THE EFFECTIVENESS OF MEMORY REPLAY IN LARGE SCALE CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We study continual learning in the large scale setting where tasks in the input sequence are not limited to classification, and the outputs can be of high dimension. Among multiple state-of-the-art methods, we found vanilla experience replay (ER) still very competitive in terms of both performance and scalability, despite its simplicity. However, a degraded performance is observed for ER with small memory. A further visualization of the feature space reveals that the intermediate representation undergoes a distributional drift. While existing methods usually replay only the input-output pairs, we hypothesize that their regularization effect is inadequate for complex deep models and diverse tasks with small replay buffer size. Following this observation, we propose to replay the activation of the intermediate layers in addition to the input-output pairs. Considering that saving raw activation maps can dramatically increase memory and compute cost, we propose the Compressed Activation Replay technique, where compressed representations of layer activation are saved to the replay buffer. We show that this approach can achieve superior regularization effect while adding negligible memory overhead to replay method. Experiments on both the large-scale Taskonomy benchmark with a diverse set of tasks and standard common datasets (Split-CIFAR and Split-miniImageNet) demonstrate the effectiveness of the proposed method.

1 INTRODUCTION

Humans naturally learn concepts and tasks in a sequential order without degrading performance on the previous ones. Can machines do the same? This setting, known as *continual learning*, poses serious challenges for the deep learning community. Deep neural networks often fail to maintain the performance on the previous tasks if they cannot fully access their data, a phenomenon called *catastrophic forgetting*. The underlying cause is the assumption that models learn from *i.i.d.* (independent and identical distributed) data. This assumption is often untrue in the real world where data can come sequentially and their distributions can always be evolving. The solution to this issue will be a key to a system that allows machines to learn continuously in the real world.

As the machine learning models become deeper and wider while the size of the data gets bigger and the tasks become more diverse, the catastrophic forgetting issue becomes more significant and concerning. In recent years, much work has been proposed to address the forgetting problem. However, existing continual learning methods are usually developed and evaluated on relatively small datasets, often with classification tasks only. The scalability of these methods and their effectiveness in real scenarios with a diverse pool of tasks are still an open question and often being overlooked.

The first question we want to study is: *how different classes of methods work in large scale continual learning with a diverse set of tasks?* We conduct a systematic evaluation of various continual learning algorithms using a comprehensive multi-task benchmark based on the Taskonomy dataset (Zamir et al., 2018). This dataset contains not only classification tasks but also regression with high dimensional outputs. Interestingly, we observe in this large scale setting that the vanilla memory replay method outperforms a variety of state-of-the-art approaches, based on a range of evaluation protocols using forgetting reduction, generalization, scalability and inference efficiency.

Though being competitive, we observe a degraded performance of vanilla experience replay with a small memory size. A t-SNE plot of the feature space, illustrated in Fig. 1 top middle, reveals that the intermediate representation still undergoes a distributional shift. We believe it is due to the
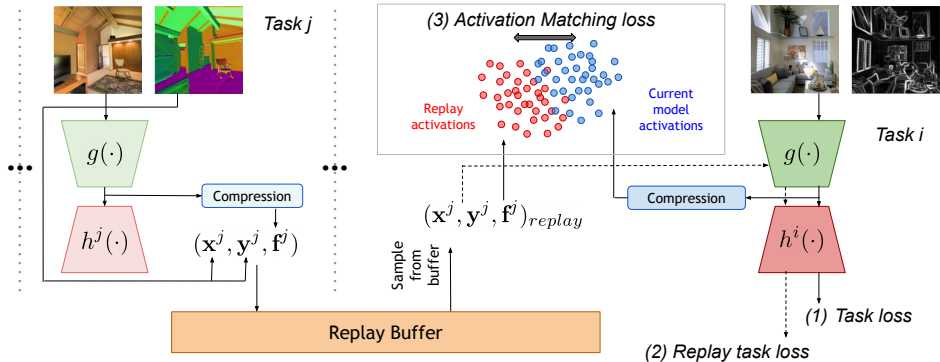
Figure 1: **Compressed Activation Replay (CAR)**: During sequential training, compressed feature maps are stored in addition to the input-output pairs in the replay buffer. While training future tasks, models are optimized using (1) task loss, (2) replay task loss using samples from the replay buffer and (3) activation matching loss between the compressed activation maps of the current encoder model and the replay samples. Activation matching loss present distribution drift from happening in the feature space.

fact that most replay methods store and regularize using only the input-output pairs from past tasks. Considering that modern neural networks are often wide and deep, we argue that replaying only the input-output pairs are not enough to constrain the feature space. A natural solution is to augment the replay memory with the activation of intermediate layers to regularize representation learning.

However, given the size of models used in large scale scenarios, these activation tensors can be very expensive in terms of both memory and computation cost. Therefore, we propose a *Compressed Activation Replay (CAR)* framework which regularizes the model by matching the compressed versions of those activation tensors (Fig. 1). The chosen compression function is based on average pooling along both spatial and channel dimensions, which results in negligible added compute cost compared to vanilla memory replay. Experimental results also suggest that the compressed replay approach produces equivalent performance compared to replaying the full activation tensors, and outperforms the baseline replay algorithm that only replays the raw input-output pairs.

We evaluate our approach on the large scale Taskonomy benchmark to demonstrate its scalability. New performance evaluation metrics are proposed to tailor the need of large scale setups with diverse tasks. In order to compare with other methods in literature, we also conduct experiments on two medium-sized popular datasets, *i.e.*, Split-CIFAR and Split-miniImageNet. Empirical results suggest that our approach achieves the state of the art in all three benchmarks.

In summary, our contributions are as follows:

1. We provide a thorough evaluation of popular continual learning methods at a large scale with diverse tasks which is mostly missing in the published literature.
2. We show the vanilla experience replay is the most effective approach in the large scale setting in the presence of task diversity.
3. We observe that intermediate representations still undergo a drift in continual learning process reinforcing the catastrophic forgetting.
4. We propose Compressed Activation Replay (CAR) to improve the performance of experience replay, reaching state-of-the-art results among replay based methods.

2 RELATED WORK

Catastrophic forgetting. Catastrophic forgetting is a major challenge in deep learning where the performance of past examples may drop significantly in a sequential training setting. Continual learning aims to address this problem. The approaches are often categorized into three major directions: replay, expansion and regularization based methods. Replay based methods require a memory module to memorize past or generative examples (see next paragraph for more discussion). Regularization methods are achieved by introducing regularization terms into the loss function which

constraint the model weights (Kirkpatrick et al., 2017; Zenke et al., 2017; Ritter et al., 2018; Farajtabar et al., 2020; Yin et al., 2020; Adel et al., 2020). Expansion based methods (Rusu et al., 2016; Li & Hoiem, 2017; Yoon et al., 2018; Li et al., 2019; Mirzadeh et al., 2020) tackle the problem by expanding the neural architecture along the training. Specifically, the predictive capability on the past examples is retained by freezing the architecture related to past tasks while the new examples are further trained with an extended set of network parameters. These approaches often need to grow all the layers of a neural network and lead to large models in the end. Lacking a proper way for parameter sharing, most of the expansion based methods are found hard to be applied to large scale tasks.

Memory replay methods. Mitigating catastrophic forgetting via memory replay is a classic idea that dates back to the 90s (Robins, 1995). These methods typically save a sampled set of input-output pairs into a memory module and mix these samples with new samples in the current task for training the models. It has been observed that replay based algorithms usually outperform regularization based methods (Nguyen et al., 2017; van de Ven & Tolias, 2019), and some explanation have been made by Knoblauch et al. (2020). Among replay based methods, some try to design new update rules using replay samples (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018), while others focus on new sampling strategies to fill the replay buffer (Rebuffi et al., 2017; Chaudhry et al., 2020; Aljundi et al., 2019; Hayes et al., 2019; Chrysakis & Moens, 2020; Borsos et al., 2020). Another line of research uses generative models to construct replay examples (Shin et al., 2017; van de Ven et al., 2020). Replaying the experience is also popular in reinforcement learning problems (Mnih et al., 2013; Schaul et al., 2015; Isele & Cosgun, 2018; Rolnick et al., 2019).

One recent work that is particularly related to our proposed approach is the embedding regularization algorithm (Pomponi et al., 2020). In this algorithm, the network embedding of replay examples are stored in memory and a regularization loss is constructed using the embedding such that the feature extraction layers remain stable. While the basic idea of this work is similar to our proposed method, we differentiate in two ways: 1) in our proposed Compressed Activation Replay method, only a compressed version of the network embedding is stored in the replay buffer, which significantly reduces the memory cost without sacrificing the regularization effect, and 2) we benchmark the performance of the approach on large scale non-classification tasks such as Taskonomy, whereas Pomponi et al. (2020) only provides results based on MNIST and CIFAR.

Large scale continual learning. Most existing continual learning literature still focuses on evaluations on small-scale dataset such as MNIST and CIFAR. The Taskonomy dataset (Zamir et al., 2018) has been used recently for continual learning study in (Zhang et al., 2020), where they propose the Side-Tuning algorithm to mitigate catastrophic forgetting. Side-Tuning is an expansion based method, which trains a separate side network for every new tasks and blend the output of the side network with that of the previous frozen networks. The method is able to achieve zero forgetting since all previous tasks' models are frozen. A potential concern for this method is that the inference time of later tasks is significantly higher than that of earlier tasks. Our proposed approach differs from Side-Tuning in that we allow the representation network to be shared and evolved over tasks with stable model capacity and inference time. Another large scale dataset that is often used in continual learning research is the CORe50 dataset (Lomonaco & Maltoni, 2017), which mainly focuses on object detection. In this work, we choose Taskonomy since it contains more diverse tasks.

3 PRELIMINARIES

In this section, we introduce the problem formulation and evaluation metrics of the type of continual learning problems considered in this work, together with a review of the memory replay basics.

Continual learning. In continual learning, we aim to solve a sequence of supervised learning tasks $(\mathcal{T}^1, \mathcal{T}^2, \ldots \mathcal{T}^t)$, where, $\mathcal{T}^i = \{(\mathbf{x}_j^i, \mathbf{y}_j^i)\}_{j=1}^{N_i}$ denotes the data examples for task i. The objective is to perform sequential training while preserving the performance on the past tasks without access to all of their data. Let $\ell^i(\cdot, \cdot)$ denote the loss function for task $\mathcal{T}^i$, which can be any generic loss function (*e.g.,* mean-squared-error or cross entropy), and is not tied to any specific task such as classification. In this work, we adopt an encode-decode model. More specifically, let $g(\cdot)$ denote the shared encoder model, and $h^i(\cdot)$ the task head for the task $\mathcal{T}^i$. Predictions on the task $\mathcal{T}^i$ are made using the model $p^i(\mathbf{x}) = h^i(g(\mathbf{x}))$. The task loss for optimizing task $\mathcal{T}^i$ is then given by

$$\mathcal{L}_{task}^i = \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{T}^i} \ \ell^i(p^i(\mathbf{x}), \mathbf{y}). \tag{1}$$

Memory replay. In the simplest version of memory replay based continual learning, we maintain a replay buffer $\mathcal{M}$ of size m per task (total buffer size is mt) for storing samples from the task sequence seen thus far. In addition to task loss, models are optimized using a replay loss given by

$$\mathcal{L}^j_{replay} = \mathbb{E}_{(\mathbf{x}^j_r, \mathbf{y}^j_r) \in \mathcal{M}} \ \ell^j(p^j(\mathbf{x}^j_r), \mathbf{y}^j_r). \tag{2}$$

Here, $(\mathbf{x}^j_r, \mathbf{y}^j_r)$ denotes the samples from the replay buffer belonging to task j. In this work, the data in the replay buffer are sampled uniformly at random from the training data for each task. The overall optimization objective in vanilla experience replay (ER) for training task i is then given by

$$\min_{g, h^i} \ \mathcal{L}^i_{task} + \frac{\lambda}{i-1} \sum_{j=1}^{i-1} \mathcal{L}^j_{replay}, \tag{3}$$

where, λ is a hyper-parameter that balances the current task loss and the replay losses.

Evaluation metric. One of the challenges in continual learning with non-classification tasks (textite.g.,, Taskonomy dataset) is the evaluation metric. In this work, for Taskonomy, we propose *forgetting* and *performance drop* as two major metrics. Let $\ell_{test}(i, j)$ denote the test loss of task j using the model snapshot obtained at the end of training task i in the task sequence, and let $\ell^{MT}_{test}(i)$ denote the test loss of task i obtained from a multi-task model trained simultaneously on all tasks. Then, the two metrics are defined as follows:

$$\text{Forgetting} := \quad \frac{1}{t} \sum_{i=1}^{t} \frac{\ell_{test}(t, i) - \ell_{test}(i, i)}{\ell_{test}(i, i)} \times 100\%$$

$$\text{Performance drop} := \quad \frac{1}{t} \sum_{i=1}^{t} \frac{\ell_{test}(t, i) - \ell^{MT}_{test}(i)}{\ell^{MT}_{test}(i)} \times 100\%$$

The *forgetting* measure indicates the average loss increase of the model compared to the model snapshot at the end of ith task, while the *performance drop* quantifies the performance drop of the final model compared to a multi-task model trained using the same model architecture.

For classification tasks, *e.g.,* Split-CIFAR and Split-miniImageNet, we report *forgetting* and *average accuracy* as two main metrics. Specifically, let $Acc(i, j)$ denotes the accuracy on task j using the model snapshot obtained at the end of training task i. We have

$$\text{Forgetting} := \quad \frac{1}{t} \sum_{i=1}^{t} \max_j \left[Acc(j, i) - Acc(t, i) \right], \quad \text{Avg accuracy} := \quad \frac{1}{t} \sum_{i=1}^{t} Acc(t, i).$$

4 VANILLA EXPERIENCE REPLAY IS A STRONG BASELINE

We begin our study by investigating the performance of different classes of continual learning algorithms on Taskonomy, which is a large scale dataset with diverse set of tasks. We compare to Side-Tuning (Zhang et al., 2020), an expansion based method mentioned in Section 2, EWC (Kirkpatrick et al., 2017), a regularization algorithm that constructs the regularizer based on the Fisher information of the model's weights, A-GEM (Chaudhry et al., 2018) a memory replay baseline that uses replay examples to construct a constrained optimization problem, and finaly ER the vanilla experience replay algorithm described in equation 3. For completeness, we also compare with the vanilla SGD algorithm that trains the model sequentially on all the tasks without storing extra information, and the multi-task (MT) algorithm which has access to all the tasks simultaneously.

4.1 EXPERIMENT SETUP

Taskonomy (Zamir et al., 2018) is a large scale dataset of indoor scenes from various buildings containing annotations for several diverse computer vision tasks. We use data from the following 11 tasks in this work: *curvature estimation, object classification, scene classification, surface normal estimation, semantic segmentation, depth estimation, reshading, edge detection (texture), edge detection (occlusion), 2D keypoint estimation and 3D keypoint estimation.* All input samples are of 256×256 resolution. We use a subset of Taskonomy dataset containing 50 buildings in all our

Table 1: **Taskonomy results** using various methods. Forgetting and performance drop (ΔPerformance) values are reported. 11 different tasks are used. All results are averaged over 3 random seeds. ER-m means we store m data examples for each task. Top two results for forgetting and performance are highlighted in **bold**.

Method	MT	SGD	Side-Tuning (expansion)	EWC (regularization)	A-GEM (replay)	ER-64 (replay)	ER-256 (replay)
	-	-					
Extra Storage (GB)	410^1	0	0.46	5.40^2	0.44	0.11	0.44
Final model size (GB)	1.30	1.30	1.76	1.30	1.30	1.30	1.30
Forgetting (%)	N/A	191.61	**0**	169.62	38.88	40.18	**14.73**
ΔPerformance (%)	**0**	101.56	13.12	96.75	49.01	20.70	**8.52**

experiments. Of these, 5 randomly chosen buildings are treated as held-out test set, while the rest of the data are used for training and validation. This yields a total training size of $324,864$ samples.

Loss function. Following Zamir et al. (2018), object and scene classification tasks are trained using cross entropy loss, semantic segmentation using pixelwise cross entropy, curvature estimation using ℓ_1 loss, and all other tasks using ℓ_2 loss.

Model. We use Resnet-50 model as our encoder network $g(\cdot)$. Since most tasks in Taskonomy dataset are pixel-wise tasks, we do not perform the spatial average pooling in the pen-ultimate layer of Resnet-50 model. This results in a feature map of dimension $2048 \times 8 \times 8$. Following Zamir et al. (2018), we use 15-layer ConvNet model with upsampling blocks as the task heads h^i.

4.2 RESULTS

Table 1 contains the result of our large scale analysis. Interestingly, we found that despite its simplicity, the vanilla experience replay algorithm is a strong baseline on Taskonomy. Similar findings were also reported by Chaudhry et al. (2019) but on conventional and mainstream CL datasets like split CIFAR, mini-ImageNet, and CUB which are less realistic, less diverse and smaller in the scale. We believe these results indicate that memory replay based algorithm can be a very promising direction for large scale CL that should not be ignored.

We note that in Table 1 that expansion based method Side-Tuning achieves zero forgetting and competitive performance drop. However, it requires a growing neural architecture which results in a large final model with more expensive inference cost. Its compute overhead will further increase when seeing larger numbers of tasks. After each task, Side-Tuning needs to add an additional side network which is also found expensive in the training storage cost. That prevents this method being applied to complex tasks that requires models with large capacity.

We also observe a degraded performance of vanilla experience replay with a small memory size, *i.e.,* $m = 64$ is worse than $m = 256$. Although not surprising, the degraded performance inspired our study on the feature space which leads to the proposed improvement approach in the next section.

5 COMPRESSED ACTIVATION REPLAY

Vanilla experience replay stores only input-output pairs and uses them to construct the replay loss. However, we suspect that, for large neural networks, intermediate features (activations) may undergo drifts when we train the model sequentially on different tasks. If that is the case, the input-output pairs may not be sufficient to mitigate this drift in the intermediate layers, leading to catastrophic forgetting. To better understand this phenomenon, we plot the t-SNE (van der Maaten & Hinton, 2008) visualization of the feature space over the course of vanilla ER in Figure 2(a). We observe that indeed as new tasks are being learnt, feature representations of the older tasks start drifting. To circumvent this issue, we propose using feature matching as an additional regularization term.

Let $(\mathbf{x}_r^j, \mathbf{y}_r^j)$ be a sample from task j, and let $\mathbf{f}_r^j = g(\mathbf{x}_r^j)$ be the encoder representation (*i.e.,* activation) of this sample at the end of the training of task j. The neural net prediction of this sample is thus $p^j(\mathbf{x}_r^j) = h^j(\mathbf{f}_r^j)$. The basic idea of feature matching regularization is that, when we train a subsequent task, say task i, the encoder representation of the replay example $\mathbf{x}_r^j$, *i.e.,* $g(\mathbf{x}_r^j)$ should stay

---

[2]This is the estimated training data size excluding the current task which needs access by multi-task learning.
[2]EWC needs access to all previous tasks' encoder weights with the diagonal of fisher information.

(a) Experience Replay *(m=64)*
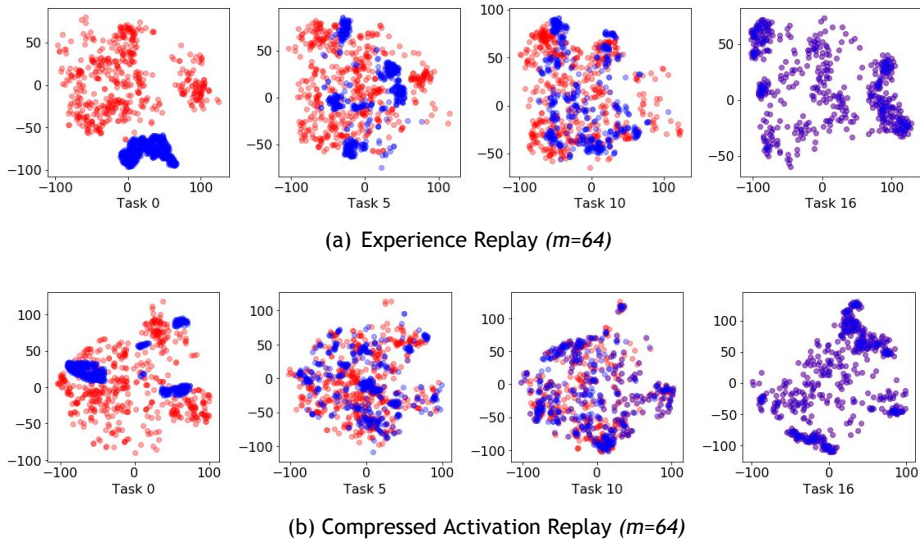

(b) Compressed Activation Replay *(m=64)*

Figure 2: **Visualizing t-SNE embeddings** of model trained using vanilla experience replay (a) and Compressed Activation Replay (b) on Split-CIFAR dataset. We compare the features of the encoder of the final model (*shown in blue*) relative to the encoder representations obtained while training the respective tasks (*shown in red*). We observe that older tasks encounter significant drift in the feature space when trained with experience replay. Using activation matching loss, our Compressed Activation Replay framework reduces the drift as the blue dots better cover the support of red dots, especially on Task 5 and 10. We note that for the very early task (Task 0), feature drift can still be observed even if we use CAR.

close to $\mathbf{f}_r^j$. To achieve this, we add an additional regularization term $\ell_{fm}(g(\mathbf{x}_r^j), \mathbf{f}_r^j)$ when training the subsequent tasks, where ℓ_{fm} is a loss function that measures the difference between $g(\mathbf{x}_r^j)$ and $\mathbf{f}_r^j$. In the following, for most experiments we use a simple MSE loss, *i.e.*, $\ell_{fm}(\mathbf{a}, \mathbf{b}) \equiv \|\mathbf{a} - \mathbf{b}\|_2^2$. In our ablation study in Section 6.3, we compare different choices of ℓ_{fm}.

The activation tensors of intermediate layers in neural networks are usually of high dimensions, and storing them in the replay buffer can be expensive in memory cost. Since we also want to avoid too much overhead to the memory and compute cost, we empirically study a variant of feature matching regularization by compressing the activation into small-length vectors and only regularize the models in regards to such compressed vector representations. Formally, we use a compression function c to map the high dimensional feature $\mathbf{f}_r^j$ to a low dimensional vector $c(\mathbf{f}_r^j)$, and then apply the feature matching loss ℓ_{fm}. In this case, the feature matching loss for task j can be written as

$$\mathcal{L}_{fm}^j = \mathbb{E}_{(\mathbf{x}_r^j, \mathbf{y}_r^j, \mathbf{f}_r^j) \in \mathcal{M}} \ \ell_{fm}(c(g(\mathbf{x}_r^j)), c(\mathbf{f}_r^j)), \tag{4}$$

and then the objective for training a subsequent task i can then be written as

$$\min_{g, h^i} \ \mathcal{L}_{task}^i + \frac{1}{i-1} \sum_{j=1}^{i-1} \left[\lambda \mathcal{L}_{replay}^j + \lambda_{fm} \mathcal{L}_{fm}^j \right], \tag{5}$$

where λ_{fm} is another coefficient balancing the feature matching loss and other terms in the loss function. We name this approach Compressed Activation Replay (CAR). As presented in Figure 2(b), by using CAR, the drift in the encoder representations is significantly reduced. As for the compression function, a simple way to compress is to perform average pooling along axes (spatial and/or channel) of the tensors. Interestingly, we found this simple compression performs well in our experiments. In our ablation studies in Section 6.3, we compare different compression techniques.

6 EMPIRICAL STUDIES

6.1 TASKONOMY

We compare the proposed CAR approach with multiple baseline methods in the Taskonomy dataset mentioned in Section 4: vanilla SGD, A-GEM, and vanilla experience replay (ER).
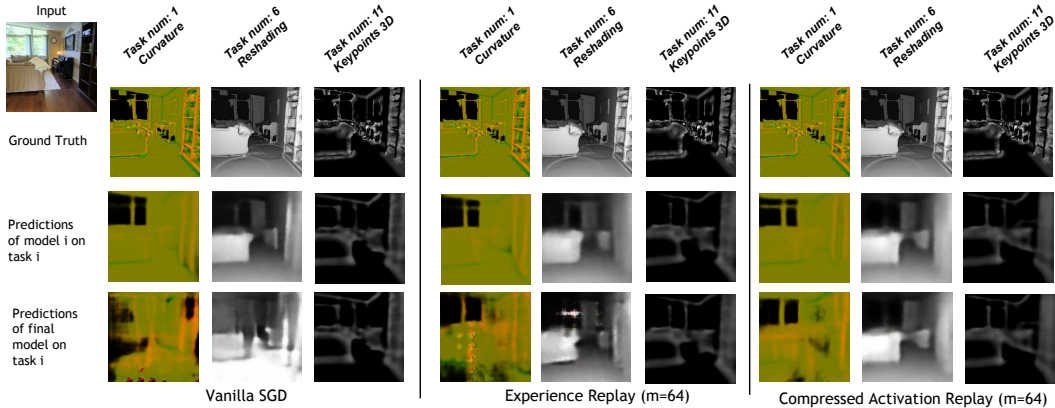
Figure 3: Visualizations of predictions made on Taskonomy dataset. Compressed Activation Replay (CAR) drives the predictions to higher quality, compared with vanilla SGD and ER.

Table 2: **Taskonomy results** using various replay methods. Forgetting ($fgt\pm$std) and performance drop ($\Delta perf\pm$std) values are reported (smaller values are better). All results are averaged over 3 random seeds.

Method	$m = 64$		$m = 128$		$m = 256$		$m = 512$	
	fgt ($\downarrow$)	$\Delta perf$ ($\downarrow$)	fgt ($\downarrow$)	$\Delta perf$ ($\downarrow$)	fgt ($\downarrow$)	$\Delta perf$ ($\downarrow$)	fgt ($\downarrow$)	$\Delta perf$ ($\downarrow$)
Sequence: Curv, Cls (obj), Normal, Seg, Depth, Reshade, Edge (tex), Cls (sc), Keypts2d, Edge (occ), Keypts3d								
SGD	191.6±13.8	101.6±9.2	191.6±13.8	101.6±9.2	191.6±13.8	101.6±9.2	191.6±13.8	101.6±9.2
A-GEM	43.6±0.7	48.0±0.9	44.1±2.3	50.2±2.8	38.9±1.9	49.0±2.9	24.9±3.1	50.9±4.7
ER	40.2±1.9	20.7±0.5	24.7±1.4	13.8±0.7	14.7±3.1	8.5±0.5	8.3±0.8	5.4±2.4
CAR (ours)	13.4±3.6	23.3±2.8	6.3±0.7	19.3±3.1	4.3±0.2	12.4±3.8	2.2±0.8	10.8±2.2
Sequence: Depth, Keypts2d, Seg, Normal, Cls (obj), Curv, Edge (occ), Reshade, Cls (sc), Keypts3d, Edge (tex)								
SGD	110.5±16.3	79.5±14.3	110.5±16.3	79.5±14.3	110.5±16.3	79.5±14.3	110.5±16.3	79.5±14.3
A-GEM	46.5±4.3	64.0±4.9	48.9±3.0	62.3±4.0	46.7±4.8	71.7±4.9	32.6±5.1	60.1±5.7
ER	43.8±2.7	33.4±3.2	24.0±4.5	24.3±3.2	19.9±6.4	12.7±4.5	9.1±3.9	5.3±2.9
CAR (ours)	15.9±4.6	35.6±3.1	7.9±2.6	26.8±4.5	6.7±1.2	14.3±0.7	5.7±3.1	10.8±1.1

Two diferent task sequences are used in the evaluation. Results are shown in Table 2. The trend in both task sequences are consistent where CAR significantly outperforms the other methods on the forgetting measure, *e.g.*, it reduces forgetting rate of vanilla ER by 28% when $m = 64$. The performance drop of CAR is slightly higher but still comparable with ER. This can be explained by the stability-plasticity dilemma in continual learning (Mermillod et al., 2013; Mirzadeh et al., 2020), where, stability against forgetting is achieved at the slight cost in the plasticity of the model, *i.e.*, the ability of quickly acquiring new knowledge. The proposed feature matching loss acts as a regularizer and stabilizes the model resulting in a significant decrease in forgetting traded by a relatively smaller drop in learning new tasks. Nevertheless, the parameter λ_{fm} is designed to control this trade-off and tune it according to the application. According to the tables, bigger memory size also improves the performance for the proposed CAR method. We visualize the model predictions in Figure 3, which show that the images predicted by CAR have higher quality than vanilla SGD and ER.

6.2 SPLIT-CIFAR AND SPLIT-MINIIMAGENET

In addition to the large-scale Taskonomy dataset, we perform experiments on two continual learning datasets - Split-CIFAR and Split-miniImageNet. We follow the protocol in Chaudhry et al. (2020; 2018) for all the experiments – Of the 20 tasks in the task sequence, 3 tasks are used for validation, while the other 17 tasks are used to report the final performance. Each task is trained for a single epoch with a batch size of 10. Since each task is a 5-way classification problem, we train models using cross-entropy loss. We use the reduced Resnet-18 model as the encoder following Chaudhry et al. (2020; 2018). Task heads are one-layer linear networks mapping the encoder layer to the logits.

In addition to the baselines listed previously, we also compare with the three additional replay based methods whose performance on Split-CIFAR and Split-miniImageNet were reported in the literature, i.e., iCaRL (Rebuffi et al., 2017), MER (Riemer et al., 2018), HAL (Chaudhry et al., 2020).

Table 3: **Split-CIFAR and Split-miniImageNet results:** Average accuracy (acc±std %) and forgetting (±std) values are reported. Results are averaged over 5 random seeds. The best result is highlighted in bold.

| Method | SPLIT-CIFAR | | | | SPLIT-MINIIMAGENET | | | |
| | $m = 85$ | | $m = 425$ | | $m = 85$ | | $m = 425$ | |
	$acc(\uparrow)$	$fgt(\downarrow)$	$acc(\uparrow)$	$fgt(\downarrow)$	$acc(\uparrow)$	$fgt(\downarrow)$	$acc(\uparrow)$	$fgt(\downarrow)$
EWC	42.4±3.0	0.25±0.02	-	-	37.7±3.3	0.21±0.03	-	-
iCaRL	46.4±1.2	0.16±0.01	51.2±1.3	0.13±0.02	-	-	-	-
A-GEM	54.9±2.9	0.14±0.03	59.9±2.6	0.10±0.02	48.2±2.4	0.13±0.02	54.3±1.6	0.08±0.01
MER	49.7±2.9	0.19±0.03	60.6±2.1	0.09±0.02	45.4±1.4	0.15±0.01	54.8±1.7	0.07±0.01
ER	56.2±1.9	0.13±0.01	62.6±1.8	0.06±0.01	49.0±2.6	0.12±0.02	54.2±3.2	0.08±0.02
HAL	60.4±0.5	0.10±0.01	64.4±2.1	0.06±0.01	51.6±2.0	0.10±0.01	**57.2±1.5**	**0.06±0.01**
CAR (ours)	**62.4±1.5**	**0.06±0.01**	**66.5±1.8**	**0.04±0.01**	**54.2±2.9**	**0.06±0.01**	56.2±1.8	**0.06±0.01**

Table 4: **Taskonomy compression techniques:** Forgetting (*fgt*), performance drop ($\Delta perf$) and memory overhead (*mem*) values are reported. All results are averaged over 3 random seeds.

| Compression Method | $m = 64$ | | | $m = 128$ | | | $m = 256$ | | | $m = 512$ | | |
	fgt	$\Delta perf$	*mem*	*fgt*	$\Delta perf$	*mem*	*fgt*	$\Delta perf$	*mem*	*fgt*	$\Delta perf$	*mem*
No compression	9.18	29.45	335.5M	4.33	22.73	671.0 M	0.77	22.06	1.3 G	0.51	15.44	2.7 G
Spatial	18.15	31.45	5.1 M	9.78	23.58	10.2 M	7.75	16.31	20.4 M	4.30	12.53	41.0 M
Channel	18.00	18.25	0.2 M	21.38	33.22	0.3 M	6.83	28.40	0.6 M	6.47	28.28	1.3 M
Spatial + Channel	12.43	25.97	5.4 M	6.25	19.32	10.8 M	4.29	15.62	21.6 M	2.18	10.80	43.3 M

The results on Split-CIFAR and Split-miniImageNet are shown in Table 3. We found Compressed Activation Replay achieves significant improvement in CIFAR in terms of both final accuracy and forgetting rate. For Split-miniImageNet, our method outperforms all the methods significantly in the small memory case, and performs comparably to HAL in large memory case.

6.3 DISCUSSION

Compression techniques. First, we compare the performance of different compression techniques used in CAR framework. Let the dimension of feature tensor be ($n_f \times w \times h$). We employ the following simple compression schemes: (a) *Spatial*: average pooling along the spatial dimension (dimension n_f); (b) *Channel*: average pooling along the channel dimension (dimension wh); (c) *Spatial + Channel*: a concatenation of spatial and channel pooling (dimension $n_f + wh$).

In addition to performance, we report in Table. 4 the memory overhead needed for storing the samples in the replay buffer. With significantly lower memory overhead, compression techniques perform comparably to those without compression. Among the compression methods, channel pooling performs poorly compared to spatial pooling. Spatial+Channel pooling achieves the best result.

Regularization functions. We compare different regularization functions ℓ_{fm} for feature matching. Natural choices are ℓ_2 and ℓ_1 distances. Besides that we also experiment with weighted version of ℓ_1 and ℓ_2 where the weights are computed by accumulating the gradients, since neurons with high magnitude of changes are often important to the corresponding tasks. In addition, we also tried the MMD loss (Gretton et al., 2007), a popular distribution matching distance. Interestingly, none of them is significantly better than the rest. Detailed results can be found in Appendix.

7 CONCLUSION

We studied the large scale continual learning using memory replay based methods on a non-conventional diverse set of tasks. We compared the performance of different methods in both Taskonomy and medium-scale CIFAR/miniImageNet datasets and found that vanilla experience replay is a strong baseline algorithm despite its simplicity. However, feature drift is observed while training with a standard replay memory. We further proposed the Compressed Activation Replay framework, which aims to reduce such drifting. Experimental results show that our approach achieves the state-of-the-art performance on both large scale Taskonomy benchmark and standard medium-size continual learning datasets.

REFERENCES

Tameem Adel, Han Zhao, and R. Turner. Continual learning with adaptive weights (claw). *ArXiv*, abs/1911.09514, 2020.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, pp. 11816–11825, 2019.

Zalán Borsos, Mojmír Mutnỳ, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *arXiv preprint arXiv:2006.03875*, 2020.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. *arXiv preprint arXiv:1812.00420*, 2018.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Arslan Chaudhry, Albert Gordo, Puneet K Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. *arXiv preprint arXiv:2002.08165*, 2020.

Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. *Proceedings of Machine Learning and Systems*, pp. 8303–8312, 2020.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, 2020.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pp. 513–520, 2007.

Tyler L. Hayes, Nathan D. Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. *2019 International Conference on Robotics and Automation (ICRA)*, May 2019. doi: 10.1109/icra.2019.8793982. URL http://dx.doi.org/10.1109/ICRA.2019.8793982.

David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. *arXiv preprint arXiv:1802.10269*, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, Mar 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal continual learning has perfect memory and is NP-hard. *arXiv preprint arXiv:2006.05188*, 2020.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. *arXiv preprint arXiv:1904.00310*, 2019.

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

Vincenzo Lomonaco and Davide Maltoni. CORe50: a new dataset and benchmark for continuous object recognition. *arXiv preprint arXiv:1705.03550*, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, pp. 6467–6476, 2017.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:504, 2013.

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, and Hassan Ghasemzadeh. Dropout as an implicit gating mechanism for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 232–233, 2020.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. *arXiv preprint arXiv:2006.06958*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

Jary Pomponi, Simone Scardapane, Vincenzo Lomonaco, and Aurelio Uncini. Efficient continual learning in neural networks with embedding regularization. *Neurocomputing*, 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. doi: 10.1109/cvpr.2017.587. URL http://dx.doi.org/10.1109/CVPR.2017.587.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured Laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, pp. 3738–3748, 2018.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2): 123–146, 1995.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pp. 350–360, 2019.

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.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pp. 2990–2999, 2017.

Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

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):1–14, 2020.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Dong Yin, Mehrdad Farajtabar, and Ang Li. SOLA: Continual learning with second-order loss approximation. *arXiv preprint arXiv:2006.10974*, 2020.

Jaehong Yoon, Eunho Yang, Jungtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*. ICLR, 2018.

Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3712–3722, 2018.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3987–3995. JMLR, 2017.

Jeffrey O. Zhang, Alexander Sax, Amir Zamir, Leonidas J. Guibas, and Jitendra Malik. Side-Tuning: Network adaptation via additive side networks. In *European Conference on Computer Vision (ECCV)*, 2020.

A   REGULARIZATION FUNCTIONS

In this experiment, we perform an ablation study to understand the effect of regularization functions $\ell_{fm}(\cdot)$. We use the following loss functions.

1. ℓ_1: Measures the ℓ_1 distance between the feature maps.

2. ℓ_2: Measures the ℓ_2 distance between the feature maps.

3. $\ell_1 + \ell_2$: Use a combination of ℓ_1 and ℓ_2 losses.

4. Weighted ℓ_1: We compute a weighted ℓ_1 loss, where weights are computed using averaged gradients of loss with respect to embeddings i.e., $\nabla\ell_\mathbf{f}$. Larger the gradient magnitudes, more influential are the feature components. We would like to point out that this loss is similar to the Fisher information used in EWC Kirkpatrick et al. (2017).

5. Weighted ℓ_2: Weighted version of ℓ_2 loss.

6. MMD: Use of MMD loss for measuring the distributional distance between the input feature maps.

The results of using these different regularization functions on Split-CIFAR and Split-miniImageNet datasets are shown in Table. 5. We observe that all techniques except MMD perform comparable to each other. So, for the ease of implementation, we use ℓ_2 in all our experiments. MMD results in low performance.

Table 5: **Distribution Matching Techniques** Average accuracy (in %) and forgetting values are reported. All results are averaged over 5 random seeds.

| Method | Split-CIFAR | | | | Split-miniImageNet | | | |
| | $m = 85$ | | $m = 425$ | | $m = 85$ | | $m = 425$ | |
	acc	fgt	acc	fgt	acc	fgt	acc	fgt
ℓ_1	62.4	0.07	66.1	0.04	54.1	0.07	55.4	0.07
ℓ_2	62.4	0.07	66.4	0.04	53.8	0.06	56.3	0.06
$\ell_1 + \ell_2$	62.0	0.07	66.1	0.04	53.5	0.07	56.4	0.06
Weighted ℓ_1	63.2	0.06	67.1	0.04	54.6	0.06	55.8	0.07
Weighted ℓ_2	61.5	0.07	67.0	0.04	53.8	0.06	56.3	0.07
MMD	56.4	0.11	63.2	0.06	51.2	0.08	55.0	0.08

B   INTERMEDIATE LAYER ACTIVATION MATCHING

The compressed activation matching framework performs activation mapping of feature representations. In all experiments reported in the paper, we used the output of encoder for this task. What happens if we use the representations of the previous layers? To study this, we perform compressed activation matching on intermediate representations of the encoder in addition to the final embedding layer. More concretely, we take the outputs of each of the 4 Resnet blocks of Resnet-50 network, perform feature compression and concatenate to obtain a single representation. We then use this concatenated representation to perform feature matching. The results of intermediate feature matching are shown in Table. 6. We observe that we do not obtain any gains from the intermediate feature matching.

C   SENSITIVITY OF λ_{fm}

Effect of varying λ_{fm} is shown in Tables.7, 8. We observe that as λ_{fm} increases, forgetting consistently falls. This is because higher values of λ_{fm} will prevent the features from drifting. However, very high values of λ_{fm} would regularize the models severely and impede learning. There exist an optimal λ_{fm} that achieves the best performance while reducing forgetting.

Table 6: **Effect of using intermediate representations:** Forgetting (*fgt*) and performance drop ($\Delta perf$) values are reported. All results are averaged over 3 random seeds.

Method	$m = 64$		$m = 128$		$m = 256$		$m = 512$	
	fgt ($\downarrow$)	$\Delta perf$ ($\downarrow$)	*fgt* ($\downarrow$)	$\Delta perf$ ($\downarrow$)	*fgt* ($\downarrow$)	$\Delta perf$ ($\downarrow$)	*fgt* ($\downarrow$)	$\Delta perf$ ($\downarrow$)
CAR	13.37	23.25	4.80	17.32	2.98	12.43	1.91	11.51
CAR (*intermediate*)	15.44	22.17	8.75	19.01	3.53	14.84	0.58	13.59

Table 7: **Sensitivity of λ_{fm}:** Forgetting (*fgt*) and performance drop ($\Delta perf$) values are reported. All results are averaged over 3 random seeds.

Method	$m = 64$		$m = 128$		$m = 256$		$m = 512$	
	fgt ($\downarrow$)	$\Delta perf$ ($\downarrow$)	Δfgt ($\downarrow$)	$\Delta perf$ ($\downarrow$)	*fgt* ($\downarrow$)	$\Delta perf$ ($\downarrow$)	*fgt* ($\downarrow$)	$\Delta perf$ ($\downarrow$)
$\lambda_{fm} = 0.25$	32.57	25.36	24.09	19.06	15.79	11.49	9.06	13.84
$\lambda_{fm} = 1.0$	28.49	25.33	20.45	20.91	11.24	17.83	7.09	12.89
$\lambda_{fm} = 5.0$	25.19	29.06	12.01	18.46	8.26	17.03	2.18	10.44
$\lambda_{fm} = 20.0$	18.21	24.72	10.09	18.86	4.29	12.43	-0.29	14.94
$\lambda_{fm} = 50.0$	13.37	23.25	6.25	19.32	2.13	20.17	-1.65	17.29

D   ADDITIONAL RESULTS

In Table 9, we provide additional results comparing CAR and the baseline algorithms on Split-CIFAR and Split-miniImageNet with memory size $m = 255$.

E   EXPERIMENT DETAILS

E.1   TASKONOMY

E.1.1   DATASET:

We use data from 54 buildings of the Taskonomy dataset in our experiments. Of these buildings, 6 randomly chosen buildings are used as training set, whiile the data from the other 48 buildings are used for training and validation. The details of the buildings used in training and test set are provided below:

Training set: *cutlerville, rockport, coffeen, tomales, pinesdale, capistrano, bettendorf, mcnary, barboursville, hacienda, uvalda, tolstoy, wyatt, haymarket, holcut, kingdom, windhorst, bonfield, monson, touhy, marksville, oyens, euharlee, glassboro, brentsville, goodview, darrtown, maben, pomaria, keiser, gloria, silva, portal, thrall, smoketown, martinville, waldenburg, lakeville, springerville, silerton, mifflintown, wainscott, superior, stokes, mayesville, wilkinsburg, archer*

Test set: *moberly, northgate, adairsville, cisne, marstons, woonsocket*

Data processing: For all pixel-wise tasks, we normalize the inputs in the range $[-1, 1]$. For depth maps, we perform a log normalization similar to Zamir et al. (2018). We resize all inputs and output maps to (256×256) resolution, and perform mean normalization and random horizontal flipping as data augmentation.

E.1.2   TRAINING

A list of model architectures and optimization algorithm we use is stated in Table. 10. Models were optimized using distributed training with a global batch size of 64. During sequential training, we train each task for $12.5k$ iterations. Multitask model was trained for $37.5k$ iterations.

Table 8: **Sensitivity of λ_{fm}:** Average accuracy (in %) and forgetting values are reported. All results are averaged over 5 random seeds.

Method	Split-CIFAR				Split-miniImagenet			
	$m = 85$		$m = 425$		$m = 85$		$m = 425$	
	acc	fgt	acc	fgt	acc	fgt	acc	fgt
$\lambda_{fm} = 0.25$	59.0	0.09	64.4	0.06	50.9	0.08	53.6	0.08
$\lambda_{fm} = 1.0$	61.4	0.07	65.3	0.05	53.3	0.07	54.2	0.07
$\lambda_{fm} = 5.0$	62.4	0.06	66.5	0.04	54.2	0.06	56.2	0.06
$\lambda_{fm} = 10.0$	62.0	0.06	66.0	0.03	53.3	0.06	53.4	0.07
$\lambda_{fm} = 25.0$	62.2	0.06	63.3	0.03	52.2	0.07	51.5	0.05
$\lambda_{fm} = 100.0$	58.6	0.04	55.8	0.02	46.1	0.04	42.3	0.03

Table 9: **Split-CIFAR and Split-miniImageNet Results:** Average accuracy (acc±std %) and forgetting (±std) values are reported. Results are averaged over 5 random seeds. The best result is highlighted in bold.

Method	SPLIT-CIFAR		SPLIT-MINIIMAGENET	
	$m = 255$		$m = 255$	
	acc($\uparrow$)	fgt($\downarrow$)	acc($\uparrow$)	fgt($\downarrow$)
iCaRL	51.7±1.4	0.13±0.02	-	-
A-GEM	56.9±3.4	0.13±0.03	51.6±2.6	0.10±0.02
MER	57.7±2.6	0.11±0.01	49.4±3.4	0.12±0.02
ER	60.9±1.4	0.09±0.01	53.5±1.4	0.07±0.02
HAL	62.9±1.5	0.08±0.01	**56.5±0.8**	**0.06±0.01**
CAR (ours)	**65.2±2.1**	**0.05±0.01**	55.1±2.1	**0.06±0.01**

Experience Replay: Since the encoder-decoder models used in Taskonomy dataset are very deep, optimizing the encoders and decoders using equation 3 can easily run out of memory. Therefore, instead of jointly minimizing the losses, at each step, we replay one of the old tasks with a replay probability p_{replay}, and train the current task with a probability $(1 - p_{replay})$. This ensures that at each step, only one model is optimized thereby preventing the out-of-memory issues.

E.2 SPLIT-CIFAR / SPLIT-MINIIMAGENET

For Split-CIFAR and Split-miniImageNet experiments, we follow the protocol used in Chaudhry et al. (2018) where the 100 classes are split into 20 tasks, yielding 5 classes per task. So, each task is a 5-way classification problem. Of these, we use 3 classes for validation purposes and report performance on the remaining 17 tasks. Each experiment is repeated for 5 random seeds.

CAR: The output of Resnet-18 encoder is a 160-dimensional feature vector. Since the feature dimension is orders of magnitude smaller than the data dimension (3072 for Split-CIFAR and 21168 for Split-miniImageNet), storing the entire feature has very small memory overhead. Hence, we do not perform any feature compression for these experiments.

Table 10: **Experiment details on Taskonomy dataset**

Encoder Architecture	Resnet-50
Decoder Architecture	15-layer ConvNet for pixelwise tasks
	Average pooling + 1 FC layer for classification
Encoder Learning rate	0.0001
Decoder Learning rate	0.0001
Optimizer	Adam
LR Decay	None
Batch Size	64
Effective number of training iters	12500 per task
Experience Replay	
Buffer filling strategy	Uniform Sampling
Replay batch size	64
Replay probability p_{replay}	0.5
CAR	
Regularization function ℓ_{fm}	MSE loss
Compression scheme	Spatial + channel pooling
Replay probability p_{replay}	0.5
Side Tuning	
Side Network Architecture	5-layer ConvNet
Side Network Learning rate	0.0001

Table 11: **Experiment details on Split-CIFAR / Split-miniImageNet dataset**

Encoder Architecture	Resnet-18 used in Chaudhry et al. (2018)
Task head	FC layer
Encoder Learning rate	0.03
Task head rate	0.03
Optimizer	SGD
LR Decay	None
Batch Size	10
Number of training iterations	Single epoch training
Experience Replay	
Buffer filling strategy	Uniform Sampling
Replay batch size	5
CAR	
Regularization function ℓ_{fm}	MSE loss
Compression scheme	No compression used
λ_{fm}	5.0