# Center Loss Regularization for Continual Learning

**Anonymous authors**
Paper under double-blind review

## Abstract

The ability to learn different tasks sequentially is essential to the development of artificial intelligence. In general, neural networks lack this capability, the major obstacle being catastrophic forgetting. It occurs when the incrementally available information from non-stationary data distributions is continually acquired, disrupting what the model has already learned. Our approach remembers old tasks by projecting the representations of new tasks close to that of old tasks while keeping the decision boundaries unchanged. We employ the center loss as a regularization penalty that enforces new tasks' features to have the same class centers as old tasks and makes the features highly discriminative. This, in turn, leads to the least forgetting of already learned information. This method is easy to implement, requires minimal computational and memory overhead, and allows the neural network to maintain high performance across many sequentially encountered tasks. We also demonstrate that using the center loss in conjunction with the memory replay outperforms other replay-based strategies. Along with standard MNIST variants for continual learning, we apply our method to continual domain adaptation scenarios with the Digits and PACS datasets. We demonstrate that our approach is scalable, effective, and gives competitive performance compared to state-of-the-art continual learning methods.

## 1 Introduction

Humans have the ability to continuously evolve, accumulate and transfer acquired knowledge to learn new skills throughout their lifetime. In contrast, in the classical machine learning paradigm, typically referred to as *isolated learning* (Chen & Liu, 2018), systems are capable of achieving high performance in learning isolated tasks or narrow domains without using previously learned knowledge. This makes them different from real-world settings where systems are expected to learn consecutive tasks with changing data distributions and unknown task boundaries. In this scenario, the intelligent agent should learn continually without forgetting the already acquired knowledge. Thus, continual learning, or traditionally called lifelong learning (Chen & Liu, 2018; Thrun, 1995; 1996; 1998; Thrun & Pratt, 2012), becomes necessary for artificial general intelligence.

A significant problem in continual learning is catastrophic forgetting in neural networks, also known as catastrophic interference (McCloskey & Cohen, 1989). The newly learned information may interfere and disrupt the already learned knowledge, leading to a performance loss on old tasks (Ratcliff, 1990). The extent to which the system should be prone to refine and integrate new knowledge and retain previous information was termed as a *stability-plasticity dilemma* and is well-studied in many previous works. (Grossberg, 1982; 2013; Mermillod et al., 2013). This issue of catastrophic forgetting is known to exist in many different types of neural networks, from standard backpropagation networks to unsupervised networks like self-organizing maps (Richardson & Thomas, 2008; Mermillod et al., 2013). There have been several attempts to overcome catastrophic forgetting in neural networks, and the various approaches are discussed in the next section.

### 1.1 Continual Learning Approaches

In general, current continual learning methods can be broadly categorized into three different types of strategies based on how they attempt to solve the problem of catastrophic forgetting.

**Architectural approaches** mitigate catastrophic forgetting by modifying the architectural properties of the networks, e.g., adding more neurons or layers or incorporating weight freezing strategies. One of the earliest strategies in this category was Progressive Neural Networks (PNN) proposed by Rusu et al. (2016) that retains a pool of pre-trained models as knowledge and learns lateral connections among them to learn the task at hand. Another simpler strategy, Copy Weights with Re-init (CWR), was proposed (Lomonaco & Maltoni, 2017) where consolidated knowledge is maintained by isolating the subsets of weights for each class and learning rest task-specific parameters. Scalability remains an issue in this category of approaches as the network parameters explode as the number of tasks increases.

**Replay or rehearsal-based approaches** maintain a memory buffer with samples from previous tasks to replay with the examples from the current task to strengthen the old memories. (Rolnick et al., 2018). Lopez-Paz & Ranzato (2017) proposed Gradient Episodic Memory (GEM) that favors positive backward transfer and hence mitigating forgetting by using episodic memory of samples from previous tasks. Later, Chaudhry et al. (2019) proposed a more efficient and faster version, called Averaged GEM (A-GEM). Inspired by the suggestion that the hippocampus is better paralleled with a generative model than a replay buffer (Ramirez et al., 2013; Stickgold & Walker, 2007), the replay approach was improved further by replacing the memory buffer with a generative model which could generate unlimited pseudo data from past tasks (Shin et al., 2017; Van de Ven & Tolias, 2018). Instead of using stored input samples, the replay of latent representations to mitigate forgetting is also explored in many recent works (Pellegrini et al., 2020; van de Ven et al., 2020).

**Regularization-based approaches** attenuate catastrophic forgetting by imposing constraints on the update of the network weights (Parisi et al., 2019). It is generally formulated via additional regularization terms that penalize changes in the weights or predictions of neural network. Learning Without Forgetting (LwF) (Li & Hoiem, 2017) distills knowledge with the network's previous version to enforce the predictions of current and previous tasks to be similar. Many recent regularization based methods apply penalty on network parameters and estimate the importance of different network parameters. Kirkpatrick et al. (2017) proposed the Elastic Weight Consolidation (EWC), which imposes a quadratic penalty on the difference between the old and new task parameters to slow down the learning on certain weights based on their importance for previous tasks (Parisi et al., 2019). In Synaptic Intelligence (SI) (Zenke et al., 2017), individual synapses are allowed to estimate their importance by computing the path integral of the gradient vector field along the parameter trajectory. Whereas Memory Aware Synapses (MAS) (Aljundi et al., 2018) computes importance based on the sensitivity of predicted output function to each parameter. On the other hand, instead of imposing penalty directly on weights, Less-Forgetful Learning (LFL) (Jung et al., 2018) regularizes the $L_2$ distance between the new and old feature representations to preserve the previously learned input-output mappings by computing auxiliary activations with the old task parameters. Recent regularization works also build upon the traditional Bayesian online learning framework with variational inference (Nguyen et al., 2017; Ahn et al., 2019; Adel et al., 2020).

### 1.2 Motivation

The architectural approaches suffer from scalability issues as the number of parameters increases with the number of tasks (Parisi et al., 2019). On the other hand, rehearsal-based strategies generally require large memory buffers to store old task data for high performance. Moreover, in real-world scenarios, it is not always possible to have access to old task data. The generative replay-based methods attempt to solve this issue but are often difficult to train and computationally expensive. On the contrary, the regularization-based strategies assume that all the information essential about the old task is contained in the network weights (Kirkpatrick et al., 2017). The over-parameterization in neural networks makes it possible for the solution to a new task to be found close to the solution for the old task (Hecht-Nielsen, 1992; Sussmann, 1992; Kirkpatrick et al., 2017). Thus, the regularization strategies are generally memory efficient and computationally less expensive than the other two approaches. Our approach belongs to this category as it focuses on alleviating the catastrophic forgetting problem by regularizing the network to project the new task representations close to the old task representations while keeping the decision boundaries unchanged. We achieve this using the center loss (Wen et al., 2016) as a regularization penalty to minimize forgetting. We show that our approach successfully prevents catastrophic forgetting in a computationally efficient manner without accessing the data from old tasks.

### 1.3 Contributions

The contributions of this paper are as follows:

1. We propose a novel regularization-based continual learning strategy which we refer to as center loss regularization (CLR).
2. We compare our approach to different continual learning strategies in domain incremental scenarios and show that our approach is scalable, computationally efficient while storing minimal additional parameters.
3. We show that our approach gives a competitive performance with the state-of-the-art techniques when applied in continual domain adaptation scenarios.

## 2 Center Loss Regularization (CLR)

Deep neural networks excel at learning the hierarchical internal representations from raw input data by stacking multiple layers, which allows the system to learn complex function mappings from input to output (Farabet et al., 2013). These representations become increasingly invariant to small changes in the input as we go up the layers towards the output layer by preserving the vital information about the input related to the task (Guest & Love, 2019). The portion till the last hidden layer is considered the feature extractor, and the last fully connected layer is regarded as a linear classifier as the features extracted from the feature extractor are usually linearly separable due to the softmax activation in the top layer (Wen et al., 2016). The Less-Forgetful Learning (LFL) approach (Jung et al., 2016) demonstrated that catastrophic forgetting can be prevented if the representations of the new task are projected close to the learned representations of the old task keeping the decision boundaries unchanged. Ramasesh et al. (2020) empirically demonstrated that if the higher layers are stabilized while learning subsequent tasks, the forgetting can be mitigated significantly.

In our approach, we freeze the weights of the last fully connected classification layer to keep the decision boundaries unchanged similar to LFL. However, it is non-trivial to make the learned features for new tasks localize nearby corresponding old task features in the latent space. The LFL solves it by using the $L_2$ distance between the current model's features and the computed features using the old task model as a regularization penalty to preserve the previously learned input-output mappings. However, this approach is highly memory-intensive and computationally expensive since it requires storing the entire model trained on the old task and does forward pass on it to compute representations for each novel task. Wen et al. (2016) introduced the center loss and demonstrated that the joint supervision of softmax loss and center loss helps to increase the inter-class dispersion and intra-class compactness of deeply learned features. During the training process, the model also learns the centers for each class features around which the deeply learned features are typically clustered (Wen et al., 2016). We exploit these properties of center loss in building our continual learning strategy. More details on the center loss are provided in the Appendix D.

We use the center loss as a regularization penalty along with softmax loss. To enforce the model to learn the features for the new task in the proximity of corresponding old task features, we utilize the already learned class feature centers of the old task instead of storing the old model weights like LFL. While learning the new task, we enforce the new task features to be close to the already learned feature centers using the center loss, making the model project the features of all tasks in the same localized region, clustered around corresponding class feature centers. For the new task, we freeze the class centers and reduce the learning rate of feature extractor parameters to prevent significant changes in it while training with the new task. We also provide the findings of our ablation study in the Section 3.3 to analyze the effects of letting the centers and decision boundaries change while learning new tasks. As our method needs to store only the feature centers for each class throughout the lifetime, the memory requirement is significantly lower than other approaches where the agent needs to store the model weights or maintain the replay buffer. The extra memory requirements are discussed in detail in Section 3.3.1

$$L_t = L_s + \lambda L_c \tag{1}$$

$$L_c = \frac{1}{2}\sum_{i=1}^{m} \|f_{L-1}(x_i;\theta^{(n)}) - c_{y_i}^{(o)}\|_2^2 \tag{2}$$

$$\hat{\theta}^{(n)} = \arg\min_{\theta^{(n)}} L_t(x, c^{(o)}; \theta^{(n)}) + R(\theta^{(n)}) \tag{3}$$

The equation 1 represents the joint loss, a combination of the softmax loss $L_s$ and the center loss $L_c$, which is minimized during training. Minimizing $L_s$ helps solve the current/new task, whereas the term $L_c$ helps retain already learned knowledge and avoid catastrophic forgetting. A scalar $\lambda$ is used for balancing the two loss functions. The equation 2 defines the modified center loss $L_c$, where $c^{(o)}$ denotes the learned values of feature centers from old task. The centers are kept frozen during the subsequent tasks. This enforces the new task representations to have the same feature centers as the old task, leading the old task and new task representations to stay within proximity in the feature space, reducing the catastrophic forgetting. We obtain the equation 3 as final objective function where $R(\cdot)$ denotes a general regularization term, such as weight decay. Finally, we propose the center loss regularization algorithm, as shown in Algorithm 1. $N$ denotes the number of training iterations, $D^{(n)}$ denotes the new task data, $\theta^{(o)}$ and $\theta^{(n)}$ denote the network parameters for the old and new task, respectively. Note that we use a single-headed model in our experiments as we primarily target domain-incremental scenario of continual learning where task identity need not be inferred at test time (Van de Ven & Tolias, 2019). The system learns to adapt to changing input distributions, but the task structure remains the same.

**Algorithm 1** Center Loss Regularization (CLR)

```
Input: $\theta^{(o)}, c^{(o)}, N, \mathcal{D}^{(n)}$
Output: $\hat{\theta}^{(n)}$
1: $\theta^{(n)} \leftarrow \theta^{(o)}$                                  ▷ initialize weights
2: Freeze the weights of the softmax classification layer.
3: for $i \leftarrow 1, N$ do                                    ▷ training iteration
4:     for each minibatch $\mathcal{B} \in \mathcal{D}^{(n)}$ do
5:         Backpropagate and update $\theta^{(n)}$ for mini-batch $\mathcal{B}$ to minimize loss $L_t + R(\theta^{(n)})$
6:     end for
7: end for
8: $\hat{\theta}^{(n)} \leftarrow \theta^{(n)}$
9: return $\hat{\theta}^{(n)}$
```

## 3 Experiments

In this section, we first detail the experimental protocols for evaluating lifelong learning algorithms (Section 3.1). We have compared our proposed method against different continual learning methods, which are specified in Section 3.2. Finally, we report our results in Section 3.3.

### 3.1 Experimental protocols

**Permuted & Rotated MNIST** (Kirkpatrick et al., 2017) are variants of the original MNIST dataset (LeCun, 1998). In Permuted MNIST, a fixed permutation of the image pixels is applied to each task's training and test set. In Rotated MNIST, each task consists of images rotated by a fixed angle between 0 and 180 degrees. We chose 10000 training and 5000 testing samples for both these variants. Each task has a test set with the same rotation transformation as the training set for that task. In these experiments, the model encounters 10 tasks in sequence, each with a unique rotation angle. We evaluate the model on the test sets of the respective datasets after training on each task. We use the fully connected neural network (MLP) with two hidden layers of 100 units, each with ReLU activation for our experiments. We train the network using Adam optimizer (Kingma & Ba, 2014) on mini-batches of 64 samples for 1 epoch over training set per task with a learning rate of $3 \cdot 10^{-3}$ for the first task and $3 \cdot 10^{-4}$ for subsequent tasks.

Our method can also be applied in the supervised continual domain adaptation settings where the model needs to adapt to the new domains without degrading the performance on the previously seen domains (Jung et al., 2018; Volpi et al., 2021).

**Digit Recognition** We consider four widely used datasets for digit recognition in continual domain adaptation setting: MNIST (LeCun, 1998), SVHN (Netzer et al., 2011), MNIST-M (Ganin & Lempitsky, 2015) and SYN (Ganin & Lempitsky, 2015). To assess the performance of our approach, we train our network on these datasets, one by one in the sequence MNIST $\rightarrow$ MNIST-M $\rightarrow$ SYN $\rightarrow$ SVHN, considering each dataset as one task. This way, the network adapts to harder domains continually. For this protocol, 10000 training and 5000 testing samples are chosen for each dataset, and all images are resized to 28 x 28 pixels. We convert the MNIST dataset to 3-channel images by repeating the original channel 3 times for compatibility with the remaining datasets. We use the ResNet18 (He et al., 2016) architecture and train the network on each domain for 20 epochs, with a batch size of 64. We use Adam optimizer (Kingma & Ba, 2014) with a learning rate of $3 \cdot 10^{-4}$, which is reduced to $3 \cdot 10^{-5}$ after the first domain.

**PACS** dataset (Li et al., 2017) is typically used to assess the domain generalization scenarios. It consists of four domains, namely Photo (1,670 images), Art Painting (2,048 images), Cartoon (2,344 images), and Sketch (3,929 images). Each domain contains seven categories. We use this dataset to evaluate our approach in a continual domain adaptation setting. We trained the network in sequence Sketches $\rightarrow$ Cartoons $\rightarrow$ Paintings $\rightarrow$ Photos where images become more realistic with the new domain. For each domain, the dataset is split into 70% training and the remaining 30% as testing samples. All images are resized to 224 x 224 pixels and standardized with the mean and standard deviation of the ImageNet (Deng et al., 2009) dataset. This is due to the fact that this protocol uses ResNet18 (He et al., 2016) network trained on ImageNet dataset. We train the network on each domain for 5 epochs, with a batch size of 64. We use Adam optimizer (Kingma & Ba, 2014) with a learning rate of $3 \cdot 10^{-4}$, which is reduced to $3 \cdot 10^{-5}$ after the first domain.

## 3.2 METHODS

We compare the performance of the proposed approach with that of the state-of-the-art regularization-based strategies. First, we test the fine-tuning approach in which a single model is trained across all tasks, which is the naive approach. Second, we test the LwF (Li & Hoiem, 2017) method, which uses only new task data to train the network while preserving the original capabilities. Further, we compare the performance with EWC (Kirkpatrick et al., 2017), SI (Zenke et al., 2017) and MAS (Aljundi et al., 2018) methods which try to estimate synaptic importance of different network parameters and use that information to regularize the weights. We also test the LFL (Jung et al., 2018) method, which tries to position the features extracted by the new network, close to the features extracted by the old network. Moreover, we explore frameworks like Uncertainty-regularized Continual Learning (UCL) (Ahn et al., 2019) and Variational Continual Learning (VCL) (Nguyen et al., 2017) based on variational inference. VCL employs a projection operator through KL divergence minimization. UCL solves the drawbacks of VCL by proposing the concept of node-wise uncertainty. Then, we consider two oracle methods: If we assume access to every domain at every point in time, we can either train on samples from the joint distribution from the beginning *oracle (all)*, or grow the distribution over iterations *oracle (cumulative)*. With access to samples from all domains, oracles are not generally exposed to catastrophic forgetting; yet, their performance is not necessarily an upper bound (Lomonaco & Maltoni, 2017).

We also compare our approach with several replay-based strategies. We use the basic experience-replay method as a baseline and examine if using CLR as surrogate loss along with experience replay can enhance the performance. Further, we compare its performance with state-of-the-art memory-based methods like average gradient episodic memory (A-GEM) (Chaudhry et al., 2019) and GDumb (Prabhu et al., 2020) methods. We also compare CLR with recent continual domain adaptation technique called domain randomization and meta-learning (Meta-DR) (Volpi et al., 2021) for continual domain adaptation experiments. For Permuted and Rotated MNIST tasks, we experiment with four different memory sizes of 10, 20, 50, and 100 examples per task. For the Digits dataset, we experiment with 100, 200, 300 replay examples per task. For the PACS dataset, we experiment with 10, 20, 30 replay examples per task.

The metrics used in our experiments to evaluate all considered methods are the average accuracy (ACC) and backward transfer (BWT). ACC is the average of the accuracy of the model across all encountered tasks, and BWT represents the amount of forgetting at the end of training on all tasks (Lopez-Paz & Ranzato, 2017). The formulae for the metrics are presented in detail in Appendix B. Each experiment in the paper is carried out for 5 trials, and the final values are reported as mean and standard deviation of results.

### 3.3 Results

Table 1 compares the performances of different regularization-based approaches on the Rotated and Permuted MNIST datasets. From this table, we can observe that our approach CLR outperforms all other methods. VCL shows competitive performance for Permuted MNIST, but the amount of forgetting (BWT) is worse than CLR. Whereas, LwF shows competitive performance in terms of forgetting but less adaptability to new information. Figure 1 presents the evolution of the average accuracy (ACC) and the first-task accuracy throughout all the tasks for the MNIST variants. This shows that the center loss regularization helps to mitigate the problem of catastrophic forgetting on earlier tasks while maintaining high performance on all the tasks.

In Table 2, we report the results of replaying samples from episodic memory along with our proposed approach for both MNIST variants. We can observe that using center loss regularization significantly improves the performance over the plain experience replay strategy. Such improvement is consistent as we increase the memory size. It also outperforms the state-of-the-art memory-based methods like A-GEM and GDumb. This shows that our approach can also be used as surrogate loss along with replay-based strategies to enhance performance and reduce forgetting.

We provide the ablation study of our approach in Table 3. We demonstrate how the performance changes if we do not freeze the centers and classifier weights after the training on the first task is completed. The first row in the table represents the approach proposed in Section 2. We try out multiple values of hyperparameter $\lambda$ for each experiment and put the best performing results for each row. These results show that the best performance is generally achieved when we freeze the centers and the decision boundaries after the first task. Moreover, we also demonstrate how the hyperparameter $\lambda$ affects the overall performance of CLR in Appendix Figure 3.

We report in Table 4 and Table 5 the performance of our proposed method compared with different methods in continual domain adaptation setting on Digits and PACS datasets respectively. We note that our approach outperforms the naive, EWC, and SI strategies with a significant margin for both benchmarks. Our method demonstrates competitive performance compared to LwF and LFL strategies. Having access to the samples from older tasks for replay can help reduce catastrophic forgetting significantly compared to regularization methods which do not have access to the data of old domains. Thus, we also examine if using CLR along with Experience Replay (ER) can help boost the performance. For both the benchmarks, we observe that using CLR with ER can significantly improve the overall performance, and it is consistent with the increase in the memory size. The memory size column denotes the number of replay samples per task. These results suggest that the center loss regularization helps the model successfully adapt to new domains without considerable performance degradation on old domains.

#### 3.3.1 Comparison of Additional Memory Requirement

Generally, the regularization-based methods store the old network parameters for regularization or knowledge distillation. In Table 6, we compare the extra memory requirement of different regularization-based methods in terms of the number of additional parameters other than the base network parameters. Further, we explain why CLR is the cheapest option from an additional memory requirement perspective compared to other regularization techniques.

In order to quantify the importance of weights to previous tasks, EWC needs to compute and store the diagonal of the Fisher matrix for each task, which has the same number of elements as the network parameters. Additionally, optimal parameters from previous tasks are also stored in order to compute the knowledge distillation loss. Moreover, a few samples from previous tasks are maintained in our experiments to compute the Fisher matrix after each task is completed. Thus, given that $k$ is the number of encountered tasks and $p$ is the number of network parameters, the space complexity for additional memory usage in EWC becomes $\mathcal{O}(k * p)$. In contrast to EWC,

Table 1: Average test accuracy (in %) (mean and std) after training on all tasks for each method

| Method | Rotated MNIST | | Permuted MNIST | |
|---|---|---|---|---|
| | ACC | BWT | ACC | BWT |
| Naive | 57.4 ± 1.2 | -30.2 ± 1.5 | 49.0 ± 1.9 | -47.3 ± 2.2 |
| LwF | 74.4 ± 0.9 | -12.1 ± 1.3 | 63.7 ± 0.8 | -12.7 ± 1.1 |
| EWC | 73.6 ± 0.7 | -16.5 ± 0.8 | 68.2 ± 0.4 | -19.5 ± 0.3 |
| SI | 74.4 ± 0.3 | -15.6 ± 0.4 | 67.5 ± 1.0 | -20.0 ± 1.2 |
| LFL | 75.5 ± 0.5 | -16.2 ± 1.4 | 71.9 ± 0.8 | -14.4 ± 1.4 |
| MAS | 67.5 ± 1.3 | -12.1 ± 1.5 | 71.8 ± 1.2 | -13.9 ± 1.6 |
| UCL | 68.2 ± 1.6 | -23.6 ± 1.2 | 66.5 ± 1.4 | -26.6 ± 1.5 |
| VCL | 63.6 ± 1.9 | -28.9 ± 1.3 | 74.6 ± 0.7 | -14.8 ± 1.7 |
| CLR (Ours) | **76.4 ± 1.2** | **-11.9 ± 0.6** | **75.6 ± 1.3** | **-12.2 ± 0.9** |

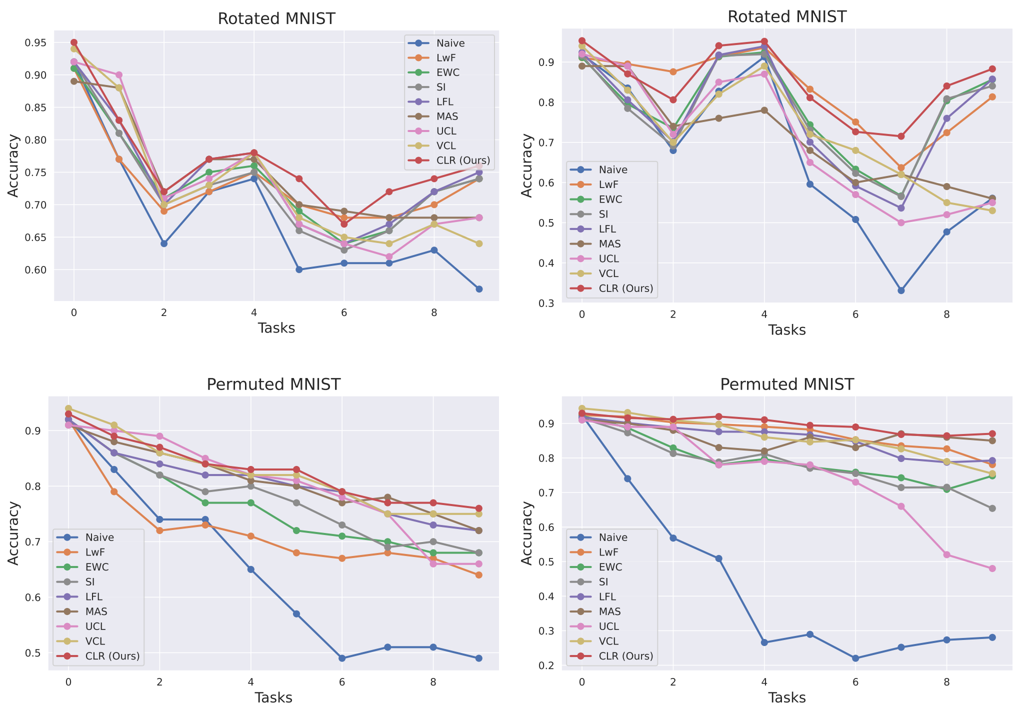

Figure 1: Left: Average accuracy (ACC) on tasks encountered so far for each method. Right: Progression of the test accuracy for the first task for each method, as more tasks are learned.

SI computes parameter-specific importance online. So, it does not require storing extra parameters for each task but maintains only the previous task model parameters and regularization strength for each network parameter to compute the surrogate loss. Thus, the space complexity for additional memory becomes $\mathcal{O}(p)$ in the case of SI. MAS also requires storing synaptic importances for each model parameter, needing a similar amount of memory as SI. VCL stores variance term for each weight parameter for current and previous models; hence, it requires thrice the number of additional parameters than the original model. UCL solves this drawback of VCL by computing uncertainty (importance) at node-level, reducing the total required parameters to almost half compared to VCL (Ahn et al., 2019). However, CLR outdoes both of them with lesser parameters.

Moreover, both LwF and LFL require storing the old network in the memory to compute the knowledge distillation loss for the previous task. Like SI, these two methods LFL and LwF, have the space complexity for additional memory as $\mathcal{O}(p)$. CLR attempts to achieve a similar objective as the LFL projecting old and new task features in close proximity. However, the CLR is computationally and

Table 2: Replay Experiments Study: Average test accuracy (mean % and std) with varying memory size (Mem size) i.e number of replay examples stored for each task

| **Method** | Rotated MNIST | | | | Permuted MNIST | | | |
|---|---|---|---|---|---|---|---|---|
| Mem size | 10 | 20 | 50 | 100 | 10 | 20 | 50 | 100 |
| ER | 78.6 ± 1.4 | 81.0 ± 0.5 | 83.3 ± 0.4 | 85.1 ± 0.4 | 68.9 ± 1.4 | 71.7 ± 1.7 | 80.6 ± 0.4 | 83.5 ± 0.6 |
| A-GEM | 77.6 ± 0.4 | 78.5 ± 0.5 | 79.3 ± 0.4 | 79.7 ± 0.6 | 76.8 ± 0.4 | 77.7 ± 0.5 | 84.3 ± 0.7 | 83.9 ± 0.4 |
| GDumb | 78.5 ± 0.3 | 81.1 ± 0.8 | 84.5 ± 0.4 | 85.2 ± 0.5 | 76.7 ± 0.8 | 78.5 ± 0.9 | 84.3 ± 1.0 | 85.3 ± 0.7 |
| ER + CLR | **81.0 ± 0.4** | **83.3 ± 0.2** | **85.0 ± 0.2** | **86.6 ± 0.1** | **78.9 ± 0.5** | **80.0 ± 0.9** | **83.7 ± 0.2** | **86.0 ± 0.4** |

Table 3: Ablation Study: Average test accuracy (mean % and std)

| Frozen | | Rotated MNIST | | Permuted MNIST | |
|---|---|---|---|---|---|
| Centers | Classifier | ACC | BWT | ACC | BWT |
| ✓ | ✓ | **76.4 ± 1.2** | **-11.9 ± 0.6** | **75.6 ± 1.3** | **-12.2 ± 0.9** |
| ✓ | ✗ | 76.3 ± 1.2 | -15.0 ± 1.0 | 73.2 ± 0.7 | -15.7 ± 0.8 |
| ✗ | ✓ | 75.7 ± 0.7 | -15.7 ± 0.9 | 74.3 ± 1.2 | -14.0 ± 1.6 |
| ✗ | ✗ | 76.4 ± 0.9 | -12.5 ± 0.4 | 73.4 ± 2.4 | -15.4 ± 2.5 |

memory-wise more efficient than the LFL and LwF because the CLR does not need to store the old model and forward pass it, significantly reducing the memory usage and training time.

Hence, all these methods require a large amount of extra memory, which further increases with the number of tasks or the network size. On the other hand, CLR only stores the feature centers of each class, which significantly reduces the additional memory requirement compared to the previous methods. Thus, the space complexity of extra memory usage in CLR comes down to $\mathcal{O}(n*d)$, where $n$ is the number of classes in each task, and each feature center is $d$-dimensional.

## 4 Limitations and Future Directions

There are several exciting research directions to extend our work for continual learning. Our method requires the knowledge of task boundaries which may not always be available. CLR does not leverage the task descriptors, which may be exploited to obtain a positive forward transfer. Further, novel approaches can also be developed to exploit our approach for supporting task-incremental and class-incremental learning, where task-IDs need to be inferred. In this paper, we applied CLR to solve the supervised classification problem in continual learning. It would be an interesting research direction to exploit the properties of CLR to solve other problems like regression and dimensionality reduction in the continual learning setting. Moreover, the effects of using other discriminative representation learning approaches (Hadsell et al., 2006; Sun, 2015; Deng et al., 2017; Zhang et al., 2017; Liu et al., 2017; Chen et al., 2017; Wan et al., 2018; Qi & Zhang, 2018; Wang et al., 2018a;b) can be studied for continual learning.

## 5 Conclusion

In this paper, we proposed a new regularization-based strategy for continual learning, referred to as *center loss regularization (CLR)*. It utilizes the power of center loss to learn discriminative features and use the learned feature centers to project new task features in the proximity of old task features to transfer knowledge and avoid catastrophic forgetting. Our method was effective in overcoming catastrophic forgetting when applied to the standard continual learning benchmarks as well as continual domain adaptation benchmarks. Our method is scalable and computationally effective, and it does not store previous data and requires minimal additional network parameters. Our extensive experiments consistently demonstrate the competitive performance of CLR against the state-of-the-art regularization strategies for continual learning.

Table 4: Continual Domain Adaptation (Digits): Test accuracy on individual datasets, average test accuracy (ACC) and BWT (mean % and std) after training on all domains with Digits protocol

| Method | M.Size | **MNIST (1)** | **MNIST-M (2)** | **SYN (3)** | **SVHN (4)** | **ACC** | **BWT** |
|---|---|---|---|---|---|---|---|
| Naive | - | 83.4 ± 6.4 | 65.8 ± 3.4 | 71.3 ± 0.4 | 92.6 ± 0.1 | 78.2 ± 0.4 | -12.8 ± 0.7 |
| LwF | - | 96.1 ± 1.2 | 72.7 ± 0.9 | 76.6 ± 0.7 | 91.1 ± 1.0 | 84.1 ± 1.3 | -3.6 ± 1.7 |
| EWC | - | 93.7 ± 0.8 | 67.9 ± 0.6 | 75.1 ± 0.3 | 92.5 ± 0.2 | 82.3 ± 0.9 | -7.8 ± 0.8 |
| SI | - | 91.8 ± 0.5 | 69.2 ± 1.2 | 72.5 ± 0.5 | 91.9 ± 1.1 | 81.3 ± 0.6 | -8.9 ± 0.4 |
| LFL | - | 95.5 ± 0.9 | 67.3 ± 0.4 | 77.7 ± 1.3 | 93.2 ± 0.9 | 83.4 ± 0.4 | -7.5 ± 0.3 |
| Meta-DR | - | 85.7 ± 1.8 | 75.4 ± 0.7 | 82.0 ± 1.9 | 98.5 ± 0.3 | 85.4 ± 1.1 | -8.4 ± 1.0 |
| CLR | - | 95.8 ± 0.6 | 71.9 ± 0.5 | 77.2 ± 0.2 | 93.5 ± 0.2 | 84.6 ± 0.9 | -5.7 ± 0.6 |
| ER | 100 | 96.2 ± 0.4 | 75.0 ± 1.1 | 79.3 ± 0.3 | 93.3 ± 0.9 | 86.2 ± 0.7 | -4.8 ± 0.7 |
|  | 200 | 96.7 ± 0.6 | 77.5 ± 1.5 | 80.8 ± 0.7 | 93.7 ± 0.2 | 87.1 ± 0.4 | -3.7 ± 0.8 |
|  | 300 | 97.3 ± 1.3 | 78.8 ± 1.2 | 78.6 ± 0.2 | 92.7 ± 0.5 | 86.9 ± 0.5 | -3.6 ± 0.5 |
| A-GEM | 100 | 95.2 ± 0.3 | 77.1 ± 1.3 | 78.6 ± 0.5 | 94.0 ± 0.8 | 85.9 ± 0.5 | -5.0 ± 1.0 |
|  | 200 | 95.8 ± 1.0 | 77.9 ± 0.8 | 79.1 ± 0.4 | 94.2 ± 0.3 | 86.7 ± 0.6 | -3.9 ± 0.3 |
|  | 300 | 96.9 ± 0.8 | 78.7 ± 0.9 | 80.0 ± 1.2 | 93.8 ± 0.6 | 87.3 ± 0.8 | -3.5 ± 0.9 |
| ER + Meta-DR | 100 | 95.4 ± 0.8 | 79.6 ± 0.4 | 80.8 ± 0.3 | 92.7 ± 0.1 | 87.1 ± 0.4 | -3.7 ± 0.3 |
|  | 200 | 96.1 ± 0.7 | 80.1 ± 0.9 | 81.2 ± 0.5 | 93.0 ± 0.4 | 87.5 ± 0.6 | -3.4 ± 0.4 |
|  | 300 | 97.2 ± 0.4 | 80.3 ± 0.5 | 82.3 ± 0.1 | 93.4 ± 0.3 | 88.3 ± 0.4 | -2.8 ± 0.4 |
| ER + CLR | 100 | 97.1 ± 0.2 | 79.0 ± 0.5 | 81.8 ± 1.4 | 94.3 ± 0.8 | 88.1 ± 0.6 | -2.7 ± 0.3 |
|  | 200 | 97.4 ± 0.5 | 80.8 ± 1.0 | 81.8 ± 0.3 | 94.1 ± 0.9 | 88.7 ± 1.2 | -2.1 ± 0.5 |
|  | 300 | 97.9 ± 0.8 | 81.3 ± 1.0 | 82.6 ± 0.3 | 94.4 ± 0.6 | 89.1 ± 0.8 | -1.7 ± 0.9 |
| Oracle (all) | - | 98.6 ± 0.3 | 90.7 ± 0.4 | 87.0 ± 0.5 | 94.7 ± 0.2 | 92.8 ± 0.8 | - |
| Oracle (cumul.) | - | 98.6 ± 0.7 | 86.6 ± 1.1 | 82.2 ± 0.3 | 92.1 ± 0.1 | 89.9 ± 0.3 | +1.7 ± 0.4 |

Table 5: Continual Domain Adaptation (PACS): Test accuracy on individual datasets, average test accuracy (ACC) and BWT (mean % and std) after training on all domains for continual domain adaptation with PACS

| Method | M.Size | **Sketches (1)** | **Cartoons (2)** | **Paintings (3)** | **Photos (4)** | **ACC** | **BWT** |
|---|---|---|---|---|---|---|---|
| Naive | - | 74.7 ± 2.1 | 65.6 ± 1.6 | 86.7 ± 1.4 | 90.4 ± 0.9 | 77.4 ± 0.5 | -15.0 ± 1.2 |
| LwF | - | 79.3 ± 0.8 | 68.4 ± 1.6 | 95.8 ± 1.1 | 88.6 ± 0.5 | 81.4 ± 0.8 | -11.9 ± 1.3 |
| EWC | - | 81.1 ± 1.6 | 71.8 ± 0.7 | 96.2 ± 0.4 | 89.1 ± 0.2 | 83.1 ± 0.9 | -8.8 ± 0.8 |
| SI | - | 82.6 ± 1.1 | 70.0 ± 1.4 | 92.2 ± 1.6 | 90.5 ± 0.7 | 83.7 ± 1.2 | -10.2 ± 0.8 |
| LFL | - | 86.6 ± 0.7 | 73.5 ± 0.9 | 95.2 ± 1.2 | 90.4 ± 1.4 | 85.7 ± 1.1 | -7.5 ± 1.0 |
| Meta-DR | - | 86.8 ± 1.2 | 75.4 ± 0.7 | 95.3 ± 0.9 | 88.5 ± 0.3 | 85.9 ± 0.9 | -7.2 ± 0.8 |
| CLR | - | 86.7 ± 1.4 | 74.0 ± 0.8 | 97.0 ± 1.2 | 89.9 ± 0.7 | 86.1 ± 1.3 | -7.9 ± 0.4 |
| ER | 10 | 89.2 ± 1.2 | 84.3 ± 0.3 | 96.8 ± 0.9 | 91.8 ± 1.9 | 89.9 ± 1.5 | -3.4 ± 0.6 |
|  | 20 | 92.3 ± 0.8 | 83.1 ± 1.2 | 96.4 ± 2.1 | 90.7 ± 0.7 | 90.1 ± 1.7 | -2.0 ± 1.3 |
|  | 30 | 91.7 ± 0.5 | 84.8 ± 1.6 | 97.6 ± 2.4 | 91.5 ± 0.4 | 91.0 ± 1.2 | -2.2 ± 1.1 |
| A-GEM | 10 | 87.2 ± 1.1 | 85.0 ± 0.8 | 97.1 ± 0.9 | 90.2 ± 1.9 | 89.8 ± 0.9 | -4.1 ± 0.8 |
|  | 20 | 91.8 ± 0.7 | 84.1 ± 0.8 | 95.4 ± 1.7 | 91.1 ± 0.7 | 90.6 ± 1.4 | -2.7 ± 1.2 |
|  | 30 | 91.2 ± 1.2 | 84.5 ± 0.9 | 97.1 ± 1.4 | 90.3 ± 2.2 | 90.7 ± 1.2 | -2.4 ± 1.4 |
| ER + Meta-DR | 10 | 90.2 ± 0.4 | 84.3 ± 1.7 | 95.1 ± 1.5 | 90.3 ± 1.9 | 90.0 ± 1.5 | -3.5 ± 1.9 |
|  | 20 | 92.5 ± 1.8 | 82.9 ± 2.2 | 95.6 ± 1.2 | 91.2 ± 1.7 | 90.5 ± 0.6 | -2.1 ± 1.1 |
|  | 30 | 90.7 ± 1.6 | 84.9 ± 0.8 | 97.8 ± 2.1 | 89.4 ± 1.4 | 90.7 ± 1.2 | -2.6 ± 1.3 |
| ER + CLR | 10 | 90.7 ± 0.5 | 84.1 ± 0.7 | 97.4 ± 1.4 | 91.5 ± 0.6 | 90.4 ± 1.1 | -3.6 ± 0.7 |
|  | 20 | 91.5 ± 0.6 | 85.4 ± 1.6 | 96.6 ± 1.4 | 91.4 ± 0.8 | 90.9 ± 1.3 | -2.8 ± 0.5 |
|  | 30 | 91.3 ± 1.0 | 87.1 ± 0.9 | 96.8 ± 0.4 | 92.4 ± 0.6 | 91.4 ± 0.8 | -2.9 ± 1.0 |
| Oracle (all) | - | 93.5 ± 1.5 | 91.3 ± 1.8 | 95.0 ± 0.7 | 85.6 ± 1.2 | 91.6 ± 0.8 | - |
| Oracle (cumul.) | - | 94.3 ± 2.3 | 92.3 ± 1.9 | 96.2 ± 1.4 | 91.0 ± 0.9 | 93.5 ± 1.6 | +0.8 ± 0.5 |

Table 6: Comparison of additional memory requirement for all methods

| Protocol/Method | LwF | EWC | SI | LFL | MAS | UCL | VCL | CLR |
|---|---|---|---|---|---|---|---|---|
| MNIST (MLP) | 89.6K | 1.79M | 179K | 89.6K | 179K | 89.8K | 269K | 1000 |
| Digits, PACS (ResNet18) | 11M | 88M | 22M | 11M | 22M | 11M | 33M | 5120, 3584 |

## REFERENCES

Tameem Adel, Han Zhao, and Richard E. Turner. Continual learning with adaptive weights (claw). In *International Conference on Learning Representations*, 2020.

Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. In *Advances in Neural Information Processing Systems*, pp. 4394–4404, 2019.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 139–154, 2018.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019.

B. Chen, W. Deng, and J. Du. Noisy softmax: Improving the generalization ability of dcnn via postponing the early softmax saturation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4021–4030, 2017.

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

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

Jiankang Deng, Yuxiang Zhou, and Stefanos Zafeiriou. Marginal loss for deep face recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2006–2014, 2017.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4685–4694, 2019.

Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8): 1915–1929, 2013.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pp. 1180–1189. PMLR, 2015.

Stephen Grossberg. How does a brain build a cognitive code? *Studies of mind and brain*, pp. 1–52, 1982.

Stephen Grossberg. Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural networks*, 37:1–47, 2013.

Olivia Guest and Bradley C. Love. Levels of representation in a deep learning model of categorization. *bioRxiv*, 2019. doi: 10.1101/626374.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pp. 65–93. Elsevier, 1992.

Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *ArXiv*, abs/1607.00122, 2016.

Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetful learning for domain expansion in deep neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

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

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550, 2017.

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

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, volume 2, pp. 7, 2016.

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *2017 IEEE conference on computer vision and pattern recognition*, pp. 212–220, 2017.

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

Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L. Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido M. van de Ven, Martin Mundt, Qi She, Keiland Cooper, Jeremy Forest, Eden Belouadah, Simone Calderara, German I. Parisi, Fabio Cuzzolin, Andreas S. Tolias, Simone Scardapane, Luca Antiga, Subutai Ahmad, Adrian Popescu, Christopher Kanan, Joost van de Weijer, Tinne Tuytelaars, Davide Bacciu, and Davide Maltoni. Avalanche: an end-to-end library for continual learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3595–3605, 2021.

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30, 2017.

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

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.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Ng. Reading digits in natural images with unsupervised feature learning. *NIPS*, 01 2011.

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

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.

Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10203–10209. IEEE, 2020.

Ameya Prabhu, Philip Torr, and Puneet Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *The European Conference on Computer Vision (ECCV)*, August 2020.

Xianbiao Qi and Lei Zhang. Face recognition via centralized coordinate learning. *arXiv preprint arXiv:1801.05678*, 2018.

Vinay V Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. *arXiv preprint arXiv:2007.07400*, 2020.

Steve Ramirez, Xu Liu, Pei-Ann Lin, Junghyup Suh, Michele Pignatelli, Roger L Redondo, Tomás J Ryan, and Susumu Tonegawa. Creating a false memory in the hippocampus. *Science*, 341(6144): 387–391, 2013.

Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

Fiona M Richardson and Michael SC Thomas. Critical periods and catastrophic interference effects in the development of self-organizing feature maps. *Developmental science*, 11(3):371–389, 2008.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*, 2018.

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.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017.

Robert Stickgold and Matthew P Walker. Sleep-dependent memory consolidation and reconsolidation. *Sleep medicine*, 8(4):331–343, 2007.

Yi Sun. *Deep learning face representation by joint identification-verification*. The Chinese University of Hong Kong (Hong Kong), 2015.

Héctor J Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural networks*, 5(4):589–593, 1992.

Sebastian Thrun. A lifelong learning perspective for mobile robot control. In *Intelligent robots and systems*, pp. 201–214. Elsevier, 1995.

Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pp. 640–646. Morgan Kaufmann Publishers, 1996.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pp. 181–209. Springer, 1998.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

Gido M Van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

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.

Riccardo Volpi, Diane Larlus, and Grégory Rogez. Continual adaptation of visual representations via domain randomization and meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4443–4453, 2021.

Weitao Wan, Yuanyi Zhong, Tianpeng Li, and Jiansheng Chen. Rethinking feature distribution for loss functions in image classification. In *2018 IEEE conference on computer vision and pattern recognition*, pp. 9117–9126, 2018.

Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018a.

Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *2018 IEEE conference on computer vision and pattern recognition*, pp. 5265–5274, 2018b.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pp. 499–515. Springer, 2016.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pp. 3987–3995. PMLR, 2017.

Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tailed training data. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5409–5418, 2017.

## A Training Details

We used one NVIDIA GeForce TITAN X GPU for training the models in all our experiments with the CUDA version 10.1 on a Linux machine. We built our models and implemented different methods using PyTorch deep learning framework and Avalanche library (Lomonaco et al., 2021) for Continual Learning. Hyperparameter search was done using the grid-search. Next, we provide the best hyperparameter values, specific to the methods for our experiments. The $\lambda$ denotes the importance of the regularization penalty for EWC, LwF, SI, LFL and MAS. Whereas $\beta$ controls the speed of standard deviation ($\sigma$) for the weight parameter in UCL. VCL does not need any additional hyperparameters. In CLR, $\lambda$ denotes the importance of the center loss and $\alpha$ denotes the rate with which the centers are allowed to change in the first task.

Table 7: Hyperparameters specific to methods in all experiments

| Method | Rotated MNIST | Permuted MNIST | Digits | PACS |
|---|---|---|---|---|
| EWC ($\lambda$) | 0.001 | 0.001 | 0.001 | 1.0 |
| LwF ($\lambda$) | 1.0 | 1.0 | 0.1 | 0.1 |
| SI ($\lambda$) | 0.001 | 0.001 | 0.1 | 1.0 |
| LFL ($\lambda$) | 1.0 | 0.001 | 0.1 | 1.0 |
| MAS ($\lambda$) | 10.0 | 1.0 | - | - |
| UCL ($\beta$) | 0.9 | 1.0 | - | - |
| CLR ($\lambda$, $\alpha$) | (0.001, 0.03) | (0.04, 0.03) | (0.01, 0.01) | (0.001, 0.5) |

## B Evaluation Metrics

Lopez-Paz & Ranzato (2017) introduced Average Accuracy (ACC) and Backward Transfer (BWT) evaluation metrics for continual learning, which we use in our experiments to evaluate our approach. For evaluation, we maintain a test set for each of the $T$ tasks. After learning on the new task $t_i$, we evaluate the model's test performance on all $T$ tasks. The formulas for calculating ACC and BWT are as follows. $A_{i,j}$ is the test classification accuracy of the model on task $t_i$ after observing the last data sample from task $t_j$.

$$\textbf{Average Accuracy (ACC)} = \frac{1}{T}\sum_{i=1}^{T} A_{i,T} \tag{4}$$

$$\textbf{Backward Transfer (BWT)} = \frac{1}{T-1}\sum_{i=1}^{T-1} A_{i,T} - A_{i,i} \tag{5}$$

The larger the value of ACC, the better is the model. Whereas, if the values of ACC of two models are similar, then the model with higher BWT is usually considered.

## C Additional Plots

Figure 2 shows the plots of average accuracy (over 10 tasks) at the end of training on each task. This is different from the plots presented in Figure 1 where we take the average over encountered tasks only. In Figure 3, we show the performance of CLR for various values of hyperparameter $\lambda$ for various protocols. The value of $\lambda$ at the highest/peak point of each protocol's graph was chosen for the corresponding experiment.

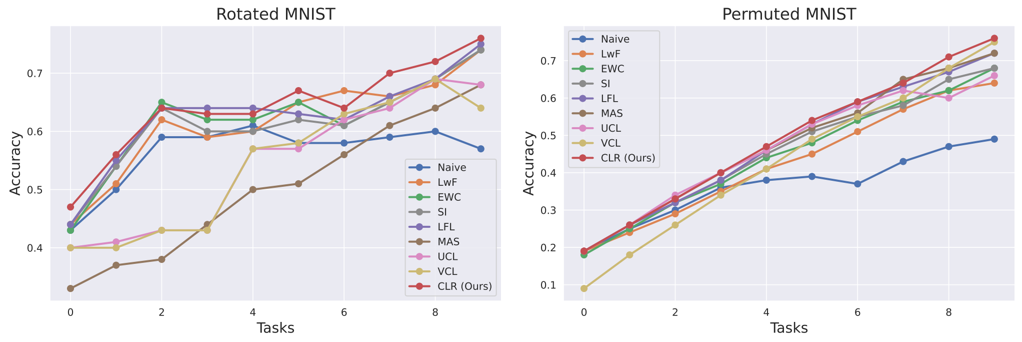

Figure 2: Average test accuracy on all T tasks after training on each task for all methods on Rotated and Permuted MNIST

## D Discriminative Feature Learning with Center Loss

Typical deep neural network architecture comprises an input layer, followed by several hidden layers with non-linear activation functions and the output layer. The output layer generally has a softmax activation function for multi-class classification. This last fully connected layer acts as a linear classifier that separates the deeply learned features produced by the last hidden layer. The softmax loss forces the deep features of different classes to stay apart. The discriminative power of learned features is enhanced if the intra-class compactness and inter-class separability are maximized simultaneously. Though the features learned using the softmax loss are separable, they are not discriminative enough for open-set supervised problems and often exhibit high intra-class variance. This adversely affects the generalization capabilities of neural networks.

Several works (Wen et al., 2016; Deng et al., 2017; Zhang et al., 2017; Liu et al., 2017; Wang et al., 2018b;a; Chen et al., 2017; Wan et al., 2018; Qi & Zhang, 2018) have proposed variants of softmax

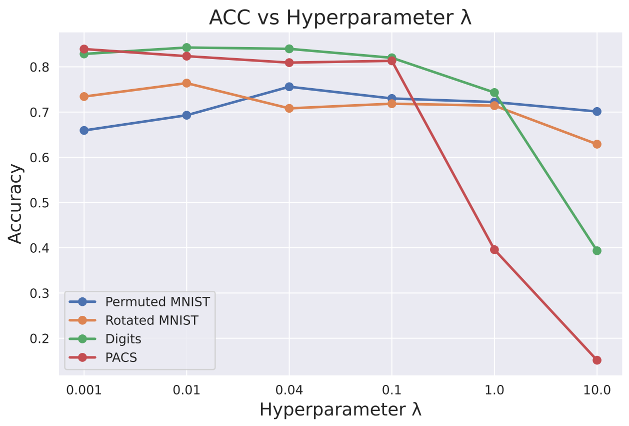

Figure 3: Average test accuracy (ACC) for various values of CLR's hyperparameter $\lambda$ for different protocols

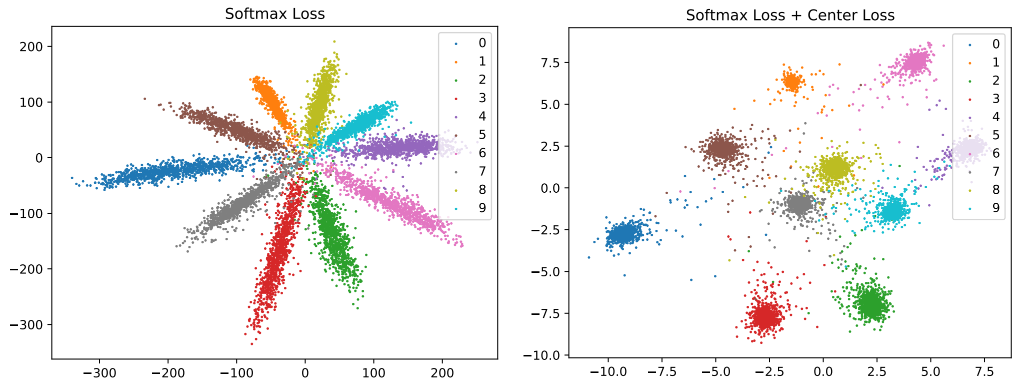

Figure 4: Left: Visualization of features in 2D from a CNN model trained on MNIST dataset LeCun (1998) with Softmax Loss. Right: Visualization of features in 2D from a CNN trained on MNIST dataset with joint supervision of Softmax Loss and Center Loss with $\lambda = 1$

loss to enhance the discriminative power. The siamese network Koch et al. (2015) based approaches which use contrastive loss Sun (2015); Hadsell et al. (2006) and triplet loss Schroff et al. (2015), learn the embeddings directly. These approaches face the problem of semi-hard sample mining and combinatorial explosion in the number of pairs or triplets, which significantly affect the effective model training Deng et al. (2019). There are also angular margin penalty-based approaches that have shown significant improvements over softmax loss and have been explored in various directions, especially for large-scale face recognition Liu et al. (2017); Wang et al. (2018b;a); Liu et al. (2016); Deng et al. (2019).

Wen et al. (2016) introduced the center loss for discriminative feature learning to solve deep face recognition. The joint supervision of softmax loss and center loss is used to obtain the inter-class dispersion and intra-class compactness by simultaneously learning the centers and minimizing the distances between the deep features and their corresponding class centers. The center loss has the same requirement as the softmax loss and needs no complex recombination of the training samples like contrastive loss and triplet loss which suffer from dramatic data expansion. The center loss is defined as follows:

$$L_c(x;\theta,c) = \frac{1}{2}\sum_{i=1}^{m} \|\mathrm{f}_{L-1}(x_i;\theta) - c_{y_i}\|_2^2 \tag{6}$$

In Equation 6, the $x_i$ denotes the $i$th sample, belonging to the $y_i$th class, $c_{y_i} \in R^d$ denotes the $y_i$ th class center of deep features. The size of mini-batch and size of the feature dimension is $m$ and $d$, respectively. $L$ is the total number of layers, and $\mathrm{f}_{L-1}$ is the feature vector of layer $L-1$, which is just before the softmax classifier layer, and the $\theta$ denotes the network parameters.

The formulation effectively characterizes the intra-class variations. In each iteration, the centers are computed by averaging the features of the corresponding classes. The deep features learned using the center loss are highly discriminative, clustered around the corresponding class centers, and linearly separable by the final fully connected layer, which acts as a linear classifier. Figure 4 presents the visualizations of features obtained using softmax loss on the left and using joint supervision of softmax and center loss on the right. Wen et al. (2016) provides detailed analysis and extensive experiments on center loss and its application in discriminative feature learning.