Controlling Forgetting with Test-Time Data in Continual Learning

Vaibhav Singh[1,2] * **Rahaf Aljundi**[3] **Eugene Belilovsky**[1,2]
[1]Concordia University [2] Mila [3] Toyota Motor Europe

Abstract

Foundational vision-language models excel in various tasks but require updates as new tasks or domains emerge. Current Continual Learning (CL) methods, which focus on supervised training, often suffer from significant forgetting, performing worse than the original models in zero-shot scenarios. This work proposes leveraging test-time, unsupervised data in a self-supervised manner to refresh the model's memory of previously learned tasks, minimizing forgetting without additional labeling. By introducing a student-teacher framework with gradient-based sparse parameter updates, the approach enhances performance on prior tasks and reduces reliance on offline memory buffers, effectively improving continual learning outcomes.

1 Introduction

Foundation models in computer vision have shown impressive performance on various down stream tasks and domains which renders them a key building block of various solutions including generative vision language models Chen et al. [2023], Bommasani et al. [2021]. Despite the increased attempts to efficiently improve foundational models performance on new streams of data Wang et al. [2022c], Smith et al. [2023], Zhou et al. [2023], Zhang et al. [2023a], Goyal et al. [2023], forgetting is still a significant problem in applications of continual learning Wang et al. [2024], Prabhu et al. [2023]. We argue that an important factor is to continuously learn irrespective of whether supervision is provided or not, however most works focus solely on training in distinct supervised sessions, while the model remain passive and frozen at test-time.

We consider a scenario where a model is continually trained on supervised datasets, while in-between receiving the supervised datasets, unsupervised data becomes available during the model deployment that can be used for controlling forgetting, as demonstrated in 2. In this work we constrain the unsupervised adaptation to be online, to allow a practical computational overhead. We propose an effective approach based on student-teacher models with sparse parameters selection based on gradient values. Student and teacher models suggest labels for test-data and the predictions from the most confident model is used to update the student model, where the teacher is updated in an exponential moving average adding a stability component to the learning process. We show that such a simple approach achieves significant improvements on all studied sequences. Our approach is stable in class incremental learning(CIL) especially in the challenging setting where no replay buffers are used, which in many cases can be a critical bottleneck.

To the best of our knowledge, we are the first to explore how test-time data can be leveraged in a continual learning setting to reduce forgetting. We consider the foundation model CLIP Radford et al. [2021] for our experiments since it has been shown to encompass an extensive knowledge base

*Correspondence to: vaibhav.singh@mila.quebec, eugene.belilovsky@concordia.ca, rahaf.al.jundi@toyota-europe.com

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

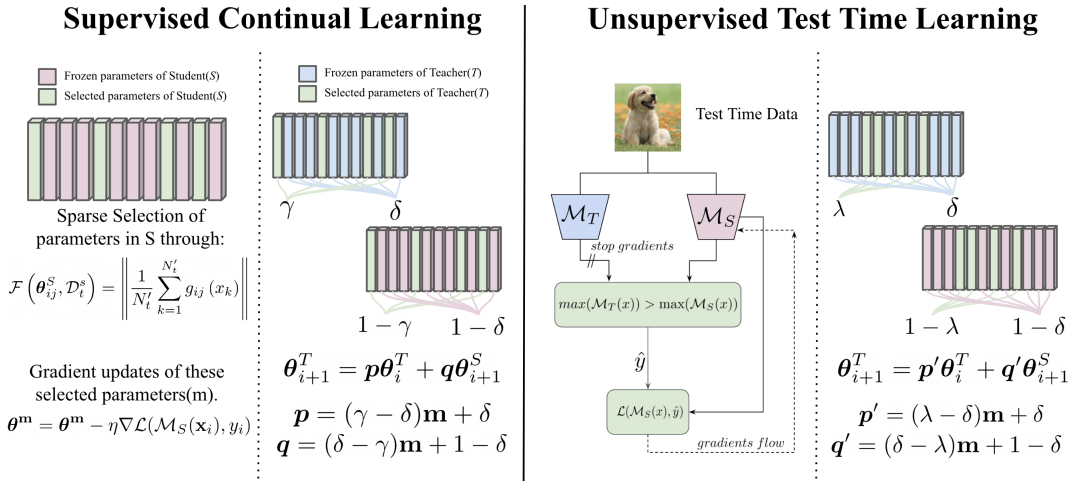

Figure 1: Our method, DoSAPP, uses teacher-student models ($\mathcal{M}_T, ; \mathcal{M}_S$) in two phases. In the supervised continual learning phase, $\mathcal{M}_S$ selects and trains sparse parameters, while $\mathcal{M}_T$ is updated via weighted smoothing with dual momentum terms. During the unsupervised test phase, $\mathcal{M}_S$ adapts using pseudo-labels from $\mathcal{M}_T$ - $\mathcal{M}_S$ logits comparison. This approach maintains generalization and adaptability across tasks.

and offer remarkable transferability Rasheed et al. [2023], Pei et al. [2023]. It undergoes through supervised and unsupervised sessions, leveraging the unsupervised data to control forgetting.

2 Related Work

Due to the abundance of powerful pre-trained models Radford et al. [2021], Oquab et al. [2023], Brown et al. [2020] continual learning that begins with a pre-trained model is becoming a popular paradigm. Recent methods Koh et al. [2022], Boschini et al. [2022] have utilised a Teacher-Student framework for knowledge distillation on previous seen tasks. But, these methods utilise an additional buffer to mitigate catastrophic forgetting. This often entails significant memory Zhou et al. [2022], Prabhu et al. [2023]. Additionally, such methods often face an outdated logit problem, as the memory-stored logits are not updated to preserve information on previous tasks. Boschini et al. [2022] addresses this issue by updating logits stored in the past using task boundary information (e.g., input's task identity) during training, but it may not always be available, especially in task-free CL setups. However, foundation models Radford et al. [2021], Oquab et al. [2023] often have a reasonable initial performance on novel tasks, indicating some pre-existing knowledge relevant to these tasks. Zhang et al. [2023b] utilises this property and preservers generic knowledge by modifying only a small set of parameters based on gradient scoring mechanism. But this method also suffers from recency bias since the gradient scores are computed for current task and only those sparse parameters are updated based on current task scores. Moreover, none of the methods utilise test data in continual learning scenario, and leave a strong potential for self supervised techniques to capture robust feature representations.

3 Methodology

3.1 Setting

We consider a class incremental scenario(CIL) where a sequence of supervised datasets $[\mathcal{D}_1^s, \mathcal{D}_2^s, ...\mathcal{D}_T^s]$ drawn from different distributions are observed at incremental training sessions t ranging from 0 to T, where $\mathcal{D}_t^s = (\mathbf{x}_i^t, y_i^t)_{i=1}^{N_t}$ is the t incremental session with N_t instances as further described in Appendix A.1. We further note that although supervised phases may permit multiple passes through the data until convergence, it would be impractical to collect unsupervised data in production and then perform adaptation on it, we thus restrict the unsupervised phase to be in the online setting Sun et al. [2020], Jang et al. [2022], Cai et al. [2021]. This is especially important in cases were data privacy is important e.g., assistant robot in a private smart home environment.

3.2 DoSAPP: Double Smoothing via Affine Projected Parameters

We propose a simple yet effective method for continual test-time learning, Double Smoothing via Affine Projected Parameters aka DoSAPP. Our approach combines two key components: 1) sparse and local updates: to reduce forgetting, maintain generalization, and ensure efficient updates, and 2) teacher-student framework to promote stability in online updates and minimizes forgetting. In the continual test time learning we can identify two distinct phases of learning as outlined in the following.

Phase 1: Continual Learning Supervised Training with Sparse Selected Parameters

Our goal is to quickly acquire new knowledge without forgetting previously learned information during both training and testing. To achieve this, we update only a small subset of selected parameters. Based on findings from Zhang et al. [2023b], updating relevant parameters in pre-trained models like CLIP minimizes forgetting. Additionally, Geva et al. [2020] suggest that MLP blocks in transformers function as key-value neural memories, with the first layer acting as pattern detectors. Therefore, we update only the top-K parameters from the first MLP layer in each transformer block, keeping the rest frozen to ensure efficient training and retention of prior knowledge. The complete algorithm can be found in A.2, and is clearly depicted in Figure 1.

Weighted exponential smoothing with dual momentum

After each gradient update step(i) for $\mathcal{M}_S$, parameters of $\mathcal{M}_T$ are updated by EMA of the student model parameters. Typically, EMA is governed by

$$\boldsymbol{\theta}_{i+1}^T = \delta \boldsymbol{\theta}_i^T + (1-\delta)\boldsymbol{\theta}_{i+1}^S \tag{1}$$

where δ is the smoothing parameter. Further it has been shown in (Tarvainen and Valpola [2017], Oquab et al. [2023], Koh et al. [2022]) that setting δ to a high value(eg 0.998), maintains a stable teacher model that can be considered as a strong reference for past tasks $\{0, \ldots, t-1\}$. But updating the teacher model with single smoothing parameter in case where parameters are masked creates a dissonance and increases forgetting because all the parameters are updated with equal importance, disregarding those parameters which are selected by the gradient scoring function(where $[\mathbf{m}_{ij} = 1]$). To account for masking, we modify Eq 1 as

$$\boldsymbol{\theta}_{i+1}^T = \boldsymbol{p}\boldsymbol{\theta}_i^T + \boldsymbol{q}\boldsymbol{\theta}_{i+1}^S \tag{2}$$

where $\boldsymbol{p}$ and $\boldsymbol{q}$ denote the smoothing parameters for the teacher and student model respectively, and can be computed as

$$\begin{aligned} \boldsymbol{p} &= (\gamma - \delta)\mathbf{m} + \delta \\ \boldsymbol{q} &= (\delta - \gamma)\mathbf{m} + 1 - \delta \end{aligned} \tag{3}$$

where $\gamma < \delta$. This means that the selected parameters of the teacher model ($[\mathbf{m}_{ij} = 1]$), move little bit faster towards the student model as compared to the frozen parameters(where $[\mathbf{m}_{ij} = 0]$). As such, parameters where $[\mathbf{m}_{ij} = 0]$ will move at a slow rate of δ and unmasked parameters would be updated with γ. When $\gamma = \delta$, the weighted scheme becomes EMA with single smoothing parameter. A detailed proof is given in appendix A.3.

Phase 2: Unsupervised Test Time Learning(TTL)

After supervised training is completed, both $\mathcal{M}_T$ and $\mathcal{M}_S$ are deployed for Test Time Learning(TTL). We consider teacher($\mathcal{M}_T$) and student($\mathcal{M}_S$) models as two experts on different data distributions, the $\mathcal{M}_S$ on the most recent and the $\mathcal{M}_T$ on previous sessions distributions.

Following Hendrycks et al. [2019], we accept the pseudo label of the selected expert. Formally the pseudo label can be calculated as follows:

$$\hat{y} = \begin{cases} \hat{y_T} & \text{if } l_T \geqslant l_S \\ \hat{y_S} & \text{otherwise.} \end{cases} \tag{4}$$

where $\hat{y}$ is the accepted pseudo label and $l_T = \max(\mathcal{M}_T(\mathbf{x}))$ and $l_S = \max(\mathcal{M}_S(\mathbf{x}))$ are the maximum logit score for teacher and student model respectively, and similarly $\hat{y_T} = \arg\max(\mathcal{M}_T(\mathbf{x}))$

and $\hat{y_S} = \arg\max(\mathcal{M}_S(\mathbf{x}))$ are the pseudo labels by teacher and student models respectively. During test-time training the student model $\mathcal{M}_S$ is updated by minimising CLIP contrastive loss given pseudo label $\hat{y}$. In realistic settings, often multiple iterations on test data is not always possible, for eg, a streaming data pipeline. We too mimic this setting, where the entire data is processed only once during TTL phase.

Similar to the above mentioned supervised phase, we also here apply sparse local updates to $\mathcal{M}_S$. However, estimation of masks based on the online data might be noisy, and largely reduce the efficiency as gradients of all parameters must be estimated for each mini batch of test samples. To overcome this, and following the assumption that test data are drawn from the distributions of all previous tasks, we leverage the masks estimated for previous tasks. We accumulate a union of the binary masks ($\mathbf{m}_u$) over all the previously seen tasks t such that $\mathbf{m}_u = \mathbf{m}_1 \cup \mathbf{m}_2 \cup .....\mathbf{m}_t$. To maintain the same sparsity level ($c = 0.1$) of performed updates, we further select the same top-K (K=c) most relevant parameters, from these new masked $\mathbf{m}_u$ parameters based on their previously computed gradient scores.

Momentum(γ, λ)	Aircraft		
	Acc. (↑)	F.(↓)	FTA. (↑)
0.9999, 0.9999	23.99	18.36	12.15
0.5, 0.9	38.41	3.27	37.64
0.7, 0.9	37.22	3.05	37.72
0.8, 0.9*	**39.40**	**2.61**	**38.13**
0.8, 0.6	37.06	5.12	29.63
0.8, 0.5	32.95	3.40	26.33

Table 1: Effect of Momentum(γ, λ) on Average Accuracy(Acc in %), Average Forgetting(F.) and First Task Accuracy(FTA.) *0.9999, 0.8, 0.9 have been used in the main results.

Finally $\mathcal{M}_T(\boldsymbol{\theta}^T)$ is updated using the same dual momentum scheme, but with different smoothing vectors $\boldsymbol{p}', \boldsymbol{q}'$ as:

$$\boldsymbol{\theta}^T_{i+1} = \boldsymbol{p}'\boldsymbol{\theta}^T_i + \boldsymbol{q}'\boldsymbol{\theta}^S_{i+1} \tag{5}$$

where $\boldsymbol{p}' = (\lambda - \delta)\mathbf{m} + \delta$ and $\boldsymbol{q}' = (\delta - \lambda)\mathbf{m} + 1 - \delta$. In TTL phase, the momentum parameter λ is kept such that $\gamma < \lambda < \delta$. This means that $\boldsymbol{\theta}^T$ moves more slowly in direction of $\boldsymbol{\theta}^S$ during TTL phase as compared to the supervised phase. As we encounter frequent, and possibly noisy, online updates, stability is better insured by a slower pace of movements towards student parameters. We show the sensitivity of our method on choice of momentum values λ, δ in Table 1. A high δ has been chosen to keep the Teacher model stable as shown in Tarvainen and Valpola [2017], Oquab et al. [2023], Koh et al. [2022]. It can be clearly seen that when $\gamma = \lambda$(single momentum EMA), the performance significantly drops. DoSAPP is less sensitive to on chocie of γ, but it highly depends on λ. We can also see that as $\lambda < \gamma$, the performance again drops. The algorithm can be fully understood as given in 1

4 Experiments

4.1 Setup

Architecture: We apply DoSAPP to vision-language classification tasks, given their relatively robust knowledge measurement in such tasks. CLIP-ViT/B-16 Radford et al. [2021], is used as backbone. We report the accuracies recorded by the Teacher model. We refer to Zhang et al. [2023b] for hyperparameters selection other than dual momentums, which are given in Appendix A.6. Detailed information on datasets, and evaluation metrics can be referred to A.4

4.2 Results

We compare various baselines with our proposed method in Table 2 for class incremental learning (CIL). In addition to these methods, we also compare against self-labeling (SL), where the pseudo labels come from the model itself without a teacher-student framework. Without ER, DoSAPP achieves state-of-the-art results across all five datasets, demonstrating the effectiveness of test-time data for improving transferability and retaining prior knowledge. Even without ER, DoSAPP performs comparably to methods using ER. While SPU+ER uses a large buffer (1000), we show that DoSAPP with a smaller buffer (ER=200) still outperforms most baselines, except on GTSRB. We also consider the case where we have a long sequence of tasks each to be trained in a class incremental fashion. For these experiments, we combined the 10 tasks of Aircraft data Maji et al. [2013] and 10 tasks of Cars

Method	Aircraft		Cars		CIFAR100		CUB		GTSRB	
	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)
CLIP-Zeroshot Radford et al. [2021]	24.45	-	64.63	-	68.25	-	55.13	-	43.38	-
Finetune Goyal et al. [2023]	18.63	39.93	51.64	25.65	46.26	37.78	45.74	26.62	21.76	55.48
SL	10.81	50.81	23.49	30.42	38.03	42.67	28.60	33.82	5.14	62.31
MAS Aljundi et al. [2018]	33.69	27.50	69.43	9.18	63.88	21.16	61.72	12.05	42.04	25.38
L2P Wang et al. [2022c]	32.20	21.73	67.04	11.22	67.71	18.81	64.04	6.82	**75.45**	2.68
DualPrompt Wang et al. [2022b]	26.61	17.20	63.30	18.67	61.72	19.87	64.38	12.94	69.65	8.43
SLCA Zhang et al. [2023a]	29.40	11.45	62.65	4.42	70.03	0.19	53.87	7.75	46.01	0.83
ZSCL Zheng et al. [2023]	30.96	15.65	67.79	8.27	**80.50**	**1.05**	61.09	7.69	62.92	13.54
SparseCL Wang et al. [2022a]	31.95	19.77	71.57	5.38	69.35	15.23	62.50	9.66	48.99	24.91
SPU Zhang et al. [2023b]	30.94	28.36	69.41	16.91	58.80	26.37	62.31	7.2	43.06	19.16
DoSAPP	**39.14**	**12.55**	**74.87**	**-0.74**	79.16	7.73	**68.17**	**2.15**	72.33	**1.02**
ER methods										
ER French [1999]	41.42	31.38	69.08	16.42	82.86	3.41	64.07	17.72	96.28	-7.48
ER + LWF Li and Hoiem [2017]	36.08	18.12	72.56	4.04	74.32	8.16	65.11	5.90	53.56	11.86
ER + PRD Asadi et al. [2023]	37.11	17.35	74.08	3.75	79.66	3.10	65.92	6.55	63.00	12.44
SPU + ER=1000	44.43	14.42	77.51	3.26	83.99	-0.39	71.51	4.84	**94.25**	-7.87
DoSAPP + ER=200	**47.32**	**8.10**	**79.17**	3.92	**88.41**	-1.96	**74.39**	**2.77**	83.67	1.92

Table 2: Acc(Average Accuracy) % (↑) and F. (Forgetting)↓ of different methods using CLIP with trainable vision and text encoders, without any Replay Buffer in CIL scenario. DoSAPP can achieve positive backward transfer - forgetting is negative on Cars data. All experiments are mean of 5 experiments with random seeds. std. is not shown for ease of reading and space constraint.

Components of DoSAPP	Aircraft		Cars		CIFAR100		CUB		GTSRB	
	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)
Only Teacher-Student	30.12	3.50	67.72	3.66	77.82	5.17	62.67	4.11	53.57	5.38
+ sparse params	34.16	8.61	69.42	3.41	71.93	8.24	66.32	3.98	55.32	5.81
dual momentum+mask union*	**39.14**	2.55	**74.87**	-0.74	**79.16**	7.73	**68.17**	2.15	**72.33**	1.02
+ imbalanced TTL	35.99	5.22	72.68	6.38	75.70	9.81	64.84	3.73	68.17	5.63

Table 3: Acc(Average Accuracy) % (↑) and F. (Forgetting)↓ of different components of DoSAPP. All the experiments are averaged over 5 randomised trials with different seeds.

data Krause et al. [2013]. This firstly creates a long sequence of tasks in a class incremental scenario, and secondly causes a domain shift after 10 tasks of aircraft. From Table 4, it can be clearly seen that our proposed method DoSAPP outperforms SPU without ER and Finetune(without any TTL phase). Further it can be inferred that in other baselines, there is a recency bias towards the current task, whereas in DoSAPP, with a marginal decrease of 3.8% on current task accuracy(CTA), there is an overall increase in the average accuracy and the first task accuracy. This shows that our approach retains the knowledge on first task as well adapts well on the current task, with strong generalisation performance.

5 Ablation

In this section we quantitatively analyse the effect of different components of our proposed method DoSAPP. We evaluate the effects of each component in an incremental fashion as seen in Table 3. We further perform ablation studies on each component of DoSAPP as mentioned in A.5

6 Discussion and Conclusion

In this work, we discuss how to leverage test-time data to improve models' representation of previous tasks, mimicking human learning and striving for real intelligent agents. In summary, to the best of our knowledge we are the first to explore test-time learning to control forgetting. We show that test-time data can provide a great source of information when leveraged correctly. Our method, DoSAPP, was able to significantly improve over the zero-shot performance of CLIP when continually learning a dataset without any replay and with no specific CL method applied at the supervised training session. DoSAPP is stable due to sparse parameter updates and the weighted EMA teacher-student framework. Further during TTL, the max-logit in distribution scores makes it more robust to class imbalance than other strategies.

References

R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European Conference on Computer Vision (ECCV)*, 2018.

N. Asadi, M. Davari, S. Mudur, R. Aljundi, and E. Belilovsky. Prototype-sample relation distillation: towards replay-free continual learning. In *International Conference on Machine Learning*, pages 1093–1106. PMLR, 2023.

R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

M. Boschini, L. Bonicelli, A. Porrello, G. Bellitto, M. Pennisi, S. Palazzo, C. Spampinato, and S. Calderara. Transfer without forgetting. In *European Conference on Computer Vision*, pages 692–709. Springer, 2022.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, 2020.

Z. Cai, O. Sener, and V. Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8281–8290, 2021.

J. Chen, D. Zhu, X. Shen, X. Li, Z. Liu, P. Zhang, R. Krishnamoorthi, V. Chandra, Y. Xiong, and M. Elhoseiny. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*, 2023.

R. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135, 1999.

M. Geva, R. Schuster, J. Berant, and O. Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

S. Goyal, A. Kumar, S. Garg, Z. Kolter, and A. Raghunathan. Finetune like you pretrain: Improved finetuning of zero-shot vision models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19338–19347, 2023.

D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

D. Hendrycks, S. Basart, M. Mazeika, A. Zou, J. Kwon, M. Mostajabi, J. Steinhardt, and D. Song. Scaling out-of-distribution detection for real-world settings. *arXiv preprint arXiv:1911.11132*, 2019.

M. Jang, S.-Y. Chung, and H. W. Chung. Test-time adaptation via self-training with nearest neighbor information. *arXiv preprint arXiv:2207.10792*, 2022.

H. Koh, M. Seo, J. Bang, H. Song, D. Hong, S. Park, J.-W. Ha, and J. Choi. Online boundary-free continual learning by scheduled data prior. In *The Eleventh International Conference on Learning Representations*, 2022.

J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

A. Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

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

D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

R. Pei, J. Liu, W. Li, B. Shao, S. Xu, P. Dai, J. Lu, and Y. Yan. Clipping: Distilling clip-based models with a student base for video-language retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18983–18992, 2023.

A. Prabhu, H. A. Al Kader Hammoud, P. K. Dokania, P. H. Torr, S.-N. Lim, B. Ghanem, and A. Bibi. Computationally budgeted continual learning: What does matter? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3698–3707, 2023.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

H. Rasheed, M. U. Khattak, M. Maaz, S. Khan, and F. S. Khan. Fine-tuned clip models are efficient video learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6545–6554, 2023.

J. S. Smith, L. Karlinsky, V. Gutta, P. Cascante-Bonilla, D. Kim, A. Arbelle, R. Panda, R. Feris, and Z. 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*, pages 11909–11919, 2023.

J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020.

A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

L. Wang, X. Zhang, H. Su, and J. Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Z. Wang, Z. Zhan, Y. Gong, G. Yuan, W. Niu, T. Jian, B. Ren, S. Ioannidis, Y. Wang, and J. Dy. Sparcl: Sparse continual learning on the edge. *Advances in Neural Information Processing Systems*, 35:20366–20380, 2022a.

Z. Wang, Z. Zhang, S. Ebrahimi, R. Sun, H. Zhang, C.-Y. Lee, X. Ren, G. Su, V. Perot, J. Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer, 2022b.

Z. Wang, Z. Zhang, C.-Y. Lee, H. Zhang, R. Sun, X. Ren, G. Su, V. Perot, J. Dy, and T. Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022c.

J. Yang, K. Zhou, Y. Li, and Z. Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

G. Zhang, L. Wang, G. Kang, L. Chen, and Y. Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. *arXiv preprint arXiv:2303.05118*, 2023a.

W. Zhang, P. Janson, R. Aljundi, and M. Elhoseiny. Overcoming generic knowledge loss with selective parameter update. *arXiv preprint arXiv:2308.12462*, 2023b.

Z. Zheng, M. Ma, K. Wang, Z. Qin, X. Yue, and Y. You. Preventing zero-shot transfer degradation in continual learning of vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19125–19136, 2023.

D.-W. Zhou, Q.-W. Wang, H.-J. Ye, and D.-C. Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. *arXiv preprint arXiv:2205.13218*, 2022.

D.-W. Zhou, H.-J. Ye, D.-C. Zhan, and Z. Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *arXiv preprint arXiv:2303.07338*, 2023.

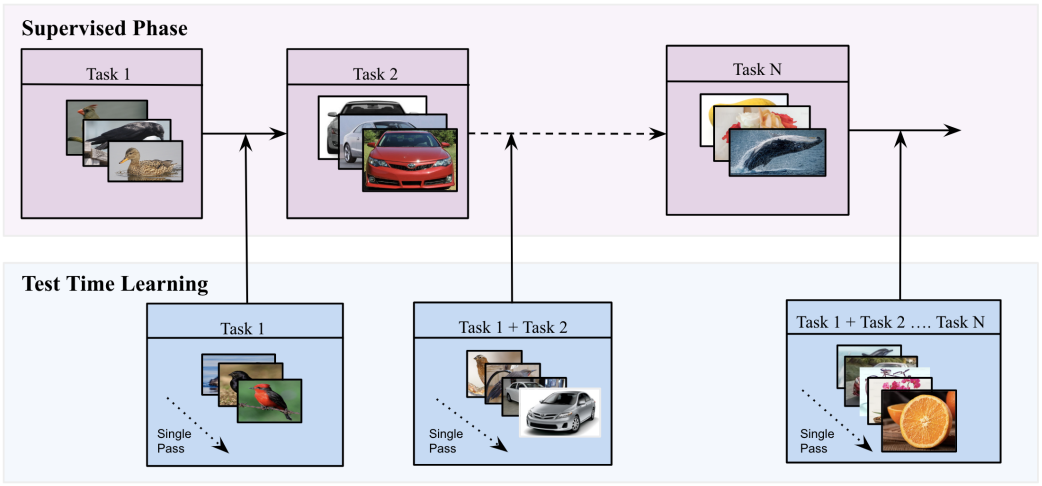

Figure 2: An illustration of our proposed Continual Learning with Interleaved Test Time Learning. Following each session of supervised learning, the model is deployed to adapt in an unsupervised setting. It can encounter data distributions encompassing all previously encountered tasks or sessions. The model adapts on the classes of the current task, while trying to minimize the forgetting on all the classes of previously seen tasks.

A Appendix / supplemental material

A.1 CIL Setting

The CIL setting as described in section 3 is where a sequence of supervised datasets $[\mathcal{D}_1^s, \mathcal{D}_2^s, ...\mathcal{D}_T^s]$ drawn from different distributions are observed at incremental training sessions t ranging from 0 to T, where $\mathcal{D}_t^s = (\mathbf{x}_i^t, y_i^t)_{i=1}^{N_t}$ is the t incremental session with N_t instances. Here the training instance $\mathbf{x}_i^t \in \mathbb{R}^D$ belongs to class $y_i \in Y_t$, where Y_t is the label space of task/dataset at t step. $Y_t \cap Y_{t'} = \phi$ for $t \neq t'$, where $t^{'}$ is any other training session. During a given training session t data samples only from $\mathcal{D}_t^s$ can be accessed. The aim of CIL is to progressively build a unified model encompassing all previously encountered classes. This involves gaining insights from new classes while retaining knowledge from previous ones. Model's performance is evaluated over all the seen classes $\mathcal{Y}_t = Y_1 \cup \cdots Y_t$ after each incremental task/dataset. Formally, the target is to fit a model $\mathcal{M}(\mathbf{x}; \boldsymbol{\theta}) : X \rightarrow \mathcal{Y}_t$ that achieves a minimal loss $\mathcal{L}$ across all testing datasets $\mathcal{D}_t^e$:

$$\sum_{(\mathbf{x}_j, y_j) \in \mathcal{D}_1^e \cup \cdots \mathcal{D}_T^e} \mathcal{L}\left(\mathcal{M}\left(\mathbf{x}_j; \boldsymbol{\theta}\right), y_j\right) \tag{6}$$

where $\mathcal{L}(.,.)$ measures the difference between prediction and groundtruth label. $\mathcal{D}_t^e$ denotes a testing set of task t. Finally θ denotes the model parameters.

After training is complete on each $\mathcal{D}_t^s$ the model is put into production until $\mathcal{D}_{t+1}^s$ becomes available for supervised training. Between supervised phases an unsupervised dataset, $\mathcal{D}_t^u$, is observed corresponding to test-time data encountered in production. Note that this unsupervised data can be drawn from a different distribution than the supervised data, including the distributions of old supervised datasets/tasks. Our goal is to leverage this data to control forgetting of the model by allowing online unsupervised adaptation. Figure 2 depicts our setting. Note that we evaluate our models on test datasets $\{\mathcal{D}^e\}$ that are distinct in terms of instances from those used during the self-supervised online adaptation phase to adequately measure models generalization.

A.2 DoSAPP algorithm

Following Zhang et al. [2023b] we use the gradient magnitude of the loss w.r.t. the incoming data as a score of how relevant a parameter is, the larger the gradient magnitude the larger the expected decrease in loss after small changes to that parameter. We refer to the model being optimized as $\mathcal{M}_S$. Upon recieving supervised data, we first estimate the most relevant parameters, $\boldsymbol{\theta}^{\mathbf{m}}$ such that

Algorithm 1 DoSAPP algorithm for continual and test time learning

Require: $\mathcal{M}_S(\theta^S)$, CLIP loss: $\mathcal{L}(.,.,.)$, sparsity threshold c

$\theta^T = \theta^S$ ▷ Initialize $\mathcal{M}_T(\theta^T)$ with $\mathcal{M}_S(\theta^S)$
for t in tasks **do**
  $\theta^{\mathbf{m}} \leftarrow$ top-K(K=c) params from MLP layers of θ^S based on $\mathcal{F}$ ▷ Sparse Selection, Eq. 7
  for (x_i, y_i) in $\mathcal{D}_t^s$ **do**
    $\theta^{\mathbf{m}} = \theta^{\mathbf{m}} - \eta\nabla\mathcal{L}(\mathcal{M}_S(x_i), y_i)$ ▷ Take one SGD step
    $\theta_{i+1}^T = p\theta_i^T + q\theta_{i+1}^S$ ▷ Dual momentum for teacher EMA update, Eq 2
  end for
  Compute union of masks for all tasks seen so far $\mathbf{m}_u$ ▷ Start of Unsupervised Phase
  Select $\mathbf{m}_u$ params in $\mathcal{M}_S$
  for x_i in $\mathcal{D}_t^u$ **do**
    $l_T = \max(\mathcal{M}_T(x_i), \text{dim} = 1)$
    $l_S = \max(\mathcal{M}_S(x_i), \text{dim} = 1)$
    if $l_T > l_S$ **then**
      $\hat{y} = \arg\max(\mathcal{M}_T(x_i))$
    else
      $\hat{y} = \arg\max(\mathcal{M}_S(x_i))$
    end if
    $\theta^{\mathbf{m}_u} = \theta^{\mathbf{m}_u} - \eta\nabla\mathcal{L}(\mathcal{M}_S(x_i), \hat{y})$ ▷ Take one SGD step
    $\theta_{i+1}^T = p'\theta_i^T + q'\theta_{i+1}^S$ ▷ Dual momentum for teacher EMA update, Eq 5
  end for
end for

($\boldsymbol{\theta}^{\mathbf{m}} \in \boldsymbol{\theta}^S$).

$$\mathcal{F}\left(\boldsymbol{\theta}_{ij}^S, \mathcal{D}_t^s\right) = \left\| \frac{1}{N_t'} \sum_{k=1}^{N_t'} g_{ij}\left(x_k\right) \right\|, \tag{7}$$

where $g_{ij}\left(x_k\right)$ is the gradient of the loss function($\mathcal{L}(\mathcal{M}_S, x_k, y_k)$) regarding the parameter $\boldsymbol{\theta}_{ij}^S$ evaluated at the data point and its label $x_k, y_k \in \mathcal{D}_t^s$. The loss function $\mathcal{L}(\mathcal{M}_S, x_k, y_k)$ is the same CLIP loss, and the entire data is iterated once to compute the gradient score as given in Eq 7. Specifying the sparsity threshold(c), top-K(K=c) most relevant parameters are selected. We set $c = 0.1$ as shown in Zhang et al. [2023b]. This results in a binary mask $\mathbf{m}$ where only selected parameters are updated and others are masked out and kept frozen.

Teacher Student Framework

To insure stability later during online updates and reduce forgetting, we utilise a Student-Teacher framework Tarvainen and Valpola [2017], Koh et al. [2022], Boschini et al. [2022] where the student model is denoted by $\mathcal{M}_S(\boldsymbol{\theta}^S)$ and the teacher model is denoted by $\mathcal{M}_T(\boldsymbol{\theta}^T)$.

During both train and test time, teacher model $\mathcal{M}_T$ parameters $\boldsymbol{\theta}^T$ move with exponentially moving average (EMA) of student model parameters $\boldsymbol{\theta}^S$. Normally in a teacher student framework, all teacher model parameters move similarly towards the student parameters with a single smoothing parameter (momentum). However, in Tables 1 and 3 we show that a single smoothing parameter is insufficient and yields poor performance. Indeed, in our case most of student model parameters remain frozen and only a small portion is updated, we propose that teacher model's parameters corresponding to the student frozen parameters should move at a different pace than those selected for updates. Therefore we use dual smoothing parameters (referred as momentum parameters) based on affine transformation of the binary mask $\mathbf{m}$ to adapt the teacher parameters $\boldsymbol{\theta}^T$.

We take inspiration from Out Of Distribution (ODD) literature Hendrycks and Gimpel [2016], where a sample has to be identified as In Distribution (ID) for a given predictor with a score function predicting high values for ID samples as opposed to OOD samples. Recently it has been shown that using the un-normalized maximum logit output of a given predictor as an ID score is significantly more robust than softmax probability Hendrycks et al. [2019]. Indeed the softmax probability is shown to provide high probability predictions even for unknown samples Yang et al. [2021], which

we want to avoid in our case. Note that for CLIP the logit corresponds to the cosine similarity of image batch with given text features.

A.3 Derivation for dual momentum

In section 3, the teacher model parameters $\boldsymbol{\theta}_i^T$ undergo exponential moving average as

$$\boldsymbol{\theta}_{i+1}^T = \boldsymbol{p}\boldsymbol{\theta}_i^T + \boldsymbol{q}\boldsymbol{\theta}_{i+1}^S \tag{8}$$

where $\boldsymbol{p}$ and $\boldsymbol{q}$ denote the smoothing parameters for the teacher and student model respectively, and can be computed as

$$\begin{aligned} \boldsymbol{p} &= \alpha_1 \mathbf{m} + \beta_1 \\ \boldsymbol{p} &= \alpha_2 \mathbf{m} + \beta_2 \end{aligned} \tag{9}$$

where α_i and β_i for $i \in \{1, 2\}$ are the coefficients for affine transformation of the boolean mask vector $\mathbf{m}$.

To account for masked parameters, two momentum values δ, γ are introduced for teacher and student models respectively, such that for the teacher model, affine coefficients $\alpha_1,\ \beta_1$ are computed by solving the equations:

$$\alpha_1[\mathbf{m}_{ij} = 1] + \beta_1 = \gamma\ , \qquad \alpha_1[\mathbf{m}_{ij} = 0] + \beta_1 = \delta \tag{10}$$

and α_2, β_2 are computed by solving the equations

$$\alpha_2[\mathbf{m}_{ij} = 1] + \beta_2 = 1 - \gamma\ , \qquad \alpha_2[\mathbf{m}_{ij} = 0] + \beta_2 = 1 - \delta \tag{11}$$

This gives

$$\begin{aligned} \alpha_1 = \gamma - \delta, \quad \beta_1 = \delta \\ \alpha_2 = \delta - \gamma, \quad \beta_2 = 1 - \delta \end{aligned} \tag{12}$$

This gives

$$\begin{aligned} \boldsymbol{p} &= (\gamma - \delta)\mathbf{m} + \delta \\ \boldsymbol{q} &= (\delta - \gamma)\mathbf{m} + 1 - \delta \end{aligned} \tag{13}$$

A.4 Datasets and Evaluation Metrics

Datasets: We consider five different vision datasets, three fine-grained(*Aircraft* Maji et al. [2013], *CUB* Wah et al. [2011], *Stanford Cars*Krause et al. [2013], *Oxford Pets* Parkhi et al. [2012], one coarse dataset(*CIFAR100* Krizhevsky [2012]) and one out-of-distribution dataset(*GSTRB* Stallkamp et al. [2012]). These datasets are chosen primarily based on their initially low zero-shot performance with CLIP pre-trained models. To form the continual learning sequences, we split each dataset into 10 subsets with disjoint classes composing 10 tasks. For all the datasets, the training data is used in supervised learning phase. The test data is divided into 2 splits, namely $\mathcal{D}^u, \mathcal{D}^e$ where $\mathcal{D}^u$ is utilised for test-time unsupervised learning and $\mathcal{D}^e$ is used for evaluation.

Evaluation Metrics: After each supervised session t_i and the following test-time adaptation session, we evaluate the model test performance on holdout datasets from all T tasks. In order to do this, we construct the matrix $R \in \mathbb{R}^{TxT}$, where $R_{i,j}$ is the test classification accuracy of the model on task t_j after observing the last sample from task t_i. Thus, we compute **Average Accuracy**(Acc. = $\frac{1}{T}\sum_{i=1}^{T} R_{T,i}.$) and **Average Forgetting**(F. = $= -\frac{1}{T-1}\sum_{i=1}^{T-1} R_{T,i} - R_{i,i}.$) Lopez-Paz and Ranzato [2017]. Taken together, these two metrics allow us to assess how well a continual learner solves a classification problem while overcoming forgetting. All experiments have been done on NVIDIA A100 GPU and each one takes approximately 1 hour for completion.

A.5 More Ablation Study

parameters are updated with a single momentum. Finally we add our dual momentum approach which gives best performance. We also subject our approach to a more challenging scenario where the tasks in TTL phases are class-imbalanced. Here we sample each task from a symmetric Dirichlet distribution whose concentration parameter is the length of each task. This causes a high imbalance of classes within each task, and sometimes, even absence of certain classes. This imbalanced case is of particular importance since in real settings, test suites are often skewed.This is done by randomly sampling classes from a Dirichlet's distribution. Although the performance is inferior to the balanced case, it should not be interpreted as a drawback. This is because, the model should adapt more to the classes that are seen often in TTL phases and loss of performance on rarely seen classes is but natural.

Method(CLIP)	Avg Acc(↑)	FTA(↑)	CTA(↑)	F.(↓)
Finetune(no TTL)	35.24 ±0.87	5.90 ±1.20	75.44 ±0.52	16.87 ±1.04
SPU	39.62 ±1.62	24.31 ±0.30	74.94 ±2.43	7.32 ±0.38
DoSAPP	45.01 ±0.31	30.63 ±0.76	71.13±1.17	2.34 ±0.75

Table 4: Averge Accuracy(Avg Acc.), First Task Accuracy(FTA), Current Task Accuracy(CTA), Average Forgetting(F.) measured for long sequnce of tasks from concatenation of aircraft Maji et al. [2013] and cars Krause et al. [2013] dataset. All experiments are mean of 5 randomised experiments with different seeds.

We highlight the innovative aspect of our approach, which leverages unsupervised test data—readily available in production environments, to enhance continual learning. Unlike our method, existing continual learning (CL) techniques are not inherently designed to incorporate unsupervised test data, making them less adaptable to this scenario. Indeed, naive approaches to using the unsupervised data alongside existing methods proved unfruitful in our preliminary analysis. To illustrate this, we combined the best performing CL method (compared to ours), SPU, with a simple pseudo-labeling baseline, namely SPU + test-time data (D_u). The model is updated with SPU-learned masks using a standard self-labeling approach on test-time data, using the max logit of the model as the label. We can clearly observe that the performance of SPU + test-time data (D_u) has significantly lowered as compared to our method as shown in Table 5. This highlights the importance and effectiveness of our design choices in robustly leveraging test-time data. In Appendx A.7, we further show the superiority of our method in adapting to noise present in the unsupervised test data.

Method	Aircraft		Cars		CIFAR100		CUB		GTSRB	
	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)	Acc. (↑)	F.(↓)
SPU	30.94	28.36	69.41	16.91	58.80	26.37	62.31	7.2	43.06	19.16
SPU+Test Time Data (D_u)	27.72	24.86	68.91	7.34	74.09	10.43	61.21	4.01	60.17	6.94
DoSAPP	39.14	12.55	74.87	-0.74	79.16	7.73	68.17	2.15	72.33	1.02

Table 5: Acc(Average Accuracy) % (↑) and F. (Forgetting)↓ for comparing SPU and DoSAPP when provided with Test time data D_u

A.6 Hyperparameters

tablele 6 shows different hyperparameters that have been used for all the experiments using CLIP backbones. The hyperparameters were selected based on the performance of the first task of Stanford Cars dataset. All the results have been gathered over experiments averaged over 5 random seeds.

Hparams	CLIP model
Batch Size	64
Optimizer	AdamW
Learning Rate	$7.5e-6$
CL Epochs	10
Buffer	0
TTL batch size	64
Momentum-EMA(δ, γ, λ)	0.9999, 0.8, 0.9
sparsity(c)	0.1

Table 6: Hyper Parameters for all the experiments using CLIP ViT-B/16 model.

A.7 Dependence on quality of test data used for unsupervised learning

We want to highlight that the trained model is expected to generalize to the distribution of the test data. We also assume that any quality degradation will be consistent across time steps. For instance, if the data is corrupted with noise, our method would generalize and adapt the model to this corruption as well. To illustrate this, we conducted a small experiment by adding random Gaussian noise (mean = 0, std = 0.1) to different combinations of the test and evaluation suite (referred to as GN in the Table 7). The results are shown below, with average accuracy(Acc) followed by forgetting(F). We observe that when corruption is present in the test-time data, the model is still able to leverage these data and improve on clean evaluation data compared to no test-time baseline by a significant margin of 17% (SPU alone). Interestingly, the model adapted to test-time data with Gausian noise performs better on evaluation data with Gausian noise than the case when the test-time data is clean. This is the evidence on our method's ability to adapt and generalize to the present test-time conditions.

Test Time Data (D_u)	Evaluation Data(D_e)	Acc. ($\uparrow$)	F.($\downarrow$)
Clean	Clean	79.16	7.73
GN	Clean	75.67	9.93
Clean	GN	69.50	12.86
GN	GN	73.42	6.86

Table 7: Performance of DoSAPP with noise added to D_u and D_e for Cifar100 Data