HETEROGENEOUS CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel framework and a solution to tackle the continual learning (CL) problem with *progressive evolution of neural networks*. Most CL methods focus on adapting a single network to a new task/class by modifying its weights. However, with rapid progress in architecture design, the problem of adapting existing solutions to novel architectures becomes relevant. For the first time, we propose *Heterogeneous Continual Learning (HCL)* to address this problem, where a wide range of evolving network architectures emerge continually together with novel data/tasks. As a solution, we build on top of the distillation family of techniques and modify it to a new setting where a weaker model takes the role of a teacher; meanwhile, a new stronger architecture acts as a student. Furthermore, we consider a setup of limited access to previous data and propose *Quick Deep Inversion (QDI)* to recover prior task visual features to support knowledge transfer. QDI significantly reduces computational costs compared to previous solutions and improves overall performance. In summary, we propose a new setup for CL with a modified knowledge distillation paradigm and design a quick data inversion method to enhance distillation. Our evaluation of various benchmarks shows that the proposed method can successfully progress over various networks while outperforming state-of-the-art methods with a $2\times$ improvement on accuracy.

1 INTRODUCTION

Over the past decade, we have seen a plethora of innovations in deep neural networks (DNNs) that have led to remarkable improvement in performance on several applications (Ramesh et al., 2021; Mehta et al., 2021; McGrath et al., 2021). AlexNet (Krizhevsky et al., 2012) was the first breakthrough showing the potential of deep learning on the ImageNet benchmark (Deng et al., 2009), it was followed by various architectural advances such as VGG (Simonyan & Zisserman, 2015), Inception (Szegedy et al., 2017), ResNet and its variants (He et al., 2016; Zagoruyko & Komodakis, 2016; Xie et al., 2017), efficient architectures (Howard et al., 2017; Tan & Le, 2019), and applications of Transformer (Vaswani et al., 2017) in computer vision (Dosovitskiy et al., 2021; Liu et al., 2021b); however, all these architectures are trained from scratch and compared on ImageNet classification (Deng et al., 2009). In real-world scenarios, the dataset size is not constant, yet storage of billions of data instances and retraining the model representations from scratch is computationally expensive and infeasible. Further, access to old datasets is often not available due to data privacy. Thus, it is crucial to transfer the knowledge learned by the previous model representations to the recent state-of-the-advances without forgetting knowledge and accessing prior datasets.

Continual learning (CL) (Thrun, 1996) is a common way to train representations on a data stream without forgetting the learned knowledge. It is inspired by the learning process of humans who can continuously learn from orthogonal experiences without mutual interference (Flesch et al., 2018). This is likely possible because the human brain is continuously changing (especially structure of the representation) during the learning process (Council, 2000; Flesch et al., 2018). However, all prior CL techniques (Zenke et al., 2017; Ahn et al., 2019; Rusu et al., 2016; Rebuffi et al., 2017; Rolnick et al., 2019; Buzzega et al., 2020; Madaan et al., 2022) assume the presence of a fixed representation structure (see Figure 1) – a fully-connected network with two hidden layers for MNIST (LeCun, 1998) and standard ResNet-18 (He et al., 2016) for all the other datasets. While recent works investigate the effect of depth and width (Mirzadeh et al., 2022b) on continual learning, to our knowledge, no work has focused on the real problem of learning on sequentially arriving tasks with changing network architectures. Moreover, most works assume the weight transfer with the same architecture, in case of the previous model being available only as black box, these methods are not applicable.

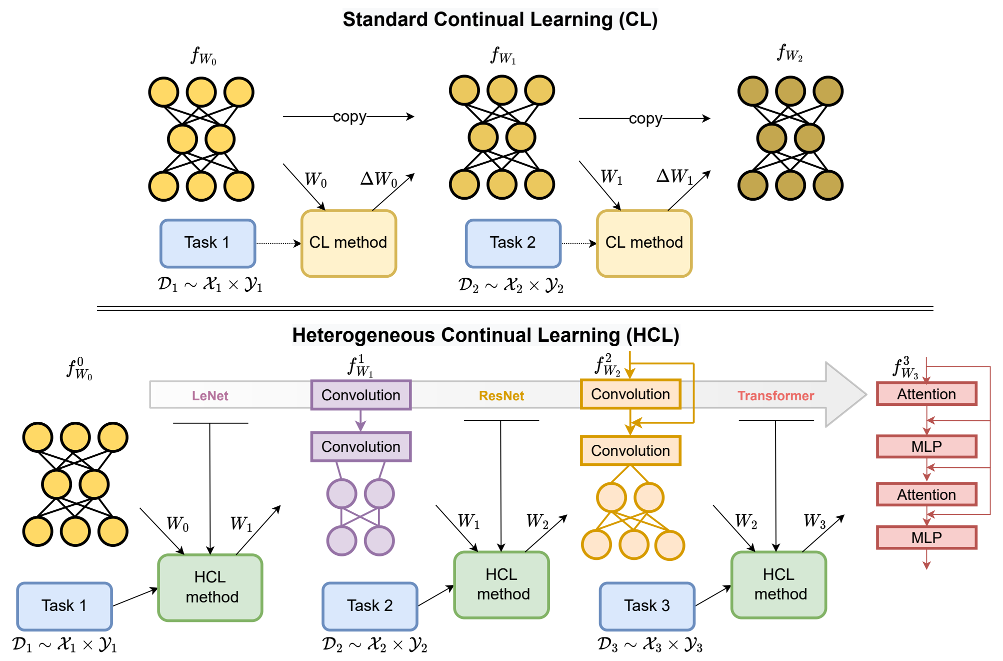
Standard Continual Learning (CL)
f_{W_0}
f_{W_1}
f_{W_2}
copy
copy
W_0
ΔW_0
W_1
ΔW_1
Task 1
CL method
Task 2
CL method
$\mathcal{D}_1 \sim \mathcal{X}_1 \times \mathcal{Y}_1$
$\mathcal{D}_2 \sim \mathcal{X}_2 \times \mathcal{Y}_2$
Heterogeneous Continual Learning (HCL)
$f^0_{W_0}$
$f^1_{W_1}$
$f^2_{W_2}$
$f^3_{W_3}$
LeNet
Convolution
ResNet
Convolution
Transformer
Attention
Convolution
Convolution
MLP
Attention
MLP
W_0
W_1
W_1
W_2
W_2
W_3
HCL method
HCL method
HCL method
Task 1
Task 2
Task 3
$\mathcal{D}_1 \sim \mathcal{X}_1 \times \mathcal{Y}_1$
$\mathcal{D}_2 \sim \mathcal{X}_2 \times \mathcal{Y}_2$
$\mathcal{D}_3 \sim \mathcal{X}_3 \times \mathcal{Y}_3$

Figure 1: **Illustration of standard and heterogenous continual learning.** Standard CL updates the same representational structure while preserving previous information with incoming tasks. On the contrary, the goal of HCL is to continuously evolve the network architecture while maintaining the accuracy of new data instances.

To address these limitations, we propose a novel CL framework called *Heterogeneous Continual Learning (HCL)* to learn continual representations on a data stream without any restrictions on the training configuration or network architecture of the representations (see Figure 1). In particular, we consider the continual learner as a stream of learners, where we train each learner with the same or different backbone architecture. To this end, we consider a diverse range of heterogeneous architectures including LeNet (LeCun et al., 1998), ResNet and its variants (He et al., 2016; Zagoruyko & Komodakis, 2016; Xie et al., 2017), and recent architectures such as RegNet (Xu et al., 2022) and vision transformers (Liu et al., 2021b). However, due to the assumption of the same architectural structure across different tasks, conventional regularization (Li & Hoiem, 2016; Zenke et al., 2017; Schwarz et al., 2018; Ahn et al., 2019) and architectural methods (Rusu et al., 2016; Yoon et al., 2018; Li et al., 2019) are not directly applicable in our HCL setting.

To continually learn heterogeneous representations, we revisit knowledge distillation (Hinton et al., 2015) – a method to transfer the knowledge learned across different networks. While knowledge distillation has been formulated for continual learning (Li & Hoiem, 2016; Rebuffi et al., 2017; Buzzega et al., 2020), prior methods do not outperform the experience replay based methods (Rebuffi et al., 2017; Rolnick et al., 2019; Aljundi et al., 2019; Buzzega et al., 2020). Moreover, the distillation pipeline has not been considered for network evolution. This work shows that knowledge distillation can outperform state-of-the-art methods with and without replay examples in standard and heterogeneous continual learning settings with proper modifications. Our solution comes at the cost of storing and reusing only the most recent model.

The continual learner might not have access to the prior data. Therefore, motivated by DeepInversion (DI) (Yin et al., 2020), we consider *data-free continual learning*. In particular, DI optimizes random noise to generate class-conditioned samples; however, it requires many steps to optimize a single batch of instances. This increases its computational cost and slows CL. In response to this, we propose *Quick Deep Inversion (QDI)*, which utilizes current task examples as the starting point for synthetic examples and optimizes them to minimize the prediction error on the previous tasks. Specifically, this leads to an interpolation between the current task and previous task instances, which promotes current task adaptation while minimizing catastrophic forgetting. We compare our proposed method on various CL benchmarks against state-of-the-art methods, where it outperforms them across all settings with a wide variety of architectures.

In summary, the contributions of our work are:

- We propose a novel continual learning framework called *Heterogeneous Continual Learning (HCL)* to learn a stream of different architectures on a sequence of tasks while transferring the knowledge from past representations.
- We revisit knowledge distillation and propose *Quick Deep Inversion (QDI)*, which inverts the previous task parameters while interpolating the current task examples with minimal additional cost.
- We benchmark existing state-of-the-art solutions in the new settings and outperform them with our proposed method across a diverse stream of architectures for both class-incremental and task-incremental continual learning.

2 Related work

Continual learning approaches can be categorized into three categories: *Replay-based approaches* formulate different criterion (Rebuffi et al., 2017; Rolnick et al., 2019; Aljundi et al., 2019; Buzzega et al., 2020) to select a representative subset that is revisited in future tasks. While most approaches (Rebuffi et al., 2017; Rolnick et al., 2019; Aljundi et al., 2019) store the input examples, Buzzega et al. (2020) retain the logits to preserve the knowledge from the past. However, all these methods are limited in practical scenarios due to privacy concerns and excessive memory requirements to store the data from previous tasks. *Architectural methods* consider the inclusion of task-dependent representations to prevent catastrophic forgetting during continual learning (Rusu et al., 2016; Yoon et al., 2018; Li et al., 2019; Dai et al., 2020; Yoon et al., 2020). Rusu et al. (2016) adds task-dependent sub-networks and Yoon et al. (2020) decomposes each layer parameters to task-specific and task-shared parameters. *Regularization methods* penalize the change in the representations by adding a regularization term in the loss (Li & Hoiem, 2016; Zenke et al., 2017; Schwarz et al., 2018; Ahn et al., 2019). Li & Hoiem (2016) minimizes the change in the output representations, and Zenke et al. (2017) estimates the important parameters and restricts their change during training.

Prior works search for the most optimal architecture for continual learning. Lee et al. (2021) formulated a data-driven layer-based transfer function to learn the layers that should be transferred to mitigate forgetting. Mirzadeh et al. (2022a) showed that increasing the width mitigates forgetting, whereas increasing the depth negatively affects forgetting, and Mirzadeh et al. (2022c) investigated the effect of various components in an architecture. Carta et al. (2022) proposed Ex-Model Continual Learning to learn the continual learner from a stream of trained models; however, their proposed scheme is not directly applicable to the HCL setting as the architecture of their learner is fixed and it requires an extra buffer to store the past expert and CL representations.

In contrast, we incorporate the recent advances in deep learning to train the continual learner without any knowledge of prior data or its architecture. Our approach does not strictly require a buffer to store data or representation, and we also do not assume that the learners' architecture is fixed. To the best of our knowledge, this is the first work that explores continual learning with different representational structures while adapting to the current task and minimizing catastrophic forgetting.

3 Heterogeneous Continual Learning

3.1 Motivation

In real-world scenarios, the data distribution continuously evolves with time; thus, adapting the model to the distribution shifts is essential while preserving past knowledge. Existing methods in the CL literature (Zenke et al., 2017; Ahn et al., 2019; Rusu et al., 2016; Rebuffi et al., 2017; Rolnick et al., 2019; Buzzega et al., 2020) either optimize the performance on a single architecture (He et al., 2016) or find an optimal architecture for continual learning (Mirzadeh et al., 2022a;c). Meanwhile, deep learning techniques are improving rapidly, providing better models with higher accuracy. Therefore, it is important to be able to deploy state-of-the-art deep learning architectures and methods while preserving the knowledge from previously learned representations. Our objective is to train the continual learner with the current state-of-the-art deep-learning approaches while continuously updating its structure and training it on the incoming stream of data.

To motivate further, we provide a simple experiment on CIFAR-10 split into two tasks demonstrating the benefit of adapting to novel neural architectures in Figure 2. With the new ability to switch the network architecture, (*e.g.,* from LeNet to ResNet) the performance surpasses standard CL setup that is limited by a rigid but weak architecture (HCL vs. CL LeNet). More appealingly, it even approximates the stronger model deployed throughout the CL process (HCL vs. CL ResNet), while alleviating the requirement of replay buffer to store prior data experiences (HCL is data-free vs. CL with buffer), as we will show next in methodology and experiment sections.

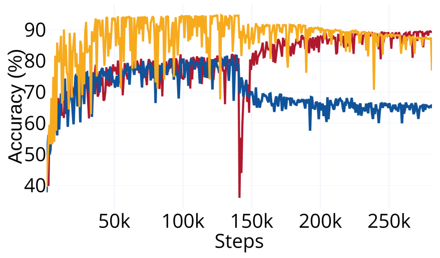

Figure 2: Average accuracy comparison between standard CL – ResNet18 and LeNet trained with buffer, ours with adaptive architectures without buffer on CIFAR-10 dataset with two tasks containing five classes each.

3.2 Problem setup

The problem of standard continual learning considers the learning objective where the continual learner $f_{W_t} : \mathcal{X}_t \to \mathcal{Y}_t$ learns a sequence of T tasks by updating the **fixed representation structure** for each task. Each task $t \in T$ contains training data $\mathcal{D}_t = \{x_i, y_i\}_{i=1}^{N_t} \sim \mathcal{X}_t \times \mathcal{Y}_t$, which is composed of N_t identically and independently distributed examples. More formally, we minimize the following objective for standard continual learning:

$$\underset{W_t}{\text{minimize}} \ \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}_t}[\ell\left(f_{W_t}(x_i), y_i\right)], \tag{1}$$

where $\ell : \mathcal{Y}_t \times \mathcal{Y}_t \to \mathbb{R}_{\geq 0}$ is the task-specific loss function. In this work, we consider the continual learner as a **stream of architectures** $\{f^1_{W_1}, \ldots, f^t_{W_t}\}$, where the learner can completely change the backbone architecture to incorporate recent architectural developments (He et al., 2016; Dosovitskiy et al., 2021; Liu et al., 2021b; Tolstikhin et al., 2021) in order to improve model performance. However, when the architectures are different, there is no natural knowledge transfer mechanism, and the network parameters are initialized randomly, which makes this problem challenging compared to standard continual learning.

Particularly, each architectural representation $f^t_{W_t} : \mathcal{X}_t \to \mathcal{Y}_t, \ \forall t \in \{1, \ldots, T\}$ is trained on task distribution $\mathcal{D}_t$ and our objective is to train these stream of networks on a sequence of tasks without forgetting the knowledge of the previous set of tasks. Additionally, when the network structure remains the same, we want to transfer the learned representations sequentially to train incoming tasks. Overall, the learning objective of the HCL is:

$$\underset{W_t}{\text{minimize}} \ \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}_t}[\ell\left(f^t_{W_t}(x_i), y_i\right)], \tag{2}$$

For notation simplicity, we discard W_t in $f^t_{W_t}$ for the rest of the paper. In this work, we focus on the CL setting, where the continual learner does not rely on task identifiers during training and uses constant memory following prior works (Buzzega et al., 2020; De Lange et al., 2021). In particular, we consider task-incremental (task-IL) and class-incremental learning (class-IL) during inference but do not use task labels during training. In task-IL, we provide the task identity to select the classification head, whereas class-IL uses a shared head across all classes. Consequently, class-IL is more challenging due to an equal weight for all the tasks and prone to higher forgetting and lower accuracy than task-IL; however, recent works (Cossu et al., 2022) also raise concerns regarding the practicality of class-IL in real-world scenarios.

3.3 Proposed method

We address the problem of HCL with our method summarized in Figure 3. As in the standard CL, new task data and an optional buffer are used to directly train a new model (stronger student) initialized from scratch with a task loss. Additionally, a knowledge transfer data (KDA) is collected given new task data, (optional) buffer and (optional) synthesised samples by quick deep inversion. The KDA is used to support network transfer via knowledge distillation between old and new models. To describe our method, we first revisit knowledge distillation (Hinton et al., 2015) in Section 3.3.1 and propose a training paradigm that outperforms state-of-the-art methods. Then, we introduce a quick deep inversion in Section 3.3.2 to recover past visual features promptly. It inverts the previous task model to generate synthetic examples with minimal cost.

3.3.1 Revisiting knowledge distillation

Due to a difference in model representations and the absence of the training data used to train previous models, we concentrate on knowledge distillation (KD) (Hinton et al., 2015) for HCL. However, our goal is orthogonal to standard KD; instead of distilling the knowledge from a large model (or ensemble) to a smaller (weaker) model for efficient deployment we distill from the latter to the former. Our motivation is to improve model performance by transferring to a state-of-the-art model.

While KD has been adopted in CL (Li & Hoiem, 2016; Rebuffi et al., 2017; Hou et al., 2019; Dhar et al., 2019; Buzzega et al., 2020), it has been (i) limited to homogeneous architectures across different tasks, and (ii) considered a weak baseline in comparison to replay-based methods. Contrarily, we tackle the HCL setting and propose a modified paradigm inspired by the recent advances in KD to improve CL performance and lift both limitations.

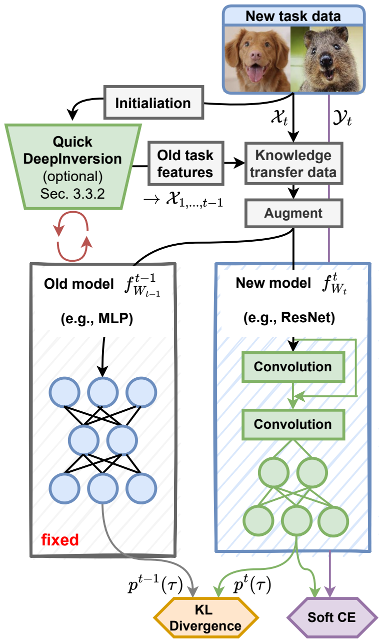

Figure 3: Proposed method to perform continual learning with neural network architecture change.

First, we incorporate label-smoothing with low temperature for training all the tasks to improve the efficiency of distillation (Chandrasegaran et al., 2022). Second, we leverage the recent use of data augmentation in KD (Beyer et al., 2022) for CL, where we use the same set of augmentations on the input presented to the previous and to the current task models instead of precomputing and storing those for the former. We remark that in contrast to standard knowledge distillation, we use the current task instances for distillation, as the previous task data is often not available in real-world applications due to data privacy and legal restrictions. Third, we do not use task identity and focus on task-free continual learning during training. In addition, we do not use warmup before the current task as we find that it is not essential to improve model performance. We conduct ablation experiments to justify the design choices of every component in Table 3. Our overall objective for the lifelong learner is:

$$\underset{W_t}{\text{minimize}} \;\; \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}_t}[\ell\left(f^t(x_i), y_i^t(\psi)\right)] + \alpha \cdot \text{KL}\left(p_i^t(\tau), p_i^{t-1}(\tau)\right), \tag{3}$$

where $y_i^t(\psi) = y(1-\psi) + \psi/C$, ψ denotes the mixture parameter to interpolate the hard targets to uniform distribution defined using ψ, C is the number of classes, $p^t(\tau)$ and $p^{t-1}(\tau)$ denote the temperature-scaled output probabilities for current task model and past-task model, respectively, with τ as the corresponding temperature. α is the hyper-parameter that controls the strength of KD loss.

3.3.2 Enhancing data efficiency

To further improve our data-free HCL paradigm, we extend DeepInversion (DI) (Yin et al., 2020), which optimizes images $\tilde{x}$ initialized with Gaussian noise to improve knowledge transfer from the prior tasks given only the trained models and no data. The objective of the optimization is to excite particular features, or, as in our work, classes from previous tasks. The optimization is guided by proxy image priors: (i) total variation $\mathcal{L}_{tv}$, (ii) $\mathcal{L}_{\ell_2}$ of the generated samples and (iii) feature distribution regularization $\mathcal{L}_{\text{feature}}$. DI requires many steps to generate examples prior to every task, which increases the computational cost of CL method and, therefore, becomes less appealing.

To tackle this limitation, we propose *Quick Deep Inversion (QDI)* that initializes the synthetic examples with the current task data prior to the optimization. There is no reason to believe that randomly sampled Gaussian noise is an optimal starting point for generating past task examples. We optimize the current task examples such that features will fall to the manifold learned by the previous model and the domain shift is minimized. As a result such imags are classified as past task classes. QDI generates examples $\tilde{x}_{\text{prior}}\{f^{t-1}, k\}$ to approximate features from all prior tasks $\{1,\dots,t-1\}$ by inverting the last model f^{t-1} with k optimization steps. Formally,

$$\tilde{x}_{\text{prior}}\{f^{t-1}, k\} = \underset{\tilde{x}}{\text{argmin}} \;\; \mathcal{L}(f^{t-1}(\tilde{x}), \tilde{y}) + \alpha_{tv}\mathcal{L}_{tv}(\tilde{x}) + \alpha_{\ell_2}\mathcal{L}_{\ell_2}(\tilde{x}) + \alpha_{\text{feature}}\mathcal{L}_{\text{feature}}, \tag{4}$$

where the synthesized examples are optimized towards prior classes $\tilde{y} \sim \mathcal{Y}_{\{1,...,t-1\}}$ to minimize forgetting. $\alpha_{tv}, \alpha_{\ell_2}, \alpha_{\text{feature}}$ denote the hyper-parameters that determine the strength of individual losses. To improve synthesis speed, we initialize $\tilde{x}$ with the current task input image x_t, which provides a $4\times$ speed-up and leads to more natural images:

$$\tilde{x}_{\text{prior}}\{f^{t-1}, 0\} = \tilde{x}_{\text{prior},\, k=0} = x_t.$$

The initialized $\tilde{x}_{\text{prior}}$ can then optimized using Equation 4, regularized for realism by the target model f^{t-1}, hence quickly unveiling previous task visual features on top of the current task image through

$$\mathcal{L}_{\text{feature}} = \sum_{l \in L} \Big[d\Big(\mu_l(\tilde{x}_{\text{prior}}), \mathbb{E}[\mu_l(x)|x \sim \mathcal{X}_{\{1,...,t-1\}}]\Big) + d\Big(\sigma_l(\tilde{x}_{\text{prior}}), \mathbb{E}[\sigma_l(x)|x \sim \mathcal{X}_{\{1,...,t-1\}}]\Big)\Big].$$

$d(\cdot,\cdot)$ denotes the distance metric for feature regularization. In this paper, we use the mean-squared distance. Akin to Yin et al. (2020), we assume Gaussian distribution for feature maps and focus on batch-wise mean $\mu_l(x)$ and variance $\sigma_l(x)$ for the layer l. We note that these statistics are implicitly captured through the batch normalization in f^{t-1} without storing input data for all previous tasks $\{1, ..., t-1\}$: $\mathbb{E}\big[\mu_l(x)|x \sim \mathcal{X}_{\{1,...,t-1\}}\big] \simeq \text{BN}_l(\text{running mean})$, $\mathbb{E}\big[\sigma_l(x)|x \sim \mathcal{X}_{\{1,...,t-1\}}\big] \simeq \text{BN}_l(\text{running var})$. Otherwise, we approximate them using values calculated with post-convolution feature maps given a current task batch to the target model, f^{t-1}, leveraging its feature extraction capability for previous tasks. This efficient QDI allows us to use it for continual learning with minimal additional cost to existing methods.

In case of QDI, the learning objective in Equation 3 can be updated as:

$$\underset{W_t}{\text{minimize}}\ \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}_t}[\ell\left(f^t(x_i), y_i^t(\psi)\right)] + \alpha \cdot \text{KL}\left(p_i^t(\tau), p_i^{t-1}(\tau)\right) + \beta \cdot \text{KL}\left(\tilde{p}_i^t(\tau), \tilde{p}_i^{t-1}(\tau)\right), \quad (5)$$

where $\tilde{p}^t(\tau)$ and $\tilde{p}^{t-1}(\tau)$ are the output probabilities of the generated examples scaled with temperature τ using the current and past task-models, β is the hyper-parameter to control the strength of QDI knowledge distillation loss. QDI utilizes the generated examples to enhance knowledge distillation with a four times speedup to prior data-inversion methods (Yin et al., 2020) due to the current task being a better prior than pixel-wise Gaussian to learn the generated data.

4 Experiments

4.1 Experimental setup

Datasets and architectures. We validate the proposed method on Split CIFAR-10 (Krizhevsky, 2012) with five tasks, Split CIFAR-100 (Krizhevsky, 2012) with 20 tasks and Split Tiny-ImageNet with ten tasks. For HCL, we consider various architectural innovations in the ImageNet challenge (Deng et al., 2009) including LeNet (LeCun et al., 1989), AlexNet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2015) with batch normalization (Ioffe & Szegedy, 2015), InceptionNet (Szegedy et al., 2017), ResNet (He et al., 2016), ResNeXt (Xie et al., 2017), DenseNet (Huang et al., 2017), SENet (Hu et al., 2018), RegNet (Xu et al., 2022). We defer the details to Appendix A.1.1.

Baselines. We compare our designed KD paradigm (Equation 3), with QDI (Equation 5) agains standard fine-tuning (FINETUNE), SI Zenke et al. (2017), LwF (Li & Hoiem, 2016) ICARL (Rebuffi et al., 2017) A-GEM (Chaudhry et al., 2019a), ER (Chaudhry et al., 2019b), DER and DER++ (Buzzega et al., 2020), DI (Yin et al., 2020) and multi task training (MULTITASK), where all tasks are trained jointly with ResNet18 – most prominent architecture in CL literature. For a fair comparison, we extend our method with the classification loss using BUFFER in Equation 3. The buffer-size for replay-based method is fixed to 200 for all the methods that use exemplar or generative replay.

Evaluation. To evaluate and compare our method in both standard CL and HCL setting, we use the average accuracy and forgetting as two metrics.

1. **Average accuracy.** It is the average of the accuracy of all tasks after the completion of training with the sequence of T tasks. More formally, average accuracy $\mathcal{A}_T = \frac{1}{T}\sum_{t=1}^{T} a_{T,t}$, where $a_{i,j}$ is the accuracy of task j after completion of task i.
2. **Average forgetting.** It is the averaged difference in accuracy between the accuracy after training and the maximum accuracy for each task. More formally, average forgetting $\mathcal{F}_T = \frac{1}{T-1}\sum_{t=1}^{T} \max_{i\in\{1,...,T\}} (a_{i,t} - a_{T,t})$

Table 1: **Accuracy and forgetting** with task-IL on standard CL and HCL. The best results are highlighted in **bold**. $\mathcal{B}$ denotes replay-buffer, $\mathcal{A}_T, \mathcal{F}_T$ denote average accuracy and forgetting after the completion of training.* denotes the methods whose numbers were used from Buzzega et al. (2020) and – indicates the unaviliability of results. All other experiments are over three independent runs.

METHOD	$\mathcal{B}$	SPLIT CIFAR-10		SPLIT CIFAR-100		SPLIT TINY-IMAGENET	
		$\mathcal{A}_T$ (↑)	$\mathcal{F}_T$ (↓)	$\mathcal{A}_T$ (↑)	$\mathcal{F}_T$ (↓)	$\mathcal{A}_T$ (↑)	$\mathcal{F}_T$ (↓)
		STANDARD CONTINUAL LEARNING					
FINETUNE	–	62.93 (± 1.41)	41.45 (± 2.39)	40.13 (± 3.60)	53.64 (± 3.65)	19.31 (± 0.71)	59.50 (± 0.81)
SI (Zenke et al., 2017)*	–	68.05 (± 5.91)	38.76 (± 0.89)	–	–	36.32 (± 0.13)	–
LWF (Li & Hoiem, 2016)*	–	63.29 (± 2.35)	32.56 (± 0.56)	–	–	15.85 (± 0.58)	–
DI (Yin et al., 2020)	–	89.11 (± 3.70)	7.12 (± 2.43)	49.64 (± 0.59)	35.29 (± 0.75)	44.29 (± 1.88)	22.42 (± 3.23)
KD (OURS)	–	95.01 (± 0.79)	0.87 (± 0.35)	87.13 (± 0.45)	4.26 (± 0.41)	65.48 (± 0.12)	7.33 (± 2.03)
KD W/ QDI (OURS)	–	**95.46 (± 0.15)**	**0.37 (± 0.09)**	**88.30 (± 0.17)**	**2.00 (± 0.15)**	**66.79 (± 0.45)**	**7.28 (± 0.49)**
ICARL (Rebuffi et al., 2017)*	✓	88.99 (± 2.13)	2.63 (± 3.48)	–	–	28.19 (± 1.47)	–
A-GEM (Chaudhry et al., 2019a)	✓	87.09 (± 0.84)	12.54 (± 1.23)	63.88 (± 1.23)	29.98 (± 1.40)	22.44 (± 0.24)	55.43 (± 0.71)
ER (Rolnick et al., 2019)	✓	91.39 (± 0.41)	5.99 (± 0.43)	67.06 (± 1.01)	25.31 (± 1.27)	21.03 (± 2.41)	57.88 (± 2.63)
DER (Buzzega et al., 2020)	✓	92.45 (± 0.26)	5.79 (± 0.20)	67.74 (± 0.88)	24.77 (± 0.94)	30.97 (± 0.14)	45.25 (± 0.48)
DER++ (Buzzega et al., 2020)	✓	92.16 (± 0.68)	5.96 (± 0.74)	69.03 (± 0.74)	23.06 (± 0.69)	33.63 (± 0.10)	40.66 (± 0.83)
KD W/ BUFFER (OURS)	✓	**95.70 (± 0.25)**	**0.52 (± 0.18)**	**80.35 (± 0.53)**	**10.76 (± 0.65)**	**64.75 (± 0.29)**	**8.96 (± 0.81)**
MULTITASK*	✓	98.31 (± 0.12)	N/A	93.89 (± 0.78)	N/A	82.04 (± 0.10)	N/A
		HETEROGENEOUS CONTINUAL LEARNING					
FINETUNE	–	61.99 (± 1.98)	42.15 (± 2.94)	27.08 (± 1.20)	64.54 (± 1.45)	14.83 (± 0.53)	56.31 (± 0.60)
DI (Yin et al., 2020)	–	81.16 (± 2.77)	15.84 (± 3.01)	38.33 (± 1.44)	47.08 (± 1.37)	31.69 (± 1.99)	32.42 (± 3.63)
KD (OURS)	–	92.88 (± 0.61)	3.97 (± 0.72)	68.97 (± 0.72)	21.45 (± 0.91)	52.86 (± 2.78)	11.73 (± 2.59)
KD W/ QDI (OURS)	–	**93.24 (± 0.09)**	**3.09 (± 0.19)**	**76.44 (± 1.36)**	**12.31 (± 0.99)**	**53.39 (± 1.67)**	**13.66 (± 1.18)**
A-GEM (Chaudhry et al., 2019a)	✓	76.70 (± 0.67)	24.31 (± 1.03)	35.37 (± 1.29)	56.41 (± 2.19)	15.79 (± 0.60)	54.10 (± 1.00)
ER (Rolnick et al., 2019)	✓	81.78 (± 2.15)	17.57 (± 2.86)	52.59 (± 1.03)	36.41 (± 0.54)	27.82 (± 1.43)	39.66 (± 1.24)
DER (Buzzega et al., 2020)	✓	79.92 (± 0.51)	19.99 (± 0.64)	53.01 (± 0.61)	37.33 (± 0.58)	25.27 (± 1.34)	40.81 (± 0.57)
DER++ (Buzzega et al., 2020)	✓	81.33 (± 0.76)	18.79 (± 0.75)	54.81 (± 1.65)	35.20 (± 1.63)	29.74 (± 1.40)	36.79 (± 2.16)
KD W/ BUFFER (OURS)	✓	**93.95 (± 0.22)**	**2.49 (± 0.44)**	**73.98 (± 0.13)**	**16.40 (± 0.18)**	**54.84 (± 0.87)**	**10.82 (± 0.80)**

4.2 QUANTITATIVE RESULTS

Task-incremental continual learning. First, we compare the performance of our proposed method with other baselines for task-IL on standard and heterogeneous continual learning settings. In Table 1, we find that the performance of all the methods drop significantly by going from the SCL to the HCL setting. Contrary, our proposed methods achieve the lowest drop across all methods; moreover, it obtains the best performance with and without buffer across all considered scenarios. Additionally, we show that in standard CL and HCL, QDI leads to a $\sim 40\%$ and $\sim 22\%$ absolute improvement in average accuracy over state-of-the-art data-free methods for Split CIFAR-100 and Tiny-ImageNet. Interestingly, we find that distillation with synthetic examples generated by QDI obtains comparable or better performance to the replay buffer highlighting the flexibility of data-free continual learning. Moreover, QDI improves synthesis speed by $4\times$ for generating the synthetic examples in comparison to DI (Yin et al., 2020) for each task. All variants of our method achieve the least forgetting in comparison to the baselines across all datasets for both the standard CL and HCL scenarios.

Class-incremental continual learning. Next, we validate our proposed method on class-IL, where task-labels are not provided during training or inference. In Table 2, we observe that all our methods outperform the baselines with and without buffer in average accuracy and forgetting. Specifically, we show that ours with QDI and BUFFER examples show a five to ten percent improvement in average accuracy for Split CIFAR-100 and Tiny-ImageNet datasets. However, we find that buffer is an important component of our method for class-IL; it is consistent to the observations in prior works (Yin et al., 2020; Carta et al., 2022). Further, we observe that the synthetic examples generated by QDI lead to unstable training for CIFAR-10, but improve the performance for other datasets. We believe this is an outcome of the model's dependence on domain over semantics (Smith et al., 2021) and further investigation of data-free continual learning techniques for class-IL would be an interesting direction for future work.

Table 2: **Accuracy and forgetting** with class-IL on standard CL and HCL. The best results are highlighted in **bold**. $\mathcal{B}$ denotes replay-buffer, $\mathcal{A}_T, \mathcal{F}_T$ denote average accuracy and forgetting after the completion of training.* denotes the methods whose numbers were used from Buzzega et al. (2020) and – indicates the unaviliability of results. All other experiments are over three independent runs.

METHOD	$\mathcal{B}$	SPLIT CIFAR-10		SPLIT CIFAR-100		SPLIT TINY-IMAGENET	
		$\mathcal{A}_T$ (↑)	$\mathcal{F}_T$ (↓)	$\mathcal{A}_T$ (↑)	$\mathcal{F}_T$ (↓)	$\mathcal{A}_T$ (↑)	$\mathcal{F}_T$ (↓)
				STANDARD CONTINUAL LEARNING			
FINETUNE	–	19.60 (± 0.03)	96.66 (± 0.12)	6.93 (± 1.13)	60.53 (± 1.11)	6.78 (± 0.14)	40.67 (± 0.92)
DI (Yin et al., 2020)	–	22.72 (± 1.02)	77.65 (± 0.30)	7.21 (± 0.75)	63.90 (± 2.87)	7.28 (± 2.05)	51.70 (± 0.63)
SI (Zenke et al., 2017)*	–	19.48 (± 0.17)	95.78 (± 0.64)	–	–	6.58 (± 0.31)	–
LWF (Li & Hoiem, 2016)*	–	19.61 (± 0.05)	96.69 (± 0.25)	–	–	8.46 (± 0.22)	–
KD (OURS)	–	**27.72** (± **0.52**)	**15.36** (± **1.31**)	16.77 (± 0.63)	80.65 (± 0.99)	20.86 (± 0.14)	42.38 (± 0.95)
KD W/ QDI (OURS)	–	19.75 (± 0.03)	0.08 (± 0.02)	**22.53** (± **1.15**)	**16.50** (± **2.72**)	**24.21** (± **0.52**)	**36.58** (± **1.56**)
ICARL (Rebuffi et al., 2017)*	✓	49.02 (± 3.20)	28.72 (± 0.49)	–	–	7.53 (± 0.79)	–
A-GEM (Chaudhry et al., 2019a)	✓	21.98 (± 0.56)	92.18 (± 1.98)	5.04 (± 0.12)	91.93 (± 0.22)	7.49 (± 0.12)	72.04 (± 0.34)
ER (Rolnick et al., 2019)	✓	48.56 (± 1.40)	58.11 (± 1.61)	9.65(± 0.95)	85.20 (± 1.27)	10.70 (± 0.27)	83.99 (± 0.18)
DER (Buzzega et al., 2020)	✓	66.08 (± 1.18)	27.40 (± 2.16)	19.01 (± 0.74)	65.06 (± 0.15)	10.01 (± 1.52)	65.66 (± 3.60)
DER++ (Buzzega et al., 2020)	✓	**67.23** (± **1.36**)	**26.13** (± **0.28**)	21.38 (± 0.48)	56.17 (± 3.37)	8.23 (± 0.31)	68.51 (± 1.17)
KD W/ BUFFER (OURS)	✓	61.17 (± 3.37)	10.13 (± 2.19)	**25.19** (± **0.15**)	**40.53** (± **0.73**)	**24.22** (± **0.65**)	**36.15** (± **1.41**)
MULTITASK*	–	92.20 (± 0.15)	N/A	70.32 (± 0.48)	N/A	59.99 (± 0.19)	N/A
				HETEROGENEOUS CONTINUAL LEARNING			
FINETUNE	–	21.45 (± 0.75)	91.24 (± 2.33)	5.27 (± 0.23)	72.99 (± 1.94)	7.90 (± 0.13)	57.23 (± 0.11)
DI (Yin et al., 2020)	–	20.67 (± 1.10)	70.97 (± 2.13)	6.20 (± 1.40)	73.70 (± 1.49)	6.39 (± 0.60)	51.14 (± 0.87)
KD (OURS)	–	**30.56** (±**1.61**)	**27.54** (±**1.91**)	12.94 (± 1.13)	58.89 (± 1.64)	14.18 (± 0.35)	46.91 (± 0.59)
KD W/ QDI (OURS)	–	18.85 (± 0.30)	0.66 (± 0.38)	**15.86** (±**1.51**)	**32.73** (±**0.77**)	**15.38** (±**1.67**)	**37.27** (±**1.31**)
ER (Chaudhry et al., 2019a)	✓	38.77 (± 1.99)	69.61 (± 2.20)	7.43 (± 0.36)	82.85 (± 0.20)	7.59 (± 0.12)	61.77 (± 0.31)
A-GEM (Chaudhry et al., 2019a)	✓	19.67 (± 0.41)	89.26 (± 2.89)	4.62 (± 0.02)	88.78 (± 0.84)	6.93 (± 0.11)	63.93 (± 0.73)
DER (Buzzega et al., 2020)	✓	44.13 (± 0.98)	57.64 (± 4.76)	10.11 (± 0.65)	80.92 (± 1.35)	8.11 (± 0.26)	56.73 (± 0.81)
DER++ (Buzzega et al., 2020)	✓	48.82 (± 1.75)	50.11 (± 2.74)	10.97 (± 0.55)	73.62 (± 0.86)	8.88 (± 0.61)	55.54 (± 2.13)
KD W/ BUFFER (OURS)	✓	**65.40** (± **0.96**)	**10.13** (± **2.19**)	**21.00** (± **1.01**)	**38.49** (± **1.39**)	**18.77** (± **0.91**)	**8.76** (± **0.79**)

4.3 ADDITIONAL ANALYSIS

Ablation studies. To dissect the contribution of different components in our proposed method for HCL, we isolate and study them on Split CIFAR-10 in Table 3. First, we find that knowledge distillation with augmented instances, in the contrast to a common setting of precomputed logits, improves the performance by $\sim 8\%$ and $\sim 13\%$ for task-IL and class-IL respectively due to the better generalization of current task representations. Next, we observe that label-smoothing improves the task-IL performance while hurting the class-IL average accuracy highlighting the presence of a trade-off between both the settings. Additionally, we measure the effect of warmup and task-identity for HCL following prior works (Jung et al., 2016; Li & Hoiem, 2016) for standard CL, where we find that warm-up of the linear head before current-task training and inclusion of task-identity hurt the performance for HCL. Further, we conduct an ablation on different distance metrics used for KD in Equation 3 for HCL. While, a modified version of cross entropy (CE) or mean-square distance (MSE) are commonly used for knowledge distillation in CL (Li & Hoiem, 2016; Rebuffi et al., 2017; Buzzega et al., 2020), we find the KL-divergence (KL) is more stable and often obtains better performance in Table 4 for both class-IL and task-IL. We believe that the ablation of these design choices will provide critical insights towards building future methods for standard and heterogeneous CL.

Transfer to vision transformer. To show the flexibility of our framework beyond convolutional architectures, we design a two task experiment with CIFAR-10 consisting of five classes each. The first task is trained with ResNet18 (He et al., 2016) followed by Swin-T (Liu et al., 2021b) on the second task. Table 5 shows the comparison of our proposed method with only KD against FINETUNE, DER++ trained with HCL and MULTITASK learned with CIFAR-10 on Swin-T (Liu et al., 2021b). First, we remark that Swin-T (MULTITASK: 88.71 ± 0.40) achieves lower accuracy in comparison to ResNet18 (MULTITASK: 98.31 ± 0.12) because ViTs are data-hungry (Liu et al., 2021a). Second, we show that our proposed method with evolving architectures obtains performance similar to the multitask baseline. Lastly, our proposed method obtains better average accuracy and lower forgetting over the evaluated baselines showing the benefit of training with multiple architectures.

Table 3: **Ablation** showing $\mathcal{A}_t$ to measure the effect of various components with Split-CIFAR10. The inclusion of distillation with augmented images (AUG-KD) and label-smoothing (LS) significantly improve performance, whereas WARMUP and task-identity (TI) degrade performance. Blue and red colors denote the components that were included and excluded from the method respectively.

MODEL	TASK-IL	CLASS-IL
KD	83.43 (± 0.42)	21.11 (± 0.19)
+ AUG-KD	91.76 (± 2.03)	34.11 (± 1.07)
+ LS	93.13 (± 0.35)	30.21 (± 0.11)
+WARMUP	91.64 (± 1.37)	28.26 (± 0.28)
+ TI	90.14 (± 1.33)	22.04 (± 0.27)

Table 4: **Ablation** showing $\mathcal{A}_t$ using validation set to measure the effect of different distance metrics. We measure the average accuracy and forgetting for HCL on Split-CIFAR10 dataset.

MODEL	TASK-IL	CLASS-IL
CE	91.06 (± 0.63)	20.99 (± 0.27)
MSE	89.68 (± 0.99)	20.09 (± 0.26)
KL	93.13 (± 0.35)	30.21 (± 0.11)

Table 5: **Transformer analysis.** Average accuracy and forgetting for HCL with ResNet18 to Swin-T with two tasks on CIFAR-10.

MODEL	$\mathcal{A}_T$ (↑)	$\mathcal{F}_T$ (↓)
FINETUNE	55.47 (± 2.06)	77.68 (± 3.68)
DER++	69.05 (± 2.56)	51.99 (± 1.91)
KD (OURS)	88.77 (± 0.55)	10.95 (± 1.34)
MULTITASK	88.71 (± 0.40)	–

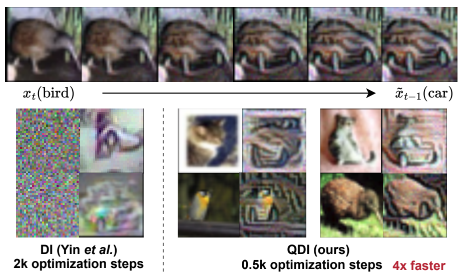

Figure 4: Visualization of QDI and comparison to prior art. **Top:** prior feature synthesis from current task data. **Bottom:** samples for DI that start from $\mathcal{N}(0,1)$ and QDI from x_t in pairs (left – optimization start point; right – optimization convergence point) for ResNet18 on CIFAR-10 dataset.

Visualization of generated examples. We visualize the generated examples by DI and QDI in the HCL setting for ResNet18 architecture with CIFAR-10 dataset in Figure 4. To explain the interplay of the generated examples initialized with the current task instances and the past task examples, we consider the example of a current task (*e.g.,* a cat or bird) to invert the visual features of the past task (car). We observe that the input-features swiftly generated by QDI have a combination of both classes and superior in realism; in contrast, DI only contains the past-task features and requires more optimization steps from random noise for full convergence. We believe this interpolation also helps in current task-training while also alleviating catastrophic forgetting.

5 DISCUSSION AND CONCLUSION

In this paper, we propose a novel continual learning setting named *Heterogeneous Continual Learning (HCL)*, where the lifelong learner learns on a sequence of tasks while adopting the state-of-the-art deep-learning techniques and architectures. We benchmark the state-of-the-art CL solutions in this new setting, and observe a large degradation in performance. To tackle this limitation, we revisit knowledge distillation and propose a modified paradigm inspired by the recent advances in knowledge distillation. Additionally, we propose *Quick Deep Inversion (QDI)* that generates synthetic examples using current task instances to enhance knowledge distillation performance in the data-free continual learning setup. Our experimental evaluation shows that without tuning the training configurations, we achieve an improvement for both the standard and heterogeneous continual learning settings across all datasets highlighting the efficacy of our approach in practical scenarios.

Limitations. Although the proposed approach is applicable for general deep-learning architectures, we found some limitations with the current approach/setup. First, in some cases, the recent architectures can be sub-optimal and degrade the model performance due to their training configuration, model size, or hyper-parameter mismatch. We note that we did not tune the training configuration or hyper-parameters for each model and a careful hyper-parameter tuning for different types of architectures can further improve model performance. Second, our work is not yet applicable to unsupervised continual learning with heterogeneous architectures, where the learner is expected to learn on unlabelled data with changing architectures because model inversion and knowledge distillation with unsupervised CL is not a trivial extension due to the absence of class-dependent information to generate class-conditioned examples.

Ethics statement. The continuous change of architectures makes continual learning extremely powerful. Therefore, it might lead to unethical applications of these networks and it is essential to regulate and audit the use of these models during deployment.

Reproducibility statement. We provide the details for all the datasets with the hyper-parameter setups in Appendix A.1. We will release the source code to the community with all hyper-parameters and scripts to facilitate reproducibility, further analysis, and future research.

References

Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 2, 3

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2, 3

Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. Knowledge distillation: A good teacher is patient and consistent. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 5

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2, 3, 4, 5, 6, 7, 8, 14

Antonio Carta, Andrea Cossu, Vincenzo Lomonaco, and Davide Bacciu. Ex-model: Continual learning from a stream of trained models. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2022. 3, 7

Keshigeyan Chandrasegaran, Ngoc-Trung Tran, Yunqing Zhao, and Ngai-Man Cheung. Revisiting label smoothing and knowledge distillation compatibility: What was missing? In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022. 5

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019a. 6, 7, 8

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and M Ranzato. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486*, 2019b. 6

Andrea Cossu, Gabriele Graffieti, Lorenzo Pellegrini, Davide Maltoni, Davide Bacciu, Antonio Carta, and Vincenzo Lomonaco. Is class-incremental enough for continual learning? *Frontiers in Artificial Intelligence*, 2022. 4

National Research Council. *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*. The National Academies Press, 2000. doi: 10.17226/9853. 1

Xiaoliang Dai, Hongxu Yin, and Niraj K Jha. Incremental learning using a grow-and-prune paradigm with efficient neural networks. *IEEE Transactions on Emerging Topics in Computing*, 2020. 3

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 4

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 1, 6, 14

Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 1, 4

Timo Flesch, Jan Balaguer, Ronald Dekker, Hamed Nili, and Christopher Summerfield. Comparing continual task learning in minds and machines. *Proceedings of the National Academy of Sciences*, 2018. 1

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 3, 4, 6, 8, 14

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2, 4, 5

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6, 14

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 14

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. 6, 14

Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016. 8

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012. 6, 14

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012. 1, 6, 14

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989. 6, 14

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 1

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 2

Seungwon Lee, Sima Behpour, and Eric Eaton. Sharing less is more: Lifelong learning in deep networks with selective layer transfer. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. 3

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019. 2, 3

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 2, 3, 5, 6, 7, 8

Yahui Liu, Enver Sangineto, Wei Bi, Nicu Sebe, Bruno Lepri, and Marco De Nadai. Efficient training of visual transformers with small-size datasets. *arXiv preprint arXiv:2106.03746*, 2021a. 8

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021b. 1, 2, 4, 8

Divyam Madaan, Jaehong Yoon, Yuanchun Li, Yunxin Liu, and Sung Ju Hwang. Representational continuity for unsupervised continual learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 1

Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *arXiv preprint arXiv:2111.09259*, 2021. 1

Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Delight: Deep and light-weight transformer. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 1

Seyed Iman Mirzadeh, Arslan Chaudhry, Huiyi Hu, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Wide neural networks forget less catastrophically. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022a. 3

Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Huiyi Hu, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Wide neural networks forget less catastrophically. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022b. 1

Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning. *arXiv preprint arXiv:2202.00275*, 2022c. 3

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021. 1

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 3, 5, 6, 7, 8

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 2, 3, 7, 8

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. 1, 2, 3

Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018. 2, 3

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 1, 6, 14

James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 7

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI National Conference on Artificial Intelligence (AAAI)*, 2017. 1, 6, 14

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019. 1

Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Neurips*, 1996. 1

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision, 2021. 4

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017. 1

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 6, 14

Jing Xu, Yu Pan, Xinglin Pan, Steven Hoi, Zhang Yi, and Zenglin Xu. Regnet: self-regulated network for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 2, 6, 14

Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via DeepInversion. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 5, 6, 7, 8

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 2, 3

Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 3

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1, 2

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 1, 2, 3, 6, 7, 8

A Supplementary Material

Organization. In the supplementary material, we provide the implementation details and hyper-parameter configurations in Appendix A.1. Further, we show additional visualization of examples generated by QDI in Appendix A.2.

A.1 Experimental Details

This section provide details on the datasets and architectures and implementation of HCL.

A.1.1 Datasets and architectures

In this subsection, we provide the details about the datasets and the architectures used for evaluation.

1. **Split CIFAR-10 (Krizhevsky, 2012).** It consists of CIFAR-10 sized 32×32 split into five tasks, where each task contains two different classes. In the HCL setting, we consider a different architecture for each task: LeNet (LeCun et al., 1989), ResNet (He et al., 2016), DenseNet (Huang et al., 2017), SENet (Hu et al., 2018), and RegNet (Xu et al., 2022).
2. **Split CIFAR-100 (Krizhevsky, 2012).** It is a split of CIFAR-100, where the 100 object classes are divided into 20 tasks. We consider nine architectures in the HCL: LeNet (LeCun et al., 1989), AlexNet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2015) with batch normalization (Ioffe & Szegedy, 2015), InceptionNet (Szegedy et al., 2017), ResNet (He et al., 2016), ResNeXt (Xie et al., 2017), DenseNet (Huang et al., 2017), SENet (Hu et al., 2018) for two tasks each and RegNet (Xu et al., 2022) for last four tasks.
3. **Split Tiny-ImageNet.** It is a variant of ImageNet (Deng et al., 2009) that consists of images sized 64×64 from 200 classes. It consists of 20 tasks, where each task contains 10 classes. The set of architectures in HCL include LeNet (LeCun et al., 1989), ResNet (He et al., 2016), ResNeXt (Xie et al., 2017), SENet (Hu et al., 2018),and RegNet (Xu et al., 2022) architectures, where each architecture is used for two consecutive tasks.

A.1.2 Hyperparameter setup

We follow the base hyper-parameter setup from Buzzega et al. (2020) for SCL and HCL experiments. We use an SGD optimizer for experiments with base learning rate as 0.03 for all the models. For each new task a total of 200 epochs are utilized to train the current model and we use early-stopping on the validation set to store the best-model. Batch size is set as 32 and training is conducted on one NVIDIA V100 GPU of 16G for CIFAR-10 experiment and 32G for Split CIFAR-100 and Tiny-ImageNet datasets. QDI previous class target synthesis labels evenly sampled across previous tasks. Optimization hyper-parameters for QDI are provided next for three datasets, each shared across all networks during the training run:

- **Split CIFAR-10 (Krizhevsky, 2012).** Optimization based on Adam optimizer of learning rate 0.005, $\alpha_{tv} = 0.001, \alpha_{\ell_2} = 0, \alpha_{\text{feature}} = 0, \alpha_{\text{feature}} = 0.1$ for 0.5K iterations.$d(\cdot, \cdot)$ is MSE loss.
- **Split CIFAR-100 (Krizhevsky, 2012).** Optimization based on Adam optimizer of learning rate 0.03, $\alpha_{tv} = 0.003, \alpha_{\ell_2} = 0.003, \alpha_{\text{feature}} = 0.2$ for 0.5K iterations. $d(\cdot, \cdot)$ is MSE loss.
- **Split Tiny-ImageNet.** Optimization based on Adam optimizer of learning rate 0.05, $\alpha_{tv} = 0.001, \alpha_{\ell_2} = 0.05, \alpha_{\text{feature}} = 0.5$ for 0.5K iterations. $d(\cdot, \cdot)$ is MSE loss.

For each dataset we found the QDI hyper-parameters based on a validation set obtainend by randomly sampling 10% of the training set. All results in main paper are on the test set, with 3 independent runs of the hyperset from random seeds for means and standard deviations.

A.2 Additional Visualization

We provide extra visualization of QDI samples for the CIFAR-100 and Tiny-ImageNet datasets as in Fig. A.5. It can be observed that the proposed method scales across datasets with high fidelity in synthesized samples. More interestingly, the optimization step can find very close proxy of the current task semantics in older domains and dream out visual features of high perceptual realism.

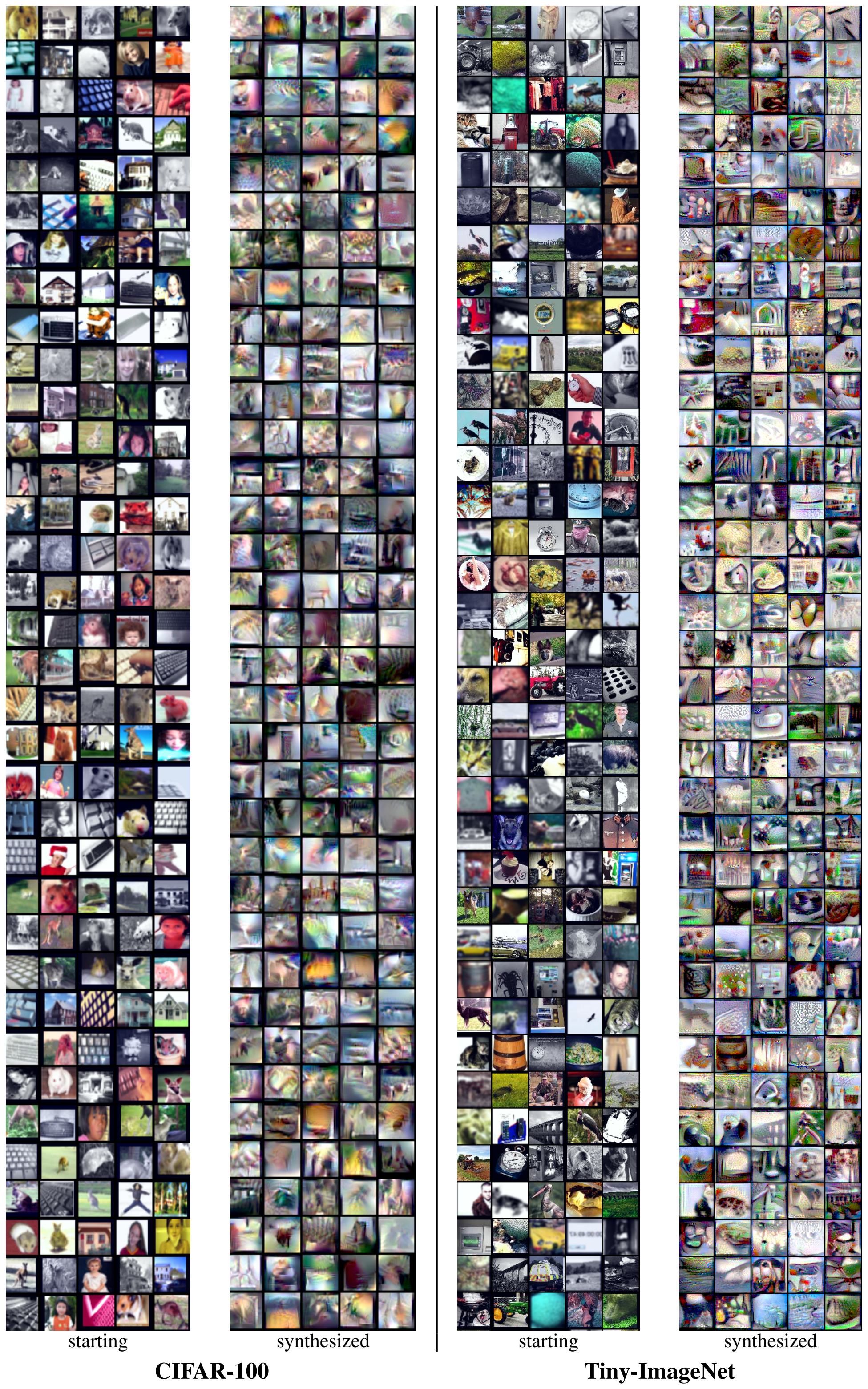

Figure A.5: More QDI visualization for CIFAR-100 and Tiny-ImageNet datasets. Best viewed in color.