SATHUR: Self Augmenting Task Hallucinal Unified Representation for Generalized Class Incremental Learning

Sathursan Kanagarajah Thanuja Ambegoda Ranga Rodrigo
University of Moratuwa, Sri Lanka
ksathursan1408@gmail.com, thanujaa@uom.lk, ranga@uom.lk

Abstract

Class Incremental Learning (CIL) is inspired by the human ability to learn new classes without forgetting previous ones. CIL becomes more challenging in real-world scenarios when the samples in each incremental step are imbalanced. This creates another branch of problem, called Generalized Class Incremental Learning (GCIL) where each incremental step is structured more realistically. Grow When Required (GWR) network, a type of Self-Organizing Map (SOM), dynamically creates and removes nodes and edges for adaptive learning. GWR performs incremental learning from feature vectors extracted by a Convolutional Neural Network (CNN), which acts as a feature extractor. The inherent ability of GWR to form distinct clusters, each corresponding to a class in the feature vector space, regardless of the order of samples or class imbalances, is well suited to achieving GCIL. To enhance GWR's classification performance, a high-quality feature extractor is required. However, when the convolutional layers are adapted at each incremental step, the GWR nodes corresponding to prior knowledge are subjected to near-invalidation. This work introduces the Self Augmenting Task Hallucinal Unified Representation (SATHUR)*, which re-initializes the GWR network at each incremental step, aligning it with the incrementally updated feature extractor. Comprehensive experimental results demonstrate that our proposed method significantly outperforms other state-of-the-art GCIL methods on CIFAR-100 and CORe50 datasets.*

1. Introduction

Humans and animals have an astonishing ability to continually acquire, process, and update knowledge throughout their lifetime [27]. The ability to continually learn over time by accumulating new knowledge while retaining and utilizing previously learned knowledge is referred to as Incremental Learning (IL) [3]. Convolutional Neural Networks (CNN) have achieved expert-level performances in various computer vision problems. In some challenges like image classification, their performance has even surpassed that of humans. However, when CNN models attempt to learn tasks incrementally, they partially or completely forget the previously learned knowledge, a phenomenon termed as catastrophic forgetting [7, 24, 25].

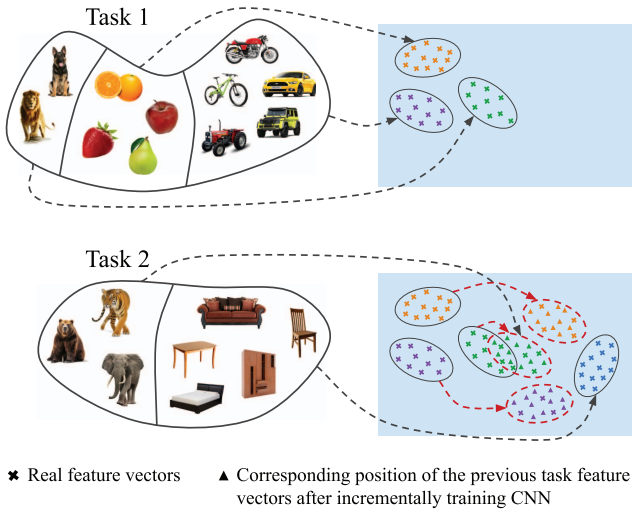


Figure 1. Feature transformation due to incremental training of the CNN: The top row indicates the feature vectors for each class in the feature space after learning Task 1. When the feature extractor (CNN) is trained on Task 2, the previous clusters become invalidated. The feature vectors in the Task 1 clusters need to be transformed to their corresponding new positions according to the current (updated) feature extractor.

Class Incremental Learning (CIL) [2, 12, 29, 42] is a widely studied setting in IL, particularly in image classification problems. CIL requires the learning of new classes for each task, where a task is defined as a group of classes. There are certain rules that must be followed while defining a task. (i) every task should consist of the same number of classes (ii) there should not be any overlap between the classes in the tasks (i.e., if a class is present in task 1, it should not be present in any of the future tasks) (iii) training samples are well-balanced across different classes.

However, this is not how humans and animals learn. They learn from imbalanced data, which is more realistic: each task does not necessarily consist of the same number of classes, tasks are intermixed in terms of classes, and class imbalance is prominent. In real scenarios, such as when a robot or a system is deployed into a real-world environment, the above three rules do not hold [36]. This paradigm creates another branch of the problem, called Generalized Class Incremental Learning (GCIL) [26], where each task is characterized by the following three quantities: the number of classes appearing in each task, the specific classes appearing in each task, and the number of samples per class. In the GCIL setting, these quantities are sampled from probabilistic distributions [26]. Hence, more diverse and realistic scenarios can be created by varying these distributions. This setup more closely resembles human learning.

Along with the challenge of catastrophic forgetting in previous CIL settings, GCIL has two other challenges, which are class imbalance and sample efficiency [26]. The Grow When Required (GWR) [23] network, a Self-Organizing Map (SOM), learns a cluster by dynamically creating and removing nodes and edges when required. For each input feature vector, only the nodes in the neighborhood of that feature vector are trained according to a learning rule. This approach naturally aligns with class imbalance, sample efficiency, and the non-deterministic ordering of training samples in each task in the GCIL setting. A CNN acts as a feature extractor, creating a feature vector for each image input. These feature vectors are then used to train the GWR network to form clusters. To enhance the GWR's classification performance, the feature extractor should be capable of extracting class-specific features, so the feature extractor has to be trained incrementally. However, when the convolutional layers are adapted at each incremental step using a CIL method, the GWR nodes corresponding to prior knowledge may become nearly invalidated (Fig. 1).

We have proposed the *Self Augmenting Task Hallucinal Unified Representation (SATHUR)*, which re-initializes the GWR network at each incremental step, aligning it with the current feature extractor. SATHUR takes the previous GWR nodes and exemplar feature vectors, which are extracted from exemplar samples by the current feature extractor, as input. It then trains a hallucinator to create a set of augmented feature vectors. The GWR network is subsequently re-initialized by training on these augmented feature vectors. The re-initialized GWR network is then trained on a mixture of feature vectors that are extracted from new task data and exemplar data. We have validated our approach using the CIFAR-100 and CORe50 datasets. The usage of our method, in conjunction with state-of-the-art replay-based CIL methods, improves accuracy by a significant margin. Specifically, it surpasses the results by 3.70% on CIFAR-100 and by 2.88% on CORe50.

2. Related Work

Class Incremental Learning. In general, two groups of incremental training protocols are considered in the current CIL literature. The first is multi-epoch CIL, where new tasks, consisting of new classes or patterns, arrive incrementally, and only data in the current task is available for model training. During training, data of each task pass through multiple epochs. The second is online CIL. In this case, although training data still arrives sequentially, this setup only allows the model to be trained on each sample once [9, 10].

We focus only on the multi-epoch CIL setting and refer to it as CIL in the rest of the paper. The main problem of CIL is catastrophic forgetting, where learning a new task leads to degradation of performance related to previously learned tasks. There are three main approaches to mitigate catastrophic forgetting in CIL [22]: (i) Regularization-based methods, (ii) Parameter-isolation-based methods (iii) Replay-based methods.

Regularization-based methods incorporate penalization terms into their objective functions to address discrepancies. These discrepancies typically exist between old and new models. This is often achieved by establishing comparisons across various elements. These elements include output logits [16, 29], intermediate features [6, 12, 17, 32], and prediction heatmaps [5].

Parameter-isolation-based methods aim to increase the trainable model parameters at each incremental step, effectively counteracting the problem often seen with parameter overwriting. There are two primary strategies within this category. One approach involves gradually increasing the neural network's size to accommodate incoming data [13, 31, 33, 38, 39]. The second strategy involves freezing a section of network parameters, ensuring that prior class knowledge remains preserved [1, 14, 18, 40].

Replay-based methods are based on the idea that there is a specific, yet small, memory allowance for keeping a few old-class exemplars in memory compared to the new class data. These exemplars can be used to re-train the model in each new incremental step [6, 12, 19, 20, 29, 34, 37]. This re-training process usually has two parts: the first part involves training the model on all new class data and the small amount of old class exemplars, while the second part fine-tunes the model using a balanced subset with an equal number of samples from each class [6, 12, 18, 19, 39].

Generalized Class Incremental Learning. GCIL was introduced by Fei Mi *et al.* [26]. They used a replay-based method [29] in combination with mixup [41], a method they referred to as ReMix [26], to address the challenges associated with GCIL. The herding technique was used to select exemplars from different classes during each incremental task training. Subsequently, mixup was applied to the mini-batches containing samples from both the current task and the exemplars. Mixup creates virtual training samples

through the linear interpolation of raw training samples.

Grow When Required network. GWR [23] network dynamically learns clusters by adjusting nodes and edges for each feature vector, as detailed in Sec. 3.2. It naturally forms distinct clusters ideal for GCIL, irrespective of sample order or class imbalances. Although a superior feature extractor boosts the performance of GWR, incrementally adapting convolutional layers risks invalidating the GWR nodes that represent prior knowledge.

3. Methods

3.1. Problem Definition

As proposed in Mi *et al.* [26], GCIL is formulated by structuring every task from a task-dependent distribution. We denote the complete set of available classes as $\mathcal{S}$ with size n. Given a sequence of tasks, C_t is the set of samples that are present in the task t, and it is sampled from a task-dependent distribution $\mathcal{H}(t)$.

$$C_t \sim \mathcal{H}(t) \tag{1}$$

A probabilistic formulation of $\mathcal{H}(t)$ can be formed through three steps.

$$K_t \sim \mathcal{D}(t) \tag{2}$$

The number of classes K_t to appear in task t follows a task-dependent discrete distribution $\mathcal{D}(t)$. Different scenarios regarding the number of appearing classes in each task can be simulated through different choices of $\mathcal{D}(t)$.

$$S_t \sim \mathcal{R}(W_t^1, K_t) \tag{3}$$

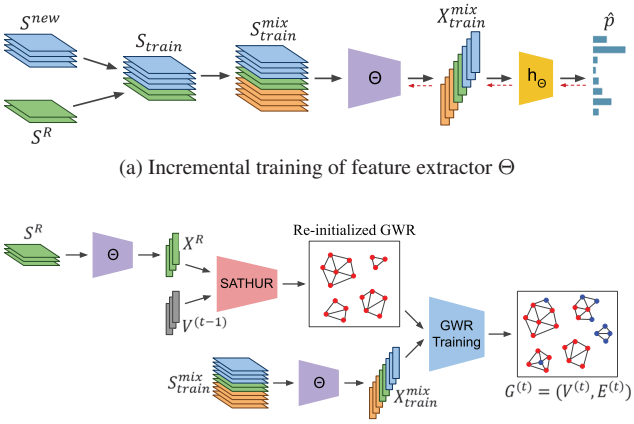
Classes appearing in task t are modeled as a random vector $S_t \in \mathbb{R}^n$. S_t is a binary indicator vector with ones corresponding to classes appearing in t. $\mathcal{R}$ depends on the class number K_t and a class appearance weight vector $W_t^1 \in \mathbb{R}^n$. Each entry of W_t^1 represents the appearing probability of the class in task t. Classes with larger weights are more likely to appear in the task.

$$C_t \sim \mathcal{M}(W_t^2, S_t) \tag{4}$$

Eq. (4) is used to determine the sample size of each appearing class in S_t, which is encoded as random vector C_t. C_t follows a distribution, which depends on the appearing class S_t and a class sample size weight vector $W_t^2 \in \mathbb{R}^n$. W_t^2 the sample size of each class appearing in task t, and it can model different degrees of class imbalance within a task. In Sec. 4.3, we outline our choices for the distributions $\mathcal{D}(t)$, W_t^1 and W_t^2. Learning in the presence of this realistic presentation of classes is the GCIL problem.

3.2. Proposed Solution

We propose SATHUR, an effective method that can incrementally re-initialize the GWR network according to the updated feature extractor and train GWR on the new task. Our method can be used as a plugin method along with existing replay-based CIL methods.



(a) Incremental training of feature extractor Θ

(b) Re-initializing GWR using SATHUR and training on new task data

Figure 2. GWR-based GCIL enabled by SATHUR. (a) Training set S^{train} incorporates new training samples S^{new} and exemplars S^R. S^{mix}_{train} is a balanced training set formed from S_{train} by adding augmented samples generated through mixup. The feature extractor Θ is then trained using S^{mix}_{train}. (b) Exemplar features X_R are extracted from exemplar samples S^R using updated Θ as the feature extractor. By using X^R and previous task GWR nodes $V^{(t-1)}$ as the inputs to the SATHUR, GWR is re-initialized. Updated feature extractor Θ is used to extract the features X^{mix}_{train} from S^{mix}_{train}. Re-initialized GWR is trained on X^{mix}_{train} to learn the unified representation. Red nodes represent re-initiated nodes before training on new task data, and blues nodes represent the nodes that are created during new task training.

Incremental training of feature extractor. At each incremental training step, new task data is combined with the exemplars to form a unified training set. The mixup [41] is applied to the unified training set to generate new augmented samples. The augmented samples are generated by taking convex combinations of pairs of inputs and their labels, which create more diverse samples for underrepresented classes by mixing them with over-represented ones. An augmented training sample $(\bar{x}, \bar{y})$ is generated by linear interpolation between raw training samples (x_i, y_i) and (x_j, y_j) as shown in Eq. (5). This increase in samples assists the model in learning better representations of under-represented classes, while facilitating the learning of smooth decision boundaries between all classes.

$$\bar{x} = \lambda x_i + (1-\lambda)x_j,\ \bar{y} = \lambda y_i + (1-\lambda)y_j \tag{5}$$

where $\lambda \sim Beta(\alpha, \alpha)$, with hyperparameter $\alpha \in (0, \infty)$

As illustrated in Fig. 2a, we combine the samples corresponding to the new task S^{new} and exemplars S^R to create a unified training set S_{train}. We apply mixup on S_{train} to create balanced training set S^{mix}_{train}, which is then used to

train the feature extractor, Θ. After training, Θ is able to extract the important high-level features corresponding to S^{mix}_{train}.

GWR-based GCIL. GWR networks are suitable for learning imbalanced feature distributions in the GCIL setting due to their ability to introduce new neurons when it is required to match new inputs.

GWR [23] network is initialized by randomly selecting two feature vectors as inputs. The weights (w) and labels (l) of the nodes correspond to those of the feature vectors. The best matching unit (s) is the node that most closely resembles the input. For each input, an edge connection is generated between the best matching unit and the second-best matching unit (t). The GWR network adds new nodes based on two conditions. First, the activity (a) of the best matching node, which is inversely related to the Euclidean distance from the input, should fall below a certain activity threshold (a_T). This indicates that the current best matching node does not closely resemble the input. Secondly, the node's firing counter (h), must also fall below a certain firing threshold (h_T).

$$h_s(t) = h_0 - \frac{1}{\alpha_b}(1 - e^{-\alpha_b t/\tau_b}) \tag{6}$$

$$h_n(t) = h_0 - \frac{1}{\alpha_n}(1 - e^{-\alpha_n t/\tau_n}) \tag{7}$$

where h_0 is the initial firing counter value. α_b, α_n and τ_b, τ_n are the constants controlling behaviour of the curve. Whenever the criterion for adding a new node is not met, the weights of the best matching node and its directly connected neighboring nodes are adjusted to learn the new input. ϵ_b and ϵ_n are learning rates for the best matching node and neighboring nodes, respectively. The firing counter of the best matching node $h_s(t)$ is updated using Eq. (6). The firing counters of directly connected neighboring nodes to the best matching node $h_n(t)$ are updated using Eq. (7). The edge connections have an associated *age*, which is initially set to zero and is incremented at each time step for each edge connected to the best matching node. The only exception is the edge that links the best matching and second-best matching units, whose *age* is reset to zero. Edges that exceed a certain age threshold, age_{max}, are removed. Any node without direct neighbors, i.e., without edge connections, is removed.

During the inference phase of GWR, the nearest neighbors to the input feature vector are determined based on Euclidean distance. The label of the input feature vector is then predicted as the most common label among these nearest neighbors.

Re-initializing GWR using SATHUR. Drawing inspiration from [35], which enhances low-shot learning through learned hallucinations, we have developed a novel methodology, SATHUR. SATHUR is used to re-initialize the previous task GWR network $G^{(t-1)}$, adapting to the updated feature extractor Θ. As shown in Fig. 3, at every training iteration we choose m samples from previous task GWR nodes $V^{(t-1)}$ and form a node batch. Exemplar features X^R are extracted using updated feature extractor Θ. We form an ex-

Algorithm 1: GWR training

Initialize the network by randomly selecting two inputs to form nodes with corresponding labels.
for each input feature x_i:
  for each node i in the network: calculate $\|x_i - w_i\|$
  Select nodes s, t; $s = argmin_{v \in \mathcal{V}}\{\|x_i - w_i\|\}$ and $t = argmin_{v \in \mathcal{V} \setminus \{s\}}\{\|x_i - w_i\|\}$
  if (s,t) not in $\mathcal{E}$: $\mathcal{E} = \mathcal{E} \cup \{(s,t)\}$
  else: $age_{(s,t)} = 0$
  Calculate activity for s; $a_s = exp(-\|x_i - w_s\|)$
  if $a_s < a_T$ and $h_s < h_T$:
    Add new node r; $w_r = (w_s + x_i)/2$ and $l_r = l_i$
    $\mathcal{E} = \mathcal{E} \cup \{(r,s),(r,t)\}$
    $\mathcal{E} = \mathcal{E} \setminus \{(s,t)\}$
  else:
    Adapt the positions of the s and its neighbors; $\Delta w_s = \epsilon_b \times h_s \times (x_i - w_s)$ $\Delta w_j = \epsilon_n \times h_j \times (x_i - w_j)$
    for all edges ending at s: $age_{(s,q)} = age_{(s,q)} + 1$
    Update the h_s using Eq. (6)
    Update the h_j of the neighbors of s using Eq. (7)
  for all $(p,q) \in \mathcal{E}$:
    if $age_{(p,q)} > age_{max}$: $\mathcal{E} = \mathcal{E} \setminus \{(p,q)\}$
  for all $v \in \mathcal{V}$:
    if no edge end at v: $\mathcal{V} = \mathcal{V} \setminus \{v\}$

Algorithm 2: SATHUR (task $t, t \geq 2$)

Input : GWR nodes $V^{(t-1)}$ of task $(t-1)$
Exemplar features X^R extracted by updated Θ
Output: Re-initialized GWR, $G^{(t-1)*} = (V^{(t-1)*}, E^{(t-1)*})$

for *epochs* **do**:
  $V_j^{(t-1)} \leftarrow$ choose m samples from $V^{(t-1)}$
  for every $V_{j,k}^{(t-1)} \in V_j^{(t-1)}$:
    Select $X_{j,k}^R$ from X^R; same class as $V_{j,k}^{(t-1)}$
  $X_j^R \leftarrow$ all selected exemplar samples $X_{j,k}^R$
  $X^{aug} = P^2(P^1(V_j^{(t-1)}) + X_j^R)$; $P = P^1 \cup P^2$
  $X^{aug}_{train} = X^R \cup X^{aug}$
  Train hallucinator P, using classifier h_P
Using updated hallucinator P, create X^{aug}_{train}
Train a GWR network on X^{aug}_{train} using Algorithm 1

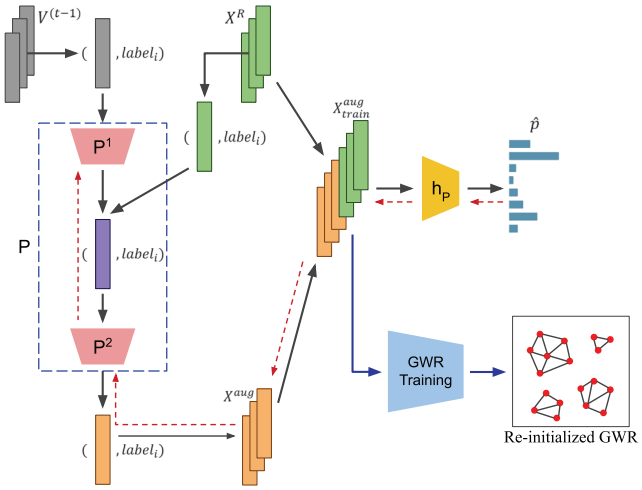


Figure 3. Re-initializing GWR using SATHUR: Following the incremental training of feature extractor Θ, previous task GWR nodes $V^{(t-1)}$ and exemplar features X^R from the same class are sampled. Sampled previous task GWR nodes are passed into P^1 and the output is combined with the corresponding class exemplar features and passed into P^2 to obtain augmented features X^{aug}. A unified training set X^{aug}_{train} is created by adding X^{aug} to X^R. The hallucinator P is trained end-to-end along with the classification algorithm h_P. Dotted red arrows indicate the flow of gradients during back-propagation. Once the hallucinator P is trained, then a unified training set X^{aug}_{train} is generated by combining X^R and augmented features X^{aug}, created by the trained hallucinator. GWR is re-initiated by training X^{aug}_{train} on the GWR algorithm. As a result, new nodes are created for previous classes validating the new task feature extractor.

emplar batch X^R_j by choosing samples from X^R, such that for every node in the node batch, there is a corresponding exemplar feature that belongs to the same class. The intermediate output obtained by passing $V^{(t-1)}$ to P^1 is combined with corresponding samples in X^R and passed to P^2 to generate augmented features X^{aug}. Each augmented example is of the form (x', y), where $x' = P(x, v; \mathbf{w}_P)$. Here, (x, y) is a sample from X^R, and (v, y) is the corresponding sample from $V^{(t-1)}$. $\mathbf{w}_P$ represents the parameters of the hallucinator, P. An augmented training set, X^{aug}_{train}, is formed by adding the set of augmented features, X^{aug}, to the set of exemplar features, X^R. The hallucinator P is trained end-to-end along with the classification algorithm h_P. After training the hallucinator P, an augmented training set X^{aug}_{train} is created. X^{aug}_{train}. The GWR network is re-initialized by training with X^{aug}_{train} using Algorithm 1.

4. Experiments

4.1. Baseline Methods

ReMix [26]: Mixup is applied to the new task data and stored exemplars to generate more augmented data, helping to reduce the class imbalance across the training data.

Mnemonics [20]: A bilevel optimization framework was used to distill new class data into exemplars before discarding them. The aim of this method is to improve the quality of exemplars without inflating their number.

MRDC [34]: The aim of this method is to establish a balance between the quality and quantity of exemplars. This was accomplished through image compression, utilizing the JPEG algorithm, which resulted in each exemplar being uniformly downsampled.

GWR [23]: Pre-trained convolutional layers are used as the feature extractor and GWR network is trained only on new task data.

Full: At each task, the model is trained on all the task data that have been arrived at the model. This is a common performance upper bound in CIL.

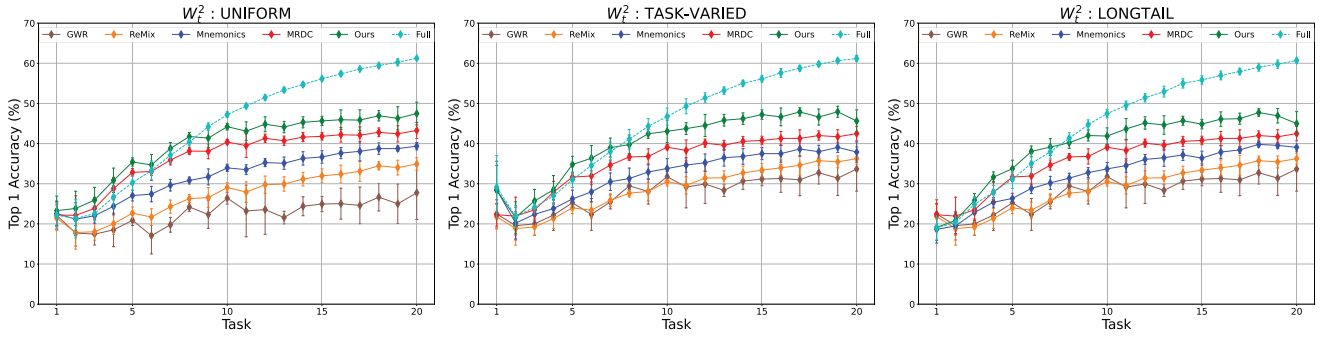
To ensure a fair comparison between the methods, we used the ImageNet pre-trained 32-layer ResNet [11] as the base initialization for the feature extractor Θ. For ReMix, Mnemonics, and MRDC, Θ is incrementally trained using the respective algorithms. In the case of Full, at every task, base initialization is done, and then the model is trained on all the data that the model has encountered up to that particular task. For GWR, feature extractor Θ cannot be trained incrementally without SATHUR. Therefore, we consistently used the ImageNet pre-trained 32-layer ResNet [11] as the feature extractor Θ without training it incrementally.

4.2. Datasets

We compare the model's performance on CIFAR-100 [15] and CORe50 [21] datasets.

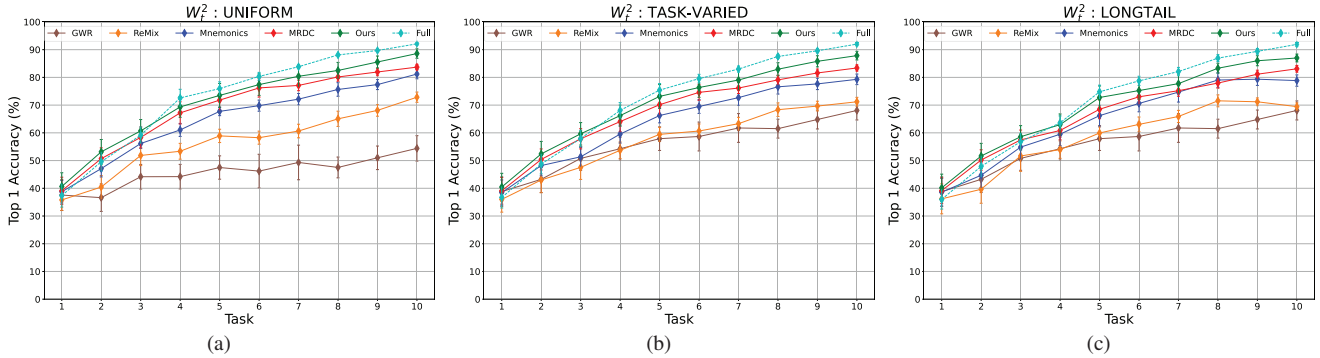
CIFAR-100: Contains 60,000 RGB images, each sized 32×32 pixels, spread across 100 classes. Each class contains 500 training images and 100 testing images. We train 20 tasks incrementally, each containing 1,000 images. In each experiment, 20,000 images are randomly selected from the collection of 60,000 images for training.

CORe50: Contains short 15-second video sequences of an object moving. It has 10 object categories, each with 5 distinct domestic objects, recorded under 11 different environmental conditions (e.g., various backgrounds, various illuminations, outdoors/ indoors, etc.), 8 for training, and 3 for testing. The videos were originally recorded at 20 fps, but we sample them at 2 fps. Due to the smoothness of the videos, down-sampling the video frame rate is a common practice with CORe50 [8, 9]. The training set contains 12,000 RGB images, and the testing set contains 4,500 images, each sized 128×128 pixels, spread across 10 classes. We incrementally train 10 tasks with 600 images in each task. In each experiment, 6,000 images are randomly selected from the collection of 12,000 images for training.



(a) (b) (c)

Figure 4. Performances with varying W_t^2 on CIFAR-100: At each incremental training task, mean top-1 accuracy and standard deviation over 5 runs with different random seeds are plotted. Our method significantly outperforms existing methods.



(a) (b) (c)

Figure 5. Performances with varying W_t^2 on CORe50: At each incremental training task, mean top-1 accuracy and standard deviation over 5 runs with different random seeds are plotted. Ours method significantly outperforms existing methods and reaches very close to the upper limit "Full".

4.3. Implementation Details

Feature extractor Θ. At each incremental task training, a 32-layer ResNet [11] is used as the feature extractor Θ, and it is trained by stochastic gradient descent with 60 epochs. α of ReMix is set to 1.2. The learning rate starts from 0.1 and is divided by 10 after 40 and 50 epochs; weight decay is 1e-3 and the momentum is 0.9.

SATHUR. For our hallucinator P, we use a two layer MLP for P^1 and three layer MLP for P^2 with ReLU as the activation function. P is trained by stochastic gradient descent with 100 epochs. The learning rate starts from 0.1 and is divided by 10 after 60 and 80 epochs; weight decay is 1e-3 and momentum is 0.9. For the GWR network activity threshold and firing threshold are set to 0.65 and 0.11, respectively.

We evaluate our method on benchmarks as done in ReMix [26]. For CIFAR-100 dataset, we set $\mathcal{D}(t)$ as a uniform distribution $\mathcal{U}(1, 100)$, and W_t^1 as a uniform distribution over all classes. For CORe50 dataset, we set $\mathcal{D}(t)$ as a uniform distribution $\mathcal{U}(1, 10)$, and W_t^1 as a uniform distribution over all classes. Three variations of W_t^2 are tested;
UNIFORM: W_t^2 is a fixed uniform distribution over all classes in S.
TASK-VARIED: W_t^2 varies across different tasks by adding independent Gaussian noises (0 mean and 20% of uniform class weight as the standard deviation) to each class weight of UNIFORM.
LONGTAIL: W_t^2 is a fixed long-tailed distribution. The weight $W_{t,i}^2$ for class i in the long-tailed distribution is generated by an exponential function $W_{t,i}^2 = \mu^i$ [4]. Different μ's correspond to different degrees of class imbalance. In our setting, the largest weight is 5 times larger than the smallest.

Models are evaluated by top-1 accuracy on the balanced test set consisting of all classes that appeared so far.

5. Results

Figure 4 and Figure 5 show the performance of the methods, with the CIFAR-100 and CORe50 datasets respectively, measured by mean top-1 accuracy and standard deviation, based on five experimental runs each. We compare (see Sec. 4.1) ReMix, Mnemonics, MRDC, GWR, and Full with our method. Our method outperforms the state-of-the-art methods by significant margins in different GCIL

Method	CIFAR-100	CORe50
GWR	27.80	55.98
ReMix	28.76	58.26
Mnemonics	32.25	64.64
MRDC	35.72	66.66
Ours (MRDC+GWR+SATHUR)	**39.42**	**69.54**

Table 1. Top-1 accuracy averaged over 20 tasks for CIFAR-100 and 10 tasks for CORe50 when W_t^2 = LONGTAIL. The contribution from SATHUR is essential for the good results achieved.

setups. When W_t^2 = LONGTAIL, it surpasses the state-of-the-art method, MRDC, on CIFAR-100 and CORe50 by 3.70% and 2.88%, respectively. During early incremental tasks, our method performs better than "Full". This is because SATHUR is creating augmented features that are so close to real features.

5.1. Ablation study

In Tab. 1, our approach is evaluated against GWR and MRDC. The fact that training GWR using the features extracted from the pretrained Θ as the feature extractor fails significantly shows that Θ needs to be optimally trained at each incremental task. MRDC exhibits good performance in the GCIL setting compared to Mnemonics and ReMix, by maintaining a higher number of compressed training samples within the memory buffer. Therefore, we used MRDC in conjunction with mixup to train Θ at each incremental task. Hence, the use of SATHUR is crucial to transform the previously generated nodes across incremental tasks. Our method outperformed GWR by a large margin of 11.62% on CIFAR-100 and 13.56% on CORe50 datasets.

By splitting the allocated memory between exemplar samples and GWR nodes, our method efficiently manages memory resources and can be used as a plugin method with replay-based CIL methods. Future research can be conducted in the area of adaptive memory optimization between exemplar samples and GWR nodes at each incremental step.

As the GWR is adapted only to the local neighborhood that is most similar to the input [23], catastrophic interference to parameters unrelated to the current input is effectively prevented. This focused local adaptation is lightweight in terms of computation and memory requirements, compared to backpropagation-based learning with entire model adaptation [28]. Once the feature extractor is well-trained on all the new classes, GWR can incrementally learn new instances [21] of each class from a very small number of samples without the need for fine-tuning the feature extractor. This makes our approach notably capable of performing collaborative cloud and edge computing [28, 30].

6. Conclusion

Generalized Class Incremental Learning (GCIL) represents a more realistic continual learning paradigm where each incremental task is customized based on probabilistic distributions. Therefore, different realistic scenarios can be simulated by altering these distributions. GWR networks inherently possess the ability to learn distributions by managing class imbalance, sample efficiency, and the non-deterministic ordering of training samples. However, the performance of GWR is constrained by the fixed feature extractor used to extract feature vectors from images. In this work, we propose a method to train the GWR network while incrementally adapting the feature extractor. We introduce a *Self Augmenting Task Hallucination Unified Representation (SATHUR)* to re-initialize the GWR network at each incremental step, thereby adapting to the updated feature extractor. Our method's effectiveness in addressing the GCIL problem is demonstrated by our successful results with the CIFAR-100 and CORe50 datasets. In the context of continual learning, our research expands the capacity to learn from data samples that represent realistic conditions. It facilitates the accumulation of new knowledge while concurrently mitigating the issues of catastrophic forgetting and class imbalance.

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *CVPR*, pages 3931–3940, 2020. 2

[2] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *ICCV*, pages 583–592, 2019. 1

[3] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018. 1

[4] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, pages 9268–9277, 2019. 6

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 2

[6] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, pages 86–102. Springer, 2020. 2

[7] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 1

[8] Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *ICRA*, pages 9769–9776. IEEE, 2019. 5

[9] Tyler L Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *CVPRW*, pages 220–221, 2020. 2, 5

[10] Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. In *CVPR*, pages 13926–13935, 2020. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5, 6

[12] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019. 1, 2

[13] Shenyang Huang, Vincent François-Lavet, and Guillaume Rabusseau. Neural architecture search for class-incremental learning. *arXiv preprint arXiv:1909.06686*, 2019. 2

[14] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 2

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 5

[16] Zhizhong Li and Derek Hoiem. Learning without forgetting. *PAMI*, 40(12):2935–2947, 2017. 2

[17] Yaoyao Liu, Yingying Li, Bernt Schiele, and Qianru Sun. Online hyperparameter optimization for class-incremental learning. *arXiv preprint arXiv:2301.05032*, 2023. 2

[18] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *CVPR*, pages 2544–2553, 2021. 2

[19] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Rmm: Reinforced memory management for class-incremental learning. *NIPS*, 34:3478–3490, 2021. 2

[20] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, pages 12245–12254, 2020. 2, 5

[21] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26. PMLR, 2017. 5, 7

[22] Zilin Luo, Yaoyao Liu, Bernt Schiele, and Qianru Sun. Class-incremental exemplar compression for class-incremental learning. In *CVPR*, pages 11371–11380, 2023. 2

[23] Stephen Marsland, Jonathan Shapiro, and Ulrich Nehmzow. A self-organising network that grows when required. *Neural Networks*, 15(8–9):1041–1058, 2002. 2, 3, 4, 5, 7

[24] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 1

[25] Ken McRae and Phil A Hetherington. Catastrophic interference is eliminated in pretrained networks. In *Proceedings of the 15h Annual Conference of the Cognitive Science Society*, volume 1, page 2, 1993. 1

[26] Fei Mi, Lingjing Kong, Tao Lin, Kaicheng Yu, and Boi Faltings. Generalized class incremental learning. In *CVPRW*, pages 240–241, 2020. 2, 3, 5, 6

[27] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019. 1

[28] Duvindu Piyasena, Miyuru Thathsara, Sathursan Kanagarajah, Siew Kei Lam, and Meiqing Wu. Dynamically growing neural network architecture for lifelong deep learning on the edge. In *FPL*, pages 262–268. IEEE, 2020. 7

[29] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017. 1, 2

[30] Jinke Ren, Guanding Yu, Yinghui He, and Geoffrey Ye Li. Collaborative cloud and edge computing for latency minimization. *IEEE Transactions on Vehicular Technology*, 68(5):5031–5044, 2019. 7

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

[32] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. On learning the geodesic path for incremental learning. In *CVPR*, pages 1591–1600, 2021. 2

[33] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *ECCV*, pages 398–414. Springer, 2022. 2

[34] Liyuan Wang, Xingxing Zhang, Kuo Yang, Longhui Yu, Chongxuan Li, Lanqing Hong, Shifeng Zhang, Zhenguo Li, Yi Zhong, and Jun Zhu. Memory replay with data compression for continual learning. *arXiv preprint arXiv:2202.06592*, 2022. 2, 5

[35] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *CVPR*, pages 7278–7286, 2018. 4

[36] Zhenyu Wen, Renyu Yang, Peter Garraghan, Tao Lin, Jie Xu, and Michael Rovatsos. Fog orchestration for iot services: issues, challenges and directions. *IEEE Internet Computing*, 21(2):16–24, 2017. 2

[37] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019. 2

[38] Ju Xu and Zhanxing Zhu. Reinforced continual learning. *NIPS*, 31, 2018. 2

[39] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, pages 3014–3023, 2021. 2

[40] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995. PMLR, 2017. 2

[41] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2, 3

[42] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *CVPR*, pages 13208–13217, 2020. 1