LESS IS MORE: SUMMARIZING PATCH TOKENS FOR EFFICIENT MULTI-LABEL CLASS-INCREMENTAL LEARNING

Thomas De Min*
University of Trento
thomas.demin@unitn.it

Massimiliano Mancini
University of Trento

Stéphane Lathuilière
LTCI, Télécom Paris,
Institut Polytechnique de Paris

Subhankar Roy
University of Aberdeen

Elisa Ricci
University of Trento
Fondazione Bruno Kessler

ABSTRACT

Prompt tuning has emerged as an effective rehearsal-free technique for class-incremental learning (CIL) that learns a tiny set of task-specific parameters (or *prompts*) to instruct a pre-trained transformer to learn on a sequence of tasks. Albeit effective, prompt tuning methods do not lend well in the multi-label class incremental learning (MLCIL) scenario (where an image contains multiple foreground classes) due to the ambiguity in selecting the correct prompt(s) corresponding to different foreground objects belonging to multiple tasks. To circumvent this issue we propose to eliminate the prompt selection mechanism by maintaining task-specific *pathways*, which allow us to learn representations that do not interact with the ones from the other tasks. Since independent pathways in truly incremental scenarios will result in an explosion of computation due to the quadratically complex multi-head self-attention (MSA) operation in prompt tuning, we propose to reduce the original patch token embeddings into *summarized* tokens. Prompt tuning is then applied to these fewer summarized tokens to compute the final representation. Our proposed method Multi Label class incremental learning via summarising pAtch tokeN Embeddings (MULTI-LANE) enables learning disentangled task-specific representations in MLCIL while ensuring fast inference. We conduct experiments in common benchmarks and demonstrate that our MULTI-LANE achieves a new state-of-the-art in MLCIL. Additionally, we show that MULTI-LANE is also competitive in the CIL setting. Source code available at https://github.com/tdemin16/multi-lane

1 INTRODUCTION

Class-incremental learning (CIL) (Masana et al., 2022) aims to learn a classification model on a sequence of tasks (i.e. datasets with annotations for new classes) without forgetting previous knowledge and running into *catastrophic forgetting* (McCloskey & Cohen, 1989). Standard CIL methods assume that an image contains an object from a single class. Whereas in practice, images rarely depict a single subject (*e.g.*, an urban scene contains cars, traffic lights, people, etc.). To relax this assumption, multi-label class incremental learning (MLCIL) (Kim et al., 2020; Dong et al., 2023) aims to correctly classify an image into multiple classes, that may be introduced across different tasks.

Different from standard CIL, MLCIL is characterized by two additional problems. *First*, when updating the model, annotations are available only for the objects learned at the current task. As a result, if such images contain previously learned categories, they become negative samples for the old classes, increasing forgetting (Dong et al., 2023). *Second*, multi-label classification datasets are governed by long-tailed distributions (Kim et al., 2020). To prevent the forgetting of old knowledge, rehearsal-based methods store and replay samples of old training sessions (Kim et al., 2020; Dong et al., 2023). While this ensures positive training instances for both old and new classes, storing examples might not be possible due to privacy regulations (Zhu et al., 2021; Wang et al., 2022b).

In CIL, parameter-efficient fine-tuning (PEFT) techniques (Wang et al., 2022a;b; Smith et al., 2023; De Min et al., 2023) are an effective alternative to rehearsal-based methods (Rebuffi et al., 2017; Buzzega et al., 2020). In particular, prompt tuning (Jia et al., 2022; Wang et al., 2022b) learns a tiny set of task-specific parameters (or *prompts*) attached to a frozen pre-trained Vision Transformer (ViT) (Kolesnikov et al., 2021). These prompts are stored and selected to condition the final representation from the pre-trained backbone, adapting it to the task at hand. Although more

* Corresponding Author

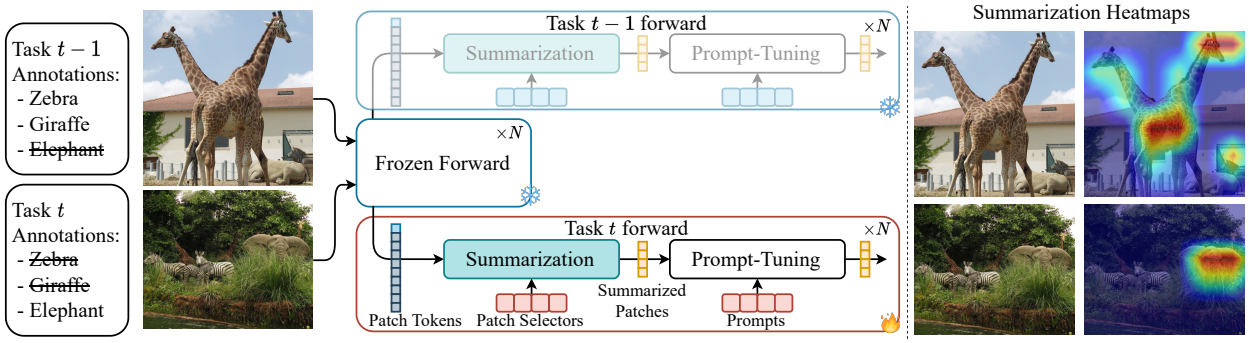


Figure 1: **A schematic overview of MLCIL and MULTI-LANE.** Annotations in MLCIL are incomplete, as shown with a strikethrough, even though images can depict objects from such classes. In MULTI-LANE, we learn task-specific subnetworks. Inside each subnetwork, Patch Selectors attend to the patch tokens and summarize them into fewer summarized patches. These summarized patches are used for prompt tuning, thus, reducing the computational complexity in each task forward. On the right, we visualize the attention heatmaps produced by Patch Selectors.

effective than the rehearsal-based CIL methods, the success of prompt tuning heavily relies on selecting the right prompt(s) at inference time (Wang et al., 2022a). The prompt selection issue is exacerbated in the MLCIL setting as multiple prompts need to be selected corresponding to foreground objects learned across tasks (e.g. in Fig. 1, the bottom image contains *zebra*, learned in task $t-1$, and *elephant*, learned in task t). Thus, if prompts are not selected carefully, the model may misinterpret foreground and background information. This makes prompt tuning a non-trivial approach for MLCIL (further details in Appx. B). As prompt tuning does not lend well to the MLCIL setting, existing methods directly address these two issues by allocating ad-hoc memory buffers (Kim et al., 2020; Liang & Li, 2022) and by pseudo-labeling new images for old classes (Dong et al., 2023).

In this work, we argue that prompt tuning can indeed be an effective solution for MLCIL if prompts across tasks do not interact with each other. To this end, we present our method, Multi Label class incremental learning via summarising pAtch tokeN Embeddings (MULTI-LANE) for MLCIL. In detail, we propose to eliminate cross-task prompt interaction and prompt selection at inference time by learning task-specific *pathways* or subnetworks (see Fig. 1). Such a design allows a more equal representation capacity for all classes across tasks while avoiding future samples becoming negatives for old classes. Albeit promising, a naïve implementation of such *pathways* would linearly scale the expensive multi-head self-attention (MSA) operations as the number of tasks increases, making inference in truly lifelong scenarios infeasible. As a remedy, we propose to summarize the patch token embeddings into fewer *summarized* patch embeddings. We do so by learning a set of tokens called *Patch Selectors* (see Sec. 3.3 for details) that attend to relevant regions of the image for the current task and summarize the semantic content (see Fig. 1 right for regions relevant to the tasks). These summarized tokens are then fed to the prompt tuning layers for learning task-specific representation. As the summarized tokens are fewer in number ($\sim 1-20$ in our experiments) than the original patch tokens (196 patch tokens with a 14×14 patchification in ViT-B/16), the number of MSA operations is drastically reduced due to quadratic nature of the MSA complexity. This allows us to sustain as many task-specific subnetworks as the number of tasks while ensuring a competitive inference speed. Moreover, MULTI-LANE is trained with only a classification loss and does not require any common regularization losses (Li & Hoiem, 2017) to prevent forgetting. During inference, the test image is forwarded through all the subnetworks in the form of summarized patches, and the resulting task logits are concatenated to yield the final class prediction.

Experiments on MS-COCO (Lin et al., 2014) and VOC2007 (Everingham et al., 2007) datasets demonstrate that MULTI-LANE achieves a new state-of-the-art in the MLCIL setting while keeping the computational requirements similar to standard ViT. Unlike existing MLCIL methods (Dong et al., 2023) that still rely on the rehearsal for good performance, MULTI-LANE does not require a memory buffer. Additional experiments on CIFAR-100 (Krizhevsky et al., 2009) and ImageNet-R (Hendrycks et al., 2021) show that our approach is also competitive in the standard CIL scenario.

In summary, our **contributions** are the following: (**i**) We propose MULTI-LANE that, to the best of our knowledge, is the first prompt-based framework for the challenging MLCIL setting; (**ii**) We introduce Patch Selectors, a simple yet effective patch summarization technique, that actualizes task-specific subnetworks while preventing the computational complexity from growing; and (**iii**) MULTI-LANE achieves a new state-of-the-art in MLCIL while not requiring rehearsal. Contrary to previous MLCIL approaches, MULTI-LANE is also competitive in the CIL setting.

2 Related Works

Continual Learning. Continual Learning methods can be broadly categorized into *regularization-based*, *rehearsal-based*, and *parameter-isolation* approaches (De Lange et al., 2021). Regularization-based methodologies tackle the problem of catastrophic forgetting via a regularization term that constrains the weights or activations of the model from deviating from the old configuration (Li & Hoiem, 2017; Kirkpatrick et al., 2017). Although regularization-based approaches improve over the sequential fine-tuning baseline, their effectiveness is limited. By allowing small portions of the past samples to be stored in a memory and replayed during learning on the current task, rehearsal-based methods (Rebuffi et al., 2017; Buzzega et al., 2020; Prabhu et al., 2020) show impressive results, comparable to joint training. However, relying on a replay buffer raises privacy-related concerns about stored images (Wang et al., 2023) and accrues computation as the task sequence grows. Finally, parameter isolation methods maintain task-specific subnetworks inside the model, where parameters across tasks do not interact with each other (Mallya & Lazebnik, 2018; Mallya et al., 2018). Very recently, ever since the advent of strong pre-trained ViT models, there has been a redux of parameter isolation methods that are based on visual prompt tuning (Jia et al., 2022) and have demonstrated performance on-par or even better than rehearsal-based methods. In detail, a tiny set of learnable parameters or prompts are trained for each task and used to condition the final representation from the ViT (Wang et al., 2022a;b; Smith et al., 2023). MULTI-LANE falls into the family of prompt tuning-based CIL methods but deviates from the existing related works because it eliminates prompt selection during inference and drastically reduces computation in MSA blocks via patch summarization.

Multi-Label Class-Incremental Learning. While multi-label classification is a fairly mature research field (Wang et al., 2017; Ridnik et al., 2021; Liu et al., 2021; Lanchantin et al., 2021), it has not received adequate attention in the incremental setting MLCIL (Dong et al., 2023). In detail, KRT (Dong et al., 2023) addresses the MLCIL problem by dynamically computing pseudo-labels for the old classes and by learning a cross-attention module to transfer knowledge from the old classes onto the new ones. Related to MLCIL, Partitioning Reservoir Sampling (PRS) (Kim et al., 2020) tackles online incremental learning in the context of multi-label classification by designing a reservoir sampling strategy for the long-tailed distribution problem. Similarly, OCDM (Liang & Li, 2022) optimizes the class distribution in memory to improve the performance over PRS. Different from the existing MLCIL approaches, MULTI-LANE is based on prompt tuning and does not require storing samples from past tasks, thus, it is the first rehearsal-free MLCIL method.

Token reduction in ViT. As the MSA in ViT scales quadratically in the number of patch tokens, token sparsification approaches have focused on reducing the input token sequence length by either token pruning or token merging. Pruning-based methods aim to reduce the token sequence length by removing either a predetermined number of tokens (Kong et al., 2021; Liang et al., 2022) or a dynamically adaptive number of tokens (Pan et al., 2021; Yin et al., 2022). Whereas, merging-based methods focus on reducing the sequence length by combining tokens (Ryoo et al., 2021; Bolya et al., 2023; Marin et al., 2023). In particular, TokenLearner (Ryoo et al., 2021) learns an MLP that outputs a spatial weight map over image features, which is then used to compress information by averaging the weighted spatial features. The patch summarization in MULTI-LANE is similar in spirit to TokenLearner, except we do not train an MLP, but task-specific learnable tokens (or Patch Selectors) learn to summarize the intermediate patch tokens into fewer summarized tokens via dot product attention. To the best of our knowledge, we are the first to incorporate the idea of token reduction in prompt tuning for continual learning, thereby bridging the two communities.

3 Method

In this Section, we first formalize the multi-label class-incremental learning problem (Sec. 3.1) and provide background on continual learning via prompt-tuning and its limitations (Sec. 3.2). Finally, we delve into the motivations behind our method MULTI-LANE and describe its components (Sec. 3.3).

3.1 Problem formulation

The goal of multi-label class incremental learning (MLCIL) is to classify multiple objects present in an input image, given that the model is incrementally trained on datasets annotated for only a subset of classes.

Formally, we aim to learn a function $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, parametrized by θ, mapping images in the space $\mathcal{X}$ to binary vectors in the space $\mathcal{Y}$, where $\mathcal{Y} = \{0, 1\}^k$ and k is the number of classes. In the incremental learning scenario, the training dataset is defined as $\mathcal{D} = \{\mathbf{D}^1, \ldots, \mathbf{D}^T\}$, where T is the number of training steps and $\mathbf{D}^t$ is the dataset available at step t. Each dataset represents a collection of image-label pairs $\mathbf{D}^t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t}$, where $\boldsymbol{x}_i^t$ and y_i^t are respectively

the i-th image and labels for task t. Note that, in each training step t, we have annotations for a different set of semantic categories $\mathcal{Y}^t$, with no overlap across training steps, i.e. $\mathcal{Y}^t \cap \mathcal{Y}^u = \varnothing$ for each $t \neq u$. As a consequence, a training image for a task t has ground-truth positive/negative annotations only for the subset $\mathcal{Y}^t$ (further details in Appx. A). The final output space is thus defined as $\mathcal{Y} = \bigcup_{t=1}^{T} \mathcal{Y}^t$.

3.2 Preliminaries: Continual Learning via Prompt-Tuning

We follow a family of CIL methods that utilize prompt tuning to circumvent the need to store past samples (Wang et al., 2022a;b; Smith et al., 2023). The main idea in prompt tuning is to prepend extra learnable tokens (or *prompts*) to the input sequences at intermediate layers of the ViT, that are optimized to learn new tasks while keeping the weights of the ViT backbone frozen. These prompts then serve as *instructions*, encoding the task-related information, to modulate the pre-trained representations for downstream tasks.

In the context of ViT (Kolesnikov et al., 2021), given an input image $\boldsymbol{x}$, let $\boldsymbol{h} \in \mathbb{R}^{L \times D}$ be the sequence-like output embedding from the input Embedding block, where L is the length of the sequence and D is the embedding dimension. To learn the current task t, the ViT backbone is kept frozen, and the task-specific learnable prompt parameters $\boldsymbol{p}_t \in \mathbb{R}^{L_{\boldsymbol{p}} \times D}$, having sequence length $L_{\boldsymbol{p}}$ and dimension D, are prepended to $\boldsymbol{h}$ to form the extended embedding features. Formally, the input to the i-th multi-head self-attention (MSA) layer (Vaswani et al., 2017) $\boldsymbol{h}^{(i)}$ is modulated by a *prompting function* as:

$$\boldsymbol{h}^{(i)}_{\text{mod}} = f_{\text{prompt}}(\boldsymbol{p}_t, \boldsymbol{h}^{(i)}), \tag{1}$$

where f_{prompt} is an MSA transformation function determining how the prompts are attached to the hidden embeddings. Specifically, applying f_{prompt} can be viewed as modifying the generic inputs $\boldsymbol{h} \in \mathbb{R}^{L \times D}$ to the MSA layer. The input $\boldsymbol{h}$ is projected into query, key, and values, denoted as $\boldsymbol{h}_Q, \boldsymbol{h}_K$ and $\boldsymbol{h}_V$, by projection matrices W^Q, W^K, and W^V. The MSA layer is defined as:

$$\begin{aligned} &\text{MSA}(\boldsymbol{h}_Q, \boldsymbol{h}_K, \boldsymbol{h}_V) = \text{Concat}(\text{h}_1, \ldots, \text{h}_\text{m}) W^O \\ &\text{where h}_\text{i} = \text{Attention}(\boldsymbol{h}_Q, \boldsymbol{h}_K, \boldsymbol{h}_V), \end{aligned}$$

where W^O is a projection matrix and m is the number of attention heads. Using this formulation of MSA, the f_{prompt} that implements prefix-tuning (Wang et al., 2022a) in particular is defined as:

$$f_{\text{prompt}}(\boldsymbol{p}_t, \boldsymbol{h}^{(i)}) = \text{MSA}(\boldsymbol{h}^{(i)}_Q, [\boldsymbol{p}_{t,K}; \boldsymbol{h}^{(i)}_K], [\boldsymbol{p}_{t,V}; \boldsymbol{h}^{(i)}_V]), \tag{2}$$

where $\boldsymbol{p}_t$ is split into $\boldsymbol{p}_{t,K}, \boldsymbol{p}_{t,V} \in \mathbb{R}^{L_{\boldsymbol{p}}/2 \times D}$, and then prepended to $\boldsymbol{h}^{(i)}_K$ and $\boldsymbol{h}^{(i)}_V$, respectively, while keeping $\boldsymbol{h}^{(i)}_Q$ unchanged (Wang et al., 2022a). Additionally, each task-specific prompt $\boldsymbol{p}_t$ is associated with a learnable key $\boldsymbol{k}_t \in \mathbb{R}^D$. The keys are trained to match the feature of the input instance, obtained with a query function $q(\cdot)$, using a matching loss. During inference, $q(\cdot)$ first selects the appropriate key $\boldsymbol{k}_t$ associated with the prompt $\boldsymbol{p}_t$. The corresponding $\boldsymbol{p}_t$ is then used to obtain the final modulated representation from the ViT backbone.

The prompt tuning methods, albeit effective for CIL, are not well-suited for MLCIL because images with similar semantics may belong to different tasks (foreground may become background in future tasks) (Dong et al., 2023), thus, query-key matching may fail during prompt selection. If selected prompts are wrong, they may misinterpret foreground information for irrelevant features.

3.3 Multi-Label Class Incremental Learning via Summarising Patch Token Embeddings

In order to avoid the critical drawbacks of prompt tuning methods, we design MULTI-LANE that entirely eliminates the need to select prompts with a query function. Instead, we create task-dependant *pathways* (or subnetworks), where each subnetwork implements prompt tuning with task-specific data. While this solution can eliminate interaction between tasks, and handle long-tailed distributions, it introduces computation costs during inference that scales linearly with the number of tasks, making it unfit for truly lifelong scenarios. To recall, the bulk of the computation in ViT is incurred in the MSA layers, causing the compuational complexity to grow quadratically in the token length $O(L^2)$ (Vaswani et al., 2017). Given we intend to have a separate forward pass for each task, the complexity rises to the order $O(TL^2)$, which scales linearly with T.

To still benefit from parallel subnetworks and yet keep computation under check, we propose the idea of *Patch Summarization*. In a nutshell, we introduce trainable tokens called *Patch Selectors* that learn to attend to the regions of the image, relevant to the current task, and *summarize* the L patch tokens into fewer summarized patch tokens. Unlike L2P (Wang et al., 2022b) or DualPrompt (Wang et al., 2022a), the prompt tuning in MULTI-LANE operates on top of the summarized patch tokens, rather than the full sequence of patch tokens. Next, we describe Patch Selectors in detail, followed by how prompt tuning is realized.

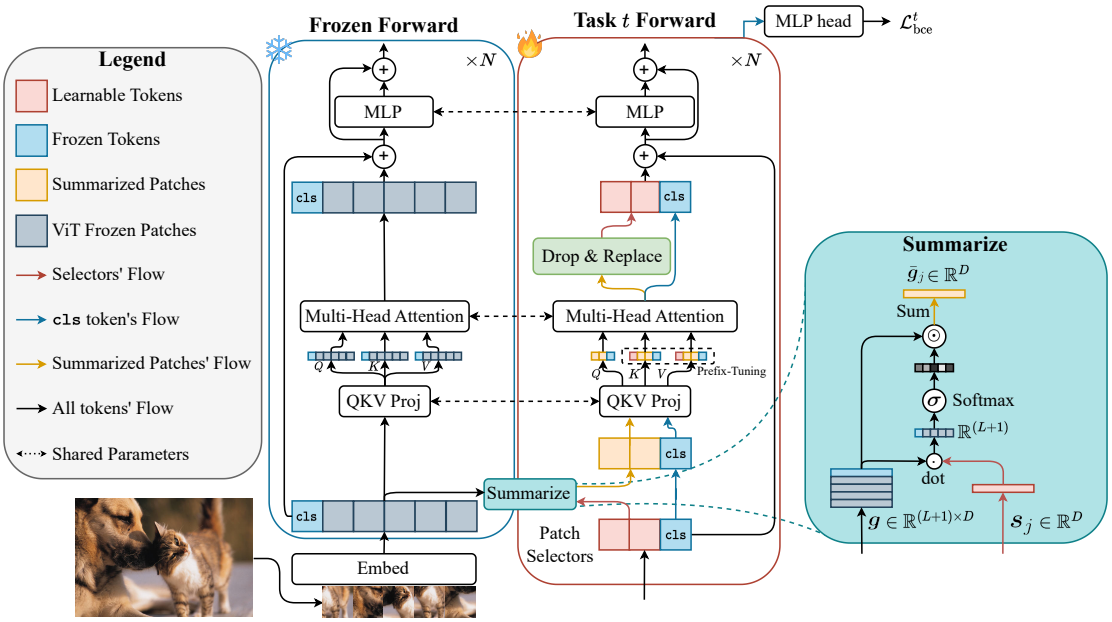


Figure 2: **Training of MULTI-LANE at task t.** Patch Selectors summarize patch token embeddings into fewer specialized tokens, reducing the computational complexity of the multi-head attention strongly. The MSA is computed between summarized tokens and the [CLS] token, and it is fine-tuned with prompt-tuning. After the computation of the MSA, summarized patches are dropped and only Patch Selectors and [CLS] token are propagated.

Patch Selectors. Let us denote the learnable Patch Selectors for a task t as $\boldsymbol{s}_t \in \mathbb{R}^{L_s \times D}$, having sequence length L_s and dimension D. We ensure that the sequence length of the Patch Selectors L_s is much smaller than the patch embedding length L. We also keep a copy of the class token ([CLS]), which is forwarded along with the Patch Selectors. We initialize one [CLS] per task with pre-trained weights (Wang et al., 2022a;b) as it allows for independent representations for each encountered task. At training time to compute the summarizations for each task, we maintain two forward passes (see Fig. 2): (i) a *frozen forward* where we obtain the intermediate representation of an image using a frozen ViT backbone, consisting of patch embeddings and the [CLS] token embedding; and (ii) *task forward* where the task-specific Patch Selectors $\boldsymbol{s}_t$ compute the importance of the input patch and the [CLS] token embeddings for the given task t. Let $\boldsymbol{g}^{(l)} \in \mathbb{R}^{(L+1)\times D}$ be the intermediate representation of the [CLS] and patch embeddings at the l-th transformer block. Dropping the layer index l for the convenience of notation, the j-th Patch Selector for task t summarizes the input embeddings $\boldsymbol{g}$ into $\bar{\boldsymbol{g}}$ as:

$$\bar{\boldsymbol{g}}_j = \sum_{k=1}^{L+1} \boldsymbol{\alpha}_k \boldsymbol{g}_k, \quad \text{where } \boldsymbol{\alpha} = \text{softmax}\left(\frac{\boldsymbol{s}_j \boldsymbol{g}^{\mathrm{T}}}{\sqrt{D}}\right). \tag{3}$$

$\boldsymbol{\alpha}$ denotes the contributions of the embeddings $\boldsymbol{g}$ corresponding to the Patch Selector $\boldsymbol{s}_j$. Similarly, by concatenating the outputs of L_s Patch Selectors, we obtain the summarized patches $\bar{\boldsymbol{g}} \in \mathbb{R}^{L_s \times D}$.

As, $L_s < L$, the computation in the MSA block of task-specific forward reduces to the order $O(L_s^2) \ll O(L^2)$. This reduction in computation allows us to afford T task-specific forward passes, making the representation learned by the current task-specific parameters (or *pathways*) independent of the previous tasks. These summarized patches are then fed to the MSA, which is discussed next.

Fine-tuning. After the patches have been summarized, we employ prefix-tuning (Wang et al., 2022a) to encode task-specific knowledge in the model. Following the notation used in Sec. 3.2, the modulated representation presented to the i-th MSA layer is given as:

$$\boldsymbol{h}^{(i)}_{\text{mod}} = f_{\text{prompt}}(\boldsymbol{p}_t, \bar{\boldsymbol{g}}^{(i)}). \tag{4}$$

Compared to Eq. (1), we use $\bar{\boldsymbol{g}}^{(i)}$ as an input to the prompting function. Note that different from the previous prompt tuning-based methods (Wang et al., 2022a;b), in our formulation the sequence length of the input embeddings is greatly reduced, which reduces the computational overhead incurred in the MSA layers. As the internal representation changes during the forward pass, we compute the summarization before each MSA, capturing salient task-specific features at

each step of the transformer. Since tokens do not interact until the next MSA, we drop the summarized patches (Drop & Replace in Fig. 2), propagating only the `[CLS]` token and the patch Patch Selectors. Algo. 1 (Appx. D) shows the pseudocode of patch selection and fine-tuning.

The `[CLS]` token conditioned by the task-specific parameters, emerging out from the final ViT block in the task forward, is used for learning a classifier. Formally, let $\hat{\boldsymbol{y}}$ be the predicted sigmoid-normalized logits for classes in task t. The learning objective is Binary Cross-Entropy loss over the current task classes:

$$\mathcal{L}^t_{\text{bce}} = -\frac{1}{|\mathcal{Y}^t|}\sum_{i\in\mathcal{Y}^t} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i) \tag{5}$$

where i is the class index for classes in $\mathcal{Y}^t$, y_i is the ground truth binary label and $\hat{y}_i$ is the corresponding model prediction.

Inference. During inference, the task identities of the image are not needed as MULTI-LANE is task agnostic. In detail, the test image is forwarded through the frozen forward, while Patch Selectors are forwarded through T task forward pathways (See Fig. 3). The `[CLS]` in each of the T forwards ends in a specific classification head, which results in T logits. The task-specific logits are then concatenated to yield the final prediction as $\hat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_1; \ldots; \hat{\boldsymbol{y}}_T]$, where $[\cdot;\cdot]$ denotes the concatenation operation. The final prediction is done by independent thresholding of each sigmoid-normalized logit, which is the standard procedure in multi-label and binary classification.

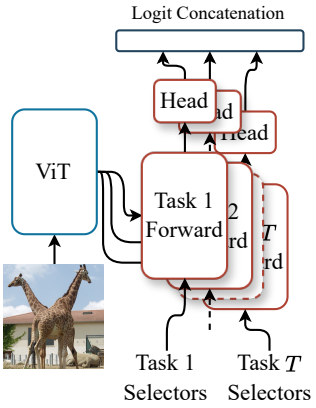


Figure 3: MULTI-LANE at inference time: all task forwards are run in parallel.

4 EXPERIMENTS

In this section, we present our experimental procedure (Sec. 4.1), and we compare MULTI-LANE with the current MLCIL state-of-the-art (Sec. 4.2). We further evaluate our method by visualizing the summarization heatmaps generated by our Patch Selectors during summarization (Sec. 4.3). Following, we provide a complete ablation of the different components of our approach by removing the multiple *pathways*, Drop & Replace, and the pre-head normalization. Additionally, we show how the performances of MULTI-LANE vary when changing the number of Patch Selectors or substituting them with Token Merging (Sec. 4.4). To demonstrate that MULTI-LANE multiple *pathways* are computationally sustainable, we compare the number or multiply-accumulate operations with the SOTA prompt-based approach. Finally, we test our approach in two common benchmarks in CIL, showing that MULTI-LANE is also competitive in this setting (Sec. 4.6).

4.1 EXPERIMENTAL SETTING

Datasets. We follow previous works in the field (Kim et al., 2020; Liang & Li, 2022; Dong et al., 2023) and test MULTI-LANE on MS-COCO (Lin et al., 2014), and VOC2007 (Everingham et al., 2007) datasets. MS-COCO consists of 300k images ranging in 80 categories from the real-world domain. Similarly, VOC2007 is a 10k image dataset annotated for 20 classes. Both are initially meant for segmentation tasks, however, by following Dong et al. (2023), we adapted them to the multi-label class-incremental setting. We trained our model in two configurations for both datasets: a) MS-COCO B0-C10, the model was trained incrementally, and each task is annotated with 10 novel classes; b) MS-COCO B40-C10, we first trained on the first task composed of 40 classes, and then trained incrementally on the remaining 4; c) VOC2007 B0-C4, we trained incrementally on 5 tasks, each with 4 new classes; d) VOC2007 B10-C2, as for MS-COCO, we first trained on the first task composed of 10 classes, and then fine-tuned incrementally on the remaining 10, spanning 5 tasks.

Metrics. Following Kim et al. (2020) and Dong et al. (2023), we report the Average mean Average Precision across different tasks (Avg. mAP), and the mean Average Precision after training on the last task and evaluating on all seen classes (mAP). We also report the class-wise F1 score and overall F1 score, in MS-COCO, for all methods in the appendix (Appx. C). For class-incremental learning, we show the accuracy of methods after training on the last task and evaluating all seen classes. For each metric, we highlight in **bold** the best approach and underline the second best.

Baselines. We focus our comparison with the current state-of-the-art in multi-label class-incremental learning, KRT (Dong et al., 2023). Because the literature is relatively unexplored, we also evaluate our approach against online MLCIL methods, PRS (Kim et al., 2020), and OCDM (Liang & Li, 2022). We proceed by comparing MULTI-LANE

Table 1: Comparison with the state-of-the-art on MS-COCO B0-C10 and B40-C10 configurations. We highlight MULTI-LANE, prompt-based methods, and MLCIL approaches.

Method	Buffer size	MS-COCO B0-C10		MS-COCO B40-C10	
		Avg. mAP (↑)	mAP (↑)	Avg. mAP (↑)	mAP (↑)
UB	-	-	81.8	-	81.8
FT (Ridnik et al., 2021)	0	38.3	16.9	35.1	17.0
PODNet (Douillard et al., 2020)	0	43.7	25.6	44.3	24.7
oEWC (Schwarz et al., 2018)	0	46.9	24.3	44.8	27.3
LWF (Li & Hoiem, 2017)	0	47.9	28.9	48.6	29.9
KRT (Dong et al., 2023)	0	74.6	65.9	77.8	74.0
CODA-P (Smith et al., 2023)	0	74.0	65.4	73.9	67.5
iCaRL (Rebuffi et al., 2017)	20/class	59.7	43.8	65.6	55.7
BiC (Wu et al., 2019)	20/class	65.0	51.1	65.5	55.9
ER (Riemer et al., 2019)	20/class	60.3	47.2	68.9	61.6
TPCIL (Tao et al., 2020)	20/class	69.4	60.6	72.4	66.5
PODNet (Douillard et al., 2020)	20/class	70.0	58.8	71.0	64.2
DER++ (Buzzega et al., 2020)	20/class	72.7	63.1	73.6	66.3
KRT-R (Dong et al., 2023)	20/class	76.5	70.2	78.3	75.2
PRS (Kim et al., 2020)	1000	48.8	27.9	50.8	33.2
OCDM (Liang & Li, 2022)	1000	49.5	28.5	51.3	34.0
KRT-R (Dong et al., 2023)	1000	75.7	69.3	78.3	75.1
MULTI-LANE (Ours)	0	**79.1**	**74.5**	**78.8**	**76.6**

with rehearsal-free approaches that are not designed for MLCIL, CODA-Prompt (Smith et al., 2023), Learning without Forgetting (Li & Hoiem, 2017), and oEWC (Schwarz et al., 2018). Additionally, we provide a comparison with rehearsal-based continual learning techniques, ER (Riemer et al., 2019), BiC (Wu et al., 2019), DER (Buzzega et al., 2020), PODNet (Douillard et al., 2020), TPCIL (Tao et al., 2020), and iCaRL (Rebuffi et al., 2017) and we evaluate them using a replay buffer of 2/20 samples per class, and 1000 exemplars, based on the benchmark dataset.

Implementation details. Following the previous works in prompt-based continual learning, we used ViT-B/16, pre-trained on ImageNet-1k, with patches of dimension 16×16. To train it, we used Adam optimizer (Kingma & Ba, 2015) with $\beta_1 = 0.9$, and $\beta_2 = 0.999$ and a batch size of 128 and 256 images for MS-COCO and VOC2007 respectively. All experiments were executed using a single A100 NVIDIA GPU. The learning rate followed a cosine annealing schedule, starting from 0.03 for MS-COCO and 0.05 for VOC2007. Input images were scaled to a dimension of 224×224, which corresponds to the default ViT-B/16 input dimension.

About the hyperparameters strictly related to our method, we set the number of Patch Selectors and the number of prompts to 20 for MS-COCO, and 10 for VOC2007, as we saw no significant improvements in increasing them over these values (see Fig. 5). We outline that using a standard ViT-B/16, each Patch Selector adds $D = 768$ parameters, thus, the total Patch Selector parameters number is counted as $T \times L_s \times D$ (*e.g.* for 8 tasks and 20 Patch Selectors, we add 122'880 parameters). Following (Wang et al., 2022a; Smith et al., 2023), we used prefix-tuning, and prompts were injected in the first five layers of ViT.

4.2 Comparison with the State-of-the-Art

In Table 1 and Table 2, we compare MULTI-LANE with the baselines in respectively the MS-COCO (Lin et al., 2014) and VOC2007 (Everingham et al., 2007) datasets (see Appx. C for additional metrics).

MULTI-LANE achieves a new state of the art in both scenarios we evaluated without storing samples for future replay. The first scenario we consider is incremental learning over the entire MS-COCO dataset, consisting of 8 tasks. Here, our method outperforms KRT-R's best mAP by 4.3% and achieves an Average mAP of 79.1% (+2.6% improvement), thus, highly reducing the gap with the joint-learning upper bound. However, in the 5-tasks B40-C10 configuration[1], we surpass the final mAP of the previous state-of-the-art by 1.5% and the Average one by about 0.5%. As the number of tasks reduces, the chance that new images are negative samples for old classes is strongly reduced, thus, also the benefit of splitting the forward pass into multiple pathways. Moreover, in MS-COCO B40-C10, the first task contains

[1]To learn the base classes and then the incremental steps, we treated the base classes as a bigger task.

Table 2: Comparison with the state-of-the-art on VOC2007 B0-C4 and B10-C2 configurations. We highlight MULTI-LANE, prompt-based methods, and MLCIL approaches.

Method	Buffer size	VOC2007 B0-C4		VOC2007 B10-C2	
		Avg. mAP (↑)	mAP (↑)	Avg. mAP (↑)	mAP (↑)
UB	-	-	93.6	-	93.6
CODA-P (Smith et al., 2023)	0	90.6	84.5	90.2	85.0
FT (Ridnik et al., 2021)	2/class	82.1	62.9	70.1	43.0
iCaRL (Rebuffi et al., 2017)		87.2	72.4	79.0	66.7
BIC (Wu et al., 2019)		86.8	72.2	81.7	69.7
ER (Riemer et al., 2019)		86.1	71.5	81.5	68.6
TPCIL (Tao et al., 2020)		87.6	77.3	80.7	70.8
PODNet (Douillard et al., 2020)		88.1	76.6	81.2	71.4
DER++ (Buzzega et al., 2020)		87.9	76.1	82.3	70.6
KRT-R (Dong et al., 2023)		90.7	83.4	87.7	80.5
MULTI-LANE (Ours)	0	**93.5**	**88.8**	**93.1**	**88.3**

Figure 4: Visualization of the attention paid by Patch Selectors of different tasks. On the left, are ground truth annotations for images. On the right, are depicted original images, two heatmaps for tasks related to the classes of the image, and two other heatmaps for tasks that are not related to the classes of the image.

approximately half of the dataset, forcing our method to rely on 20 summarized patches to predict 40 different classes. Despite being disadvantaged, our method still achieves a new state of the art in this configuration.

In VOC2007 our approach improves CODA-P performances (previous state-of-the-art) mAP and Average mAP by about 4.3% and 2.3% respectively, getting closer to the upper bound. Contrary to what happens in MS-COCO B40-C10, we maintain a larger margin from the previous SOTA in the 6-tasks B10-C2 VOC2007 configuration. Compared to MS-COCO B40-C10, where the number of tasks drops from eight to five, the number of tasks of VOC2007 B10-C2 grows from 5 to 6. Thus, the likelihood of an image becoming a negative sample remains almost identical. Additionally, having a lot of tasks with a low number of classes allows our model to exploit the multiple pathways even better, making incremental tasks easier.

Compared with previous approaches, we finally notice MULTI-LANE robustness across different datasets and configurations, highlighting the benefits of designing a prompt-based approach that relies on all trained parameters, an essential feature in MLCIL. Furthermore, we remark on the benefits of using a multiple *pathways* architecture in long-tail distributed datasets in Appx. E.

4.3 Qualitative Results

As explained in Sec. 3, for each task MULTI-LANE learns a set of Patch Selectors that focus on specific parts of the image to produce a summarization tailored to the specific task. In Fig. 4, we present a visualization of the attention

Table 3: Component ablations of our approach. In the last row, we report MULTI-LANE with all 3 components enabled.

Component			mAP (↑)
Norm	Parallel *Pathways*	D&R	
	✓	✓	72.66±0.18
✓		✓	67.96±0.30
✓	✓		20.26±0.36
✓	✓	✓	74.57±0.06

Table 4: Comparison in terms of Giga Multiply-Accumulate operations, frames per second (FPS), and trainable parameters on a 10-task benchmark. Patch Selectors drastically reduce the computational overhead of a naïve multiple *pathways* architecture.

Approach	GMACs (↓)	FPS (↑)	Trainable Params (M)
ViT-B/16	16.9	918	85.8
Naïve multiple *pathways*	168.7	92	858.0
CODA-P	33.7	436	3.7
MULTI-LANE - 1 Selector	18.6	790	1.2
MULTI-LANE - 20 Selectors	34.7	393	1.3

paid by Patch Selectors of different incremental steps. In particular, we show side by side two heatmaps produced by tasks containing classes that are in the image (left) and two for tasks whose classes are not (right). In the first row of Fig. 4, we can appreciate how Patch Selectors of task 3 focus on the region where the chair and the dining table happen to be. Analogously, task 6 Patch Selectors are summarizing the image by giving the most attention to the skateboard. Despite focusing on important aspects of the scene, Patch Selectors seem to be also activated by shadows or elements that could be in the scene. An example of such behavior can be seen in the heatmaps of tasks 4 and 7 of the first row (whose classes are not in the image); where they show peaked activation in regions that are not directly relevant in predicting correct classes. We hypothesize that task 4 and task 7 strong activations may be generated by Patch Selectors focusing on finding respectively a handbag and a snowboard[2]. In the second row, instead, Patch Selectors seem to activate also in the presence of shadows or certain patterns in the terrain. Thus, we argue that they do not only activate in the presence of a specific object but also activate to retrieve context from the scene (*e.g.* by looking at regions that are usually populated by certain objects or patterns). We show additional failure cases in Appx. G.

4.4 Ablations

Components. In Table 3 we ablate the components of our approach by averaging the final mAP on MS-COCO B0-C10 across 3 runs with different class orderings and seeds. We set as a reference MULTI-LANE trained with 20 Patch Selectors per task, Drop & Replace, pre-head normalization, and multiple *pathways*, as described in Sec. 3.3. We begin our analyses by removing the pre-head normalization, which causes MULTI-LANE performance to drop by about 2% from the reference. Then, we test our method without the multiple *pathways* structure we proposed. We instead employed a query-key selection mechanism, typical of prompt-based approaches (Wang et al., 2022a). We used frozen forward features as a query vector and selected the most probable task-specific parameters (prompts and Patch Selectors) to compute the final prediction for the image. By using only one *pathway* our method suffers from the same architectural issues of prompt-based approaches (see Sec. 3.2), causing a performance drop. Finally, we ablate the Drop&Replace operation. Instead of summarizing patch tokens at each ViT block, we summarize them only at the first layer and propagate them, causing the mAP to drop to ∼20% (see Fig. 8 in the Appendix).

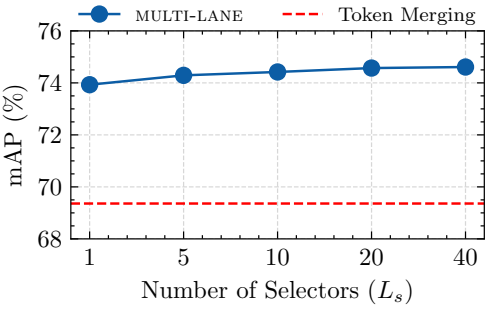


Figure 5: Patch Selectors vs Token Merging in MS-COCO B0-C10.

Patch Summarization. In Fig. 5, we investigate the importance of Patch Selectors in our architecture. In detail, we show the performance of our model when varying the number of Patch Selectors L_s. Additionally, we show MULTI-LANE performance when Token Merging (ToME) (Bolya et al., 2023) is used to summarize patch tokens (details in Appx. F). ToMe proves to be suboptimal as it does not isolate the foreground from the background for a specific task. On the other hand, Patch Selectors are more effective as they are tailored for the task they have been trained on. Moreover, we notice how the performances are not much affected by the number of Patch Selectors, demonstrating that enough information can be encoded even into a single patch.

[2]Classes that belong to respectively task 4 and 7.

Table 5: Comparison with the state-of-the-art in popular class-incremental learning benchmarks. We highlight MULTI-LANE, prompt-based methods, and MLCIL approaches.

Method	Setting	Buffer size	Accuracy (↑)	
			CIFAR-100	ImageNet-R
UB	-	-	90.85±0.12	79.13±0.18
KRT-R (Dong et al., 2023)	MLCIL	5000	65.10±2.60	63.67±1.27
L2P (Wang et al., 2022b)	CIL	0	82.50±1.10	69.29±0.73
DualPrompt (Wang et al., 2022a)			83.05±1.16	71.31±0.62
CODA-P (Smith et al., 2023)			**86.25**±0.74	**75.45**±0.56
MULTI-LANE (Ours)	MLCIL	0	84.84±0.34	73.69±0.40

4.5 Computational requirements

In Sec. 3.3, we provide a theoretical perspective on the computational demands of the multiple *pathways* architecture and how Patch Selectors can solve it. To further demonstrate the validity of our analyses, we empirically demonstrate that Patch Selectors do strongly reduce the computational requirements. In Table 4, we report the number of multiply-accumulate operations, the frames per second, and the number of trained parameters required by our method on 10-task incremental sessions and we compare them with CODA-Prompt, a Naïve implementation of multiple *pathways*, and the unaltered ViT-B/16 forward. On a 10-task benchmark MULTI-LANE requires GMACs operations and FPS similar to a ViT-B/16 forward while being capable of achieving a new state-of-the-art in MS-COCO B0-C10. Note that MS-COCO B0-C10 is divided into 8 tasks, thus, it requires slightly fewer GMACs for this setting. By raising the number of Patch Selectors to 20, our approach reaches computational requirements close to CODA-P. Furthermore, we want to stress that, in their lighter configuration, Patch Selectors require almost 10× fewer GMAC operations and are almost 10× faster compared to Naïve multiple *pathways*. This proves that, compared to the previous state-of-the-art, our methodology is not only superior in mAP terms but also as "light" as a plain ViT-B/16.

4.6 Additional results in CIL

To show the versatility of our method, we further evaluate our approach in the class-incremental learning setting (see Table 5). We compare MULTI-LANE against the state-of-the-art prompt-based class-incremental learning approaches in the CIFAR-100 and ImageNet-R benchmarks. Although our method is designed for MLCIL, MULTI-LANE is also robust in class-incremental learning, as it consistently outperforms seminal prompt-based approaches (Wang et al., 2022b;a). However, MULTI-LANE is not designed for CIL and it is not capable of reaching the same performance as CODA-Prompt, which is the SOTA in prompt-based CIL. Finally, we show that methods designed for MLCIL may not be directly applied to CIL. Indeed, KRT (Dong et al., 2023) fails in achieving meaningful performances in this setting.

5 Conclusions

In this work, we present MULTI-LANE, a novel prompt-based approach that deals with multi-label class-incremental. Contrary to previous prompt-based works that output a single representation for each image, we propose to compute multiple representations in parallel. By outputting multiple feature vectors, our method can put more focus on each salient point of the image. To avoid the computational complexity from growing too much, we introduce a novel component we call Patch Selectors that heavily summarizes the internal representation of the ViT. Patch Selectors allow our method to generate a feature vector for each encountered task dataset, limiting forgetting. Experiments in MS-COCO and VOC2007 datasets show that MULTI-LANE achieves a new state-of-the-art in multi-label class-incremental learning.

Acknowledgments

We acknowledge the CINECA award under the ISCRA initiative for the availability of high-performance computing resources and support. E.R. and M.M. are supported by the MUR PNRR project FAIR - Future AI Research (PE00000013), funded by NextGeneration EU. E.R. is also supported by the EU projects AI4TRUST (No.101070190) and ELIAS (No.01120237) and the PRIN project LEGO-AI (Prot.2020TA3K9N). T.D.M. is funded by NextGeneration EU. This work has been supported by the French National Research Agency (ANR) with the ANR-20-CE23-0027.

REFERENCES

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *ICLR*, 2023.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *NeurIPS*, 2020.

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. *TPAMI*, 2021.

Thomas De Min, Massimiliano Mancini, Karteek Alahari, Xavier Alameda-Pineda, and Elisa Ricci. On the effectiveness of layernorm tuning for continual learning in vision transformers. In *ICCVW*, 2023.

Songlin Dong, Haoyu Luo, Yuhang He, Xing Wei, Jie Cheng, and Yihong Gong. Knowledge restore and transfer for multi-label class-incremental learning. In *ICCV*, 2023.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, 2020.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007.

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*, 2021.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, 2022.

Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. Imbalanced continual learning with partitioning reservoir sampling. In *ECCV*, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

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. *PNAS*, 2017.

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

Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Mengshu Sun, Wei Niu, Xuan Shen, Geng Yuan, Bin Ren, Minghai Qin, et al. Spvit: Enabling faster vision transformers via soft token pruning. In *ECCV*, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Jack Lanchantin, Tianlu Wang, Vicente Ordonez, and Yanjun Qi. General multi-label image classification with transformers. In *CVPR*, 2021.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 2017.

Yan-Shuo Liang and Wu-Jun Li. Optimizing class distribution in memory for multi-label online continual learning. *arXiv preprint arXiv:2209.11469*, 2022.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *ICLR*, 2022.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Shilong Liu, Lei Zhang, Xiao Yang, Hang Su, and Jun Zhu. Query2label: A simple transformer way to multi-label classification. *arXiv preprint arXiv:2107.10834*, 2021.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, 2018.

Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. Token pooling in vision transformers for image classification. In *WACV*, 2023.

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. *TPAMI*, 2022.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. 1989.

Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red^2: Interpretability-aware redundancy reduction for vision transformers. *NeurIPS*, 2021.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

Tal Ridnik, Emanuel Ben-Baruch, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. In *ICCV*, 2021.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019.

Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? *arXiv preprint arXiv:2106.11297*, 2021.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *ICML*, 2018.

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*, 2023.

Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *ECCV*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*, 2023.

Zhouxia Wang, Tianshui Chen, Guanbin Li, Ruijia Xu, and Liang Lin. Multi-label image recognition by recurrently discovering attentional regions. In *ICCV*, 2017.

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*, 2022a.

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*, 2022b.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, 2019.

Hongxu Yin, Arash Vahdat, Jose Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. Adavit: Adaptive tokens for efficient vision transformer. In *CVPR*, 2022.

Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, 2021.

Appendix

A Task Transition

In this paragraph we provide an overview of the task transition in MLCIL, highlighting how negative samples for past tasks are formed. In class-incremental learning, object categories appear only in one task during the incremental learning (*e.g.* if the class bicycle is observed in task N, no image with bicycle annotated can appear in task M, with $M \neq N$). However, real-world images rarely depict only one class: thus, multi-label class-incremental learning allows images to belong to multiple categories. Following standard definitions for this task (Dong et al., 2023), a class is *annotated* only in one incremental step while it can still appear but *unlabeled* in other steps. In Fig. 6, we show a toy model of the task transition just described. In task $t-1$, only *bicycle* and *bus* are annotated, leaving the other three classes as background. Moving to task t, previously annotated classes are now considered background, and only *car* and *person* are annotated. As a result, task t image becomes a negative sample for task $t-1$ classes. Similarly, for task $t+1$, labels are provided only for the traffic light class, making it a negative sample for previous incremental steps. Note that, for simplicity, in Fig. 6 we considered the same image with different annotations. In practice, no assumption is made on the overlap between sets of images across different tasks.

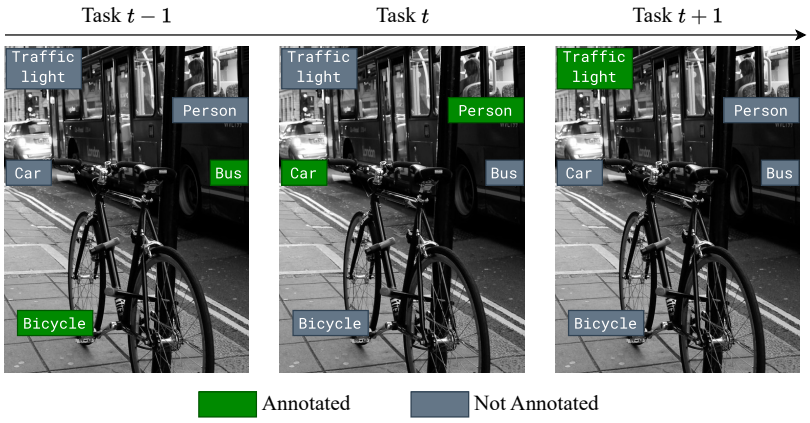


Figure 6: **Annotations across task transitions**: Past task objects can appear in future tasks in unlabeled form.

B Prompt selection issues in MLCIL

In Sec. 1, we argue that prompt-tuning-based methods are more prone to select incorrect prompts in MLCIL than in CIL, as distinct foreground objects require different prompts to be selected. In this section, we elaborate on this with the help of a toy visualization shown in Fig. 7.

At inference, given an image containing a cat and a dog, the visual encoder will extract embeddings close to the ones of *cat* and *dog* concepts. We refer to such embedding as the query. Task prompts are represented by task centroids (or keys), which lie in the same latent space as queries. By computing the cosine similarity between the query and all the keys, prompt-based methods either do a hard (Wang et al., 2022a) or a soft prompt selection (Wang et al., 2022b; Smith et al., 2023).

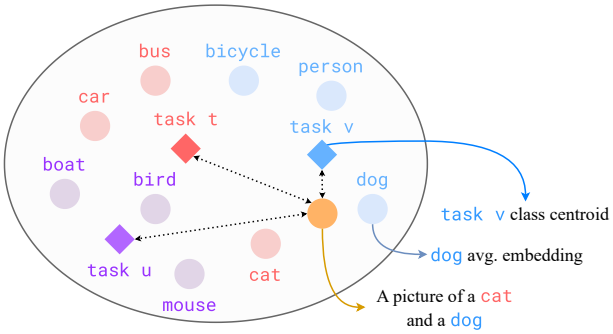


Figure 7: **Prompt selection**: The query vector that embeds an image of a *cat* and a *dog* is close to task v but distant to task t in the embedding space. This distance could result in a missed detection of the class *cat*.

A hard prompt selection, as done in (Wang et al., 2022a), will result in a prompt corresponding to only a subset of the image content. In the example shown in Fig. 7, *cat* is part of task t, while *dog* of task v. However, only task v key would be selected, being the nearest key from the

Figure 8: **Visualizing the summarization at different layers**: The summarization is refined at each step as representations of the frozen pre-trained model become more high-level, better capturing the image semantics. In layer 1, heatmaps tend to be uniform and quite homogeneous among tasks, suggesting that information is not rich enough to provide meaningful summarizations. In layer 6, the specificity of the summarized patches increases, starting to focus on more discriminative elements of the image (e.g. *skateboard* and *human* in the first row, *road* and *person* in the second). Finally, in layer 12, Patch Selectors focus on specific image regions, retaining only the semantic information (e.g. *skateboard* in the first row, *bicycle* and *person* in the second).

query, leading to a prompt selection not specialized for *cats*. While a soft selection mechanism remedies the omission of relevant prompts by considering all task prompts, their importance is weighted proportionally to the query-task centroid distance (Smith et al., 2023). However, the task centroids may not faithfully represent the semantic relevance of a task (e.g. task t contains *cat* but its centroid is influenced by other classes such as *bus* and *car*). Moreover, unrelated tasks (e.g. task u in Fig. 7) may have a similar distance with the query as task t, and thus, it could interfere with the final classification. Contrarily, our approach sidesteps the issues of representing tasks by task centroids and their selection altogether by extracting task-specific representations in parallel.

C Additional results in MS-COCO

In Table 6, we show results with the same experimental procedure of Table 1 with two additional metrics: the class-wise F1 score and the overall F1 score. We did not search for the optimal threshold for both metrics and used 0.8 as a decision boundary like in Dong et al. (2023). Indeed, MULTI-LANE does not achieve state-of-the-art results, in such metrics, in three out of four cases. However, we consider mAP and Avg. mAP more descriptive metrics and they are hyperparameter-free, thus, we focused our analyses on those and we neglected the other two. Here we report them for completeness.

D MULTI-LANE PSEUDOCODE

In Algo. 1, we report the pseudocode of the multi-head attention of MULTI-LANE in a PyTorch-like style. First, we compute the patch tokens summarization and concatenate summarized patches to task-specific class tokens. We compute the query, key, and value projections, and, if provided, we concatenate prompts to key and value matrices. The three projections are used to compute the task MSA, and summarized patches are dropped. Finally, we use the original patch tokens to compute the MSA for the frozen forward.

E MULTI-LANE ENSURES A MORE EQUAL REPRESENTATION CAPACITY

As we introduce in Sec. 1 and Sec. 3, by allocating task-specific forwards, MULTI-LANE provides an equal representation capacity for classes along all tasks. By assigning different weights to distinct tasks and by exploiting all task-specific parameters in parallel, MULTI-LANE can put more focus on less represented classes. To demonstrate that multiple *pathways* architecture does mitigate the long-tailed distribution issues of multi-label datasets (see Fig. 9a), we computed the difference in class-wise F1 score between our method and CODA-Prompt. In detail, we computed the F1 score independently for each class, then we averaged the resulting F1 scores and computed the difference between the metric we obtained with MULTI-LANE and with CODA-P. We grouped classes into 10 bins depending on their frequencies and in Fig. 9b we show the differences between the two methods. Our method consistently shows better

Algorithm 1 MULTI-LANE Multi-head Attention

```
def summarize_patches(patches, patch_selectors):
    # Input:
    #   - patches:  [batch_size, num_tokens, token_dim]
    #   - patch_selectors:  [num_tasks, num_selectors, token_dim]
    # Output:
    #   - summarized_patches:  [num_tasks, batch_size, num_selectors, token_dim]
    attention_scores = einsum("tcd, bnd -> tbcn", patch_selectors, detach(patches))
    norm_scores = softmax(attention_scores * (token_dim**-0.5), dim=-1)
    summarized_patches = einsum("tbcn, bdn -> tbcd", norm_scores, detach(patches))
    return summarized_patches

def multi_head_attention(patches, task_cls_tokens, patch_selectors, prompts:Optional):
    # Input:
    #   - patches:  [batch_size, num_tokens, token_dim]
    #   - task_cls_tokens:  [num_tasks, batch_size, 1, token_dim]
    #   - patch_selectors:  [num_tasks, num_selectors, token_dim]
    #   - prompts:  [2, num_tasks, batch_size, num_heads, length, head_dim]
    # Output:
    #   - patches:  [batch_size, num_tokens, token_dim]
    #   - task_cls_tokens:  [num_tasks, batch_size, 1, token_dim]
    summarized_patches = summarize_patches(patches, patch_selectors)
    task_tokens = cat((task_cls_tokens, summarized_patches), dim=2)

    # task_qkv:  [3, num_tasks, batch_size, num_heads, num_selectors+1, head_dim]
    task_qkv = qkv_proj(task_tokens)
    task_q, task_k, task_v = unbind(task_qkv, dim=0)
    if prompts:
        task_k = cat((prompts[0], task_k), dim=3)
        task_v = cat((prompts[1], task_v), dim=3)
    task_tokens = MHA(task_q, task_k, task_v)

    # Drop&Replace:  Drop summarized_patches
    task_cls_tokens = task_tokens[:, :, 0:1]

    # Frozen MHA
    with no_grad():
        qkv = qkv_proj(patches)
        q, k, v = unbind(qkv, dim=0)
        patches = MHA(q, k, v)
    return patches, task_cls_tokens
```

Table 6: Comparison with the state-of-the-art on MS-COCO B0-C10 and B40-C10 configurations. We highlight MULTI-LANE, prompt-based methods, and MLCIL approaches.

Method	Buffer size	MS-COCO B0-C10				MS-COCO B40-C10			
		Avg. mAP ↑	CF1 ↑	OF1 ↑	mAP ↑	Avg. mAP ↑	CF1 ↑	OF1 ↑	mAP ↑
UB	-	-	76.4	79.4	81.8	-	76.4	79.4	81.8
FT (Ridnik et al., 2021)	0	38.3	6.1	13.4	16.9	35.1	6.0	13.6	17.0
PODNet (Douillard et al., 2020)		43.7	7.2	14.1	25.6	44.3	6.8	13.9	24.7
oEWC (Schwarz et al., 2018)		46.9	6.7	13.4	24.3	44.8	11.1	16.5	27.3
LWF (Li & Hoiem, 2017)		47.9	9.0	15.1	28.9	48.6	9.5	15.8	29.9
KRT (Dong et al., 2023)		74.6	55.6	56.5	65.9	77.8	64.4	63.4	74.0
CODA-P (Smith et al., 2023)		74.0	48.1	47.7	65.4	73.9	56.1	57.4	67.5
TPCIL (Tao et al., 2020)	5/class	63.8	20.1	21.6	50.8	63.1	25.3	25.1	53.1
PODNet (Douillard et al., 2020)		65.7	13.6	17.3	53.4	65.4	24.2	23.4	57.8
DER++ (Buzzega et al., 2020)		68.1	33.3	36.7	54.6	69.6	41.9	43.7	59.0
KRT-R (Dong et al., 2023)		75.8	60.0	61.0	68.3	78.0	66.0	65.9	74.3
iCaRL (Rebuffi et al., 2017)	20/class	59.7	19.3	22.8	43.8	65.6	22.1	25.5	55.7
BiC (Wu et al., 2019)		65.0	31.0	38.1	51.1	65.5	38.1	40.7	55.9
ER (Riemer et al., 2019)		60.3	40.6	43.6	47.2	68.9	58.6	61.1	61.6
TPCIL (Tao et al., 2020)		69.4	51.7	52.8	60.6	72.4	60.4	62.6	66.5
PODNet (Douillard et al., 2020)		70.0	45.2	48.7	58.8	71.0	46.6	42.1	64.2
DER++ (Buzzega et al., 2020)		72.7	45.2	48.7	63.1	73.6	51.5	53.5	66.3
KRT-R (Dong et al., 2023)		76.5	63.9	**64.7**	70.2	78.3	**67.9**	**68.9**	75.2
PRS (Kim et al., 2020)	1000	48.8	8.5	14.7	27.9	50.8	9.3	15.1	33.2
OCDM (Liang & Li, 2022)		49.5	8.6	14.9	28.5	51.3	9.5	15.5	34.0
KRT-R (Dong et al., 2023)		75.7	61.6	63.6	69.3	78.3	67.5	68.5	75.1
MULTI-LANE (Ours)	0	**79.1**	**65.1**	62.8	**74.5**	**78.8**	66.0	66.6	**76.6**

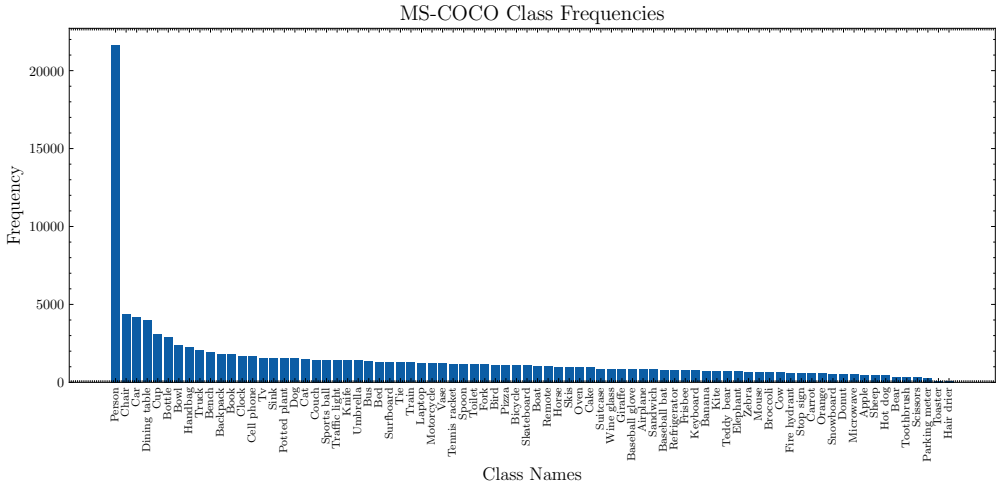
F1 scores, moreover, the difference with CODA-P tends to grow as the number of samples per class lowers. This clearly shows that MULTI-LANE is more robust to long-tailed distributed datasets.

F Token Merging for Summarization

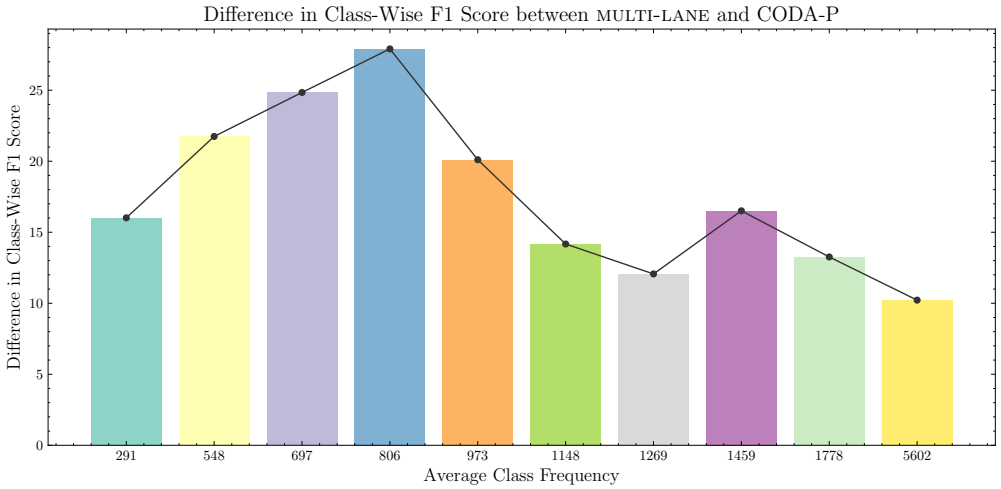
In Fig. 5, we present the performance of MULTI-LANE when replacing Patch Selectors with Token Merging (ToMe). Token Merging (Bolya et al., 2023) works through a bipartite soft matching, where tokens are first split into two sets and then matched based on a similarity measure. The r most similar pairs are merged and concatenated to the remaining ones. This means after applying Token Merging, the length of the resulting patches is halved at most. We aim to reduce the number of summarized patches even further, thus, we applied Token Merging multiple times until the length of the summarized patches was short enough. In particular, we started from the original patch tokens $\boldsymbol{g} \in \mathbb{R}^{(L+1)\times D}$ and applied Token Merging to produce $\bar{\boldsymbol{g}}' \in \mathbb{R}^{L'_s \times D}$, then we applied again ToMe to $\bar{\boldsymbol{g}}'$ to obtain $\bar{\boldsymbol{g}}'' \in \mathbb{R}^{L''_s \times D}$ and so on until the final length L_s is less than a specified threshold. We set the upper bound for L_s to 30. Our objective is to strongly reduce the number of patches while retaining as much information as possible. As for Patch Selectors-generated summarized patches, ToMe ones are dropped after the MSA and calculated in each ViT Block.

G Qualitative Results - Failure Cases

In Section 4, we present some qualitative representations of the attention paid by our Patch Selectors when summarizing patch tokens. In this Section, we report some evident failure cases we encountered when analyzing the heatmaps produced by MULTI-LANE. First, we noticed that Patch Selectors, especially in the last layers, tend to have peaked activations on specific tokens rather than a more uniform one. However, when the object takes up most of the image space (*e.g.* airplane picture in Fig. 10), the activation is rather uniform. When reporting failure cases, we are interested in small objects that do not receive high attention values or the attention is focused on regions not concerning the class MULTI-LANE should look for. In Fig. 10, we show six failure pictures with one corresponding task heatmap. We also show the class for which the Patch Selectors, of the corresponding task, should look. Finally, we remark that a failure in qualitative results does not imply a failure in classification.



(a) Long tail distribution of MS-COCO.



(b) Difference in Class-Wise F1 Score between MULTI-LANE and CODA-P. Classes are grouped based on their frequencies.

Figure 9: MULTI-LANE is more effective than CODA-P in long-tailed distributed datasets.

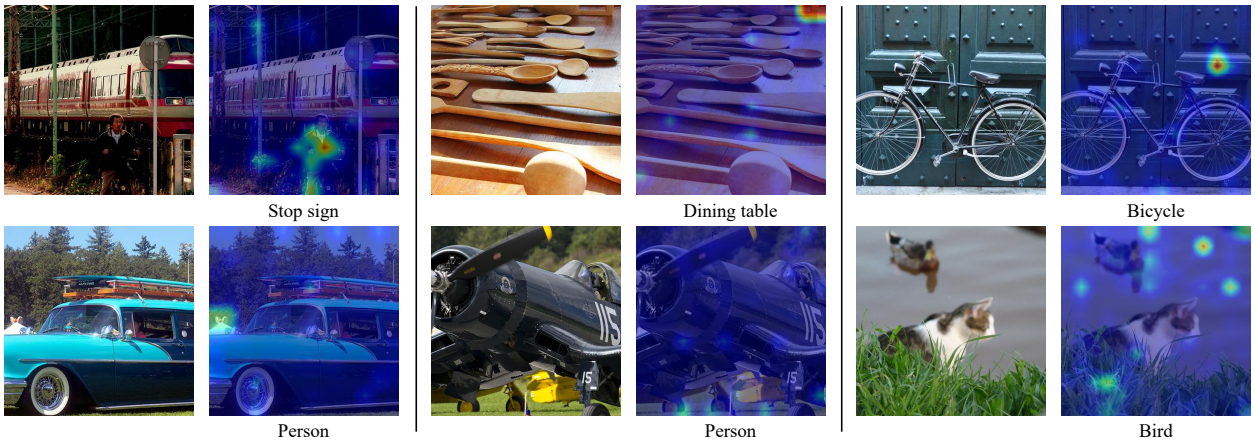


Figure 10: **Qualitative failure cases**. In these pictures, MULTI-LANE either presents peaked activations in wrong regions of the image, or the activations are very low.