Contents

1. Gradient Analysis of Contrastive Prototypical Loss

Here, we provide an analysis of gradients for contrastive prototypical loss. To ease the notion, we abbreviate similarity between vector $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ as $s_{i,j}$ and denote negative set $N(i) \cup \hat{U}$ as $\hat{N}(i)$. Therefore, the loss of a data sample $\boldsymbol{x}_i$ is:

$$\mathcal{L}_i = \frac{-1}{|P(i)|} \sum_{\boldsymbol{z}_p \in P(i)} \log \frac{\exp(s_{i,p}/\tau)}{\sum\limits_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau)}. \tag{1}$$

The gradient with respect to the similarity $s_{i,j}$ between a positive pair $(\boldsymbol{z}_i, \boldsymbol{z}_j)$ for $z_j \in P(i)$ can be derived as:

$$\begin{aligned}
\frac{\partial \mathcal{L}_i^k}{\partial s_{i,j}} &= \frac{-1}{|P(i)|} \sum_{\boldsymbol{z}_p \in P(i)} \frac{\partial}{\partial s_{i,j}} \left(s_{i,p}/\tau - \log \sum_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau) \right), \\
&= \frac{-1}{|P(i)|} \sum_{\boldsymbol{z}_p \in P(i)} \left(\frac{1}{\tau} \cdot \mathbb{1}[p=j] - \frac{\frac{\partial}{\partial s_{i,j}} \left(\sum\limits_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau) \right)}{\sum\limits_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau)} \right), \\
&= \frac{-1}{|P(i)|} \sum_{\boldsymbol{z}_p \in P(i)} \left(\frac{1}{\tau} \cdot \mathbb{1}[p=j] - 0 \right), \\
&= \frac{-1}{\tau |P(i)|},
\end{aligned} \tag{2}$$

where $\mathbb{1}$ is an indicator function. Similarly, the gradient with respect to the similarity $s_{i,m}$ between a negative pair $(\boldsymbol{z}_i, \boldsymbol{z}_m)$ for $\boldsymbol{z}_m \in N(i)$ is:

$$\begin{aligned}
\frac{\partial \mathcal{L}_i^k}{\partial s_{i,m}} &= \frac{-1}{|P(i)|} \sum_{\boldsymbol{z}_p \in P(i)} \frac{\partial}{\partial s_{i,m}} \left(s_{i,p}/\tau - \log \sum_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau) \right), \\
&= \frac{-1}{|P(i)|} \sum_{\boldsymbol{z}_p \in P(i)} \left(0 - \frac{\frac{\partial}{\partial s_{i,m}} \left(\sum\limits_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau) \right)}{\sum\limits_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau)} \right), \\
&= \frac{1}{|P(i)|} \sum_{\boldsymbol{z}_p \in P(i)} \left(\frac{\sum\limits_{\boldsymbol{z}_n \in \hat{N}(i)} \left(\exp(s_{i,n}/\tau) \cdot \frac{1}{\tau} \cdot \mathbb{1}[n=m] \right)}{\sum\limits_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau)} \right), \\
&= \frac{1}{\tau} \frac{\exp(s_{i,m}/\tau)}{\sum\limits_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau)}.
\end{aligned} \tag{3}$$

We can see that, the gradient of the proposed loss follows the same pattern as the standard supervised contrastive loss. Positive pairs are treated equally and scaled by the temperature and the cardinality of the positive set. The property of implicit hard-case mining, *i.e.*, proportional to the exponential term $\exp(s_{i,m}/\tau)$, is inherited from a typical contrastive loss in the negative term.

2. CPP as an Energy-based Model

The overall objective of an energy-based model [10] (EBM) is to obtain an energy function $E_\theta(\boldsymbol{x}) : \mathbb{R}^D \rightarrow \mathbb{R}$ parameterized by θ that maps the high dimensional input $\boldsymbol{x}$ to a scalar value. Giving an energy function $E_\theta(\cdot)$, its probability density $p(\boldsymbol{x})$ can be expressed through Gibbs distribution:

$$p_\theta(y|\boldsymbol{x}) = \frac{\exp(-E_\theta(\boldsymbol{x}, y)/\tau)}{\int_{y'} \exp(-E_\theta(\boldsymbol{x}, y')/\tau)} = \frac{\exp(-E_\theta(\boldsymbol{x}, y)/\tau)}{\exp(-E_\theta(\boldsymbol{x})/\tau)}, \tag{4}$$

where $E_\theta(x)$ is the *Helmholtz free energy* and τ is the temperature factor. Then we have:

$$E_\theta(x) = \tau \cdot -\log \int_{y'} \exp(-E_\theta(\boldsymbol{x}, y')/\tau). \tag{5}$$

When making predictions under our framework, the categorical distribution can be represented as:

$$p(y|\boldsymbol{x}) = \frac{\exp(s_{x,y}/\tau)}{\sum_{y'=1}^{K} \exp(s_{x,y'}/\tau)}, \tag{6}$$

where $s_{x,y} = \langle f_{\{\theta,P\}}(\boldsymbol{x}), \hat{\boldsymbol{\mu}}_y \rangle$. When connecting Eq. 6 with Eq. 4 and let $E_\theta(\boldsymbol{x}, y) = -s_{x,y}$, we see that the energy of $\boldsymbol{x}$ can be expressed as:

$$E_\theta(\boldsymbol{x}) = \tau \cdot -\log \sum_{y=1}^{K} \exp(s_{x,y}/\tau), \tag{7}$$

which is dominated by the largest similarity s_{x,y^*} given an appropriate temperature τ. The above analysis drives to a conclusion that assigning a data sample to its nearest prototype will generate the lowest energy for the system (*i.e.*, a more stable system). Therefore, the question becomes whether the proposed contrastive prototypical loss serves as a qualified energy loss function.

To see this, we first simplify Eq. 1 to a formula where there is only one positive sample $\boldsymbol{z}_p$:

$$\begin{aligned} \mathcal{L}_i &= -\log \frac{\exp(s_{i,p}/\tau)}{\sum\limits_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau)}, \\ &= -s_{i,p} + \log \sum_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau). \end{aligned} \tag{8}$$

By substituting $\boldsymbol{z}_p$ with the target value prototype $\hat{\boldsymbol{\mu}}$, we have:

$$\mathcal{L}_i = \underbrace{-\langle \boldsymbol{z}_i, \hat{\boldsymbol{\mu}} \rangle}_{\text{push down energy for target prototype}} + \underbrace{\log \sum_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau)}_{\text{pull up energies for other prototypes}}. \tag{9}$$

When the above loss is minimized, the first term will push down the energy for target value prototype $\hat{\boldsymbol{\mu}}$ and the second term will increase energies for other prototypes. Hence, the above simplified loss is an effective loss function for the energy model. Note that the *ground-truth* prototype $\hat{\boldsymbol{\mu}}$ is unavailable during training stage and it is approximated by a group of dynamically evolved positive embeddings from train data. It is worth pointing out that, unlike the contrastive divergence approximation used in [11], we also include previous classifiers in our loss. Nevertheless, our formulation does not suffer from the over suppression issue of previous classes likewise, as we use static non-parametric prototypes as classifiers.

Finally, we show that encouraging uniformity is against the principle of the energy model. By encouraging uniformity as a typical supervised or self-supervised contrastive loss [5,9], we rewrite Eq. 8 into:

$$\begin{aligned} \mathcal{L}_i &= -\log \frac{\exp(s_{i,p})}{\sum\limits_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau) + \sum\limits_{\boldsymbol{z}_n \in \hat{P}(i)} \exp(s_{i,n}/\tau)}, \\ &= -s_{i,p} + \log \left(\sum_{\boldsymbol{z}_n \in \hat{N}(i)} \exp(s_{i,n}/\tau) + \underbrace{\sum_{\boldsymbol{z}_n \in \hat{P}(i)} \exp(s_{i,n}/\tau)}_{\text{harmfully pull up energy for target prototype}} \right), \end{aligned} \tag{10}$$

where $\hat{P}(i) = \{z_j : y_j = y_i, i \neq j\}$. As shown in Eq 10, we can see that the second term in $\log$ operation acts adversely with respect to $-s_{i,p}$ which minimizes the energy between a sample and its corresponding prototype. Hence, pairs that encourage uniformity should be, in principle, removed from the negative set. Through the lens of energy-based models, we acknowledge that the training process of CPP is equivalent to minimize the energies of data samples and the inference process can be interpreted as selecting a task-specific prompt that minimizes the energy of a data sample in a given system.

Algorithm 1: Model Training

```
Input: Frozen embedding function $f_\theta$, number of tasks $T$, training epochs $E$, training set $\{\{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t}\}_{t=1}^{T}$, prompt
  length $L_p$ and centriod number $C$.
for $t = 1, \cdots, T$ do
    Initialize: MLP $m_{\sigma^t}$, task-specific group $P^t$, $U = \varnothing$, $\hat{U} = \varnothing$
    for class $k \in \mathcal{Y}^t$ do
        Generate key prototype $\boldsymbol{\mu}_k$ with Eq.1
        $U \leftarrow U \cup \boldsymbol{\mu}_k$
    end
    for $e = 1, \cdots, E$ do
        Optimize $\sigma^t$, $P^t$ through Eq.6
    end
    for class $k \in \mathcal{Y}^t$ do
        Generate value prototype $\hat{\boldsymbol{\mu}}_k$ with Eq.1 after prepending $P^t$
        $\hat{U} \leftarrow \hat{U} \cup \hat{\boldsymbol{\mu}}_k$
    end
end
Output: $\{P^t\}_{t=1}^{T}$, $U$ and $\hat{U}$
```

Algorithm 2: Model Inference

```
Given: $f_\theta$, $U$, $\hat{U}$, $\{P^t\}_{t=1}^{T}$, query function $q(,,r)$.
Input: Test image $\boldsymbol{x}$
Initialize: $\hat{Q} = \varnothing$
$\boldsymbol{q} = f_\theta(\boldsymbol{x})[0,:]$ ;                          // Use class token as query vector.
$J = q(\boldsymbol{q}, U, r)$ ;                     // Retrieve indexes of J candidate prompts.
for $j \in J$ do
    $\boldsymbol{q}^j = f_{\theta, P^j}(\boldsymbol{x})$
    $\hat{Q} \leftarrow \hat{Q} \cup \boldsymbol{q}^j$
end
Make prediction following Eq.7
Output: label $y$
```

3. Training and Inference Algorithms

The training and inference algorithms of CPP are summarized in Algorithm 1 and Algorithm 2, respectively. Note that when multi-centroid prototypes are used, the prototype generation process is substituted by a spectrum clustering process as described in Sec. 3.4.

4. Relation with Existing Taxonomy

The field of continual learning features several classification schemas [7,15]. In this discussion, we anchor our interpretation to the taxonomy presented in [7]. At its core, our proposed method, CPP, represents a hybrid approach. When examining the task-specific prompt design, CPP echoes principles of modular architecture approaches [12, 19, 24]. These strategies typically augment learning capabilities in the face of novel tasks, ensuring task-specific refinements. Yet, CPP carves out its distinction by sidestepping the pitfalls of high memory consumption and computational demands. This efficiency is credited to the prompt-tuning mechanism we've adopted. Moreover, our method ingeniously deploys prototypes, which, when married with contrastively optimized task-specific prompts, avoid the requirement of task identities during inference. Assessing from the vantage point of these prototypes, CPP aligns with memory-based strategies [2, 3] and more pointedly with episodic memory techniques [4]. Yet, while conventional memory strategies often rely on storing and revisiting explicit exemplars, CPP innovates by distilling these into compressed prototypes, which then double as classifiers. Lastly, from a regularization perspective, task-specific prompt-tuning inherently provide strong regularization in the parameter space by freezing most parameters. Drawing from the model functionality perspectives of [1], our contrastive prototypical loss manifests as a functional space regulator. The prime constraint is to ensure minimal functional convergence between disparate class samples.

5. Real-world Applicability of CPP

The pragmatic viability of a continual learning approach is of great importance. We delve into the practicality of CPP by examining facets such as memory consumption, data privacy, and computational efficiency.
Memory footprint. It would be over-optimistic to generalize a model with the fixed learning capacity to "life-long" learning scenarios without forgetting. Thus, a judicious expansion strategy for the model is indispensable. Our proposal leverages prototypes with task-specific prompts to realize memory-efficient continual learning. Incremental memory demands in our scheme are essentially linked to storing these prototypes and task-specific prompts. For each novel class, CPP necessitates the addition of 12.2×768 parameters equating to roughly $1/15$ of a singular ImageNet image (with size $224 \times 224 \times 3$). Given such frugality, CPP holds promise for scaling, accommodating thousands of classes given a modern computational device.
Data privacy. Under privacy sensitive scenarios (*e.g.*, healthcare applications), the retention of explicit data exemplars could be fraught with risks or outright impractical. In contrast to general memory-based methods, CPP encapsulates data as fictitious prototypes, offering an enhanced privacy shield. Therefore, CPP is arguably preferable for privacy sensitive scenarios.
Computational efficiency. During the learning phase, the prompt-tuning methodology ensures that back-propagation predominantly impacts only a tiny portion of parameters. As such, CPP is more computational efficient than methods that need to update all parameters under the same architecture. Moreover, the knowledge retention in CPP is architectured around prototypes, used as reference anchors, obviating the need to forward raw exemplars. While the inference phase could impart some computational cost due to our task-specific prompt architecture, our empirical evaluations (as seen in Sec. 4.5) demonstrate that even on the most challenging benchmark, *i.e.*, split ImageNet-R, this overhead remains affordable. Crucially, the granularity of this computational overhead can be fine-tuned by modulating the centroid number, C, and the neighbor radius, r.

6. Experimental Setups

6.1. Evaluation Metrics

Let $A_{i,j}$ denotes the classification accuracy for the j-th task after training on the i-th task. The Average Accuracy (A_i) and Forgetting (F_i) aftering learning task i is then defined as follows:

$$A_i = \frac{1}{i}\sum_{j=1}^{i} A_{i,j},$$

$$F_i = \frac{1}{i-1}\sum_{j=1}^{i-1} \max_{j' \in \{1,\cdots,i-1\}} \left(A_{j',j} - A_{i,j}\right).$$

Assuming there are T tasks in total, we report the accuracy from the end session as $Acc = A_T$ following [14, 20]. We notice that some previous studies [6, 13, 25] report the macro average over all sessions, *i.e.*, $Acc = \frac{1}{T}\sum_{i=1}^{T} A_i$. To ease reference, we provide results under both protocols in Appendix 8.1.

6.2. Data Augmentations

In this section, we detail transformations used in our experiments. In general, we adopt standard data transformations used for contrastive learning, which include:

- RandomResizedCrop (size= 224, scale= $[0.8, 1.0]$)
- RandomHorizontalFlip ($p = 0.5$)
- RandomColorJitter ($p = 0.8$, brightness= 0.4, contrast= 0.4, saturation= 0.2, hue= 0.1)
- RandomGrayscale ($p = 0.2$)
- RandomGaussianBlur ($p = 0.1$, min radius= 0.1, max radius=2)

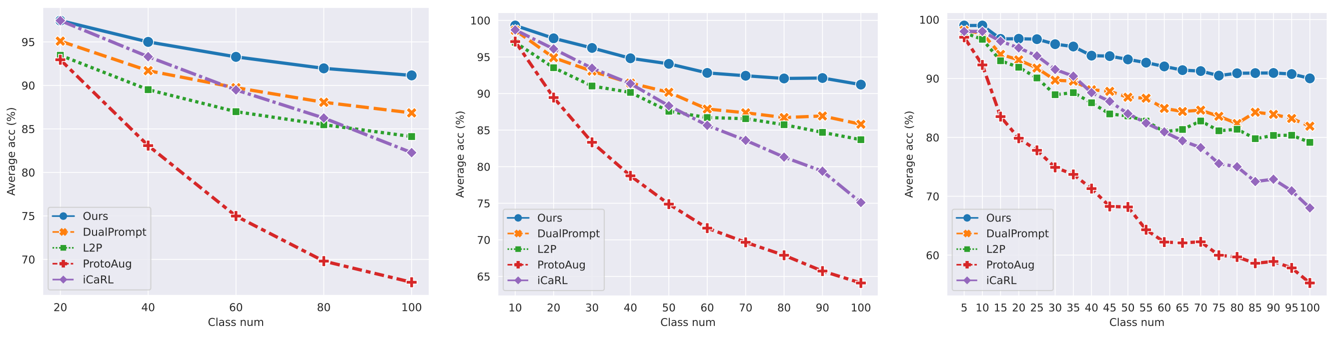

Figure 1. Comparison with state-of-the-art methods on CIFAR-100 under multiple splits. **Left:** 5-splits, **Middle:** 10-splits **Right:** 20-splits.

- RandomSolarization ($p = 0.2$)
- Normalization

Note that the normalization operation of an embedding function uses the same statistics as the ones used in its corresponding pre-training method.

7. Implementation Details for Reproducibility

- **Upper-bound**. For establishing upper-bound performance, we fine-tune the pre-trained embedding functions on the target datasets with all classes visible. The initial learning rate is set to 1×10^{-4}. All other settings follow the same configuration as training CPP to keep a fair comparison. One exception is fine-tuning MAE on ImageNet-subset where we found the transformations tailored for CPP generate inferior results. As such, we have adhered to the data transformations detailed in the original work [8].
- **iCaRL**. For iCaRL, we initiate our experimentation using the ResNet-18 model, following the original setting [18]. Given that the choice of the embedding function is orthogonal to the technical design in iCaRL, we switch the embedding function to the same pre-trained Transformer as used in CPP. The learning rate is set to 1×10^{-4} according to a grid search and the memory size is set to 2000.
- **PASS**. For PASS [25], the original paper uses 50 classes in the initial task and 5 class for each incremental task. To synchronize with our configuration, we modify the split to 10 classes per task and 10 tasks in total. Similar to reproducing iCaRL, we run experiments under both ResNet-18 and pre-trained Transformer backbones and the learning rate is set to 1×10^{-4} for Transformer. Other hyper-parameters remain the same as in the original paper.
- **DualPrompt**. DualPrompt [20] is originally built upon Transformer structures, so we only change the pre-trained weights to MAE to avoid information leakage. All other parameters adhere strictly to the guidelines of the originating paper. Specifically, we set $L_e = 20$, $L_g = 5$, $start_e = 3$, $end_e = 5$, $start_g = 1$, and $end_g = 2$. We train the model for 50 epochs with the constant learning rate of 0.005 and the Adam optimizer is used.

8. Additional Empirical Results

8.1. Results under Different Protocols

Detailed results on CIFAR-100 under different splits. Here, we provide detailed task-wise results on split CIFAR-100 under different splits. As shown in Fig. 1, CPP exhibits clear and consistent improvements over other methods and the gaps are enlarged as the length of the task sequence increases.
Results under different metric. In Table 1, we provide results under two different protocols as described in Appendix 6.1.

8.2. Comparison to Architectural Methods

In this section, we compare CPP with different architectural methods. It is non-trivial to migrate ConvNet-based methods to Transformer-based methods, so we follow the practice in DualPrompt [20] to measure the difference between a method and its

Task num	**Dataset**	**Pre-train**	**Accuracy** Avg. (↑)	Last (↑)	**Forgetting** Avg. (↓)	Last (↓)
5	split CIFAR-100	ViT	93.78±0.12	91.20±0.06	2.77±0.18	2.98±0.21
10	split CIFAR-100	ViT	94.18±0.09	91.12±0.12	2.39±0.22	3.33±0.18
20	split CIFAR-100	ViT	93.49±0.07	89.81±0.17	2.84±0.15	3.70±0.13
5	5-datasets	ViT	94.77±0.19	92.92±0.17	0.16±0.08	0.19±0.07
10	split ImageNet-Sub	MAE	95.14±0.06	93.82±0.06	0.82±0.04	1.98±0.06
10	split ImageNet-R	ViT	78.22±0.14	74.88±0.07	3.32±0.19	3.65±0.03

Table 1. Results of CPP under different protocols.

Method	**Backbone**	**Avg. Acc** (↑)	**Diff** (↓)	**Pretrained**	**Buffer size**	**Additional Parameters** MB	%
Upper-bound	ResNet18	80.41†	-	-	-	-	-
SupSup [22]		28.34±2.45‡	52.07	✗	0	3.00	6.50%
DualNet [16]		40.14±1.64‡	40.27	✗	1000	5.04	10.90%
RPSNet [17]		68.60†	11.81	✗	2000	181.00	40.4%
DynaER [23]		74.64†	5.77	✗	2000	19.80	43.80%
Upper-bound	ResNet152	88.54†	-	-	-	-	-
DynaER [23]		71.01±0.58‡	17.53	✗	2000	159.00	68.50%
Upper-bound	Customized ViT	76.12†	-	-	-	-	-
DyTox [6]		62.06±0.25†	14.06	✗	2000	0.04	0.38%
Upper-bound	ViT-B/16	93.15±0.09	-	-	-	-	-
L2P [21]		83.86±0.28‡	9.29	✓	0	1.94	0.56%
DualPrompt [20]		86.51±0.33‡	6.64	✓	0	1.90	0.55%
CPP (ours)		**91.12**±0.12	**2.03**	✓	0	**0.35**	**0.10%**

Table 2. Comparison with architecture-based methods on split CIFAR-100. `Diff` (lower is better) measures performance gap between a method and its corresponding upper-bound under the same backbone. † denotes the results reported from the original papers. ‡ represents results copied from DualPrompt [20].

Methods	**Split CIFAR-100** Avg. Acc (↑)	Forget (↓)
K-means	90.77±0.28	3.39±0.34
Spectral Clustering	**91.12**±**0.12**	**3.33** ±**0.18**

Table 3. Results under different clustering algorithms used for generating multi-centroid prototypes.

own upper-bound. As shown in Table 2, CPP largely bridges the gap between incremental learning and joint learning. Besides, CPP is also more memory efficient than alternatives.

8.3. Extra Ablation Studies

Ablation study on MLP designs. As the MLP neck is important for training task-specific prompts in our framework, we here probe relations between the MLP width (the number of hidden units), depth (the number of layers), and prompt quality. As shown in Fig. 3, either monotonously increasing the number of layers or hidden units does not bring benefits. A three-layer MLP with 2048 hidden units, which is the same as the conventional MLP neck used in self-supervised representation learning, produces the best performance. Therefore, we adopt this setting as default across our experiments.
Generation of multi-centriod prototypes. In CPP, we leverage spectral clustering to produce multi-centroid prototypes. Herein, we provide results using the commonly used k-means clustering algorithm. As shown in Table 3, spectral clustering empirically demonstrates a better performance.
Effectiveness of the query function. In the design of our nearest neighbor query function, we surmise that data samples tend to locate near to their corresponding mass centers giving an appropriate embedding function, so it is important to verify that,

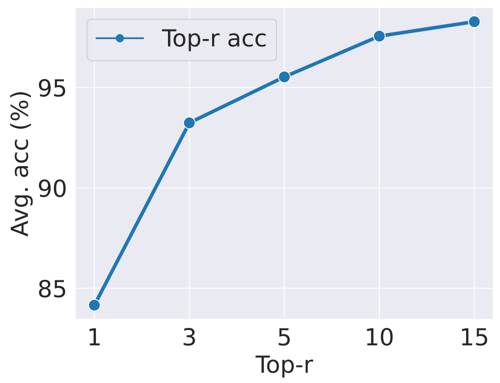

Figure 2. Top-r retrieval accuracy on split CIFAR-100 using 5-centroid prototype.

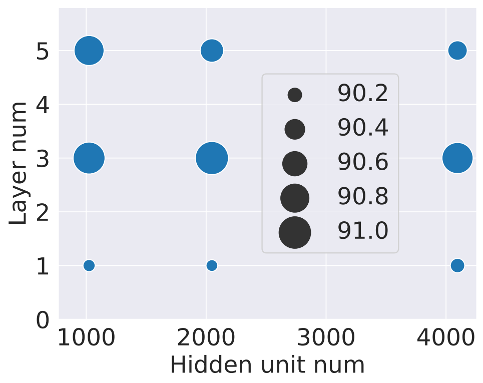

Figure 3. Ablation studies on the layer number and hidden units of MLP neck.

given r nearest neighbors, whether or not the target prompt falls in the candidate group. To this end, we demonstrate the top-r accuracy under 5-centroid prototypes in Fig. 2. We can see that the top-r accuracy increases monotonically according to the value of r and $r = 3$ works fairly well. Hence, a query vector in combination with key prototypes and a reasonable vicinity range can be effectively leveraged to retrieve the candidate task-specific prompts.

9. Detailed Visualizations and Analysis

Fig. 4 displays training samples from CIFAR-100 in the latent space. The first row shows original data samples with their corresponding class means and multi-centroid key prototypes. As shown in the figure, both class means and multi-centroid key prototypes effectively characterize the distribution of each class. In the second row, when replacing key prototypes with their corresponding value prototypes, there is a clear mismatch between the class distributions and their value prototypes. This observation manifests clear distribution shifts in the latent space after adding task-specific prompts, justifying the necessity of decoupling prototypes into the key prototypes and value prototypes. The third row exhibits value prototypes and embeddings of data samples after adding task-specific prompts. We can find that both class means and multi-centroid value prototypes fit the learned class distributions well. Nevertheless, multi-centroid value prototypes can better capture outliers, thus being more representative.

In Fig. 4, we show the efficacy of both key and value prototypes for representing train data. Here, in Fig. 5, we exhibit their performances on test data. As shown in the first row, key prototypes work fairly well in representing the original test data embeddings and thus can be safely leveraged in the coarse retrieval process. The second row further validates the necessity of decoupling key and value prototypes from the perspective of test data. As shown in the second row, there are a few classes whose key prototypes can still effectively characterize their corresponding data distributions after task-specific prompts were added, suggesting minor distribution shifts. However, most classes fail to reuse key prototypes after adding task-specific prompts. In contrast, as shown the third row, value prototypes can always be reliably used as classifiers for predictions.

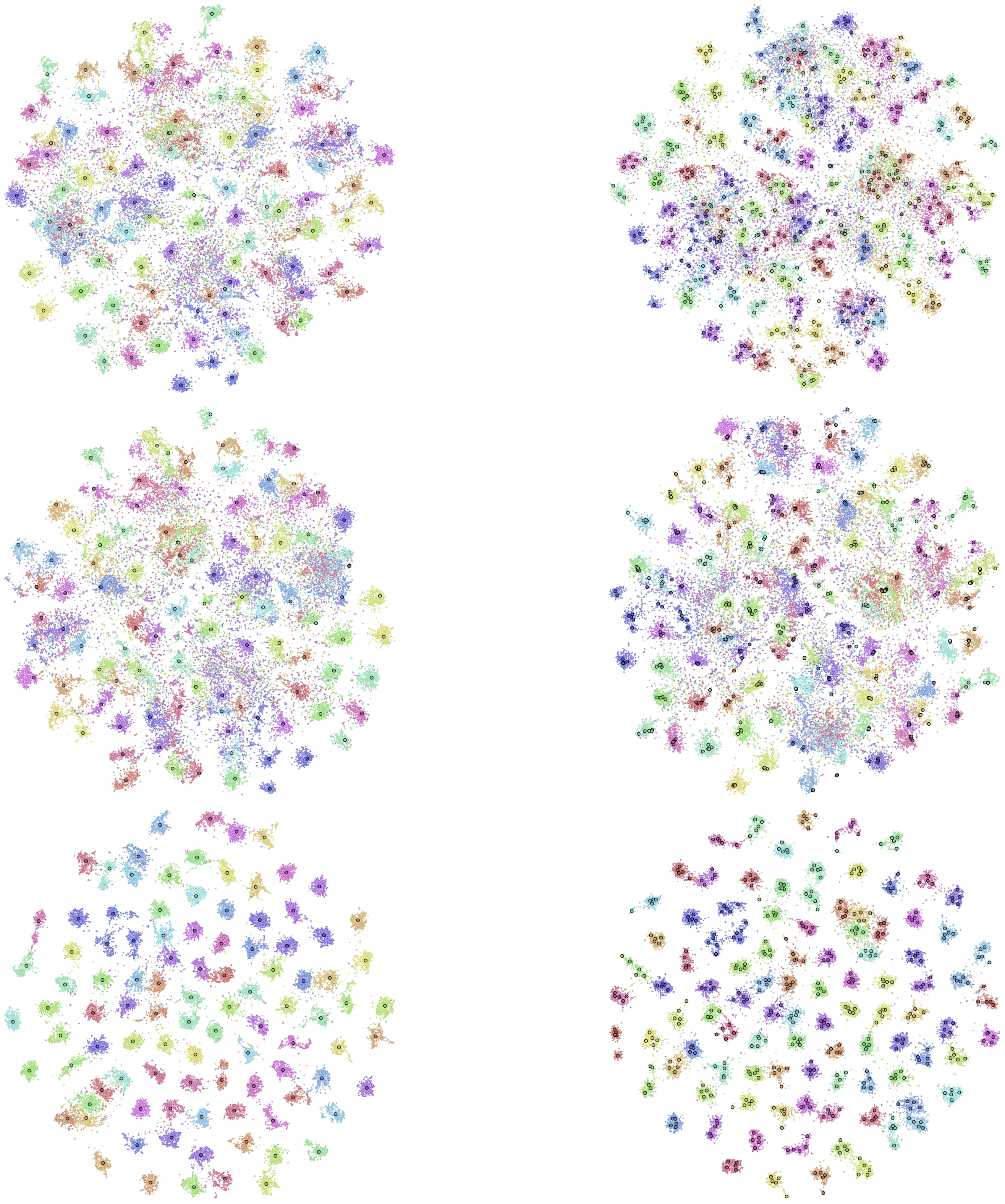

Figure 4. Visualizations for train data in CIFAR-100. **First row**: Original data embeddings with class mean (left) and multi-centriod (right) *key* prototypes. **Second row**: Original data embeddings with class mean (left) and multi-centriod (right) *value* prototypes. **Third row**: Data embeddings after adding task-specific prompts with class mean (left) and multi-centriod (right) *value* prototypes.

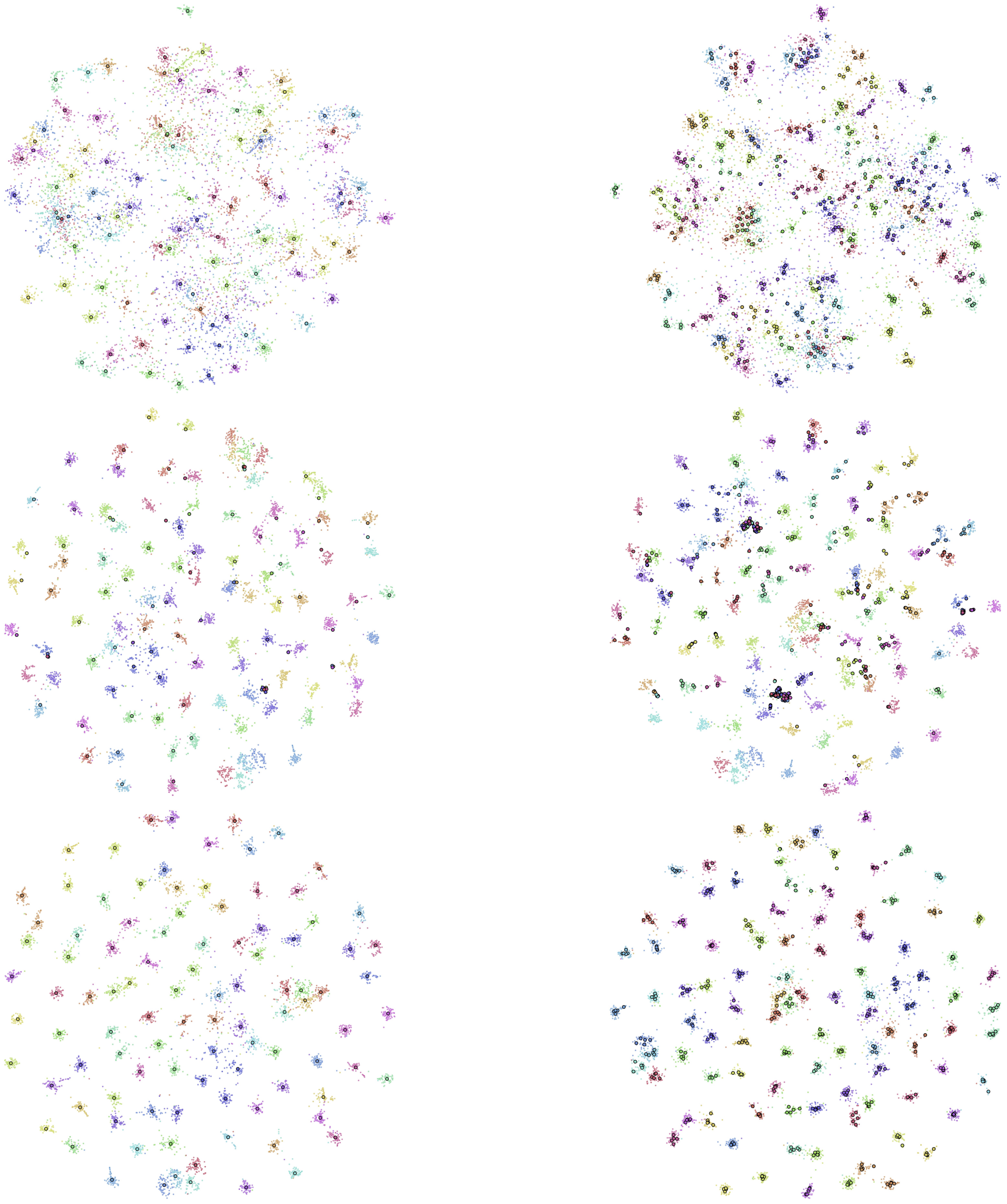

Figure 5. Visualizations for test data in CIFAR-100. **First row**: Original data embeddings with class mean (left) and multi-centriod (right) *key* prototypes. **Second row**: Original data embeddings with class mean (left) and multi-centriod (right) *value* prototypes. **Third row**: Data embeddings after adding task-specific prompts with class mean (left) and multi-centriod (right) *value* prototypes.

References

[1] Ari S. Benjamin, David Rolnick, and Konrad P. Körding. Measuring and regularizing networks in function space. *CoRR*, abs/1805.08289, 2018. 5

[2] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and SIMONE CALDERARA. Dark experience for general continual learning: a strong, simple baseline. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15920–15930. Curran Associates, Inc., 2020. 4

[3] H. Cha, J. Lee, and J. Shin. Co2l: Contrastive continual learning. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9496–9505, 2021. 4

[4] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486, 2019. 4

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020. 3

[6] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 5, 7

[7] Raia Hadsell, Dushyant Rao, Andrei Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24:1028–1040, 12 2020. 4

[8] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022. 6

[9] 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, pages 18661–18673. Curran Associates, Inc., 2020. 3

[10] Yann LeCun, Sumit Chopra, Raia Hadsell, Fu Jie Huang, and et al. A tutorial on energy-based learning. In *PREDICTING STRUCTURED DATA*. MIT Press, 2006. 2

[11] Shuang Li, Yilun Du, Gido M van de Ven, and Igor Mordatch. Energy-based models for continual learning. *arXiv preprint arXiv:2011.12216*, 2020. 3

[12] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *ICML*, 2019. 4

[13] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. 5

[14] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 5

[15] 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. 4

[16] Quang Pham, Chenghao Liu, and Steven Hoi. Dualnet: Continual learning, fast and slow. In *Advances in Neural Information Processing Systems*, volume 34, pages 16131–16144, 2021. 7

[17] Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for incremental learning. *Advances in Neural Information Processing Systems*, 2019. 7

[18] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, G. Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, 2017. 6

[19] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016. 4

[20] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. *European Conference on Computer Vision*, 2022. 5, 6, 7

[21] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022. 7

[22] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. In *Advances in Neural Information Processing Systems*, volume 33, pages 15173–15184. Curran Associates, Inc., 2020. 7

[23] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 7

[24] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018. 4

[25] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 5, 6