INTERPRETABLE DIFFERENTIALLY PRIVATE CONTINUAL LEARNING WITH PROTOTYPE-BASED CLASSIFIERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning (CL) under differential privacy (DP) involves two competing objectives: a model must learn from new tasks while avoiding catastrophic forgetting, and the training pipeline must preserve privacy. Requiring the model's decisions to be interpretable introduces an additional competing objective. These trade-offs become particularly pronounced when no sensitive state, such as memory buffers, can be retained across tasks. We introduce a DP prototype-based CL framework built on pretrained backbones, where each class is represented by a single DP prototype and predictions are made by nearest-prototype classification in a shared embedding space. This eliminates the need to store samples while making the decision rule inherently interpretable. We control the CL trade-offs through novel DP-friendly loss functions adapted from prior CL work. On the Split-CIFAR-100 and Split-ImageNet-R benchmarks, our method achieves similar or higher accuracy than a state-of-the-art interpretable DP CL baseline while remaining competitive with a stronger but non-interpretable DP CL baseline across challenging CL settings. Finally, we introduce two complementary methods for interpreting the learned DP prototypes: one based on publicly available pretraining data and another using synthetic samples generated by a generative model.

1 INTRODUCTION

The objective of continual learning (CL) (McCloskey & Cohen, 1989; Parisi et al., 2019; De Lange et al., 2021) is to learn from a sequence of new tasks, while retaining utility on earlier tasks. This becomes more challenging under differential privacy (DP) (Dwork et al., 2006b). The model must preserve privacy of the sensitive task datasets and avoid catastrophic forgetting. In addition, derived sensitive information must not be retained across tasks. These competing objectives can substantially limit model utility. The current state-of-the-art in DP CL (Tobaben et al., 2026) uses pretrained backbones to mitigate this utility loss.

Interpretability (Doshi-Velez & Kim, 2017; Rudin, 2019) introduces an additional challenge for DP CL, as the model must also have a decision rule that can be directly understood. Prototype-based methods (Chen et al., 2019; De Lange & Tuytelaars, 2021; Aghasanli et al., 2025) provide a natural way to retain information about previously learned classes using vectors in the embedding space of the model. These vectors are referred to as *prototypes*. A new input can then be classified according to the class associated with its nearest prototype. This contrasts with standard max-logit classifiers, where the predicted class corresponds to the largest output logit.

The method of Aghasanli et al. (2025) uses a novel combination of losses for learning and preserving prototype structure. However, these losses do not directly apply to the DP CL problem, and need to be adapted. Inspired by this idea, we introduce an interpretable DP CL framework built on pretrained backbones, where a DP prototype is released for each class in the sensitive task dataset, and then updated as necessary across future tasks. Because predictions are made in terms of similarity to released DP prototypes, the resulting classifier provides an interpretable decision rule.

Contributions.

1. Inspired by the work of Aghasanli et al. (2025), we introduce an interpretable DP prototype-based CL framework; see Sections 4.1 to 4.3.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2. We theoretically analyse how DP perturbation affects prototype stability; see Section 4.4. We show that DP perturbation reduces classification stability by requiring larger inter-class separation, while stability improves with more data, weaker privacy, lower dimensionality, and fewer labels.
3. We compare our method against two state-of-the-art DP CL methods of Tobaben et al. (2026); see Section 5. Our method outperforms the interpretable cosine classifier while remaining competitive with the non-interpretable PEFT ensemble.
4. We present an interpretability interface for the released DP prototypes in image classification using proxy images, retrieved from a public dataset or synthesised by a generative model; see Section 5.4.

2 Related work

Interpretable CL. Interpretable CL has been explored through both prototype-based classifiers and models that explain predictions using human-understandable concepts. ICICLE (Rymarczyk et al., 2023) introduces interpretability regularization to preserve previously learned prototypical parts and reduce changes in explanations across tasks. CIP-Net (Valerio et al., 2025) also adopts an exemplar-free prototype-based architecture to combine interpretable predictions with knowledge retention. Among concept-based approaches, IN2 (Yang et al., 2024) preserves learned concepts through freezing and classifier regularization, while CONCIL (Lai et al., 2024) uses recursive updates to incrementally learn concepts and classes. Language-guided concept bottleneck models (Yu et al., 2025) combine semantic alignment with prototype augmentation to mitigate forgetting. However, these methods do not establish DP guarantees for their learned representations or explanations.

DP CL. Prior work on DP continual learning includes replay-based methods (Desai et al., 2021) and architecture-specific approaches (Lai et al., 2022; Hassanpour et al., 2022). These works primarily focus on protecting model updates, but do not consider the privacy risks from side-channels or storing samples between tasks. Side-channels (Debenedetti et al., 2024) are auxiliary components or operations outside the main model, and they can invalidate privacy guarantees. Tobaben et al. (2026) showed that data-dependent output label spaces can leak private information, since the presence of a label may itself distinguish adjacent datasets.Tobaben et al. (2026) further establish two strong baselines that are built on pretrained backbones. First, the PEFT ensemble uses DP-SGD to fine-tune the backbone through FiLM adapters (Tobaben et al., 2023) and a zero-initialized linear classification head independently for each task, starting from the original pretrained backbone. The released models from all tasks are combined into an ensemble for prediction. The logits from the linear head are not interpretable beyond representing relative label confidence. Second, the cosine classifier sums normalized training data embeddings within each class to form prototypes. Each prototype is perturbed for privacy, and can then be used to classify new examples. Although the cosine classifier is an interpretable model, its embeddings are fixed with no further adaptation. To our knowledge, there is no existing work on interpretability in DP CL.

3 Background

3.1 Class-incremental continual learning

In the class-incremental setting (van de Ven et al., 2022), we observe a sequence of sensitive datasets $\mathcal{D}_t = \{(\boldsymbol{x}_t^{(k)}, y_t^{(k)})\}_{k=1}^{N_t}$, indexed by $t \in \{1, \dots, T\}$, where $\boldsymbol{x}_t^{(k)} \in \mathcal{X}$ are the features and $\mathcal{Y}_t^{\text{data}} := \{y_t^{(k)}\}_{k=1}^{N_t}$ is the set of labels present in task t. Following Tobaben et al. (2026), we denote $\mathcal{O}_t$ as the released set of task-specific labels and define $\mathcal{O}_{1:t} := \bigcup_{s=1}^{t} \mathcal{O}_s$. This notation distinguishes $\mathcal{O}_t$ from $\mathcal{Y}_t^{\text{data}}$ to enable protection against classifier label space side-channel attacks. The objective is to learn a classifier $f_t : \mathcal{X} \to \mathcal{O}_{1:t}$ that adapts to the new classes while retaining performance on previous classes without access to the task labels. Each dataset $\mathcal{D}_t$ is discarded after training.

3.2 Differential privacy

DP provides mathematically provable privacy guarantees for releasing statistics or training models on sensitive datasets. This is achieved by ensuring that the probability of an outcome of an algorithm stays similar whether or not any single individual's record is included in the dataset. Throughout the paper we use add/remove adjacency, i.e. $\mathcal{D} \simeq \mathcal{D}'$ if $\mathcal{D}' = \mathcal{D} \cup \{(\mathbf{x}, y)\}$ or $\mathcal{D}' = \mathcal{D} \setminus \{(\mathbf{x}, y)\}$.

Definition 1 (DP; Dwork et al. 2006b)**.** *Given an adjacency relation $\simeq$ between datasets, a randomized mechanism $\mathcal{A}$ is (ϵ, δ)-DP if for any adjacent datasets $\mathcal{D} \simeq \mathcal{D}'$ and any measurable event S in the output range of $\mathcal{A}$,* $\Pr[\mathcal{A}(\mathcal{D}) \in S] \leq e^{\epsilon} \Pr[\mathcal{A}(\mathcal{D}') \in S] + \delta$.

Gaussian mechanism. Let f be a vector-valued query. Denote the ℓ_2-sensitivity as $\Delta_f = \max_{\mathcal{D} \simeq \mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2$. The Gaussian mechanism (Dwork et al., 2006a) is defined by $\tilde{f}(\mathcal{D}) = f(\mathcal{D}) + \xi$, where $\xi \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{DP}}^2 \Delta_f^2 \boldsymbol{I})$, and is (ϵ, δ)-DP for an appropriate choice of σ_{DP}, which depends on ϵ and δ (Balle & Wang, 2018).

We train with DP-SGD (Rajkumar & Agarwal, 2012; Song et al., 2013; Abadi et al., 2016) which clips per-sample gradients and then applies the Gaussian mechanism to their aggregated sum at each iteration, thereby producing DP model updates. Cumulative privacy loss is tracked with an accountant (Koskela et al., 2020). Following Tobaben et al. (2026), we assume that each individual appears in at most one task, and thus applying an (ϵ, δ)-DP mechanism for every task guarantees that the final classifier is (ϵ, δ)-DP, which is known as parallel composition (McSherry, 2010).

DP output label spaces. To protect against catastrophic privacy leakage, we use the DP label release method introduced by Tobaben et al. (2026) for determining the classifier output label space. Let $\mathcal{D}_{t,c} = \{(\boldsymbol{x}, y) \in \mathcal{D}_t : y = c\}$ be the subset of task-t examples from class c, then

$$\mathcal{O}_t^{\text{learned}} = \left\{ c \in \mathcal{Y}_t^{\text{data}} : |\mathcal{D}_{t,c}| + Z_c > k \right\}, \tag{1}$$

where Z_c is k-truncated symmetric geometric noise calibrated to $(\epsilon_{\text{label}}, \delta_{\text{label}})$ (Desfontaines et al., 2022). We set $\mathcal{O}_t := \mathcal{O}_t^{\text{learned}}$, for each task t.

4 Method

4.1 Model, prototypes, and prediction

Our model consists of a pretrained backbone $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}^{d_h}$ followed by a trainable linear projection layer $g_{\boldsymbol{\phi}} : \mathbb{R}^{d_h} \to \mathbb{R}^d$, so that for each input $\boldsymbol{x} \in \mathcal{X}$, we have an embedding $\boldsymbol{z} = g_{\boldsymbol{\phi}}(h_{\boldsymbol{\theta}}(\boldsymbol{x})) \in \mathbb{R}^d$. Following Aghasanli et al. (2025), to have better control on the CL problem, we utilize the backbone feature space. Denote $\boldsymbol{z}^{\text{pre}}(\boldsymbol{x}) = h_{\boldsymbol{\theta}}(\boldsymbol{x})$, and $\bar{\boldsymbol{z}}^{\text{pre}}(\boldsymbol{x}) = h_{\boldsymbol{\theta}}(\boldsymbol{x}) / \|h_{\boldsymbol{\theta}}(\boldsymbol{x})\|_2$. Let $\tilde{\boldsymbol{p}}_c^{\text{pre}(0)} = \mathbf{0}$, and for each class $c \in \mathcal{O}_{1:t}$,

$$\tilde{\boldsymbol{p}}_c^{\text{pre}(t)} = \tilde{\boldsymbol{p}}_c^{\text{pre}(t-1)} + \sum_{(\boldsymbol{x}, y) \in \mathcal{D}_{t,c}} \bar{\boldsymbol{z}}^{\text{pre}}(\boldsymbol{x}) + \xi_c^{(t)}, \qquad \xi_c^{(t)} \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{DP}}^2 \boldsymbol{I}). \tag{2}$$

Note that the prototypes are updated according to Equation (2) only if the class $c \in \mathcal{O}_t$. The final released prototype is $\tilde{\boldsymbol{p}}_c^{(t)} = g_{\boldsymbol{\phi}}(\tilde{\boldsymbol{p}}_c^{\text{pre}(t)}) / \|g_{\boldsymbol{\phi}}(\tilde{\boldsymbol{p}}_c^{\text{pre}(t)})\|_2$, which has the same privacy guarantees as $\tilde{\boldsymbol{p}}_c^{\text{pre}(t)}$ due to post-processing. We use sums rather than means because under the add-remove adjacency, the class count $|\mathcal{D}_{t,c}|$ is itself sensitive. If a class's examples are distributed across multiple tasks, the sum makes the prototype invariant to how those examples are distributed among the tasks. At test time, the predicted label is the class with the highest cosine similarity:

$$\hat{y} = \arg \max_{c \in \mathcal{O}_{1:t}} \frac{\boldsymbol{z}^\top \tilde{\boldsymbol{p}}_c^{(t)}}{\|\boldsymbol{z}\|_2 \|\tilde{\boldsymbol{p}}_c^{(t)}\|_2}. \tag{3}$$

4.2 DP-friendly losses for CL

Our loss function for adapting $\boldsymbol{\phi}$ is a weighted sum of three DP-friendly losses introduced in Equations (5), (6) and (8), where $\lambda_{\text{pcl}}, \lambda_{\text{push}}, \lambda_{\text{preserve}} > 0$:

$$\mathcal{L} = \lambda_{\text{pcl}} \mathcal{L}_{\text{pcl}} + \lambda_{\text{push}} \mathcal{L}_{\text{push}} + \lambda_{\text{preserve}} \mathcal{L}_{\text{preserve}}. \tag{4}$$

Our key design constraint under DP is to retain the prototype-based geometry of the classes (Aghasanli et al., 2025) while avoiding sample-to-sample cross terms, as these have high sensitivity: if an individual's data is changed, it affects multiple cross terms. Moreover, we avoid using any sensitive quantity such as statistics derived from the sensitive dataset. Further details about the original non-DP versions of these losses can be found in Section A. Let $B = \mathbb{E}[|\mathcal{B}|]$ denote the expected minibatch size, for the DP-SGD objectives below, we normalize minibatch sums by B.

Prototype-based contrastive loss. This loss aligns samples with their corresponding DP class prototypes. Let $\boldsymbol{z}_i = g_{\boldsymbol{\phi}}(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$, $\bar{\boldsymbol{z}}_i = \boldsymbol{z}_i / \|\boldsymbol{z}_i\|_2$, and let $\tau > 0$ be a temperature. If same-class features concentrate around a representative in the non-DP supervised contrastive loss in Equation (13), their mean can be approximated by the corresponding class prototype: $\frac{1}{|\mathcal{P}(i)|}\sum_{q\in\mathcal{P}(i)} \bar{\boldsymbol{z}}_q \approx \tilde{\boldsymbol{p}}_{y_i}^{(t)}$. Replacing $\bar{\boldsymbol{z}}_q$ and $\bar{\boldsymbol{z}}_a$ in Equation (13) by their corresponding released class prototypes yields the DP-friendly version of the loss:

$$\mathcal{L}_{\text{pcl}} = -\frac{1}{B}\sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{B}} \log \frac{\exp(\bar{\boldsymbol{z}}_i^\top \tilde{\boldsymbol{p}}_{y_i}^{(t)}/\tau)}{\sum_{c\in\mathcal{O}_t,\, c\neq y_i}\exp(\bar{\boldsymbol{z}}_i^\top \tilde{\boldsymbol{p}}_c^{(t)}/\tau)}. \tag{5}$$

If the denominator in Equation (5) also included the prototype corresponding to y_i, the resulting classifier would have logits $\ell_c(\boldsymbol{x}) = \frac{1}{\tau}\bar{\boldsymbol{z}}^\top \tilde{\boldsymbol{p}}_c^{(t)}$, i.e., a bias-free linear classifier on the normalized embeddings, with the prototypes as weights. This motivates additional CL-specific losses.

Push-away loss. To reduce confusion with previously released prototypes, we adapt the push-away loss of Aghasanli et al. (2025) in Equation (15) by removing private class statistics, and only use the released DP prototypes. Replacing the sum with a maximum over the released classes, yields

$$\mathcal{L}_{\text{push}} = \frac{1}{B}\sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{B}} \max_{c\in\mathcal{O}_{1:t-1},\, c\neq y_i} \bar{\boldsymbol{z}}_i^\top \tilde{\boldsymbol{p}}_c^{(t-1)}. \tag{6}$$

Preservation loss. To preserve the representation geometry of previously learned classes, we adapt the cluster preservation loss of Aghasanli et al. (2025) in Equation (16). Because previous-task support samples cannot be retained, we replace them with synthetic support points around each released DP pre-projection prototype using fixed radii and data-independent orthonormal directions $\boldsymbol{d}_\ell$. The directions are generated randomly once and then used throughout all the tasks, which yields

$$\mathcal{H}_c^{(t-1)} = \left\{\tilde{\boldsymbol{p}}_c^{\text{pre}(t-1)}\right\} \cup \left\{\tilde{\boldsymbol{p}}_c^{\text{pre}(t-1)} \pm br\boldsymbol{d}_\ell : b\in\{1,2,3\},\ \ell\in\{1,\dots,L\}\right\}, \tag{7}$$

$$\mathcal{L}_{\text{preserve}} = \frac{1}{|\mathcal{O}_{1:t-1}|}\sum_{c\in\mathcal{O}_{1:t-1}} \widehat{\text{MMD}}_\kappa^2\left(g_{\boldsymbol{\phi}^{(t-1)}}(\mathcal{H}_c^{(t-1)}), g_{\boldsymbol{\phi}}(\mathcal{H}_c^{(t-1)})\right), \tag{8}$$

where κ is the RBF kernel with bandwidth $\sigma_{\text{MMD}} = 1$. Because the released prototypes are DP and the directions are data-independent, constructing the supports $\mathcal{H}_c^{(t-1)}$ is post-processing.

4.3 DP TRAINING

We freeze the backbone $h_{\boldsymbol{\theta}}$ and train only the linear projection head $g_{\boldsymbol{\phi}}$ using DP-SGD on the task-specific dataset $\mathcal{D}_t$. As summarised in Algorithm 1, each task involves three private mechanisms: label release, prototype computation, and DP-SGD training. Given the privacy budgets allocated to releasing the labels, $(\epsilon_{\text{label}}, \delta_{\text{label}})$ in Equation (1), and the prototypes, $(\epsilon_{\text{proto}}, \delta_{\text{proto}})$ in Equation (2), we compose their privacy loss distributions (PLDs) (Sommer et al., 2019; Koskela et al., 2020) and calibrate the DP-SGD noise multiplier using binary search to meet the target (ϵ, δ) budget; see Appendix D. The final classifier is (ϵ, δ)-DP by parallel composition (McSherry, 2010).

In our work, we used hyperparameter optimization to enable a fair comparison between methods and to isolate the effect of hyperparameter choices. We do not account for additional privacy loss incurred during hyperparameter tuning, as it does not increase observed MIA vulnerability (Pradhan et al., 2025).

4.4 HOW DP PERTURBATION AFFECTS PROTOTYPE STABILITY

We give a conceptual analysis of how the Gaussian perturbation in Equation (2) affects classification, which we refer to as stability; proofs are in Appendix B. For simplicity, we assume that each class appears in a single task. For the following, assume we are given a privacy budget $(\epsilon_{\text{proto}}, \delta_{\text{proto}})$, and the released set of labels $\mathcal{O}_{1:t}$. For each $c \in \mathcal{O}_{1:t}$, let $\boldsymbol{u}_c^{(t)} \in \mathbb{R}^{d_h}$ be its unperturbed pre-projection sum and $\tilde{\boldsymbol{u}}_c^{(t)} = \boldsymbol{u}_c^{(t)} + \xi_c^{(t)}$, where $\xi_c^{(t)} \sim \mathcal{N}(\boldsymbol{0}, \sigma_{\text{DP}}^2 \boldsymbol{I}_{d_h})$. The sensitivity $\Delta_f = 1$ because $\|\bar{\boldsymbol{z}}^{\text{pre}}(\boldsymbol{x})\|_2 = 1$ in Equation (2). The non-DP and DP prototypes are $\boldsymbol{p}_c^{(t)} = g_{\boldsymbol{\phi}}(\boldsymbol{u}_c^{(t)})/\|g_{\boldsymbol{\phi}}(\boldsymbol{u}_c^{(t)})\|_2$,

and $\tilde{\boldsymbol{p}}_c^{(t)} = g_{\boldsymbol{\phi}}(\tilde{\boldsymbol{u}}_c^{(t)})/\|g_{\boldsymbol{\phi}}(\tilde{\boldsymbol{u}}_c^{(t)})\|_2$. For a unit embedding vector $\bar{\boldsymbol{z}}$ with label y, define

$$\gamma_t = \max_{c \in \mathcal{O}_{1:t}} \|\tilde{\boldsymbol{p}}_c^{(t)} - \boldsymbol{p}_c^{(t)}\|_2, \qquad m_t(\bar{\boldsymbol{z}}) = \bar{\boldsymbol{z}}^\top \boldsymbol{p}_y^{(t)} - \max_{j \in \mathcal{O}_{1:t} \setminus \{y\}} \bar{\boldsymbol{z}}^\top \boldsymbol{p}_j^{(t)}, \tag{9}$$

and let $\tilde{m}_t(\bar{\boldsymbol{z}})$ be the analogous margin.

Theorem 1. *For every unit-norm feature vector $\bar{\boldsymbol{z}}$, $\tilde{m}_t(\bar{\boldsymbol{z}}) \geq m_t(\bar{\boldsymbol{z}}) - 2\gamma_t$. Hence, if $m_t(\bar{\boldsymbol{z}}) > 2\gamma_t$, the DP and non-private cosine classifiers both predict y.*

DP noise can make the correct prototype less similar to the query and a competing prototype more similar to it. If the non-private margin exceeds $2\gamma_t$, the correct class remains the most similar.

Theorem 2. *Let $|\mathcal{O}_{1:t}| \geq 2$ and, for $\alpha \in (0,1)$, set $r_\alpha = \sigma_{\mathrm{DP}} \left(\sqrt{d_h} + \sqrt{2\log(|\mathcal{O}_{1:t}|/\alpha)}\right)$. With probability at least $1-\alpha$, if $\|\boldsymbol{W}_{\boldsymbol{\phi}}\|_{\mathrm{op}} r_\alpha < \|\boldsymbol{W}_{\boldsymbol{\phi}} \boldsymbol{u}_c^{(t)}\|_2$ for every $c \in \mathcal{O}_{1:t}$, then every example satisfying*

$$m_t(\bar{\boldsymbol{z}}) > 4 r_\alpha \|\boldsymbol{W}_{\boldsymbol{\phi}}\|_{\mathrm{op}} \max_{c \in \mathcal{O}_{1:t}} (\|\boldsymbol{W}_{\boldsymbol{\phi}} \boldsymbol{u}_c^{(t)}\|_2)^{-1} \tag{10}$$

is classified as y by both the DP and non-private cosine classifiers.

Thus, looking at r_α defined in Theorem 2 and the right-hand side of the inequality in Equation (10), stability improves with weaker privacy, lower dimensionality, and smaller output label space size; and larger projected class sums (more informative data).

4.5 Visualizing DP prototypes

We consider image classification, where the encoder is a pretrained vision backbone. The DP prototypes $\tilde{\boldsymbol{p}}_c$ live in the embedding space defined by the frozen backbone $h_{\boldsymbol{\theta}}$ and the learned projection $g_{\boldsymbol{\phi}}$, and are not themselves images. To make them interpretable, we propose two complementary methods that turn a prototype into a picture, which we call a *proxy*. We write $\bar{\boldsymbol{z}}(x)$ for the normalized embedding of an image x in this space. Throughout, a proxy is chosen by its similarity to its own prototype alone, as it should be the image closest to the prototype, and this choice requires no knowledge of prototypes yet to be released.

Public Proxy. Following Kangin & Angelov (2024), we represent each prototype by its nearest real image, taken from a publicly available $\mathcal{D}_{\text{pub}}$ dataset. For each class c, we select the public image whose embedding is closest to the prototype, $x_c^{\text{pub}} = \arg\max_{x \in \mathcal{D}_{\text{pub}}} \bar{\boldsymbol{z}}(x)^\top \tilde{\boldsymbol{p}}_c$.

Generated Proxy. Our second method aims to visualize the prototypes using an image-conditioned diffusion model (ICDM), that is trained to generate new images given an embedding of an existing image (Ramesh et al., 2022; Ye et al., 2023). In our setting, the conditioning input is not an image embedding but a class prototype. Denote the ICDM encoder by ψ. Given a public image $x \in \mathcal{D}_{\text{pub}}$, we generally have $\dim(\psi(x)) = d_{ICDM} \neq d = \dim(g_\phi(h_\theta(x)))$. We therefore fit a translator $B : \mathbb{R}^d \to \mathbb{R}^{d_{ICDM}}$ on the public images s.t. $B(\bar{\boldsymbol{z}}(x)) \approx \bar{\boldsymbol{z}}_\psi(x)$ (see Appendix G.1 for details). The diffusion model $\mathcal{G}$ is then conditioned on the translated prototype to draw n candidates with random seeds, $x_c^{(i)} = \mathcal{G}\big(B(\tilde{\boldsymbol{p}}_c)\big)$, where $i = 1, \ldots, n$, and keep the one whose embedding is closest to the prototype in the classifier's own space: $x_c^{\text{gen}} = \arg\max_{i \in \{1,\ldots,n\}} \bar{\boldsymbol{z}}(x_c^{(i)})^\top \tilde{\boldsymbol{p}}_c$.

Gated combination. Although both proxies produce a usable image for most classes, they can fail for different reasons. The public proxy is bounded by the coverage of the public dataset and the generated proxy by how well the prototype decodes. We therefore use the public proxy by default and fall back to the generated proxy when the nearest public image is far from the prototype, that is, when $\bar{\boldsymbol{z}}(x_c^{\text{pub}})^\top \tilde{\boldsymbol{p}}_c < s_{\min}$ for some cut-off value $s_{\min}$.

Proxy fidelity. To assess a proxy $x_c \in \{x_c^{\text{pub}}, x_c^{\text{gen}}\}$, we pass it through the classifier and compare its latent embedding with all released prototypes. The proxy is correct if it is assigned to class c, equivalently if its margin is positive, $m(x_c) = \bar{\boldsymbol{z}}(x_c)^\top \tilde{\boldsymbol{p}}_c - \max_{j \neq c} \bar{\boldsymbol{z}}(x_c)^\top \tilde{\boldsymbol{p}}_j > 0$. In contrast to the selection, this criterion accounts for all other prototypes. We used the proportion of correct proxies as the fidelity measure (more in Appendix E.1).

5 EXPERIMENTS

We begin by outlining the experimental setup and implementation details, then evaluate the proposed method in the disjoint (Section 5.2) and blurry (Section 5.3) class-incremental settings. Finally, we analyse the interpretability of the learned prototypes in Section 5.4.

5.1 EXPERIMENTAL SETUP

Datasets. We use two CL benchmarks: Split-CIFAR-100 with 10 tasks and 10 classes per task (Krizhevsky, 2009) and Split-ImageNet-R with 10 tasks and 20 classes per task (Hendrycks et al., 2021; Wang et al., 2022). For the interpretability interface, we use ImageNet-1k as public data (Russakovsky et al., 2015).

Continual learning settings. We consider both disjoint and blurry class-incremental learning settings. In the disjoint setting, classes are separated across tasks, such that all examples of each class are contained within a single task. In the blurry setting, examples of some classes are distributed across multiple tasks, resulting in less clearly separated task boundaries and a more realistic continual learning scenario.

Privacy setting. We report privacy in terms of (ϵ, δ)-DP with add/remove adjacency. Moreover, we assume that each individual's data appears in at most one task which corresponds to parallel composition across tasks (see Section 3). All methods are evaluated at $\epsilon \in \{1, 8\}$ with $\delta = 10^{-5}$.

Baselines. We evaluate the performance of the proposed method against the following baselines:

- **Cosine classifier** (Tobaben et al., 2026) forms private cumulative per-class feature sums and classifies inputs by cosine similarity to these representations.
- **PEFT ensemble** (Tobaben et al., 2026) fine-tunes a separate task-specific model with DP-SGD, using FiLM adapters, and predicts using the maximum logit across all models learned so far.
- **Naive baseline**: a single pretrained model is sequentially fine-tuned across all tasks with DP-SGD. This serves as a lower bound, as no mechanisms are applied to mitigate catastrophic forgetting.
- **Non-CL Baseline**: a single pretrained model is fine-tuned using DP-SGD on the full dataset without task splits, showing the cost of DP training in the absence of continual learning.

Evaluation metrics. Following Tobaben et al. (2026), we evaluate continual-learning performance using average accuracy (AA) and the forgetting measure (FM) (Chaudhry et al., 2018). AA summarizes classification performance over the tasks learned so far, and FM quantifies the degradation in performance on previously learned tasks. Formal definitions are provided in Appendix E.1.

All experiments are repeated over five independent runs, using different data splits, DP noise, and hyperparameter optimization. We report the median (measured on the final task) across the five runs, together with the minimum and maximum.

Implementation details. Following prior work (Tobaben et al., 2026), we utilize a ViT-Base-16 network (Dosovitskiy et al., 2021) pretrained on the ImageNet21K dataset (Russakovsky et al., 2015). We assume that the pretraining data is public, while the task datasets $\mathcal{D}_t$ are sensitive and thus require protection via DP. For our method, the pretrained backbone is kept frozen, and a trainable linear projection maps the backbone representations from 768 dimensions to a 512-dimensional latent space, as described in Section 4.3.

We optimise the trainable linear projection using DP-Adam (Kingma & Ba, 2015). Hyperparameter optimisation is performed on the first task only, using Optuna (Akiba et al., 2019) with the Gaussian-process sampler (Williams & Rasmussen, 2006) for 50 iterations. The hyperparameter search spaces and optimization protocol are provided in Appendix E.3.

For the interpretability interface, we use the Split-CIFAR-100 model ($\epsilon = 8$) and its latent space for both proxies. The generated proxy uses SDXL (Podell et al., 2024) with IP-Adapter (Ye et al., 2023) as the decoder, which lets it take an image embedding in place of text, and the adapter's image encoder CLIP ViT-H (Radford et al., 2021) is ψ. All three have been pretrained on public image-text data (Schuhmann et al., 2022) and used as released. Between the two spaces, the translator B is a two-layer MLP trained on the public dataset (more details in Appendix G.1). For generation, we draw $n = 4$ candidates per class (see Appendix G.3 for details), and for the gate we set $s_{\min} = 0.73$.

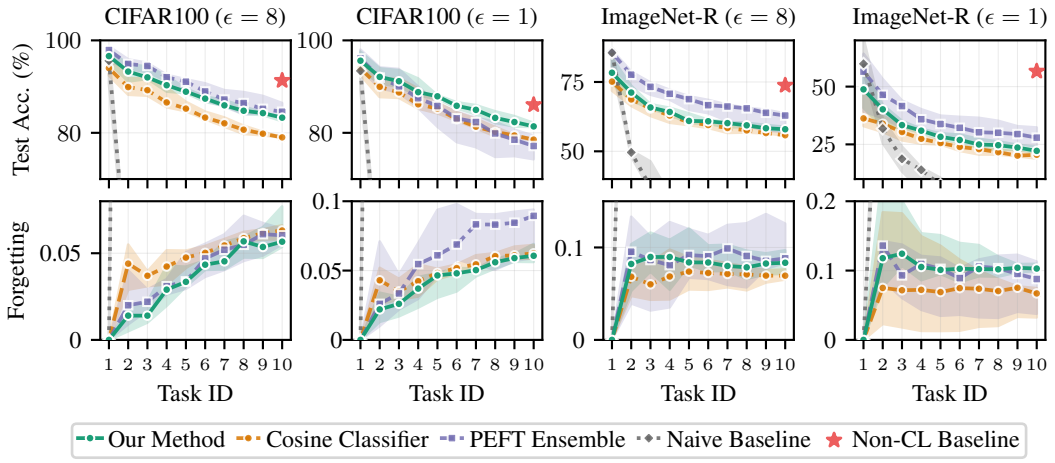

Figure 1: Test accuracy and forgetting measure across the ten tasks in the disjoint setting on Split-CIFAR-100 and Split-ImageNet-R for $\epsilon \in \{1, 8\}$: our method outperforms the cosine classifier and remains competitive with the PEFT ensemble on Split-CIFAR-100. On Split-ImageNet-R, it performs comparably to the cosine classifier, while the PEFT ensemble achieves the highest accuracy. It also generally achieves the lowest forgetting on Split-CIFAR-100 and comparable forgetting on Split-ImageNet-R. We report the median test accuracy and forgetting measure per task. The error bars indicate the minimum and maximum values over five repeats with different data splits, with $\delta = 10^{-5}$. Results for the cosine classifier, Naive, and Non-CL baselines are from the original experimental runs of Tobaben et al. (2026).

5.2 Disjoint class-incremental setting results

We first consider the conventional disjoint class-incremental setting, where each class appears in a single task and does not reappear later (Koh et al., 2022; Moon et al., 2023). We present the test accuracy and forgetting results for Split-CIFAR-100 and Split-ImageNet-R under $\epsilon \in \{1, 8\}$ in Figure 1.

Our method improves over the cosine classifier on Split-CIFAR-100 and remains competitive with the PEFT Ensemble, even outperforming it at $\epsilon = 1$. On Split-ImageNet-R, our method performs on par with the cosine classifier, while the PEFT ensemble achieves the highest accuracy, highlighting a utility–interpretability trade-off. The larger performance drop on Split-ImageNet-R at $\epsilon = 1$ across all methods further suggests that this setting is particularly challenging under stronger privacy constraints.

For forgetting, our method generally performs best on Split-CIFAR-100 and remains comparable on Split-ImageNet-R. The cosine classifier shows lower forgetting on Split-ImageNet-R at $\epsilon = 1$, but also substantially lower accuracy, so forgetting should be interpreted jointly with task performance. The Naive baseline exhibits severe catastrophic forgetting, which illustrates the need for an explicit continual-learning mechanism.

5.3 Blurry class-incremental setting results

We next consider blurry class-incremental settings, where examples from some classes may appear across multiple tasks. We evaluate both i-Blurry (Koh et al., 2022) and Si-Blurry (Moon et al., 2023). For i-Blurry, we use 50% disjoint classes and a full blurry sample ratio of 100%. Si-Blurry further introduces stochastic variation in class composition across tasks, resulting in a more imbalanced and dynamic data stream.

Figure 2 compares our method with the cosine classifier and PEFT ensemble in the Not-Blurry, i-Blurry, and Si-Blurry settings. Our method outperforms the cosine classifier in most blurry configurations and remains comparatively robust as blurriness increases. Both prototype-based methods aggregate information at the class level, which makes them less sensitive to how examples are distributed across tasks, although their performance still depends on which labels are privately released.

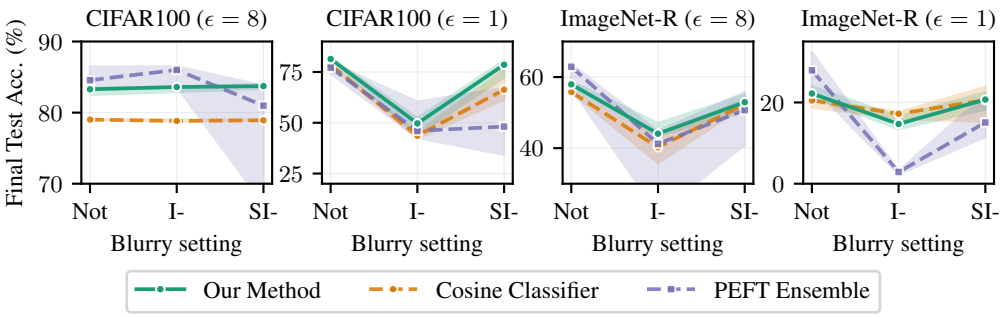

Figure 2: Final test accuracy under the Not-Blurry, i-Blurry, and Si-Blurry settings on Split-CIFAR-100 and Split-ImageNet-R for $\epsilon \in \{1, 8\}$: our method remains comparatively robust across increasingly blurry settings and is competitive with the cosine classifier, while PEFT ensemble shows greater sensitivity to blurriness. Results are the median over five repeats at $\delta = 10^{-5}$ with the error bars showing min and max seed. For Split-CIFAR-100 under i-Blurry, PEFT ensemble results are from the original experimental runs of Tobaben et al. (2026). All other results are obtained from our own runs.

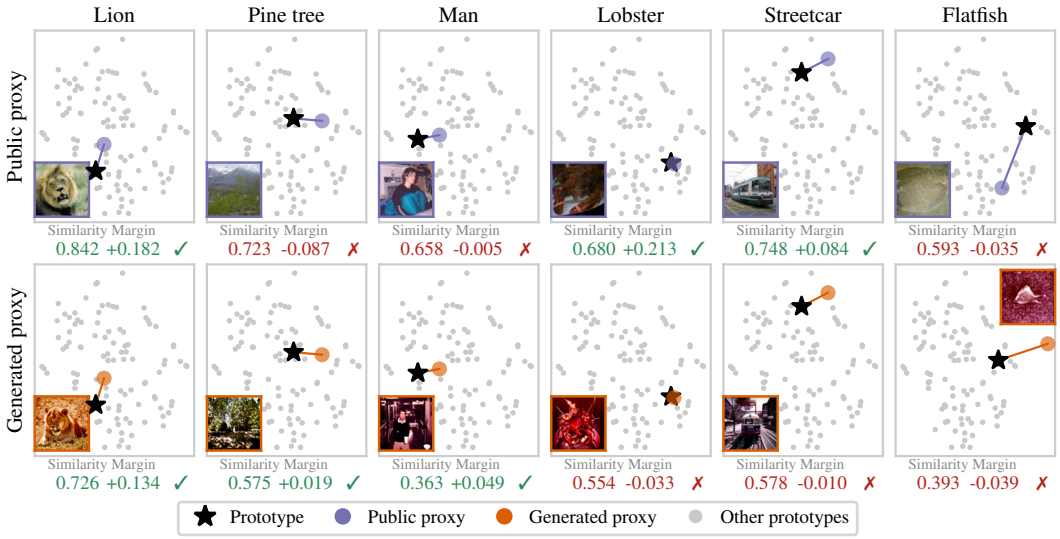

Figure 3: Public and generated proxies for six Split-CIFAR-100 prototypes, projected with Bi-RSNE (Aghasanli & Angelov, 2025). Both rows lie in the classifier's space. Numbers give the similarity of the proxy to its own prototype and its margin over the closest other prototype, with a tick (green) when the proxy is correct and a cross (red) otherwise.

The PEFT ensemble is more sensitive to the task-wise data distribution, especially in the Si-Blurry, where different ensemble members are trained on substantially different amounts of data.

5.4 Interpretability interface results

Since the interpretation modes depend only on the released DP model parameters and the test input, they are post-processing of the DP release. Prototype visualizations must likewise be constructed from public data rather than unreleased private samples. This is how the proxies of Section 4.5 are built. For interpretability, we report the *fidelity* of each proxy method, that is, the percentage of correct proxies.

Both proxies are recognised for the large majority of classes, each reaching a fidelity score of 92% (Table 1). In most cases the proxy is a clear instance of the class. For *lion*, for example, both methods produce an image that the classifier matches with positive margin Figure 3. When a proxy does fail, it is read as a neighbour class of similar appearance rather than something unrelated, though for different reasons for each side.

Table 1: Fidelity of each proxy method on Split-CIFAR-100 ($\epsilon = 8$), that is, the percentage of classes whose proxy the classifier assigns to its own prototype. For the gated variant, the second column gives the number of classes handed over to the generated proxy.

Proxy	Handed over	Fidelity (%)
Public	-	92
Generated	-	92
Gated ($s_{\min} = 0.70$)	25	92
Gated ($s_{\min} = 0.73$)	46	95
Gated ($s_{\min} = 0.75$)	60	93

A closer look at similarity and margin in Figure 3 shows what these failures consist of (Appendix G.2 for all cases). A public proxy is a real photograph and rarely shows the intended object alone. Its similarity can therefore be as high as that of a correct proxy, while the margin reveals where the rest of the image leads (Figure 3, top). For example, the proxy of *pine tree* is as similar to its prototype as the most correct proxy (0.72). Yet the nearest image the public dataset offers shows trees against a mountain range, and the mountain pulls it to that prototype by -0.09. The proxy of *man* shows a person with long hair. It is read as a woman by only -0.005. The generated proxy, on the other hand, produces a single object. It fails instead when that object could also belong to a neighbouring class. This is because a class prototype is the average of many images and may have lost the details that separate the class from a close neighbour (Ye et al., 2023).

Details that the prototype does not capture are left to the decoder to fill in (Ramesh et al., 2022), and the result may lean toward the neighbouring class. For example, its *street car* is a tram in every draw, but the averaged prototype may keep little of what sets a tram apart from a train, and the kept draw, shown head-on on its tracks, is read as a train by -0.01 (Figure 3, bottom). Its *lobster* is a red crustacean on a plate whose body the decoder appears to have filled in as that of a crab, and it is read as one by -0.03. Conversely, the generated proxy recovers the classes the public proxy fails.

This complementarity is what the gated combination turns to advantage. With $s_{\min} = 0.73$, 46 classes are routed to the generated proxy and fidelity rises to 95%, six of the eight public failures receive a correct generated image. At 0.7 the classes handed over first are those generated proxy itself fails, such as *lobster*, and fadility falls to 90%. A higher threshold recovers *cloud* but hands over classes the public proxy had right, such as *streetcar*, and at 0.75 fidelity is 93%. The useful range is therefore narrow, since the classes each proxy misses lie close together in similarity.

6 Conclusion and Discussion

We introduced an interpretable differentially private continual learning framework based on class prototypes. Our method freezes the pretrained backbone and continually adapts a linear projection to new tasks using three prototype-based losses: prototype-contrastive, push-away, and preservation losses. Across disjoint and blurry continual-learning settings on Split-CIFAR-100 and Split-ImageNet-R, our method outperforms or is on par with the interpretable Cosine Classifier, while remaining competitive with the stronger but non-interpretable PEFT Ensemble in several settings. We further provide an interface for interpreting the learned private prototypes using public and synthetic examples.

Our interpretability interface assumes access to suitable public data and optionally a generative model. Extending interpretable DP continual learning beyond computer vision and replacing the translator with a generative model that shares the classifier's encoder remain an important direction for future work. More broadly, our results highlight the difficulty of jointly achieving privacy, continual adaptation, and interpretability, and motivate further development of methods that balance these objectives.

AI USE STATEMENT

In this work, we used generative AI tools to polish and improve the readability of text originally written by the authors. We also used generative AI tools to assist with implementation-related coding tasks, including debugging, code cleaning and reformatting, and the preparation of plotting and experiment-launch scripts.

The research ideas, methodological contributions, experimental design, and scientific conclusions presented in this work were developed by the authors. Generative AI tools were not used to generate the novel ideas or core methodological contributions of this work.

All AI-assisted text and code were reviewed by the authors, and AI-assisted code was inspected and tested for correctness. We take responsibility for the final content of this work, including text, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our work does not involve human subjects or the collection of new personal data. All experiments use publicly available datasets and publicly released pretrained backbones. The proposed method aims to advance privacy-preserving machine learning by providing formal differential privacy guarantees while supporting interpretable predictions in continual-learning settings. The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

The main paper describes the proposed method and experimental setup. The additional implementation details, hyperparameter tuning, and evaluation protocols are provided in Section E. The source code and experiment configurations required to reproduce the experiments are available at `https://anonymous.4open.science/r/Interpretable-Differentially-Private-Continual-Learning-with-Prototype-Based-Classifiers-C36B/`.

REFERENCES

Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016.

Agil Aghasanli and Plamen P Angelov. Recursive SNE: Fast prototype-based t-SNE for large-scale and online data. *Transactions on Machine Learning Research*, 2025.

Agil Aghasanli, Yi Li, and Plamen Angelov. Prototype-based continual learning with label-free replay buffer and cluster preservation loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 6545–6554, 2025.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

Borja Balle and Yu-Xiang Wang. Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, volume 80, pp. 403–412, 2018.

Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Computer Vision – ECCV 2018*, volume 11215, pp. 556–572, 2018.

Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, 2019.

Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8230–8239, 2021.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. Continual learning in neural networks: An overview. *Neural Networks*, 136:76–95, 2021.

Edoardo Debenedetti, Giorgio Severi, Milad Nasr, Christopher A. Choquette-Choo, Matthew Jagielski, Eric Wallace, Nicholas Carlini, and Florian Tramèr. Privacy side channels in machine learning systems. In *USENIX Security Symposium*, 2024.

Pradnya Desai, Phung Lai, NhatHai Phan, and My T. Thai. Continual learning with differential privacy. In *International Conference on Neural Information Processing*, pp. 334–343, 2021.

Damien Desfontaines, James Voss, Bryant Gipson, and Chinmoy Mandayam. Differentially private partition selection. *Proceedings on Privacy Enhancing Technologies*, 2022(1):339–352, 2022.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017. URL https://arxiv.org/abs/1702.08608.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques.*, pp. 486–503, 2006a.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pp. 265–284, 2006b.

Google. Google's DP accounting library. GitHub repository, https://github.com/google/differential-privacy/tree/main/python/dp_accounting, 2020.

Diego Granziol, Stefan Zohren, and Stephen Roberts. Learning rates as a function of batch size: A random matrix theory approach to neural network training. *Journal of Machine Learning Research*, 23(173):1–65, 2022.

Ahmad Hassanpour, Majid Moradikia, Bian Yang, Ahmed Abdelhadi, Christoph Busch, and Julian Fierrez. Differential privacy preservation in robust continual learning. *IEEE Access*, 10:24273–24287, 2022.

Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021.

Dmitry Kangin and Plamen Angelov. Unsupervised domain adaptation within deep foundation latent spaces. *arXiv preprint arXiv:2402.14976*, 2024.

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on class incremental blurry task configuration with anytime inference, 2022. URL https://arxiv.org/abs/2110.10031.

Antti Koskela and Tejas D Kulkarni. Practical differentially private hyperparameter tuning with subsampling. *Advances in Neural Information Processing Systems*, 36:28201–28225, 2023.

Antti Koskela, Joonas Jälkö, and Antti Honkela. Computing tight differential privacy guarantees using FFT. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*, volume 108, pp. 2560–2569, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

Songning Lai, Mingqian Liao, Zhangyi Hu, Jiayu Yang, Wenshuo Chen, Hongru Xiao, Jianheng Tang, Haicheng Liao, and Yutao Yue. Learning new concepts, remembering the old: Continual learning for multimodal concept bottleneck models. In *Proceedings of the ACM International Conference on Multimedia*, 2024.

Yuting Lai et al. Lifelong machine learning with differential privacy. In *IEEE International Conference on Data Mining Workshops*, 2022.

Béatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *The Annals of Statistics*, 28(5):1302–1338, 2000.

Valentino Maiorca, Luca Moschella, Antonio Norelli, Marco Fumero, Francesco Locatello, and Emanuele Rodolà. Latent space translation via semantic alignment. *Advances in Neural Information Processing Systems*, 36:55394–55414, 2023.

Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the SDEs and scaling rules for adaptive gradient algorithms. *Advances in Neural Information Processing Systems*, 35: 7697–7711, 2022.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pp. 109–165, 1989.

Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Communications of the ACM*, 53(9):89–97, 2010.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations*, 2021.

Jun-Yeong Moon, Keon-Hee Park, Jung Uk Kim, and Gyeong-Moon Park. Online class incremental learning on stochastic blurry task boundary via mask and visual prompt tuning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11697–11707, 2023.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations*, 2024.

Gauri Pradhan, Joonas Jalko, Marlon Tobaben, and Antti Honkela. Hyperparameters in Score-Based Membership Inference Attacks . In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 362–384, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, A. Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Arun Rajkumar and Shivani Agarwal. A differentially private stochastic gradient descent algorithm for multiparty classification. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 22, pp. 933–941, 2012.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Dawid Damian Rymarczyk, Joost van de Weijer, Bartosz Zieli'nski, and Bartłomiej Twardowski. ICICLE: Interpretable class incremental continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1887–1898, 2023.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. LAION-5B: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

Sarath Sivaprasad, Dmitry Kangin, Plamen Angelov, and Mario Fritz. COMIX: Compositional explanations using prototypes. *arXiv preprint arXiv:2501.06059*, 2025.

David M. Sommer, Sebastian Meiser, and Esfandiar Mohammadi. Privacy loss classes: The central limit theorem in differential privacy. *Proceedings on Privacy Enhancing Technologies*, 2019(2): 245–269, 2019.

Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. Stochastic gradient descent with differentially private updates. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 245–248, 2013.

Marlon Tobaben, Aliaksandra Shysheya, John Bronskill, Andrew Paverd, Shruti Tople, Santiago Zanella-Beguelin, Richard E. Turner, and Antti Honkela. On the efficacy of differentially private few-shot image classification. *Transactions on Machine Learning Research*, 2023.

Marlon Tobaben, Talal Alrawajfeh, Marcus Klasson, Mikko A. Heikkilä, Arno Solin, and Antti Honkela. Privacy leakage via output label space and differentially private continual learning. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=ZshFgRQWrm. Expert Certification.

Federico Di Valerio, Michela Proietti, Alessio Ragno, and Roberto Capobianco. CIP-Net: Continual interpretable prototype-based network. In *AAAI Conference on Artificial Intelligence*, 2025.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pp. 631–648, 2022.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Sin-Han Yang, Tuomas Oikarinen, and Tsui-Wei Weng. Towards interpretable continual learning through controlling concepts. OpenReview manuscript, 2024. URL https://openreview.net/forum?id=pGL4P2kg6V.

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. IP-Adapter: Text-compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. In *International Conference on Learning Representations*, 2022.

Lu Yu, Haoyu Han, Zhe Tao, Hantao Yao, and Chang-Sheng Xu. Language guided concept bottleneck models for interpretable continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14976–14986, 2025.

Xu Zheng, Farhad Shirani, Tianchun Wang, Wei Cheng, Zhuomin Chen, Haifeng Chen, Hua Wei, and Dongsheng Luo. Towards robust fidelity for evaluating explainability of graph neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

A ORIGINAL LOSSES ADAPTED FOR OUR METHOD

Non-DP supervised contrastive loss (Khosla et al., 2020; Aghasanli et al., 2025). Let $\mathcal{B} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ be a minibatch of N examples, and let $\bar{\boldsymbol{z}}_i = \boldsymbol{z}_i / \|\boldsymbol{z}_i\|_2$ denote the normalized representation of input $\boldsymbol{x}_i$. For each anchor sample $(\boldsymbol{x}_i, y_i)$, define

$$\mathcal{A}(i) = \{1, \dots, N\} \setminus \{i\} \tag{11}$$

as the set of indices of all other minibatch samples, and

$$\mathcal{P}(i) = \{q \in \mathcal{A}(i) : y_q = y_i\} \tag{12}$$

as the set of indices of positive samples sharing the same label as the anchor. The non-DP supervised contrastive loss which our loss in Equation (5) is based on is given by

$$\mathcal{L}_{\text{supcon}} = -\frac{1}{N} \sum_{i=1}^{N} \frac{1}{|\mathcal{P}(i)|} \sum_{q \in \mathcal{P}(i)} \log \frac{\exp(\bar{\boldsymbol{z}}_i^\top \bar{\boldsymbol{z}}_q / \tau)}{\sum_{a \in \mathcal{A}(i)} \exp(\bar{\boldsymbol{z}}_i^\top \bar{\boldsymbol{z}}_a / \tau)}. \tag{13}$$

Non-DP push-away loss (Aghasanli et al., 2025). For each class c, the mean of the class representations is given by

$$\boldsymbol{\mu}_c = \frac{1}{|\mathcal{D}_c|} \sum_{(\boldsymbol{x}, y) \in \mathcal{D}_c} \bar{\boldsymbol{z}}(\boldsymbol{x}). \tag{14}$$

The push-away loss is defined as

$$\mathcal{L}_{\text{push}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C_{\text{prev}}} \frac{\bar{\boldsymbol{z}}_i^\top \boldsymbol{\mu}_j}{(1 - \sigma_j)\tau_{\text{push}}}. \tag{15}$$

Here, C_{prev} denotes the number of labels from earlier tasks, σ_j denotes the standard deviation of the latent representations of the j-th class, and τ_{push} is a hyperparameter that scales the similarity scores.

Non-DP cluster preservation loss (Aghasanli et al., 2025). The idea is to preserve previously learned cluster structure using stored prototypes and support samples. After learning a task, clusters are formed in the latent space, and the sample whose latent representation is closest to each cluster centre is selected as its prototype. For each cluster, additional target points are constructed as $\boldsymbol{\mu}_k \pm b\sigma_{k,\ell}$, for $b \in \{1, 2, 3\}$, along informative latent dimensions ℓ, where $\boldsymbol{\mu}_k$ is the centre of cluster k and $\sigma_{k,\ell}$ is its standard deviation along dimension ℓ. For each target point, the training sample whose latent representation is closest to it is selected as a support sample. For both the prototype and support samples, their pre-projection representations and corresponding latent representations are stored.

Let $\mathbf{Z}_{\text{old}}$ denote the stored latent representations of the prototypes and support samples, and let $\mathbf{Z}_{\text{new}}$ denote the latent representations obtained by passing their stored pre-projection representations through the updated projection. The preservation loss is

$$\mathcal{L}_{\text{preserve}} = \text{MMD}^2 \left(\mathbf{Z}_{\text{old}}, \mathbf{Z}_{\text{new}} \right). \tag{16}$$

B PROOFS FOR PROTOTYPE STABILITY THEORY

Proof of Theorem 1. Fix any unit-norm feature vector $\bar{\boldsymbol{z}}$. For every $j \in \mathcal{O}_{1:t}$,

$$\left|\bar{\boldsymbol{z}}^\top \tilde{\boldsymbol{p}}_j^{(t)} - \bar{\boldsymbol{z}}^\top \boldsymbol{p}_j^{(t)}\right| \le \|\bar{\boldsymbol{z}}\|_2 \|\tilde{\boldsymbol{p}}_j^{(t)} - \boldsymbol{p}_j^{(t)}\|_2 \le \gamma_t, \tag{17}$$

where the first inequality is Cauchy–Schwarz and the second follows from the definition of γ_t. Therefore,

$$\bar{\boldsymbol{z}}^\top \tilde{\boldsymbol{p}}_y^{(t)} \ge \bar{\boldsymbol{z}}^\top \boldsymbol{p}_y^{(t)} - \gamma_t \tag{18}$$

and, for every $j \ne y$,

$$\bar{\boldsymbol{z}}^\top \tilde{\boldsymbol{p}}_j^{(t)} \le \bar{\boldsymbol{z}}^\top \boldsymbol{p}_j^{(t)} + \gamma_t. \tag{19}$$

It follows that

$$\begin{aligned} \tilde{m}_t(\bar{\boldsymbol{z}}) &= \bar{\boldsymbol{z}}^\top \tilde{\boldsymbol{p}}_y^{(t)} - \max_{j \in \mathcal{O}_{1:t} \setminus \{y\}} \bar{\boldsymbol{z}}^\top \tilde{\boldsymbol{p}}_j^{(t)} \\ &\ge \bar{\boldsymbol{z}}^\top \boldsymbol{p}_y^{(t)} - \max_{j \in \mathcal{O}_{1:t} \setminus \{y\}} \bar{\boldsymbol{z}}^\top \boldsymbol{p}_j^{(t)} - 2\gamma_t \\ &= m_t(\bar{\boldsymbol{z}}) - 2\gamma_t. \end{aligned} \tag{20}$$

If $m_t(\bar{\boldsymbol{z}}) > 2\gamma_t$, then $\tilde{m}_t(\bar{\boldsymbol{z}}) > 0$. Moreover, $m_t(\bar{\boldsymbol{z}}) > 0$. Thus, both classifiers predict y. □

Lemma 1. *Let $\boldsymbol{a} \ne \boldsymbol{0}$ and $\tilde{\boldsymbol{a}} = \boldsymbol{a} + \boldsymbol{e}$, and define*

$$\boldsymbol{p} = \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|_2}, \qquad \tilde{\boldsymbol{p}} = \frac{\tilde{\boldsymbol{a}}}{\|\tilde{\boldsymbol{a}}\|_2}. \tag{21}$$

If $\rho = \|\boldsymbol{e}\|_2 / \|\boldsymbol{a}\|_2 < 1$, then

$$\boldsymbol{p}^\top \tilde{\boldsymbol{p}} \ge \frac{1-\rho}{1+\rho}, \qquad \|\tilde{\boldsymbol{p}} - \boldsymbol{p}\|_2 \le 2\rho. \tag{22}$$

Proof. First, the reverse triangle inequality gives

$$\|\tilde{\boldsymbol{a}}\|_2 \ge \|\boldsymbol{a}\|_2 - \|\boldsymbol{e}\|_2 = (1-\rho)\|\boldsymbol{a}\|_2 > 0, \tag{23}$$

so $\tilde{\boldsymbol{p}}$ is well defined. Next, Cauchy–Schwarz and the triangle inequality give, respectively,

$$\boldsymbol{a}^\top \tilde{\boldsymbol{a}} = \|\boldsymbol{a}\|_2^2 + \boldsymbol{a}^\top \boldsymbol{e} \ge (1-\rho)\|\boldsymbol{a}\|_2^2, \qquad \|\tilde{\boldsymbol{a}}\|_2 \le (1+\rho)\|\boldsymbol{a}\|_2. \tag{24}$$

Substituting these bounds into the cosine similarity yields

$$\boldsymbol{p}^\top \tilde{\boldsymbol{p}} = \frac{\boldsymbol{a}^\top \tilde{\boldsymbol{a}}}{\|\boldsymbol{a}\|_2 \|\tilde{\boldsymbol{a}}\|_2} \ge \frac{1-\rho}{1+\rho}, \tag{25}$$

where the numerator lower bound is positive because $\rho < 1$.

For the distance bound, add and subtract $\tilde{\boldsymbol{a}}/\|\boldsymbol{a}\|_2$ and apply the triangle inequality:

$$\begin{aligned} \|\tilde{\boldsymbol{p}} - \boldsymbol{p}\|_2 &= \left\| \frac{\tilde{\boldsymbol{a}}}{\|\tilde{\boldsymbol{a}}\|_2} - \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|_2} \right\|_2 \\ &\le \left\| \frac{\tilde{\boldsymbol{a}}}{\|\tilde{\boldsymbol{a}}\|_2} - \frac{\tilde{\boldsymbol{a}}}{\|\boldsymbol{a}\|_2} \right\|_2 + \frac{\|\tilde{\boldsymbol{a}} - \boldsymbol{a}\|_2}{\|\boldsymbol{a}\|_2} \\ &= \frac{|\|\boldsymbol{a}\|_2 - \|\tilde{\boldsymbol{a}}\|_2|}{\|\boldsymbol{a}\|_2} + \frac{\|\boldsymbol{e}\|_2}{\|\boldsymbol{a}\|_2}. \end{aligned} \tag{26}$$

Finally, $|\|\tilde{\boldsymbol{a}}\|_2 - \|\boldsymbol{a}\|_2| \le \|\tilde{\boldsymbol{a}} - \boldsymbol{a}\|_2 = \|\boldsymbol{e}\|_2$ by the reverse triangle inequality. Each term in the last display is therefore at most ρ, proving $\|\tilde{\boldsymbol{p}} - \boldsymbol{p}\|_2 \le 2\rho$. □

Lemma 2. *Assume that $\xi_c^{(t)} \sim \mathcal{N}(\boldsymbol{0}, \sigma_{\mathrm{DP}}^2 \boldsymbol{I}_{d_h})$ for all $c \in \mathcal{O}_{1:t}$, and let $K_t = |\mathcal{O}_{1:t}|$. Then, for every $\alpha \in (0,1)$, with probability at least $1-\alpha$,*

$$\|\xi_c^{(t)}\|_2 \le r_\alpha = \sigma_{\mathrm{DP}} \left(\sqrt{d_h} + \sqrt{2\log(K_t/\alpha)} \right) \qquad \textit{for every } c \in \mathcal{O}_{1:t}. \tag{27}$$

Proof. For $\xi \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathrm{DP}}^2 \boldsymbol{I}_{d_h})$, a standard consequence of the Laurent–Massart concentration inequality for chi-square variables (Laurent & Massart, 2000) gives

$$\Pr\left(\|\xi\|_2 \geq \sigma_{\mathrm{DP}}(\sqrt{d_h} + \sqrt{2u})\right) \leq e^{-u} \tag{28}$$

for every $u > 0$. Setting $u = \log(K_t/\alpha)$ gives

$$\Pr\left(\|\xi_c^{(t)}\|_2 \geq r_\alpha\right) \leq \frac{\alpha}{K_t} \tag{29}$$

for each fixed class c. A union bound over the K_t released classes gives

$$\Pr\left(\exists c \in \mathcal{O}_{1:t} : \|\xi_c^{(t)}\|_2 \geq r_\alpha\right) \leq \alpha, \tag{30}$$

which proves the claim. □

Proof of Theorem 2. Assume the event in Lemma 2 holds. For each $c \in \mathcal{O}_{1:t}$, set

$$\boldsymbol{a}_c^{(t)} = \boldsymbol{W}_{\boldsymbol{\phi}} \boldsymbol{u}_c^{(t)}, \qquad \boldsymbol{e}_c^{(t)} = \boldsymbol{W}_{\boldsymbol{\phi}} \xi_c^{(t)}. \tag{31}$$

Then

$$\|\boldsymbol{e}_c^{(t)}\|_2 \leq \|\boldsymbol{W}_{\boldsymbol{\phi}}\|_{\mathrm{op}} \|\xi_c^{(t)}\|_2 \leq \|\boldsymbol{W}_{\boldsymbol{\phi}}\|_{\mathrm{op}} r_\alpha. \tag{32}$$

From the theorem's condition,

$$\rho_c^{(t)} := \frac{\|\boldsymbol{e}_c^{(t)}\|_2}{\|\boldsymbol{a}_c^{(t)}\|_2} \leq \frac{\|\boldsymbol{W}_{\boldsymbol{\phi}}\|_{\mathrm{op}} r_\alpha}{\|\boldsymbol{W}_{\boldsymbol{\phi}} \boldsymbol{u}_c^{(t)}\|_2} < 1. \tag{33}$$

Apply Lemma 1 with $\boldsymbol{a} = \boldsymbol{a}_c^{(t)}$ and $\boldsymbol{e} = \boldsymbol{e}_c^{(t)}$. This yields

$$\|\tilde{\boldsymbol{p}}_c^{(t)} - \boldsymbol{p}_c^{(t)}\|_2 \leq 2\rho_c^{(t)} \leq \frac{2\|\boldsymbol{W}_{\boldsymbol{\phi}}\|_{\mathrm{op}} r_\alpha}{\|\boldsymbol{W}_{\boldsymbol{\phi}} \boldsymbol{u}_c^{(t)}\|_2} \tag{34}$$

simultaneously for all $c \in \mathcal{O}_{1:t}$. Taking the maximum over c bounds γ_t. If Equation (10) holds, then

$$m_t(\bar{\boldsymbol{z}}) > 2\gamma_t. \tag{35}$$

The classification claim follows from Theorem 1. The event used above has probability at least $1 - \alpha$ by Lemma 2. □

C Details of our method

Algorithms 1 and 2 provide pseudocode for the training and inference procedures of our method introduced in Section 4, respectively.

D PLD accounting

To account for the privacy loss of the multiple private mechanisms used within each task, we apply composition based on privacy loss distributions (PLDs), rather than basic composition, which directly sums the privacy parameters of the individual mechanisms.

D.1 Privacy loss distribution

A privacy loss distribution (PLD) describes the distribution of the privacy loss induced by a randomized mechanism. We first define the privacy loss function and then the corresponding PLD.

Definition 2 (Privacy loss function)**.** *Let P_X and P_Y denote the distributions of $\mathcal{M}(X)$ and $\mathcal{M}(Y)$, respectively, for neighbouring datasets $X \sim Y$, and assume $P_X \ll P_Y$. The privacy loss function of P_X relative to P_Y is*

$$\mathcal{L}_{X/Y}(z) = \log \frac{dP_X}{dP_Y}(z). \tag{36}$$

When P_X and P_Y admit densities f_X and f_Y, this reduces to

$$\mathcal{L}_{X/Y}(z) = \log \frac{f_X(z)}{f_Y(z)}. \tag{37}$$

Algorithm 1 Training algorithm of our method

Require: Number of tasks T, target privacy budget (ϵ, δ), DP-SGD hyperparameters Λ_{DP}

Initialize projection $g_{\boldsymbol{\phi}^{(0)}}$ randomly
Initialize an empty pre-projection prototype store
for $t \in [T]$ **do**
  Get task dataset $\mathcal{D}_t$
  Calibrate DP-SGD noise $\sigma_{\text{DP-SGD}}$ using the privacy budget and PLD composition
  ▷ **Private mechanism 1: label release**
  Privately release task labels $\mathcal{O}_t$
  for $c \in \mathcal{O}_t \setminus \mathcal{O}_{1:t-1}$ **do**
    $\tilde{\boldsymbol{p}}_c^{\mathrm{pre}(t-1)} \leftarrow \mathbf{0}$
  end for
  ▷ **Private mechanism 2: prototype computation**
  for $c \in \mathcal{O}_t$ **do**
    $\tilde{\boldsymbol{p}}_c^{\mathrm{pre}(t)} \leftarrow \tilde{\boldsymbol{p}}_c^{\mathrm{pre}(t-1)} + \sum_{(\boldsymbol{x},y)\in\mathcal{D}_{t,c}} \frac{h_{\boldsymbol{\theta}}(\boldsymbol{x})}{\|h_{\boldsymbol{\theta}}(\boldsymbol{x})\|_2} + \xi_c^{(t)}$
  end for
  ▷ **Construct projected prototypes**
  for $c \in \mathcal{O}_{1:t}$ **do**
    $\tilde{\boldsymbol{p}}_c^{(t)} \leftarrow \frac{g_{\boldsymbol{\phi}^{(t-1)}}(\tilde{\boldsymbol{p}}_c^{\mathrm{pre}(t)})}{\left\| g_{\boldsymbol{\phi}^{(t-1)}}(\tilde{\boldsymbol{p}}_c^{\mathrm{pre}(t)}) \right\|_2}$
  end for
  ▷ **Private mechanism 3: DP-SGD training**
  $\mathcal{L} = \lambda_{\mathrm{pcl}}\mathcal{L}_{\mathrm{pcl}} + \lambda_{\mathrm{push}}\mathcal{L}_{\mathrm{push}} + \lambda_{\mathrm{preserve}}\mathcal{L}_{\mathrm{preserve}}$
  $\boldsymbol{\phi}^{(t)} \leftarrow \text{DP-SGD}\left(\boldsymbol{\phi}^{(t-1)}, \mathcal{D}_t, \mathcal{L}, \Lambda_{\mathrm{DP}}, \sigma_{\text{DP-SGD}}\right)$
  ▷ **Update projected prototypes**
  for $c \in \mathcal{O}_{1:t}$ **do**
    $\tilde{\boldsymbol{p}}_c^{(t)} \leftarrow \frac{g_{\boldsymbol{\phi}^{(t)}}(\tilde{\boldsymbol{p}}_c^{\mathrm{pre}(t)})}{\left\| g_{\boldsymbol{\phi}^{(t)}}(\tilde{\boldsymbol{p}}_c^{\mathrm{pre}(t)}) \right\|_2}$
  end for
  Save $\boldsymbol{\phi}^{(t)}$, $\{\tilde{\boldsymbol{p}}_c^{\mathrm{pre}(t)}\}_{c\in\mathcal{O}_{1:t}}$, $\{\tilde{\boldsymbol{p}}_c^{(t)}\}_{c\in\mathcal{O}_{1:t}}$
end for
return $\boldsymbol{\phi}^{(T)}$, $\{\tilde{\boldsymbol{p}}_c^{\mathrm{pre}(T)}\}_{c\in\mathcal{O}_{1:T}}$, $\{\tilde{\boldsymbol{p}}_c^{(T)}\}_{c\in\mathcal{O}_{1:T}}$

Algorithm 2 Inference with DP prototypes at task $t \in \{1, \ldots, T\}$

Require: Test input $\boldsymbol{x}$, frozen backbone $h_{\boldsymbol{\theta}}$, projection parameters $\boldsymbol{\phi}^{(t)}$, prototypes $\{\tilde{\boldsymbol{p}}_c^{(t)}\}_{c\in\mathcal{C}}$

$\boldsymbol{z} \leftarrow g_{\boldsymbol{\phi}^{(t)}}(h_{\boldsymbol{\theta}}(\boldsymbol{x}))$
$\hat{y} \leftarrow \arg\max_{c\in\mathcal{C}} \cos\left(\boldsymbol{z}, \tilde{\boldsymbol{p}}_c^{(t)}\right)$
return $\hat{y}$

Definition 3 (Privacy loss distribution)**.** *The privacy loss distribution (PLD) of $\mathcal{M}(X)$ relative to $\mathcal{M}(Y)$ is the pushforward of P_X under $\mathcal{L}_{X/Y}$:*

$$\omega_{X/Y} = (\mathcal{L}_{X/Y})_{\#} P_X. \tag{38}$$

Equivalently, for every measurable set $A \subseteq \mathbb{R}$,

$$\omega_{X/Y}(A) = P_X\left(\mathcal{L}_{X/Y}^{-1}(A)\right) = \Pr_{Z\sim P_X}\left[\mathcal{L}_{X/Y}(Z) \in A\right]. \tag{39}$$

Representing each mechanism through its PLD enables composition at the level of the full privacy-loss distribution, rather than only through individual (ϵ, δ) guarantees. We use this framework for the privacy accounting of our method.

Algorithm 3 PLD-based privacy calibration

Require: Target privacy guarantee (ϵ, δ), label-release fraction f_{label}, prototype fraction α, sampling probability q, number of training steps S, clipping norm C

Ensure: Prototype noise scale σ_{proto} and DP-SGD noise multiplier $\sigma_{\text{DP-SGD}}$

▷ **Allocate privacy budget**

$(\epsilon_{\text{label}}, \delta_{\text{label}}) \leftarrow f_{\text{label}}(\epsilon, \delta)$

$(\epsilon_{\text{rem}}, \delta_{\text{rem}}) \leftarrow (\epsilon, \delta) - (\epsilon_{\text{label}}, \delta_{\text{label}})$

$(\epsilon_{\text{proto}}, \delta_{\text{proto}}) \leftarrow \alpha(\epsilon_{\text{rem}}, \delta_{\text{rem}})$

▷ **Construct non-training PLDs**

Construct $\mathcal{P}_{\text{label}}$ from the truncated geometric label-release mechanism

Calibrate σ_{proto} for a unit-sensitivity Gaussian mechanism with budget $(\epsilon_{\text{proto}}, \delta_{\text{proto}})$

Construct the Gaussian PLD $\mathcal{P}_{\text{proto}}$

▷ **Calibrate DP-SGD noise**

Define ComposedEpsilon(σ):

  Construct the single-step subsampled Gaussian PLD $\mathcal{P}_{\text{step}}(\sigma; q, C)$

  $\mathcal{P}_{\text{DP-SGD}}(\sigma) \leftarrow \mathcal{P}_{\text{step}}(\sigma; q, C)^{\circ S}$

  $\mathcal{P}_{\text{total}}(\sigma) \leftarrow \mathcal{P}_{\text{label}} \circ \mathcal{P}_{\text{proto}} \circ \mathcal{P}_{\text{DP-SGD}}(\sigma)$

  return $\epsilon_{\mathcal{P}_{\text{total}}(\sigma)}(\delta)$

Use binary search to find $\sigma_{\text{DP-SGD}}$ such that

$$\text{ComposedEpsilon}(\sigma_{\text{DP-SGD}}) \leq \epsilon$$

return $\sigma_{\text{proto}}, \sigma_{\text{DP-SGD}}$

D.2 Implementation details

Our method contains three DP mechanisms within each task: DP label release, DP prototype computation, and DP training of the projection layer. For each task, we construct the PLD of each DP mechanism and numerically compose them using Google's dp-accounting library (Google, 2020). Given a target privacy budget (ϵ, δ), we allocate a fraction of the budget to label release and use a fraction α of the remaining budget to calibrate the prototype mechanism. We then calibrate the DP-SGD noise multiplier such that the composition of all mechanisms satisfies the target privacy guarantee. The complete procedure is summarised in Algorithm 3.

To be specific, the label-release mechanism is represented by a discrete PLD constructed from neighbouring output distributions of the truncated geometric mechanism. The prototype computation is modeled as a Gaussian mechanism with sensitivity 1. DP-SGD is represented as a subsampled Gaussian mechanism whose single-step PLD is self-composed over all training steps. We then compose the PLDs of label release, prototype computation, and DP-SGD.

Importantly, α is used only to calibrate the prototype mechanism to $(\alpha\epsilon_{\text{rem}}, \alpha\delta_{\text{rem}})$. We do not independently assign the complementary budget $((1-\alpha)\epsilon_{\text{rem}}, (1-\alpha)\delta_{\text{rem}})$ to DP-SGD. Instead, we calibrate the DP-SGD noise multiplier by binary search directly against the composed PLD so that the overall mechanism satisfies the target (ϵ, δ) guarantee.

D.3 Privacy accounting ablation

Privacy accounting is part of the overall method. When evaluating our method against baselines in Section 5, we preserve the original basic composition used by the baselines instead of modifying their privacy accounting.

To isolate the effect of privacy accounting from the methodological contributions of our method, we additionally evaluate the cosine classifier and PEFT ensemble using PLD composition. We keep the model architecture, hyperparameters, and training procedure fixed, and change only the composition method. When switching from basic to PLD composition, we only recalibrate the DP noise required to satisfy the same target (ϵ, δ) under PLD accounting. Figure 4 and Table 2 show only minor differences between basic and PLD composition. This suggests that the improvements observed for our method are not primarily attributable to the use of tighter privacy accounting.

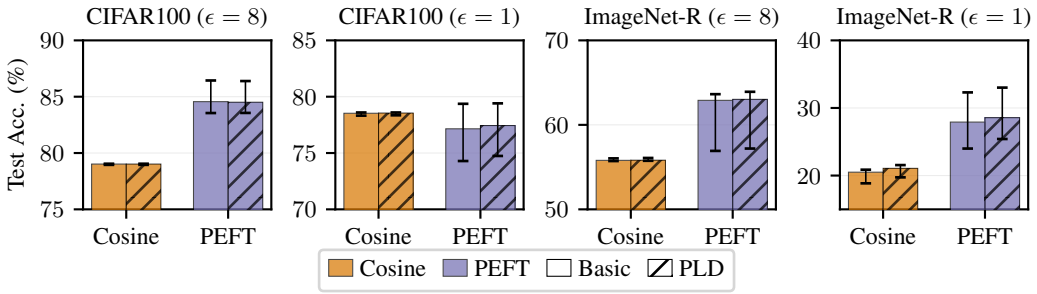

Figure 4: Final test accuracy of the cosine classifier and PEFT ensemble under basic and PLD composition on Split-CIFAR-100 and Split-ImageNet-R for $\epsilon \in \{1, 8\}$. Switching from basic to PLD composition results in negligible changes in performance. Bars show the median over five runs, with error bars indicating the minimum and maximum.

Table 2: Final test accuracy of the Cosine Classifier and PEFT Ensemble under basic and PLD composition, shown as the tabular version of Figure 4. Values are reported as median [minimum, maximum] over five runs.

Dataset	ϵ	Method	Basic composition	PLD composition
CIFAR-100	8	Cosine Classifier	79.02 [78.95, 79.05]	79.02 [78.95, 79.05]
		PEFT Ensemble	84.56 [83.55, 86.44]	84.51 [83.56, 86.39]
	1	Cosine Classifier	78.53 [78.32, 78.60]	78.55 [78.35, 78.60]
		PEFT Ensemble	77.15 [74.29, 79.37]	77.44 [74.74, 79.41]
ImageNet-R	8	Cosine Classifier	55.80 [55.72, 56.03]	55.82 [55.75, 56.08]
		PEFT Ensemble	62.90 [56.92, 63.63]	63.02 [57.20, 63.92]
	1	Cosine Classifier	20.50 [18.85, 20.87]	21.07 [19.73, 21.55]
		PEFT Ensemble	27.91 [23.99, 32.30]	28.56 [25.41, 33.01]

E Experimental Details

E.1 Evaluation Metrics

We provide the formal definitions of average accuracy and the forgetting measure used in our experiments, following prior work (Chaudhry et al., 2018; Mirzadeh et al., 2021; Yoon et al., 2022; Tobaben et al., 2026; Wang et al., 2024). Let $A_{t,i}$ denote the test accuracy on task i after training on task t.

- **Average accuracy (AA).** AA measures the model's overall classification performance across all tasks observed so far. After learning task t, it is computed as the mean test accuracy over these tasks:

$$\mathrm{AA}_t = \frac{1}{t} \sum_{i=1}^{t} A_{t,i}. \tag{40}$$

  Thus, AA_T denotes the average accuracy across all tasks after learning the final task T.

- **Forgetting measure (FM).** FM evaluates memory stability by quantifying how much performance on earlier tasks degrades as new tasks are learned. For an old task $i < t$, forgetting is the difference between the best accuracy previously achieved on that task and its accuracy after training on task t:

$$\mathrm{FM}_{t,i} = \max_{k \in \{1,\ldots,t-1\}} A_{k,i} - A_{t,i}. \tag{41}$$

  The forgetting measure after task t averages this quantity over previously seen tasks:

$$\mathrm{FM}_t = \frac{1}{t-1} \sum_{i=1}^{t-1} \mathrm{FM}_{t,i}. \tag{42}$$

Lower values indicate better retention. At the final step, FM_T is averaged over the first $T-1$ tasks, since the last learned task T has no subsequent task on which it could be forgotten.

Fidelity is the metric of the interpretability interface. After the final task T, when a proxy has been obtained for every class, fidelity is the fraction of the C classes whose proxy is assigned to its own prototype:

$$\text{Fidelity} = \frac{1}{C}\sum_{c=1}^{C} \mathbb{1}\Big[\, \tilde{\boldsymbol{z}}(x_c)^\top \tilde{\boldsymbol{p}}_c - \max_{j \neq c} \tilde{\boldsymbol{z}}(x_c)^\top \tilde{\boldsymbol{p}}_j > 0 \,\Big]. \tag{43}$$

The indicator is one when the embedding of x_c is closer to the prototype $\tilde{\boldsymbol{p}}_c$ of its own class than to any other prototype, so higher values indicate that more proxies are read as their own class. The name follows fidelity metrics for interpretability methods, which ask whether the model gives the same answer when shown only the interpretation (Zheng et al., 2024; Sivaprasad et al., 2025). Accordingly, fidelity asks the classifier whether a proxy stands for its prototype, since the meaning of a prototype lies only in the images the classifier assigns to it. The check uses the latent space and decision rule, and requires nothing beyond the released prototypes. Since selection considers the own prototype alone, a proxy can be close to its prototype and still be read as a neighbouring class, which is the failure mode analysed in Section 5.4.

E.2 Loss configration

For the prototype-based contrastive loss, we set the temperature to $\tau = 0.7$. The loss weights in Equation (4) are fixed to $\lambda_{\text{pcl}} = 1$, $\lambda_{\text{push}} = 10$, and $\lambda_{\text{preserve}} = 10$ across all datasets and privacy levels. We selected these values based on preliminary experiments so that the gradient contributions of the individual loss terms are on comparable scales. We kept them fixed in the main experiments to avoid substantially increasing the dimensionality of the hyperparameter search.

For cluster preservation, we generated five orthonormal directions once and kept them fixed throughout training. Along each direction, we constructed synthetic support points in both positive and negative directions at radii $r_k \in \{r, 2r, 3r\}$, with $r = 0.1$.

E.3 Hyperparameter tuning

We perform hyperparameter tuning using Optuna with the Gaussian process sampler for 50 iterations. We optimise the DP-Adam learning rate and batch size, and two privacy allocation hyperparameters: label release fraction, which specifies the fraction of the total privacy allocated to private label release, and α, which specifies the fraction of the remaining privacy budget allocated to prototype computation. The corresponding search ranges are summarised in Table 3.

Table 3: Hyperparameter ranges used for tuning.

Hyperparameter	**Lower bound**	**Upper bound**
Epochs	40	
Learning rate	10^{-4}	3×10^{-2}
Batch size	512	Tuning dataset size
Clipping norm	1	
Noise multiplier	Based on target ϵ	
Alpha (α)	0.7	0.9
Label release fraction	Based on target dataset and target ϵ	

For each dataset and target privacy budget, we first compute a deterministic label-release fraction that guarantees the release of all labels under the label-release mechanism, as described in Section E.4. Since this criterion is conservative, particularly when some classes contain relatively few samples, we use the resulting value as an upper bound and tune the label-release fraction over smaller values. The search ranges are reported in Table 4.

We follow the hyperparameter tuning protocol of Tobaben et al. (2026) to reduce the computational cost of HPO. For Split-CIFAR-100 and Split-ImageNet-R, we construct a representative tuning

Table 4: Label release fraction tuning ranges. $\delta = 10^{-5}$ for all settings.

Dataset	ϵ_{target}	Lower bound	Upper bound
CIFAR-100	1	0.005	0.05
CIFAR-100	8	0.001	0.01
ImageNet-R	1	0.01	0.50
ImageNet-R	8	0.01	0.10

dataset corresponding to a single task (10 and 20 classes, respectively). We split this dataset into 70% training and 30% validation data and select the hyperparameter configuration that achieves the highest validation accuracy. For both target datasets, we linearly scale the expected batch size to maintain a constant subsampling ratio (Koskela & Kulkarni, 2023) and scale the learning rate of Adam by the square root of the corresponding scaling factor, following Granziol et al. (2022); Malladi et al. (2022). The clipping norm is kept fixed, and the number of training epochs is set to 40 for all experiments.

E.4 Determining the label-release fraction upper bound

For a candidate fraction f, we allocate $f\epsilon$ and $f\delta$ to the label-release mechanism. From this privacy budget, the mechanism determines the truncation threshold k.

A class y is released if

$$|C_t(y)| + Z_y > k,$$

where $|C_t(y)|$ is the true class count and $Z_y \sim k\text{-TSGD}(p)$ is the truncated geometric noise. By definition of the probability mass function of the k-TSGD distribution (Desfontaines et al., 2022),

$$\Pr(Z_y = z) = \begin{cases} c(1-p)^{|z|}, & z \in [-k, k] \cap \mathbb{Z}, \\ 0, & \text{otherwise}, \end{cases}$$

for a normalizing constant c. Hence,

$$Z_y \in \{-k, -k+1, \ldots, k\},$$

and the smallest possible noise realization is $Z_y = -k$.

Therefore, even though the mechanism is stochastic, we can derive a deterministic worst-case guarantee.

The worst case for releasing a class is $Z_y = -k$. Hence,

$$|C_t(y)| - k > k,$$

which gives

$$|C_t(y)| > 2k.$$

Thus, if

$$\min_y |C_t(y)| > 2k,$$

all classes are guaranteed to be released regardless of the sampled noise.

Finally, we use binary search over f to find the smallest privacy-budget fraction that satisfies this condition. The procedure is summarised in Algorithm 4.

F Ablation of loss components

To evaluate the individual contribution of the three loss components: prototype-based contrastive loss, push-away loss, and preservation loss (described in Section 4.2), we conducted an ablation study on Split-CIFAR-100, as shown in 5.

The full loss function, $\mathcal{L} = \mathcal{L}_{\text{pcl}} + \lambda_{\text{push}}\mathcal{L}_{\text{push}} + \lambda_{\text{preserve}}\mathcal{L}_{\text{preserve}}$, achieved the best performance at both privacy levels, reaching 83.28% for $\epsilon = 8$ and 81.39% for $\epsilon = 1$. Removing the push-away loss leads to only a small decrease in performance, whereas removing the preservation loss leads to a

Algorithm 4 Minimum label release fraction for all label release

Require: Privacy budget (ϵ, δ), minimum class count $n_{\min}$, maximum fraction $f_{\max}$, tolerance τ
Ensure: Minimum label-release fraction $f^\star$

function GUARANTEEDRELEASE(f)
  $\epsilon_{\text{label}} \leftarrow f\epsilon$
  $\delta_{\text{label}} \leftarrow f\delta$
  $k \leftarrow \left\lceil \frac{1}{\epsilon_{\text{label}}} \log\left(\frac{\exp(\epsilon_{\text{label}}) + 2\delta_{\text{label}} - 1}{(\exp(\epsilon_{\text{label}}) + 1)\delta_{\text{label}}} \right) \right\rceil$
  return $n_{\min} > 2k$
end function
$l \leftarrow 0$
$h \leftarrow f_{\max}$
while $h - l > \tau$ **do**
  $f \leftarrow (l + h)/2$
  if GUARANTEEDRELEASE(f) **then**
    $h \leftarrow f$
  else
    $l \leftarrow f$
  end if
end while
return h

Table 5: Ablation of the proposed loss components on Split-CIFAR-100. We report the final test accuracy (%) for $\epsilon \in \{1, 8\}$. Values are reported as median [minimum, maximum] over five runs.

Loss Function			Final Test Accuracy (%)	
$\mathcal{L}_{\text{pcl}}$	$\mathcal{L}_{\text{push}}$	$\mathcal{L}_{\text{preserve}}$	$\epsilon = 8$	$\epsilon = 1$
✓	✓	✓	83.28 [82.51, 83.44]	81.39 [81.27, 82.14]
✓	✗	✓	82.72 [82.47, 83.25]	81.05 [80.47, 81.93]
✓	✓	✗	40.43 [31.47, 55.04]	58.82 [44.97, 70.43]

much larger drop. This suggests that the preservation loss is particularly important for maintaining performance, while the push-away loss provides an additional complementary benefit.

We kept all training and privacy hyperparameters fixed to those used by the full method and varied only which loss terms are included. We use the same loss weights as in the main experiments, $\lambda_{\text{pcl}} = 1$, $\lambda_{\text{push}} = 10$, and $\lambda_{\text{preserve}} = 10$, and do not retune the ablated variants. This allows us to assess the effect of removing each loss term without introducing additional differences from hyperparameter tuning.

G INTERPRETABILITY INTERFACE

G.1 TRANSLATOR

The translator B of Section 4.5 builds on Maiorca et al. (2023), who show that a map between the latent spaces of two separately trained encoders can be estimated in closed form, as a linear transformation, from a set of parallel anchors. We apply this idea to map the latent space into the ψ space, with three differences. B is trained on public images only, yet applied to DP prototypes of the private dataset, an input it never sees during training. We minimize cosine rather than Euclidean distance, since both spaces are used through cosine similarity. This objective has no closed-form minimiser, so B is trained by gradient descent whether it is linear or not. And since the two spaces differ in dimension, we use a two-layer MLP instead of a linear map, which lowers the held-out cosine distance from 0.19 to 0.18.

B was trained on the 50,000 images of ImageNet-1k, the same pool used for the public proxy. Each image was embedded once with the classifier and once with ψ, giving the input and target of B. The

two encoders need not share an input resolution, as each resizes the image with its own preprocessing before embedding. In this experiment, both take 224×224 inputs, to which the 64×64 public images are upsampled. B minimises $1 - \bar{\boldsymbol{z}}_\psi(x)^\top B(\bar{\boldsymbol{z}}(x))/\|B(\bar{\boldsymbol{z}}(x))\|$ averaged over the images. It maps the 512-dimensional latent space to the 1024-dimensional space of ψ through 2048 hidden units, and was trained for 60 epochs with AdamW, with 10% of the images reserved for validation. Since the decoder that consumes the translated prototype reads the norm of its input as well as its direction, the output of B is rescaled to the median norm of the class-mean CLIP embeddings of the public dataset, so that a translated prototype has the length of a class-level embedding.

G.2 Failure cases

Table 6 lists every class on which at least one proxy fails. The remaining public failures are of the same kind as discussed in Section 5.4. *Forest*, *maple tree* and *willow tree* retrieve landscapes in which a mountain, a house or another tree species dominates, while *cloud* and *mouse* retrieve plausible images whose nearest prototype is a close neighbour. The generated failures are all confusions between visually close classes and most are near ties, with *shrew* read as a mouse, *boy* as a baby, *plain* as a sea, *road* as a train and *oak tree* as a forest. Only *flatfish* fails for both proxies, read as a ray in either case.

The same table can also explain the threshold behaviour, where $s_{\min} = 0.70$ and 0.75 both fall short of 0.73. Three classes on which the generated proxy fails, *boy*, *lobster* and *shrew*, sit just below this band at 0.67 to 0.68 and are handed over first, which is the cost of -3 that every threshold above 0.68 pays. Another three, *oak tree*, *road* and *street car*, sit at 0.74 to 0.75, right where *cloud* would be reached, so raising the threshold to recover *cloud* loses more than it gains. The threshold of 0.73 is thus not a tuned value, but the only gap in which the public failures can be handed over without the second group.

Table 6: Classes on which at least one proxy fails on Split-CIFAR-100 ($\epsilon = 8$). Sim. is the cosine similarity between the proxy and its own prototype, margin its lead over the closest other prototype, and read as the class the classifier assigns the proxy to.

		Public proxy			Generated proxy		
Outcome	Class	Sim.	Margin	Read as	Sim.	Margin	Read as
Public fails	cloud	0.75	−0.013	sea	0.63	+0.094	cloud
	forest	0.73	−0.082	mountain	0.49	+0.050	forest
	man	0.66	−0.005	woman	0.36	+0.049	man
	maple tree	0.71	−0.017	house	0.61	+0.048	maple tree
	mouse	0.70	−0.041	hamster	0.49	+0.039	mouse
	pine tree	0.72	−0.087	mountain	0.57	+0.019	pine tree
	willow tree	0.73	−0.011	oak tree	0.57	+0.012	willow tree
Generated fails	boy	0.68	+0.006	boy	0.48	−0.009	baby
	lobster	0.68	+0.213	lobster	0.55	−0.033	crab
	oak tree	0.74	+0.011	oak tree	0.56	−0.028	forest
	plain	0.78	+0.088	plain	0.56	−0.005	sea
	road	0.74	+0.161	road	0.52	−0.003	train
	shrew	0.67	+0.075	shrew	0.57	−0.015	mouse
	streetcar	0.75	+0.084	streetcar	0.58	−0.010	train
Both fail	flatfish	0.59	−0.035	ray	0.39	−0.039	ray

G.3 Generated candidates

The generated proxy is the closest of four candidates drawn with different seeds. For well-formed prototypes such as *lion*, the candidates vary in pose and background but depict the same object at similar distance from the prototype, so any of them would serve as a proxy (Figure 5). For *rose*, one draw is clearly closer than the rest, yet all four are still assigned to their class, as are all draws of *man*

but one, despite their low similarity. For *lobster*, no draw is assigned to its class, so a different seed would not have changed the outcome. *streetcar* and *flatfish* show the limit of the nearest-candidate rule. In both cases two of the four draws are assigned to the correct class, but the draw closest to the prototype is not, and it is the one kept.

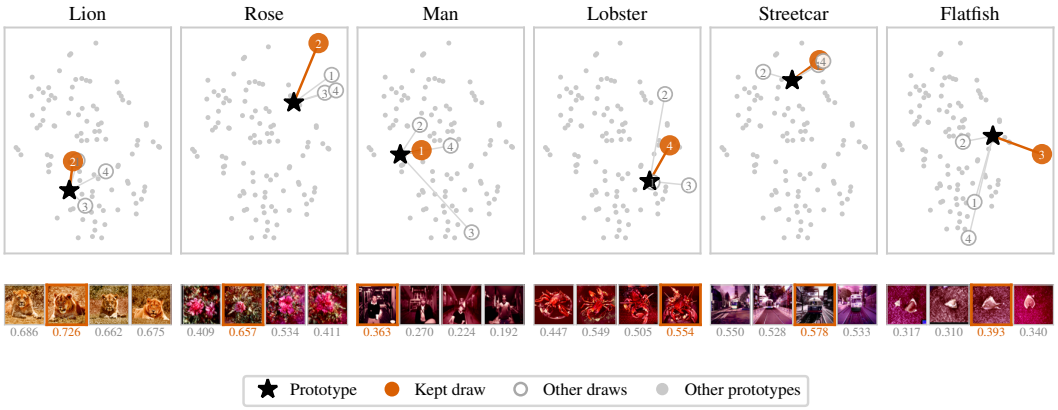

Figure 5: All four generated candidates for six prototypes, with the kept candidate highlighted. Numbers give the cosine similarity to the prototype in the latent space.