# Caption, Create, Continue: Continual Learning with Pre-trained Generative Vision-Language Models

Indu Solomon
International Institute of Information Technology Bangalore (IIITB), India
indu.solomon@iiitb.ac.in

Aye Phyu Phyu Aung
Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore
aye_phyu_phyu_aung@i2r.a-star.edu.sg

Uttam Kumar
International Institute of Information Technology Bangalore (IIITB), India
uttam@iiitb.ac.in

Senthilnath Jayavelu
Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore
J_Senthilnath@i2r.a-star.edu.sg

## Abstract

Continual learning (CL) enables models to adapt to evolving data streams without catastrophic forgetting, a fundamental requirement for real-world AI systems. However, the current methods often depend on large replay buffers or heavily annotated datasets which are impractical due to storage, privacy, and cost constraints. We propose CLTS (**C**ontinual **L**earning via **T**ext-Image **S**ynergy), a novel class-incremental framework that mitigates forgetting without storing real task data. CLTS leverages pre-trained vision-language models, BLIP (Bootstrapping Language-Image Pre-training) for caption generation and stable diffusion for sample generation. Each task is handled by a dedicated Task Head, while a Task Router learns to assign inputs to the correct Task Head using the generated data. On three benchmark datasets, CLTS improves average task accuracy by up to 54% and achieves 63 times better memory efficiency compared to four recent continual learning baselines, demonstrating improved retention and adaptability. CLTS introduces a novel perspective by integrating generative text-image augmentation for scalable continual learning.

## CCS Concepts

• **Computing methodologies** → **Lifelong machine learning**.

## Keywords

Pre-trained, task, Catastrophic forgetting, Continual learning.

**ACM Reference Format:**
Indu Solomon, Aye Phyu Phyu Aung, Uttam Kumar, and Senthilnath Jayavelu. 2025. Caption, Create, Continue: Continual Learning with Pre-trained Generative Vision-Language Models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3746252.3760926

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*CIKM '25, Seoul, Republic of Korea*
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3760926

## 1 Introduction

Conventional machine learning (ML) models are typically trained on curated datasets and often fail when exposed to evolving data distributions. This leads to mis-classification, as the model forgets previously learnt knowledge when adapting to new data, a phenomenon known as catastrophic forgetting [13, 14]. While re-training on the entire dataset can mitigate this, but it is often impractical due to computational, storage and privacy constraints. Continual learning (CL) addresses this issue by enabling models to incrementally learn from data streams without forgetting prior knowledge [4]. A stream of data arriving at a specific time is referred to as a task and CL settings are broadly classified into Task Incremental Learning (Task-IL), Domain Incremental Learning (Domain-IL), and Class Incremental Learning (Class-IL) [23, 25]. Task-IL assumes the task identity is known during inference, while Domain-IL introduces distribution shifts with shared label space. Class-IL, the most challenging setting, requires the model to classify among all the learned classes without task identity. In this work, we focus on Class-IL setting.

Catastrophic forgetting in CL is typically addressed using memory replay, regularisation based constraints, or dynamic architectural expansion [1, 5, 11]. Earlier CL models were trained from scratch, demanding substantial time and computational resources. In contrast, the use of pre-trained models, trained on large scale datasets like ImageNet has emerged as a practical alternative, enabling efficient task adaptation while minimising resource consumption and training effort. These models offer strong general-purpose representations that can be fine-tuned with domain knowledge, enabling scalable and efficient continual learning for rapid deployment. In this paper, we propose Continual Learning via Text-Image Synergy (CLTS), a novel approach overcoming major limitations of existing CL methods–catastrophic forgetting, long training, and high resource costs. CLTS expands architecture to mitigate forgetting and leverages pre-trained models to cut data needs and computation. The key contributions are:

1. A modular architecture that supports dynamic architectural expansion, integrating Task Heads (TH), a Task Router (TR), and pre-trained models to enable continual adaptation in the Class-IL setting.
2. A lightweight memory replay strategy that retains textual captions of prior tasks instead of raw images, substantially reducing the memory requirements without sacrificing the performance.

3. A novel use of text-to-image generation: training the Task Router (TR) using synthetic images generated by the Stable Diffusion (*SD*) model, conditioned on stored captions.

We benchmark CLTS on CIFAR-10, CIFAR-100, and TinyImageNet, showing consistent gains over four state-of-the-art baselines and effective mitigation of forgetting.

## 2 Related Works

Continual learning approaches are broadly classified into replay-based, regularization-based, and architectural expansion methods. Replay techniques mitigate forgetting by mixing stored or generated exemplars from previous tasks with new data [5, 15, 16, 18]. Regularization imposes constraints to preserve prior knowledge [9, 11]. Architectural methods grow the network to accommodate new tasks [1, 12, 19]. Diffusion-based CL methods like SDDR [8] and DDGR [7] still depend on memory replay and require task labels at inference—limiting scalability and autonomy.

CaSSLe [6] uses an auxiliary predictor for distillation to align current with prior representations, but its reliance on task boundaries at inference makes it unsuitable for Class-IL. SCALE [24] integrates contrastive learning, pseudo-supervision, a forgetting loss, and uniform replay, but remains sensitive to replay data quality and the SSL backbone. U-TELL [22] leverages task experts, a task assigner, and structured generators to approximate task distributions without storing samples; however, low-quality image generation restricts its effectiveness to simple digit datasets, failing on real-world data. Dynamic self-evolving clustering networks [14, 20] support online learning through on-the-fly neuron and cluster creation, but centroid-based replay offers limited forgetting mitigation and weak real-world performance. UPL-STAM [21] replaces exemplars with stored centroids via clustering, novelty detection, and selective forgetting, but its reliance on preprocessing and memory management limits robustness and scalability.

Recognising the constraints of earlier modular approaches in CL [1, 14, 20, 22], we present CLTS, a scalable framework that integrates pre-trained modules and avoids the limitations of task-specific replay or data generation. In contrast, our method discards both exemplar storage and test-time labels, using caption conditioned diffusion to train a Task Router, which enables task inference by dynamically routing the inputs to appropriate Task Heads.

## 3 Proposed Architecture

**Problem setting.** We consider a class-incremental learning (Class-IL) setup, where the task sequence is defined as $\tau = \{\tau_t\}_{t=1}^{k}$, with each task $\tau_t$ comprising $n_t$ samples and a disjoint label space. The input data for task $\tau_t$ is denoted as $X_t = x_j \in \mathbb{R}^{n\times m\times p}$. A low-dimensional latent representation of task data is denoted as $Z_t \in \mathbb{R}^d$, with $d \ll n \times m \times p$. Additionally, $\mathcal{X}_t$, $t$ refer to synthetic samples generated by Stable Diffusion (*SD*), and their associated taskIDs respectively. The generated captions are denoted by $C_t$, while TH and TR represent the Task Head and Task Router, respectively.

**CLTS Architecture** The proposed CLTS framework (Fig. 1a) comprises two core components: the Task Head (TH) and the Task Router (TR), alongside a pre-trained Bootstrapping Language-Image Pre-training (BLIP) [10] and Stable Diffusion (*SD*) [17] module. The TH captures task-specific distributions in a low-dimensional latent space and uncovers hidden patterns in the feature embeddings via $K$-Means clustering. Under the Class-IL setting, where task identity is unavailable at test time, the TR predicts the task distribution of incoming samples and routes them to the appropriate TH. An initial mini-batch of labelled data seeds the clustering process, allowing the construction of a lookup table that maps clusters to class labels. This mapping is used during inference for label recovery.

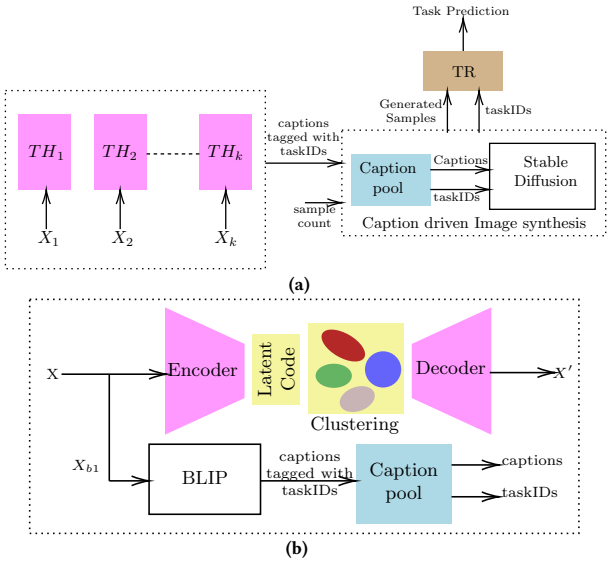


**Figure 1: CLTS architecture: (a) Block diagram; (b) Architectural details of Task Head network**

The CLTS training proceeds in two stages. First, the TH networks are incrementally trained as task data arrives, encoding inputs into a low-dimensional latent space where clustering is performed. In the second stage, the TR is trained using synthetic samples generated by prompting a pre-trained SD model with task-specific captions. These captions are generated using BLIP and each synthetic sample is labelled with its original taskID. Both SD and BLIP are kept frozen, having been pre-trained on large-scale, domain-relevant datasets. The overall training workflow is outlined in Algorithm 1.

**Task Head (TH).** The Task Head (TH) processes a stream of sequential tasks by instantiating a new module upon the arrival of a new task. TH comprises of three key components: i) variational autoencoder (VAE) that learns a low-dimensional latent representation of the task data distribution, enabling efficient encoding; ii) a $K$-Means clustering block that organizes these latent task data into semantically meaningful clusters; and iii) a BLIP module, pre-trained on large-scale datasets, which generates descriptive captions for training samples. Unlike conventional CL models that store task samples for replay, CLTS replaces data storage with caption storage, using these captions to prompt image generation. This significantly reduces memory overload. The TH is trained on each incoming task and generates between 60—100 descriptive captions, depending on the training batch size. Each caption is tagged with a corresponding task identifier (taskID) and stored in a dynamic caption pool. This pool is incrementally updated as new tasks arrive. The stored taskIDs are subsequently used to train the Task Router (TR) in a supervised manner. Fig. 1b illustrates the TH architecture, highlighting its components.

**Algorithm 1** Continual Learning via Text-Image Synergy (CLTS)

**Require:** Task data $\{X_t\}$, $t = 1, \ldots, k$
**Ensure:** Task Heads $\{TH_t\}$, $t = 1, \ldots, k$ and Task Router $TR$
1: Let $X_t$, $t = 1, \ldots, k$ is the sequential task data
2: **while** $t$ **do**
3: Train $TH_t$ using $X_t$ and obtain latent representations $Z_t$
4: Generate captions for the initial mini-batch of task training data with pre-trained BLIP captioning model $C_{t_{b_1}} = BLIP(X_{t_{b_1}})$
5: Incrementally update the caption pool by adding captions tagged with taskIDs $t$ as $\{C_{t_{b_1},t}\}$.
6: Perform clustering on latent data $Z_t$
7: **end while**
8: **for** $t \leq k$ **do**
9: Generate task samples using SD model, $\mathcal{X}_t = SD(C_t)$ and assign taskID $t$, $t = 1, \ldots, k$
10: Resize $\mathcal{X}_t$ to the size of $X_t$
11: **end for**
12: Train Task Router $TR$ with $\{(\mathcal{X}_t, t)\}$, $t = 1, \ldots, k$

**Task Router (TR).** The Task Router (TR) module is responsible for selecting the most suitable TH to process each incoming test sample. TR is trained in a supervised manner using synthetic samples generated by the Stable Diffusion (SD) along with their associated taskIDs. These taskIDs are distinct from the class labels and are algorithmically assigned based on the task information inferred from the text captions. For the TR, we employ a VGG19 model pre-trained on ImageNet, which is then fine-tuned by replacing its final two layers. The output layer is configured to produce logits corresponding to the total number of tasks.

**Training Objective** The training objective of CLTS involves minimization of two key loss functions. The total loss is defined as:

$$\mathcal{L} = \lambda_1 \sum_t \mathcal{L}_{TH}^{(t)} + \lambda_2 \mathcal{L}_{TR}, \tag{1}$$

where $\lambda_1$ and $\lambda_2$ are tunable hyperparameters. For each task $t$, the loss $\mathcal{L}_{TH}^{(t)}$ comprises of two components:

$$\mathcal{L}_{TH}^{(t)} = \mathcal{L}_{vae} + \mathcal{L}_{clust}, \tag{2}$$

where $\mathcal{L}_{vae}$ is the encoder-decoder loss of $VAE$ and the second term $\mathcal{L}_{clust}$ is a clustering loss given by:

$$\mathcal{L}_{clust} = \sum_{j=1}^{J} \sum_{z_i \in M_j} \|\mathbf{z_i} - \mu_\mathbf{j}\|^2 \tag{3}$$

where $z_i \in M_j$ denotes the latent vector assigned to the $j^{th}$ cluster, and $\mu_j$ is its centroid. The loss $\mathcal{L}_{TR}$ trains the TR module to predict taskIDs based on the synthetic inputs. The loss $\mathcal{L}_{TR}$ is defined as a cross-entropy loss between reference taskIDs $t_i$ and the predicted taskIDs $p_i$.

$$\mathcal{L}_{TR} = -\sum_i t_i \log(p_i), \tag{4}$$

**Testing Procedure.** In the testing phase of the CLTS model, TR, Encoder block of VAE and the K-Means clustering module work in unison to process the incoming test samples. The TR first predicts the task, $t$ for each incoming test sample and the corresponding $TH_t$ is selected for processing. The VAE encoder then transforms these test samples into low-dimensional latent representations, which are subsequently grouped into clusters by the K-Means algorithm. Finally, these clusters are cross-verified with the pre-constructed lookup table for validation.

## 4 Experimental Results

We evaluate CLTS model under Class-IL setting using average task accuracy (*ACC*) as primary metric, $ACC = \frac{1}{k}\sum_{t=1}^{k} A_t$, consistent with prior works [5, 25]. *ACC* captures the model's performance across all incremental stages, with higher values indicating stronger performance across multiple tasks throughout these stages [25].

**Datasets.** We evaluate the performance of the CLTS model across three real-world Class-IL benchmarks: Split CIFAR10 (SCIFAR10) [14], Split CIFAR100 (SCIFAR100) and Split TinyImageNet (STinyImageNet) [24]. Among the chosen benchmarks, the SCIFAR100 datastream provides a larger number of tasks, enabling a detailed analysis of our model's performance. The SCIFAR100 datastream is structured as a sequence of 10 tasks $(t_1, \ldots, t_{10})$, with each task introducing two new classes out of a total of 20 coarse classes [24]. Within Class-IL setting, we organize the SCIFAR10 [14] and STinyImageNet datasets into a sequence of five tasks $(t_1, t_2, t_3, t_4, t_5)$. Each task is defined by a unique pair of mutually exclusive classes $(\{\{0, 1\}, \{2, 3\}, \{4, 5\}, \{6, 7\}, \{8, 9\}\})$.

**Table 1: Performance comparison of CLTS with selected baselines on CIFAR10, CIFAR100 and TinyImagenet datasets**

| Technique | Dataset | | |
|---|---|---|---|
| | SCIFAR10 | SCIFAR100 | STinyImageNet |
| SCALE | 21.72±1.01 | 13.68±0.78 | 21.66±0.68 |
| CaSSLe | 16.91±1.00 | 10.97±0.84 | 20.66±1.35 |
| UPL-STAM | 28.02±1.99 | 13.22±0.32 | 21.68±1.30 |
| U-TELL | 29.65±0.67 | 16.79±0.38 | 27.78±1.13 |
| **CLTS(ours)** | **32.27±1.18** | **25.86±0.80** | **35.6±1.22** |

**Baselines.** We chose U-TELL [22], SCALE [24], CaSSLe [6] and UPL-STAM [21] as baselines for our experimental study. The CaSSLe model originally designed under the assumption of known task boundaries, was modified to function within unknown task boundaries to ensure compatibility with the Class-IL framework.

**Performance Comparison.** Table 1 compares the CLTS model against selected baselines. The reported values are average task accuracy computed from the mean and standard deviation over ten runs. The CLTS model outperforms the best performing baseline by a considerable margin. The Class-IL experiments on SCIFAR10 show an 8.8% improvement in average task accuracy. The SCIFAR100 dataset (10 tasks) demonstrates an improvement of 54.02%, while STinyImageNet dataset shows a 28.15% improvement. These results are driven by integrating the pre-trained models into the architecture. The SD module generates a diverse, high quality image dataset that trains the TR module, thereby improving the task prediction and TH selection. Fig. 2a and 2b show a few examples of BLIP-generated captions, along with corresponding diverse images generated by SD for the SCIFAR10 and STinyImageNet datasets, respectively.

Fig. 3 compares the task-wise performance of the CLTS model and baselines on SCIFAR100 dataset under expanding clustering tasks. Despite SCIFAR100's increased task count, the CLTS model

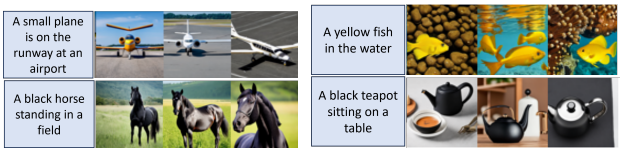


**Figure 2: Examples of generated images from text prompt: (a) SCIFAR10; (b) STinyImageNet**

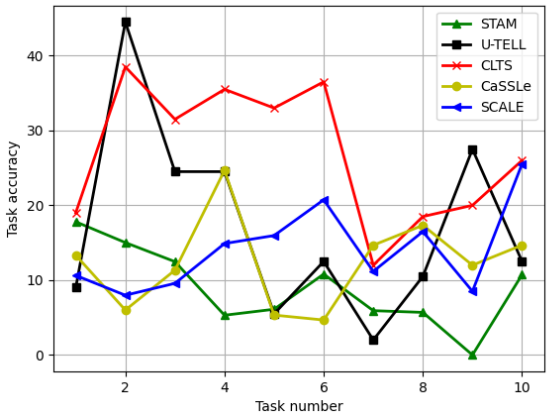


**Figure 3: Individual task performance comparison of CLTS and baselines on SCIFAR100 dataset.**

maintains stable performance, showing minimal catastrophic forgetting and outperforming the baselines on 7 out of 10 tasks. Table 2 compares the memory efficiency of the CLTS model against the selected baselines, based on the memory required to store previous task samples or centroids for replay. Unlike the baselines, which rely on storing the task exemplars or centroids, the CLTS model maintains only a pool of text captions, resulting in negligible memory usage.

We report the key hyperparameters used in our experiments. All models use a batch size between 60—100, learning rate of 0.0002, and Adam optimizer. The loss weights $\lambda_1$ and $\lambda_2$ are set to 1.0, the number of clusters varies from 2—20, and the latent dimension is fixed at 128. Training epochs range between 50—350 depending on the dataset. The TH module has 3 layer CNN architecture, while the BLIP and SD modules are used as frozen without fine-tuning. The hardware setup used for the experiments consisted of an Intel® Xeon® CPU E5-2630 v4 (2.20 GHz) and an NVIDIA Quadro RTX 8000 GPU, equipped with 48 GB of memory and 4608 cores.

**Table 2: Memory efficiency comparison in MB**

| | Method | | | |
|---|---|---|---|---|
| Dataset | SCALE | UPL-STAM | U-TELL | **CLTS** |
| SCIFAR10 | 15.72 (+6287%) | 3.09 (+1235%) | 0.17 (+67%) | **0.0025** |
| STinyImageNet | 62.91 (+1497%) | 5.36 (+126.62%) | 0.17 (+3.05%) | **0.042** |

**Ablation Study.** We conduct an ablation study on CLTS model to analyse the effect of replacing BLIP captioning with two alternatives, ViT+GPT2 and class label names as captions. We also evaluate the effect of replacing SD module with DALL·E mini for image generation. ViT+GPT2 model combines a pre-trained ViT (Vision Transformer) image encoder with a GPT2 (Generative Pre-trained Transformer) decoder, and is primarily fine-tuned for image captioning task [2]. In our ablation experiments conducted on SCIFAR10 and STinyImageNet, we used the ViT+GPT2 model as a one-to-one replacement for BLIP. The superior caption quality of BLIP is enabled by its contrastive pre-training and multi-task fine-tuning, led to improved performance of CLTS, as shown in Table 3. In our second experiment, BLIP-generated captions were replaced with single-word class label names. These label-based captions lack the descriptive qualities that SD module relies on, which led to inferior performance as shown in Table 3. In our third experiment, we used DALL·E mini as a one-to-one replacement for the SD module to analyse the impact of image generation on CLTS performance. DALL·E mini combines a BART-like (Bidirectional and Auto-Regressive Transformer) model for mapping text to image tokens with a pre-trained VQGAN decoder (Vector-Quantized Generative Adversarial Network) for image synthesis [3]. As shown in the Table 3, CLTS performs better with the SD module, due to the higher-quality images generated by the diffusion process, compared to noisy and unnaturally textured images created by DALL·E mini.

**Table 3: Ablation study**

| | Method | | | |
|---|---|---|---|---|
| Dataset | CLTS-BLIP +(ViT+GPT2) | CLTS-BLIP +Label names | CLTS–SD +DALL·E mini | **CLTS** |
| SCIFAR10 | 26.99 ± 1.09 | 24.68 ± 2.46 | 17.87 ± 1.12 | **32.77 ± 0.97** |
| STinyImageNet | 29.55 ± 1.23 | 26.35 ± 1.30 | – | **35.2 ± 1.43** |

**Discussion.** The performance of the proposed CLTS model highlights the importance of high-quality, caption conditioned image generation in continual learning. CLTS can handle frequent task arrivals by storing only the lightweight task-specific captions and enabling fast adaptation through generative replay, thereby avoiding large memory buffers. Furthermore, the use of BLIP introduces negligible overhead, while the SD-based image generation is a per-task, sample-size dependent dataset preparation step. Although TR training must wait for the generated dataset to be ready, this cost is incurred only once per task, after which the data can be reused across multiple training runs. On the downside, since the pre-trained models are typically trained on common everyday images (e.g., animals, vehicles, everyday objects), applying CLTS to specialized domains such as satellite imagery, medical imaging, or industrial defect detection may require additional fine-tuning of the pre-trained modules, leading to increased training complexity.

## 5 Conclusion

We propose CLTS, a modular continual learning framework that combines Task Heads (TH), Task Router (TR) and pre-trained vision-language modules BLIP and Stable Diffusion (SD). BLIP generates text captions from task training samples, which are stored in caption pool in place of raw image samples. During replay, these captions serve as prompts for SD module to generate synthetic images, enabling TR training. In Class-IL setting, CLTS outperforms baselines U-TELL, UPL-STAM, CaSSLe and SCALE on SCIFAR10, SCIFAR100 and STinyImageNet datasets. CLTS architecture supports scalable task expansion and rapid adaptation, making it suitable for evolving real-world scenarios.

## 6 GenAI Usage Disclosure

ChatGPT (OpenAI, GPT-4o, 2025) was used to clarify language, composition, and grammar corrections, under the supervision and final editorial control of the authors. No AI-generated content was used without thorough human review, and the tool was not used to generate code, figures, or fabricate data, results.

## References


[1] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, IEEE Corporate Headquarters (NY), 3366–3375.

[2] NLP Connect. 2022. vit-gpt2-image-captioning (Revision 0e334c7). doi:10.57967/hf/0222

[3] Boris Dayma, Suraj Patil, Pedro Cuenca, Khalid Saifullah, Tanishq Abraham, Phúc Lè Khác, Luke Melas, and Ritobrata Ghosh. 2021. DALL-E Mini. doi:10.5281/zenodo.5146400

[4] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence* 44, 7 (2021), 3366–3385.

[5] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. 2020. Adversarial continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer, 386–402.

[6] Enrico Fini, Victor G Turrisi Da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. 2022. Self-supervised models are continual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9621–9630.

[7] Rui Gao and Weiwei Liu. 2023. Ddgr: Continual learning with deep diffusion-based generative replay. In *International Conference on Machine Learning*. PMLR, 10744–10763.

[8] Quentin Jodelet, Xin Liu, Yin Jun Phua, and Tsuyoshi Murata. 2023. Class-incremental learning using diffusion model for distillation and replay. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3425–3433.

[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. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[10] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*. PMLR, 12888–12900.

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

[12] Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 7765–7773.

[13] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural networks* 113 (2019), 54–71.

[14] Mahardhika Pratama, Andri Ashfahani, and Edwin Lughofer. 2021. Unsupervised Continual Learning via Self-adaptive Deep Clustering Approach. In *International Workshop on Continual Semi-Supervised Learning*. 48–61.

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

[16] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in Neural Information Processing Systems* 32 (2019).

[17] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.

[18] Mohammad Rostami, Soheil Kolouri, Praveen Pilly, and James McClelland. 2020. Generative continual concept learning. In *Proceedings of the AAAI conference on artificial intelligence*. 5545–5552.

[19] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *CoRR* (2016).

[20] J Senthilnath, Adithya Bhattiprolu, Ankur Singh, Bangjian Zhou, Min Wu, Jón Atli Benediktsson, and Xiaoli Li. 2024. Evolving Restricted Boltzmann Machine-Kohonen Network for Online Clustering. *arXiv preprint arXiv:2402.09167* (2024).

[21] James Smith, Cameron Taylor, Seth Baer, and Constantine Dovrolis. 2021. Unsupervised progressive learning and the STAM architecture. In *International Joint Conference on Artificial Intelligence*.

[22] Indu Solomon, Aye Phyu Phyu Aung, Uttam Kumar, and Senthilnath Jayavelu. 2024. U-TELL: Unsupervised Task Expert Lifelong Learning. *ICIP* (2024).

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

[24] Xiaofan Yu, Yunhui Guo, Sicun Gao, and Tajana Rosing. 2023. SCALE: Online Self-Supervised Lifelong Learning without Prior Knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2483–2494.

[25] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. 2024. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).