External Knowledge Injection for CLIP-Based Class-Incremental Learning

Da-Wei Zhou[1,2], Kai-Wen Li[1,2], Jingyi Ning[2], Han-Jia Ye[1,2(✉)], Lijun Zhang[1,2], De-Chuan Zhan[1,2]
[1] School of Artificial Intelligence, Nanjing University
[2] National Key Laboratory for Novel Software Technology, Nanjing University
{zhoudw,likw,yehj,zhanglj,zhandc}@lamda.nju.edu.cn, ningjy@nju.edu.cn

Abstract

Class-Incremental Learning (CIL) enables learning systems to continuously adapt to evolving data streams. With the advancement of pre-training, leveraging pre-trained vision-language models (e.g., CLIP) offers a promising starting point for CIL. However, CLIP makes decisions by matching visual embeddings to class names, overlooking the rich contextual information conveyed through language. For instance, the concept of "cat" can be decomposed into features like tail, fur, and face for recognition. Besides, since the model is continually updated, these detailed features are overwritten in CIL, requiring external knowledge for compensation. In this paper, we introduce ExterNal knowledGe INjEction (ENGINE) for CLIP-based CIL. To enhance knowledge transfer from outside the dataset, we propose a dual-branch injection tuning framework that encodes informative knowledge from both visual and textual modalities. The visual branch is enhanced with data augmentation to enrich the visual features, while the textual branch leverages GPT-4 to rewrite discriminative descriptors. In addition to this on-the-fly knowledge injection, we also implement post-tuning knowledge by re-ranking the prediction results during inference. With the injected knowledge, the model can better capture informative features as data evolves. Extensive experiments demonstrate its state-of-the-art performance. Code is available at: https://github.com/LAMDA-CL/ICCV25-ENGINE.

1. Introduction

Recent advancements in deep learning have significantly impacted various aspects of life [13, 21, 72]. However, real-world data often presents substantial challenges to these models, particularly streaming data that requires continual learning [2, 34]. Class-Incremental Learning (CIL) [51] has been proposed to address this challenge by enabling models to absorb new knowledge incrementally.

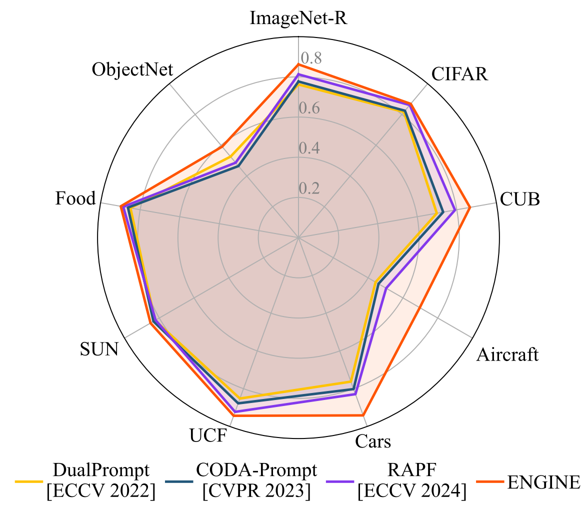


Figure 1. Last accuracy comparison on nine benchmark datasets. All methods are deployed with the same CLIP for fair comparison. ENGINE outperforms other competitors with a substantial margin.

A major issue in CIL is catastrophic forgetting [18, 19], where learning new classes leads to the loss of previously acquired knowledge. As a result, much research has focused on mitigating forgetting in deep learning [12, 42]. In contrast to traditional approaches that train models from scratch, recent progress in pre-training has shifted attention towards using pre-trained models (PTMs) [65], such as Vision Transformers[15] and CLIP [49]. With extensive training data serving as pre-learned knowledge, PTMs offer a more generalizable starting point for CIL, showing promise for real-world applications.

CLIP [49], as a pioneering pre-trained vision-language model [27, 33, 71, 75, 78], exhibits strong zero-shot performance by matching the embeddings of query images to class names. However, since visual features often contain fine-grained information [45], these detailed descriptors are neglected when using class names as matching targets. For example, the text concept of "a photo of a cat" can be further decomposed into features like "whiskers," "tail," and "fur" in order to match the visual embeddings, while these informative features are not well utilized. Furthermore, as the model is updated in CIL, the sub-features can be overwritten during the learning process, resulting in

†Correspondence to: Han-Jia Ye (yehj@lamda.nju.edu.cn)

forgetting and a mismatch between cross-modal features.

Recent studies [62, 65, 73, 86] have highlighted the limitations of using fixed template text in vision-language models, which often leads to a loss of rich contextual information. To address this, researchers have proposed learnable text inputs, such as task-specific prompts [88], which allow the model to encode additional, task-relevant information. However, these learnable inputs can become biased towards the visual features in the training set, ultimately compromising the model's generalizability [87].

In CLIP, a more detailed description in the textual space can provide a more informative matching target, enhancing prediction accuracy. For example, instead of using the generic template "a photo of a cat," directly describing its features, such as "whiskers" and "tail," can significantly improve recognition. In this way, these detailed features serve as anchors in the embedding space, and aligning to them helps reduce the risk of catastrophic forgetting.

Learning with informative descriptors offers a promising approach to enhancing CLIP's continual learning capabilities. However, two major challenges remain: **1)** acquiring informative features. Given the diversity of classes in downstream tasks, it is impractical for human experts to annotate all relevant features for recognition. **2)** mitigating forgetting when utilizing external knowledge. As CIL involves continuous data streams, the model risks forgetting previously acquired knowledge after each update. Therefore, it is crucial to efficiently extract and retain external knowledge while preventing forgetting during sequential updates.

To address these challenges, we propose ExterNal knowledGe InjEction (ENGINE) for CLIP-based CIL. To acquire informative features, we enhance knowledge transfer by incorporating external knowledge from both visual and textual modalities through a dual-branch tuning framework. The visual branch is enhanced via data augmentation to enrich visual features, while the textual branch uses GPT-4 to rewrite discriminative prompts. In this way, we can encode diverse and informative class descriptions in the continual updating process. In addition to on-the-fly knowledge injection, we also apply post-tuning by re-ranking predictions during inference. Post-tuning is done by considering the local pair-wise features, which can further calibrate the wrong predictions to boost incremental learning.

2. Related Work

Class-Incremental Learning (CIL): is a long-standing problem, aiming to absorb new knowledge without forgetting [12, 42]. Typical CIL algorithms train a model from scratch, which can be divided into several groups. Knowledge distillation-based methods aim to build the mapping target [23] between old and new models to resist forgetting [16, 36, 51], which involve logit-wise alignment [36, 51], feature-wise alignment [24, 39, 46], and group-wise alignment [14, 20, 57]. Replay-based methods aim to recover previous knowledge by saving and replaying the subset of seen classes [5, 8, 9, 38, 40, 81]. Besides, parameter regularization-based methods estimate the importance of parameters and restrict important ones not to change [3, 4, 30, 79]. Model rectification-based methods observe and rectify the inductive bias in the model, *e.g.*, the biased logits [52, 77, 80] and classifier weights [67, 80]. Model expansion-based methods [60, 70, 83] adjust the network structure to fit the characteristics of the evolving data, *e.g.*, expanding neurons [69, 74], the whole backbone [60, 61, 70, 82, 83], and lightweight modules [17].

Pre-Trained Model-Based CIL: improves the capability of CIL by starting with generalizable pre-trained models [35, 48, 56, 85]. To maintain the pre-trained knowledge, most works seek to freeze the backbone and append lightweight modules, *e.g.*, prompts [53, 63–65, 88] and adapters [10, 50, 76]. For example, L2P [65] and DualPrompt [64] design the visual prompt [28] pool to select the instance-specific prompts when learning with the pre-trained Vision Transformer. Several works build complex prompt combination/generation targets with attention mechanism [53] or generative network [29]. Other works directly utilize the pre-trained features to construct strong classifiers by matching class prototypes to the embeddings [44, 54, 84]. When the pre-trained CLIP is available as initialization, existing works seek to enhance cross-modal matching information via learning multi-modal prompts [62, 63]. MOE-Adapter [76] further extends the lightweight module selection process via mixture-of-experts [43], while PROOF [86] extends the representation ability of CLIP by appending new projection layers for new tasks. RAPF [25] learns the adapter modules for new tasks by decomposed parameter fusion to resist forgetting in the updating process.

3. Preliminaries

3.1. Class-Incremental Learning

CIL is designed to continually build a unified classifier for all seen classes of the data stream [51]. We denote training sets as $\left\{\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^B\right\}$, where each task $\mathcal{D}^b = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{n_b}$ contains n_b instances. Each training instance $\mathbf{x}_i \in \mathbb{R}^D$ belongs to class $y_i \in Y_b$, *i.e.*, the label space of task b is Y_b. We have $Y_b \cap Y_{b'} = \varnothing$ for $b \neq b'$. This paper follows the **exemplar-free** CIL setting [64, 65, 89], where the model cannot hold any historical instances in the memory. In other words, we can only access data from $\mathcal{D}^b$ for model training when learning the b-th incremental task. In CIL, the target is to build a unified classifier for all seen classes $\mathcal{Y}_b = Y_1 \cup \cdots Y_b$, *i.e.*, find a model $f(\mathbf{x}) : X \to \mathcal{Y}_b$ that minimizes the expected risk:

$$f^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \; \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_t^1\cup\cdots\mathcal{D}_t^b} \mathbb{I}\left(y \neq f(\mathbf{x})\right) . \quad (1)$$

In Eq. 1, $\mathcal{H}$ represents the hypothesis space and $\mathbb{I}(\cdot)$ is the indicator function. $\mathcal{D}_t^b$ denotes the b-th task's data distribution. In this paper, we follow [25, 76, 86] and assume that a pre-trained CLIP model (contrastive language-image pre-training) [49] is available as the initialization for $f(\mathbf{x})$. Specifically, CLIP contains the visual and textual encoder, *i.e.*, $g_i(\cdot) : \mathbb{R}^D \to \mathbb{R}^d$, $g_t(\cdot) : \mathbb{R}^{Dt} \to \mathbb{R}^d$, where the D/Dt-dimensional images and texts are projected to the same d-dimensional embedding space. During inference, it constructs the template text $\mathbf{t}_i$ using the class name, *i.e.*, "a photo of a [CLASS]$_i$", and matches the query image to text embeddings of all classes:

$$\begin{aligned} f_{y_i}(\mathbf{x}, \mathbf{t}_i) &= \frac{\exp\left(\cos\left(\mathbf{z}, \mathbf{w}_i\right)/\tau\right)}{\sum_{j=1}^{|\mathcal{Y}_b|} \exp\left(\cos\left(\mathbf{z}, \mathbf{w}_j\right)/\tau\right)} \\ &= \frac{\exp\left(\cos\left(g_i(\mathbf{x}), g_t(\mathbf{t}_i)\right)/\tau\right)}{\sum_{j=1}^{|\mathcal{Y}_b|} \exp\left(\cos\left(g_i(\mathbf{x}), g_t(\mathbf{t}_j)\right)/\tau\right)}, \end{aligned} \tag{2}$$

where $\cos(\cdot, \cdot)$ is cosine similarity and τ is temperature. $\mathbf{z} = g_i(\mathbf{x})$ and $\mathbf{w}_i = g_t(\mathbf{t}_i)$ are visual and textual features, respectively. In Eq. 2, the logit is assigned by the relative similarity of the visual embedding to textual embedding.

3.2. Baselines in Class-Incremental Learning

To overcome forgetting, there are two typical solutions for CIL with pre-trained models by learning prompts [28, 88].

Learning Visual Prompts: As fully-finetuning will harm the generalizability of the PTM, several works [28, 53, 64, 65] propose to freeze the pre-trained visual encoder and append visual prompts for subsequent tasks. They design a prompt pool to select instance-specific prompts, with which the visual embedding is represented as:

$$\mathbf{z} = \bar{g}_i\left(\mathbf{x}, \mathcal{P}\right), \tag{3}$$

where $\bar{g}_i$ is the frozen image encoder, $\mathcal{P}$ is the prompt pool for prompt selection. By replacing $g_i(\mathbf{x})$ in Eq. 2 with Eq. 3, the model efficiently encodes downstream information into the visual prompts and tackles forgetting.

Learning Textual Prompts: Since CLIP contains two branches for the visual and textual encoder, there are also works learning textual prompts instead of using the template text. Specifically, the textual input is formulated into a set of learnable prompts in CoOp [88], *i.e.*, $\tilde{\mathbf{t}}_i = [\mathrm{V}]_1[\mathrm{V}]_2 \cdots [\mathrm{V}]_M[\mathrm{CLASS}]_i$. In this way, the textual embedding is denoted as:

$$\mathbf{w}_i = \bar{g}_t\left(\tilde{\mathbf{t}}_i\right), \tag{4}$$

where $\bar{g}_t$ is the frozen text encoder. Similarly, we can replace the textual embedding in Eq. 2 via Eq. 4 to encode task information into these learnable prompts.

Discussions: Eq. 3 and Eq. 4 focus on different aspects to adjust the pre-trained CLIP into downstream tasks. However, when adapting the model with only visual prompts, the problem occurs since the model has to decompose the class name into fine-grained concepts to match the visual features. The sequential updating process will cause the degradation of such decomposition ability, resulting in the mismatch between visual and textual features. By contrast, if we utilize textual prompts, the model is only focused on the specific training instance and lacks holistic information about the specific class. When the testing data contains the image of the same class with different distributions, the model shall be confused due to the prompt-level overfitting. Hence, the ideal text input should be general and holistic to match the visual features and convey all informative features that the target class possesses.

4. ENGINE: External Knowledge Injection

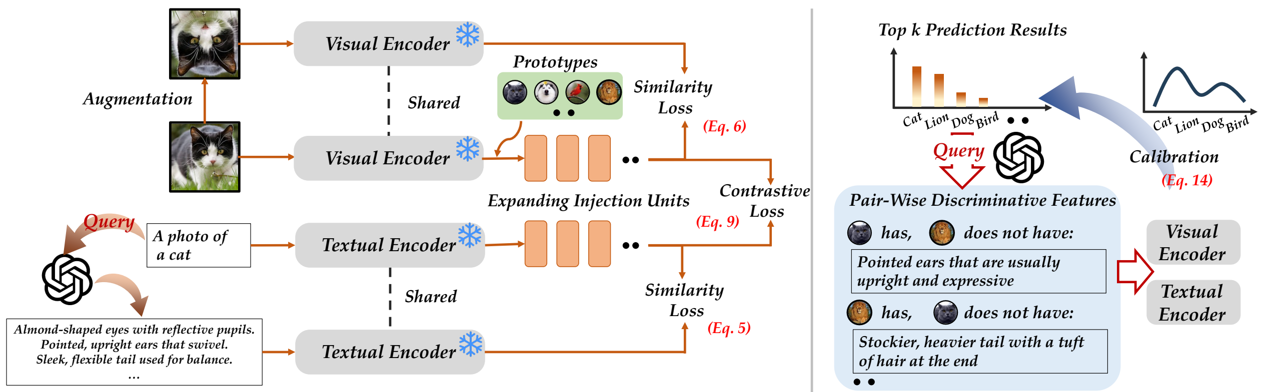
Noting that prompt-based methods are highly limited by the training instances, we need to extract comprehensive information for the model to adapt to downstream tasks while preventing forgetting. To encode *holistic information* into the model, we seek help from large language models like GPT-4 [1] to provide the general descriptions instead of using the template features. Apart from using detailed descriptions in the training process, we also make full use of them during inference by seeking pair-wise discriminative features. On the other hand, as external knowledge is continually injected into the model, we also design the injection unit expansion and prototype replay strategy. In the following sections, we first introduce how to inject external knowledge during model training and then introduce how to utilize it during inference.

4.1. On-the-fly Knowledge Injection

Since the target is to encode general information into the CLIP during incremental learning, we first design the model training pipeline that can enhance recognition. For example, a "cat" contains visual features like "soft, short fur", "long, thin tail", and "round face with large eyes". It would be better to match the cat image to these visual features than "a photo of a cat", since those features contain more discriminative and fine-grained information. Correspondingly, to enhance CLIP's knowledge of general class description, we seek help from GPT-4 [1] to provide *discriminative visual features*:

> **Q**: What are unique visual features of [CLASS]$_i$ in a photo? Focus on the key visual features.
> **A**: **1.** Long, thin tail that aids in balance. **2.**$\cdots$

In this way, we can get a set of visual descriptions for each class, denoted as $\mathbf{d}_i$. A naive solution is to replace the template text $\mathbf{t}_i$ into $\mathbf{d}_i$ during training, while we propose a more intuitive way to inject the knowledge into the model. Specifically, we design a knowledge injection unit u_t to dynamically encode these textual descriptors. We implement



(a) On-the-fly Knowledge Injection *(b) Post-tuning Knowledge Injection*

Figure 2. Illustration of ENGINE. **Left**: On-the-fly knowledge injection. We utilize GPT-4 to provide visual features of each class, and append injection units to encode external knowledge into the model. We also adopt random augmentation to enhance image diversity using the same network structure. To overcome forgetting, we append the prototypes of previous classes in each learning stage. **Right**: Post-tuning knowledge injection. We extract top-k predictions and generate pair-wise discriminative features locally. The input image is forwarded with every textual feature to further refine the prediction.

u_t with a linear layer, *i.e.*, $u_t(\cdot):\mathbb{R}^d\to\mathbb{R}^d$, and append it after the textual encoder. We have the similarity loss:

$$\mathcal{L}_t = -\mathrm{Sim}\left(u_t\left(\bar{g}_t\left(\mathbf{t}_i\right)\right), \bar{g}_t\left(\mathbf{d}_i\right)\right), \tag{5}$$

where we use cosine similarity to calculate $\mathrm{Sim}(\cdot,\cdot)$. In Eq. 5, we maximize the similarity between $u_t\left(\bar{g}_t\left(\mathbf{t}_i\right)\right)$ and $\bar{g}_t\left(\mathbf{d}_i\right)$, *i.e.*, making the adapted textual feature similar to external descriptions. Since the textual encoder $\bar{g}_t$ is frozen, we can inject this external information into the injection unit u_t. In this way, the extracted textual features will contain detailed descriptions even using the template text, *e.g.*, highlighting the features of "long thin tail" and "round face" when the input is "a photo of a cat". In the implementation, since GPT-4 will output a set of descriptions, we randomly choose one of the descriptions $\mathbf{d}_i$ to calculate Eq. 5 in each iteration.

Learning external visual knowledge: Similar to the textual branch, we can also inject visual information into the model during training. Given that we cannot fetch external data in the learning process, we utilize random data augmentation to adjust the input images, and denote the augmented input as $\mathcal{A}(\mathbf{x})$. Correspondingly, we can learn the image knowledge injection unit $u_i(\cdot):\mathbb{R}^d\to\mathbb{R}^d$, and maximize the similarity between augmented and vanilla input:

$$\mathcal{L}_i = -\mathrm{Sim}\left(u_i\left(\bar{g}_i\left(\mathbf{x}\right)\right), \bar{g}_i\left(\mathcal{A}(\mathbf{x})\right)\right). \tag{6}$$

Similar to Eq. 5, Eq. 6 freezes the visual encoder $\bar{g}_i$, and only optimizes the image injection unit u_i. In the matching process, the model maximizes the similarity of the adapted visual feature to the augmented view, which helps the model extract diverse visual information.

Learning task-specific injection unit: In CIL, new tasks will emerge as data evolves, and sequentially tuning a single injection unit will result in forgetting previous knowledge. Consequently, we conduct the injection process for each incremental task to avoid forgetting. Specifically, we initialize a new injection unit u_t^b, u_i^b when learning the b-th task, and freeze all previous injection units when learning the current task. With the set of visual and textual units, we utilize the summation of them for the final representation:

$$G_i(\mathbf{x}) = \textstyle\sum_{p=1}^{b-1}\bar{u}_i^p\left(\bar{g}_i\left(\mathbf{x}\right)\right) + u_i^b\left(\bar{g}_i\left(\mathbf{x}\right)\right) \tag{7}$$

$$G_t(\mathbf{t}_i) = \textstyle\sum_{p=1}^{b-1}\bar{u}_t^p\left(\bar{g}_t\left(\mathbf{t}_i\right)\right) + u_t^b\left(\bar{g}_t\left(\mathbf{t}_i\right)\right) \tag{8}$$

In the original CLIP, we have the cross-modal contrastive loss to match the image to the corresponding text description. Correspondingly, we utilize the injected visual and textual features to calculate the contrastive loss:

$$\mathcal{L}_c = \ell(f_{\mathrm{inj}}(\mathbf{x}), y) \quad \text{where} \tag{9}$$

$$f_{\mathrm{inj},y_i}(\mathbf{x}) = \frac{\exp\left(\cos\left(G_i(\mathbf{x}), G_t(\mathbf{t}_i)\right)/\tau\right)}{\sum_{j=1}^{|\mathcal{Y}_b|}\exp\left(\cos\left(G_i(\mathbf{x}), G_t(\mathbf{t}_j)\right)/\tau\right)}. \tag{10}$$

In Eq. 9, we utilize the injected visual and textual features to calculate the contrastive loss, aiming to increase the similarity of pair-wise injected features.

Preventing forgetting of injection unit In the updating process, we combine Eq. 5, Eq. 6, and Eq. 9 to update the model. Since we cannot hold historical instances for replay, the model may still forget previous concepts due to the sequential updating. To this end, utilizing the substitution of previous concepts can create a calibration across all seen classes. Observing that the visual backbone g_i is not updating throughout the learning process, we generate a subset of previous classes using visual prototypes. Specifically, during the learning process, we can calculate the average visual embedding of each class as:

$$\mathbf{p}_k = \frac{\sum_{j=1}^{|\mathcal{D}^b|}\mathbb{I}(y_j=k)g_i(\mathbf{x}_j)}{\sum_{j=1}^{|\mathcal{D}^b|}\mathbb{I}(y_j=k)}, \tag{11}$$

which stands for the most representative features of the corresponding class. We can treat them as the visual embedding of previous classes when learning new ones, *i.e.*, we construct the auxiliary training set $\mathbf{P} = \{(\mathbf{p}_1, 1), (\mathbf{p}_2, 2), \cdots, (\mathbf{p}_{|\mathcal{Y}_b|}, |\mathcal{Y}_b|)\}$, and optimize:

$$\min_{\{u_t^b, u_i^b\}} \sum_{(\mathbf{x},y)\in\mathcal{D}^b\cup\mathbf{P}} \mathcal{L}_t + \mathcal{L}_i + \mathcal{L}_c \,. \tag{12}$$

Specifically, the prototype set $\mathbf{P}$ is only adopted in calculating $\mathcal{L}_c$. To enhance diversity across training stages, we also add Gaussian noise on the prototype embeddings, *i.e.*, $\mathbf{p}' = \mathbf{p} + \epsilon$, $\epsilon \in \mathcal{N}(0, \alpha^2\mathbf{I})$, and utilize $\mathbf{p}'$ in Eq. 12. In this way, we can revisit previous knowledge to prevent forgetting when learning new injection units. Additionally, we also consider matching the visual features to these prototypes to further assist recognition [66, 86].

Summary of on-the-fly injection: We visualize the training stage knowledge injection in Figure 2 (left). ENGINE learns two sets of knowledge injection units, aiming to encode the extra knowledge into the unit module. We utilize GPT-4 to extract informative class descriptors, and utilize similarity loss to encode them into the injection units. Similarly, data augmentation is utilized to provide visual information. Hence, the model can extract more informative features as it sees more tasks. To prevent forgetting, we also design the learning objective considering class prototypes. Since injection units are implemented with a single layer, the appended parameter size ($d \times d$) per unit is negligible for CLIP. Besides, since injection units are linear layers, they can be further re-parameterized by aggregating weights into a single one, *i.e.*, $\sum_p u_i^p$ and $\sum_p u_t^p$.

4.2. Post-tuning Knowledge Injection

We have introduced the way to inject external knowledge during training in Section 4.1. Apart from the model training stage, we can also utilize external knowledge to calibrate the prediction results during inference, and we call this step "post-tuning knowledge injection". For example, after the model makes inference by matching visual and textual features, we can treat the current prediction results as a *preliminary result*, which needs further rectification. For the top-k prediction results in this preliminary result, we treat them as *competitors* to the ground truth that could result in confusion. Hence, we need to construct a further prediction considering the image and all local competitors, which can focus more on the top-k classes to refine the prediction.

Refining the results: After learning the injection units, we can get the prediction results using the injected features for more comprehensive features. However, the model may still make wrong predictions, *e.g.*, classifying a "cat" into a similar class "lion" since they have similar visual features. To refine the prediction results, we design an extra consolidation process using external knowledge. Denoting the prediction results of injected feature matching in Eq. 10 as $f_{\text{inj}}(\mathbf{x})$, we first extract the top-k predictions as the competitor candidates, *i.e.*, $\{f_{\text{inj},i_1}(\mathbf{x}), f_{\text{inj},i_2}(\mathbf{x}), \ldots, f_{\text{inj},i_k}(\mathbf{x})\} = \text{Top}_k\left(f_{\text{inj}}(\mathbf{x})\right)$, where $\text{Top}_k\left(f_{\text{inj}}(\mathbf{x})\right)$ indicates selecting the top-k largest value in $f_{\text{inj}}(\mathbf{x})$. Their labels are denoted as:

$$\{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\} = \{y_i \mid f_{\text{inj},i}(\mathbf{x}) \in \text{Top}_k\left(f_{\text{inj}}\left(\mathbf{x}\right)\right)\} \,. \tag{13}$$

In Eq. 13, the label set $\{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\}$ corresponds to the possible confusing labels for the input image. To further refine the predictions, we aim for pair-wise disambiguation among this label set. We again utilize GPT-4 to generate *pair-wise* discriminative features:

> **Q**: What are unique visual features of $[\text{CLASS}]_i$ compared to $[\text{CLASS}]_j$ in a photo? Focus on their key visual differences.
> **A**: $[\text{CLASS}]_i$: **1.** Long, thin tail that aids in balance $\cdots$ $[\text{CLASS}]_j$: **1.** Stockier, heavier tail with a tuft of hair at the end $\cdots$

With this process, GPT-4 provides the most discriminative features that can differentiate two classes, *e.g.*, the differences in "tail" to differentiate "cat" from "lion" and vice versa. Given the top-k classes, we can generate their pair-wise descriptors as $\mathbf{D} = [\mathbf{d}_{ij}]_{i,j=1}^{k}$, where $\mathbf{d}_{ij}$ denotes the descriptions to differ class i from class j. We feed these fine-grained descriptions as the textual input, and match the query embedding to them:

$$f_{\text{pt},y_i}(\mathbf{x}) = \frac{1}{k-1}\sum_{j=1}^{j\neq i} f_{y_i}(\mathbf{x}, \mathbf{d}_{ij}) \,, \tag{14}$$

where we utilize zero-shot CLIP in Eq. 2 to match the query image to these pair-wise discriminative features, and average the similarities as the post-tuning logits.

Effect of post-tuning knowledge injection: We visualize the post-tuning knowledge injection in Figure 2 (right). To discriminate similar concepts and refine the wrong predictions, we first extract the top-k predictions of the model, and utilize GPT-4 to generate pair-wise discriminative features locally. Eq. 14 provides the fine-grained local prediction among these top-k classes, which can further help refine the results as data evolves. We adopt the zero-shot CLIP for Eq. 14, which does not require incremental updates.

4.3. Summary of ENGINE

In ENGINE, we utilize external knowledge to help continual learning in two aspects, *i.e.*, on-the-fly injection and post-tuning injection. With the help of GPT-4, we can extract more informative descriptors suitable for the incremental learning classes, and inject the external knowledge into the knowledge units. Besides, we also extract local descriptors to discriminate pair-wise classes in the post-tuning stage to help refine the predictions. During training, we update the model via Eq. 12 to inject knowledge into the knowledge units. During inference, we aggregate the following outputs as the prediction:

$$f(\mathbf{x}) = f_{\text{inj}}(\mathbf{x}) + f_{\text{pt}}(\mathbf{x}) \,. \tag{15}$$

Table 1. Average and last performance comparison of different methods. The best performance is shown in bold. **All methods are initialized with the same pre-trained CLIP without exemplars for a fair comparison.**

Method	Aircraft				CIFAR100				Cars			
	B0 Inc10		B50 Inc10		B0 Inc10		B50 Inc10		B0 Inc10		B50 Inc10	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	3.16	0.96	1.72	1.05	7.84	4.44	5.30	2.46	3.14	1.10	1.54	1.13
CoOp [88]	14.54	7.14	13.05	7.77	47.00	24.24	41.23	24.12	36.46	21.65	37.40	20.87
SimpleCIL [84]	59.24	48.09	53.05	48.09	84.15	76.63	80.20	76.63	92.04	86.85	88.96	86.85
ZS-CLIP [49]	26.66	17.22	21.70	17.22	81.81	71.38	76.49	71.38	82.60	76.37	78.32	76.37
L2P [65]	47.19	28.29	44.07	32.13	82.74	73.03	81.14	73.61	76.63	61.82	76.37	65.64
DualPrompt [64]	44.30	25.83	46.07	33.57	81.63	72.44	80.12	72.57	76.26	62.94	76.88	67.55
CODA-Prompt [53]	45.98	27.69	45.14	32.28	82.43	73.43	78.69	71.58	80.21	66.47	75.06	64.19
RAPF [25]	50.38	23.61	40.47	25.44	86.14	78.04	82.17	77.93	82.89	62.85	75.87	63.19
ENGINE	**69.69**	**58.69**	**64.38**	**59.02**	**86.92**	**79.22**	**83.15**	**79.47**	**94.14**	**90.08**	**91.61**	**90.03**

Method	ImageNet-R				CUB				UCF			
	B0 Inc20		B100 Inc20		B0 Inc20		B100 Inc20		B0 Inc10		B50 Inc10	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	1.37	0.43	1.01	0.88	2.06	0.64	0.56	0.47	4.51	1.59	1.21	0.80
CoOp [88]	60.73	37.52	54.20	39.77	27.61	8.57	24.03	10.14	47.85	33.46	42.02	24.74
SimpleCIL [84]	81.06	74.48	76.84	74.48	83.81	77.52	79.75	77.52	90.44	85.68	88.12	85.68
ZS-CLIP [49]	83.37	77.17	79.57	77.17	74.38	63.06	67.96	63.06	75.50	67.64	71.44	67.64
L2P [65]	75.97	66.52	72.82	66.77	70.87	57.93	75.64	66.12	86.34	76.43	83.95	76.62
DualPrompt [64]	76.21	66.65	73.22	67.58	69.89	57.46	74.40	64.84	85.21	75.82	84.31	76.35
CODA-Prompt [53]	77.69	68.95	73.71	68.05	73.12	62.98	73.95	62.21	87.76	80.14	83.04	75.03
RAPF [25]	81.26	70.48	76.10	70.23	79.09	62.77	72.82	62.93	92.28	80.33	90.31	81.55
ENGINE	**86.22**	**80.37**	**83.63**	**80.98**	**86.65**	**80.20**	**82.59**	**79.30**	**94.35**	**90.03**	**92.51**	**89.58**

Method	SUN				Food				ObjectNet			
	B0 Inc30		B150 Inc30		B0 Inc10		B50 Inc10		B0 Inc20		B100 Inc20	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	4.51	1.59	0.78	0.72	3.49	1.71	2.14	1.52	1.34	0.47	0.69	0.54
CoOp [88]	45.93	23.11	39.33	24.89	36.01	14.18	33.13	18.67	21.24	6.29	16.21	6.82
SimpleCIL [84]	82.13	75.58	78.62	75.58	87.89	81.65	84.73	81.65	52.06	40.13	45.11	40.13
ZS-CLIP [49]	79.42	72.11	74.95	72.11	87.86	81.92	84.75	81.92	38.43	26.43	31.12	26.43
L2P [65]	82.82	74.54	79.57	73.10	85.66	77.33	80.42	73.13	51.40	39.39	48.91	42.83
DualPrompt [64]	82.46	74.40	79.37	73.02	84.92	77.29	80.00	72.75	52.62	40.72	49.08	42.92
CODA-Prompt [53]	83.34	75.71	80.38	74.17	86.18	78.78	80.98	74.13	46.49	34.13	40.57	34.13
RAPF [25]	82.13	72.47	78.04	73.10	88.57	81.15	85.53	81.17	48.67	27.43	39.28	28.73
ENGINE	**85.04**	**78.54**	**81.57**	**78.45**	**89.81**	**83.89**	**86.89**	**83.94**	**59.11**	**45.19**	**51.32**	**44.99**

Table 2. Comparison to traditional exemplar-based CIL methods. ENGINE does not use any exemplars.

Method	Exemplars	Aircraft B0 Inc10		SUN B0 Inc30	
		$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
iCaRL [51]	20 / class	53.60	43.98	78.56	67.30
MEMO [83]	20 / class	42.24	25.41	81.48	73.45
PROOF [86]	20 / class	61.00	53.59	83.57	77.28
ENGINE	**0**	**69.69**	**58.69**	**85.04**	**78.54**

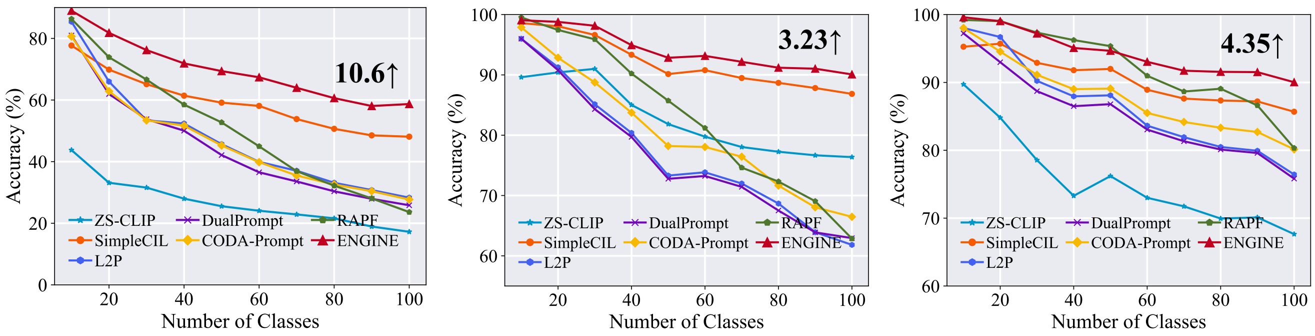
5. Experiments

In this section, we conduct extensive experiments on nine benchmark datasets and compare ENGINE to state-of-the-art methods. We visualize the incremental learning curve and provide ablation studies on prompt engineering and parameter sensitivity to test the model's robustness. Besides, we also visualize the learned embeddings and logit changes after injecting external knowledge to show its effectiveness. We provide more results in the supplementary.

5.1. Implementation Details

Dataset: We follow [65, 86, 88] to evaluate the performance on nine benchmark datasets that have domain gap to CLIP's pre-training dataset, *i.e.*, CIFAR100 [32], CUB200 [59], ObjectNet [6], ImageNet-R [22], FGVCAircraft [41], StanfordCars [31], Food101 [7], SUN397 [68] and UCF101 [55]. Following [86], we utilize the sampled 100 classes from CIFAR100, Aircraft, Cars, Food, UCF, 200 classes from CUB200, ObjectNet, ImageNet-R, and 300 classes from SUN to ease the data split. More details are reported in the supplementary.

Dataset split: Following [51, 65], we use 'B-m Inc-n' to split the classes in CIL. m indicates the number of classes in the first stage, and n represents that of every following stage. We follow [51] to randomly shuffle the class order with random seed 1993 for all compared methods, and keep this same for every method.

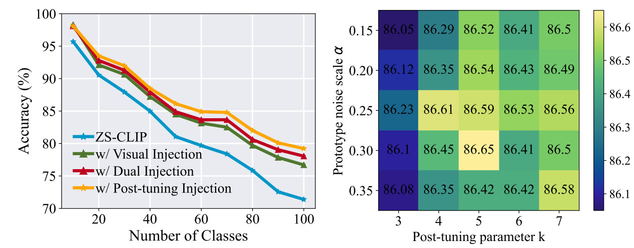


(a) Aircraft Base0 Inc10 (b) Cars Base0 Inc10 (c) UCF Base0 Inc10

Figure 3. Incremental performance of different methods. We report the performance gap after the last incremental stage of ENGINE and the runner-up method at the end of the line. All methods utilize the same CLIP pre-trained weight. More figures are shown in supplementary.

Table 3. Results when all methods use the same textual description generated by GPT-4 and data augmentation.

Method	Template	ImageNet-R B0 Inc20 $\bar{\mathcal{A}}$	ImageNet-R B0 Inc20 $\mathcal{A}_B$	CIFAR B0 Inc10 $\bar{\mathcal{A}}$	CIFAR B0 Inc10 $\mathcal{A}_B$
ZS-CLIP	GPT Generated	83.68	77.59	82.26	71.70
RAPF	GPT Generated	81.89	71.04	86.49	78.52
ENGINE	GPT Generated	**86.22**	**80.37**	**86.92**	**79.22**



(a) Ablation study (b) Parameter sensitivity

Figure 4. Ablation study and parameter sensitivity.

Comparison methods: We first compare to SOTA pre-trained model-based CIL algorithms, *e.g.*, L2P [65], DualPrompt [64], CODA-Prompt [53], SimpleCIL [84]. Besides, we also compare to SOTA CLIP-based CIL algorithms, *e.g.*, CoOp [88], PROOF [86], RAPF [25]. The baseline that finetunes CLIP for incremental tasks is denoted as Finetune. **All methods are deployed with the same CLIP as initialization.**

Training details: The experiments are deployed with NVIDIA 4090 using PyTorch [47]. We follow [25, 86] to consider CLIP with ViT-B/16 for all compared methods for *fair comparison*. For vision-based methods that cannot utilize the textual prompt (*e.g.*, L2P, DualPrompt, CODA-Prompt), we utilize CLIP's visual branch as their initialization. We report the results using LAION-400M pre-trained CLIP [26] in the main paper, and report results using OpenAI [49] in the supplementary. In ENGINE, we use SGD optimizer with a batch size of 64 to optimize the model for 10 epochs. The learning rate decays from 0.05 with cosine annealing. We set the prototype noise ratio α to 0.25 and k in post-tuning to 5. We use OpenAI GPT-4o mini [1] to generate the prompts for textual knowledge and AutoAugment [11] to augment the visual features. We explore other LLMs to extract external knowledge in the supplementary. The source code will be publicly available upon acceptance.

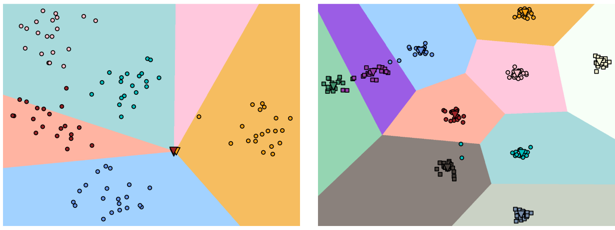
Evaluation metric: Following [51, 86], we use $\mathcal{A}_b$ to represent the model's accuracy after the b-th stage. Specifically, we adopt $\mathcal{A}_B$ (the performance after the last stage) and $\bar{\mathcal{A}} = \frac{1}{B}\sum_{b=1}^{B}\mathcal{A}_b$ (average performance along incremental stages) as measurements.

5.2. Benchmark Comparison

We first compare ENGINE to other state-of-the-art methods on the benchmark datasets and report the results in Table 1 and Figure 3. Specifically, we find ENGINE outperforms current SOTA by 3%∼10% on these benchmark datasets. We find the baseline method Finetune has the worst performance, indicating that the decomposition ability to tackle the class template into fine-grained features is totally forgotten. For visual prompt-based methods like L2P, DualPrompt, and CODA-Prompt, the performance is limited due to the inability to utilize textual information. For the textual prompt tuning method CoOp, we find that it achieves poor performance due to the forgetting of textual prompts. For other CLIP-based methods like RAPF, our method still shows substantial improvement, verifying its strong continual learning ability without forgetting.

Apart from these non-exemplar-based methods, we also consider typical exemplar-based CIL methods for comparison. We report the comparison results in Table 2, where all methods are initialized with the same CLIP. As we can infer from the table, ENGINE still outperforms them even without using exemplars.

Finally, what if all CLIP-based methods utilize the same external knowledge? Correspondingly, we also consider using the same generated prompt and data augmentation for ZS-CLIP and RAPF, and report the results in Table 3. In this table, we replace the template text of all methods using the same text in Section 4.1 and adopt the same data augmentation for image input. We find ENGINE still outperforms them even using the same prompt, indicating that ENGINE can better utilize the external information for incremental learning.



(a) ZS-CLIP, single stage (b) ENGINE, two stages

Figure 5. t-SNE [58] visualizations of zero-shot CLIP and ENGINE. ENGINE aligns the cross-modal features without forgetting. We represent the visual features of the first task with dots and classes of the second task with squares. The textual features are represented by triangles, and we utilize the shadow area to denote the decision boundary.

5.3. Further Analysis

Ablation study: We first conduct ablations to investigate the effectiveness of each component in ENGINE. We report the incremental performance on CIFAR100 B0 Inc10 in Figure 4a. We can infer that '**ZS-CLIP**' has the worst performance since the downstream distribution is different from the pre-training stage, and we view it as the baseline. Correspondingly, we first equip the model with visual knowledge injection (Eq. 6), and denote the model as '**w/ Visual Injection**'. It shows that the performance drastically improves as the model has been adapted to extract more informative visual features. Similarly, we then append the textual information using Eq. 5, denoted as '**w/ Dual Injection**.' As shown in the figure, using external knowledge for both modalities can further improve the performance than visual knowledge only. Finally, we equip the model with post-tuning knowledge injection, and utilize Eq. 15 for inference (*i.e.*, '**w/ Post-tuning Injection**'). This corresponds to the full version of ENGINE, and we find it achieves the best performance. Ablations verify that every component in ENGINE boosts the CIL performance.

Parameter robustness: There are two hyperparameters in ENGINE, *i.e.*, the prototype noise scale α in Section 4.1 and post-tuning candidate k in Eq. 14. We conduct experiments on CUB B0 Inc20 to investigate the robustness by changing these parameters. Specifically, we choose α among $\{0.15, 0.2, 0.25, 0.3, 0.35\}$, and k among $\{3, 4, 5, 6, 7\}$. We report the last performance in Figure 4b. As shown in the figure, the performance is robust with the change of these parameters, and we suggest $\alpha = 0.25, k = 5$ as default.

Visualizations: In this section, we utilize t-SNE [58] to visualize the cross-modal features learned by ENGINE on CIFAR100 B0 Inc5. We represent the visual features of the first task with dots (◯) and classes of the second task with squares (□). The textual features are represented by triangles (△), and we utilize the shadow area to denote the decision boundary divided by the textual features. Specifically, we first visualize the instances of the first stage in zero-shot CLIP's embedding space in Figure 5a. The results indicate that pre-trained CLIP has difficulty in aligning the visual and textual features since they are situated in two distinct areas in the embedding space [37]. While for Figure 5b, we visualize the embedding of ENGINE across two incremental stages. As we can infer from the figure, ENGINE has two advantages: **1)** The visual and textual features are aligned in the same cluster, and **2)** The model has not forgotten previous classes when learning new ones.

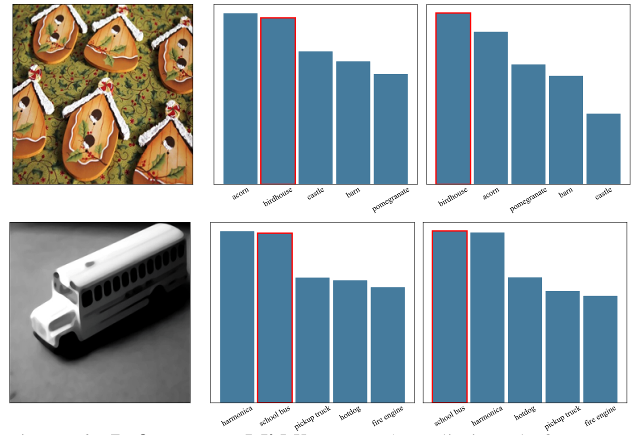


Figure 6. **Left:** Input. **Middle:** Top-5 predictions before post-tuning knowledge injection. **Right:** Top-5 predictions after post-tuning. More cases are shown in the supplementary.

We also visualize the model output before and after post-tuning knowledge injection on ImageNet-R in Figure 6. We utilize bars with red edges to denote the ground-truth class. As we can infer from the figure, post-tuning knowledge injection can help the model refine the predictions by generating local comparisons. For the case on top of Figure 6, the model can focus on the differences between "acorn" and "birdhouse" to make the final prediction.

6. Conclusion

Class-incremental learning is essential to real-world applications. This paper aims to enable CLIP with CIL ability by injecting external knowledge. Specifically, we consider on-the-fly knowledge injection by learning dual-branch injection units. We augment images for the visual branch and generate detailed feature descriptions with GPT-4 for the textual branch. By maximizing the similarity between external information and the injection module, we encode informative features into the pre-trained CLIP. Besides, we also consider post-tuning knowledge injection with local class information to calibrate the predictions. Extensive experiments verify ENGINE's effectiveness.

Limitations and future works: This paper relies on GPT-4 to provide expert knowledge for specific tasks, which may be impossible for the specific cases where GPT-4 cannot generalize. Future works include designing other formats of external knowledge, *e.g.*, knowledge graphs.

Acknowledgments

This work is partially supported by NSFC (U23A20382, 62476123, 62376118, 62250069), Fundamental Research Funds for the Central Universities (2024300373,14380021), CCF-Tencent Rhino-Bird Open Research Fund (RAGR20240101), Postdoctoral Fellowship Program of CPSF (GZB20250389), China Postdoctoral Science Foundation (2025M771535), and Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 3, 7

[2] Charu C Aggarwal. A survey of stream clustering algorithms. In *Data Clustering*, pages 231–258. Chapman and Hall/CRC, 2018. 1

[3] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154, 2018. 2

[4] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *CVPR*, pages 11254–11263, 2019. 2

[5] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *NeurIPS*, pages 11816–11825, 2019. 2

[6] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *NeurIPS*, pages 9448–9458, 2019. 6

[7] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, pages 446–461, 2014. 6

[8] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, pages 532–547, 2018. 2

[9] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2018. 2

[10] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *NeurIPS*, pages 16664–16678, 2022. 2

[11] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, pages 113–123, 2019. 7

[12] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021. 1, 2

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 1

[14] Songlin Dong, Xiaopeng Hong, Xiaoyu Tao, Xinyuan Chang, Xing Wei, and Yihong Gong. Few-shot class-incremental learning via relation knowledge distillation. In *AAAI*, pages 1255–1263, 2021. 2

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 1

[16] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, pages 86–102, 2020. 2

[17] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *CVPR*, pages 9285–9295, 2022. 2

[18] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 1

[19] Robert M French and André Ferrara. Modeling time perception in rats: Evidence for catastrophic interference in animal learning. In *Proceedings of the 21st Annual Conference of the Cognitive Science Conference*, pages 173–178. Citeseer, 1999. 1

[20] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-DFCIL: relation-guided representation learning for data-free class incremental learning. In *ECCV*, pages 423–439, 2022. 2

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2015. 1

[22] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, pages 8340–8349, 2021. 6

[23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[24] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019. 2

[25] Linlan Huang, Xusheng Cao, Haori Lu, and Xialei Liu. Class-incremental learning with clip: Adaptive representation adjustment and parameter fusion. In *ECCV*, pages 214–231, 2024. 2, 3, 6, 7

[26] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 7

[27] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, pages 4904–4916, 2021. 1

[28] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, pages 709–727, 2022. 2, 3

[29] Dahuin Jung, Dongyoon Han, Jihwan Bang, and Hwanjun Song. Generating instance-level prompts for rehearsal-free continual learning. In *ICCV*, pages 11847–11857, 2023. 2

[30] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 2

[31] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshop*, pages 554–561, 2013. 6

[32] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009. 6

[33] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, pages 19730–19742, 2023. 1

[34] Li Li, Jiawei Peng, Huiyi Chen, Chongyang Gao, and Xu Yang. How to configure good in-context sequence for visual question answering. In *CVPR*, pages 26710–26720, 2024. 1

[35] Lan Li, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Addressing imbalanced domain-incremental learning through dual-balance collaborative experts. In *ICML*, 2025. 2

[36] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, pages 614–629, 2016. 2

[37] Victor Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Y Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. *NeurIPS*, pages 17612–17625, 2022. 8

[38] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, pages 12245–12254, 2020. 2

[39] Yichen Lu, Mei Wang, and Weihong Deng. Augmented geometric distillation for data-free incremental person reid. In *CVPR*, pages 7329–7338, 2022. 2

[40] Zilin Luo, Yaoyao Liu, Bernt Schiele, and Qianru Sun. Class-incremental exemplar compression for class-incremental learning. In *CVPR*, pages 11371–11380, 2023. 2

[41] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 6

[42] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022. 1, 2

[43] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42: 275–293, 2014. 2

[44] Mark D McDonnell, Dong Gong, Amin Parveneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. In *NeurIPS*, 2023. 2

[45] Sachit Menon and Carl Vondrick. Visual classification via description from large language models. In *ICLR*, 2023. 1

[46] Jaeyoo Park, Minsoo Kang, and Bohyung Han. Class-incremental learning for action recognition in videos. In *ICCV*, pages 13698–13707, 2021. 2

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019. 7

[48] Zhi-Hong Qi, Da-Wei Zhou, Yiran Yao, Han-Jia Ye, and De-Chuan Zhan. Adaptive adapter routing for long-tailed class-incremental learning. *Machine Learning*, 114(3):1–20, 2025. 2

[49] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763, 2021. 1, 3, 6, 7

[50] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *NIPS*, pages 506–516, 2017. 2

[51] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017. 1, 2, 6, 7

[52] Yujun Shi, Kuangqi Zhou, Jian Liang, Zihang Jiang, Jiashi Feng, Philip HS Torr, Song Bai, and Vincent YF Tan. Mimicking the oracle: An initial phase decorrelation approach for class incremental learning. In *CVPR*, pages 16722–16731, 2022. 2

[53] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *CVPR*, pages 11909–11919, 2023. 2, 3, 6, 7

[54] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4080–4090, 2017. 2

[55] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 6

[56] Hai-Long Sun, Da-Wei Zhou, Hanbin Zhao, Le Gan, De-Chuan Zhan, and Han-Jia Ye. Mos: Model surgery for pre-trained model-based class-incremental learning. In *AAAI*, pages 20699–20707, 2025. 2

[57] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *ECCV*, pages 254–270, 2020. 2

[58] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(11), 2008. 8

[59] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 6

[60] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *ECCV*, pages 398–414, 2022. 2

[61] Fu-Yun Wang, Da-Wei Zhou, Liu Liu, Han-Jia Ye, Yatao Bian, De-Chuan Zhan, and Peilin Zhao. BEEF: Bi-compatible class-incremental learning via energy-based expansion and fusion. In *ICLR*, 2023. 2

[62] Runqi Wang, Xiaoyue Duan, Guoliang Kang, Jianzhuang Liu, Shaohui Lin, Songcen Xu, Jinhu Lü, and Baochang Zhang. Attriclip: A non-incremental learner for incremental knowledge learning. In *CVPR*, pages 3654–3663, 2023. 2

[63] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. *NeurIPS*, pages 5682–5695, 2022. 2

[64] 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. In *ECCV*, pages 631–648, 2022. 2, 3, 6, 7

[65] 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 *CVPR*, pages 139–149, 2022. 1, 2, 3, 6, 7

[66] Zhengbo Wang, Jian Liang, Lijun Sheng, Ran He, Zilei Wang, and Tieniu Tan. A hard-to-beat baseline for training-free clip-based adaptation. In *ICLR*, 2024. 5

[67] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019. 2

[68] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492, 2010. 6

[69] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *NeurIPS*, pages 899–908, 2018. 2

[70] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, pages 3014–3023, 2021. 2

[71] Xu Yang, Zhangzikang Li, Haiyang Xu, Hanwang Zhang, Qinghao Ye, Chenliang Li, Ming Yan, Yu Zhang, Fei Huang, and Songfang Huang. Learning trajectory-word alignments for video-language tasks. In *ICCV*, pages 2504–2514, 2023. 1

[72] Xu Yang, Yongliang Wu, Mingzhuo Yang, Haokun Chen, and Xin Geng. Exploring diverse in-context configurations for image captioning. In *NeurIPS*, pages 40924–40943, 2023. 1

[73] Chao Yi, Lu Ren, De-Chuan Zhan, and Han-Jia Ye. Leveraging cross-modal neighbor representation for improved clip classification. In *CVPR*, pages 27402–27411, 2024. 2

[74] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *ICLR*, 2018. 2

[75] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *Transactions on Machine Learning Research*, 2024. 1

[76] Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *CVPR*, pages 23219–23230, 2024. 2, 3

[77] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *CVPR*, pages 6982–6991, 2020. 2

[78] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021. 1

[79] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995, 2017. 2

[80] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *CVPR*, pages 13208–13217, 2020. 2

[81] Bowen Zheng, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Multi-layer rehearsal feature augmentation for class-incremental learning. In *ICML*, pages 61649–61663, 2024. 2

[82] Bowen Zheng, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Task-agnostic guided feature expansion for class-incremental learning. In *CVPR*, pages 10099–10109, 2025. 2

[83] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *ICLR*, 2023. 2, 6

[84] Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, 133:1012–1032, 2025. 2, 6, 7

[85] Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, Lijun Zhang, and De-Chuan Zhan. Dual consolidation for pre-trained model-based domain-incremental learning. In *CVPR*, pages 20547–20557, 2025. 2

[86] Da-Wei Zhou, Yuanhan Zhang, Yan Wang, Jingyi Ning, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Learning without forgetting for vision-language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(6): 4489–4504, 2025. 2, 3, 5, 6, 7

[87] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, pages 16816–16825, 2022. 2

[88] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *IJCV*, 130(9):2337–2348, 2022. 2, 3, 6, 7

[89] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, pages 5871–5880, 2021. 2