Joint Diffusion Models in Continual Learning

Paweł Skierś
Warsaw University of Technology
pawel.skiers.stud@pw.edu.pl

Kamil Deja
Research Institute IDEAS
Warsaw University of Technology
kamil.deja@pw.edu.pl

Abstract

In this work, we introduce JDCL – a new method for continual learning with generative rehearsal based on joint diffusion models. Neural networks suffer from catastrophic forgetting defined as an abrupt loss in the model's performance when retrained with additional data coming from a different distribution. Generative-replay-based continual learning methods try to mitigate this issue by retraining a model with a combination of new and rehearsal data sampled from a generative model. In this work, we propose to extend this idea by combining a continually trained classifier with a diffusion-based generative model into a single – jointly optimized neural network. We show that such shared parametrization, combined with the knowledge distillation technique, allows for stable adaptation to new tasks without catastrophic forgetting. We evaluate our approach on several benchmarks, where it outperforms recent state-of-the-art generative replay techniques. Additionally, we extend our method to the semi-supervised continual learning setup, where it outperforms competing buffer-based replay techniques, and evaluate, in a self-supervised manner, the quality of trained representations.

1. Introduction

Contemporary deep neural networks can be trained to human-level performance on a variety of different problems. However, contrary to humans, when retraining the same models to learn an additional task, neural networks suffer from *catastrophic forgetting* [15] defined as an abrupt loss in performance on the previous task when adapting to a new one. Continual learning approaches try to mitigate this issue. In particular, replay-based techniques retrain the model with a mix of new data examples and samples from the past tasks stored in the memory buffer [1, 48]. However, since the memory buffer has to constantly grow, new methods [58, 69] use generative models to generate synthetic versions of past experiences.

In the majority of the recent works, the generative replay technique follows the simple algorithm:

Algorithm 1 Generative replay

Input: G, C, sequence of tasks $\{D_i\}_{i=1}^{N}$
$G \leftarrow$ **train_generative_model**(G, D_1)
$C \leftarrow$ **train_classifier**(C, D_1)
for i in $2 \dots N$ **do**
  $S \leftarrow$ **sample_dataset**(G)
  $B \leftarrow D_i \cup S$
  $G \leftarrow$ **train_generative_model**(G, B)
  $C \leftarrow$ **train_classifier**(C, B)
end for

Such an approach introduces an imbalance between the generative model and the classifier. The generative model plays a crucial role as a knowledge consolidation method that stores information about old tasks, while the classifier is highly dependent on the quality of generated samples, and the effectiveness of the knowledge transfer from the generative part of the system. Consequently, several studies [8, 32, 73] suggest that it may be advantageous to reset the classifier's weights entirely and retrain it from scratch for each task, using a blend of new data and sampled generations – a solution that contradicts the principles of continual learning.

At the same time, stable classifier retraining using only generated data remains a challenge. Even state-of-the-art diffusion-based generative models are known to have significant limitations in precise modeling of data distribution [56], which results in the degradation of the classifier's performance.

In this work, we propose to address this limitation by introducing a joint model that combines the generative and discriminative models into a single parametrization. In particular, we extract the latent features from the UNet [49] trained as a denoising diffusion model and use them for classification. We optimize our joint model with a sum of generative and discriminative objectives. Such an approach directly removes the need for transferring the knowledge from the generative model to the classifier when retraining the model on a sequence of tasks.

To visualize the main benefit of this approach, we conduct a simple experiment that highlights the drawbacks of

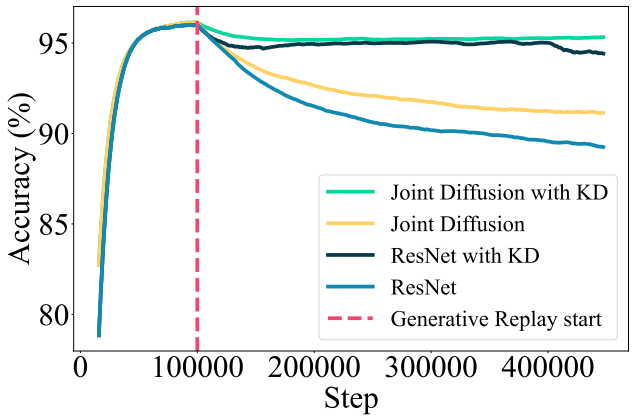

Figure 1. Further training of a classifier using data generated from the diffusion model trained with the same dataset harms the classifier's performance (blue line). However, joint modeling, especially in combination with knowledge distillation, significantly limits this degradation and retains almost the initial performance. We report accuracy averaged across three different seeds.

recent techniques. We train the classifier and our joint-diffusion model (ensuring that both models are of similar sizes) on the CIFAR10 dataset until convergence (for 100,000 optimizer steps)[1]. Then, we continue to optimize the models, but instead of using real data, we sample data from the diffusion model. As presented in Fig. 1, the performance of the classifier (blue line), drastically decreases as the effect of the rehearsal with generated samples. At the same time, the performance of our joint diffusion (yellow) converges after a much smaller drop in performance. Furthermore, the addition of knowledge distillation significantly decreases the degradation of both models, while keeping the trend of the joint diffusion model outperforming the classifier in terms of final performance and stability. We conclude that a combination of joint modeling and knowledge distillation is key for knowledge retention in generative replay-base CL.

Therefore, we introduce a joint-diffusion continual learning method, that in combination with knowledge distillation and two-stage training allows for high-quality modeling in CL scenarios. In our experiments, we show that JDCL achieves state-of-the-art performance when compared with other generative replay methods, while significantly reducing the computational cost. We show that this performance gain is attributed to the higher quality of rehearsal samples when compared with the standard diffusion model. Moreover, we show that our approach can be easily applied in the continual semi-supervised setup, where thanks to the shared parametrization we can observe an alignment between labeled and unlabeled data. In this task, we show the superiority of our method over the buffer-based techniques. Finally, we also highlight the quality and stability of representations built by our method by evaluating it in a self-supervised manner. Our main contributions can be summarized as follows:

- We introduce JDCL – a joint diffusion continual learning approach that mitigates the problem of knowledge transfer between generative and discriminative models in generative replay.
- We show that in class-incremental setup, our approach outperforms recent state-of-the-art generative replay techniques and approaches the soft-upper bound defined by a method with an infinite memory buffer, while significantly reducing the computational cost.
- We extend our studies to continual representation and semi-supervised learning where we compare JDCL against buffer-based rehearsal techniques.

2. Related Work

Continual Learning The increasing size of machine learning models, and hence the increased computational cost of their training highlights the need for effective methods for model updates. Therefore, we can observe a growing number of continual learning techniques [43, 45] that aim to mitigate the problem of catastrophic forgetting [15], without reducing the ability to adjust to new data. CL methods can be categorized into three groups: (1) Regularization-based approaches such as EWC [26], LwF [34] reduce the interference from the new tasks on the parts of the neural network that are crucial for the previous ones. (2) Architecture-based methods (e.g.RCL [76], PNN [53]) directly build different model versions for different tasks. Finally (3) Replay techniques (e.g. [1, 48]) use a limited number of past samples as *rehearsal* examples, by mixing them with new data samples.

Generative models in continual learning A significant drawback of replay approaches is that storing past samples requires a growing memory buffer. Therefore, Shin et al. [58] proposed to replace it with a Generative Adversarial Network [19]. This idea was further extended to VAEs [42, 67], GMMs [51], or Diffusion-based generative models [7, 17]. In [32] authors provide a summary of different generative replay techniques. Apart from different architectures, similar to our idea, an interesting approach is proposed in Replay Through Feedback (RTF) [67], where authors combine the variational autoencoder with a classifier. This improves the performance of the final solution, which is in line with results presented in [40, 65], where authors show that additional reconstruction or generative loss added to the classifier alleviates catastrophic forgetting.

Several works further extend the basic generative replay technique. In [69], followed by [35], authors propose to replay internal data representations instead of the original inputs. This approach improves model stability but limits its adaptation to new tasks. On the other hand, several

[1]Details on this experiment are provided in the Appendix

methods drew inspiration from neuroscience and introduced techniques that mimic the human memory system by dividing the update of the model into two [9, 23] or even three stages [70].

Recently, diffusion-based generative models (DDGMs) [20, 60] have come to the forefront as a state-of-the-art generative method. Therefore, there is growing research on continual learning with DDGMs. Zajac et al. [78] provide an overview of how DDGMs work when combined with baseline CL methods. In [17] authors employ DDGMs with classifier guidance [12] as a source of rehearsal examples. This idea is further extended in [7], where authors overview different sampling strategies with DDGMs. There are several methods that explore the possibilities for efficient updates of large-scale text-to-image diffusion models. In DreamBooth [52] authors introduce a method for finetuning text-to-image DDGMs with *Prior Preservation Loss*, that employs generative replay to mitigate catastrophic forgetting. Similarly, [59] extend this idea with Low Rank Adaptation (LORA [21]) technique, while in [57] apply this method for object and style update. On the other hand, Basu et al. [3], show that knowledge in text-to-image models is highly localized and can be updated directly in the attention layers of the text encoder.

H-space in diffusion models Several methods explore the potential of using the latent representations within the U-Net model used as a denoising decoder in DDGM. In particular, Kwon et al. [29] show that the bottleneck of the U-Net dubbed *H-space* can be used as a semantic latent space for image manipulation. This idea is further extended in [46]. Similarly, some techniques employ the H-space in downstream tasks outside of generative modeling. Authors of [2], [66] and [50], show that such features can be used for image segmentation, in [37] Luo et al. extend this idea to image correspondence, while [10] use H-features for classification. In this work, we relate to the utilization of H-space by employing it in a continually trained joint diffusion model with a classifier.

Semi-supervised learning Semi-supervised methods confront the problem of training models on large datasets where only a few samples have associated annotations [61, 77]. Recently, most of the best-performing semi-supervised methods belong to the family of hybrid methods that combine consistency regularization and pseudo-labeling [4, 28, 61, 77, 79]. The main idea behind consistency regularization is an assumption that the model should yield similar outputs when fed perturbed versions of the same image or when its weights are slightly perturbed [30, 41, 54, 64]. On the other hand, pseudo-labeling techniques exploit the concept of using synthetic labels for unlabeled data. More precisely, this involves utilizing "hard" labels (i.e., the arg max of the model's output) for which the predicted class probability surpasses a predetermined threshold [6, 31, 47, 72, 75, 77]. Apart from consistency regularization and pseudo-labeling, a third group of SSL methods employs additional deep generative models [77] such as Variational Auto-Encoder (VAE) [13, 25, 33, 38, 44], or Generative Adversarial Networks (GAN) [16, 36, 55, 62, 74], to learn the underlying class-conditional distribution of partially labeled data.

Despite their proliferation, as noted by [24], the standard semi-supervised approaches do not address the problem of learning in a continual setup. Consequently, a new line of research called continual semi-supervised learning emerged to address the challenges of the combination of those two domains. To that end, [71] introduced ORDisCo, which utilizes a conditional GAN with a classifier to learn from partially labeled data in an incremental setup. On the other hand, [5] proposed CCIC, a method that enforces consistency by contrasting samples among different classes and tasks. In contrast, the recent state-of-the-art method called NNCSL [24] leverages the nearest-neighbor classifier to learn the current task, while also retaining the relevant information from the previous tasks. In this work, we adapt the mechanism of pseudo-labeling and consistency regularization, which we combine with our joint diffusion model to combat the problems of semi-supervised continual learning.

3. Background

3.1. Continual Learning

In this work, we tackle the problem of continual learning, which as a machine learning paradigm assumes that data used for training of a function f is provided in separate portions known as tasks. Particularly, in class-incremental learning [68], we assume that each task has a disjoint set of non-overlapping labels. In this setup, the objective is to minimize the prediction error on all of the labels without any additional information such as the identity of the associated task.

3.2. Joint Diffusion Model

We follow the parametrization of joint diffusion models introduced in [10], where features from the different levels of the UNet architecture are pooled into a single vector z used for classification. Let p_θ be the denoising model, which is of the UNet architecture, with parameters θ. As such, the denoising model consists of the encoder e_ν and the decoder d_ψ with parameters ν and ψ respectively, such that $\theta = \{\nu, \psi\}$. The encoder takes input x_t and returns a set of tensors $\mathcal{Z}_t = e_\nu(x_t)$ that is, a set of representation tensors $\mathcal{Z}_t = \{z_t^1, z_t^2 \dots z_t^n\}$ obtained from each depth level of the UNet architecture. The decoder reconstructs the denoised sample x_{t-1}, from the set of representations, i.e., $x_{t-1} = d_\psi(\mathcal{Z}_t)$. Additionally to the standard diffusion de-

noiser, we propose to pool the representation set into a single vector $z_t = f(\mathcal{Z}_t)$ via average pooling. Finally, we introduce classifier g_ω, with parameters ω - the final part of the model that predicts target class $\hat{y} = g_\omega(z_t)$.

We optimize the whole model jointly by modeling the joint probability as follows:

$$p_{\nu,\psi,\omega}(x_{0:T}, y) = p_{\nu,\omega}(y|x_0)\, p_{\nu,\psi}(x_{0:T}), \tag{1}$$

which after applying the logarithm yields:

$$\ln p_{\nu,\psi,\omega}(x_{0:T}, y) = \ln p_{\nu,\omega}(y|x_0) + \ln p_{\nu,\psi}(x_{0:T}). \tag{2}$$

We use the logarithm of the joint distribution (2) as the training objective, where $\ln p_\theta(x_{0:T})$ is approximated by the simplified DDGM objective [22]:

$$L_{t,\text{diff}}(\nu, \psi) = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}\|^2\right], \tag{3}$$

where ϵ is the noise added to the initial image and $\hat{\epsilon}$ is a prediction from the decoder:

$$\{z_t^1, z_t^2 \dots z_t^n\} = e_\nu\left(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t\right) \tag{4}$$

$$\hat{\boldsymbol{\epsilon}} = d_\psi(\{z_t^1, z_t^2 \dots z_t^n\}). \tag{5}$$

For the classifier, we employ the logarithm of the categorical distribution:

$$L_{\text{class}}(\nu, \omega) = -\mathbb{E}_{\mathbf{x}_0,y}\left[\sum_{k=0}^{K-1} \mathbb{1}[y=k] \log \frac{\exp(\varphi_k)}{\sum_{c=0}^{K-1}\exp(\varphi_c)}\right] \tag{6}$$

which corresponds to the cross-entropy loss, where y represents the target class, φ is a vector of probabilities generated by the classifier $g_\omega(e_\nu(x_0))$ and $\mathbb{1}[y=k]$ is the indicator function that is 1 if y equals k and 0 otherwise. The final loss function that we optimize with a single optimizer over parameters $\{\nu, \psi, \omega\}$ is:

$$L_{JD}(\nu, \psi, \omega) = \alpha \cdot L_{\text{class}}(\nu, \omega) - L_0(\nu, \psi) - \sum_{t=2}^{T} L_{t,\text{diff}}(\nu, \psi) - L_T(\nu, \psi), \tag{7}$$

To train the model over a batch of data examples (x_0, y), we first noise the example x_0 with a forward diffusion to a random timestep x_t, so that the training loss for the denoising model is a Monte-Carlo approximation of the sum over all timesteps. After that, we feed x_0 to a classifier and calculate the cross-entropy loss on returned probabilities φ and the given class labels y.

4. Method

Following the introduction in the previous section, we introduce JDCL - a new continual learning method with generative replay that allows for stable adaption to new tasks. Our approach takes advantage of three main ideas: joint-diffusion parametrization, two-stage local-to-global training, and knowledge distillation.

Joint diffusion modeling in CL The main component of our solution is based on the joint-diffusion model. We adapt the method described in section Sec. 3.2 to the generative replay strategy. We train jointly a single model with generative and discriminative objectives, and benefit from this fact, by using the generative properties of our model to generate rehearsal samples.

Two-stage local-to-global training One of the crucial problems in continually trained neural networks is the plasticity-stability dilemma. As presented in Fig 1, we can easily achieve high stability of our continually trained model through a self-rehearsal mechanism combined with knowledge distillation. However, to improve the plasticity, we relate to the two-stage training [9, 18] where we first fine-tune the copy of the current model using only newly available data, which results in perfect plasticity. Then, we distill the adapted model with the previous one into a new global model as presented in Fig. 2.

More precisely, we first create a copy of our current joint diffusion model $p_{\nu,\psi,\omega}(x_{0:T}, y)$, and train it with the currently available dataset D_τ. We call that model *local* and we optimize it with the loss provided in Equation 7.

Following that, we move to the global stage. We begin by generating a rehearsal dataset $S_{1\dots\tau-1}$ for tasks 1 to $\tau-1$ by sampling from the original *global* joint diffusion model. We then combine this dataset with a **synthetic dataset** S_τ, which we generate with the *local* model. All the samples are generated unconditionally by the diffusion part of the joint diffusion. The labels for those generations are then assigned by the classifier part of the appropriate models.

Finally, to acquire the next *global* model, we fine-tune the previous *global* joint diffusion on the combined synthetic dataset with a joint diffusion loss L_{JD} defined in Equation 7.

Knowledge distillation To further mitigate catastrophic forgetting and improve the model's stability, on top of the standard joint diffusion loss, we apply the knowledge distillation technique. To that end, we adapt standard KD loss to our joint model, by summing up separate parts from the diffusion and classifier. For a frozen joint diffusion model p^f, updated model $p_{\nu,\psi,\omega}(x_{0:T}, y)$ and dataset D we define the diffusion knowledge distillation loss as:

$$L^{KD}_{t,\text{diff}}(\nu, \psi) = \mathbb{E}_{\mathbf{x}_0\in D,\boldsymbol{\epsilon_f}}\left[\|\boldsymbol{\epsilon_f} - \hat{\boldsymbol{\epsilon}}\|^2\right], \tag{8}$$

where $\boldsymbol{\epsilon_f}$ is the noise predicted by the frozen model p^f.

Classification knowledge distillation loss $L^{KD}_{\text{class}}(\nu, \omega)$ is then a cross-entropy loss between the vectors of probabilities φ^f and φ assigned for the same input by models p^f, and $p_{\nu,\psi,\omega}$ respectively:

$$L^{KD}_{\text{class}}(\nu, \omega) = -\mathbb{E}_{\mathbf{x}_0}\left[\sum_{k=0}^{K-1} \log \frac{\exp(\varphi_k)}{\sum_{c=0}^{K-1}\exp(\varphi_c)}\varphi_k^f\right], \tag{9}$$

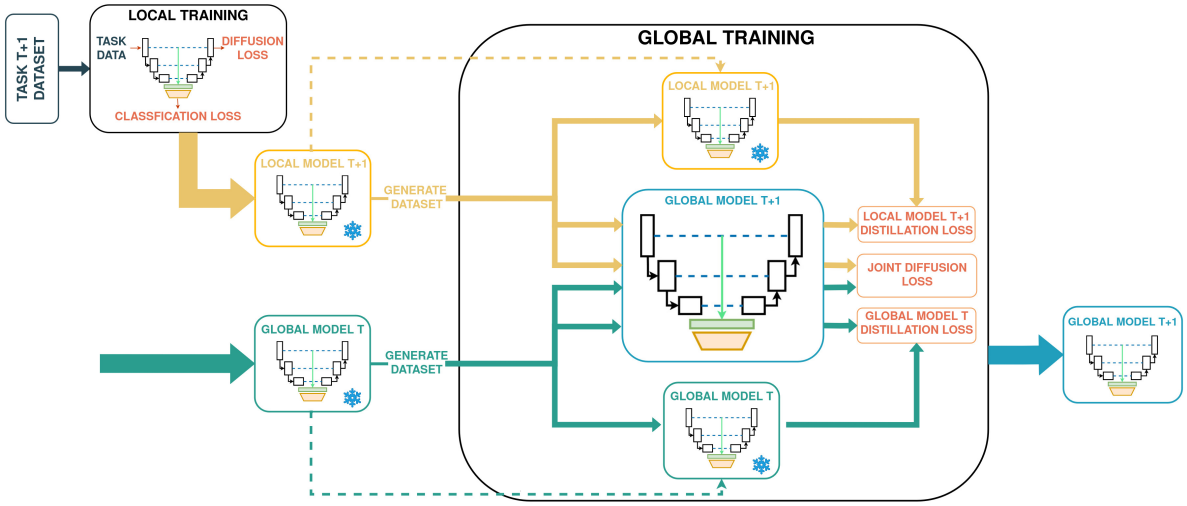

Figure 2. Overview of JDCL. To adapt the global model to task $\tau+1$, we first train a local joint diffusion model on task $\tau+1$ only. We then use the global model and the local model to generate a synthetic dataset consisting of samples from tasks $1, 2 \ldots \tau+1$. Finally, we fine-tune the global model with the generated dataset and a combination of joint diffusion and knowledge distillation losses.

Therefore, the full joint diffusion knowledge distillation loss is defined as a sum:

$$\begin{aligned} L_{JDKD}(\nu,\psi,\omega;p^f) =& \alpha_{KD}\cdot L^{KD}_{\text{class}}(\nu,\omega) - L^{KD}_0(\nu,\psi) \\ &- \sum_{t=2}^{T} L^{KD}_{t,\text{diff}}(\nu,\psi) - L^{KD}_T(\nu,\psi), \end{aligned} \tag{10}$$

where α_{KD} is a knowledge distillation classification loss scale.

Finally, the training objective, which we optimize to obtain the *global* model for task t can be expressed as:

$$\begin{aligned} L_{CL}(\nu,\psi,\omega) =& \mathbb{E}_{\mathbf{x}_0,y\in S_{1\ldots t-1}}\left[L_{JDKD}(\nu,\psi,\omega;p^f)\right] \\ &+ \mathbb{E}_{\mathbf{x}_0,y\in S_t}\left[L_{JDKD}(\nu,\psi,\omega;p_n)\right] \\ &+ \beta\cdot\mathbb{E}_{\mathbf{x}_0,y\in S_{1\ldots t}}\left[L_{JD}(\nu,\psi,\omega)\right], \end{aligned}$$

where p^f is the previous frozen *global* joint diffusion model, p_n is the frozen *local* model, $S_{1\ldots t}$ is the synthetic dataset for tasks 1 to t and β is a non-knowledge-distillation loss scale.

Joint Diffusion Model in semi-supervised setup One of the biggest advantages of our approach is its flexibility. The design of JDCL specifically allows for stable consolidation of knowledge from two fully independent models. That means that during the *local* training stage – the only phase that deals with the data from the current task – we can train the *local* model without any consideration for the incremental objective. This observation allows us to easily adapt our method to the semi-supervised continual setup. In this section, we describe changes in the *local* training stage that allow us to continually train the model on partially labeled data.

First of all, thanks to our joint modeling, we can benefit from our shared parametrization and simply use the unlabeled examples only with the generative part of our model. However, to further improve the results, we propose to combine our joint modeling with consistency regularization and pseudo-label techniques introduced in [61]. To that end, similarly to [61] we use two different sets of augmentations. With weak augmentations α, we assign pseudo-labels to the unlabeled examples, and then, with strong augmentations A, we calculate additional training objective $L_{\text{ssl}}(\nu,\omega)$. To obtain an artificial label q_b for unlabeled data x_u we calculate the argmax of our model's prediction for weakly augmented image φ_q: $\varphi_q = g_\omega(e_\nu(\alpha(x_\text{u})))$; $\hat{q}_b = \arg\max(\varphi_q)$. Then, we calculate the semi-supervised classification loss l_{ssl} as a cross-entropy loss between the model's output for a strongly-augmented data and the assigned pseudo-label:

$$\begin{aligned} L_{\text{ssl}}(\nu,\omega) =& -\mathbb{E}_{\mathbf{x}_0,y}\{\mathbb{1}[max(\varphi_q)\geq\tau] \\ & H(\hat{q}_b, g_\omega(e_\nu(A(x_\text{u}))))\} \end{aligned} \tag{11}$$

where τ corresponds to the threshold below which we discard a pseudo-label and $H(p,q)$ denotes the cross-entropy between two probability distributions p and q.

The full loss function in our semi-supervised approach is then the following:

$$\begin{aligned} L(\nu,\psi,\omega) =& \alpha\cdot L_{\text{class}}(\nu,\omega) + \alpha\cdot L_{\text{ssl}}(\nu,\omega) - L_0(\nu,\psi) \\ &- \sum_{t=2}^{T} L_{t,\text{diff}}(\nu,\psi) - L_T(\nu,\psi). \end{aligned}$$

Training the model in a semi-supervised setup, over a batch of data (x_0, y, u_0) proceeds very similarly to the training in a fully supervised setup. Firstly, we follow the train-

Table 1. Comparison of our JDCL with non-generative distillation-based techniques and generative rehearsal methods, including feature replay techniques that we mark in gray color. Results of competing methods from [7, 63].

METHOD	CIFAR-10	CIFAR-100		IMAGENET100
	$T=5$	$T=5$	$T=10$	$T=5$
JOINT	93.14 ± 0.16	72.32 ± 0.24		66.85 ± 2.25
CONTINUAL JOINT	86.41 ± 0.32	73.07 ± 0.01	64.15 ± 0.98	50.59 ± 0.35
FINE-TUNING	18.95 ± 0.20	16.92 ± 0.03	9.12 ± 0.04	13.49 ± 0.18
EWC	20.08 ± 0.51	22.28 ± 1.03	13.95 ± 1.01	21.35 ± 0.26
GKD (LWF)	49.60 ± 0.42	37.46 ± 1.18	27.91 ± 0.47	33.90 ± 0.25
MKD (ICARL)	50.37 ± 2.15	34.01 ± 0.69	25.95 ± 0.50	33.03 ± 0.23
TKD (SS-IL)	50.46 ± 1.96	38.47 ± 0.83	28.96 ± 0.19	35.22 ± 0.22
ANCL	44.70 ± 3.24	29.46 ± 0.28	18.76 ± 0.95	29.40 ± 0.23
DGR VAE	28.23 ± 3.84	19.66 ± 0.27	10.04 ± 0.17	9.54 ± 0.26
DGR+DISTILL	27.83 ± 1.20	21.38 ± 0.61	13.94 ± 0.13	11.77 ± 0.47
RTF	30.36 ± 1.40	17.45 ± 0.28	12.80 ± 0.78	8.03 ± 0.05
MERGAN	51.65 ± 0.40	9.65 ± 0.14	12.34 ± 0.15	-
BIR	36.41 ± 0.82	21.75 ± 0.08	15.26 ± 0.49	8.63 ± 0.19
GFR	26.70 ± 1.90	34.80 ± 0.26	21.90 ± 0.14	32.95 ± 0.35
DDGR	43.69 ± 2.60	28.11 ± 2.58	15.99 ± 1.08	25.59 ± 2.29
DGR DIFFUSION	59.00 ± 0.57	28.25 ± 0.22	15.90 ± 1.01	23.92 ± 0.92
GUIDE	64.47 ± 0.45	41.66 ± 0.40	26.13 ± 0.29	39.07 ± 1.37
JDCL	$\mathbf{83.69} \pm 1.44$	$\mathbf{47.95} \pm 0.61$	$\mathbf{29.04} \pm 0.41$	$\mathbf{54.53} \pm 2.15$

ing steps from the supervised setup, with one exception - instead of noising and then calculating the denoising model's training loss on x_0 we use both x_0 and u_0. Following that, we compute pseudo-labels for unlabeled data and use them to calculate the semi-supervised loss.

5. Experiments

5.1. Experimental Setup

Datasets We evaluate our method on three commonly used datasets – CIFAR10, CIFAR100 [27] and ImageNet100 [11]. The first two of those datasets contain 50000 training and 10000 test images of size 32×32 divided into 10 or 100 classes. ImageNet100 on the other hand contains 120000 train images in 64×64 resolution from 100 different classes. For the class-incremental setup, we split the CIFAR10 and ImageNet100 datasets into 5 equal tasks and the CIFAR100 dataset into 5 or 10 equal tasks. For semi-supervised setups, we train our models on two different ratios of labeled data in the dataset, i.e. 0.8% and 5%. Those ratios correspond to 4 and 25 labeled examples per class in CIFAR100, and to 40 and 250 samples for CIFAR10.

Implementation details We use the same joint diffusion model architecture in all experiments on CIFAR10 and CIFAR100, and an upscaled version of it for ImageNet100 experiments to adjust to the bigger dataset resolution. For the UNet, we follow the implementation provided in [12]. For the classifier, we use a 2-layer feed-forward net with a hidden layer of 1024 and a LeakyRELU activation function. The detailed description of the training hyperparameters is provided in the Appendix, and in the code repository[2].

[2]https://github.com/pskiers/Joint-Diffusion-in-Latent-Space

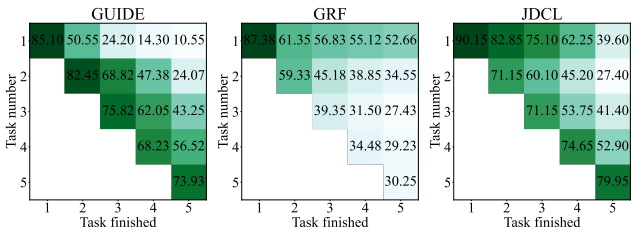

Figure 3. Accuracy on each task after each phase of incremental training on CIFAR100 with 5 tasks.

5.2. Main results

To compare our method with other generative rehearsal approaches, we measure its performance on several CL benchmarks. In Tab. 1, we show that our method outperforms other evaluated generative replay techniques in terms of average accuracy after the last task. On CIFAR10 and ImageNet100, our method approaches the soft upper limit, calculated as a continual joint method trained with an infinitely long replay buffer. On those benchmarks, we surpass the previous state-of-the-art score by over 19 and 15 points, which is an increase of around 30% and 40% for CIFAR10 and ImageNet100 respectively.

To better understand the advantages of JDCL over the competing methods, we perform a detailed comparison of our method against the GFR and GUIDE. To that end, in Fig. 3 we present the accuracy of each approach, on each task after each training phase on CIFAR100 with 5 tasks. JDCL significantly outperforms GUIDE in terms of knowledge retention from preceding tasks. This is especially visible when we examine the change in performance of the methods on the older tasks. Whereas GUIDE's knowledge obtained during the first task is quickly forgotten and is almost completely lost at the end of the training, JDCL is able to maintain a good accuracy on the first task for the whole training process. Notably, the performance degradation in our method seems to be related rather to the increasing complexity of the problem instead of forgetting, as we do not see fast degradation on the oldest tasks. On the other hand, when comparing to the GFR, we can observe that our method, though worse at preserving knowledge, is much better at assimilating new information. This phenomenon can be attributed to the significantly limited plasticity of the GFR that slows down the training of its feature extractor and restricts its update to the feature distillation loss. Consequently, the model has a substantially worse final performance.

Semi-supervised continual learning Apart from the standard supervised setup, we evaluate our method on a set of semi-supervised continual scenarios and compare it to the existing approaches. As shown in Tab. 2 JDCL outperforms other methods on all CIFAR10 setups, while also achieving comparable accuracy on CIFAR100. It is worth

Table 2. Average accuracy with a standard deviation of different methods tested with 5-task CIFAR-10 and 10-task CIFAR-100 semi-supervised settings. The number in brackets indicates the size of the memory buffer for the labeled data. Results of competing approaches from [24]

Method	CIFAR-10		CIFAR-100	
	0.8% labeled	5% labeled	0.8% labeled	5% labeled
Fine-tuning	13.6±2.9	18.2±0.4	1.8±0.2	5.0±0.3
ER (500)	36.3±1.1	51.9±4.5	8.2±0.1	13.7±0.6
iCaRL (500)	24.7±2.3	35.8±3.2	3.6±0.1	11.3±0.3
FOSTER (500)	43.3±0.7	51.9±1.3	4.7±0.6	14.1±0.6
X-DER (500)	33.4±1.2	48.2±1.7	8.9±0.3	18.3±0.5
PseudoER (500)	50.5±0.1	56.5±0.6	8.7±0.4	11.4±0.5
CCIC (500)	54.0±0.2	63.3±1.9	11.5±0.7	19.5±0.2
PAWS (500)	51.8±1.6	64.6±0.6	16.1±0.4	21.2±0.4
CSL (500)	64.5±0.7	69.6±0.5	23.6±0.3	26.2±0.5
NNCSL (500)	73.2±0.1	77.2±0.2	**27.4**±0.5	**31.4**±0.4
PseudoER (5120)	55.4±0.5	70.0±0.3	15.1±0.2	24.9±0.5
CCIC (5120)	55.2±1.4	74.3±1.7	12.0±0.3	29.5±0.4
ORDisCo (12500)	41.7±1.2	59.9 ±1.4	-	-
CSL (5120)	64.3±0.7	73.1±0.3	23.7±0.5	41.8±0.4
NNCSL (5120)	73.7±0.4	79.3±0.3	**27.5**±0.7	**46.0**±0.2
JDCL	**78.93**±0.72	**79.96**±0.86	22.19±0.3	26.39±1.7

Table 3. Ablation study on CIFAR10/5 depicting the importance of individual components. Our method greatly relies on the synergy of joint modeling with knowledge distillation.

Joint modeling	Knowledge distillation	2-stage training	Accuracy
✓	✓	✓	83.7
✗	✓	✓	68.4
✓	✗	✓	48.2
✓	✓	✗	36.7
✗	✗	✓	32.7

noting that JDCL is able to reach state-of-the-art performance on CIFAR10 without a memory buffer, while also in a majority of tasks sampling fewer examples from the diffusion model than the buffer size in the related works. For CIFAR100, we believe that the reduced performance of our method is caused by an extremely weak signal from the classifier, which introduces an imbalance in our joint training. To further explain the issue, we visualize how JDCL builds latent representations in the SSL setup in the appendix.

5.3. Additional Experiments

The interplay between shared parametrization, distillation, and two-stage training

In this work, we propose a new generative replay method for continual learning. Apart from the main idea of our approach, which is joint diffusion and classification modeling, we combine it with two important techniques: knowledge distillation and two-stage training. In this section, we ablate

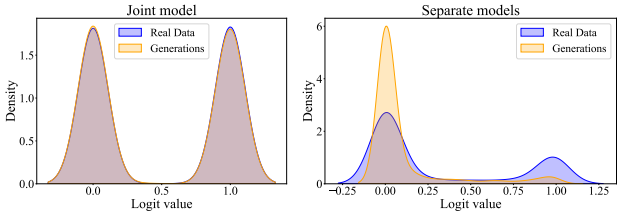

Figure 4. Difference between the distribution of logit value corresponding to class 1 from CIFAR-10 for real and synthetic data. Samples generated by the joint model closely match the real data's logits distribution, whereas those from the separately trained classifier diverge significantly.

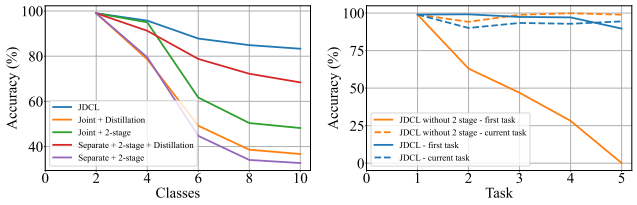

Figure 5. Mean accuracy after each task for the ablation methods (left) and accuracy of JDCL on current and the first task with and without two-stage training (right)

the impact of each component. Notably, for experiments without joint modeling, we use a separate diffusion model with the same architecture as the joint diffusion model, and ResNet-18 as a classifier. We train and distill them in the same manner as the joint model. In Tab. 3 we show the results for the model trained without individual components. As visible, high performance of JDCL is attributed to the synergy of the proposed joint modeling, knowledge distillation technique, and two-stage training. To further highlight the interplay between those three components, we perform additional experiments.

First, we emphasize the importance of joint diffusion for effective knowledge distillation. To that end, we train a joint diffusion model and a separate classifier and diffusion on a single task. In Fig. 4, we compare the classifier's logits distributions for a single class from the real data and rehearsal samples generated by the diffusion model trained either separately or jointly with the classifier. As visible, generations from the joint model closely match the real data logits distribution. However, for the separate classifier, the distributions diverge significantly. This discrepancy means that when updating the global model via knowledge distillation, the rehearsal data remains in-distribution for the joint classifier while being OOD for the separate classifier, which explains the observed performance gap.

To further highlight this synergy, in Fig 5 (left) we present how the average accuracy changes after each task for the ablated models. We observe that knowledge distillation and two-stage training can largely mitigate the absence of joint modeling on a task-to-task basis. However, in the

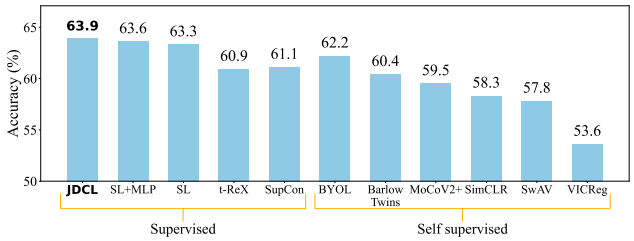

Figure 6. Accuracy of a logistic regressor trained on representations learned with different supervised and self-supervised methods on CIFAR100. Results of competing approaches from [14, 39]

Table 4. Comparison of the generation quality of JDCL and Deep Generative Replay with standard DDPM after the last task.

Method	CIFAR-10/5			CIFAR-100/5		
	FID	Precision	Recall	FID	Precision	Recall
DGR	49.56	0.54	0.34	50.36	0.48	**0.30**
JDCL	**32.12**	**0.60**	**0.43**	**39.93**	**0.71**	0.27

CL setup, the performance differences steadily accumulate, leading to significantly worse final results.

Finally, we show that the use of two-stage training with distillation significantly stabilizes the alignment of new and old knowledge. In Fig. 5 (right) we plot the accuracy on the newly observed task, and the remaining accuracy on the first one. We can observe that although two-stage training cannot achieve as high plasticity, stable distillation from a local copy of a model to the global one leads to significantly reduced forgetting in the first task.

JDCL as a continual representation learner

In this section, we show that our method is able to learn meaningful data representation in a continual setup. We follow the evaluation approach common in self-supervised continual learning techniques, as presented in [14]. In particular, we report the accuracy of a logistic regressor trained on the features provided by our model i.e. on the set of representations of the UNet model, pooled into a single vector. In Fig. 6 we compare our model with supervised and self-supervised continual methods. We can observe that the representations learned by our model are indeed very informative. Moreover, we find that JDCL outperforms other approaches on the CIFAR100 setup with 5 tasks.

Incremental generative setup comparison

Generative-replay-based continual learning methods rely heavily on the underlying generative model to preserve knowledge about the previous tasks. The quality of the samples generated by the method is therefore of utmost importance. In this section, we perform a comparison of JDCL and the diffusion model trained continually on CIFAR10 and CIFAR100, both split into 5 tasks. We present the achieved FID, Precision, and Recall after the last task in Table 4. JDCL outperforms the baseline method on all but one metric. We can observe significant improvement in the precision of generations, which results in generated rehearsal samples being clearly assigned to one of the previous classes, bringing less noisy signals to the model. As argued in [17], this trait can be the reason for the improved performance of the continually trained classifier.

Table 5. Runtime analysis of all baseline methods on the CIFAR-10/5 benchmark. Our approach is computationally more efficient than related diffusion-based methods.

Method	Time [GPU-hours]	Accuracy
DDGR	41.29	43.69
DGR diffusion	<u>28.02</u>	59.00
GUIDE	30.60	64.47
JDCL (fast)	**27.21**	<u>78.10</u>
JDCL	53.83	**83.69**

Runtime analysis of JDCL for CIFAR-10/5

One of the biggest drawbacks of generative-replay-based techniques is an excessive computational cost associated with the training of the generative model and additional time spent on sampling of the rehearsal examples. However, thanks to the joint modeling our method is very data-efficient. For example, in the case of the CIFAR dataset, at the beginning of each task, we sample only 400 examples per class for CIFAR-100 or 1000 examples per class for CIFAR10. This occupies approximately 13% of the total GPU time spent on each task. The combination of the joint model training additionally allows us to achieve state-of-the-art results with computational costs comparable to other diffusion-based solutions. Moreover, with simple tricks such as training with reduced float precision (JDCL fast) our method achieves performance significantly higher than other approaches, while also having the lowest computational cost. In Table 5, we present a runtime analysis to compare the training times of JDCL with diffusion-based baselines. All methods were trained on the same machine with a single NVIDIA A100 GPU.

6. Conclusions

In this work, we introduce JDCL – a new continual learning method that employs joint-diffusion modeling for generative replay. We propose a shared parametrization that mitigates the problem of knowledge transfer in generative replay. In our experimental section, we show that JDCL outperforms recent generative replay techniques in standard class incremental scenarios. Moreover, we propose the adaptation of our method to semi-supervised learning where we are able to outperform buffer-based approaches. Finally, we show in additional experiments that the performance of our method should be attributed to its ability to learn and maintain useful representations.

Acknowledgments

This work is supported by National Centre of Science (NCP, Poland) Grants No. 2022/45/B/ST6/02817, and 2023/51/B/ST6/03004. We acknowledge PLGrid for providing computer facilities under grants no. PLG/2025/018424 and PLG/2025/018551.

References

[1] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online Continual Learning with Maximally Interfered Retrieval. In *Advances in Neural Information Processing Systems*, 2019. 1, 2

[2] Dmitry Baranchuk, Ivan Rubachev, A. Voynov, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. *International Conference on Learning Representations*, 2021. 3

[3] Samyadeep Basu, Nanxuan Zhao, Vlad Morariu, Soheil Feizi, and Varun Manjunatha. Localizing and editing knowledge in text-to-image generative models. *arXiv preprint arXiv: 2310.13730*, 2023. 3

[4] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. 3

[5] Matteo Boschini, Pietro Buzzega, Lorenzo Bonicelli, Angelo Porrello, and Simone Calderara. Continual semi-supervised learning through contrastive interpolation consistency. *Pattern Recognition Letters*, 162:9–14, 2022. 3

[6] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020. 3

[7] Bartosz Cywiński, Kamil Deja, Tomasz Trzciński, Bartłomiej Twardowski, and Łukasz Kuciński. Guide: Guidance-based incremental learning with diffusion models. *arXiv preprint arXiv: 2403.03938*, 2024. 2, 3, 6

[8] Kamil Deja, Paweł Wawrzyński, Daniel Marczak, Wojciech Masarczyk, and Tomasz Trzciński. Binplay: A binary latent autoencoder for generative replay continual learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021. 1

[9] Kamil Deja, Paweł Wawrzyński, Wojciech Masarczyk, Daniel Marczak, and Tomasz Trzciński. Multiband vae: Latent space alignment for knowledge consolidation in continual learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2902–2908. International Joint Conferences on Artificial Intelligence Organization, 2022. Main Track. 3, 4

[10] Kamil Deja, Tomasz Trzcinski, and Jakub M Tomczak. Learning data representations with joint diffusion models. *arXiv preprint arXiv:2301.13622*, 2023. 3

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[12] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021. 3, 6

[13] M Ehsan Abbasnejad, Anthony Dick, and Anton van den Hengel. Infinite variational autoencoder for semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5888–5897, 2017. 3

[14] Enrico Fini, Victor G Turrisi Da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2022. 8

[15] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 1999. 1, 2

[16] Zhe Gan, Liqun Chen, Weiyao Wang, Yuchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. Triangle generative adversarial networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 3

[17] Rui Gao and Weiwei Liu. Ddgr: continual learning with deep diffusion-based generative replay. In *International Conference on Machine Learning*, pages 10744–10763. PMLR, 2023. 2, 3, 8

[18] Alex Gomez-Villa, Bartlomiej Twardowski, Kai Wang, and Joost Van de Weijer. Plasticity-optimized complementary networks for unsupervised continual learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1690–1700, 2024. 4

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 2

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 3

[21] J. E. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 2021. 3

[22] Paul Kuo-Ming Huang, Si-An Chen, and Hsuan-Tien Lin. Improving conditional score-based generation with calibrated classification and joint training. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022. 4

[23] Nitin Kamra, Umang Gupta, and Yan Liu. Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*, 2017. 3

[24] Zhiqi Kang, Enrico Fini, Moin Nabi, Elisa Ricci, and Karteek Alahari. A soft nearest-neighbor framework for continual semi-supervised learning. *ICCV*, 2023. 3, 7

[25] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in Neural Information Processing Systems*, 27, 2014. 3

[26] 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. 2

[27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[28] Alex Kurakin, Colin Raffel, David Berthelot, Ekin Dogus Cubuk, Han Zhang, Kihyuk Sohn, and Nicholas Carlini. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *ICLR*, 2020. 3

[29] Mingi Kwon, Jaeseok Jeong, and Youngjung Uh. Diffusion models already have a semantic latent space. *arXiv preprint arXiv:2210.10960*, 2022. 3

[30] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning, 2017. 3

[31] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, page 896. Atlanta, 2013. 3

[32] Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative Models from the perspective of Continual Learning. In *IJCNN*, 2019. 1, 2

[33] Yang Li, Quan Pan, Suhang Wang, Haiyun Peng, Tao Yang, and Erik Cambria. Disentangled variational auto-encoder for semi-supervised learning. *Information Sciences*, 482:73–85, 2019. 3

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

[35] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 226–227, 2020. 2

[36] Yi Liu, Guangchang Deng, Xiangping Zeng, Si Wu, Zhiwen Yu, and Hau-San Wong. Regularizing discriminative capability of cgans for semi-supervised generative learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[37] Grace Luo, Lisa Dunlap, Dong Huk Park, Aleksander Holynski, and Trevor Darrell. Diffusion hyperfeatures: Searching through time and space for semantic correspondence. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[38] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International conference on machine learning*, pages 1445–1453. PMLR, 2016. 3

[39] Daniel Marczak et al. Revisiting supervision for continual representation learning. In *ECCV*, 2024. 8

[40] Wojciech Masarczyk, Kamil Deja, and Tomasz Trzcinski. On robustness of generative representations against catastrophic forgetting. In *Neural Information Processing*, pages 325–333, Cham, 2021. Springer International Publishing. 2

[41] Takeru Miyato, Shin-Ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2019. 3

[42] Martin Mundt, Sagnik Majumder, Iuliia Pliushch, Yong Won Hong, and Visvanathan Ramesh. Unified Probabilistic Deep Continual Learning through Generative Replay and Open Set Recognition, 2020. 2

[43] Martin Mundt, Yongwon Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *Neural Networks*, 2023. 2

[44] Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah Goodman, Pushmeet Kohli, Frank Wood, Philip Torr, et al. Learning disentangled representations with semi-supervised deep generative models. *Advances in neural information processing systems*, 30, 2017. 3

[45] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019. 2

[46] Yong-Hyun Park, Mingi Kwon, Jaewoong Choi, Junghyo Jo, and Youngjung Uh. Understanding the latent space of diffusion models through the lens of riemannian geometry. *Advances in Neural Information Processing Systems*, 36: 24129–24142, 2023. 3

[47] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11557–11568, 2021. 3

[48] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience Replay for Continual Learning. In *Advances in Neural Information Processing Systems*, 2019. 1, 2

[49] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015. 1

[50] Margherita Rosnati, Mélanie Roschewitz, and Ben Glocker. Robust semi-supervised segmentation with timestep ensembling diffusion models. In *Machine Learning for Health (ML4H)*, pages 512–527. PMLR, 2023. 3

[51] Mohammad Rostami, Soheil Kolouri, and Praveen K Pilly. Complementary learning for overcoming catastrophic forgetting using experience replay. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3339–3345, 2019. 2

[52] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023. 3

[53] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks, 2016. arXiv:1606.04671. 2

[54] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2016. 3

[55] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in Neural Information Processing Systems*, 29, 2016. 3

[56] Vikash Sehwag, C. Hazirbas, Albert Gordo, Firat Ozgenel, and Cristian Canton Ferrer. Generating high fidelity data from low-density regions using diffusion models. *Computer Vision And Pattern Recognition*, 2022. 1

[57] Viraj Shah, Nataniel Ruiz, Forrester Cole, Erika Lu, Svetlana Lazebnik, Yuanzhen Li, and Varun Jampani. Ziplora: Any subject in any style by effectively merging loras. *arXiv preprint arXiv:2311.13600*, 2023. 3

[58] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. In *Advances in Neural Information Processing Systems*, 2017. 1, 2

[59] James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. Continual diffusion: Continual customization of text-to-image diffusion with c-lora. *arXiv preprint arXiv:2304.06027*, 2023. 3

[60] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 3

[61] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems*, pages 596–608. Curran Associates, Inc., 2020. 3, 5

[62] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015. 3

[63] Filip Szatkowski, Mateusz Pyla, Marcin Przewięźlikowski, Sebastian Cygert, Bartłomiej Twardowski, and Tomasz Trzciński. Adapt your teacher: Improving knowledge distillation for exemplar-free continual learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1977–1987, 2024. 6

[64] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 3

[65] Anh Thai, Stefan Stojanov, Isaac Rehg, and James M Rehg. Does continual learning= catastrophic forgetting. *arXiv preprint arXiv:2101.07295*, 2021. 2

[66] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1921–1930, 2023. 3

[67] Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning, 2018. arXiv:1809.10635. 2

[68] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *NeurIPS - Continual Learning workshop*, 2019. 3

[69] Gido M Van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):4069, 2020. 1, 2

[70] Liyuan Wang, Bo Lei, Qian Li, Hang Su, Jun Zhu, and Yi Zhong. Triple-memory networks: A brain-inspired method for continual learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5):1925–1934, 2021. 3

[71] Liyuan Wang, Kuo Yang, Chongxuan Li, Lanqing Hong, Zhenguo Li, and Jun Zhu. Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5383–5392, 2021. 3

[72] Xiao Wang, Daisuke Kihara, Jiebo Luo, and Guo-Jun Qi. Enaet: A self-trained framework for semi-supervised and supervised learning with ensemble transformations. *IEEE Transactions on Image Processing*, 30:1639–1647, 2020. 3

[73] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay gans: Learning to generate new categories without forgetting. In *Advances in Neural Information Processing Systems*, 2018. 1

[74] Si Wu, Guangchang Deng, Jichang Li, Rui Li, Zhiwen Yu, and Hau-San Wong. Enhancing triplegan for semi-supervised conditional instance synthesis and classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[75] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698, 2020. 3

[76] Ju Xu and Zhanxing Zhu. Reinforced Continual Learning. In *Advances in Neural Information Processing Systems*, 2018. 2

[77] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A survey on deep semi-supervised learning. pages 8934–8954, 2023. 3

[78] Michał Zajac, Kamil Deja, Anna Kuzina, Jakub M. Tomczak, Tomasz Trzciński, Florian Shkurti, and Piotr Miłoś. Exploring continual learning of diffusion models. *arXiv preprint arXiv: Arxiv-2303.15342*, 2023. 3

[79] Bowen Zhang, Yidong Wang, Wenxin Hou, HAO WU, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. 34:18408–18419, 2021. 3