Slow and Steady Wins the Race
Maintaining Plasticity with Hare and Tortoise Networks

Hojoon Lee [1] **Hyeonseo Cho** [2] **Hyunseung Kim** [1] **Donghu Kim** [1] **Dukgi Min** [2] **Jaegul Choo** [1] **Clare Lyle** [3]

Abstract

This study delves into the loss of generalization ability in neural networks, revisiting warm-starting experiments from Ash & Adams (2020). Our empirical analysis reveals that common methods designed to enhance plasticity by maintaining trainability provide limited benefits to generalization. While reinitializing the network can be effective, it also risks losing valuable prior knowledge. To this end, we introduce the Hare & Tortoise, inspired by the brain's complementary learning system. Hare & Tortoise consists of two components: the Hare network, which rapidly adapts to new information like the hippocampus, and the Tortoise network, which gradually integrates knowledge akin to the neocortex. By periodically reinitializing the Hare network to the Tortoise's weights, our method preserves plasticity while retaining general knowledge. Hare & Tortoise can effectively maintain the network's ability to generalize, which improves advanced reinforcement learning algorithms on the Atari-100k benchmark. The code is available at https://github.com/dojeon-ai/hare-tortoise.

1. Introduction

In neural networks, *plasticity* refers to the ability to learn and adapt to new data. Maintaining plasticity is crucial in continual learning and reinforcement learning, where networks must consistently adapt to incoming data. With the emergence of large-scale models like GPT-4 (Achiam et al., 2023) and Gemini (Google et al., 2023), understanding plasticity is becoming crucial, considering a substantial resource that is required to retrain these models from scratch.

Recent studies have shown that neural networks tend to lose plasticity as training progresses (Lyle et al., 2023; Dohare et al., 2023). In experiments where the dataset labels were periodically randomized, some networks gradually lost their ability to minimize their training loss. This phenomenon has been linked to several factors such as the increasing distance from initial weights, emergence of gradient starvation, and accumulating dormant neurons (Lyle et al., 2023; Dohare et al., 2023; Lewandowski et al., 2023). To mitigate this problem, various methods have been developed (Lyle et al., 2022; Kumar et al., 2023b; Abbas et al., 2023; Sokar et al., 2023), proving their efficacy in maintaining trainability throughout repetitive learning tasks.

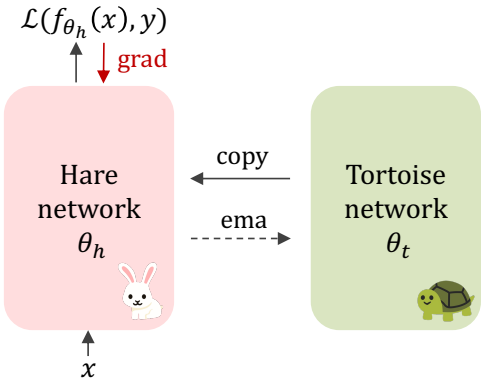

Figure 1. **Hare & Tortoise architecture**. The Hare Network rapidly updates its weights for new data, while the Tortoise Network slowly integrates knowledge through an exponential moving average (ema) of the Hare's weights. Periodic reinitialization of the Hare Network to the Tortoise Network's weights ensures a balance between fast, fleeting adaptation and slow, steady generalization.

Ultimately, the goal of maintaining plasticity is not just to memorize the new training data but to enable the network to generalize to unseen data. However, it is not well understood whether, and if so how, improved trainability translates to better generalization. The first contribution of this paper is to conduct a rigorous empirical study into the relationship between these dual faces of plasticity. To do so, we expand the warm-starting experiments from Ash & Adams (2020) by first training some networks on noisy dataset subsets and then retraining them on complete, noise-free datasets. Here, we observed a consistent decline in generalization when models were initially trained on smaller and noisier subsets.

[1]KAIST [2]Konkuk University [3]Deepmind. Correspondence to: Hojoon Lee <joonleesky@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

On top of this setup, we investigate whether trainability-enhancing methods can mitigate this decline.

Surprisingly, while these methods effectively maintain the network's ability to reduce the training loss in later epochs, broadly speaking they do not improve generalization. Instead, we observe that standard training techniques such as L2 regularization (Krogh & Hertz, 1991) and data augmentation (Takahashi et al., 2019) tend to have a much greater effect on generalization. However, they are not sufficient to completely address the loss of generalization ability. Weight Re-initialization methods such as Shrink & Perturb (Ash & Adams, 2020) emerge as the most effective solution, enhancing plasticity by periodically resetting and scaling weights.

While effective, resetting the network (even if it's partial) is costly. In large models, resetting resembles a form of less intensive retraining and demands considerable effort to restore prior performance. When data access is limited due to privacy (Abadi et al., 2016; McMahan et al., 2017) or storage constraints (Shin et al., 2017; Smith et al., 2023), it risks losing previously learned valuable information. In the end, we seek a system that allows the network to quickly adapt to new data and overwrite spurious correlations without erasing all of the prior knowledge. In our efforts towards this goal, we draw inspiration from the human brain.

Humans maintain plasticity through a dynamic interaction within the brain's complementary learning systems (McClelland et al., 1995). The human brain possesses two distinct memory modules: the fast-learning hippocampus and the slow-learning neocortex (Kumaran et al., 2016). The hippocampus rapidly encodes and transfers new information to the neocortex, which is responsible for storing and retaining long-term knowledge and skills. While the hippocampus periodically forgets its knowledge to preserve the brain's plasticity, the neocortex serves as a long-term module to preserve general knowledge (Frankland et al., 2013).

Inspired by the complementary learning theory, we introduce a novel network architecture, Hare & Tortoise, reflecting the brain's fast and slow learning mechanisms. As illustrated in Figure 1, the Hare network, like the hippocampus, rapidly updates information and explores optimization landscapes. On the other hand, the Tortoise network, akin to the neocortex, slowly integrates knowledge from the Hare network by an exponential moving average. To maintain plasticity, the Hare network is periodically reinitialized to the Tortoise network's weights, which naturally retains the generalizable knowledge of the slow-learning Tortoise.

Hare & Tortoise consistently yields impressive results across various experimental setups. In warm-starting and continual learning experiments, Hare & Tortoise effectively maintain generalization abilities without any sudden performance drop after resets, showing competitive performance compared to reinitialization methods. Furthermore, in reinforcement learning, Hare & Tortoise can be seamlessly integrated into modern algorithms that employ momentum target networks. By just periodically reinitializing the online network to the target network, Hare & Tortoise enhances the sample efficiency of both DrQ (Kostrikov et al., 2020) and BBF (Schwarzer et al., 2023) in the Atari-100k benchmark.

2. Related Work

2.1. Loss of Plasticity

The design of neural network architectures and initialization schemes that both train and generalize well has produced a rich scientific literature (Glorot & Bengio, 2010; He et al., 2015; 2016), allowing supervised training to reliably scale to immense model and dataset sizes (Kaplan et al., 2020). In recent years, however, many works have identified the insufficiency of standard optimization algorithms in non-stationary domains, such as reinforcement learning (Lyle et al., 2023), continual learning (Kumar et al., 2023a; Dohare et al., 2023), and lifelong learning (Sodhani et al., 2020). In particular, these works identify a phenomenon known as *loss of plasticity*, whereby neural networks progressively lose their ability to learn and adapt to new data.

Loss of plasticity, as noted by Berariu et al. (2021), can be decomposed into two distinct factors: a reduced ability of networks to minimize the training loss on new data, *trainability*, and a reduced ability to generalize well to unseen data, *generalizability*. In the former case, Lyle et al. (2022) observed that neural networks can exhibit reduced performance over time on sequential memorization tasks, where no generalization is required. In the latter case, Ash & Adams (2020) observed reduced generalization performance after warm-starting on a subset of the training data, despite achieving zero training error.

Understanding the precise causes of plasticity loss for both training and generalization remains unclear (Lyle et al., 2023; 2024), but efforts to mitigate plasticity loss have been diverse. To enhance trainability, methods include maintaining active units (Abbas et al., 2023; Elsayed & Mahmood, 2024), preventing gradient starvation (Gogianu et al., 2021; Lyle et al., 2022; Dohare et al., 2023), using small batch size (Obando et al., 2024), and limiting deviation from initial weights (Lewandowski et al., 2023; Kumar et al., 2023b). For generalization, periodicaly reinitializing the network has been effective (Ash & Adams, 2020; Zhou et al., 2022; Zaidi et al., 2023; Noukhovitch et al., 2023; Frati et al., 2023), particularly in data-efficient RL (Nikishin et al., 2022; Lee et al., 2023; Xu et al., 2023; Nauman et al., 2024).

Despite these efforts, the relationship between enhancing trainability and achieving better generalization under non-stationary learning conditions is not fully understood. Our

research aims to bridge this gap by investigating whether modern neural networks suffer from a loss of trainability and whether enhancing trainability can counteract this decline.

2.2. Complementary Learning System

The Complementary Learning System (CLS) in the human brain is crucial for us to continuously acquire, consolidate, and transfer knowledge. The CLS operates with dynamic interaction between the hippocampus and the neocortex. The hippocampus is responsible for rapid learning and short-term adaptation of episodic information, while the neocortex is responsible for the slower, structured integration of long-term knowledge (McClelland et al., 1995; Kumaran et al., 2016). Another key process in the brain is forgetting, which discards old information to make room for new knowledge, a critical component in maintaining brain plasticity (Frankland et al., 2013; Gravitz, 2019; Ryan & Frankland, 2022).

In neural networks, the CLS principle has been applied to fields that require continual learning. In continual learning, dual-network architectures have been designed to replicate the mechanisms of the hippocampus and neocortex. The hippocampal network, fast learner, undergoes standard supervised learning, while the neocortical network, slow learner, is trained by self-supervised objectives (Pham et al., 2021), ensembles (Arani et al., 2022; Pham et al., 2022), or knowledge distillation (Gomez-Villa et al., 2024). Similarly, in RL, distinct architectural designs (Duan et al., 2016; Pritzel et al., 2017) or decomposed value functions (Anand & Precup, 2023) have been employed to echo this concept.

This work distinguishes itself from existing methods by explicitly integrating a forgetting mechanism to enhance the network's plasticity. Unlike previous studies based on the CLS principle, we emphasize the crucial role of forgetting to acquire new knowledge. In our approach, the Hare network is periodically reinitialized to the Tortoise network's weights, forgetting obsolete information while preserving generalizable knowledge within the Tortoise network.

3. Investigating the Effect of Warm-Starting on Neural Network Generalization

Many methods in the literature that aim to mitigate the loss of plasticity focus on ensuring the network remains *trainable*, meaning the network can continually minimize the training loss. It is not guaranteed, however, that trainable networks necessarily avoid overfitting (Zhang et al., 2021; Xiao et al., 2020). This section will use the warm-starting regime of Ash & Adams (2020) to disentangle the dual aspects of plasticity into *trainability* and *generalizability*. Our analysis will aim both to identify features of a task that increase the risk of overfitting and to assess whether existing approaches can maintain a network's *generalizability*.

3.1. Experimental Setup

Adopting the experimental setup from Ash & Adams (2020); Zaidi et al. (2023), we begin with training networks on noisy data subsets, followed by retraining it on complete, noise-free datasets. This setup is particularly relevant to deep reinforcement learning using a temporal difference loss (Tesauro et al., 1995; Sutton & Barto, 2018). Here, the network begins training with a small dataset containing noisy targets, followed by a gradual increase in data size and target accuracy. Our experiments are designed to yield insight into neural networks' generalization abilities under analogous non-stationary learning dynamics.

Dataset and Architecture. For our study, we select datasets and architectures that are widely used. For dataset, we used MNIST (LeCun et al., 1998), CIFAR-10, CIFAR-100 (Krizhevsky, 2009) and Tiny ImageNet (Le & Yang, 2015). For each dataset, we paired a network architecture of 3-layer MLP, ResNet-18 (He et al., 2016), ViT-Tiny (Dosovitskiy et al., 2020), and VGG-16 (Simonyan & Zisserman, 2014).

Warm-starting. Our networks were initially warm-started using a subset of the dataset, with varying ratios from $\{0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$, where 1.0 represents the entire dataset. Following Zhang et al. (2021), we inject label noise by substituting true labels with random ones at varying ratios of $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. Subsequently, the network was trained on the complete dataset without label noise.

Training Details. In both warm-starting and subsequent training phases, we employed the AdamW optimizer (Loshchilov & Hutter, 2017) with L2 weight decay and a batch size of 256. For each dataset, the learning rate was tuned via grid search, ranging from $\{0.01, 0.001, 0.0001, 0.00001\}$. Since the optimizer statistics within the Adam optimizer (β_1, β_2) can bias the optimization trajectory, we reinitialized the optimizer before proceeding to the subsequent training phase (Gogianu et al., 2021; Lyle et al., 2023; Asadi et al., 2023).

Training epochs were fixed to 100 for both phases, with a proportional increase in epochs during the warm-starting phase with subsets, to equalize the number of total gradient updates (e.g., 1000 epochs for 10% of the dataset). 5 random seeds were used for experiments, except for Tiny ImageNet where 3 seeds were employed.

3.2. Effects of Warm-Starting on Generalizability

First, we study the effects of warm-starting on generalization, w.r.t reduced data diversity and increased label noise.

In Figure 2.(a), we examine the relationship between initial dataset size and test accuracy, without label noise. Our analysis reveals that reductions in dataset size impair generalization, as evidenced by lower test accuracy scores. Figure

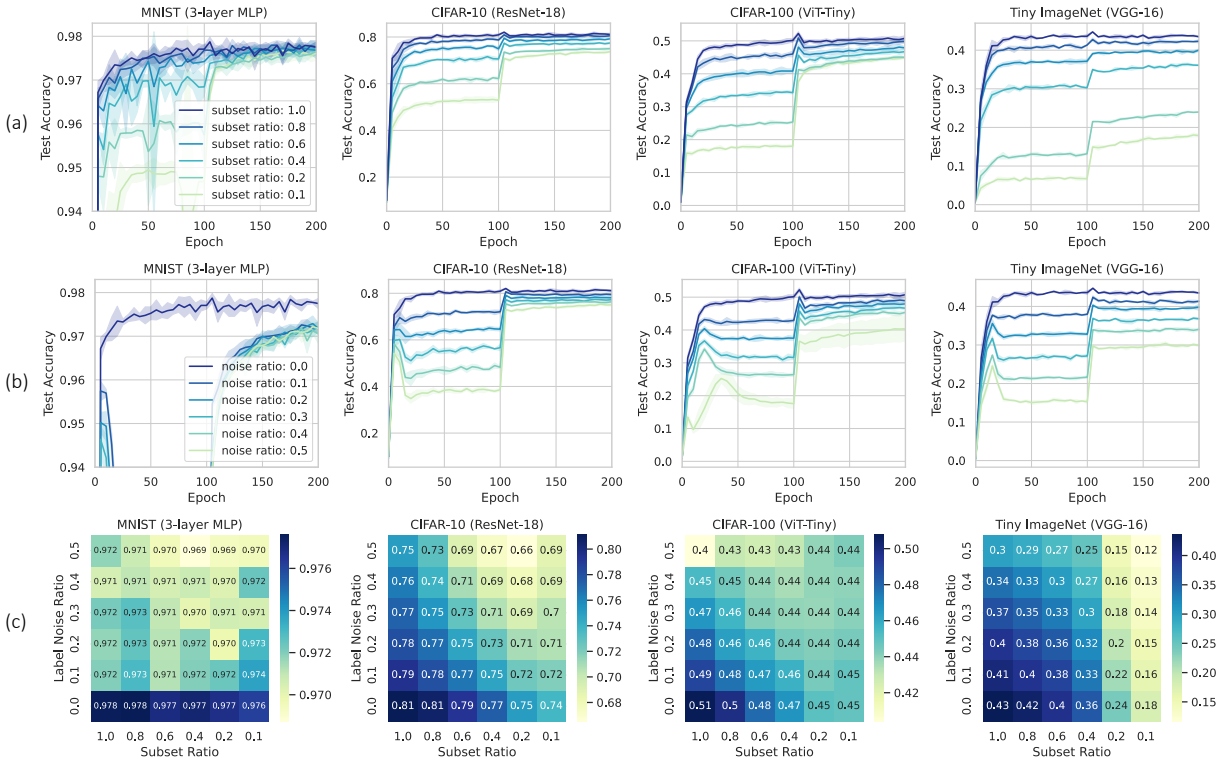

Figure 2. **Impact of Warm-Starting on Generalization. (a)** Shows a negative correlation between test accuracy and subset ratio without label noise. **(b)** Presents a negative correlation between test accuracy and label noise ratio, with a full dataset. **(c)** Presents the combined impact, indicating both reduced data size and increased label noise detrimentally affect generalization.

2.(b) explores the impact of injecting varying levels of label noise with a complete dataset. We observe a significant degradation in generalization when these models are subsequently trained on noise-free datasets. Figure 2.(c) explores the combined effects of smaller subsets and increased label noise, showing that these factors exacerbate performance deterioration. Despite this decline, warm-started models achieved near-perfect training accuracy, comparable to models that were freshly initialized (Appendix B).

Our analysis also reveals that both dataset type and network architecture influence the network's plasticity. For instance, warm-starting the VGG architecture on Tiny ImageNet resulted in a substantial loss compared to warm-starting an MLP architecture on the MNIST dataset. While both factors are important, the dataset had a more pronounced effect. This is further corroborated in Appendix B.2, where we conduct additional experiments with CIFAR-10 using VGG16 and Tiny ImageNet using ResNet-18.

3.3. Effects of Optimizer on Generalizability

Modern neural networks commonly utilize adaptive optimizers like Adam (Kingma & Ba, 2014) or LARS (You et al., 2019), which adjust convergence speed through parameters such as β_1, β_2, and ϵ. To assess the potential of these parameters in mitigating generalization loss during warm-starting, we conducted a grid search over β_1, β_2, and ϵ in Adam optimizer. Results are depicted in Figure 3.

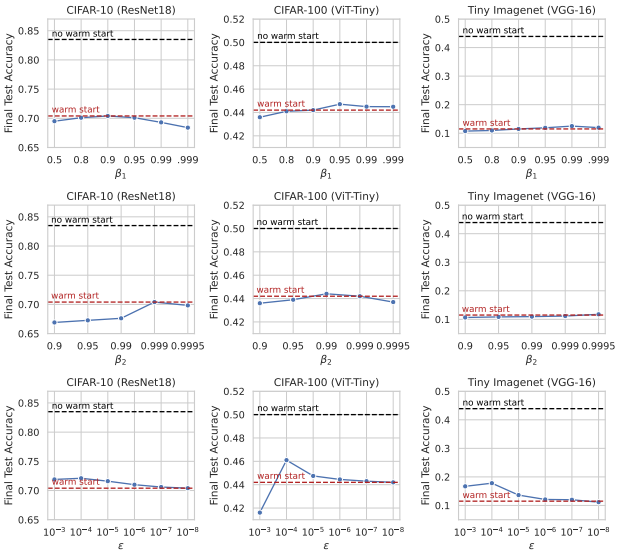

Figure 3. **Effect of Optimizer Parameters.** We observed marginal improvements with varying β_1 and β_2. Larger ϵ alleviate generalization loss but are insufficient to entirely address it.

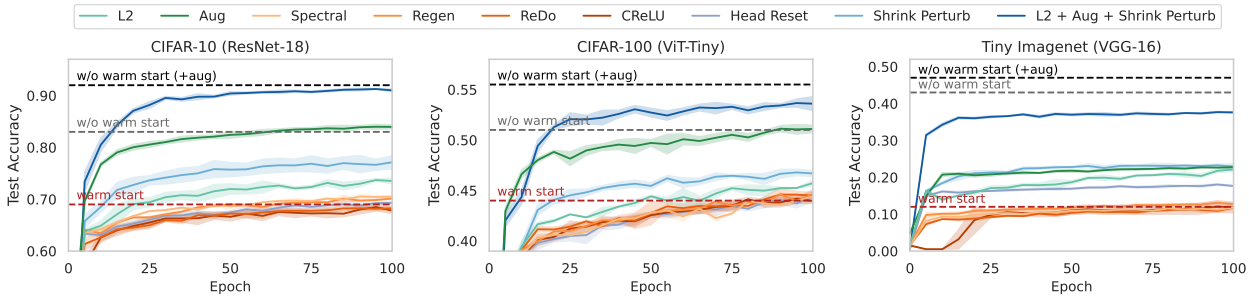

Figure 4. **Comparison of Existing Methods.** This figure presents a comparative analysis of test accuracies for different methods applied to networks warm-started with a 10% subset ratio and 50% label noise. Dashed lines indicate the performance of a warm-started network (lower bound) and a fresh network without warm-starting (upper bound). Generalizability methods (L2, Aug) are marked in green, Trainability methods (Spectral, Regen, ReDo, CReLU) in red, and Re-initialization methods (Head Reset, Shrink & Perturb) in blue.

Our findings indicate that tuning β_1 and β_2 showed marginal improvements, with default values generally performing best (β_1, β_2 = (0.9, 0.999)). Decreasing momentum in Adam may slow convergence but does not effectively counteract generalization loss.

Conversely, larger epsilon values seemed to alleviate generalization loss, aligning with prior studies (Gogianu et al., 2021; Lyle et al., 2023). This indicates that larger epsilon induces smoother gradients, mitigating the generalization aspect of plasticity loss. Notably, large epsilon values are commonly used in RL algorithms as implementation details (Hessel et al., 2018), possibly contributing to this loss of generalizability. However, despite their benefits, they were insufficient to entirely address generalization loss.

3.4. Enhancing Trainability is Insufficient for Maintaining Generalizability

Having observed a significant degradation in networks' generalization ability from warm-starting, we now investigate whether existing methods can mitigate this decline. We categorized existing methods into three groups: Generalizability, Trainability, and Re-initialization. Generalizability includes approaches that aim to prevent overfitting and enhance generalization. Trainability involves methods designed to consistently minimize the training loss, which has been established to address the plasticity loss on the training side. Re-initialization encompasses methods that reinitialize parts of the network to their initial weight distribution.

We conduct these evaluations under the most challenging conditions observed in our earlier analysis: a subset ratio of 0.1 and a label noise ratio of 0.5. We include a vanilla warm-started network as a lower bound and a fresh network without warm-starting as an upper bound.

Generalizability. We investigated widely used techniques, such as L2 regularization (Krogh & Hertz, 1991) and Data Augmentation (Takahashi et al., 2019). As illustrated in Figure 4, L2 regularization demonstrated improved generalization across datasets, leading to enhanced test accuracy. Data Augmentation proved its effectiveness, yielding results similar to those of models without warm-starting. However, a performance gap persists between warm-started models with Data Augmentation and their non-warm-started counterparts, indicating persistent generalization loss.

Trainability. We explored various trainability methods, including Spectral decoupling (Pezeshki et al., 2021), Regenerative regularization (Kumar et al., 2023b), ReDo (Sokar et al., 2023), and CReLU (Abbas et al., 2023). While these methods accelerated the convergence of training loss, none of them enhanced generalization on unseen data.

We also examined the relationship between generalization performance and potential indicators of plasticity loss, such as weight magnitude, active unit fraction, and feature rank. However, none of these metrics show a consistent correlation with test accuracy, thus not elucidating the loss of generalization ability. For details, refer to Appendix C.

Re-initialization. We explore two methods: Head Reset (Nikishin et al., 2022) and Shrink & Perturb (Ash & Adams, 2020). For Head Reset, the MLP network after the encoder is reinitialized to its initial weight. While Head Reset has shown its effectiveness in RL literature (Nikishin et al., 2022), it only improves generalization over the other baselines in the VGG architecture, where the head network contains a relatively larger fraction of the total parameters (18%). This observation aligns with RL setups, where nearly 90% of the parameters are included in the Head network (D'Oro et al., 2022). Therefore, we conjecture that Head Reset may not scale effectively with architectures featuring larger and deeper encoders.

Shrink & Perturb involve shrinking the network's weights towards their initial values and perturbing with Gaussian noise. For simplicity, we focus solely on the shrinking process with a shrink ratio set to 0.8, where 80% of the weights come

from the initial values and 20% from the current weights. For all experiments, Shrink & Perturb significantly reduce plasticity loss and enhance generalization across all datasets. Further incorporation of standard generalization techniques (L2 + Aug + Shrink & Perturb) narrows the gap between this approach and training a fresh network without warm-starting. For details on each method, see Appendix A.4.

In summary, our findings reveal that while loss of trainability was often observed in small neural networks with prolonged training (Lyle et al., 2022; Abbas et al., 2023; Kumar et al., 2023b), modern neural networks generally do not suffer from this issue. Subsequently, enhancing trainability does not necessarily improve generalization in these cases. While certain generalization strategies (i.e., L2 regularization, data augmentation) showed moderate success, integrating them with weight reinitialization methods, particularly Shrink & Perturb, proved to be most effective.

However, naive reinitialization has certain limitations. While reinitialization recovers network plasticity, it concurrently loses valuable information. This drawback is particularly prominent when data access is limited due to privacy constraints or during the training process of larger models, where it exacerbates computational costs.

Takeaways:

- Plasticity can be decoupled into *trainability* (ability to train) and *generalizability* (ability to generalize).
- Modern networks tend to retain *trainability* but lose *generalizability* after warm starting.
- Warm-starting on smaller datasets with larger label noise exacerbates the loss of *generalizability*.
- Varying optimizer parameters ($\beta_1, \beta_2, \epsilon$ in Adam) are insufficient to mitigate loss of *generalizability*.
- Enhancing *trainability* does not necessarily improve *generalizability* when *trainability* is retained.
- Standard regularization techniques, such as data augmentation and weight regularization, are generally effective in maintaining *generalizability*.
- Reinitializing the network can recover *generalizability*, though it risks losing learned information.

4. Method

4.1. Hare & Tortoise Architecture

To maintain network plasticity while retaining valuable information, we present the Hare & Tortoise architecture, inspired by the complementary learning systems in the human brain (McClelland et al., 1995). This architecture comprises two networks: the Hare network for rapid adaptation and the Tortoise network for stable knowledge consolidation.

Hare network (h): Resembling the hippocampus, the Hare network rapidly adapts to new data by adjusting its parameters based on input-output pairs at each training step.

Tortoise network (t): Imitating the neocortex, the Tortoise network consistently accumulates knowledge over time by momentum updates (Tarvainen & Valpola, 2017) from the Hare network. This ensures slow and steady updates of knowledge acquired from the Hare network.

4.2. Training Process

The training process of the Hare & Tortoise involves distinct yet interconnected updates to the networks.

At each training step, the Hare network's parameters, θ_h, are updated using the gradient descent. Given an input x and its corresponding output y, the network is updated as:

$$\theta_h \leftarrow \theta_h - \alpha \nabla_{\theta_h} \mathcal{L}(h(x; \theta_h), y)$$

where α denotes the learning rate, and $\mathcal{L}$ is the loss function.

Subsequently, the parameters of the Tortoise network, θ_t, are updated using an exponential moving average based on the Hare network's parameters:

$$\theta_t \leftarrow \mu\theta_t + (1 - \mu)\theta_h$$

where μ denotes the momentum.

To maintain plasticity in the Hare network, we incorporate a soft form of Re-initialization by periodically resetting its parameters to the Tortoise network's parameters:

$$\theta_h \leftarrow \theta_t$$

This updating scheme has a number of desirable properties. The update rate for the Tortoise network μ gives a means of controlling how far from the initialization the parameters can deviate. By setting a small update rate (i.e., large momentum value), the Tortoise network can gradually incorporate useful information from the Hare network. Since the Hare network re-starts from the Tortoise weights upon every reset, it can converge more quickly than if it had been randomly initialized. Hard resets of the Hare network also provide an opportunity to escape from bad local minima which may contain spurious features.

The PyTorch-like pseudocode is described in Algorithm 1.

4.3. Implementation

In all experimental setups, we utilized the Tortoise network for predictions, yielding stable and generalized outputs by ensembling the Hare network's parameters over time (Tarvainen & Valpola, 2017; Anonymous, 2023).

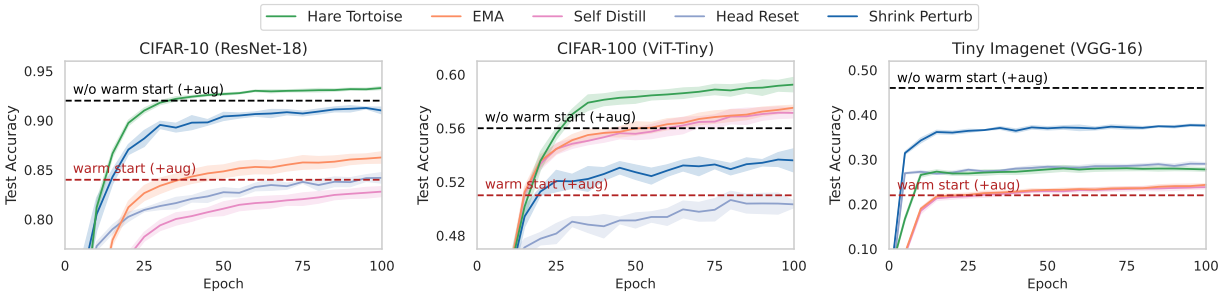

Figure 5. **Warm-Starting Results.** This graph presents the effectiveness of Hare & Tortoise in warm-starting experiments, compared to EMA, Self-Distillation, and Re-initialization methods. Hare & Tortoise shows superior performance in CIFAR-10 (ResNet-18) and CIFAR-100 (ViT-Tiny), while reinitialization shows greater effectiveness in Tiny ImageNet (VGG-16) with severe generalization loss.

Algorithm 1 Hare & Tortoise Pseudocode (Pytorch-like)

```
# h:  hare network
# t:  tortoise network
# m:  momentum
# r:  reset interval
for step, (x, y) in enumerate(loader):
    # update hare network
    logits = h(x)
    loss = loss_fn(logits, y)
    loss.backward()
    optimizer.step(h.params)
    # update tortoise network
    h.parms = m*h.params + (1-m)*t.params
    if step % r == 0:
        h.params = t.params
```

For our warm-starting and continual learning experiments, we fixed the momentum value of $\mu = 0.999$ and set the reset interval to occur every 10 epoch across all experiments.

In reinforcement learning setup, the Hare & Tortoise architecture can be effortlessly integrated into the modern algorithms that employ momentum target networks (D'Oro et al., 2022; Schwarzer et al., 2023; Hansen et al., 2023; Fujimoto et al., 2023). Here, the online network acts as the Hare network, and the target network serves as the Tortoise network. This integration requires only a single line of additional code, which periodically reinitializes the online network to the target network. Following the previous setups (D'Oro et al., 2022; Schwarzer et al., 2023), we used a momentum value of $\mu = 0.995$ and set the reset interval to 4000 for every gradient update step.

5. Experiments

We evaluate the effectiveness of Hare & Tortoise on maintaining generalizability in three distinct scenarios: warm-starting, continual learning, and reinforcement learning.

5.1. Warm-Starting

Following our warm-starting experiments in Section 3, we assessed the Hare & Tortoise architecture against reinitialization methods like Shrink & Perturb and Head Reset. We integrated L2 Regularization and Data Augmentation in all experiments due to their proven generalization benefits.

We also compared our method with Exponential Moving Average (EMA) and Self-Distillation (Tarvainen & Valpola, 2017; Allen-Zhu & Li, 2020), to evaluate the impact of reinitialization of the Hare network on the Tortoise network. The Hare & Tortoise architecture is a unique form of self-distillation, where the Tortoise guides the Hare to not deviate, but allows more freedom to explore the optimization landscape. For both EMA and Self-Distillation, we maintained a constant momentum value of $\mu = 0.999$, and tuned the distillation strengths from $\{1, 5, 10\}$.

As illustrated in Figure 5, using EMA consistently brings benefits, particularly in CIFAR-100 using ViT-Tiny architecture. This improved performance is likely due to ViT's convergence on sharp loss curvature, which further benefits from EMA's parameter smoothing (Chen et al., 2021; Park & Kim, 2022). The Hare & Tortoise consistently outperformed both EMA and Self-Distillation, which verifies its effectiveness in maintaining plasticity and encouraging the Hare network's exploration in the optimization landscape.

In comparison to other reinitialization strategies, Hare & Tortoise not only outperformed them but also surpassed the performance of fresh networks in CIFAR-10 with ResNet-18 and CIFAR-100 with ViT-Tiny. However, its effectiveness was less pronounced on Tiny ImageNet with VGG-16. This implies that reinitialization can be particularly beneficial in scenarios with substantial generalization loss.

To further understand the robustness of Hare & Tortoise, we conducted an ablation study by varying momentum value and reset interval (Appendix D). The results indicate that Hare & Tortoise remains resilient to their changes, with larger momentum generally improving performance.

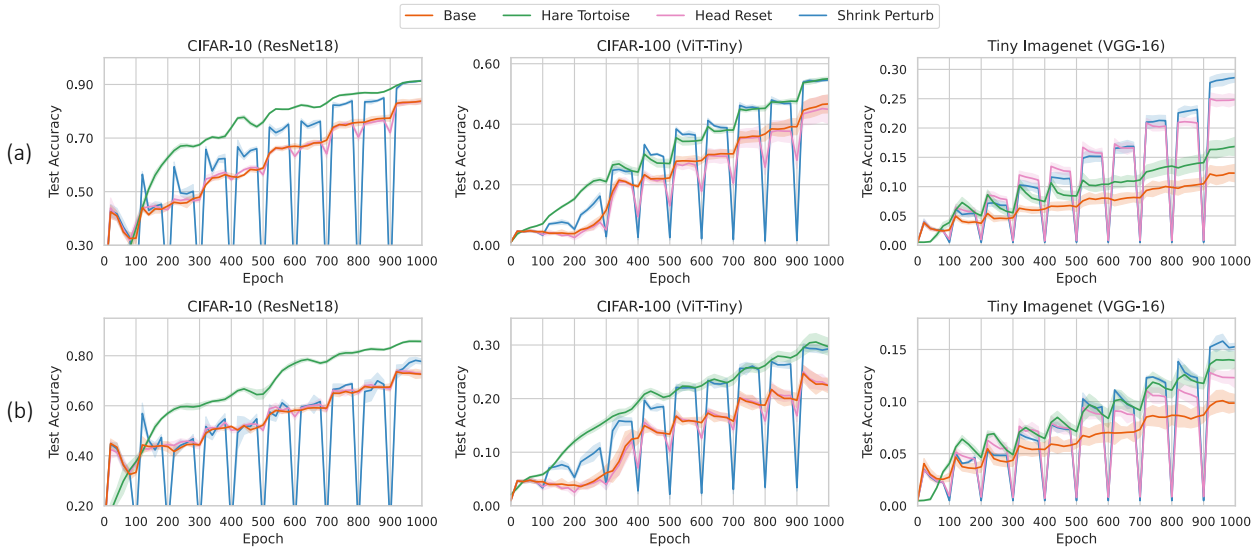

Figure 6. **Continual Learning Results.** The dataset was divided into 10 chunks, each undergoing 100 epochs of training with label noise decreasing from 0.5 to 0.0. **(a)** Shows outcomes with full dataset access, where Hare & Tortoise consistently outperform Shrink & Perturb in CIFAR-10 and CIFAR-100, but are less effective in Tiny ImageNet with significant generalization loss. **(b)** Illustrates results under limited access, with a buffer size of 5000. Here, Hare & Tortoise shows strong performance in CIFAR-10 and CIFAR-100 and narrows the gap with Shrink & Perturb in Tiny ImageNet, demonstrating its effective knowledge retention.

5.2. Continual Learning

Although Hare & Tortoise architecture effectively prevented plasticity loss in our warm-starting experiments, the Shrink & Perturb showed notable effectiveness in Tiny ImageNet. However, standard reinitialization methods face challenges in continual learning scenarios, which causes abrupt drops in online accuracy and increases computational costs for performance recovery. Moreover, when data access is limited, they risk losing valuable information as well.

To assess whether the Hare & Tortoise can overcome these challenges, we extended our warm-starting experiments into a continual learning framework, which extends the training into 10 phases. Each phase had a fixed subset ratio of 0.1 and label noise was decreased from 0.5 to 0, using 100 epochs for training. Two data access scenarios were tested: one with full dataset access and another with limited access with a buffer size of 5000.

As depicted in Figure 6.(a), under full data access, Hare & Tortoise excelled in CIFAR-10 and CIFAR-100, avoiding the abrupt performance drops associated with reinitialization. However, in Tiny ImageNet, where generalization loss is severe, reinitialization proved to be a more effective choice, despite its periodic performance drops.

In the limited data access scenario (Figure 6.(b)), Hare & Tortoise maintained strong performance in CIFAR-10 and CIFAR-100. The gap between Hare & Tortoise and other reinitialization strategies in Tiny ImageNet was less pronounced, suggesting that while reinitialization has its advantages, it can also lead to the loss of useful information for succeeding in future tasks. In contrast, Hare & Tortoise exhibited improved long-term information retention through effective knowledge consolidation.

5.3. Reinforcement Learning

Finally, we evaluate Hare & Tortoise (H&T) in reinforcement learning setup, analyzing its performance with two algorithms on the Atari-100K benchmark (Kaiser et al., 2019). The first, DrQ (Kostrikov et al., 2020), follows Mnih et al. (2015)'s architecture, featuring three convolutional layers followed by two fully connected layers, and includes random cropping augmentation. The second, BBF (Schwarzer et al., 2023), uses a ResNet-style architecture (Espeholt et al., 2018) with a custom reset protocol, higher replay ratio, and its use of auxiliary self-supervised objective (Schwarzer et al., 2020).

We integrated Hare & Tortoise into these algorithms by reinitializing the online network to the target network every 4000 steps, which only requires a single line of additional code. Since the hyperparameters of the BBF were carefully tuned based on reset intervals, we integrated Hare & Tortoise with reinitializations but intentionally excluded the self-supervised objective. This decision was based on our interpretation that while BBF's self-supervised objective aids in understanding temporal dynamics, it simultaneously constrains the online network's divergence from the momen-

Table 1. **Atari-100k Results.** BBF results without Hare & Tortoise come from the original paper (Schwarzer et al., 2023). All the other experiments, including DrQ, were conducted based on their original code and averaged over 5 random seeds with a replay ratio of 2.

Algorithm	Architecture	S&P	HR	H&T	SSL	GPU hours	IQM ↑	Median ↑	Mean ↑	OG ↓
DrQ (Kostrikov et al., 2020)	3-layer ConvNet	-	-	-	-	0.5	0.243	0.193	0.468	0.642
		✓	-	-	-		0.139	0.138	0.458	0.728
		-	-	✓	-		0.287	0.260	0.471	0.617
		-	20k	-	-		**0.332**	0.254	**0.694**	**0.580**
		-	40k	-	-		0.288	0.241	0.532	0.607
		-	40k	✓	-		**0.328**	**0.329**	0.584	**0.583**
BBF (Schwarzer et al., 2023)	15-layer ResNet	✓	✓	-	-	1.4	0.826	0.711	1.737	0.397
		✓	✓	✓	-		0.891	**0.749**	1.719	**0.372**
		✓	✓	-	✓	2.8	**0.940**	**0.755**	**2.175**	0.377

tum target network, similar to Hare & Tortoise. Thus, we aimed to isolate the effects of this constraint and assess the effectiveness of Hare & Tortoise.

Table 1 presents our experimental results. Adding Hare & Tortoise in DrQ led to a modest improvement in the median score. However, the most effective approach was resetting the head every 20,000 steps, likely due to the limited depth of DrQ's encoder and the head network's significant parameter portion (90%). While a 40,000-step interval Head Reset wasn't wholly effective in mitigating plasticity loss, combining it with Hare & Tortoise achieved comparable performance to the 20,000-step interval Head Reset.

For BBF, adding Hare & Tortoise significantly improved both IQM and OG scores, without incurring any extra computational costs. While Hare & Tortoise's performance on IQM and Mean scores was lower than using self-supervised learning objective, it surpassed OG scores and required only half of the computational resources. It's important to note that we did not adjust any original hyperparameters, including the momentum value. Therefore, with further optimization and exploration, we believe that Hare & Tortoise has great potential to effectively alleviate plasticity loss.

6. Conclusion and Future Work

In the research community, there's a common belief that enhancing a neural network's trainability will naturally improve its generalizability in continuous learning scenarios. Past studies have shown that when neural networks are continually trained over extended periods, they may struggle to minimize training loss, leading to diminished generalization. Consequently, efforts have been made to bolster generalizability by focusing on improving trainability (Abbas et al., 2023; Lyle et al., 2022; Sokar et al., 2023). However, these studies often employ smaller network architectures (e.g., 3 to 5 layers) compared to today's deep and large networks.

Our study investigates whether trainability concerns persist with modern datasets and architectures. By revisiting warm-starting experiments (Ash & Adams, 2020) with various modern network architectures, we found no trainability issues even with prolonged training on small, noisy datasets. Furthermore, methods aimed at enhancing trainability did not improve generalization and often led to overfitting. Surprisingly, simply reinitializing the network proved effective, despite the potential risk of losing valuable knowledge.

To address this, we developed the Hare & Tortoise algorithm. It combines two types of networks: the Hare, which optimizes weights rapidly, and the Tortoise, which updates weights slowly by momentum average. The Tortoise's slowly updated weights serve as starting points for periodic resets, which helps the Hare to rapidly adapt and escape from bad local minima. It outperformed Shrink & Perturb in continual learning experiments and enhanced the efficacy of state-of-the-art reinforcement learning algorithms.

However, there's still much room to explore. A key unanswered question is why warm-started models fail to generalize new tasks. The Hare & Tortoise offers valuable insights for this question; the Tortoise's effective performance implies that certain regions in the optimization landscape can perform well on current tasks while preserving their plasticity to generalize on new tasks. Future improvements should focus on identifying and exploiting these regions, possibly reducing or removing the need for hard reinitializations.

7. Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00075 Artificial Intelligence Graduate School Program (KAIST), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1A2B5B02001913 and No. 2021R1A2C209494311).

Also, we would like to express our gratitude to David Abel for his valuable feedback on this paper.

8. Broader Impact

Our research focuses on enhancing neural network plasticity, which is important to develop intelligent robotics. This enhancement allows intelligent robots to effectively learn, adapt, and respond to dynamic environments, leading to improved efficiency in manufacturing, logistics, and autonomous driving.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Abbas, Z., Zhao, R., Modayil, J., White, A., and Machado, M. C. Loss of plasticity in continual deep reinforcement learning. *arXiv preprint arXiv:2303.07507*, 2023.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Allen-Zhu, Z. and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

Anand, N. and Precup, D. Prediction and control in continual reinforcement learning. *arXiv preprint arXiv:2312.11669*, 2023.

Anonymous. Exponential moving average of weights in deep learning: Dynamics and benefits. *Submitted to Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=2M9CUnYnBA. Under review.

Arani, E., Sarfraz, F., and Zonooz, B. Learning fast, learning slow: A general continual learning method based on complementary learning system. *arXiv preprint arXiv:2201.12604*, 2022.

Asadi, K., Fakoor, R., and Sabach, S. Resetting the optimizer in deep rl: An empirical study. *arXiv preprint arXiv:2306.17833*, 2023.

Ash, J. and Adams, R. P. On warm-starting neural network training. *Advances in neural information processing systems*, 33:3884–3894, 2020.

Berariu, T., Czarnecki, W., De, S., Bornschein, J., Smith, S., Pascanu, R., and Clopath, C. A study on the plasticity of neural networks. *arXiv preprint arXiv:2106.00042*, 2021.

Chen, X., Hsieh, C.-J., and Gong, B. When vision transformers outperform resnets without pre-training or strong data augmentations. *arXiv preprint arXiv:2106.01548*, 2021.

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

Dohare, S., Hernandez-Garcia, J. F., Rahman, P., Sutton, R. S., and Mahmood, A. R. Maintaining plasticity in deep continual learning. *arXiv preprint arXiv:2306.13812*, 2023.

D'Oro, P., Schwarzer, M., Nikishin, E., Bacon, P.-L., Bellemare, M. G., and Courville, A. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Elsayed, M. and Mahmood, A. R. Addressing loss of plasticity and catastrophic forgetting in continual learning. *arXiv preprint arXiv:2404.00781*, 2024.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Frankland, P. W., Köhler, S., and Josselyn, S. A. Hippocampal neurogenesis and forgetting. *Trends in neurosciences*, 36(9):497–503, 2013.

Frati, L., Traft, N., Clune, J., and Cheney, N. Reset it and forget it: Relearning last-layer weights improves continual and transfer learning. *arXiv preprint arXiv:2310.07996*, 2023.

Fujimoto, S., Chang, W.-D., Smith, E. J., Gu, S. S., Precup, D., and Meger, D. For sale: State-action representation learning for deep reinforcement learning. *arXiv preprint arXiv:2306.02451*, 2023.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Gogianu, F., Berariu, T., Rosca, M. C., Clopath, C., Busoniu, L., and Pascanu, R. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *International Conference on Machine Learning*, pp. 3734–3744. PMLR, 2021.

Gomez-Villa, A., Twardowski, B., Wang, K., and van de Weijer, J. Plasticity-optimized complementary networks for unsupervised continual learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1690–1700, 2024.

Google, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Gravitz, L. The forgotten part of memory. *Nature*, 571 (7766):S12–S12, 2019.

Hansen, N., Su, H., and Wang, X. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

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

Kostrikov, I., Yarats, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krogh, A. and Hertz, J. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.

Kumar, A., Agarwal, R., Ghosh, D., and Levine, S. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020.

Kumar, S., Marklund, H., Rao, A., Zhu, Y., Jeon, H. J., Liu, Y., and Van Roy, B. Continual learning as computationally constrained reinforcement learning. *arXiv preprint arXiv:2307.04345*, 2023a.

Kumar, S., Marklund, H., and Van Roy, B. Maintaining plasticity via regenerative regularization. *arXiv preprint arXiv:2308.11958*, 2023b.

Kumaran, D., Hassabis, D., and McClelland, J. L. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

LeCun, Y., Cortes, C., and Burges, C. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

Lee, H., Cho, H., Kim, H., Gwak, D., Kim, J., Choo, J., Yun, S.-Y., and Yun, C. Plastic: Improving input and label plasticity for sample efficient reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Lewandowski, A., Tanaka, H., Schuurmans, D., and Machado, M. C. Curvature explains loss of plasticity. *arXiv preprint arXiv:2312.00246*, 2023.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lyle, C., Rowland, M., and Dabney, W. Understanding and preventing capacity loss in reinforcement learning. *Proc. the International Conference on Learning Representations (ICLR)*, 2022.

Lyle, C., Zheng, Z., Nikishin, E., Pires, B. A., Pascanu, R., and Dabney, W. Understanding plasticity in neural networks. *Proc. the International Conference on Machine Learning (ICML)*, 2023.

Lyle, C., Zheng, Z., Khetarpal, K., van Hasselt, H., Pascanu, R., Martens, J., and Dabney, W. Disentangling the causes of plasticity loss in neural networks. *arXiv preprint arXiv:2402.18762*, 2024.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Nauman, M., Bortkiewicz, M., Ostaszewski, M., Miłoś, P., Trzciński, T., and Cygan, M. Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning. *arXiv preprint arXiv:2403.00514*, 2024.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P.-L., and Courville, A. The primacy bias in deep reinforcement learning. 2022.

Noukhovitch, M., Lavoie, S., Strub, F., and Courville, A. Language model alignment with elastic reset. *arXiv preprint arXiv:2312.07551*, 2023.

Obando, C., Johan, Bellemare, M., and Castro, P. S. Small batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Park, N. and Kim, S. How do vision transformers work? *arXiv preprint arXiv:2202.06709*, 2022.

Pezeshki, M., Kaba, O., Bengio, Y., Courville, A. C., Precup, D., and Lajoie, G. Gradient starvation: A learning proclivity in neural networks. *Advances in Neural Information Processing Systems*, 34:1256–1272, 2021.

Pham, Q., Liu, C., and Hoi, S. Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, 34:16131–16144, 2021.

Pham, Q., Liu, C., Sahoo, D., and Hoi, S. C. Learning fast and slow for online time series forecasting. *arXiv preprint arXiv:2202.11672*, 2022.

Pritzel, A., Uria, B., Srinivasan, S., Badia, A. P., Vinyals, O., Hassabis, D., Wierstra, D., and Blundell, C. Neural episodic control. In *International conference on machine learning*, pp. 2827–2836. PMLR, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115: 211–252, 2015.

Ryan, T. J. and Frankland, P. W. Forgetting as a form of adaptive engram cell plasticity. *Nature Reviews Neuroscience*, 23(3):173–186, 2022.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

Shang, W., Sohn, K., Almeida, D., and Lee, H. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *international conference on machine learning*, pp. 2217–2225. PMLR, 2016.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

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

Smith, J. S., Tian, J., Halbe, S., Hsu, Y.-C., and Kira, Z. A closer look at rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2409–2419, 2023.

Sodhani, S., Chandar, S., and Bengio, Y. Toward training recurrent neural networks for lifelong learning. *Neural computation*, 32(1):1–35, 2020.

Sokar, G., Agarwal, R., Castro, P. S., and Evci, U. The dormant neuron phenomenon in deep reinforcement learning. *arXiv preprint arXiv:2302.12902*, 2023.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Takahashi, R., Matsubara, T., and Uehara, K. Data augmentation using random image cropping and patching for deep cnns. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9):2917–2931, 2019.

Tarvainen, A. and Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

Tesauro, G. et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Xiao, L., Pennington, J., and Schoenholz, S. Disentangling trainability and generalization in deep neural networks. In *International Conference on Machine Learning*, pp. 10462–10472. PMLR, 2020.

Xu, G., Zheng, R., Liang, Y., Wang, X., Yuan, Z., Ji, T., Luo, Y., Liu, X., Yuan, J., Hua, P., et al. Drm: Mastering visual reinforcement learning through dormant ratio minimization. *arXiv preprint arXiv:2310.19668*, 2023.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Zaidi, S., Berariu, T., Kim, H., Bornschein, J., Clopath, C., Teh, Y. W., and Pascanu, R. When does re-initialization work? In *Proceedings on*, pp. 12–26. PMLR, 2023.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Zhou, H., Vani, A., Larochelle, H., and Courville, A. Fortuitous forgetting in connectionist networks. *arXiv preprint arXiv:2202.00155*, 2022.

Zilly, J., Achille, A., Censi, A., and Frazzoli, E. On plasticity, invariance, and mutually frozen weights in sequential task learning. *Advances in Neural Information Processing Systems*, 34:12386–12399, 2021.

A. Implementation Details for Warm-Starting

This section outlines the implementation details of our exploration into the impact of warm-starting on neural network performance, as discussed in Section 3. Our study encompasses a variety of datasets and neural network architectures.

A.1. Datasets

Our study utilized four datasets, each with distinct complexity and characteristics, to thoroughly evaluate the effects of warm-starting. The following datasets were used:

MNIST (LeCun et al., 1998): Contains 70,000 grayscale images (60,000 for training, 10,000 for testing), with each image being a 28x28 pixel representation of handwritten digits across 10 classes. MNIST is a basic dataset commonly used to benchmark machine learning algorithms due to its simplicity.

CIFAR-10 (Krizhevsky, 2009): Consists of 60,000 color images (50,000 for training, 10,000 for testing) across 10 classes, each image sized at 32x32 pixels. This dataset features objects and animals, offering a higher level of classification challenge than MNIST because of its color diversity and intricate patterns.

CIFAR-100 (Krizhevsky, 2009): Similar in structure to CIFAR-10 but with 100 classes, totaling the same number of 32x32 color images but divided among more categories, with each class having 600 images. CIFAR-100 raises the complexity by requiring finer differentiation between a broader array of classes.

Tiny ImageNet: A scaled-down version of the ImageNet dataset (Russakovsky et al., 2015), Tiny ImageNet includes 100,000 training and 10,000 testing images, resized to 64x64 pixels, across 200 classes. It challenges models on a much wider range of classes, significantly more than the MNIST or CIFAR datasets.

Table 2. Description of Dataset.

Datasets	MNIST	CIFAR-10	CIFAR-100	Tiny ImageNet
Type	Grayscale	Color	Color	Color
Image size	28×28	32×32	32×32	64×64
# Classes	10	10	100	200
Train size	70,000	50,000	50,000	10,0000
Test size	10,000	10,000	10,000	10,000

A.2. Architectures

In our research, we employed well-known neural network architectures, each paired with a specific benchmark dataset. These combinations include:

3-layer MLP: Used for analyzing the MNIST dataset, this architecture consisted of a 3-layer Multi-Layer Perceptron (MLP). Each hidden layer contained 100 units, and the model comprised a backbone (first and second layers) and a head (last fully connected layer). Hyperparameters for this model included a learning rate of 0.001, a weight decay of 0.0001, and batch normalization in the backbone.

ResNet-18 (He et al., 2016): Employed for CIFAR-10, this architecture is known for its deep residual learning framework and residual connections. To optimize for computational efficiency and input resolution, we removed the stem layers and utilized specific hyperparameters, including a learning rate of 0.001, a weight decay of 0.00001, and batch normalization.

ViT-Tiny (Dosovitskiy et al., 2020): Utilized for CIFAR-100, this model was adapted from the original DeiT-Ti model (Touvron et al., 2021). It featured a 4x4 patch size, an embedding dimension of 192, three attention heads, and a depth of twelve layers. We used a learning rate of 0.003, weight decay of 0.05, layer normalization, and a dropout rate of 0.1.

VGG-16 (Simonyan & Zisserman, 2014): Selected for Tiny ImageNet, this model is known for its traditional ConvNet architecture, comprising several consecutive convolutional layers followed by fully connected layers with hidden dimensions of 1024. Hyperparameters included a learning rate of 0.001, weight decay of 0.0001, batch normalization, and no dropout.

Learning rates and weight decay were meticulously determined through grid search.

A.3. Training Details

In this section, we provide a comprehensive overview of the hyperparameters and configurations for each baseline model. We begin by presenting the base hyperparameters, which are consistently applied across all models and experiments. These global hyperparameters are summarized in Table 3.

Table 3. Base Hyperparameters for Warm-Starting.

Hyperparameter	Value
Epochs	100
Optimizer	AdamW
Optimizer Hyperparameters (β_1, β_2)	(0.9, 0.999)
Batch Size	256
Learning Rate Scheduler	Warmup
Warmup Ratio	0.1
Initial Learning Rate	0.0
Gradient Clip.	0.5

Our warm-starting approach, following the methodology outlined by Ash & Adams (2020), involves initializing the network with subsets of the original dataset. These subsets are created at ratios of 0.1, 0.2, 0.3, 0.4, 0.6, 0.8, and 1.0. To ensure an equivalent number of updates across different initializations, we adjust the training epochs for each subset, setting them at 1000, 500, 250, 166, 125, and 100, respectively. Furthermore, we incorporate label noise into our experiments as described by Zhang et al. (2021). This entails replacing the original labels with uniformly distributed random labels.

To ensure the robustness and reliability of our results, each experimental setup is repeated with multiple random seeds. Specifically, we utilize 5 random seeds for all datasets and models, except for Tiny ImageNet, where we employ 3 random seeds for computational efficiency.

For all experiments, we used an NVIDIA RTX 3090 GPU for neural network training and a 32-core AMD EPYC 7502 CPU for multi-threaded tasks.

A.4. Baseline Methods

In this section, we provide details of baseline methods and the hyperparameters we used in Section 3.4.

L2 Regularization: L2 Regularization is a widely adopted technique in neural network training. It works by adding a penalty term to the loss function, which is proportional to the square of the magnitude of the network weights. This penalty encourages the model to learn simpler patterns, thus increasing generalizability. In our experiments, we tested L2 regularization with varying intensities $\{1.0, 0.1, 0.01, 0.001, 0.0001\}$.

Data Augmentation: We applied data augmentation techniques, including horizontal flipping and random cropping with 4-pixel padding, to the original datasets for improved model generalization (Laskin et al., 2020; Kostrikov et al., 2020).

Concatenated Rectified Linear Units (CReLU): CReLU is an activation function introduced by Shang et al. (2016), which combines the positive and negative parts of the ReLU function as $[\text{ReLU}(x), -\text{ReLU}(-x)]$. It has been demonstrated to be effective in preserving trainability in continual reinforcement learning Abbas et al. (2023). We incorporated CReLU in the head of each model, reducing the parameter count by half in each hidden layer. For the 3-Layer MLP, we added CReLU in every activation function.

Recycling Dormant Neurons (ReDo): Sokar et al. (2023) identified the Dormant Neuron Phenomenon, where certain neurons exhibit significantly reduced activity levels during training, negatively impacting trainability. ReDo periodically scans all layers for τ-dormant neurons, reinitializes their weights, and sets their outgoing weights to zero. We tested various τ values including $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$, with reinitialization applied every 5 epochs.

Regenerative Regularization (Regen): Developed by Kumar et al. (2023b), this method embeds L2 regularization towards initial parameters in the loss function. It has shown effectiveness in retaining plasticity during continual learning by implicitly and smoothly resetting low-utility weights. We tested different regeneration rates $\{1.0, 0.1, 0.01, 0.001, 0.0001\}$, and used the best value for each experiment.

Spectral Decoupling (Spectral): Spectral Decoupling, as discussed by Pezeshki et al. (2021), addresses the Gradient Starvation issue in over-parameterized neural networks. Gradient Starvation is a phenomenon in which the network disproportionately focuses on a few features, neglecting others which leads to sub-optimal utilization of predictive features and affects trainability. Spectral Decoupling adds an L2 penalty to the network's logits, promoting a more balanced feature learning. We optimized the spectral rate from various values $\{1.0, 0.1, 0.01, 0.001, 0.0001, 0.00001\}$.

Shrink & Perturb: The study by Ash & Adams (2020) discovered that training neural networks with only half of the dataset initially can negatively affect their generalizability. To address this, they introduced the Shrink & Perturb method. This approach involves periodically reducing the current weights (shrinking) and introducing small random noises (perturbation). In our experiment, we tested shrinkage values of $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ and used the optimal value to be 0.8. This process was used before proceeding to the subsequent training phase.

Head Reset: The Primacy Bias refers to an agent's tendency to overfit earlier experiences. This often results in a loss of learning ability in Reinforcement Learning, as described by (Nikishin et al., 2022). To address this issue, it has been discovered that regularly resetting the agent's head can be effective. In our experiments, we reinitialized the head network before proceeding to the subsequent training phase.

B. Experimental Results of Warm-Starting

B.1. Train and Test Accuracies

In Section 3, we detail our warm-start experiments, which assess model performance across various subset ratios and label noise levels, with results summarized in Figure 2.(c). The experiments in Table 4 yield near-perfect training accuracies for MNIST, CIFAR-10, and Tiny ImageNet, as shown in Table 1, indicating robust training irrespective of subset or noise variations. CIFAR-100 also demonstrates high training accuracy across on par with a freshly initialized network.

However, Table 5 highlights a decline in test accuracies compared to training, underscoring a gap between the models' ability to learn from training data and their generalization to unseen data. This contrast points to the challenges in maintaining model generalizability across different experimental setups.

Table 4. **Final Train Accuracy.** The overall train accuracy is similar regardless of the subset ratio and label noise ratio.

Subset Ratio	Label Noise	MNIST (3-layer MLP)	CIFAR-10 (ResNet-18)	CIFAR-100 (ViT-Tiny)	Tiny ImageNet (VGG-16)
0.1	0.0	1.00±0.0003	1.00±0.0006	0.91±0.0014	0.98±0.0011
0.1	0.1	1.00±0.0002	1.00±0.0002	0.91±0.0019	0.98±0.0018
0.1	0.2	1.00±0.0002	1.00±0.0004	0.91±0.0020	0.98±0.0002
0.1	0.3	1.00±0.0003	1.00±0.0004	0.91±0.0011	0.97±0.0001
0.1	0.4	1.00±0.0002	1.00±0.0008	0.91±0.0025	0.97±0.0012
0.1	0.5	1.00±0.0001	1.00±0.0005	0.91±0.0016	0.97±0.0003
0.2	0.0	1.00±0.0003	1.00±0.0006	0.91±0.0018	0.98±0.0005
0.2	0.1	1.00±0.0001	1.00±0.0003	0.91±0.0009	0.98±0.0004
0.2	0.2	1.00±0.0004	1.00±0.0005	0.91±0.0019	0.98±0.0003
0.2	0.3	1.00±0.0003	1.00±0.0007	0.91±0.0006	0.98±0.0003
0.2	0.4	1.00±0.0003	1.00±0.0008	0.91±0.0014	0.98±0.0003
0.2	0.5	1.00±0.0004	1.00±0.0004	0.91±0.0009	0.98±0.0006
0.4	0.0	1.00±0.0001	1.00±0.0008	0.92±0.0015	0.99±0.0002
0.4	0.1	1.00±0.0002	1.00±0.0009	0.92±0.0016	0.99±0.0002
0.4	0.2	1.00±0.0002	1.00±0.0005	0.92±0.0013	0.98±0.0001
0.4	0.3	1.00±0.0002	1.00±0.0007	0.91±0.0020	0.98±0.0003
0.4	0.4	1.00±0.0001	1.00±0.0006	0.91±0.0014	0.98±0.0002
0.4	0.5	1.00±0.0002	1.00±0.0002	0.91±0.0014	0.98±0.0004
0.6	0.0	1.00±0.0002	1.00±0.0004	0.92±0.0014	0.99±0.0007
0.6	0.1	1.00±0.0001	1.00±0.0006	0.92±0.0012	0.99±0.0005
0.6	0.2	1.00±0.0004	1.00±0.0003	0.92±0.0013	0.99±0.0008
0.6	0.3	1.00±0.0003	1.00±0.0004	0.92±0.0011	0.99±0.0006
0.6	0.4	1.00±0.0004	1.00±0.0012	0.91±0.0013	0.98±0.0004
0.6	0.5	1.00±0.0002	1.00±0.0008	0.91±0.0011	0.98±0.0007
0.8	0.0	1.00±0.0001	1.00±0.0005	0.92±0.0010	0.99±0.0006
0.8	0.1	1.00±0.0001	1.00±0.0006	0.92±0.0022	0.99±0.0008
0.8	0.2	1.00±0.0004	1.00±0.0002	0.92±0.0009	0.99±0.0004
0.8	0.3	1.00±0.0002	1.00±0.0005	0.92±0.0015	0.99±0.0002
0.8	0.4	1.00±0.0002	1.00±0.0007	0.92±0.0027	0.99±0.0006
0.8	0.5	1.00±0.0001	1.00±0.0007	0.91±0.0051	0.98±0.0003
1.0	0.0	1.00±0.0002	1.00±0.0006	0.93±0.0007	0.99±0.0005
1.0	0.1	1.00±0.0001	1.00±0.0009	0.92±0.0013	0.99±0.0004
1.0	0.2	1.00±0.0004	1.00±0.0003	0.92±0.0010	0.99±0.0005
1.0	0.3	1.00±0.0002	1.00±0.0005	0.92±0.0011	0.99±0.0004
1.0	0.4	1.00±0.0003	1.00±0.0015	0.92±0.0017	0.99±0.0004
1.0	0.5	1.00±0.0003	1.00±0.0009	0.91±0.0022	0.98±0.0002

Table 5. **Final Test Accuracy.** Unlike train accuracy, test accuracy showed different results depending on the subset ratio and label noise ratio, indicating the existence of loss of generalizability.

Subset Ratio	Label Noise	MNIST (3-layer MLP)	CIFAR-10 (ResNet-18)	CIFAR-100 (ViT-Tiny)	Tiny ImageNet (VGG-16)
0.1	0.0	0.98±0.0016	0.74±0.0023	0.45±0.0027	0.18±0.0036
0.1	0.1	0.97±0.0008	0.72±0.0050	0.45±0.0032	0.16±0.0029
0.1	0.2	0.97±0.0006	0.71±0.0123	0.44±0.0053	0.15±0.0039
0.1	0.3	0.97±0.0012	0.70±0.0054	0.44±0.0066	0.14±0.0057
0.1	0.4	0.97±0.0013	0.69±0.0095	0.44±0.0033	0.13±0.0043
0.1	0.5	0.97±0.0019	0.69±0.0067	0.44±0.0089	0.12±0.0046
0.2	0.0	0.98±0.0010	0.75±0.0047	0.45±0.0018	0.24±0.0008
0.2	0.1	0.97±0.0009	0.72±0.0049	0.44±0.0039	0.22±0.0034
0.2	0.2	0.97±0.0010	0.71±0.0047	0.44±0.0026	0.20±0.0036
0.2	0.3	0.97±0.0009	0.69±0.0028	0.44±0.0032	0.18±0.0020
0.2	0.4	0.97±0.0011	0.68±0.0032	0.44±0.0041	0.16±0.0039
0.2	0.5	0.97±0.0012	0.66±0.0057	0.44±0.0029	0.15±0.0032
0.4	0.0	0.98±0.0016	0.77±0.0038	0.47±0.0068	0.36±0.0012
0.4	0.1	0.97±0.0012	0.75±0.0027	0.46±0.0047	0.33±0.0015
0.4	0.2	0.97±0.0018	0.73±0.0035	0.44±0.0027	0.32±0.0026
0.4	0.3	0.97±0.0013	0.71±0.0053	0.44±0.0027	0.30±0.0061
0.4	0.4	0.97±0.0014	0.69±0.0066	0.44±0.0027	0.27±0.0026
0.4	0.5	0.97±0.0018	0.67±0.0092	0.43±0.0043	0.25±0.0029
0.6	0.0	0.98±0.0018	0.79±0.0042	0.48±0.0042	0.40±0.0032
0.6	0.1	0.97±0.0009	0.77±0.0045	0.47±0.0042	0.38±0.0012
0.6	0.2	0.97±0.0014	0.75±0.0024	0.46±0.0042	0.36±0.0032
0.6	0.3	0.97±0.0012	0.73±0.0020	0.44±0.0065	0.33±0.0041
0.6	0.4	0.97±0.0011	0.71±0.0048	0.44±0.0030	0.30±0.0016
0.6	0.5	0.97±0.0010	0.69±0.0040	0.43±0.0040	0.27±0.0005
0.8	0.0	0.98±0.0014	0.81±0.0045	0.50±0.0050	0.42±0.0045
0.8	0.1	0.97±0.0006	0.78±0.0050	0.48±0.0076	0.40±0.0019
0.8	0.2	0.97±0.0014	0.77±0.0026	0.46±0.0054	0.38±0.0024
0.8	0.3	0.97±0.0014	0.75±0.0063	0.46±0.0041	0.35±0.0023
0.8	0.4	0.97±0.0019	0.74±0.0033	0.45±0.0054	0.33±0.0037
0.8	0.5	0.97±0.0018	0.73±0.0075	0.43±0.0093	0.29±0.0022
1.0	0.0	0.98±0.0011	0.81±0.0053	0.51±0.0078	0.43±0.0009
1.0	0.1	0.97±0.0012	0.79±0.0060	0.49±0.0052	0.41±0.0019
1.0	0.2	0.97±0.0013	0.78±0.0071	0.48±0.0035	0.40±0.0081
1.0	0.3	0.97±0.0011	0.77±0.0075	0.47±0.0052	0.37±0.0022
1.0	0.4	0.97±0.0022	0.76±0.0051	0.45±0.0037	0.34±0.0029
1.0	0.5	0.97±0.0010	0.75±0.0076	0.40±0.0376	0.30±0.0033

B.2. Influence of Dataset on Loss of Generalizability

In this section, as an extension of our investigation into the effects of dataset size and label noise discussed in Section 3.2, we aimed to understand the complex relationship between datasets and models in terms of generalization. To do this, we conducted experiments in which we swapped the models for the CIFAR-10 and Tiny ImageNet datasets. Specifically, we used ResNet-18 for Tiny ImageNet and VGG-16 for CIFAR-10. Each configuration was tested with two different random seeds, and the results are illustrated in Figure 7.

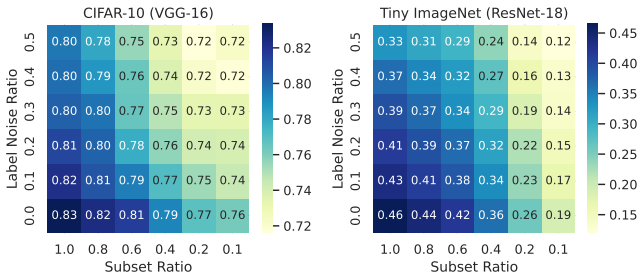

Figure 7. **Datasets and Models Affect Plasticity.** Results when swapping models between CIFAR-10 and Tiny ImageNet in the previous setting. It highlights the similar aspects of loss of generalizability when datasets are paired with different models.

Figure 7 demonstrates how the choice of datasets and models impacts plasticity. Comparing these results to the earlier findings shown in Figure 2.(c), we observed a consistent trend across datasets. Notably, Tiny ImageNet consistently exhibited a more significant loss in generalization ability compared to CIFAR-10, regardless of the model used. This highlights the critical role of the characteristics of the dataset in understanding the loss of generalizability.

B.3. Baseline Comparison: Train and Test Accuracies

Below are the final train and test accuracies from our experiments in Section 3.4.

Table 6. **Final Train Accuracy**. Warm-starting has no adverse effect on train accuracy in widely used architectures, achieving near-100% accuracy on CIFAR-10 and Tiny ImageNet. This highlights the significant gap between trainability and generalizability.

Dataset	Architecture	Baseline			Generalizability		Trainability				Re-initialization		
		WS	w/o WS	w/o WS + Aug	L2	Aug	Spectral	Regen	ReDo	CReLU	HR	SnP	SnP + L2 + Aug
CIFAR-10	ResNet-18	0.998	0.998	0.994	0.990	0.993	0.997	0.996	0.998	0.997	0.997	0.997	0.994
CIFAR-100	ViT-Tiny	0.907	0.912	0.852	0.865	0.867	0.913	0.906	0.908	0.890	0.911	0.915	0.872
T-ImageNet	VGG-16	0.972	0.984	0.974	0.925	0.963	0.997	0.979	0.972	0.964	0.979	0.981	0.970

Table 7. **Final Test Accuracy.** While trainability enhancements can yield high training accuracies, the best test performance is achieved by combining Shrink & Perturb with generalization techniques..

Dataset	Architecture	Baseline			Generalizability		Trainability				Re-initialization		
		WS	w/o WS	w/o WS + Aug	L2	Aug	Spectral	Regen	ReDo	CReLU	HR	SnP	SnP + L2 + Aug
CIFAR-10	ResNet-18	0.704	0.835	0.924	0.735	0.840	0.705	0.701	0.682	0.678	0.693	0.771	**0.910**
CIFAR-100	ViT-Tiny	0.442	0.500	0.555	0.457	0.511	0.444	0.447	0.445	0.440	0.445	0.467	**0.536**
T-ImageNet	VGG-16	0.115	0.439	0.463	0.221	0.227	0.118	0.127	0.118	0.118	0.177	0.229	**0.376**

In all experiments, training accuracy was on par with a freshly initialized model (w/o WS) for most methods, except for those aimed at preventing overfitting. Table 5 displays the final test accuracy results. In these experiments, trainability methods had limited impact, despite the increase in training accuracy in the Spectral Decomposition method. While generalizability methods (L2, Data Augmentation) generally improved performance compared to the baseline (warm-start), the performance was still significantly lower than freshly initialized models, indicating that these methods couldn't fully mitigate plasticity loss. Shrink & Perturb with generalizability methods outperformed all the other approaches, with its performance reaching close to training the fresh networks.

C. Relationship between Generalizability and Potential Indicators

The precise mechanisms underlying the loss of plasticity in neural networks remain incompletely understood. However, several factors have been proposed as potential indicators of this phenomenon. These factors include weight magnitude, the distance of current weight values from their initial values, the number of zero gradients or non-active ReLU units, and feature rank collapse, as highlighted in prior studies (Dohare et al., 2021; Zilly et al., 2021; Dohare et al., 2023; Kumar et al., 2023a;b; Abbas et al., 2023; Lyle et al., 2023; Lewandowski et al., 2023).

This study delves into the relationship between these metrics and the generalization ability of neural networks. Specifically, we focus on examining the Pearson correlation coefficients between these metrics and the final test accuracy, with each metric being measured at the end of the warm-starting phase. As depicted in Figure 8, we observed that none of these metrics exhibit a strong and consistent correlation across different datasets and architectures, although weight distance showed a subtle trend. Below, we provide details on how each metric is calculated:

Weight Magnitude: This metric is calculated as the cumulative average of the squared weights across all layers of the model, given by the formula:

$$\text{norm}(\mathbf{W}) = \sum_{l \in L} \frac{1}{\text{count}(l)} \sum_{w \in l} w^2$$

Here, L stands for the collection of layers, and w denotes the individual weights within layer l.

Weight Distance: It measures the mean squared distance between the current weights and their initial values for each layer:

$$\text{dist}(\mathbf{W}) = \sum_{l \in L} \frac{1}{\text{count}(l)} \sum_{w \in l} (w - w_0)^2$$

where w_0 represents the initial weight value.

Zero Gradient Ratio: Indicates the percentage of zero gradients in the features produced by the backbone network's output.

Feature Zero Activation Ratio: Measures the proportion of zero activations in the feature output of the backbone network.

Feature Rank (srank): Introduced by Kumar et al. (2020), this metric assesses the effective number of unique bases in the feature matrix (Φ), which we calculated from the output of the backbone network. Given the ordered set of singular values $\sigma_1(\mathbf{\Phi}) > \sigma_2(\mathbf{\Phi}) > ... > \sigma_n(\mathbf{\Phi})$, we measure it as:

$$\text{srank}(\mathbf{\Phi}) = \min_k \left(\frac{\sum_{i=1}^{k} \sigma_i(\mathbf{\Phi})}{\sum_{j=1}^{n} \sigma_j(\mathbf{\Phi})} \right) \geq 1 - \delta$$

where δ is set to 0.01, following from Kumar et al. (2020).

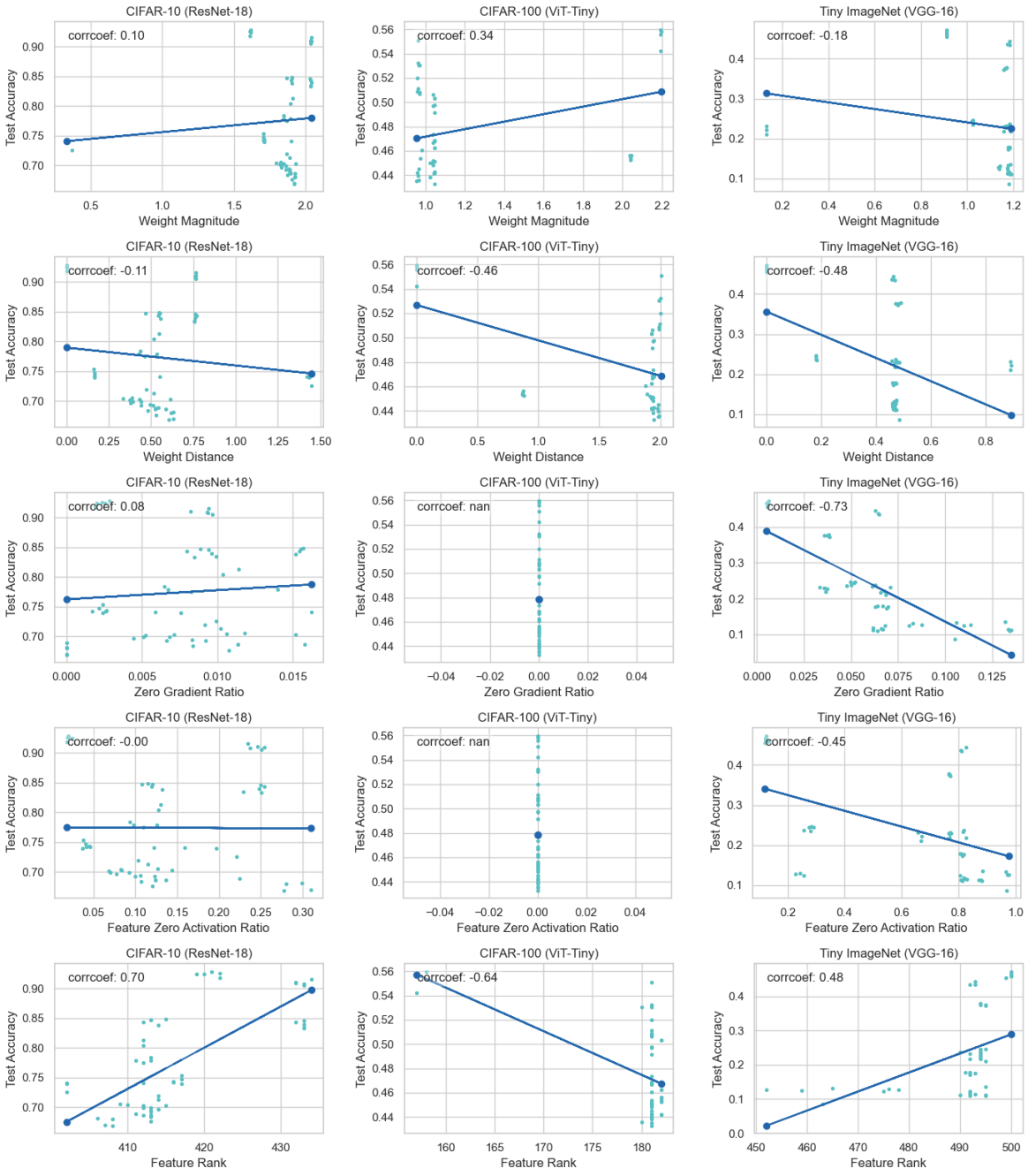

Figure 8. **Metrics**. None of the metrics exhibit a strong and consistent correlation to generalization performance.

D. Ablation Studies

In our ablation studies, we explore the robustness of our proposed method by varying two hyperparameters: momentum value and reset interval.

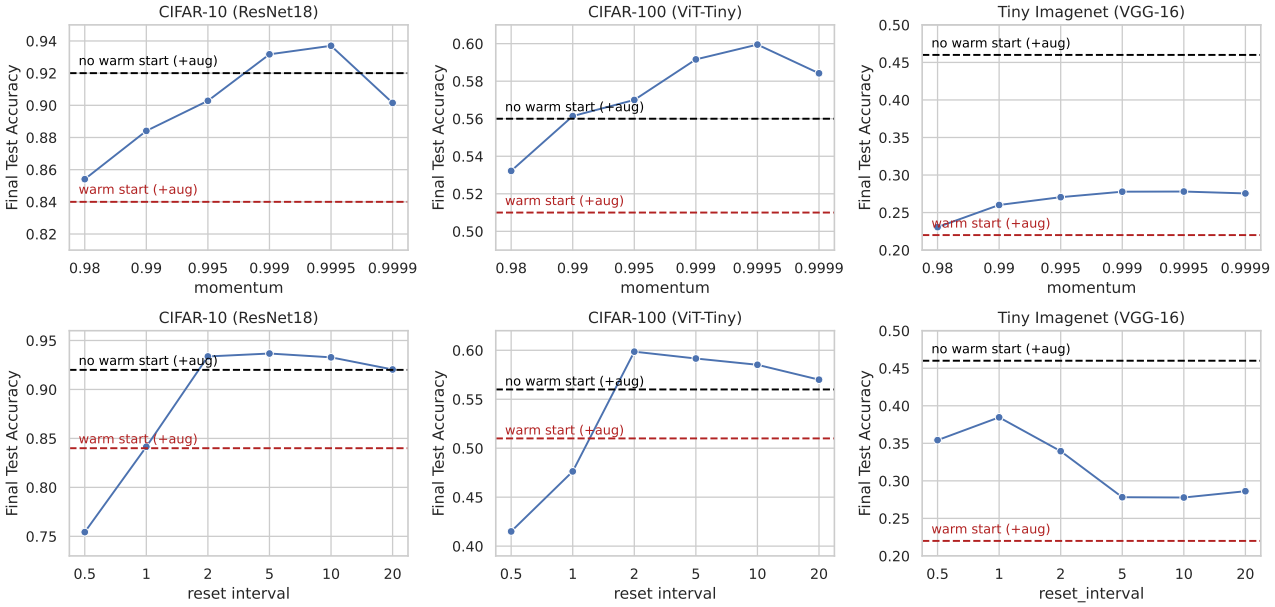

Figure 9. **Ablation Study.** The figure illustrates the robust performance of our method under variations in (a) momentum value and (b) reset interval.

Our main observation is that regardless of the specific values chosen for these hyperparameters, our method consistently delivers stable and improved performance compared to the warm-starting model. This stability and performance enhancement are consistent across various datasets and architectures.

E. Reinforcement Learning Results

Table 8. **Mean trajectory scores on Atari-100k Benchmark.** We report the individual scores on the 26 Atari games. All the other experiments, including DrQ, were conducted based on their original code and averaged over 5 random seeds with a replay ratio of 2.

Method	DrQ						BBF		
S&P	-	✓	-	-	-	-	✓	✓	✓
HR	-	-	-	20k	40k	40k	✓	✓	✓
H&T	-	-	✓	-	-	✓	-	✓	-
SSL	-	-	-	-	-	-	-	-	✓
Alien	841.52	729.1	730.06	838.6	947.22	924.9	1228.7	1180.0	1121.7
Amidar	182.46	122.78	148.82	181.26	166.01	221.67	219.8	293.1	236.6
Assault	559.98	848.25	604.08	765.18	653.28	638.26	1657.3	1844.2	2004.5
Asterix	831.4	572.5	823.5	791.8	813.4	819.3	4699.0	4121.6	3169.7
BankHeist	138.94	24.74	263.56	51.56	86.78	56.32	315.4	712.8	768.8
BattleZone	6404.0	2762.0	8038.0	4040.0	6186.0	4694.0	23752.1	21638.5	23681.4
Boxing	10.6	42.84	14.29	40.98	23.61	34.26	60.8	62.5	77.3
Breakout	11.95	9.8	13.13	18.91	12.55	15.69	245.6	223.3	331.0
CrazyClimber	12031.2	6364.0	13808.4	16158.0	14649.4	18154.2	56440.7	45579.0	60864.5
ChopperCommand	917.2	600.6	673.4	976.0	922.6	816.6	2149.3	3238.0	4251.5
DemonAttack	663.93	906.47	657.41	1013.14	819.02	813.4	11502.5	8248.7	18298.3
Freeway	27.28	21.99	27.29	28.33	29.21	28.95	21.4	21.9	23.1
Frostbite	828.96	561.16	2218.9	1262.06	1923.42	1939.32	2266.2	2838.2	2023.0
Gopher	397.52	602.24	521.76	533.32	577.84	455.76	1758.3	2068.1	1209.4
Hero	7198.58	6293.84	6193.58	7119.41	7677.46	6367.63	4905.8	7448.5	5741.8
Jamesbond	349.0	192.5	317.5	442.6	281.6	358.2	753.1	921.7	1124.6
Kangaroo	3582.0	614.4	2685.2	5534.8	2922.8	2211.4	5584.2	6366.7	5032.0
Krull	3996.46	4697.5	3577.46	4414.6	3897.84	3920.04	7504.2	7695.1	8069.8
KungFuMaster	17319.6	5106.2	12035.4	13909.2	13032.4	19138.8	18166.8	17296.7	16616.8
MsPacman	1251.24	849.66	1424.3	1106.18	1459.34	1062.72	1941.0	1749.3	2217.8
Pong	−7.13	−20.34	−15.01	−10.82	−13.96	−4.11	13.0	12.8	13.6
Qbert	2304.05	763.05	3382.6	3188.45	3620.3	3408.35	3530.5	4237.0	3245.3
RoadRunner	12821.6	6594.8	14674.2	18029.0	16383.0	13995.6	27572.1	29422.4	26419.0
Seaquest	498.12	390.88	531.92	564.08	457.6	550.56	1188.1	1168.8	988.6
PrivateEye	80.0	33.0	85.84	34.62	82.4	100.0	34.8	28.5	39.0
UpNDown	4214.26	4420.98	4269.64	9608.54	6508.24	8451.4	10266.5	7742.0	15122.6
IQM	0.243	0.139	0.287	0.332	0.288	0.328	0.826	0.891	0.940
Median	0.193	0.138	0.260	0.254	0.241	0.329	0.711	0.749	0.755
Mean	0.468	0.458	0.471	0.694	0.532	0.584	1.737	1.719	2.175
OG	0.642	0.728	0.617	0.580	0.607	0.583	0.397	0.372	0.377