Convolutional Prompting meets Language Models for Continual Learning (Supplementary Material)

This supplementary material contains the following.

- Section 1: Finding the right Lambda value.
- Section 2: More details about the Image Based similarity.
- Section 3: Results of different tasks with and without task-similarity.
- Section 4: A closer look into class attribute based task similarity.
- Code for the ConvPrompt approach.

1. Sensitivity Analysis of λ

In this section, we discuss in more detail, about the hyper-parameter search, especially the λ value for weighing the $\mathcal{L}_{norm}$ regulariser. As shown in Fig: 1, the performance of the model increases from $1.00E-4$ to $1.00E-3$ peaks at $1.00E-02$ and saturates around that value, hence the λ value for `ConvPrompt` is chosen as 0.01.

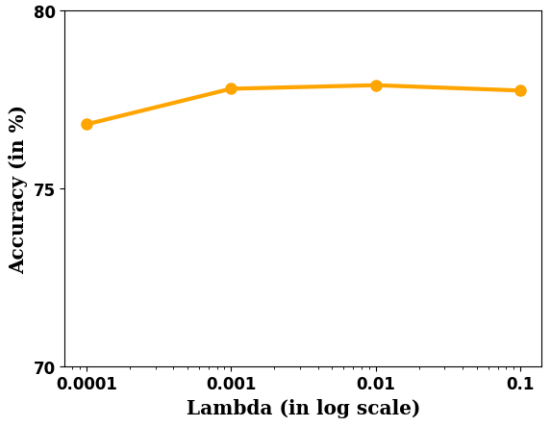


Figure 1. **Average Accuracy ($\mathbf{A_T}$) vs. λ value(Log Scaled) plot for `ConvPrompt`.** The performance peaks at $1.00E-02$ and saturates thereafter.

2. Image-based Similarity Calculation

To calculate image-based task similarity, we fetch the image features of each of the images of the classes in the tasks seen by the model till now. We take class-wise average of these features and store them in a pool of seen classes. Therefore, for each class we have a single embedding representative for all images belonging to that class. For extraction of image features we use the final layer `[cls]` embeddings of the ViT-B/16 model, pre-trained on ImageNet-21K. In order to find the task-wise similarities, we compute the cosine similarity of the embeddings of all classes that had arrived in the previous tasks (0 to $t-1$) with the embeddings of the classes in task t. For each of the classes in task t, their maximum similarity score with any of the previous classes is considered. Finally, the similarity of task t is computed as the maximum similarities averaged over all the classes in task t.

3. With and Without Task Similarity

We report the comparison on the performance and the number of parameters required by the `ConvPrompt` with and without language based task similarity based attribute reduction. The results with and without task similarity remain almost similar, while a 50% reduction in parameters is observed with task-similarity. Furthermore for datasets with very similar tasks, such as CUB-200 [2], task-similarity also improves performance by 1%, by preventing overfitting.

Tasks	Without Sim		With Sim	
	A_T	N_{params}	A_T	N_{params}
Split CIFAR-100	88.91 ± 0.29	$3.72/103.72$	88.87 ± 0.33	$2.2/102.2$
Split ImageNet-R	77.96 ± 0.54	$3.7/103.7$	77.86 ± 0.25	$2.0/102.0$
Split CUB-200	79.25 ± 0.38	$3.68/103.68$	80.2 ± 0.52	$1.8/101.8$

Table 1. **Results with and without task-similarity:** We report the A_T values for 10 task trials for each of the datasets averaged over 5 trials.

4. Attribute similarity - A more closer look

We provide a closer look into our class attribute-based task similarity calculation process with the help of some representative examples from the ImageNet-R [1] dataset, shown

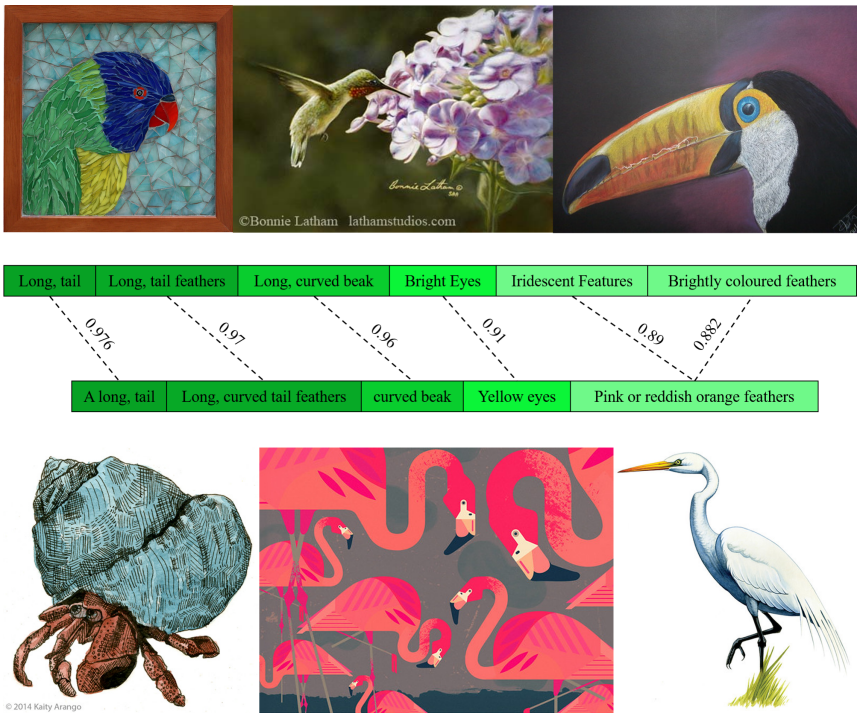
in 2. The attribute matching mechanism has been described in-detail in **Sec. 4.3** of the main paper. For this demonstration, we consider three tasks, with each task containing three classes.

The first example 2a contains two similar tasks with Task 1 containing classes - *lorikeet*, *hummingbird*, *toucan* and Task 2 containing classes - *hermit crab*, *flamingo*, *american egret*. Each attribute in the new task is matched with the most similar attribute from the old tasks. The top matching attributes with the highest similarity scores are shown in 2a. The final similarity scores for these two tasks using these attribute similarity scores are obtained to be 0.86. The second example 2b contains two relatively dissimilar tasks with Task 1 containing classes - *goldfish*, *great white shark*, *hammerhead* and Task 2 containing classes - *school bus*, *schooner*, *shield*. Each attribute in the new task is matched with the most similar attribute from the old tasks. The top matching attributes (with the highest similarity scores) are shown in 2b. The final similarity scores for these two tasks using these attribute similarity scores are obtained to be 0.70. Hence it is evident that our task-similarity technique captures the inter-task similarity well, resulting in a high similarity score for more similar tasks, and a low similarity score for dissimilar tasks.

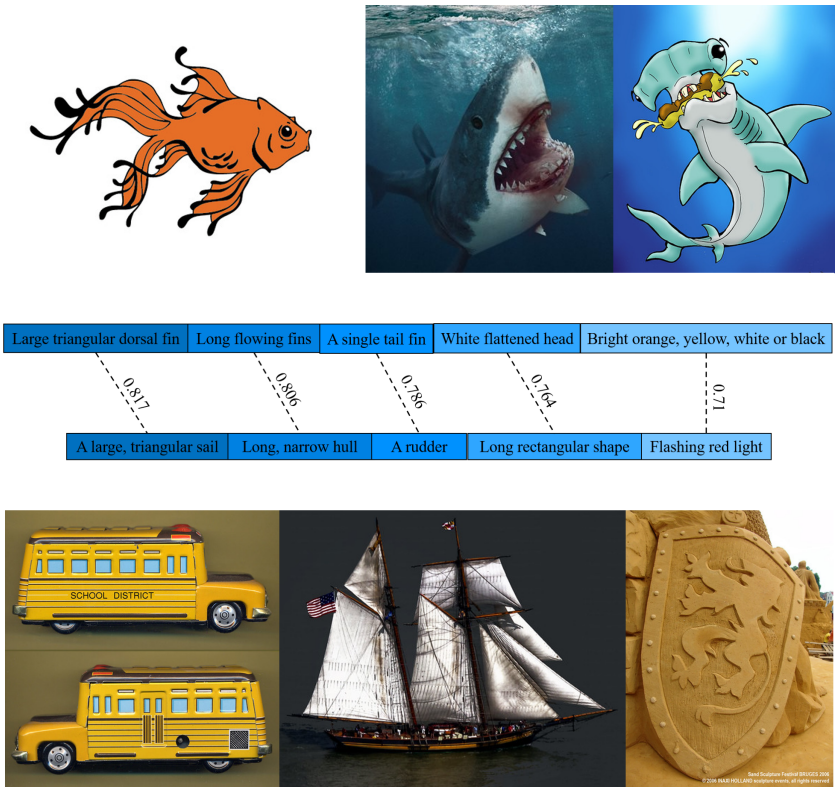
References

[1] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021. 1

[2] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. In *California Institute of Technology*, 2011. 1



(a) **An example of two similar tasks: Task 1** contains the classes *lorikeet*, *hummingbird*, *toucan* and **Task 2** contains the classes *hermit crab*, *flamingo*, and *american egret*.



(b) **An example of two dissimilar tasks: Task 1** contains the classes *goldfish*, *great white shark*, *hammerhead* and **Task 2** contains the classes *school bus*, *schooner*, and *shield*.

Figure 2. **Inter-task attribute based task similarity calculation**: For each attribute of the new task, the most similar attributes in the old tasks are found, and the corresponding cosine similarity values are computed (some of these have been shown above). The mean of all such max similarities of attributes in the new task, gives the overall task similarity.