Reset It and Forget It: Relearning Last-Layer Weights Improves Continual and Transfer Learning

Lapo Frati[a,1,2], **Neil Traft**[a,1,2], **Jeff Clune**[b,c,d] **and Nick Cheney**[a]

[a]University of Vermont
[b]University of British Columbia
[c]Vector Institute
[d]Canada CIFAR AI Chair

Abstract. This work identifies a simple pre-training mechanism that leads to representations exhibiting better continual and transfer learning. This mechanism—the repeated resetting of weights in the last layer, which we nickname "zapping"—was originally designed for a meta-continual-learning procedure, yet we show it is surprisingly applicable in many settings beyond both meta-learning and continual learning. In our experiments, we wish to transfer a pre-trained image classifier to a new set of classes, in few shots. We show that our zapping procedure results in improved transfer accuracy and/or more rapid adaptation in both standard fine-tuning and continual learning settings, while being simple to implement and computationally efficient. In many cases, we achieve performance on par with state of the art meta-learning without needing the expensive higher-order gradients by using a combination of zapping and sequential learning. An intuitive explanation for the effectiveness of this zapping procedure is that representations trained with repeated zapping learn features that are capable of rapidly adapting to newly initialized classifiers. Such an approach may be considered a computationally cheaper type of, or alternative to, meta-learning rapidly adaptable features with higher-order gradients. This adds to recent work on the usefulness of resetting neural network parameters during training, and invites further investigation of this mechanism.

1 Introduction

Biological creatures display astounding robustness, adaptability, and sample efficiency; while artificial systems suffer "catastrophic forgetting" [23, 14] or struggle to generalize far beyond the distribution of their training examples. It has been observed in biological systems that repeated exposure to stressors can result in the evolution of more robust phenotypes [28, 8]. However, it is not clear what type of stressor during the training of a neural network would most effectively and efficiently convey robustness and adaptability to that system at test time.

It is common practice to have the training of a machine learning system mimic the desired use-cases at test time as closely as possible. In the case of an image classifier, this would include drawing independent training samples from a distribution identical to the test set (i.i.d. training). When the test scenario is *itself* a learning process—such as few-shot transfer learning from a limited number of novel
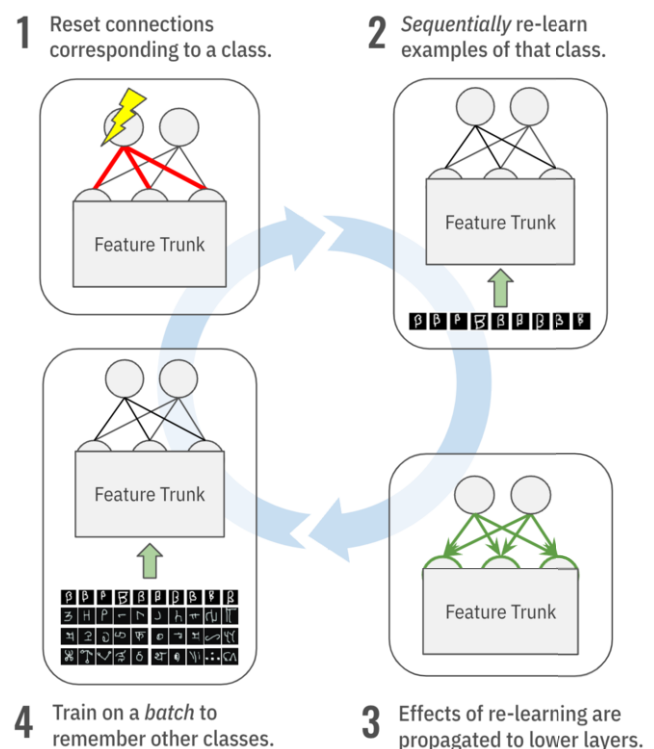

Figure 1. *Alternating Sequential and Batch* learning (ASB) alternates between phases of (Step 2) individual examples from a single class, and (Step 4) multi-class batches of examples. Before each sequential phase the existing class is forgotten by ⚡ zapping.

examples—training can include repeated episodes of rapid adaptation to small subsets of the whole dataset, thereby mimicking the test scenario. Meta-learning algorithms (also known as learning-to-learn) for such contexts are able to identify patterns that generalize across individual learning episodes [29, 4, 16, 27, 10].

Going even further, we are interested in challenging scenarios of *online/continual learning with few examples per class*. We ask, if this is the test scenario, then what kind of training would best mimic this? What kind of stressor could be applied *in pre-training* to confer robustness *upon deployment* into the difficult setting of few-shot continual learning?

---

[1] Equal contribution.
[2] Corresponding authors. Email: {lfrati,ntraft}@uvm.edu.

Recent work by Javed and White [18] and Beaulieu et al. [3] has tackled this question with the application of meta-learning. When the learning process itself is differentiable, one way to perform meta-optimization is by differentiating through several model updates using second-order gradients, as is done in Model-Agnostic Meta-Learning (MAML) [10]. When applied to episodes of continual learning, this approach induces an implicit regularization (it penalizes weight changes that are disruptive to previously learned concepts, without needing a separate heuristic to quantify the disruption).

The authors of Online-Aware Meta-Learning (OML) [18] divide their architecture into two parts, where later layers are updated in a fast inner loop but earlier layers are only updated in a slower outer meta-update. Subsequently, A Neuromodulated Meta-Learning Algorithm (ANML) [3] restructured the OML setup by moving the earlier meta-only layers into a parallel neuromodulatory path which meta-learned an attention-like context-dependent multiplicative gating, achieving state of the art performance in sequential transfer.

However, in this work we show that neither the asymmetric training of different parts of the network (like OML), nor a neuromodulatory path (like ANML), nor even expensive second-order updates (like both), are necessary to achieve equivalent performance in this setting. Instead, we reveal that the key contribution of the OML-objective appears to be a previously overlooked mechanism: *a weight resetting and relearning procedure*, which we call "zapping". This procedure consists of frequent random resampling of the weights leading into one or more output nodes of a deep neural network, which we refer to as "zapping" those neurons.[3]

After reinitializing the last layer weights, the gradients for the weights in upstream layers quantify how the representation should have been different to reduce the loss *given a new set of random weights*. This is exactly the situation that the representation will be presented with during transfer. In the case where all weights of the last layer are reset, zapping closely matches the common technique of transfer learning by resetting the classifier layer(s) on top of pre-trained feature extraction layers (as in Yosinski et al. [34]). Over multiple repetitions, this leads to representations which are more suitable for transfer.

We are able to show the efficacy of this forgetting-and-relearning by introducing a variation which rehearses a continual learning process *without* meta-learning. We call this *Alternating Sequential and Batch* learning (ASB; Figure 1). This process runs through episodes of continual learning just like OML/ANML, but does not perform meta-optimization. However, it does zap a neuron at the beginning of each continual learning episode, giving the model a chance to re-learn the forgotten class. We hypothesize that this better prepares the model for a similar learning process at transfer time.

We show that:

- The *zapping* forget-and-relearn mechanism accounts for the majority of the meta-learning improvements observed in prior work (Section 3.1).
- Dedicating second-order optimization paths for certain layers doesn't explain the performance of meta-learning for continual transfer (Section 3.1).
- *Alternating Sequential and Batch* (ASB), with zapping, is often sufficient to match or outperform meta-learning without any expensive bi-level optimization (Sections 3.1 and 3.2).

---

[3] In OML, weight resampling was employed primarily as a method for maintaining large meta-gradients throughout meta-training and was deemed non-essential (Javed and White [18, V1–Appendix A.1: Random Reinitialization]).

- Representations learned by models pre-trained using zapping are better for general transfer learning, not just continual learning (Section 3.2).
- Zapping and ASB can be useful across different datasets and model architectures (Section 3.3).

Source code for our methods are available at: github.com/uvm-neurobotics-lab/reset-it-and-forget-it

2 Methods

As described in Javed and White [18] and Beaulieu et al. [3], we seek to train a model capable of learning a large number of tasks $\mathcal{T}_{1..n}$, in a few shots per task, with many model updates occurring as tasks are presented *sequentially*. Tasks $\mathcal{T}_i$ come from a common domain $\mathcal{D}$. In our experiments we consider the domain to be a natural images dataset and tasks to be individual classes $\mathcal{C}_i$ in that dataset.

Recent works show that reinitialization of some layers during training can be used as a regularization method [37, 21, 1, 38, 5]. But there are several ways in which this reinitialization can be applied. We focus our attention on the last fully connected layer, right before the output. While the information value within this last layer may be marginal [17, 36] interventions in the last layer will affect the gradient calculation of all the other layers in the network during backpropagation (Figure 1, Step 3).

For consistency with prior approaches, our primary experiments employ the model architecture from Beaulieu et al. [3] (minus the neuromodulatory path). This model consists of a small convolutional network with two main parts: a stack of convolutional layers that act as a feature extractor, and a single fully connected layer that acts as a linear classifier using the extracted features (see Appendix D [13]). We then extend our findings on the larger VGG architecture [31] to demonstrate the scalability of our approach.

Our "zapping" procedure consists of re-sampling all the connections corresponding to one of the output classes. Because of the targeted disruption of the weights, the model suddenly forgets how to map the features extracted by previous convolutional layers to the correct class. To recover from this sudden change, the model is shown examples from the forgotten class, one at a time. By taking several optimization steps, the model recovers from the negative effect of zapping. This procedure constitutes the *inner loop* in our meta-learning setup and is followed by an *outer loop* update using examples from all classes [18]. The outer loop update is done with a higher-order gradient w.r.t. the initial weights of that inner loop iteration [10]. But, as we will see, these inner and outer updates do not actually need to be performed in a nested loop—they can also be arranged in a flat sequence without meta-gradients, yielding similar performance with much more efficient training.

2.1 Training Phases

Since we want to learn each class in just a few shots, it behooves us to start with a pre-trained model rather than starting tabula rasa. Therefore our set-up involves two stages. Within each stage, we examine multiple possible configuration options, described in more detail in the next sections.

1. (Sec. 2.1.1) **Pre-Training:** We use one of the following algorithms to train on a subset of classes: (1) standard *i.i.d.* pre-training, (2) alternating sequential and batch learning (*ASB*), or (3) meta-learning through sequential and batch learning (*meta-ASB*). Each

of these may or may not include the **zapping** procedure to forget and relearn.

2. (Sec. 2.1.2) **Transfer:** Following pre-training, we transfer the classifier to a separate subset of classes using (1) *sequential transfer* (continual learning) or (2) standard *i.i.d. transfer* (fine-tuning).

2.1.1 Stage 1: Pre-Training

Our pre-training algorithm is described in Algorithm 1, and visualized in Appendix E [13]. Our algorithm is based on the "Online-aware" Meta-Learning (OML, Javed and White [18]) procedure, which consists of two main components:

- **Adapting** (*inner loop; sequential learning*): In this phase the learner is sequentially shown a set of examples from a single random class and trained using standard SGD. The examples are shown one at a time, so that the optimization performs one SGD step per image.
- **Remembering** (*outer loop; batch learning*): After each adapting phase, the most recent class plus a random sample from all classes are used to perform a single outer-loop batch update. Those samples serve as a proxy of the true meta-loss (learning new things without forgetting anything already learned). The gradient update is taken w.r.t. the initial inner-loop parameters, and those updated initial weights are then used to begin the next inner-loop adaptation phase following the MAML paradigm [10].

Compared to OML we use a different neural architecture (which improves classification performance; see Appendix D [13]), and do not draw a distinction between when to update the feature extraction vs. classification layers (updating both in the inner and outer loops). Furthermore, while the original OML procedure included both zapping and higher-order gradients, we ablate the effect of each component by allowing them to be turned on/off as follows.

In configurations with **zapping** (denoted as **zap** in Algorithm 1), prior to each sequential adaption phase on a single class $\mathcal{C}_i$, the final layer weights corresponding to that class are re-initialized—in other words, they are re-sampled from the initial weight distribution[4]. We call this procedure zapping as it destroys previously learned knowledge that was stored in those connections.

In the **meta-learning** conditions (denoted as **meta** in Algorithm 1), the "remembering" update is performed as an *outer*-loop meta-update w.r.t. the initial weights of each inner-loop (as described above), and these meta-updated weights are then used as the starting point of the next inner-loop. However, we also wish to examine the effect of zapping independent of meta-learning, and introduce a new pre-training scenario in which we *alternate* between the adapting phase and the remembering phase. Different from meta-learning, this new method does not backpropagate through the learning process nor rewind the model to the beginning of the inner loop. Instead, it simply takes normal (non-meta) gradient update steps for each image/batch seen. The weights at the end of each sequence-and-batch are used directly on the next sequence.

We refer to this approach as *Alternating Sequential and Batch* learning (**ASB**), and the difference between ASB and meta-ASB can be seen visually in Appendix E [13]. This approach—like Lamarckian inheritance rather than Darwinian evolution [9]—benefits from not throwing away updates within each inner-loop adaptation sequence, but loses the higher-order updates thought to be effective

---

[4] Weights are sampled from the Kaiming Normal [15] and biases are set to zero.

Algorithm 1 Pre-Training: ASB and Meta-ASB, with or without zapping

Require: Dataset $\mathcal{D}$: C classes, N examples per class, (H, W, Ch) images
Require: Network f : (H, W, Ch) $\rightarrow$ $\mathcal{C}$ with parameters θ : $[\theta^{conv}, \theta^{fc}]$
Require: η_{in}, η_{out} inner and outer learning rates
Require: K number of sequential inner-loop examples
Require: R number of outer-loop "remember" examples
Require: S number of outer-loop steps

1: **for** iteration $= 1, 2, \ldots, S$ **do** {**outer loop; remembering**}
2:    $\mathcal{C} \sim \mathcal{D}$ {Sample one class}
3:    $X_{\text{inner}} \sim \mathcal{C}$ {K examples from class $\mathcal{C}$}
4:    $X_{\text{rand}} \sim \mathcal{D}$ {R examples from the whole dataset}
5:    $X_{\text{outer}} \leftarrow X_{\text{inner}} \cup X_{\text{rand}}$
6:    **if** zap **then**
7:      Reset connections in θ^{fc} corresponding to class $\mathcal{C}$ {**zapping**}
8:    **end if**
9:    $\theta_0 \leftarrow \theta$
10:   **for** i = 0, ..., K-1 **do** {**inner loop; adapting**}
11:     $\hat{y} \leftarrow f(X_{\text{inner}}^i; \theta_i)$
12:     $\theta_{i+1} \leftarrow \theta_i - \eta_{\text{in}}\nabla_{\theta_i}\mathcal{L}(\hat{y}, \mathcal{C})$ {single example SGD}
13:   **end for**
14:   **if** meta **then**
15:     $\theta \leftarrow \theta_0 - \eta_{\text{out}}\nabla_{\theta_0}\mathcal{L}(f(X_{\text{outer}}; \theta_K), Y)$ {meta batch SGD (expensive)}
16:   **else**
17:     $\theta \leftarrow \theta_K - \eta_{\text{out}}\nabla_{\theta_K}\mathcal{L}(f(X_{\text{outer}}; \theta_K), Y)$ {standard batch SGD (cheap)}
18:   **end if**
19: **end for**

---

for these continual learning tasks [18, 3]. This sequential approach allows us to employ the same zapping procedure as above: resetting the output node of the class which we are about to see a sequence of.

We also wished to study how zapping may influence the learning of generalizable features without being coupled with sequential learning. Thus, we also apply zapping to **i.i.d. pre-training**, which uses standard mini-batch learning with stochastic gradient descent. In this setting, a random sample of classes are zapped at a configurable cadence throughout training. For example, we might resample the entire final layer once per epoch, allowing the network to experience an event which is similar to fine-tuning.

2.1.2 Stage 2: Transfer

We evaluate our pre-trained models using two different types of few-shot transfer learning.

In **sequential transfer** (Alg. 2), we follow the evaluation method used in prior works [18, 3]. The model is trained on a long sequence of different unseen classes (*continual learning*). Examples are shown one at a time, and a gradient update is performed for each image. Only the weights in the final layer are updated (also referred to as "linear probing" [2]).

We also test **i.i.d. transfer** (Alg. 3), where the model is trained on batches of images randomly sampled from unseen classes (*standard fine-tuning*).

In both transfer scenarios, the new classes were not seen during the pre-training phase. There are only 15-30 images per class (*few-shot learning*). Between the end of pre-training and transfer, the final

linear layer of the model is replaced with a new, randomly initialized linear layer, so it can learn a new mapping of features to these new classes. Both sequential and i.i.d. transfer use the same set of classes and images—the only difference is how they are presented to the model.

Algorithm 2 Sequential Transfer Protocol (adapted from Beaulieu et al. [3], Algorithm 2)

Require: $\mathcal{C} \leftarrow$ sequential trajectory of N unseen classes
Require: $\theta \leftarrow$ pre-trained weights of the network
Require: $\beta \leftarrow$ learning rate hyperparameter
  1: $S_{train} = [\,]$
  2: **for** $n = 1, 2, \ldots, N$ **do**
  3:    $S_{traj} \sim \mathcal{C}_n$ {get training examples from next class}
  4:    $S_{train} = S_{train} \bigcup S_{traj}$ {add to seq. transfer train set}
  5:    **for** $i = 1, 2, \ldots, k$ **do**
  6:      $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}(\theta, S_{traj}^{(k)})$ {SGD update on a single image}
  7:    **end for**
  8:    record $\mathcal{L}(\theta, S_{train})$ {eval current θ on classes trained so far}
  9:    $S_{test} = (\bigcup_{1 \ldots n} \mathcal{C}_i) - S_{train}$ {held-out from seen classes}
10:    record $\mathcal{L}(\theta, S_{test})$ {eval current θ on held-out examples}
11: **end for**

Algorithm 3 I.I.D. Transfer Protocol

Require: $\mathcal{D}_{tr}, \mathcal{D}_{te} \leftarrow$ training and held-out examples from N unseen classes from domain $\mathcal{D}$
Require: $\theta \leftarrow$ pre-trained weights of the network
Require: $\beta \leftarrow$ learning rate hyperparameter
Require: $E \leftarrow$ number of training epochs
  1: **for** $i = 1, 2, \ldots, E$ **do**
  2:    **for** $i = 1, 2, \ldots, N$ **do** {N is the number of batches in $\mathcal{D}_{tr}$}
  3:      $B_i \sim \mathcal{D}_{tr}$ {uniformly sample from $\mathcal{D}_{tr}$ w/o replacement}
  4:      $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}(\theta, \mathcal{B}_i)$ {standard batch SGD update}
  5:      record $\mathcal{L}(\theta, \mathcal{D}_{tr})$ {eval current θ on all classes}
  6:      record $\mathcal{L}(\theta, \mathcal{D}_{te})$ {eval current θ on all held-out examples}
  7:    **end for**
  8: **end for**

---

3 Results

We evaluate two significantly different datasets, both in the few-shot regime.

Omniglot (Lake et al. [20]; handwritten characters, 1600 classes, 20 images per class) is a popular dataset for few-shot learning. Its large number of classes allows us to create very long, challenging trajectories for testing catastrophic forgetting under continual learning. However, due to its simple imagery we also include a dataset consisting of more complex natural images.

Mini-ImageNet (Vinyals et al. [33]; natural images, 100 classes, 600 images per class) contains hundreds of images per class, but in transfer we limit ourselves to 30 training images per class. This allows us to test the common scenario where we are allowed a large, diverse dataset in our pre-training, but our transfer-to dataset is of the limited few-shot variety.

For each pre-training configuration (Meta-ASB / ASB / i.i.d. and with/without zapping; Section 2.1.1), we report the average performance across 30 trials for transfer/continual learning results (3 random pre-train seeds and 10 random transfer seeds). We sweep over

three pre-training learning rates and seven transfer learning rates, and present the performance of the top performing learning rate for each configuration. We only evaluate the models at the end of training (i.e. no early stopping), but the number of epochs is tuned separately for each training method and dataset so as to avoid overfitting. See Appendix C [13] for more details on pre-training and hyperparameters.

Here we review the results on sequential transfer (Sec. 3.1) and i.i.d. transfer (Sec. 3.2). In the supplementary materials [13] we also show improvements due to zapping in the *unfrozen* sequential transfer setting (Appendix H), and how zapping alone can improve training but not as much as when it is combined with our ASB method (Appendix F).

3.1 Continual Learning

We evaluate the pre-trained variants described in 2.1.1 on the sequential transfer task as described in Section 2.1.2. To quickly recap, the models are fine-tuned using linear probing on a few examples from classes not seen during pre-training, the examples are shown one at a time, and an optimization step is taken after each one.

As we see in Figure 2, our Convnet Meta-ASB model performs similarly to the prior state of the art, ANML. In the prior work, ANML achieved 63.8% accuracy after sequential learning of 600 classes. Our reimplementation shows a slightly higher performance of 67% for both ANML and Meta-ASB.

However, our setup doesn't use heuristics on where/when in the model to apply optimization (like OML; Javed and White [18]), nor context-dependent gating (like ANML; Beaulieu et al. [3]), and uses fewer parameters than both prior works (see Appendix B [13]). This begs the question: what is it about these meta-learning algorithms that is contributing such drastic improvements?

As an answer, we see from the solid green and red lines in Figure 2, the models trained without zapping show significantly lower performance (41.5% and 42.2% vs 67%)—even though they *were* trained with meta-learning. Figure 3 shows that in all datasets, the meta-learned models with zapping significantly outperform their non-zapping counterparts, and outperform i.i.d. pre-training by an additional margin.

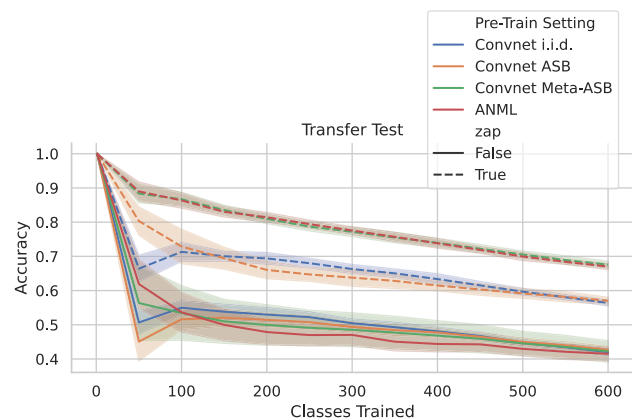On Omniglot, the best model without zapping achieves only

Figure 2. Sequential learning trajectories on Omniglot. Removing the neuromodulation layers from ANML has no impact on accuracy (Meta-ASB and ANML both achieve 67% final accuracy). Removing zapping, however, drastically affects accuracy, even when employing meta-learning. We do not compare directly to OML since ANML represents the state of the art.
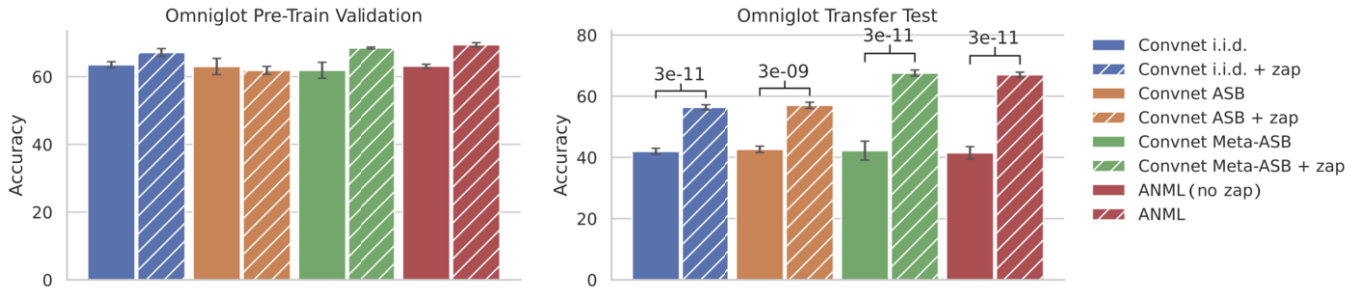
Figure 3. Average accuracy (and std dev error bars) for the sequential transfer learning problem, on Omniglot. **Pre-Train** is the final validation accuracy of the model on the *pre-training* dataset. All the layers are trained during pre-train. **Transfer** is the accuracy on held-out instances from the *transfer-to* dataset at the very end of sequential fine-tuning. Only the last layer is trained (linear probing) during transfer. Models trained **with zapping** produce significantly ($p < 10^{-8}$) better transfer accuracy than their counterparts without zapping in all cases (p-values of a two-sided Mann-Whitney U test are shown above each pair of bars). Note that the ANML model contains zapping by default and is therefore shaded in the legend.

42.6%. In fact, when applying zapping to i.i.d. pre-training, we can even achieve better performance (56.4%) than the models which are *meta-learned* but *without* zapping (~42%). This suggests that zapping may be an efficient and effective alternative (or complement) to meta-learning during pre-training.

On Mini-ImageNet (Figure 4), we again see a substantial difference between zapping models and their non-zapping counterparts (except for i.i.d.+**zap**). While on Omniglot Meta-ASB+**zap** outperformed ASB+**zap**, on Mini-ImageNet we observe that ASB+**zap** achieves the best accuracy, further demonstrating the effectiveness of the ASB method as a way to emulate the challenges of transfer during pre-training.

In Figure 3, we also include the *pre-train validation accuracy*: this is the validation accuracy of the pre-trained model *on the pre-training dataset*, before it was modified for transfer. We observe that ranking

Figure 4. Accuracy on classes seen so far during continual transfer learning on Mini-ImageNet. Models are trained on 30 examples from 20 new classes not seen during pre-training. All 30 images from a class are shown sequentially one at a time before switching to the next class. After each class, validation accuracy on the transfer set is measured using 100 examples per class, from all classes seen up to that point. Models pre-trained **with ASB** (with or without meta-gradients) significantly outperform i.i.d. pre-training. ASB+zapping further outperforms plain ASB ($p < 10^{-10}$).

models by validation performance *is not well correlated with ranking of transfer performance*. This lack of pre-train ↔ transfer correlation introduces a dilemma, whereby we may not have a reliable way of judging which models will be better for transfer until we actually try them.

Across all three datasets, we have observed that:

1. Zapping is a significant driver of performance improvements (see dashed vs. solid lines per treatment in Figures 2 and 4).
2. Zapping sometimes also improves normal (pre-)training performance, although this trend is less consistent (Figure 3; more details in Appendix A, Table 1 [13]).
3. Counter-intuitively, even the Alternating Sequential and Batch learning (ASB) sampling routine alone (without meta-gradients) appears to provide some benefits for transfer (see ASB vs i.i.d. in Figure 4). It may sometimes be unnecessary to use the much more expensive and complex higher-order gradients.

3.2 Transfer Learning

Although the zapping and meta-learning methods described in Algorithm 1 were originally designed to learn robust representations for continual learning, we show that they are beneficial for transfer learning in general. Here we feature the results of standard i.i.d. transfer, as described in Section 2.1.2. We train each model for five epochs of unfrozen fine-tuning on mini-batches of data using the Adam optimizer.

Figure 5 shows results on Omniglot and Mini-ImageNet. As in the continual learning tests, here we also see substantial gains for the models employing zapping over those that do not. When zapping is not employed, models pre-trained with meta-gradients are comparable to those trained simply with standard i.i.d. pre-training. See Appendix A [13] for a detailed comparison of final values. Despite both the zapping and ASB pre-training methods stemming from attempts to reduce catastrophic forgetting in continual learning settings, zapping consistently provides advantages over non-zapped models for all pre-training configurations on standard i.i.d. transfer learning. We hypothesize that these two settings—continual and transfer learning—share key characteristics that make this possible. Both cases may benefit from an algorithm which produces more adaptable, robust features that can quickly learn new information while preserving prior patterns that may help in future tasks.
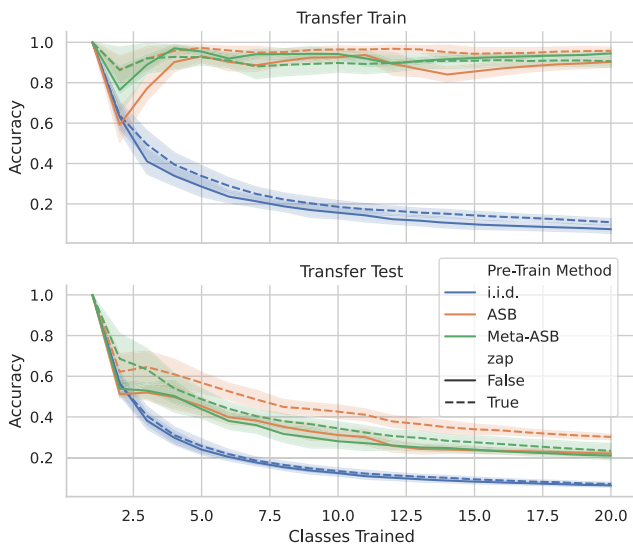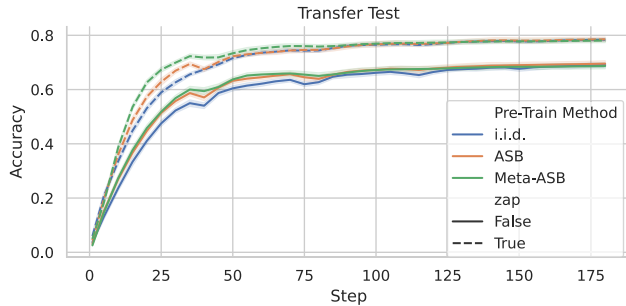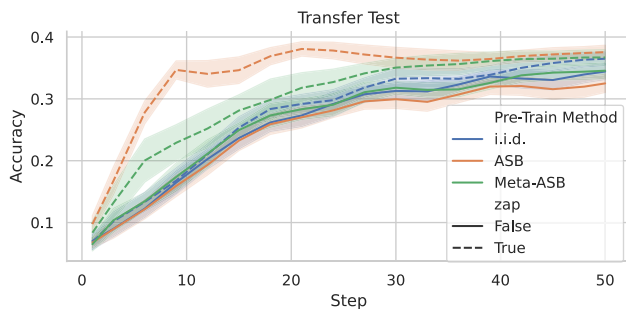
(a) Omniglot (15 training images / 5 testing images per class).

(b) Mini-ImageNet (30 training / 100 testing images per class).

Figure 5. Validation accuracy over training time on all classes in the transfer set during fine-tuning with standard i.i.d. batches. For all datasets, models pre-trained **with zapping** achieve significantly higher transfer accuracy at end of fine-tuning. While ASB methods (green, orange) do not dramatically improve final performance, they achieve more rapid fine-tuning relative to i.i.d.+zap pre-training (blue).

3.3 Toward Larger Architectures

We conclude our investigation showing how zapping and ASB influence the training dynamics of a larger model, specifically the widely-used VGG-16 architecture [31]. For this larger model, the simplicity of Omniglot images presents a limitation; therefore we use a variant called Omni-image [12]. The Omni-image dataset retains the task structure of Omniglot (i.e. 20 images per class, high within-class visual consistency) but uses natural images (instead of handwritten characters) taken from the 1000 classes available in ImageNet-1k [6]. Omni-image was designed for few-shot and continual learning, and we show some examples in Appendix G [13].

Figure 6 shows that models trained using zapping not only learn faster but also achieve a better final performance. These results suggest that zapping and ASB could potentially be applied to other architectures but the advantages of ASB may depend on the specific task and dataset structure. For instance, consider the differing results observed with Omniglot and Omni-image (Figures 3 & 6). ASB+**zap** outperformed i.i.d.+**zap** in the former, but was slightly worse in the latter.

4 Discussion

Across three substantially different datasets, zapping consistently results in better representations for transferring to few-shot datasets, leading to improvements in both a continual learning and standard transfer setting. In many cases, we are still able to achieve the best performance by just applying zapping and alternating optimizations
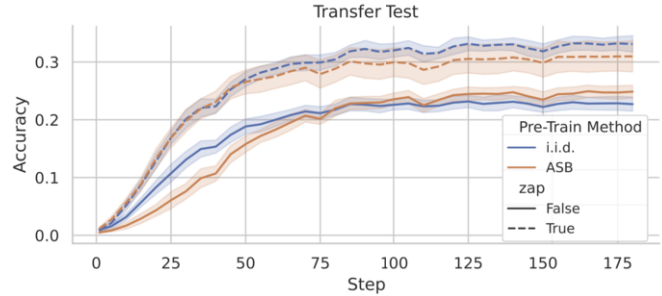

Figure 6. VGG-16 transfer on the Omni-image dataset. Zapping improves both the speed of adaptation and the final accuracy, but ASB does not contribute any further improvement.

of sequential learning and batch learning (ASB), without applying any meta-gradients.

We even see some benefit from applying zapping directly to i.i.d. training, without any sequential learning component. This raises the question of whether we can match the performance of meta-learning using only zapping and standard i.i.d. training. However, this setting introduces new choices of when and where to reset neurons, since we are learning in batches and not just one class at a time. We include ablations in Appendix F [13] that examine this question; in most cases, more zapping leads to better performance, but it is still outmatched by ASB. This serves as a promising starting point, and better variants of this scheme could likely be discovered.

It is reasonable to suppose that the constant injection of noise by resetting weights during training helps to discover weights which are not as affected by this disruption, thus building some resilience to the shock of re-initializing layers. If it can be shown that the noise injections reduce the *co-adaptation of layers* [34], thereby increasing their resilience, it begs the question of how it relates to other co-adaptation reducing mechanisms such as dropout, which is also shown to improve continual [24] and transfer [30] learning.

The approaches explored here include pre-training by alternating between sequential learning on a single class and batches sampled from all pre-training classes (ASB), and resetting classifier weights prior to training on a new class (zapping). The information accumulated by repeating these simple methods across many tasks during the pre-training process mimics the condition experienced during transfer learning at test time. Episodes of sequential training are likely to cause both catastrophic forgetting and overfitting. Models that manage to overcome those hurdles during training seem to develop resilient features, beyond what the loss function selects for. We thus argue that the results above demonstrate a simple yet effective alternative to meta-learning—one without expensive meta-gradients to backpropagate through tasks.

5 Related work

As we have shown, the zapping operation of resetting last-layer weights provides clear performance improvements, but what is it about continually injecting random weights that enables this improved learning?

The work of Frankle et al. [11] investigates the dynamics of learning during the early stages of network training. They show that large gradients lead to substantial weight changes in the beginning of training, after which gradients and weight signs stabilize. Their work suggests that these initial drastic changes are linked to weight-level noise.

Dohare et al. [7] also investigate the relationship between noise and learning, showing that stochastic gradient descent is not enough to learn continually, and that this problem can be alleviated by repeated injections of noise. Rather than resetting classification neurons of the last layer, they choose weights to reset based on a pruning heuristic.

The reinitialization of weights in a neural network during training is an interesting emerging topic, with many other works investigating this phenomenon in a number of different settings. Like us, Zhao et al. [37] periodically reinitialize the last layer of a neural network during training. Their focus is on ensemble learning for a single dataset, rather than transfer learning. Li et al. [21] also periodically reinitialize the last layer, but they do it *during transfer*, rather than pre-training, which may not be possible depending on the application. Both Alabdulmohsin et al. [1] and Zhou et al. [38] investigated the idea of reinitialization of upper layers of the network, building upon the work of Taha et al. [32]. They show performance improvements in the few-shot regime. However, they focus on learning of a single dataset rather than transfer learning. The same is true of Zaidi et al. [35], who evaluate an extensive number of models to find under which circumstances reinitialization helps. Nikishin et al. [25] apply a similar mechanism to deep reinforcement learning. They find that periodically resetting the final layers of the Q-value network is beneficial across a broad array of settings. Ramkumar et al. [26] study the application of resetting to a version of online learning where data arrives in mega-batches. They employ resetting as a compromise between fine-tuning a pre-trained network and training a new network from scratch. More generally, Lyle et al. [22] attempt to discover the reasons for plasticity loss in neural networks, and show the resetting of final layers to be one of a few effective methods in maintaining plasticity.

One unique aspect of our work is the zapping + ASB routine, where we forget one class at a time and focus on relearning that class. Another major difference from prior investigations is that we examine how resetting weights better prepares a pre-trained model for transfer learning. In this regard, the concurrent work of Chen et al. [5] examines the same topic, albeit in the domain of natural language processing. Their method repeatedly forgets the early embedding layers of a language model, rather than the later classification layers of an image model. As with our work, they also find that this repeated forgetting results in a meta-learning-like effect which better prepares the model for downstream transfer. This gives us a new lens through which to view weight resetting.

6   Conclusion & Future Work

We have revealed the importance of "zapping" and relearning for pre-training, and its connection to meta-learning. We have shown that zapping can lead to significant gains in transfer learning performance across multiple settings and datasets. The concept of forgetting and relearning has been investigated in other recent works, and our observations add to the growing evidence of the usefulness of this concept.

Aside from the benefits of zapping and relearning, our results highlight the disruptive effect of the re-initialization of last layers in general. Resetting of the final layer is routine in the process of fine-tuning a pre-trained model [34], but the impact of this "transfer shock" is still not fully clear. For instance, it was only recently observed that fine-tuning in this way underperforms on out-of-distribution examples, and Kumar et al. [19] suggest to freeze the lower layers of a network to allow the final layer to stabilize. A deeper understanding of these mechanisms could significantly bene-

fit many areas of neural network research.

Finally, this work explores a simpler approach to meta-learning than meta-gradient approaches. It does so by repeatedly creating transfer shocks during pre-training, encouraging a network to learn to adapt to them. Future work should explore other methods by which we can approximate transfer learning during pre-training, how to influence the features learned to maximize transfer performance with the most computational efficiency, and how the benefits of zapping scale to larger models.

Acknowledgements

References

[1] I. Alabdulmohsin, H. Maennel, and D. Keysers. The impact of reinitialization on generalization in convolutional neural networks. *arXiv preprint arXiv:2109.00267*, 2021.

[2] G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[3] S. Beaulieu, L. Frati, T. Miconi, J. Lehman, K. O. Stanley, J. Clune, and N. Cheney. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.

[4] Y. Bengio, S. Bengio, and J. Cloutier. *Learning a synaptic learning rule*. Citeseer, 1990.

[5] Y. Chen, K. Marchisio, R. Raileanu, D. I. Adelani, P. Stenetorp, S. Riedel, and M. Artetxe. Improving language plasticity via pretraining with active forgetting. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] S. Dohare, A. R. Mahmood, and R. S. Sutton. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021.

[8] M.-A. Félix and A. Wagner. Robustness and evolution: concepts, insights and challenges from a developmental model system. *Heredity*, 100(2):132–140, 2008.

[9] C. Fernando, J. Sygnowski, S. Osindero, J. Wang, T. Schaul, D. Teplyashin, P. Sprechmann, A. Pritzel, and A. Rusu. Meta-learning by the baldwin effect. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1313–1320, 2018.

[10] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[11] J. Frankle, D. J. Schwab, and A. S. Morcos. The early phase of neural network training. *arXiv preprint arXiv:2002.10365*, 2020.

[12] L. Frati, N. Traft, and N. Cheney. Omnimage: Evolving 1k image cliques for few-shot learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, New York, NY, USA, 2023. Association for Computing Machinery.

[13] L. Frati, N. Traft, J. Clune, and N. Cheney. Reset it and forget it: Relearning last-layer weights improves continual and transfer learning. *arXiv preprint arXiv:2310.07996*, 2023. Full version of this paper.

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

[15] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In

Proceedings of the IEEE international conference on computer vision, pages 1026–1034, 2015.

[16] S. Hochreiter, A. S. Younger, and P. R. Conwell. Learning to learn using gradient descent. In *Artificial Neural Networks—ICANN 2001: International Conference Vienna, Austria, August 21–25, 2001 Proceedings 11*, pages 87–94. Springer, 2001.

[17] E. Hoffer, I. Hubara, and D. Soudry. Fix your classifier: The marginal value of training the last weight layer. *arXiv preprint arXiv:1801.04540*, 2018.

[18] K. Javed and M. White. Meta-learning representations for continual learning. *arXiv preprint arXiv:1905.12588*, 2019.

[19] A. Kumar, A. Raghunathan, R. M. Jones, T. Ma, and P. Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=UYneFzXSJWh.

[20] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350 (6266):1332–1338, 2015.

[21] X. Li, H. Xiong, H. An, C.-Z. Xu, and D. Dou. Rifle: Backpropagation in depth for deep transfer learning through re-initializing the fully-connected layer. In *International Conference on Machine Learning*, pages 6010–6019. PMLR, 2020.

[22] C. Lyle, Z. Zheng, E. Nikishin, B. A. Pires, R. Pascanu, and W. Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, 2023.

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

[24] S. I. Mirzadeh, M. Farajtabar, and H. Ghasemzadeh. Dropout as an implicit gating mechanism for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 232–233, 2020.

[25] E. Nikishin, M. Schwarzer, P. D'Oro, P.-L. Bacon, and A. Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, pages 16828–16847. PMLR, 2022.

[26] V. R. T. Ramkumar, E. Arani, and B. Zonooz. Learn, unlearn and relearn: An online learning paradigm for deep neural networks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=WN1O2MJDST.

[27] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2017.

[28] T. Rohlf and C. R. Winkler. Emergent network structure, evolvable robustness, and nonlinear effects of point mutations in an artificial genome model. *Advances in Complex Systems*, 12(03):293–310, 2009.

[29] J. Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[30] T. Semwal, P. Yenigalla, G. Mathur, and S. B. Nair. A practitioners' guide to transfer learning for text classification using convolutional neural networks. In *Proceedings of the 2018 SIAM international conference on data mining*, pages 513–521. SIAM, 2018.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] A. Taha, A. Shrivastava, and L. S. Davis. Knowledge evolution in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12843–12852, 2021.

[33] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

[34] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[35] S. Zaidi, T. Berariu, H. Kim, J. Bornschein, C. Clopath, Y. W. Teh, and R. Pascanu. When does re-initialization work? *arXiv preprint arXiv:2206.10011*, 2022.

[36] C. Zhang, S. Bengio, and Y. Singer. Are all layers created equal? *The Journal of Machine Learning Research*, 23(1):2930–2957, 2022.

[37] K. Zhao, T. Matsukawa, and E. Suzuki. Retraining: A simple way to improve the ensemble accuracy of deep neural networks for image classification. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 860–867. IEEE, 2018.

[38] H. Zhou, A. Vani, H. Larochelle, and A. Courville. Fortuitous forgetting in connectionist networks. In *International Conference on Learning Representations*, 2022.