Sequentially Acquiring Concept Knowledge to Guide Continual Learning

Shivanand Kundargi
University of Maryland, Baltimore County
shivank2@umbc.edu

Kowshik Thopalli
Lawrence Livermore National Laboratory
thopalli1@llnl.gov

Tejas Gokhale
University of Maryland, Baltimore County
gokhale@umbc.edu

Abstract

The goal of continual learning (CL) is to adapt to new data (plasticity) while retaining the knowledge acquired from old data (stability). Existing methods focus on balancing stability and plasticity to mitigate the challenge of catastrophic forgetting while promoting learning. However, the impact of order and nature of new samples that the network is trained on remains an underexplored factor. A CL algorithm should ideally also have the ability to rank incoming samples in terms of their relationship with prior data and their effect on the learning process. In this work, we investigate if scoring and prioritizing incoming data based on their semantic relationships with the model's current knowledge can boost CL performance. We propose SACK, short for Sequentially Acquiring Concept Knowledge, a scalable and model-agnostic two-step technique for continual learning. SACK dissects categorical knowledge of the model into fine-grained concepts, computes the relationships between previously learned concepts and new concepts in each experience, and uses this relationship knowledge for prioritizing new samples. Experiments across several types of CL methods (regularization, replay, and prompt-based) in class-incremental and task-incremental settings demonstrate that our approach consistently results in higher accuracy, reduces forgetting, and enhances plasticity. Code: https://github.com/abcxyz709/SACK

1. Introduction

The standard machine learning paradigm can be summarized as a three-step pipeline: `Collect Data` $\longrightarrow$ `Train Model on Data` $\longrightarrow$ `Evaluate Model on Data`. This is rarely the case in real-world deployments, where models encounter continuous streams of data that differ from what they have seen before. The new streams of data are characterized by often having new object categories, style changes, or photometric variations, all of which pose a significant challenge to the models developed under static assumptions. Moreover, deep learning models when trained sequentially on such streams often suffer from catastrophic forgetting [6], where learning new tasks erases knowledge acquired by learning earlier tasks. Consequently, a broad body of work in the CL literature [9, 12, 18, 20, 24] has been in addressing this fundamental trade-off between learning effectively from novel data (plasticity) while mitigating the catastrophic forgetting of already acquired knowledge (stability). While existing approaches attempt to strike this balance through various strategies, they typically treat all samples in the new experience (labeled novel data) with equal importance, without considering which examples are most informative or beneficial for the model to learn from. This contrasts sharply with how humans learn: new information is acquired and understood in the context of prior knowledge. This principle is central to curriculum design, where concepts are introduced in a sequence that builds on what has already been mastered. Most continual learning methods, however, treat all new examples equally, ignoring their relevance to what the model already knows. This can lead to inefficient learning -where the model struggles to integrate unfamiliar concepts or reinforces less relevant patterns at the expense of stability. In this work, we propose an alternative approach: one that enables the learner to prioritize examples based on their conceptual relevance to previously acquired knowledge, thereby supporting more effective and balanced continual learning.

Motivated by this intuition, we propose a concept-based framework, SACK, that allows continual learners to assess the relevance of new data based on what has already been learned. The framework works in two steps: (i) it interprets and represents the model's knowledge from previous experiences using natural language concepts, and (ii) it computes alignment scores between these existing concepts and

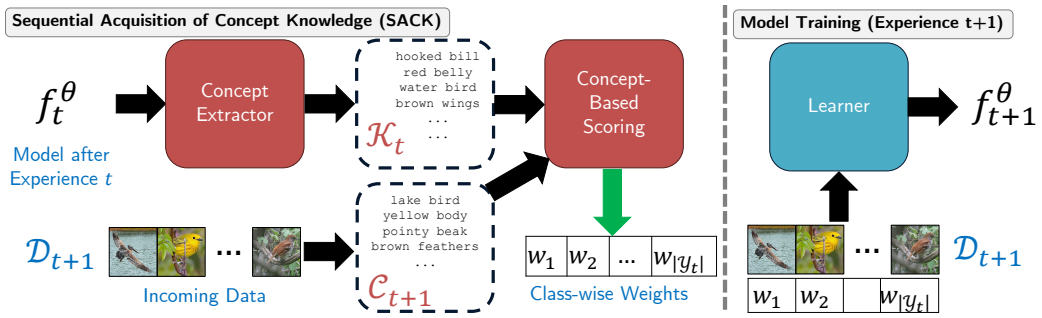

Figure 1. Overview of SACK, where we connect activated concepts obtained after concept extraction with concepts corresponding to classes in the new experience via concept-based scoring. These class-specific connection weights are further used as sample importances while learning new experience.

the concepts required to understand the new data. These scores are then used to guide training: SACK prioritizes classes that are more conceptually similar to past knowledge early in training, and gradually shifts focus toward less similar, harder-to-learn classes as training progresses.

This approach marks a significant departure from standard continual learning practices. To the best of our knowledge, SACK is the first framework that explicitly studies the impact of ordering samples within an experience based on the model's evolving understanding. Moreover, SACK is fully modular and can be seamlessly integrated into any existing CL method without requiring architectural changes. We validate the effectiveness of our approach through extensive experiments on three datasets—CIFAR-100 [11], CUB-200 [27], and ImageNet-R [8]. We incorporate SACK into five widely used CL baselines: replay-based methods (iCaRL [18], DER, and DER++[4]), the regularization-based method LwF[12], and the recent prompt leanring-based method CoDA-Prompt [24]. Across all combinations and evaluation metrics, we observe that SACK consistently enhances performance, demonstrating its broad utility and effectiveness.

2. Related Work

Continual and Curriculum Learning The challenge of enabling models to accumulate knowledge over time without overwriting prior information [6] has led to extensive research in continual learning (CL), also known as lifelong or incremental learning. To address the stability-plasticity dilemma, three major CL strategies have emerged: regularization-based methods [9, 12], replay-based techniques [20, 23], and architectural modifications [21]. Recently, prompt-based methods [24] have gained attention, where task-specific prompts are learned while keeping the model backbone fixed. For further comprehensive information and detailed overview about more continual methods, we refer readers to recent literature survey studies [16, 26, 28]. Since CL deals with stream of data, certain aspects of curriculum learning[2, 5] in context of CL has also been explored by works like [1] where the effect of ordering the tasks in CL has been investigated. Unlike these approaches, our framework SACK dynamically assesses the relative importance of each training class by probing the model's prior semantic knowledge and uses that information to construct a suitable curriculum in a given experience.

Interpretability Tools for Knowledge Discovery. Since our approach relies on effectively capturing the concepts learned by the model in previous experiences, the research on interpretability methods is highly relevant to our effort. Classical methods like SHAP [13], LIME [19], and Grad-CAM [22] offer local insights but fall short in capturing higher-level semantic structure. Recent methods like CLIP-Dissect [14] address this by leveraging CLIP's [17] vision-language space to semantically interpret internal representations of a given deep model. While other intepretability tools such as concept bottleneck models (CBMs) [10, 15] have been explored in CL [30], they require complex architectural changes. Given the demonstrated effectiveness of CLIP-Dissect [14] in vision models, we adopt it as the interpretability tool of choice in this work and note that SACK can be readily used with any tool, including CBMs.

3. SACK

The core idea of SACK is to guide learning at each experience by leveraging the model's evolving understanding. Specifically, it involves two steps: first, extracting the concepts acquired from previous experiences and those required to learn the new classes—*concept extraction*; and second, computing relative scores between them—*concept-based scoring*—to design a targeted curriculum that prioritizes learning based on conceptual relevance. An overview of the approach is shown in Figure 1.

Preliminaries Consider a classifier f parameterized by weights θ that maps inputs $x \in \mathcal{X}$ to categorical outputs $y \in \mathcal{Y}$. In a CL setting, the classifier is exposed to a sequence of experiences E_t at discrete time steps $t \in \{1, \ldots, T\}$.

Model parameters θ are updated sequentially using one experience at a time.

3.1. Concept Extraction

Extracting Concepts from Incoming Data At each experience, following standard CL practice, the learner has access to the names of classes $\mathcal{Y}_t$ from that experience. We use a large language model (e.g., GPT-4) to expand the names of classes to a set of concepts that describe each class by appropriately prompting the model as described in [15, 25]. For instance, the attributes/concepts for the class ``bear'' are defined by ``thick coat, mighty roar, large paws, round ears, dark eyes'', etc. Let $\mathcal{C}_t$ be the set of the concepts for all classes $\mathcal{Y}_t$ in the experience E_t. Further, let $\mathcal{C}_{1:t} = \{\mathcal{C}_i\}_{i=1}^{t}$ be the combined set of concepts obtained for all experiences until E_t.

Extracting Concepts Learned by the Model To extract the concepts learned by the model up to the current experience t, we use CLIP-Dissect [14], which associates natural language concepts and confidence scores with individual neurons at a given layer. We retain only the top 25% of concepts based on their confidence scores to filter out noisy or weakly aligned concepts. The resulting set of high-confidence concepts is denoted by $\mathcal{K}_t$.

3.2. Concept-Based Scoring

Now that we have computed concepts corresponding to the incoming experience $\mathcal{C}_{t+1} = \{\mathcal{C}_i\}_{i=1}^{\mathcal{Y}}$ and activated concepts acquired by the model $\mathcal{K}_t$, we will use them to assign weights to each class $y \in \mathcal{Y}_{t+1}$. Using the CLIP text encoder, we obtain embeddings $u \in \mathbb{R}^{768\times L}$ corresponding to L activated concepts. Similarly, we obtain embeddings $v^y \in \mathbb{R}^{768\times R}$ corresponding to R concepts for each class $y \in \{1, \dots |\mathcal{Y}|\}$. We then compute the cosine similarity between u and each v^y to obtain $H^y \in \mathbb{R}^{L\times R}$. Each row H^y_ℓ of this similarity matrix represents the similarity of a particular neuron activated concept with every concept extracted from class y. To obtain the class-specific weights for a class $y \in \mathcal{Y}$, we average the max value from each row l of H^y.

3.3. Designing a Concept-Based Sampler

As described in Algorithm 1, to train on incoming experience E_{t+1}, we use the class-specific weights from the previous section to guide the sampling of training data for each epoch within that experience. Let $\mathbf{p}_g$ represent the class-wise sampling probabilities used by the sampler at epoch g, and let $\mathcal{W}$ represent the connection weights obtained for each class in the incoming new experience E_{t+1}. Inspired from the recent results in anti-curriculum learning [29], initially, we set the sampling to uniform, and as the training progresses, the sampling probabilities are linearly interpolated towards the class-specific importance weights:

$$\mathbf{p}_g = (1-\lambda_g)\cdot\frac{1}{|\mathcal{Y}_{t+1}|}\cdot\mathbf{1} + \lambda_g\cdot\frac{\mathbf{w}_{t+1}}{\sum_{y=1}^{|\mathcal{Y}_{t+1}|} w^y_{t+1}},$$

$$\text{where } \lambda_g = \frac{g-1}{G-1}; \quad \forall g \in \{1, G\}.$$

where λ_g is the interpolation coefficient that increases linearly over time, $|\mathcal{Y}_{t+1}|$ is the number of classes, and G is the number of epochs. At the beginning of each experience (epoch $g = 1$), we have $\lambda_g = 0$ and therefore the sampler starts with a uniform distribution. At the end of each experience (epoch $g = G$), we have $\lambda_g = 1$, i.e. the sampler ends with a distribution based on the class-based scores.

Algorithm 1 Sequentially Acquiring Concept Knowledge

Input: Labeled data $D_t(\mathcal{X}_t, \mathcal{Y}_t)$ for each experience E_t. Model f with initial parameters θ
Initialize: Experience index $t \leftarrow 1$; parameters θ_1
$\theta_1 \leftarrow \textsc{Train}(D_1, \theta_1)$ ▷ train model f on first experience
for each experience E_t, $t \in \{2, \dots T\}$ **do**
  Initialize: $\theta_t \leftarrow \theta_{t-1}$ ▷ continual parameter update
  $\mathcal{K}_t \leftarrow \textsc{Dissect}(f_\theta)$ ▷ obtain concepts for model's current knowledge
  $\mathcal{C}_t \leftarrow \text{LLM}(\mathcal{Y}_t)$ ▷ obtain concepts for new experience classes
  $w \leftarrow \textsc{Concept\_Scoring}(\mathcal{K}_t, \mathcal{C}_t)$ ▷ obtain class-wise scores
  for each epoch $g = 0$ to G **do**
    $p_g = (1-\lambda_g)\cdot\frac{1}{|\mathcal{Y}_t|}\mathbf{1} + \lambda_g\frac{\mathcal{W}}{\sum_{j=1}^{|\mathcal{Y}_t|}\mathcal{W}^j}$, where $\lambda_g = \frac{g}{G}$
    $\hat{D}^g_t \leftarrow \textsc{Weighted\_Sampling}(D_t, p_g)$ ▷ sample data
    $\theta_t \leftarrow \textsc{Train}(\hat{D}^g_t, \theta_t)$ ▷ train model f
  end for
end for
return θ^* ▷ final updated model parameters

4. Experiments

We perform an extensive evaluation of SACK under different continual learning settings (CIL, TIL) with models of varying scales (ResNets, CLIP, Vision Transformers) on multiple benchmark datasets. We describe our experimental setup and results in this section.

Experimental Setup In all of our experiments, we utilize CLIP-dissect [14] to extract the knowledge/concepts learnt by the model through data from previous experiences. Since SACK is not specific to any particular choice of continual learning method, we incorporate SACK in a variety of CL methods. To integrate SACK across different CL methods, we utilize the `mammoth` [3] library. **Datasets.** We use CIFAR-100 [11], CUB-200 [27], and ImageNet-R [8] for our experiments. To evaluate SACK under fine-grained classification and under distribution shifts we use CUB-200 [27] and ImageNet-R [8] respectively.

Implementation Details. Concept Extraction. For iCaRL, DER, DER++, and LwF using ResNets [7], we extract concepts from the last fully connected layer and for CODA-Prompt [24], which uses CLIP, we dissect the learn-

Method	CIFAR-100			CUB-200			IMAGENET-R		
	Avg.ACC	Last.ACC	BWT	Avg.ACC	Last.ACC	BWT	Avg.ACC	Last.ACC	BWT
LWF[12]	28.80	92.60	-89.50	10.78	37.75	-20.41	8.40	8.49	-14.03
LWF + SACK	**29.33**	**92.70**	**-88.70**	**10.87**	**42.34**	**-20.81**	**12.83**	**17.62**	**-12.36**
iCaRL[18]	82.81	91.4	-4.88	17.29	14.45	-7.37	5.47	**9.27**	**-23.66**
iCaRL + SACK	**83.30**	**91.90**	**-4.38**	**24.71**	**20.91**	**1.69**	**6.35**	8.54	-24.32
DER[4]	80.33	91.10	-10.50	26.30	56.29	-21.89	24.18	58.17	-45.62
DER + SACK	**81.36**	**92.50**	**-9.70**	**26.60**	**57.82**	**-22.50**	**25.30**	**58.59**	**-45.27**
DER++[4]	80.47	93.20	-10.97	51.81	60.20	0.655	35.74	48.61	**-9.07**
DER++ + SACK	**82.82**	**93.50**	**-8.27**	**52.25**	**61.05**	**0.66**	**36.85**	**54.77**	**-9.07**
CODA-Prompt[24]	**98.56**	**99.50**	**0.17**	89.23	**93.02**	-0.02	91.24	**90.44**	0.50
CODA-Prompt + SACK	**98.56**	**99.50**	0.08	**89.34**	**93.02**	**0.28**	**91.35**	89.38	**0.80**

Table 1. Performance comparison of various continual learning (CL) methods with and without integration of SACK in the Task-Incremental Learning (TIL) setting. Each dataset is split into 10 sequential experiences with an equal number of classes per experience. Results are reported using Average Accuracy, Last Accuracy, and Backward Transfer. Bold represents best values.

Method	CIFAR-100			CUB-200			IMAGENET-R		
	Avg.ACC	Last.ACC	BWT	Avg.ACC	Last.ACC	BWT	Avg.ACC	Last.ACC	BWT
LWF[12]	9.34	92.60	-68.51	3.77	37.75	-28.77	0.84	8.49	-24.66
LWF + SACK	**9.43**	**92.70**	**-66.58**	**4.23**	**42.34**	**-28.19**	**1.76**	**17.62**	**-22.42**
iCaRL[18]	48.06	53.60	**-18.40**	3.89	2.38	-7.84	1.81	**4.73**	-49.63
iCaRL + SACK	**49.83**	**55.80**	-19.22	**7.06**	**3.40**	**-2.52**	**1.89**	4.03	**-44.22**
DER[4]	**49.22**	75.90	-38.70	5.68	56.12	-45.20	5.81	58.17	-25.95
DER + SACK	**49.24**	**80.80**	**-38.60**	**5.82**	**57.65**	**-45.30**	**5.83**	**58.38**	**-24.18**
DER++[4]	45.93	**85.70**	-44.86	28.11	49.82	-17.26	12.34	40.12	**-26.92**
DER++ + SACK	**49.04**	83.00	**-40.62**	**28.26**	**52.04**	**-17.14**	**11.28**	**52.01**	-33.45
CODA-Prompt[24]	86.79	90.20	**-5.40**	71.31	85.71	-8.48	75.24	74.94	-5.75
CODA-Prompt + SACK	**87.26**	**90.70**	-5.50	**71.78**	**85.20**	**-8.05**	**75.65**	**75.37**	**-5.49**

Table 2. Evaluating the improvements after integration of SACK in the Class-Incremental Learning (CIL) setting.

able head of CLIP's vision encoder. Hyperparameters. All hyperparameters follow the standard settings from the `mammoth` library [3], which we also use to train baselines for fair comparison.

Evaluation Metrics. To evaluate the efficacy of the proposed approach, we utilize three standard CL evaluation metrics- underline(i) Average Accuracy, (ii) Last Accuracy and (iii) Backward Transfer (BWT). In all cases, higher values indicate better performance.

4.1. Results

SACK Improves All CL methods in Task-Incremental Learning From table 1 we can observe that, on CIFAR-100, all baseline methods benefit from the integration of SACK. For instance, iCaRL+SACK improves Avg.ACC by 0.59% With SACK, DER++ achieves a 2.92% gain in Avg.ACC and a 24.6% reduction in forgetting. In contrast, CODA-Prompt's performance remains unchanged with SACK —likely due to its already saturated performance, achieving 99.50% on Last.ACC, leaving little room for improvement. On the more challenging and fine-grained CUB-200 dataset, SACK yields striking improvements, clearly demonstrating the benefits of our principled approach. When combined with iCaRL, Avg.ACC improves by 42.9%, and BWT turns positive, indicating strong knowledge retention. Similar improvements are observed for other methods as well. Notably, even the stronger CODA-Prompt benefits from SACK, with BWT increasing from -0.02 to 0.28. Furthermore, on the ImageNet-R benchmark—which introduces significant real-world distribution shifts—SACK enhances the performance of all methods. For example, when integrated with LWF, Avg.ACC improves by 52.7% and Last.ACC more than doubles. Even with other methods, SACK shows consistent gains across all metrics thereby demonstrating its generality and effectiveness in TIL setting.

Benefits Continue to Persist even In The Challenging Class Incremental Learning Setup Beyond the TIL setting, we also evaluate the performance of SACK in the more challenging class-incremental learning (CIL) setup, and results are summarized in Table 2. In CIL, task identities are not available at inference time, making CL substantially harder. Despite this increased difficulty, SACK continues to deliver consistent improvements across multiple baselines and datasets. On CIFAR-100, SACK offers modest but meaningful gains across most methods. For example, integrating SACK with iCaRL improves Avg.ACC from 48.06 to 49.83 (a 3.69% increase). DER++ also sees noticeable improvements, with Avg.ACC increasing from 45.93 to 49.04 While gains for CODA-Prompt are smaller, SACK still improves Avg.ACC without compromising on the BWT. Similar to results obtained in TIL, on CUB-200, the benefits of SACK are especially apparent. When combined with iCaRL, Avg.ACC increases by over 81% (from 3.89 to 7.06). Interestingly, even with strong baselines such as CODA-Prompt, SACK brings modest improvements to both Avg.ACC and BWT. Finally, results on the ImageNet-R benchmark highlight SACK 's robustness. With LWF, Avg.ACC more than doubles from 0.84 to 1.76, and Last.ACC jumps from 8.49 to 17.62. DER++ benefits significantly as well, with Last.ACC improving by nearly 12 points however this increases comes at the cost of BWT. These consistent improvements across diverse methods and benchmarks validate the effectiveness of SACK in the more

realistic and demanding CIL setting.

5. Conclusion

In this work, we have developed SACK (Sequentially Acquiring Concepts), a method-, model-, and setting-agnostic two-step continual learning framework that helps CL methods prioritize samples in the new experience based on the prior concept knowledge of the learner.

Acknowledgements

SK was supported by the UMBC Cybersecurity Graduate Fellows program. TG was supported by the SURFF award from UMBC ORCA. Computing support was provided by UMBC HPCF. KT's work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract No. DE-AC52-07NA27344, Lawrence Livermore National Security, LLC and was supported by the LLNL-LDRD Program under Project No. 25-SI-001. The views and opinions of the authors expressed herein do not necessarily state or reflect those of the funding agencies and employers.

References

[1] Samuel J Bell and Neil D Lawrence. The effect of task ordering in continual learning. *arXiv preprint arXiv:2205.13323*, 2022. 2

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009. 2

[3] Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended der-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 3, 4

[4] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020. 2, 4

[5] Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1): 71–99, 1993. 2

[6] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. 1, 2

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[8] 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. 2, 3

[9] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 1, 2

[10] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pages 5338–5348. PMLR, 2020. 2

[11] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 3

[12] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 1, 2, 4

[13] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017. 2

[14] Tuomas Oikarinen and Tsui-Wei Weng. Clip-dissect: Automatic description of neuron representations in deep vision networks. *arXiv preprint arXiv:2204.10965*, 2022. 2, 3

[15] Tuomas Oikarinen, Subhro Das, Lam M Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. *arXiv preprint arXiv:2304.06129*, 2023. 2, 3

[16] Haoxuan Qu, Hossein Rahmani, Li Xu, Bryan Williams, and Jun Liu. Recent advances of continual learning in computer vision: An overview. *IET Computer Vision*, 19(1):e70013, 2025. 2

[17] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 2

[18] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 1, 2, 4

[19] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016. 2

[20] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019. 1, 2

[21] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 2

[22] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 2

[23] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 2

[24] 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*, pages 11909–11919, 2023. 1, 2, 3, 4

[25] Rakshith Subramanyam, Kowshik Thopalli, Vivek Narayanaswamy, and Jayaraman J Thiagarajan. Decider: Leveraging foundation model priors for improved model failure detection and explanation. In *European Conference on Computer Vision*, pages 465–482. Springer, 2024. 3

[26] Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022. 2

[27] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2, 3

[28] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory. *Method and Application*, 2302:00487, 2023. 2

[29] Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? *arXiv preprint arXiv:2012.03107*, 2020. 3

[30] Sin-Han Yang, Tuomas Oikarinen, and Tsui-Wei Weng. Concept-driven continual learning. *Transactions on Machine Learning Research*, 2024. 2