Fig.1. We will sometimes shorthand the set of past and present classes $C^{1:t}$ as the *seen* classes, and the set of future classes $C^{t+1:T}$ as the *unseen* classes. We denote individual samples by a superscript $\mathbf{x}^{(i)}$, and the class label by a subscript $\mathbf{x}_c$. We denote on which parameters a loss is applied by a subscript $\mathcal{L}_\Theta$.

3. Ghost model

To address the setting described in the previous section, we propose our *ghost model*, comprising three components: a convolutional feature extractor f, a feature generator g, and a classifier *clf*. The feature extractor is the backbone of the model: it learns to extract a feature vector from actual samples that can be fed to the classifier. The generator learns the distribution of the features for all classes, aiming to generate plausible samples of features for the future classes. The classifier makes the final decision for all classes: past, present, and future. The classifier is trained on future classes with features sampled from the generator, which we call *ghost features* (since they must be "hallucinated" from the seen classes and some prior knowledge about the classes).

All three components are learned continuously throughout the T tasks; we denote the component learned at task t with a superscript: f^t, g^t, and clf^t. To avoid catastrophic forgetting of past classes, we constrain the model's evolution, placing a cost on the divergence between f^{t-1} and f^t. To make the best guess for future classes, we take inspiration from zero-shot learning, training the classifier with the

ghost features. Finally, we adjust the classifier so that the ghost features "make room" in the representation space for future classes.

The next subsections detail the model, explaining the different terms in the loss that act in concert to obtain the desired balance.

3.1. Base model for continual learning

The base model is a representation-based architecture with a convolutional feature extractor $\mathbf{h} = f(\mathbf{x})$ (where $\mathbf{x}$ is the input image and $\mathbf{h}$ is the feature vector) and a cosine classifier *clf* [33, 21], a fully-connected layer with the dot-product replaced by a cosine similarity:

$$clf(\mathbf{h}, \boldsymbol{\theta})_c = \hat{\mathbf{y}}_c = \frac{\langle \mathbf{h}, \boldsymbol{\theta}_c \rangle}{\|\mathbf{h}\|_2 \|\boldsymbol{\theta}_i\|_2} . \tag{1}$$

Remark that $\boldsymbol{\theta}_c$, the parameter vector for the c-th class in *clf* ($c \in C^{1:t}$), may be interpreted as a representative or proxy for that class. The classification loss could be either a cross-entropy loss preceded by softmax activations, or an NCA loss [15, 35, 11]:

$$\mathcal{L}^{\text{nca}}_{\Theta_f, \Theta_{clf}} = \left[-\log \frac{\exp\left(\hat{\mathbf{y}}_y - \delta\right)}{\sum_{c \neq y} \exp \hat{\mathbf{y}}_c} \right]_+ . \tag{2}$$

To counteract catastrophic forgetting, we must limit the evolution of the model. We impose — as usual for continual learning — a distillation loss between the previous model iteration ($t-1$) and the current one (t). We evaluate several distillation losses applied to intermediate and final outputs of the feature extractor [11, 21]. The final loss of the base model is:

$$\mathcal{L} = \mathcal{L}^{\text{nca}}_{\Theta_f, \Theta_{clf}} + \lambda_1 \mathcal{L}^{\text{distil}}_{\Theta_f} . \tag{3}$$

3.2. Capacitating ghost model for future classes

The base model can deal with both present classes (with training samples constrained only by their availability in the training set) and past classes (with training samples severely constrained by the rehearsal memory). As discussed, the introduction of a distillation loss prevents catastrophic forgetting of the latter. We will now address *future classes*, with *no* training samples available. First, taking inspiration from zero-shot learning, we will use prior information about the classes to generate *ghost features*, plausible stand-ins for the unseen future classes' features. Next, we will adapt the classifier to incorporate those ghost features into the learning objective seamlessly. The representation learning framework will allow us to integrate the entire learning apparatus into one coherent loss.

Generator. The generative model estimates the distribution of the unseen classes directly in terms of their features (instead of the input images). For the feature generation to work, we must have exploitable prior information

about the classes, more precisely, we must be able to map the class labels c into a *class attribute space* that makes semantic sense. The exact way to perform that mapping will be data-dependent, but most often, either we will have an explicit set of attributes linked to each class (color, size, material, provenance, etc.), or we will be able to extract a latent semantic vector, using a technique like Word2vec [34, 36]. The generator learns to link the attributes of the *seen* classes to the actual feature vectors extracted from the training samples of those classes. Thus the first generator training must happen after the features extractor (its ground-truth) is learned. The generator is fine-tuned after each task to handle distribution shift. Next, we ask the generator to draw random samples, using the attributes of the *unseen* classes, creating counterfeit features that we call *ghost features*. The strategy is agnostic to the generator model as long as it can be conditioned by class attributes. At present, as detailed in implementation details, we choose a Generative Moment Matching Network [30]: a shallow multi-layer perceptron conditioned by class attributes and a noise vector trained to minimize the Maximum Mean Discrepancy [17, 18]. The computational overhead of training a generator amounts to only 30 minutes over a 5-hour experiment on a Titan Xp GPU with 12 Go of VRAM because this model is trained on the low-dimensional features and not on the images. More details about the generator can be found in section 4.2.

Complete classifier. Remind that the parameters $\{\boldsymbol{\theta}_c, \forall c \in C^{1:t}\}$ on the representation-based classifier (Eq.1) may be interpreted as proxies for the classes $C^{1:t}$. The base model for task t will, thus, learn $|C^{1:t}|$ such proxies, one for each of the seen classes. To extend the model for the unseen future classes, the complete classifier will learn $|C^{1:t}| + |C^{t+1:T}|$ proxies, which changes Eq.2 to:

$$\mathcal{L}^{\text{nca-ghost}}_{\Theta_f, \Theta_{clf}} = \left[-\log \frac{\exp\left(\hat{\mathbf{y}}_y - \delta\right)}{\sum_{\substack{c \neq y \\ c \in C^{1:t}}} \exp \hat{\mathbf{y}}_c + \sum_{\substack{c \neq y \\ c \in C^{t+1:T}}} \exp \hat{\mathbf{y}}_c} \right]_+ . \tag{4}$$

This classification loss maximizes the similarity $\hat{\mathbf{y}}_y$ (or, conversely, minimizes the distance) between sample feature $\mathbf{h}_y$ and correct class proxy $\boldsymbol{\theta}_y$ in the numerator. In the denominator, the loss pushes away all wrong class proxies, from both seen and unseen (future) classes, by minimizing the similarities with $\mathbf{y}_c, \forall c \neq y$.

The participation of future classes in the classification loss has two effects. Most obviously, it allows the model to perform zero-shot-like guesses for those classes during test time. The representation-learning paradigm allows performing both continual and zero-shot learning seamlessly, as we advance through the tasks and future classes become present classes, and then past classes. Less evidently, but vitally important, the learning of proxies for the future classes
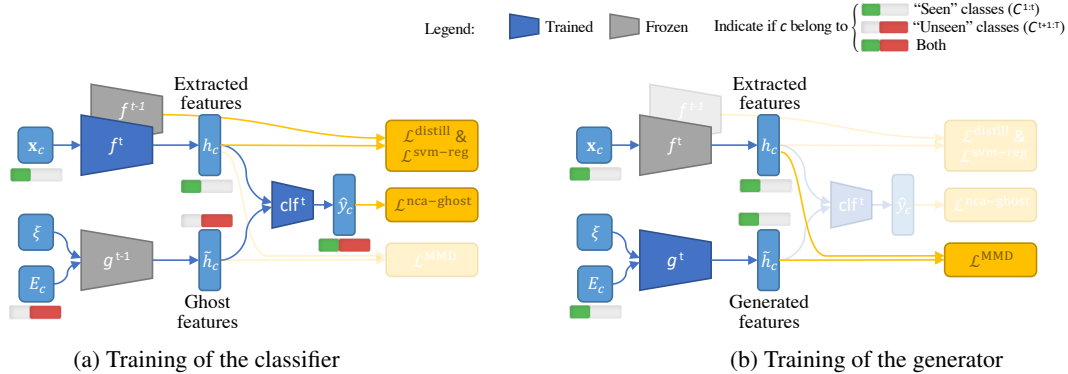
(a) Training of the classifier

(b) Training of the generator

Figure 2: Procedure to train our model applied at each task/step: (a) a complete classifier is learned with seen and unseen features ($\mathcal{L}^{\text{nca-ghost}}$). The feature extractor is protected from catastrophic forgetting ($\mathcal{L}^{\text{distill}}$), and constrained to separate seen classes from unseen/ghosts classes ($\mathcal{L}^{\text{svm-reg}}$). (b) Once a task is done, the generator is fine-tuned on the new latent space ($\mathcal{L}^{\text{MMD}}$) on seen classes. Notice that for the first and last tasks, the classifier does not use the ghost features.

Figure 3: Latent-space regularization establishes margin-based one-unseen-class vs. all-seen-classes linear separations. Those separations are employed to directly condition the feature space, creating space for future unseen classes. In the following task, some unseen class will become seen, and may occupy the feature space with less interference.

makes room in the representation space for those classes, creating effective empty spaces that push away the actual features of the seen classes (due to the repulsive term in the denominator). As we advance through training, future classes become present, their ghost placeholders disappear, and they can neatly fit in the newly vacant region. Such a strategy reduces interference between classes throughout continual learning, and, as we will see in both visual and quantitative experiments, has long-range positive effects.

Naturally, the complete classifier has to be trained with samples from all classes. For seen classes, actual data is available from the training and rehearsal data. For unseen future classes data is not available, so we employ ghost features sampled from the generator. Note that ghost features are produced once per task by the generator and are kept fixed for the task duration.

Latent-space regularization. As explained above, our Ghost classification loss minimizes the intra-class distances and maximizes the inter-class distances. The loss enforces those constraints to all proxies regardless of whether they represent seen classes or not. We further promote an inter-

class separation by optimizing the latent representation of seen classes to avoid overlapping with the representation of Ghosts. That loss constrains the features space directly and does not affect the proxies and the intra-class distances.

We based this regularization loss on SVM [9] for simplicity, but other methods could have similar behavior. To compute that loss, we learn binary one-unseen-class-Vs-all-seen-classes SVM classifiers, one for each unseen class. We employ a linear kernel, since the feature extractor and feature vector dimensionality (512) allows good linear separation, but more complex kernels could be used. Those SVMs define hyperplanes $\mathbf{w}_c$ and biases b_c, $\forall c \in C^{t+1:T}$, separating each unseen region from the mass of seen features $\mathbf{h}^{(i)}$ (Fig.3):

$$\mathcal{L}^{\text{svm-reg}}_{\Theta_f} = \frac{1}{N \times |C^{t+1:T}|} \sum_{i=1}^{N} \sum_{c \in C^{t+1:T}} [\mathbf{w}_c \cdot \mathbf{h}^{(i)} + b_c + \tau]_+ ,$$
(5)

where $\mathbf{h}^{(i)}$ are seen features (classes in $C^{1:t}$), $[\cdot]_+$ the hinge loss, and τ an additional margin (higher values of τ push seen features further away from the ghost regions, in practice, we set $\tau = 1$ to repel beyond the support vectors).

The margin-based regularization of Eq.5 refines the ghost classification loss of Eq.4. While the latter acts over the classifier conditioning the feature space indirectly via the action of the class proxies, the former acts directly over the latent/feature space and the feature extractor backbone that creates it. The computational overhead of training several SVMs (5 minutes), is negligible compared to the total training time (5 hours).

Complete optimization strategy. All modules and losses fit neatly into the goal of learning continuously over seen and unseen classes. We train feature extractor (plus classifier) and generator in alternation. We train the latter
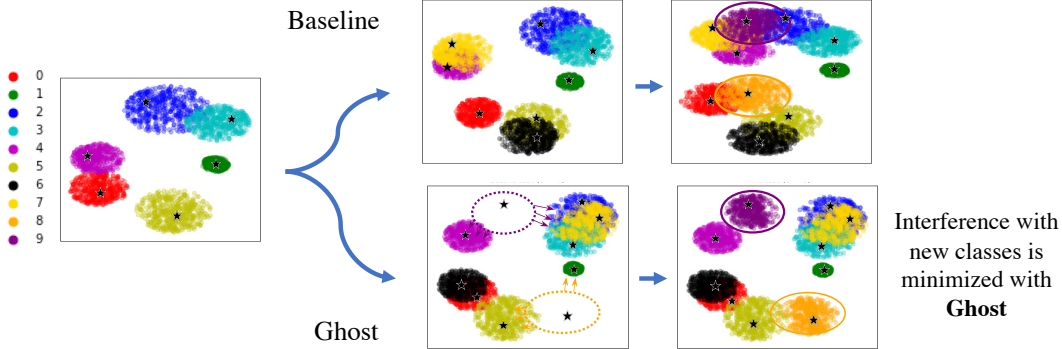
Figure 4: Small-scale PODNet on MNIST with 3 steps (digits '0' to '5'; then '6' and '7'; then '8' and '9') with a features space of only two dimensions. The early incorporation of ghost features/proxies in the second task, denoted by dotted circles in the bottom row, enforces vacant space for those unseen classes. When filled in the third task (last column), there is less interference/overlap with previous classes. Such a strategy improves the final accuracy by 50% (+22 p.p.). A star symbol denotes the learned centroid of each class, ghosts included.

to mimic the features of seen classes, and then ask it to extrapolate to unseen classes (ghost features). Ghost features allow us both to unify seen and unseen classes into a complete classifier ($\mathcal{L}^{\text{nca-ghost}}$), and to enforce early allocation in the feature space for unseen classes ($\mathcal{L}^{\text{svm-reg}}$). The complete loss, in addition to a distillation loss to counter-act catastrophic forgetting ($\mathcal{L}^{\text{distill}}$), is:

$$\mathcal{L} = \mathcal{L}^{\text{nca-ghost}}_{\Theta_f, \Theta_{clf}} + \lambda_1 \mathcal{L}^{\text{distill}}_{\Theta_f} + \lambda_2 \mathcal{L}^{\text{svm-reg}}_{\Theta_f}. \quad (6)$$

The Figure 2 showcases the complete optimization strategy.

4. Experiments

4.1. Pictorial experiments

Before running our full-scale experiments, we will perform a set of experiments with a model that keeps the main components from the proposed model in a simplified form that will allow us to link quantitative performances to intuitive visual plots of the feature spaces. For the experiments in this section we employ the MNIST [28] dataset, with an initial task of 6 classes (digits 0 to 5), then two more tasks of two classes each; the feature extractor comprises two convolutional layers followed by a fully connected layer outputting a feature vector of only two dimensions — a purposeful choice to allow easy visualization of the feature space. Because MNIST classes have no attributes, we cannot apply zero-shot learning directly. Instead, we employ the features of actual images from the future classes instead of samples from the generator — which corresponds, in some ways, to having a perfectly calibrated generator. Those features are extracted once per task, and the feature extractor is never trained on unseen classes images. As we will see in Table 4, integrating the generator in our full-scale model outperforms using actual extracted features, so

that necessary substitution does not exaggerate the abilities of this small-scale model. The losses used to train the small and the full-scale models are the same, but the SVM-based regularization was not employed since it made little sense for a 2D latent space.

The 2D feature space allows us to directly visualize the evolution of the feature space as the tasks progress, without the need for dimensionality reduction techniques that complicate the analysis (e.g. t-SNE [43]). Fig.4 may be interpreted upfront: as the three tasks progress left-to-right, we see the evolution of the feature space on the base model (PODNet, on the top branch) and on the proposed model with ghost features (bottom branch). The base model presents strong overlap between the initial classes, and the later added 8 (orange) and 9 (dark purple). That comes partially from shape similarities ('8' is similar to '0' and '5'), partially from continual learning, and results in severe forgetting of old classes in favor of new ones. The proposed model organizes better the feature space, which is particularly visible between the second and third steps, where the early allocation of ghost zones for the future classes (displayed as empty black circles) is prominent. That better arrangement of the feature space increases the final accuracy from 44 to 66%, a 22 p.p. improvement. The small latent space of 2 dimensions with a continual learning paradigm explains the low performance for both models; a model that learns on all classes in one step (i.e. not in a continual setting) reaches only 69% of final accuracy. That said, our model shows a clear improvement both qualitatively in Fig.4 and quantitatively (+50%).

4.2. Main experiments

Datasets & Protocols. We perform our experiments on two datasets: AwA2 [46], with 50 animals categories, each

| | AwA2 | |
| | 25 classes + 5 × 5 classes | |
	PODNet	UCIR
Base Continual Learner	62.92 ± 0.12	62.66 ± 0.17
+ $\mathcal{L}^{\text{nca-ghost}}$	68.31 ± 0.36	64.96 ± 0.12
+ $\mathcal{L}^{\text{nca-ghost}}$ + $\mathcal{L}^{\text{svm-reg}}$	**68.46 ± 0.47**	**65.37 ± 0.35**
	aP&Y	
	16 classes + 8 × 2 classes	
	PODNet	UCIR
Base Continual Learner	58.64 ± 0.66	45.96 ± 0.22
+ $\mathcal{L}^{\text{nca-ghost}}$	62.08 ± 0.25	59.67 ± 0.84
+ $\mathcal{L}^{\text{nca-ghost}}$ + $\mathcal{L}^{\text{svm-reg}}$	**62.73 ± 0.60**	**60.12 ± 0.74**

Table 1: Continual Accuracy on AwA2 and aP&Y for POD-Net and UCIR. We ran each experiment thrice.

| | AwA2 | |
| | 25 classes + 5 × 5 classes | |
	PODNet	UCIR
Base Continual Learner	77.63 ± 0.06	76.36 ± 0.28
+ $\mathcal{L}^{\text{nca-ghost}}$	78.70 ± 0.46	**77.32 ± 0.40**
+ $\mathcal{L}^{\text{nca-ghost}}$ + $\mathcal{L}^{\text{svm-reg}}$	**79.08 ± 0.53**	77.27 ± 0.35
	aP&Y	
	16 classes + 8 × 2 classes	
	PODNet	UCIR
Base Continual Learner	57.80 ± 0.97	45.40 ± 0.17
+ $\mathcal{L}^{\text{nca-ghost}}$	62.47 ± 0.40	54.13 ± 0.90
+ $\mathcal{L}^{\text{nca-ghost}}$ + $\mathcal{L}^{\text{svm-reg}}$	**63.30 ± 0.98**	**56.07 ± 1.29**

Table 2: Final Accuracy on AwA2 and aP&Y for PODNet and UCIR. We ran each experiment thrice.

with 85 attributes; and AP&Y [12], with 32 everyday object classes, each with 64 attributes. We employ two experimental protocols: one typical for continual learning, following [21, 11], starting the first task with half the classes (i.e., 25 for AwA2, and 16 for aP&Y), then adding the remaining classes in evenly-sized tasks (5 tasks of 5 classes for AwA2, and 8 tasks of 2 classes for aP&Y); another inspired from zero-shot learning, following [46], starting with a standard selection of classes for each of the datasets (40 for AwA2, and 20 for aP&Y), and adding the remaining classes in small increments (5 tasks of 2 classes for AwA2, and 6 tasks of 2 classes for aP&Y). For both settings, the seen classes are trained with supervision, while the remaining unseen classes are tackled with zeroshot-inspired methods. Our evaluation protocol is akin to the challenging and realist Generalized Zero-shot Learning [40, 7] protocol — with no information on whether a sample is from a seen or unseen class — but harsher, since classes are seen gradually, and training samples for past classes data are limited by rehearsal memory.

Base Continual Learner. We evaluate our contributions on top of two different representation-learning based Continual Leaners, both based on ResNet18 [20] feature extractor backbones (with feature vector size of 512) and cosine classifiers. They differ on the distillation loss $\mathcal{L}_{\text{distil}}$ employed, the first model (PODNet) using Douillard et al.'s distillation [11] constraining the statistics of the intermediate features after each residual block, and the second (UCIR) using Hou et al.'s distillation [21] enforcing a cosine constraint on the final flat latent space. The former performs better than the latter, but both are improved by the innovations proposed in this work. The cosine classifier has a single proxy/representative per class but could easily be generalized to multiple proxies. Both base Continual Learners are competitive, with PODNet being the most recent State-of-the-Art from ECCV 2020. We show in this paper how incorporating predictions of the future further improve them.

Generator. Following the work of Bucher et al. [5] for zero-shot learning, our generator is a Generative Moment Matching Network (GMMN) [30] $g^t(\xi, \mathbf{E}_c)$, which takes as inputs a Gaussian noise vector ξ and a class attributes vector $\mathbf{E}_c$, and outputs a sample from the estimated distribution of features for a class with the given attributes. In our experiments on AwA2 and AP&Y, the class attributes vectors are the average of the attributes for the training samples in the class. For each task t, the feature extractor f^t and the generator g^t are trained to minimize the Maximum Mean Discrepancy (MMD) [17, 18] between the actual features of seen classes $\mathbf{h}_c = f^t(\mathbf{x}_c), \forall c \in C^{1:t}$ and their distribution on the generator $\hat{\mathbf{h}}_c = g^t(\xi, \mathbf{E}_c), \forall c \in C^{1:t}$.

Implementation Details For all datasets and settings, we set the classification margin $\delta = 0.6$, and the SVM latent-space regularization additional margin $\tau = 1$. We train the feature-extractor-and-classifier pipeline for 90 epochs with an SGD optimizer, learning rate of 0.1, cosine scheduling, and weight decay of 10^{-4}. We train the generator for 1200 epochs, with an Adam optimizer and a learning rate of 10^{-5}. Finally, following [21, 11], we fine-tune the classifier for 60 epochs (with the feature extractor frozen and a small learning rate of 10^{-4}) at the end of every task (except the last one). We found useful to balance the bias towards the seen classes against the unseen classes. With the POD distillation [11], we set $\lambda_1 = 3$ for AwA2, and $\lambda_1 = 15$ for aP&Y; with the Less-Forget distillation [21], we set $\lambda_1 = 4$ for both datasets. We always set $\lambda_2 = 10^{-3}$, moreover we apply it on L2-normalized features. Finally, we do not reinitialize the models between tasks: f^t results from training f^{t-1} on task t, etc. On the rehearsal memory limitation, we follow the strict setting of Hou et al. [21], keeping only

| | AwA2 | | aP&Y | |
| | 40 classes + 5 × 2 classes | | 20 classes + 6 × 2 classes | |
	Continual	Final	Continual	Final
PODNet	82.84 ± 0.10	84.70 ± 0.10	67.57 ± 0.41	65.23 ± 0.50
+ $\mathcal{L}^{\text{nca-ghost}}$	84.99 ± 0.17	86.57 ± 0.49	68.80 ± 0.98	67.93 ± 1.24
+ $\mathcal{L}^{\text{nca-ghost}}$ + $\mathcal{L}^{\text{svm-reg}}$	84.47 ± 0.15	85.73 ± 0.40	69.02 ± 0.46	67.97 ± 0.60

Table 3: Further experiments where the initial task size correspond to standard zero-shot seen classes [46]. We report Continual and Final Accuracies for PODNet on AwA2 and aP&Y. We ran each experiment thrice.

| | AwA2 | | aP&Y | |
| | 25 classes + 5 × 5 classes | | 16 classes + 8 × 2 classes | |
	Continual	Final	Continual	Final
Our model	68.46 ± 0.47	79.08 ± 0.53	62.73 ± 0.60	63.30 ± 0.98
with real features of the future classes	67.65 ± 0.50	78.83 ± 0.31	61.88 ± 0.52	61.70 ± 0.26
Partial oracle	72.94 ± 0.25	84.60 ± 0.28	63.81 ± 0.29	68.03 ± 1.42
Full oracle	—	95.40 ± 0.02	—	97.40 ± 0.30

Table 4: Comparison of generated ghost features vs. actual features extracted from future classes' samples with PODNet on AwA2 and aP&Y.

$s = 20$ training images per past class. The code will be released publicly upon paper acceptance.

Continual Accuracy. For continual learning, it is usual to take into account the model's performance as it evolves. We adapt the traditional average incremental accuracy [38] to take into account *all* classes, including the future ones, and call that metric *continual accuracy*: the average of accuracy over all seen classes after each task. The results appear in Table 1, which shows, for the datasets and protocols explained in the top of this section, the performance for our two base models [21, 11], and the improvements on those base models as we implement our proposed model, with and without the SVM latent-space regularization refinement. The ability to guess on future class brings large improvements on both datasets, for both competittive base models. The SVM-based regularization refinement also improves the results, by up to 0.65 p.p.

Final Accuracy. Once we reach the final task, the proposed model ability to guess future classes provides no advantage due to all classes being now seen. Still, as shown in Table 2 — where the metric is simply the accuracy at the final task of each run — the proposed method outperforms the baselines, due to a better organization of the feature space. Although the numerical advantages in this table are smaller than in the previous one, these results are consequential, showing that the ability of the proposed model of incorporating knowledge about the classes is useful beyond the zero-shot scenario. Again, the SVM-regularization refinement helps by up to 1.94 p.p. Note that the oracle model, which is trained on the whole dataset in a single task, has an

accuracy of 95.40% on AwA2. Thus, state-of-the-arts methods are still far from this upper bound which highlights the difficulty of the considered datasets.

Model Evolution. To showcase how the models evolve, plots contrasting the proposed methods with each base model (PODNet and UCIR) task by task appear in Fig.5. The plots show how, on early tasks, the main advantage of the proposed model is its ability to guess on the future classes, while on the final task no future classes remain, but the proposed model still keeps an advantage.

Zero-shot-like Initial Task Setting. This set of experiments (Table 3) is intended for comparison with zero-shot learning benchmarks [46], which always use the same split of seen/unseen classes for a given dataset. Our first task in the continual learning contains the classes defined in zero-shot benchmark as seen, and we learn next, in small increment, the remaining classes, i.e., those defined in the zero-shot benchmark as unseen. Because the initial task is larger than previously, fewer future classes remain, and the base models have better performance. Still, the proposed method improves both base methods in both datasets significantly. The setting proposed is different than the — markedly less challenging — setting appearing in Kankuekul et al.[23] and Xue et al.[48], where the set of unseen classes is fixed, and only more seen classes are added incrementally, without any sample limitations given by rehearsal memory.

Generator Validation. The generator approximates the feature extractor for the unseen future classes. To validate its effectiveness, we replace the generated features by the actual features from the future images. This form of "cheat-

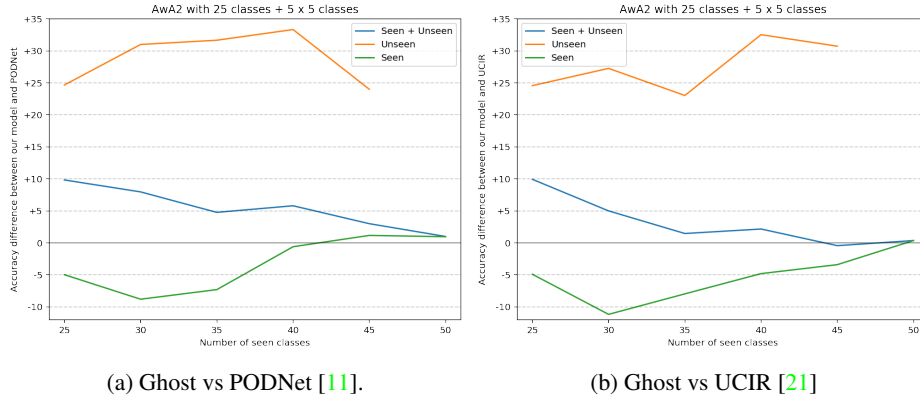(a) Ghost vs PODNet [11].　　　　　(b) Ghost vs UCIR [21]

Figure 5: Ghost model vs base models on AwA2, difference of accuracy over all classes, only seen classes, and only unseen classes.

ing", of course, is not possible in *actual* real-world scenarios, but serves as a metric. Table 4 shows the comparison, with the surprising result that generated features performed better than the actual features from samples (respectively first and second row). Note that the latter are extracted once per task. The lower performance of using real features is explained because the features extrator was never adapted for the unseen classes distribution which acts as an Out-of-Distribution [2]. The "oracle" experiments in the third and fourth row in Table 4 establish an upper bound for what we could achieve by "cheating" around the experimental protocol restrictions. The partial oracle from third row is the same model as the second row, fine-tuning the feature extractor with samples coming from the future. The full oracle of the fourth line uses all images from all classes unrestrictedly in a single task. Despite the partial oracle had full access to real future data, we stress that our model's performance with generated future data is close to this upper bound.

5. Conclusion

In this work, we introduced prescient continual learning, a generalization of the continual learning setting, where the model trains on a sequence of tasks, each introducing new classes, but has access to prior information about the classes. Although we give the model awareness of future classes, we also test it on all classes: past, present, and future, resulting in a more challenging setting. While this setting enables Zeroshot capabilities, it differs from it by opening new challenges on how to integrate weak information from the future into a continuously growing set of classes.We proposed ghost model, first of its kind, which uses the paradigm of representation learning to incorporate capabilities of zero-shot learning into the continual learning model in a seamless way. We refined that model with a novel SVM-based regularization loss acting over the fea-

ture space to reinforce exclusion zones, reserved for future classes. Finally, we established, in extensive quantitative and qualitative experiments, the advantage of the proposed model over two base models.

References

[1] inaturalist. https://www.inaturalist.org. Accessed: 2020.

[2] Inference with the universum. In *International Conference on Machine Learning (ICML)*, 2006.

[3] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2018.

[4] Rahaf Aljundi and Tinne Tuytelaars. Selfless sequential learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[5] Maxime Bucher, Stephane Herbin, and Frederic Jurie. Generating visual representations for zero-shot classification. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2017.

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

[7] Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2016.

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with

a-gem. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 2015.

[10] Guy Davidson and Michael C. Mozer. Sequential mastery of multiple visual tasks: Networks naturally learn to learn and forget to forget. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2020.

[12] A. Farhadi, I. Endres, D. Hoiem, and D.A. Forsyth. Describing objects by their attributes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[13] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. *arXiv preprint library*, 2017.

[14] Robert French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 1999.

[15] Jacob Goldberger, Geoffrey E Hinton, Sam T. Roweis, and Ruslan R Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2005.

[16] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *Advances in Neural Information Processing Systems (NeurIPS), Neuro AI Workshop*, 2019.

[17] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2007.

[18] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 2012a.

[19] Kai Han, Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Andrea Vedaldi, and Andrew Zisserman. Automatically discovering and learning new visual categories with ranking statistics. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] 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.

[22] Steven C.Y. Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[23] Pichai Kankuekul, Aram Kawewong, Sirinart Tangruamsub, and Osamu Hasegawa. Online incremental attribute-base zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[24] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[25] James Kirkpatrick, Razvan Pascanu, Neil 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. *Proceedings of the National Academy of Sciences*, 2017.

[26] Vinay Kumar Verma, Gundeep Arora, Ashish Mishra, and Piyush Rai. Generalized zero-shot learning via synthesized examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[27] C. H. Lampert, H. Nickisch, and S. Hermeling. Learning to detect unseen object classes by between-class attribute transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[28] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs*, 2010.

[29] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[30] Yujia Li, Kevin Swersky, and Richard Zemel. Generative moment matching networks. In *International Conference on Machine Learning (ICML)*, 2015.

[31] Z. Li and D. Hoiem. Learning without forgetting. *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2016.

[32] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[33] Chunjie Luo, Jianfeng Zhan, Xiaohe Xue, Lei Wang, Rui Ren, and Qiang Yang. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *International Conference on Artificial Neural Networks*, 2018.

[34] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint library*, 2013.

[35] Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[36] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[37] Federico Pernici, Matteo Bruni, Claudio Baecchi, Francesco Turchini, and Alberto Del Bimbo. Class-incremental learning with pre-allocated fixed classifiers. In *International Conference on Pattern Recognition*, 2021.

[38] 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.

[39] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 1995.

[40] W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult. Toward open set recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2013.

[41] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[42] Ivan Skorokhodov and Mohamed Elhoseiny. Class normalization for (continual)? generalized zero-shot learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[43] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 2008.

[44] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: Similarity control and knowledge transfer. In *Journal of Machine Learning Research*, 2015.

[45] Kai Wang, Luis Herranz, Anjan Dutta, and Joost van de Weijer. Bookworm continual learning: beyond zero-shot learning and continual learning. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV) Workshop*, 2020.

[46] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[47] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[48] Nan Xue, Yi Wang, Xin Fan, and Maomao Min1. Incremental zero-shot learning based on attributes for image classification. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2017.

[49] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.