SUPPLEMENTARY MATERIAL FOR: IMPROVING PROMPT-BASED CONTINUAL LEARNING WITH KEY-QUERY ORTHOGONAL PROJECTION AND PROTOTYPE-BASED ONE-VERSUS-ALL

Anonymous authors
Paper under double-blind review

A APPENDIX

A.1 MORE DETAILS ABOUT OUR ALGORITHM

When training a new task, data is first passed through the pretrained ViT to get the query vectors, $q(\mathbf{x}_t^{tr})$. These queries are then paired with the keys of each task learnt so far to compute their weights, α^i, corresponding to their contribution to the current task. Specifically, the aggregated prompt for task t, $\mathbf{P}_{\mathbf{x}_t^{tr}}$, is the weighted sum of all prompts up to this task. The keys and prompts of old tasks are frozen and only those of the current task are learnt. To tackle the *mismatch* in prompt representation, we explicitly constrain $\mathbf{K}^t$ to be orthogonal to the subspace of queries of old tasks, $\mathcal{Q}^{t-1}$. This is achieve by our *Orthogonal Look-ahead optimization*, which seeks to simultaneously optimizes $\mathbf{P}^t$ and $\mathbf{K}^t_\perp$. Furthermore, we strengthen task classification head distinction by employing the OVA loss, $\mathcal{L}_{OVA}$, whose inputs are the prototypes of old classes and current classes. All the model parameters are learnt in an end-to-end manner.

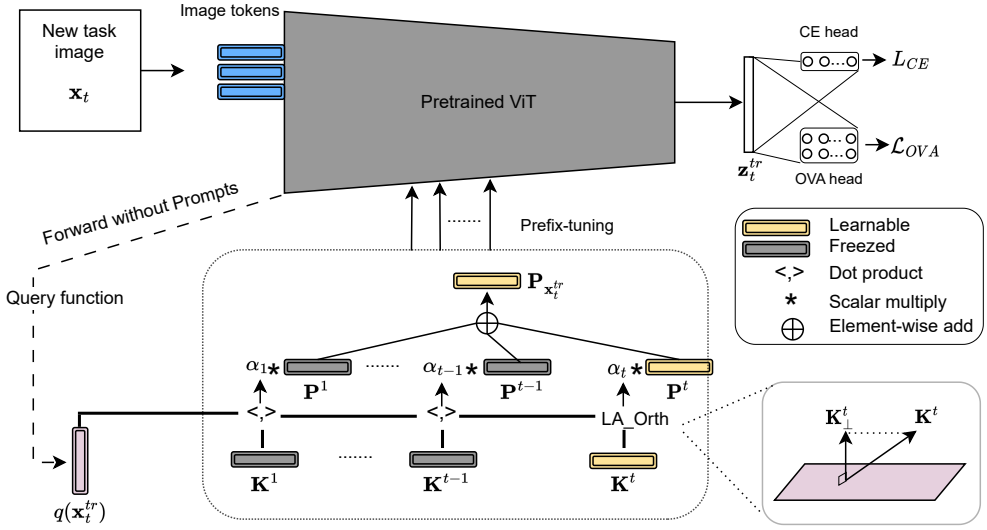

Figure 1: Training framework of KOPPA. Better viewd in color.

A.2 HOW TO UPDATE THE MATRIX $\mathcal{Q}^t$

- In our model, we denote $\mathcal{S}^t$ as the subspace containing the query vectors of task t, and $\mathcal{Q}^t = \bigcup_{i=1}^t \mathcal{S}^i$ as the subspace spanned by query vectors from task 1 to the task t. We do not calculate $\mathcal{S}^t$ directly; instead, we obtain $\mathcal{Q}^t$ from $\mathcal{Q}^{t-1}$ as follow:

We use matrix $\mathbf{Q^t}$ to represent the subspace $\mathcal{Q}^t$, which spanned by a set of query vectors from task 1 to task t: $\{q(\mathbf{x_1}), q(\mathbf{x_2}), ...q(\mathbf{x_t})\}$, and this matrix is used to constraint $\mathbf{K^{t+1}}$ when learning task $t+1$. $\mathbf{Q^t}$ was computed by using SVD to choose the most representative bases. Specifically:

- For $t = 0$, let $\mathbf{R^0} = [q(\mathbf{x_1^0}), q(\mathbf{x_2^0}), ...]$ be the matrix with column vectors as query vectors of task 0. We perform SVD on $\mathbf{R^0} = \mathbf{U_0 \Sigma_0 (V_0)^T}$ and then k new orthogonal bases are choosen, such that $\left\|\mathbf{R^0}_k\right\|_F^2 \geq \epsilon \left\|\mathbf{R^0}\right\|_F^2$, given the threshold $\epsilon \in (0, 1)$ *(k-rank approximation)*. Finally, matrix $\mathbf{Q^0} = [u_1^0, ..., u_k^0]$ is formed with the columns being the selected basis vectors.
- For $t > 0$, let $\mathbf{R^t} = [\mathbf{Q_1^{t-1}}, \mathbf{Q_2^{t-1}}, ..., q(\mathbf{x_1^t}), q(\mathbf{x_2^t}), ...]$ where $\mathbf{Q_i^{t-1}}$ is a column vector of $\mathbf{Q^{t-1}}$. Similarly, performing SVD and k-rank approximation on $\mathbf{R^t}$, we obtain $\mathbf{Q^t}$.

In practice, in order to compute $\mathbf{Q}^t$, we use a set of 200 latent vectors $q(\mathbf{x})$ of task t (and matrix $\mathbf{Q}^{t-1}$, if $t > 1$).

- Furthermore, in our implementation, the latent space has a dimension of 784. Thus, the theoretical maximum size of $\mathbf{Q}$ is 784×784 and we only need to maintain matrix $\mathbf{Q} = \mathbf{Q^{t-1}}$ to constrain $\mathbf{K^t}$ when learning task $t > 1$. In fact, the largest size of $\mathbf{Q}$ on datasets is reported in Table 1:

Table 1: Size of basis matrix $\mathbf{Q}$ after final task on datasets (KOPPA)

Dataset	S-CIFAR-100	S-Imagnet-R-5	S-Imagnet-R-10	S-Imagnet-R-20
Size of $\mathbf{Q}$	768×80	768×86	768×86	768×87

A.3 Discussion of Additional Memory Consumption

Our KOPPA needs to store the orthonormal basis of the subspace $\mathcal{Q}^T$ corresponding to the matrix $\mathbf{Q}^T$ up to the last task T. This matrix has at most dimension of 768×768. In addition, for training the OVA head, we retain 100 prototypes with a dimension of 768 for each task. We view them as additional parameters to the model, which totally cost the additional memory as in Table 2.

Table 2: Additional memory that needs to store additional matrix and prototypes of KOPPA

	S-CIFAR-100	S-ImageNet-R-5	S-ImageNet-R-10	S-ImageNet-R-20
Additional Memory	4.89MB	3.42MB	4.89MB	7.82MB

For S-Imagenet-derived datasets, it is evident that our memory usage does not exceed the storage capacity equivalent to 40 images of the resolution $224 \times 224 \times 3$. Obviously, with the limited number of images for the ImageNet-R-derived datasets, achieving satisfactory results is extremely challenging. For CIFAR100, using $10 \times 100 \times 768 \times 4bytes$ prototypes equals saving 1000 raw images / 10 tasks (i.e, 100 images/tasks) to replay. We apply this raw data replay with (i) KOPPA, and (ii) just CE to update classifier head fr all tasks. The results presented in table 3 shows that replaying features is as effective as replaying raw data. Besides, replaying raw data using traditional learning methods with CE loss to finetune classifier head cannot replace the use of our OVA-based module.

Method	$A_N(\uparrow)$	$F_N(\downarrow)$
CE + OVA (Replay $\mathbf{z}$) - Ours	**97.82 ± 0.80**	**0.43 ± 0.12**
CE + OVA (Replay $\mathbf{x}$)	97.82 ± 0.80	0.43 ± 0.12
CE (Replay $\mathbf{x}$)	93.25 ± 0.73	0.85 ± 0.15

Table 3: Comparison between KOPPA and raw data replay solutions with the same size of buffer.

A.4 Limitations and Potential future work

About the limitations and potential future work, we would like to discuss some points as follows:

- Although this orthogonality constraint ensures (almost) no forgetting and as well as features shift, it could also hinder potential improvement of past tasks, i.e. positive backward transfer. Such cases may happen when the new task contains positively related knowledge for previous ones, hence model could learn prompts which are useful for them. Therefore, an interesting future work might be to decide when positive backward transfer may happen in the context of prompt-based CL.
- In addition, the results in Table 5 in the main paper show that the number of features used to train the OVA-module has a certain influence on the overall performance of the model. Therefore, in the future, we will consider replacing the use of raw data with effective yet flexible solutions that help the model operate more stably.
- Furthermore, when a higher level of data privacy is concerned such that no prototypes or latent features are stored, as they can be inverted to generate original images [1], our method with the OVA-based module cannot work.

A.5 Additional Implementation Details

Baselines. In the main paper, we use LwF (Li & Hoiem, 2017), L2P (Wang et al., 2022b), DualPrompt (Wang et al., 2022a), CODA (Smith et al., 2023), Joint training, Fine-tuning (FT) and FT++ as data-free CIL baselines. We also add a typical replay-based method, ER (Chaudhry et al., 2019), with a buffer of size 5000.

(1) **LwF**: is a regularized-based method that enforcing the outputs of current model on new task data to be similar to those of previously trained model. This is be done using KL divergence between the two distributions outputs.
(2) **ER**: is e memory-based approach that save a fixed-size buffer of old data which is randomly selected during training or at the end of each task. When the buffer is full, reservoir sampling is adopted to decide which buffer sample will be replaced.
(3) **FT**: usually known as the lower bound for a CL method as it only trains the networking using current data without any forgetting mitigation strategy. In particular, we let FT modify classifier heads of all classes so far.
(4) **FT+**: a simple improvement of FT that only modify classifier heads of classes in the current task.
(5) **L2P++** extends L2P by using pre-fix tuning (Li & Liang, 2021), rather than prompt-tuning as in its original implementation. The first attention layer applied in this baseline. Prefix-tuning is chosen because it has beed reported to improve performance over prompt-tuning Wang et al. (2022a).
(6) **Deep L2P++** injects pre-fix tuning to four more attention layers than L2P++, resulting in a 5-layer-prefix-tuning method, which is similar to DualPrompt and CODA.

Protocols. We use the ImageNet-pretrained ViT-B/16 backbone (Dosovitskiy et al., 2021) for all methods. We learn model parameters with Adam optimizer, learning rate 0.001 and cosine scheduler. We implement baselines following their reported default hyper-parameters. For our method, similar to DualPrompt, we insert a component of 100 prompts with length 8 to the first 5 attention layers. We remove the attention masks in CODA, so our model has fewer trainable parameters than CODA. We train the model using 20 and 50 epochs for Split Cifar-100 and Split ImageNet-R, respectively. For the two dataset, the numbers of look-ahead epochs are 10 and 15 in that order. To effectively learn OVA head, we save a buffer size of 100 prototypes per tasks, and to find the subspace $\mathcal{Q}$, we randomly sample 200 data points of the current task to perform SVD.

Metrics. We use two commonly used metrics: Average Accuracy and Average Forgetting at the end of the training sequence. Specifically, denote acc_t^i is the accuracy of task t after the model has been trained with task i. Then Average Accuracy, $A_N(\uparrow)$, and Average Forgetting, $F_N(\downarrow)$, are respectively defined as:

$$A_N = \frac{1}{T}\sum_{t=1}^{T} acc_t^T, \quad F_N = \frac{1}{T-1}\sum_{i=1}^{T-1} \max_{j' \in \{1,\cdots,T-1\}} (acc_i^{j'} - acc_i^T).$$

Intuitively, a good CL model should maintain high overall results on all tasks while minimizing performance drop of previous tasks, high A_N and low F_N. Between the two, the former is more informative in terms of balancing plasticity and stability, because a low value of accuracy A_N can come with a low value of forgetting F_N, e.g. when the model underfits and thus forgets less.

A.6 EXTENDED EXPERIMENTAL RESULTS

A.6.1 RESULTS ON DOMAINNET

Table 4 presents average accuracy and forgetting of our method KOPPA and baselines on DomainNet dataset (Peng et al., 2019). Clear improvement over previous prompt-based methods can be observed with around 10 % increase in accuracy compared with the runner up, CODA-P, and with the least forgetting.

Table 4: Average accuracy and forgetting (%) of methods on DomainNet

Method	$A_N(\uparrow)$	$F_N(\downarrow)$
JOINT	79.65	
ER (5000)	58.32 ± 0.47	26.25 ± 0.24
FT	18.00 ± 0.26	43.55 ± 0.27
FT++	39.28 ± 0.21	44.39 ± 0.31
LwF.MC	74.78 ± 0.43	5.01 ± 0.14
L2P++	69.58 ± 0.39	2.25 ± 0.08
Deep L2P++	70.54 ± 0.51	2.05 ± 0.07
DualPrompt	70.73 ± 0.49	2.03 ± 0.22
CODA-P	73.24 ± 0.59	3.46 ± 0.09
KOPPA (Ours)	**84.14 ± 0.62**	**0.23 ± 0.10**

A.6.2 MORE ON THE ROLE OF KEY ORTHOGONALITY CONSTRAINT IN REDUCING FEATURE SHIFTS

As mentioned in the main paper, the primary aim of the orthogonal projection component is to minimize feature shifts rather than pursuing a general accuracy improvement. The effectiveness of this strategy is evident in the results presented in Table **??** and Figure **??**, demonstrating the superior ability to avoid shifts compared to CODA. Moreover, the outcomes in Table 5 highlight that enhanced shift avoidance plays a crucial role in enabling KOPPA to achieve superior results over CODA, particularly when combined with our proposed OVA-based module, especially on Domainnet and S-Imagenet-10 with gaps greater than 3%.

Methods	S-CIFAR-100	S-Imagnet-R-5	S-Imagnet-R-10	S-Imagnet-R-20	DomainNet
Deep L2P++ + OVA	95.53 ± 0.83	84.86 ± 0.39	89.23 ± 0.77	91.92 ± 0.94	80.01 ± 0.54
DualP + OVA	96.06 ± 0.75	85.28 ± 0.55	88.11 ± 0.82	92.13 ± 0.84	79.83 ± 0.52
CODA + OVA	96.88 ± 0.74	85.32 ± 0.51	88.02 ± 0.65	92.10 ± 0.98	79.76 ± 0.55
KOPPA	**97.82 ± 0.80**	**86.02 ± 0.42**	**91.09 ± 1.53**	**92.89 ± 1.2**	**84.14 ± 0.62**

Table 5: Performance of other baselines when using our OVA-based module

A.6.3 OVA VERSUS OTHER CLASSIFIERS

First, it is worth noting that OVA loss is a well-established and widely recognized loss function within the community (Rifkin & Klautau, 2004; Saito & Saenko, 2021). In this work, one of our contributions is finding out the necessity to observe all classes, then introducing a module based on the OVA loss, and how to use it to significantly improve CL performance. The high effectiveness of this module for CL should be significant for the CL literature.

Second, to illustrate the effectiveness of our proposed OVA-based module in comparison with alternative solutions, such as using a prototype-based classifier, we have conducted extended experiments. In specific:

- Prototype-based classifier (I): remove OVA head; calculate and store prototypes of classes to give predictions.
- Prototype-based classifier (II): remove OVA head, and use old prototypes and current data to learn CE head together with the backbone; calculate and store prototypes of classes to give predictions.

- CE † + CE: replace OVA head with an additional CE head and train it by using prototypes. Give predictions in the similar way that described in the main paper.

The results in Table 6 demonstrate

- (i) the essence of using a module which can observe all classes (i.e., * CE † + CE is better than Just CE);
- (ii) using just prototype to predict is a not a good option due to the indistinguishable distributions of classes, which may be the result of the uncontrolled overlapping between features from different tasks;
- (iii) our OVA module is a superior choice to all the above alternatives.

Table 6: Comparison between KOPPA's classifier vs. alternative solution. * indicates approaches that train an additional classification head using prototypes from all tasks.

Methods	S-CIFAR-100	S-Imagnet-R-5	S-Imagnet-R-10	S-Imagnet-R-20
* OVA + CE (Ours)	**97.82 ± 0.80**	**86.02 ± 0.42**	**91.09 ± 0.53**	**92.89 ± 1.25**
Just CE	86.28 ± 0.81	76.32 ± 0.45	75.62 ± 0.43	72.42 ± 1.20
* CE † + CE	94.71 ± 0.85	85.08 ± 0.51	90.02 ± 0.54	90.92 ± 0.88
* Prototype-based classifier (I)	67.28 ± 0.77	0.38 ± 0.65	4.75 ± 0.62	41.85 ± 0.92
* Prototype-based classifier (II)	69.75 ± 0.75	0.49 ± 0.54	4.91 ± 0.62	43.42 ± 0.85

A.6.4 WHY DOES KOPPA SURPASSES JOINT?

In Tables 1 (in the main paper) and 4, it can be seen that KOPPA outperforms JOINT by a large margin. In this part, we will present in-depth reasons for this.

First, JOINT is better than other baselines, but we should not always consider JOINT as their upper bound.

- **Why is JOINT better than other baselines?**
  JOINT employs a training strategy where the model learns from data of all tasks simultaneously, treating them as a single task. Thus, JOINT exhibits two key advantages over other baselines: (i) no forgetting or feature shifts, and (ii) the model learns the relationships between observed classes, aligning the corresponding features effectively and minimizing misclassification. This might have been the reason why JOINT has higher results than the remaining baselines.
- **Why should we not consider JOINT as the upper bound of other baselines?**
  (i) From the code released by CODA's authors, we found that JOINT is trained on a single task, with data from all tasks. Specifically, the pre-trained model and a classification head are fine-tuned without any additional parameters such as prompts. Therefore, basically, the design of the backbone in JOINT and the other prompt-based incremental learning methods mentioned is completely different.
  (ii) Moreover, taking S-CIFAR-100 as an example, while the JOINT's model must learn to classify 100 different classes simultaneously, that of CODA's (or DualP or L2P's) only needs to learn to classify 10 classes per task. Therefore, when comparing JOINT with baselines in such a small task, JOINT may be outperformed by these baselines in terms of performance. If these methods can effectively avoid forgetting and can utilize task ID information, the final average accuracy A_N can approach the average of accuracies of those small tasks which are learned independently. In other words, it is possible for JOINT to be surpassed.

Second, the main reasons that KOPPA surpasses JOINT by such a large margin:

Firstly, KOPPA's strategies enable the model to achieve the two advantages mentioned above (like JOINT) over other baselines: (i) reducing forgetting better by the proposed key-query strategy, (ii) making use of task ID information when using the OVA-based module in prediction. *Secondly*, by

sequentially learning sub-tasks, KOPPA can achieve higher accuracy on each of these small tasks than JOINT (on one task of a bigger dataset). Combining the above arguments, it is convincing that KOPPA can surpass JOINT.

Third, we would like to provide experimental evidences. Specifically, Table 7 shows the accuracy of each task (small task) of KOPPA on S-CIFAR-100. Moreover, the corresponding triggering rate is 100% for all tasks, as shown in Figure 3 of the main paper. This implies that the final average accuracy obtained is the average of these 'sub-accuracies,' which amounts to 97.99% — higher than JOINT's 89.3%.

Task	1	2	3	4	5	6	7	8	9	10
Accuracy	99.11	97.78	97.44	98.11	98.22	97.78	96.22	97.44	98.56	99.22

Table 7: Accuracy of each task, on S-CIFAR-100 (KOPPA)

A.6.5 Sensitivity to Prompt Hyper-parameters

We follow (Smith et al., 2023) to set the number of prompts used for each task and the prompt length for fair comparison. Figure 2 shows the changes of average accuracy of KOPPA and CODA when the size of prompt pool and the length of prompts vary. It can be seen that KOPPA consistently outperforms CODA in all cases, and additionally, KOPPA's performance tends to increase when increasing the pool size.

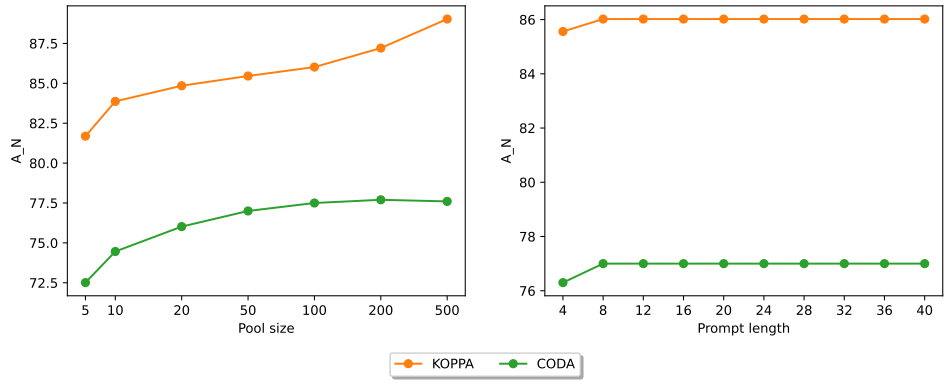

Figure 2: Average Accuracy of CODA and KOPPA with varied prompt pool sizes and prompt lengths.

A.6.6 Comparison with HiDE and RanPAC

Among methods using the same settings and pretrained ViT as the backbone, HiDE (Wang et al., 2023) and RanPAC (McDonnell et al., 2023) recently stand out as the two latest state-of-the-art methods, both featuring interesting ideas and impressive results.

HiDE introduces a technique for effective representation learning, employing a contrastive regularization strategy in the form of hierarchical constraints. This approach leverages both instructed and uninstructed representations, thereby enhancing the quality of the prompt-based continual learning (CL) model. Similar to KOPPA, HiDE preserves information from old tasks through feature vector encoding to apply constraints when learning new tasks to improve prediction. However, HiDE does not address the issue of feature shifts: uninstructed representations might inadvertently select an incorrect task id. This can lead to the combination of incorrect prompts with the pre-trained backbone, resulting in uncontrolled representations, which could negatively affect the final classification quality.

Unlike HiDE and KOPPA, which are categorized under the strategy of Prompting in transformer networks, RanPAC belongs to the Class-Prototype (CP) accumulation category. RanPAC offers a comprehensive and insightful analysis of CP-based methods and introduces a solution involving a Random-Projection (RP) layer with frozen, untrained weights. This layer enhances the latent

Table 8: Average Accuracy of RanPAC, HiDE and KOPPA on serveral bechmarks

Methods	S-CIFAR-100	S-ImageNet-R-5	S-ImageNet-R-10	S-ImageNet-R-20
RanPAC	92.20	79.90	77.90	74.50
HiDE	93.02	77.82	77.12	75.03
KOPPA (Ours)	**97.82**	**86.02**	**91.09**	**92.89**

space from pre-trained models significantly. However, relying solely on the pre-trained model may hinder the model's adaptability and its ability to learn specific knowledge from new tasks, potentially limiting the method's effectiveness.

Next, we would like to provide experimental results comparing these two methods with KOPPA. The RanPAC's results are directly obtained from the original paper as we follow the same pretrained ViT architecture, while those of HiDE are reproduced using the same pretrained backbone as KOPPA and RanPAC.

The results, illustrated in Table 8, demonstrate KOPPA's superiority over these two innovative methods.

A.7 What is the classification sub-head and the problem of wrongly triggering sub-head?

We all know that in CIL setting for classification tasks, we usually use a common prediction head. However, when we consider that the classification head can be divided into ***subheads corresponding to tasks***, then in a model where feature shift happens, there is a risk of incorrectly triggering these task-specific classification heads.

Taking the experiment on S-CIFAR100-10 as an example, we have an output of size 100 for 100 classes/10 tasks, which is returned from the common head h_θ. At that time, we can consider subhead h_θ^1 for task 1, which returns the first 10 components of that output; h_θ^2 for task 2, which returns the next 10 components of the output; and similarly for the remaining tasks. Assume that sub-heads h_θ^t are only learned to fit with $f^c_{\Phi,P}(\mathbf{x}^{tr})$ of sample of task t. Because $f^e_{\Phi,P}(\mathbf{x}^{te})$ is different from $f^c_{\Phi,P}(\mathbf{x}^{tr})$ due to feature shift so that when it interacts with these sub-heads, these heads can return uncontrollable output, leading to incorrect answers. That is, sample $\mathbf{x}$ from task t could yield the highest probability prediction at the class belonging to the subhead corresponding to task $j \neq t$.

REFERENCES

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

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

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Mark D McDonnell, Dong Gong, Amin Parveneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *arXiv preprint arXiv:2307.02251*, 2023.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1406–1415, 2019.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5: 101–141, dec 2004. ISSN 1532-4435.

Kuniaki Saito and Kate Saenko. Ovanet: One-vs-all network for universal domain adaptation. In *Proceedings of the ieee/cvf international conference on computer vision*, pp. 9000–9009, 2021.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11909–11919, June 2023.

Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=9XieH21Tlf.

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 *European Conference on Computer Vision*, pp. 631–648. Springer, 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 139–149, 2022b.