Continual Learning with Weight Interpolation

Jędrzej Kozal
Wrocław University of Science and Technology
Wrocław, Poland
jedrzej.kozal@pwr.edu.pl

Jan Wasilewski
Rochester Institute of Technology
Rochester NY, USA
jw7630@g.rit.edu

Bartosz Krawczyk
Rochester Institute of Technology
Rochester NY, USA
bartosz.krawczyk@rit.edu

Michał Woźniak
Wrocław University of Science and Technology
Wrocław, Poland
michal.wozniak@pwr.edu.pl

Abstract

Continual learning poses a fundamental challenge for modern machine learning systems, requiring models to adapt to new tasks while retaining knowledge from previous ones. Addressing this challenge necessitates the development of efficient algorithms capable of learning from data streams and accumulating knowledge over time. This paper proposes a novel approach to continual learning utilizing the weight consolidation method. Our method, a simple yet powerful technique, enhances robustness against catastrophic forgetting by interpolating between old and new model weights after each novel task, effectively merging two models to facilitate exploration of local minima emerging after arrival of new concepts. Moreover, we demonstrate that our approach can complement existing rehearsal-based replay approaches, improving their accuracy and further mitigating the forgetting phenomenon. Additionally, our method provides an intuitive mechanism for controlling the stability-plasticity trade-off. Experimental results showcase the significant performance enhancement to state-of-the-art experience replay algorithms the proposed weight consolidation approach offers. Our algorithm can be downloaded from https://github.com/jedrzejkozal/weight-interpolation-cl.

1. Introduction

The properties of loss landscape and their effects on training and generalization were objects of study for a long time [13, 27, 43, 45]. Training of neural network is an optimization process in a highly dimensional non-convex parameter space with many local minima and saddle points [37]. Overabundance of local minima may arise due to the overparameterization of neural networks [21]. It was hypothesized that local minima are connected by non-linear paths with a low loss [15]. This property is known as mode connectivity. One feature that may be considered when studying this phenomenon is the permutation invariance of neural networks [12]. Neurons or kernels of network layers can be permuted and, if neighboring layers' outputs and inputs are adjusted, one can obtain a solution that has the same properties as the original model but lies in a completely different part of the loss landscape. Considering this fact, one may conclude that the abundance of local minima in the loss landscape of neural networks results from permutation invariance. In a follow-up work, Ainsworth et al. [1] showed how to find permutations of weights that allow for a linear interpolation of weights with low or even near zero barriers. They also showed that there exist solutions in the loss landscape that cannot be reached by applying permutation to units of a neural network.

Previous experiments on loss barriers were made mostly with the assumption that networks trained from two independent initializations are in two different local minima and have similar loss values [1, 20]. In the case of continual learning, this assumption cannot be met, as models are subject to forgetting [14] of previously seen data. In [30] weight averaging was proposed to mitigate catastrophic forgetting for pretrained models, however, parameter symmetries were not considered. Similarly, authors of [25] utilize weight interpolation to mitigate forgetting in BERT models, but they do not apply weight permutation. Pena et al. [38] propose a new weight interpolation method based on Sinkhorn differentiation, but continual learning is not their primary focus, and the scope of continuous learning experiments is very limited. Authors of [47] introduced a new interpolation method that could be used for models trained with disjoint data distributions, however, they do not carry out continual learning evaluation.

Research goal. We propose a novel approach to continual learning that combines weight interpolation for better consolidation of the network capabilities before and after new tasks become available, with experience replay for enhanced robustness to catastrophic forgetting.

Motivation. The impact of loss landscape properties on continual learning is a very important, yet largely unexplored area [19, 32]. We know that it plays a crucial role in the process of balancing exploration (learning new tasks) and exploitation (retaining previously learned knowledge) [16]. Sudden changes in the loss landscape can cause the model to forget previously learned information, while inhibiting the loss adaptation will hinder the accumulation of the new concepts [35]. The presence of local minima associated with new tasks can interfere with the optimization process for previous tasks, affecting the model's robustness to catastrophic forgetting [36]. Therefore, properly understanding and utilizing loss landscape under the continuous nature of data is of vital importance.

Summary. In this work, we study the potential applications of recent findings from the field of weight interpolation in continual learning [10]. Based on recent weight interpolation techniques, we propose a remarkably simple continual learning algorithm that performs weight interpolation after each task to mitigate forgetting. In this work, we abuse the conjecture about low loss volume being convex modulo permutation symmetries [12], as each task will have separate data distribution and, consequently, different loss landscapes. However, in our theoretical analysis, we show what conditions should be met to increase the chances of finding good weight permutation and successful interpolation.

We base our approach on widely used experience replay methods. Before training with new data from a new task we store network weights. The training with new data is carried out without any changes from standard replay algorithms. After training we utilize weights trained on the current task and stored old weights to perform permutation and then interpolation. The permutation step aligns units of both networks, while interpolations allow for better knowledge consolidation, compared to replay-based algorithms alone. We show that our method can reduce forgetting in several rehearsal-based methods.

Main contributions. This work offers the following contributions to the continual learning domain:

- we show the necessary conditions required for the successful application of weight interpolation to continual learning problems, and verify these claims experimentally;
- we propose novel and simple continual learning algorithm that is compatible with popular rehearsal-based methods;
- we perform an extensive experimental evaluation of the proposed method, showing its potential for significantly boosting the performance of any experience replay algorithm;
- we show that the proposed method has a built-in, intuitive mechanism for controlling stability-plasticity trade-off.

2. Related Works

2.1. Continual Learning

Continual learning [10] is a domain where, instead of a single i.i.d. dataset, we are dealing with a sequence of tasks with different data distributions. Training without access to data from previous tasks may lead to catastrophic forgetting [14] - a phenomenon where neural network's performance on previous tasks degrades rapidly. The performance here could be defined as losing the ability to solve previously learned tasks when a neural network learns to solve a new one. Catastrophic forgetting could lead to dramatic performance deterioration on the previous tasks. In the domain of continual learning, algorithms are typically categorized into three primary groups:

Regularization-based methods aim to control forgetting by modifying the learning process. Elastic Weight Consolidation (EWC) [22] introduces an additional regularization term that constrains the learning of important parameters. Learning without Forgetting (LwF) [28] leverages pseudo-labels derived from classification heads of previous tasks to enhance knowledge retention. Synaptic intelligence [55] is a structural regularizer that enforces penalty on each synapse based on its importance for previous tasks.

Rehearsal-based methods rely on memory buffers to store samples from previous tasks [9]. Gradient Episodic Memory (GEM) [29] utilizes examples from memory to project gradients in directions that minimize loss for previous tasks. Averaged GEM (aGEM) [8] is a refined version that offers computational and memory efficiency. Recent investigations [11] have explored asymmetric update rules and additional classifier updates to address biases introduced by small rehearsal buffers. Moreover, Buzzega et al. [5] stores model logits alongside images and labels in a memory buffer. These logits are subsequently utilized to regulate the model by introducing an additional loss term for knowledge distillation. This method was refined in [3] by recalculating logits over time, segregating loss for new data, and pretraining logits responsible for new tasks. There are also other research directions, such as iCARL [41], where instead of cross-entropy loss, a minimal distance classifier is trained on top of a convolutional neural network. Another interesting and simple algorithm is GDumb [40], which utilizes only greedily stored samples to train the model with the small balanced dataset.

Expansion-based methods involve augmenting the network structure to accommodate shifts in data distribution. Progressive Neural Networks (PNN) [42] add new backbones connected to previous layers to leverage knowledge learned from earlier tasks. Similarly, [26] propose expanding network parameters alongside selective retraining to adapt to new tasks. Authors of [53] expand the model by introducing more convolutional features for new tasks, and they propose a new loss function to train a more diverse set of representations for new data.

2.2. Weight interpolation

Garipov et al. [15] showed that local minima obtained by training with different random weight initialization in the loss landscape are connected by non-linear paths with low loss values. This property was introduced as *Mode Connectivity*. It is also known [4] that there exists a lot of possible weight permutations that give raise to equivalent networks located in completely different fragments of loss landscape. In [49], a new algorithm for finding network permutations and the connection curve between two points in the loss landscape was introduced. Authors of [34] showed that weights of MLP trained from the same initialization can be linearly connected. Entezari et al. [12] suggested that when we consider neural network permutation invariance, solutions found by SGD should be connected by linear path no loss barrier. Indeed, Ainsworth et al. [1] proposed several algorithms for finding permutations of neural networks that allow for linear interpolation between weights with near-zero barrier. REPAIR [20] improved the performance for residual networks on bigger datasets by introducing the recomputation of batch normalization statistics after interpolation.

3. Continual learning with weight interpolation

This work introduces a simple method that could be used as a plugin to enhance the effectiveness of any rehearsal algorithm. The core idea is to interpolate weights of a neural network before and after training with new data. This should allow for better knowledge consolidation and inhibit forgetting. An overview of the proposed method is provided in Fig. 1.

3.1. Notation

In continual learning, we are dealing with stream S, arriving in the form of tasks. Each task t may be represented by a dataset $D_t = \{(x_i, y_i)\}_{i=0}^{n_t}$, where x_i is image, y_i is label, and $n_t = |D_t|$. The goal is to train a neural network f with parameters θ on each task, having access only to the most recent data, i.e., $\min_\theta \mathcal{L}(f(\theta), D_t)$. Rehearsal-based algorithms utilize an additional small buffer for data $\mathcal{M} = \{(x_j, y_j)\}_{j=0}^{m}$ of size $m \ll n_t$ to store data from the previous task and use them to mitigate forgetting.

3.2. Motivation

The main objective of continual learning is the optimization of the joint test loss across all tasks in the stream. We can only access the training data from the current task, but our main goal is to train the network with a low loss across all tasks. Joint loss for all tasks seen so far by the model can be defined as:

$$\mathcal{L}_D(\theta) = \sum_{t=1}^{T} \mathcal{L}(\theta, D_t) \tag{1}$$

where $D = D_1 \cup \cdots \cup D_T$. We can divide this sum into two parts, namely, loss induced by the classes from the last task and all other classes seen before:

$$\mathcal{L}_D(\theta) = \sum_{t=1}^{T-1} \mathcal{L}(\theta, D_t) + \mathcal{L}(\theta, D_T) \tag{2}$$

The first term corresponds to performance on all previous tasks and is mainly affected by forgetting in a continual learning setup. The second term can be directly optimized for, as we have access to data for the task T. Let's define an increase in loss induced by forgetting tasks before T as:

$$\Delta\mathcal{L}_{Fi} = \sum_{t=1}^{i-1} (\mathcal{L}(\theta_i, D_t) - \mathcal{L}(\theta_t, D_t)) \tag{3}$$

The first term inside the sum is the current loss for task i, and the second term is the loss directly after training with data from the same task. This definition is analogous to forgetting measure [7] - a commonly used metric designed for evaluation of accuracy decrease during continual training. By plugging Eq. (3) into Eq. (2) we can rewrite joint loss function after task T as:

$$\mathcal{L}_D(\theta_T) = \sum_{t=1}^{T} \mathcal{L}(\theta_t, D_t) + \Delta\mathcal{L}_{Fi=T} \tag{4}$$

Therefore, loss obtained by the network depends on two factors: (i) how well the network can fit current data, which corresponds to plasticity; and (ii) how well the network handles the previously seen data, which corresponds to forgetting. The second network used for interpolation is the one trained on the previous task $T-1$. We can make the same argument about loss being dependent on plasticity and forgetting/

When we search for good candidates for interpolation between θ_{T-1} and θ_T we require both $\mathcal{L}_D(\theta_T)$ and $\mathcal{L}_D(\theta_{T-1})$ to be low. This is because we must have solutions either in the local minima or close to some local basin. As shown by Eq. (4), this can achieved only when both plasticity is high and forgetting is low. If that is not the case, then the loss term induced by any of these terms could increase the overall loss value, moving away the solution in

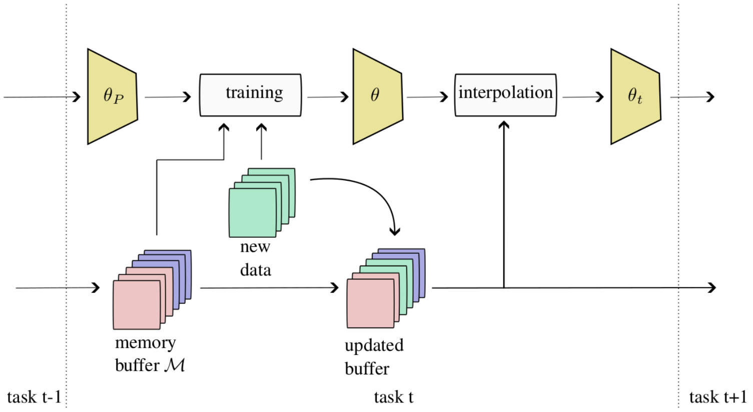

Figure 1. Continual learning with weight interpolation.

the loss landscape from the locally connected modes. Interpolation with weight permutation alone can, in principle, align the activations of the networks trained on the different tasks, so there could be a gain in accuracy directly after interpolation. Still, if the activations learned by the network on the new task are completely different from the previous ones due to forgetting, then alignment between activations can be inaccurate. On the other hand, if there is no plasticity, then alignment could be easier, but there would be no significant difference between the two sets of activations.

For this reason, we conclude that weight interpolation should not be used as a sole source of forgetting prevention in continual learning. Weight interpolation could be used in tandem with other continual learning algorithms that do not limit network plasticity too much. To further justify this claim, we verify experimentally in the appendix 7 that using interpolation without rehearsal does not yield good results.

3.3. Weight interpolation with memory buffer

For each task $t > 0$, we perform weight interpolation of previously trained weights θ_P with the newest parameters trained with current data distribution θ. First, we find the weight permutation π that aligns the activations of θ_P and θ as in [20] (for more details about interpolation and REPAIR algorithms, please refer to Sec. 9). We utilize memory buffer $\mathcal{M}$ to obtain activations of θ and θ_P and update batch normalization statistics. Please note that if we use reservoir sampling during training to update the buffer with new data, the buffer will contain the data from all previous and current tasks. For this reason, during the evaluation of activations for permutation, all previously seen data will be considered, including data from the latest task. We apply the permutation to network parameters and carry out linear interpolation of weights:

$$\theta = (1 - \alpha)\theta + \alpha\pi(\theta_P) \tag{5}$$

where α is a hyperparameter of our algorithm. We provide the pseudocode of our method in Algorithm 1. The function *calc_permutation* is responsible for obtaining permutation of θ_P that aligns activations of θ_P with θ. The function *update_batchnorm* updates the batch normalization layers statistics after interpolation. The proposed method is compatible with most rehearsal-based algorithms and may be used as a plugin for existing or future methods for improving their performance.

Algorithm 1 Continual Learning with Weight Interpolation (CLeWI)

```
**Require:** $S = \{D_1, D_2, ...\}$ - stream with tasks, $f(\theta)$ - network, $\mathcal{M}$ - memory buffer, $\alpha$ - interpolation coefficient
$t \leftarrow 0$
while $D_t$ arrives do
    for $x, y \sim D_t$ do
        $\mathcal{L} \leftarrow \sum_{x,y} \mathcal{L}(f(x, \theta), y)$
        $x_m, y_m \leftarrow \mathcal{M}$
        $\mathcal{L}_{\mathcal{M}} \leftarrow \sum_{x_m,y_m} \mathcal{L}(f(x_m, \theta), y_m)$
        $\theta \leftarrow \theta - \lambda\nabla_\theta(\mathcal{L} + \mathcal{L}_{\mathcal{M}})$
        resevoir_sampling $(\mathcal{M}, x, y)$
    end for
    if $t > 0$ then
        $\pi \leftarrow$ calc_permutation$(\theta, \theta_P, \mathcal{M})$
        $\theta \leftarrow (1 - \alpha)\theta + \alpha\pi(\theta_P)$
        $\theta \leftarrow$ update_batchnorm$(\theta, \mathcal{M})$
    end if
    $\theta_P \leftarrow \theta$
    $t \leftarrow t + 1$
end while
```

4. Experiment setup

Our experiments compare the performance of commonly used rehearsal algorithms with and without weight interpolation applied after each task. This evaluation mode, similar to the ablation study, should allow for an easy verification of our theoretical claims made in the previous section. We also provide in-depth analysis of the weight interpolation impact on the overall performance, and stability-plasticity dilemma. We also evaluate the impact of the training with increased model width on the results.

Baselines. In this work, we have used the following baselines:

- joint - training with cumulative datasets over all tasks, with full access to previous data. It is upperbound on the continual learning performance.
- finetuning - training with standard SGD optimization, with no consideration for forgetting. It is lowerbound of performance
- online Elastic Weight Consolidation (oEWC) [46] - an extension of existing EWC method [22], that use both regularisation and knowledge distillation to prevent forgetting.
- Synaptic Inteligence [56] - regularisation method that determines the importance of network parameters.
- Incremental Classifier and Representation Learning (iCARL) [41] - method that replaces cross entropy with prototype-based learning
- GDumb [39] - Greedily stores samples in memory and trains model only with balanced dataset
- Experience Replay (ER) [9] - simplest rehearsal method that stores samples in the buffer using reservoir sampling and samples data from the buffer to train with it alongside data new from a new task
- averaged Gradient Episodic Memory (aGEM) [8] rehearsal method, that projects gradient onto direction, that prevents forgetting
- Experience Replay with Asymmetric Cross-Entrop (ER-ACE) [6] - eliminates representation overlap of new classes and old ones from the buffer by changing the loss function
- Maximally Interfered Retrieval (MIR) [2] - method with the buffer that uses virtual gradient update to select useful samples for rehearsing
- Bias Correction (BIC) [52] - a method that introduces several parameters for correction of bias in the last fully connected layer of the network
- Dark Experience Replay (DER++) [5] - method that combines rehearsal with knowledge distillation [18]

Datasets. In this work, we consider only the class-incremental scenario [50] and utilize standard continual learning benchmarks obtained by splitting classes into several tasks. We use Cifar10 [24], Cifar100 [24], and Tiny ImageNet [51] datasets, with 5, 10, and 20 tasks, respectively. We shuffle class order in tasks based on random seeds.

Metrics. We use three evaluation metrics. The test set accuracy averaged over all tasks after finished training, defined as $Acc = \frac{1}{K}\sum_{t}^{K} \frac{1}{n_t}\sum_{i=1}^{n_t} \mathbb{1}[f(x_i, \theta_K) = y_i]$, where K is the number of tasks, and $\mathbb{1}$ is an indicator function. The test set accuracy for classes from the last task $Acc_K = \frac{1}{n_K}\sum_{(x_i,y_i)\in D_K} \mathbb{1}[f(x_i, \theta_K) = y_i]$, and forgetting measure (FM) [7] defined as average difference between maximum accuracy, and final accuracy for given task.

Evaluation details. For all datasets, we use ResNet18 architecture [17] with a changed number of filters in the first layer following [29]. For all rehearsal-based methods, we use a buffer of size 500. Whenever possible, we use the best hyperparameters reported by the authors of corresponding papers. In other cases, we performed a search of hyperparameters for the seq-cifar100 benchmark and used those values for other datasets as well. This shortcut has been made due to limitations in computational power availability. All experiments were implemented using Mammoth library [5]. We made our code available online[1].

5. Results

5.1. Evaluation with standard benchmarks

We perform an experimental evaluation of the proposed method, following the steup described in the previous section. The results are presented in Tab. 1.

In most cases, we may see that the proposed method improves the average accuracy on all tasks and leads to better task retention, as depicted by reducing the forgetting measure. The biggest gains in accuracy can be observed for simpler forms of replay, such as ER and MIR. CLeWI obtains the best accuracy when combined with these methods. Other methods, such as ER-ACE, BIC, or DER++, can also benefit from applying interpolation. However, the final average accuracy after training is lower compared to simpler methods. At the same time, the forgetting rate of these methods is lower than that of others. These methods limit the plasticity of the networks. In the interpolation process, we are losing some of the performance for the newest task at the cost of forgetting mitigation. For this reason, after applying interpolation, the methods that obtain lower forgetting on their own can sometimes obtain lower accuracy and forgetting measure. These results are in line with our theoretical analysis. Low plasticity can contribute to high overall loss and, in consequence, make interpolation harder.

We noted a decrease in performance for ER-ACE, where average accuracy is lower, but forgetting measure still improves, and BIC on the Cifar100 dataset, where both accuracy and forgetting measure are worse. This is in line with our previous analysis, as these methods introduce strong inductive bias and obtain higher accuracy compared to ER. CLeWI, when combined with these methods, inherits this bias, and therefore, average accuracy can decrease.

[1] https://github.com/jedrzejkozal/weight-interpolation-cl

Table 1. Average accuracy and forgetting measure averaged over 5 runs for cifar10, cifar100, and tinyimagenet datasets.

method	Cifar10(T=5)		Cifar100(T=10)		Tiny-ImageNet(T=20)	
	Acc(↑)	FM(↓)	Acc(↑)	FM(↓)	Acc(↑)	FM(↓)
Joint	91.79±0.36	0.0±0.0	70.54±0.75	0.0±0.0	58.34±0.24	0.0±0.0
Finetuning	19.37±0.32	77.77±0.85	9.07±0.1	80.57±0.41	3.92±0.27	74.75±1.34
oEWC	17.21±2.89	69.94±3.98	8.86±0.51	76.05±0.43	3.71±0.23	70.14±1.6
SI	19.28±0.4	78.11±0.38	6.36±0.53	36.99±1.32	3.64±0.4	67.97±2.1
iCARL	58.98±1.21	25.27±4.72	46.91±0.66	25.56±0.57	19.69±0.37	20.24±0.54
GDumb	39.7±1.57	0.66±0.65	9.99±0.68	0.0±0.0	3.2±0.31	0.22±0.15
ER	53.22±2.98	44.02±3.59	22.45±1.26	65.59±1.07	6.44±0.38	75.88±0.23
CLeWI+ER	62.8±2.31(+9.58)	31.8±2.61(-12.22)	40.31±1.08(+17.86)	12.81±0.79(-52.78)	11.68±0.45(+5.24)	66.82±0.49(-9.06)
aGEM	21.88±1.15	75.63±0.96	9.17±0.18	80.33±0.34	3.62±0.54	73.61±3.29
CLeWI+aGEM	34.74±4.05(+12.86)	4.16±1.92(-71.47)	22.75±1.41(+13.58)	39.07±2.26(-41.26)	6.8±0.4(+3.18)	60.22±1.17(-13.39)
ER-ACE	70.63±1.15	10.11±0.95	37.75±1.23	35.15±1.33	15.98±1.64	42.47±2.43
CLeWI+ER-ACE	64.22±2.5(-6.41)	4.84±0.67(-5.27)	36.97±0.55(-0.78)	17.72±0.76(-17.43)	19.15±0.72(+3.17)	19.7±0.88(-22.77)
MIR	48.17±3.23	49.02±3.72	21.96±1.13	66.07±0.95	6.25±0.41	76.06±0.3
CLeWI+MIR	73.06±0.74(+24.89)	6.71±0.84(-42.31)	40.06±0.84(+18.10)	13.54±0.54(-52.53)	19.75±0.56(+13.50)	25.47±0.43(-50.59)
BIC	69.63±2.28	22.04±3.04	37.55±1.64	44.42±1.87	7.09±0.78	71.47±0.87
CLeWI+BIC	51.15±9.53(-18.48)	26.48±4.96(+4.44)	39.46±1.34(+1.91)	33.46±1.46(-10.96)	7.35±1.4(+0.26)	65.34±0.79(-6.13)
DER++	70.13±1.16	21.11±1.46	36.64±1.59	48.06±2.62	13.52±1.53	55.68±4.36
CLeWI+DER++	71.82±2.11(+1.69)	11.21±2.16(-9.90)	38.16±1.86(+1.52)	14.32±1.95(-33.74)	16.61±0.87(+3.09)	25.17±6.34(-30.51)

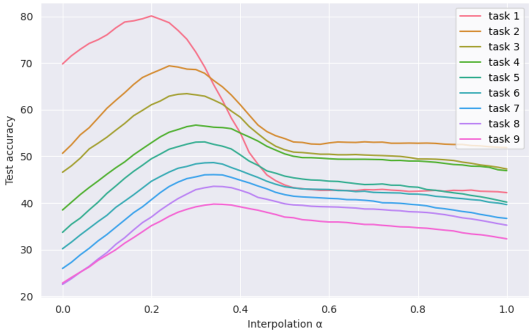

Figure 2. The effect of the α parameter (Eq. (5)) on the test set accuracy for all tasks. Interpolation with smaller values of α allows for obtaining weights that are closer in loss landscape to the current task, while increasing α means more weights are carried over from previous tasks.

5.2. Impact of weight interpolation

To illustrate the influence of interpolation hyperparameter α on obtained results we plot accuracy for different interpolation α and all tasks. We use all the classes the model has seen for each task. This means that the older model will always obtain worse performance, as it has not seen the classes from the latest task. Results are presented in Fig. 2. At the beginning of training, better overall accuracy is obtained after training with a new task compared to the model weights before training. This is probably due to underfitting on the first tasks caused by a small number of learning examples in each task. Over the course of training, the difference in accuracy between these two models falls quickly. After a few tasks, the old model performs better, while the new one suffers from forgetting. The interpolation plot for continual learning is asymmetrical. Interpolating models closer to the model trained on a new task gives better accuracy. This is probably due to the longer training of the model with data from the new task.

Table 2. Average accuracy, accuracy for the last task, and forgetting measure averaged over 3 runs for different values of interpolation α.

interpolation coefficient	Acc(↑)	Acc_K(↑)	FM(↓)
α=0.1	27.6±0.46	87.77±2.09	59.11±0.66
α=0.2	34.75±0.51	83.87±2.57	47.14±0.23
α=0.3	39.95±0.67	72.23±3.32	30.67±0.55
α=0.4	42.01±0.82	44.6±4.78	18.9±0.24
α=0.5	40.26±1.25	16.27±4.47	12.61±0.96

5.3. Stability-plastisity dilemma

We show that interpolation hyperparameter α allows for direct control of the plasticity-stability dilemma by running additional experiments with multiple values of this hyperparameter. The results are presented in Tab. 2. With higher α (interpolation closer to the old model), the model is prone to remembering the older tasks. This can be directly observed by looking at forgetting measures. Higher α usage promotes stability and limits performance for the current task. With smaller α (interpolation closer to a newer model), the network archives better accuracy on the last task at the price of higher forgetting. This simple mechanism could be useful for controlling the learning properties of neural networks. Interpolation α may also be changed during training with multiple tasks to adapt to the changing dynamics of the learning environment.

Comparing these results to Fig. 2, one can notice that the best test accuracy was obtained for a value of α that is not aligned with the local maximum of the interpolation plot. This suggests that selecting α only to optimize the performance on tasks seen so far is a misleading approach that can lead to lower accuracy at the end of training.

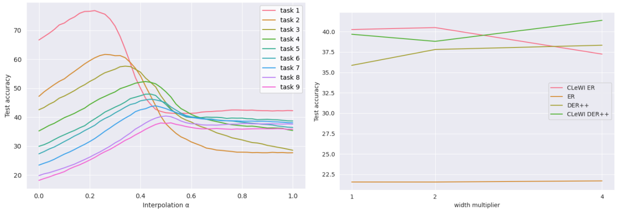

Figure 3. Impact of increasing the network width on the accuracy barrier and continual learning performance. (Left) the interpolation plot for the WideResNet with width multiplier $= 4$. (Right) test accuracy for split-Cifar100 benchmark as a function of ResNet width.

5.4. Wider networks

It has been reported that interpolation works better when network architecture has more filters [1, 20]. Also, recent studies suggest that wider architectures could lead to improved performance in continual learning [33]. For this reason, we have carried out additional experiments with WideResNets [54] on the split-Cifar100 benchmark. We kept the same hyperparameter setting, only the width was changed. The results are presented in Fig. 3

On the left-hand side, we present the interpolation plot for the WideResNet with an increased number of convolutional filters by 4. We may see that compared to results from Fig. 2, the localization of the local maximum accuracy shifts more dynamically during training. It is also worth noting that in Fig. 2, the accuracy for α close to 1 decreases slightly. This is not the case for a wider network, where the accuracy on the plot is mostly flat for $\alpha > 0.7$. This suggests that wider networks are indeed better at preserving previously gained knowledge. However, this does not translate well into overall continual learning performance when using CLeWI due to dynamic changes in the shape of interpolation plot curves for different tasks. We hypothesize that these changes arise due to the small amount of training data for WideResNet in a single task of Cifar100 benchmark. However, we are aware that experiments with other benchmarks, such as split-ImageNet, could provide different results and further investigations are needed.

The right-hand side shows the test accuracy in the function of ResNet width. We may see that increasing the width could significantly improve the performance of DER++, but even with this improvement, CLEWI-DER++ with standard width obtains better performance. At the same time, increasing the width of the backbone for the CLeWI ER decreases accuracy. The small amount of training data in each task may be the cause behind this phenomenon. Overfitting may occur when we increase the network's capacity but keep the same amount of training data. The dynamic changes in local maxima of the interpolation plots for increased width are in line with this explanation. For the first task, local maxima are obtained for smaller α - corresponding to interpolation closer to the newer model. All these information suggest that experience replay alone is insufficient when training networks with larger capacity. When stronger forgetting prevention mechanisms are introduced, such as DER++, the performance of CLeWI further improves with increased width. This shows that our algorithm is a versatile approach to boosting the performance of CL methods due to the ease of combining CLeWI with other forms of rehearsal. The proposed method can be easily adjusted to other settings by combining it with the form of rehearsal that works well in a given scenario. This shows the flexibility of CLeWI and its strength as a low-risk, low-cost plugin for existing methods.

6. Conclusion

Summary. We proposed a simple algorithm, compatible with most of the rehearsal-based continual learning methods that can significantly boosts their performance and improve robustness to catastrophic forgetting. CLeWI introduces only a single additional hyperparameter that allows for direct control of the stability-plasticity dilemma. The experiments suggested that α selection should be carried out with great care, as local maxima for the current task not necessarily align well with higher accuracy for all tasks in the training stream. In the interpolation plots, we may see that local maxima's location can shift over time. In earlier tasks, the maxima occur for lower values of α, probably due to too small amount of training data in each task. Experiments with bigger datasets could provide more insight here, as we hypothesise, that with enough data in the first task, the location of local minima in the interpolation plot will be more stable.

Limitations. Storing a second copy of model weights in memory can be prohibitive for large models. For example, when training 1.4B parameter transformer storing previous model state could be too costly. Additional memory requirements may also be prohibitive in the memory-scarce area of edge computing. We carried out additional experiments (see Sec. 10 in appendix) that take into consideration memory usage. We found settings where using weight interpolation over increasing buffer size alone can be beneficial.

Future works. Future work will focus on exploring of the weight interpolation should be performed after every new task, or would a selective mechanism deciding when to perform interpolation lead to more robust results. Furthermore, we will explore the potential of using CLeWI as a part of concept drift adaptation mechanisms [23] and study the possibilities of extending it for other computer vision tasks, such as object detection or continual segmentation.

Acknowledgment

This work is supported by the CEUS-UNISONO programme, which has received funding from the National Science Centre, Poland under grant agreement No. 2020/02/Y/ST6/00037.

References

[1] Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries, 2023. 1, 3, 7

[2] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *CoRR*, abs/1908.04742, 2019. 5

[3] Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended der-verse. *CoRR*, abs/2201.00766, 2022. 2

[4] Johanni Brea, Berfin Simsek, Bernd Illing, and Wulfram Gerstner. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *CoRR*, abs/1907.02911, 2019. 3

[5] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems*, pages 15920–15930. Curran Associates, Inc., 2020. 2, 5

[6] Lucas Caccia, Rahaf Aljundi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. Reducing representation drift in online continual learning. *CoRR*, abs/2104.05025, 2021. 5

[7] Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. *CoRR*, abs/1801.10112, 2018. 3, 5

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. *CoRR*, abs/1812.00420, 2018. 2, 5

[9] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486, 2019. 2, 5

[10] Zhiyuan Chen, Bing Liu, Ronald Brachman, Peter Stone, and Francesca Rossi. *Lifelong Machine Learning*. Morgan & Claypool Publishers, 2nd edition, 2018. 2

[11] Aristotelis Chrysakis and Marie-Francine Moens. Online bias correction for task-free continual learning. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[12] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. 1, 2, 3

[13] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization, 2021. 1

[14] Robert French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135, 1999. 1, 2

[15] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018. 1, 3

[16] Yanan Gu, Xu Yang, Kun Wei, and Cheng Deng. Not just selection, but exploration: Online class-incremental continual learning via dual view consistency. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 7432–7441. IEEE, 2022. 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 5, 2

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 5

[19] Zhongzhan Huang, Mingfu Liang, Senwei Liang, and Wei He. Altersgd: Finding flat minima for continual learning by alternative training. *CoRR*, abs/2107.05804, 2021. 2

[20] Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. Repair: Renormalizing permuted activations for interpolation repair, 2022. 1, 3, 4, 7, 2

[21] Kedar Karhadkar, Michael Murray, Hanna Tseran, and Guido Montúfar. Mildly overparameterized relu networks have a favorable loss landscape, 2024. 1

[22] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. 2, 5

[23] Lukasz Korycki and Bartosz Krawczyk. Class-incremental experience replay for continual learning under concept drift. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops*, pages 3649–3658, 2021. 7

[24] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 5

[25] Lisa Kühnel, Alexander Schulz, Barbara Hammer, and Juliane Fluck. Bert weaver: Using weight averaging to enable lifelong learning for transformer-based models in biomedical semantic search engines, 2023. 1

[26] Jeongtae Lee, Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *CoRR*, abs/1708.01547, 2017. 3

[27] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *CoRR*, abs/1712.09913, 2017. 1

[28] Zhizhong Li and Derek Hoiem. Learning without forgetting. *CoRR*, abs/1606.09282, 2016. 2

[29] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 2, 5

[30] Imad Eddine Marouf, Subhankar Roy, Enzo Tartaglione, and Stéphane Lathuilière. Weighted ensemble models are strong continual learners, 2024. 1

[31] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. *CoRR*, abs/1710.03740, 2017. 3

[32] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Dilan Görür, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. *CoRR*, abs/2010.04495, 2020. 2

[33] Seyed-Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Görür, and Mehrdad Farajtabar. Architecture matters in continual learning. *CoRR*, abs/2202.00275, 2022. 7

[34] Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. 3

[35] Kaustubh Olpadkar and Ekta Gavas. Center loss regularization for continual learning. *CoRR*, abs/2110.11314, 2021. 2

[36] Dongmin Park, Seokil Hong, Bohyung Han, and Kyoung Mu Lee. Continual learning by asymmetric loss approximation with single-side overestimation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 3334–3343. IEEE, 2019. 2

[37] Razvan Pascanu, Yann N. Dauphin, Surya Ganguli, and Yoshua Bengio. On the saddle point problem for non-convex optimization. *CoRR*, abs/1405.4604, 2014. 1

[38] Fidel A. Guerrero Peña, Heitor Rapela Medeiros, Thomas Dubail, Masih Aminbeidokhti, Eric Granger, and Marco Pedersoli. Re-basin via implicit sinkhorn differentiation, 2022. 1

[39] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 524–540. Springer, 2020. 5

[40] Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision – ECCV 2020*, pages 524–540, Cham, 2020. Springer International Publishing. 2

[41] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *CoRR*, abs/1611.07725, 2016. 2, 5

[42] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016. 2

[43] Levent Sagun, Léon Bottou, and Yann LeCun. Singularity of the hessian in deep learning. *CoRR*, abs/1611.07476, 2016. 1

[44] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. 2

[45] Adepu Ravi Sankar, Yash Khasbage, Rahul Vigneswaran, and Vineeth N. Balasubramanian. A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. *ArXiv*, abs/2012.03801, 2020. 1

[46] Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning, 2018. 5

[47] George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training, 2024. 1

[48] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. 3

[49] N. Joseph Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing mode connectivity via neuron alignment. *CoRR*, abs/2009.02439, 2020. 3

[50] Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *CoRR*, abs/1904.07734, 2019. 5

[51] Jiayu Wu. Tiny imagenet challenge. 2017. 5

[52] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Raymond Fu. Large scale incremental learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 374–382, 2019. 5

[53] Shipeng Yan, Jiangwei Xie, and Xuming He. DER: dynamically expandable representation for class incremental learning. *CoRR*, abs/2103.16788, 2021. 3

[54] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. 7

[55] Friedemann Zenke, Ben Poole, and Surya Ganguli. Improved multitask learning through synaptic intelligence. *CoRR*, abs/1703.04200, 2017. 2

[56] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017. 5