Supplementary Material

A. Societal Impacts

This work aims to enrich the knowledge from a given training dataset. While this approach holds significant promise, it also carries both positive and negative societal implications, as detailed below.

Positive Societal Impacts

1. **Enhanced Generalization in AI Systems**
   By enriching knowledge from training datasets, this work can help AI systems generalize better to new tasks and unseen scenarios. This could lead to more robust and adaptable applications in areas such as healthcare diagnostics, autonomous navigation, and disaster response systems, where flexibility and reliability are critical.
2. **Reducing Data Collection Needs**
   With improved knowledge extraction, the reliance on collecting large-scale, diverse datasets could be significantly reduced. This would be especially beneficial in privacy-sensitive fields, such as medical research or financial analysis, where data acquisition is limited due to regulatory and ethical constraints.
3. **Promoting Fairer AI Models**
   By enabling a deeper understanding of training data, this approach could help detect and mitigate biases, resulting in AI systems that are more fair and inclusive. For instance, it could be selectively applied to underrepresented or small-sized classes in long-tailed datasets, improving performance in a more equitable manner.
4. **Improved Resource Efficiency**
   Knowledge enrichment allows for better utilization of existing datasets, potentially reducing the computational resources and energy consumption associated with repeated retraining or fine-tuning of models. This contributes to the sustainability of AI development.

Negative Societal Impacts

1. **Amplification of Hidden Biases**
   If biases or harmful patterns exist within the dataset, enriching this knowledge could unintentionally amplify these issues. For example, an AI system trained on enriched data might reinforce stereotypes or discriminatory practices if the original dataset contained such biases.
2. **Increased Vulnerability to Adversarial Exploits**
   Enriching knowledge from datasets could also increase susceptibility to adversarial attacks, such as backdoor injections or data poisoning. This is because malicious patterns within the training data might be magnified, leading to AI models that are more exploitable in critical applications.
3. **Overfitting to Dataset-Specific Traits**
   Knowledge enrichment might inadvertently cause models to overfit to peculiarities of the training data, reducing their ability to perform well in dynamic or real-world environments. This could undermine the intended generalization benefits and limit the system's practical usability.

B. Design of KNOWN

In this section, we delve into the detailed design aspects and architectural considerations of KNOWN. We discuss the key components, principles, and strategies employed in the design to achieve the desired outcomes.

B.1. Architecture of KNOWN

The architecture of KNOWN, as shown in Figure 1, largely follows the design principles of WNN. Our KNOWN is also composed of two-stream architecture and few fully-connected layers. Thanks to this similarity, we could reuse the pretrianed WNN to this work as initial weights. A key distinction in the approach is removal of normalization. WNN relies on min-max normalization for each sequence and applies denormalization after predicting the weights. This normalization assumes that the weight prediction problem aligns with the behavior of a homogeneous function, characterized by:

$$\lambda f(x) = f(\lambda x), \tag{1}$$

where λ is a scalar, and f represents the function. Under this assumption, sequences such as $[1, 2, 3, 4, 5, 6, 7]$ and $[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7]$ would be treated equivalently after normalization, effectively ignoring their original scales.

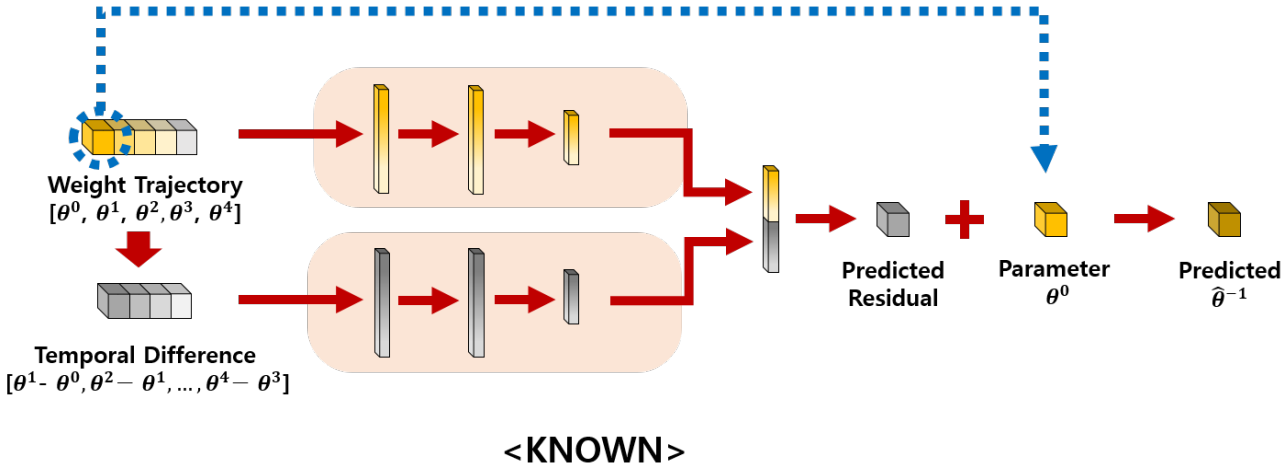


Figure 1. Illustration of the KNOWN architecture. KNOWN is designed to be highly lightweight, consisting solely of MLP layers and containing approximately 9K parameters in total.

However, we propose that the norm of a weight sequence is a critical feature, as it may encode meaningful information about its significance. For example, sequences with smaller norm values could correspond to less significant weights or early training steps near initialization. To better preserve this information, we chose to forgo min-max normalization in the design of KNOWN for past weight prediction.

C. Experimental Details

C.1. Qualitative Validation

To visualize the performance, we conducted an experiment using a small-scale vanilla CNN (approximately 25K parameters) trained on the CIFAR10 dataset. The vanilla CNN consisted of three convolutional layers with batch normalization, followed by global average pooling and a fully connected layer for classification. During training, we recorded its weight trajectory and subsequently collected over 100K data points, each associated with corresponding test accuracies. These points were generated by exhaustively injecting random noise into the saved weight trajectory, followed by a few iterations of fine-tuning.

Additionally, we applied our KNOW prediction to generate enriched weights along the trajectory. To analyze the collected data, we performed Principal Component Analysis (PCA) on all the recorded weights and visualized the results as a surface plot using the first two principal components, with test accuracies represented as the surface height.

On this visualization, we highlighted the original weight trajectory as well as the enriched weights derived from KNOWN, providing a clear comparison of their behavior and performance.

C.2. Image Classification

For this experiment, we pretrained a ResNet18 model on full CIFAR100 training dataset and then sequentially fine-tuned it on progressively smaller subsets. At each pretraining and fine-tuning step, the corresponding weights $[\Theta^0, \Theta^1, \Theta^2, \Theta^3, \Theta^4]$ were saved. This process resulted in a gradual loss of knowledge in the model, exhibiting trends consistent with our earlier landscape visualization.

Next, we employed KNOW prediction methods to predict knowledge-enriched weights from the collected weights. We then evaluated the fine-tuning performance of these enriched weights and the baseline Θ^0 on the CIFAR-10 dataset, which shares a similar domain with CIFAR100 but features fewer and distinct classes.

For all pretraining, fine-tuning, and post-enrichment fine-tuning steps, we used the Adam optimizer with cosine learning rate decay. Additionally, diverse data augmentation techniques, including random flips, translations, rotations, mixup, and cutout, were applied to enhance generalization during training.

We repeated this process starting from $[50\%, 25\%, 12.5\%, 6.25\%]$ of the full training dataset to validate robustness to dataset size. As shown, our method demonstrates effectiveness across all data sizes.

C.3. Generalization within Image Classification

To verify the generality and robustness of our approach, we extended our experiments by training the sequence of weights $\left[\Theta^0, \Theta^1, \Theta^2, \Theta^3, \Theta^4\right]$ using the PVTv2-B0 architecture on the ImageNet dataset with a 224×224 input size. As in previous experiments, we applied the same comprehensive data augmentation strategy, including random flips, translations, rotations, mixup, and cutout to ensure consistency in training. In this setup, we used a sampling rate of 0.33 to generate enriched weights, aiming to validate their effectiveness across various downstream tasks.

To evaluate the versatility of the enriched weights, we applied them to five diverse downstream datasets: CIFAR100, TinyImageNet, Stanford Cars, CUB, and Flowers. For each dataset, the classification head was replaced with a randomly initialized fully connected layer tailored to the target task. We then conducted full fine-tuning for 50 epochs using the Adam optimizer with learning rate scheduling (warmup and cosine decay) and standard data augmentation (crop, rotation, and flip). We used the same training recipe for all methods, differing only in the choice of initial weights, and reported the average performance over ten runs.

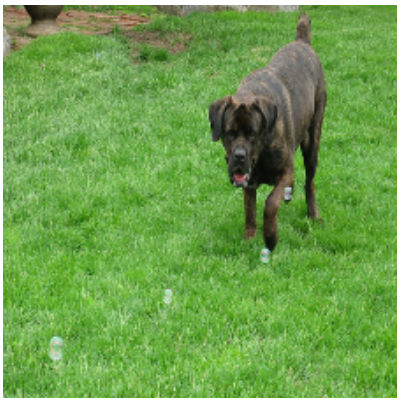
All experiments followed the default train/test splits provided by the TensorFlow Datasets library. We trained models solely on the training sets for 50 epochs without a validation process and reported the final test accuracy. For the Flowers dataset, only 1,020 training images were used, demonstrating the model's robustness under limited-data conditions.

C.4. Generalization to Further Tasks

We also applied the pretrained PVTv2 to Flickr8K (image captioning) and PACS (domain generalization). Each task presented unique challenges and required specific adjustments to the model architecture. Similar to the previous case, we performed 50 epochs of fine-tuning using the Adam optimizer, learning rate scheduling, and data augmentation.

By evaluating Flickr8K for vision-language alignment in image captioning and PACS for domain generalization, we aimed to demonstrate the adaptability and effectiveness of enriched weights in addressing diverse challenges. These experiments highlight the broad applicability of our approach and its potential to generalize across multiple domains and tasks.

Here, some examples are provided for image captioning results of the proposed method and baseline.



Baseline: A dog runs through the grass

KNOWN (×3): A brown dog is running in the grass

Baseline: A man in a black shirt is standing on a rock

KNOWN (×3): A young boy is playing in a bouncy house

Figure 2. Examples of image captioning output.

C.5. Semantic Segmentation

Lastly, we validated our approach on the semantic segmentation task using DeepLabV3+ with two datasets: Pascal VOC and Cityscapes. As preprocessing, all images were resized to 512×512. For pretraining, we trained DeepLabV3+ on Pascal VOC for 200 epochs using Adam optimizer, cosine decay, and data augmentation. Additionally, we progressively downsampled the training dataset and applied continual fine-tuning to induce intentional forgetting.

Following this, KNOW weights were predicted based on the forgetting trajectory using LogFit, Task Vector, and KNOWN. We then transferred the model to the Cityscapes dataset, using the same optimizer for 150 epochs. After the 150th epoch,

mIoU was measured as described in the manuscript table. Following figures illustrate some more examples specifically focusing on missing traffic lights and unstable segmentations.

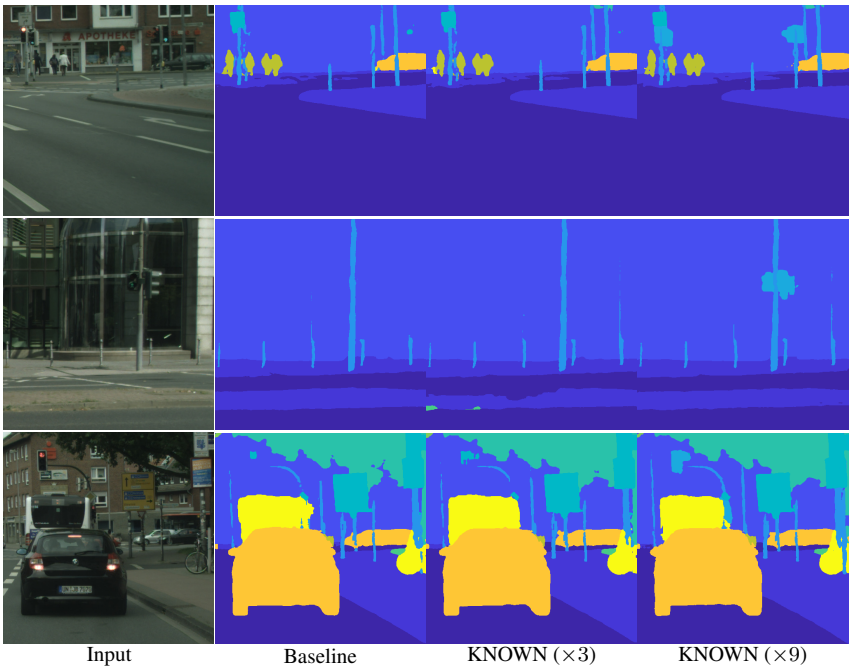


Figure 3. More results for missing traffic light cases.

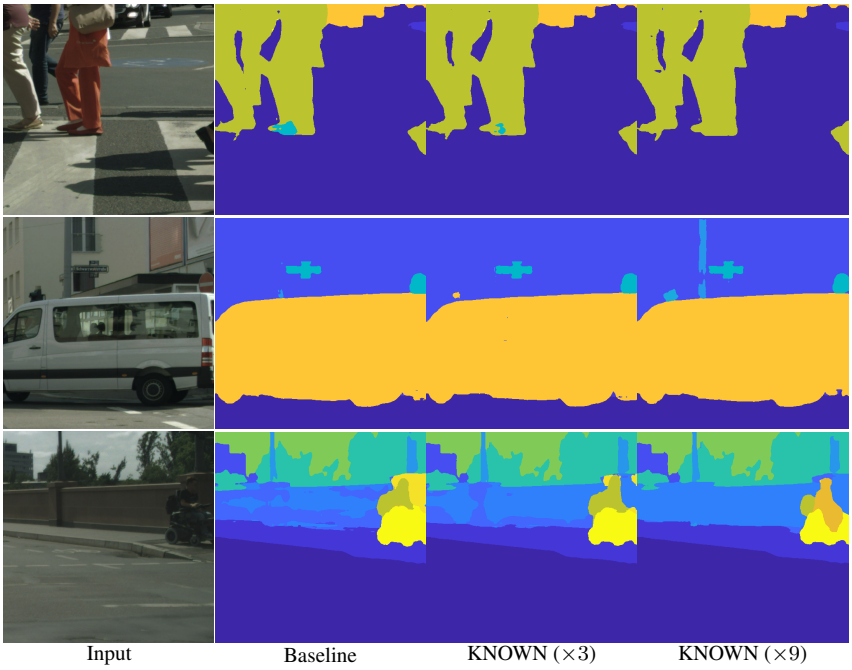


Figure 4. More results for unstable cases.

D. Additional Experiments

D.1. Application to Larger Architectures

We further evaluated the scalability of our approach by applying it to larger architectures, ConvNeXt-Base (88.5M) and ViT-B/16 (86.5M), using CIFAR100 for pretraining and CIFAR10 as the downstream task. As shown in Table below, our predicted weights consistently outperform the original pretrained weights in both models, demonstrating that our method scales effectively to larger architectures.

Methods	ConvNext-Base	ViT-B/16
Naïve Transfer (Baseline)	95.89 ± 0.08	93.07 ± 0.22
KNOWN ($\times 3$)	$\mathbf{96.08 \pm 0.07}$	$\mathbf{93.23 \pm 0.11}$

Table 1. Application results on larger architectures (ConvNext-Base and ViT-B/16) using CIFAR100 for pretraining and CIFAR10 for downstream finetuning. Each value represents the test accuracy of CIFAR10.

D.2. Application to Additional Domains

We fine-tuned both the ImageNet-pretrained PVTv2 and its KNOW ($\times 3$) on DermaMNIST, OrganSMNIST, and BreastMNIST, three representative subsets of the MedMNIST dataset. Despite the substantial domain gap between natural images and medical imagery, KNOW-predicted weights consistently yielded performance improvements, as shown below:

Methods	DermaMNIST	OrganSMNIST	BreastMNIST
Naïve Transfer (Baseline)	89.64 ± 0.44	82.72 ± 0.35	89.62 ± 1.66
LogFit	89.89 ± 0.41	82.87 ± 0.40	89.81 ± 0.81
Task Vector	89.74 ± 0.36	82.75 ± 0.36	90.32 ± 1.02
KNOWN	$\mathbf{90.05 \pm 0.19}$	$\mathbf{83.02 \pm 0.42}$	$\mathbf{91.02 \pm 1.11}$

Table 2. Application of KNOW prediction using ImageNet-pretrained PVTv2 on MedMNIST datasets. Each value represents the test accuracy of MedMNIST datasets.

D.3. Further Extrapolation

The gain from scaling relates to the general tendency in extrapolation: *regression uncertainty increases with the prediction horizon.* Thus, selecting a scaling factor balances gain and prediction uncertainty. This trade-off varies by dataset, so lower factors can sometimes outperform higher ones. We additionally present the $\times 27$ case for PVTv2 pretrained on ImageNet. As commonly observed in regression-based extrapolation (e.g., increasing uncertainty), excessive extrapolation becomes unreliable. Based on our findings, we suggest using $\times 8$ to $\times 9$ for consistent and reliable gains.

	Methods	CIFAR100	TinyImageNet	Car	CUB	Flowers
$\times$**1**	Naïve Transfer (Baseline)	82.03 ± 0.25	76.17 ± 0.33	88.12 ± 0.35	70.49 ± 0.58	87.98 ± 0.38
$\times$**27**	LogFit	81.39 ± 0.21	77.10 ± 0.20	88.83 ± 0.34	70.60 ± 0.57	87.98 ± 0.68
	TaskVector	82.18 ± 0.31	76.97 ± 0.16	88.09 ± 0.50	70.66 ± 0.77	88.03 ± 0.75
	KNOWN	$\mathbf{82.40 \pm 0.21}$	$\mathbf{76.78 \pm 0.17}$	$\mathbf{88.74 \pm 0.25}$	$\mathbf{70.76 \pm 0.36}$	$\mathbf{87.99 \pm 0.38}$

Table 3. Further extrapolation ($\times 27$) results of KNOW prediction using ImageNet as pre-training and various image classification datasets as downstream tasks. Each value denotes the test accuracy on the corresponding downstream dataset. The performance is slightly lower than the $\times 9$ case due to increased regression uncertainty in long-range extrapolation.

E. Additional Discussion

In this section, we provides some additional discussions about out work.

E.1. Why Reversing Forgetting Works: A Conceptual Explanation

We provide an intuitive explanation for why reversing a progressive forgetting trajectory can yield weights with improved generalization, without requiring any additional data. Our method exploits the structured way in which weights evolve when a model is progressively overfitted to smaller subsets of data.

Progressive forgetting as controlled overfitting. When a pre-trained model is fine-tuned on progressively smaller subsets of the original dataset, the process does not just "erase" knowledge in an unstructured manner. Further, it drives the parameters toward *overfitting* with respect to the reduced training distribution. Let W^0 denote the original pre-trained weights and $W^1, \ldots, W^S$ the weights obtained after fine-tuning on nested subsets

$$\mathcal{D}^0 \supset \mathcal{D}^1 \supset \cdots \supset \mathcal{D}^S.$$

As the subset size decreases, the effective data diversity and coverage of the underlying distribution are reduced. The model is repeatedly adapted to a narrower slice of the data, which causes the decision boundary and representations to become increasingly biased toward the chosen subset and to gradually forget patterns supported by the excluded data.

Forgetting induces a structured trajectory in weight space. A key property of this progressive forgetting is that it induces a *structured* trajectory in weight space. Across multiple architectures and datasets, we observe that:

- Consecutive checkpoints W^t and W^{t+1} maintain high similarity.
- The trajectory $\{W^t\}_{t=0}^{S}$ lies on a low-dimensional manifold.
- Changes in weights along the sequence are smooth, and the trajectory can be locally well-approximated by a low-order function of the step index.

In other words, the path from $W^)$ to W^S is not arbitrary; it is a smooth and predictable curve induced by gradually restricting the data used for fine-tuning. This structure makes the trajectory amenable to extrapolation.

Reversing overfitting to recover generality. Conceptually, progressive forgetting moves the model along a continuum: each step toward smaller subsets reinforces behavior that fits the chosen subset well, while gradually discarding behavior that relies on the broader diversity of the original data. As a result, later checkpoints encode a biased, overfitted view of the task.

Reversing this trajectory can be interpreted as reversing the forgetting and overfitting process. By learning how W^t evolves as the subset size shrinks, our meta-model identifies a direction in weight space that corresponds to *moving back* toward configurations that are better aligned with the full, more diverse data distribution. The KNOW prediction then extrapolates beyond W^0 along this structured direction, yielding a weight configuration that effectively behaves as if it had been optimized with access to a larger dataset, even though no additional data are introduced.

Implicit relation to data scaling. From this perspective, the progressive forgetting sequence acts as an *implicit probe* of how the model responds to different levels of data diversity and coverage. Each subset $\mathcal{D}^t$ corresponds to a different effective training regime, and the sequence

$$W^0, W^1, \ldots, W^S$$

summarizes how the corresponding solutions shift as the available data are reduced. Learning to extrapolate along this trajectory is therefore akin to learning a mapping from "smaller-data" solutions back toward solutions that are more consistent with "larger-data" behavior.

We emphasize that this does not imply strict equivalence to training on an actual larger dataset. Rather, the model uses the *observed response* of the weights under controlled data reduction to identify a consistent backward trend in parameter space that naturally recovers and further expands the knowledge embedded in W^0.

E.2. Why the Weight Prediction is Beneficial to Finetuning

As illustrated in our landscape visualization, there are two primary reasons that explain the observed behavior of the predicted point in the parameter space. First, the predicted point is positioned in close proximity to the ground truth enriched weights. This closeness plays a crucial role during finetuning, as it facilitates rapid convergence during the optimization process. By starting from a position near the optimal weights, the model requires fewer updates to reach a well-performing state, which not only speeds up training but also contributes to improved generalization. This generalization arises because the enriched weights effectively encode task-relevant information, reducing the risk of overfitting to specific patterns in the dataset.

Second, the predicted point exhibits a steeper gradient compared to the mode-connected region, which is depicted in yellow in the landscape visualization. A steeper gradient in this context implies that the optimization landscape is more responsive to changes in the parameters. This characteristic is beneficial during fine-tuning, as it enables the model to quickly adapt to new tasks or domains by effectively navigating the parameter space. While the initial performance of the model (e.g., test accuracy) may appear suboptimal during the early stages of fine-tuning, the steep gradient ensures that further optimization converges efficiently to a high-performing solution.

Third, the steeper gradient associated with the predicted point provides an advantage in momentum-based optimization. In contrast, mode-connected regions, with their smaller and more consistent gradients, increase the risk of the optimization process becoming trapped in local minima or saddle points. Momentum techniques, such as heavy ball momentum, Nesterov, and Adam, are widely used in modern DNN training to overcome such issues by building inertia across iterations. The steep gradients at the predicted point help initialize this momentum, guiding the optimization trajectory effectively from high-gradient to low-gradient regions. This dynamic reduces the likelihood of stagnation in flat or poorly conditioned areas of the loss landscape, enabling more efficient convergence. As a result, the predicted points not only facilitate rapid convergence but also provide robustness against local minima, aligning well with the characteristics of momentum-based optimizers.

The interplay between these two factors—the proximity to enriched weights and the steep gradient—provides a compelling explanation for the ability to achieve both rapid convergence and robust generalization. This insight underscores the importance of predicting weights that are not only close to the enriched ground truth but also situated in regions of the parameter space that enable efficient learning dynamics.

E.3. Coordinate-wise Prediction & Scalability with a Fixed KNOWN Size

Our method adopts a **coordinate-wise independent prediction** strategy. Given a target model with 1B parameters and S progressive forgetting steps, this produces 1B $\times$ S weight values along the trajectory. After decomposing the trajectory into a $(1 \times S)$ vector for each parameter, **KNOWN** predicts each coordinate independently using only this vector, repeating the process across all 1B parameters.

Because each parameter is predicted independently, the size of **KNOWN** remains fixed ($\sim$9K parameters), enabling it to scale to any architecture, including large-scale models, without additional complexity. Moreover, our approach does not require loading the full 1B $\times$ S trajectory into GPU memory. Only the small **KNOWN** model and a single $(1 \times S)$ vector are needed at a time, regardless of model size.

This lightweight computation and memory footprint make the approach inherently scalable and practically efficient.

E.4. Memory Cost

We provide a detailed analysis of time cost in the main manuscript. Here, we additionally discuss the memory burden of our method. Our approach stores the weight history for the five most recent steps of progressive forgetting, which introduces additional memory proportional to five copies of the model weights, along with a small amount of memory for running KNOWN. Note that only the model parameters need to be stored, not any intermediate features. Since KNOWN is extremely lightweight (9K parameters) and predicts each parameter independently, its computation can be executed entirely on the CPU. Following the strategy used in NiNo [1], the weight history can also be placed on the CPU, which avoids any impact on GPU VRAM usage and does not interfere with the original training process. Consequently, the memory overhead introduced by our method remains modest and practically feasible.

E.5. Requirement for Sampling Rate Selection

To ensure consistency throughout the sequential forgetting process, we used a fixed batch size. This fixed batch size was intended to maintain consistency across different stages of the process, providing a more uniform basis for analyzing the impact of sequential data reduction. However, using a fixed batch size introduces a limitation on the selection of the sampling rate r. The sampling rate r determines the proportion of data retained in each subset as we progressively reduce the dataset size. Given the size of the original training data $|D^0|$ and the requirement to sample sequentially $S-1$ times, the sampling rate r must be carefully chosen to avoid overly reducing the data available in later steps. Specifically, the sampling rate r must satisfy the condition:

$$r < \left(\frac{|D^0|}{B}\right)^{-\frac{1}{S-1}},$$

where B represents the batch size. This constraint ensures that, after multiple rounds of sampling, there remains a sufficient amount of data for meaningful learning. If r is set too high, the data size will diminish too quickly, making the later stages of the sequential forgetting process impractical.

As mentioned earlier, the gain from scaling relates to the general tendency in extrapolation: *regression uncertainty increases with the prediction horizon.* Then, this sampling rate is the inversion of scaling factor, so selecting an adequate sampling rate balances gain and prediction uncertainty.

E.6. Analysis about Overfitting

Our method intentionally induces controlled overfitting through progressive forgetting. When we pre-trained ResNet on CIFAR100 under this setup, the results were as follows:

Dataset	Selected Subset During Progressive Forgettings				
	100%	50%	25%	12.5%	6.25%
Training (Remaining Subset)	99.90 ± 0.16	99.97 ± 0.01	99.98 ± 0.02	99.98 ± 0.01	99.99 ± 0.01
Training (Forgotten)	N/A	95.87 ± 1.60	92.37 ± 1.13	88.68 ± 1.35	83.00 ± 1.99
Test	71.95 ± 0.49	70.36 ± 0.53	68.40 ± 0.44	65.74 ± 0.41	62.34 ± 0.80

Table 4. Overfitting and forgetting tendencies during progressive fine-tuning on sequential subsets of CIFAR100. Each value denotes the accuracy on the remaining subset, the out-of-subset data, and the full test set. The results clearly show strong overfitting on the remaining subset while simultaneously exhibiting substantial forgetting on the out-of-subset data.

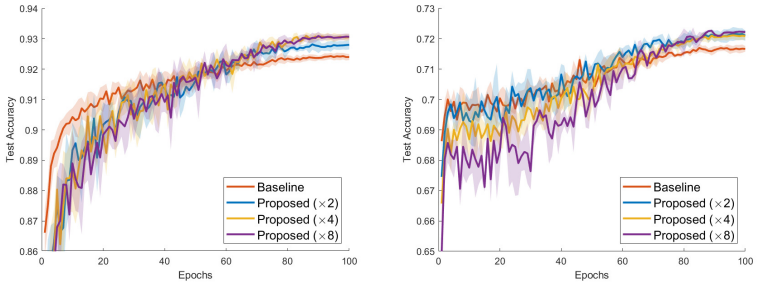
The substantial gap between near-perfect training accuracy and the steadily decreasing test accuracy clearly indicates overfitting induced by progressive forgetting. During this process, fine-tuning causes the model to specialize on the selected subset while forgetting the out-of-subset data, effectively amplifying both overfitting and specialization. Our method later reverses this overfitting trajectory, recovering more generalized representations and effectively mimicking the behavior of models trained on larger datasets.

E.7. Finetuning Tendency

To show the finetuning tendency, we used CIFAR100 pretraining with a much smaller sampling rate $r = 0.3$. It was the minimum value (with 0.1 scale) that could be jointly used with a batch size of 128. If a lower value were used, the data size would become smaller than the batch size. Then, we applied the predicted weights to two downstream tasks: CIFAR10 and CIFAR100. Note that CIFAR100 was used for pretraining and reused for finetuning. Fig. 5 depicts the changes in test accuracies of the two downstream datasets during the finetuning process. There are three important observations.

First, the KNOW predicted by our KNOWN underperforms the baseline during early finetuning because the predicted weights are slightly outside the convergence region. However, it surpasses the baseline in the later phase. This phenomenon can be explained by our landscape visualization. The predicted weights are positioned in a region with lower test accuracy initially but are closer to the ground truth Θ^{-1}. Therefore, the proposed weights exhibit lower performance in the early phase but ultimately achieve better convergence.

Second, iterative forecasting gradually enhances convergence performance, consistent with the results of our manuscript. Lastly, the proposed approach is applicable to the pretrained dataset itself. This indicates that our method successfully approximates more knowledge, and the enriched knowledge can further improve training even on the previously learned dataset.



CIFAR10 CIFAR100

Figure 5. Tendency of test accuracy during finetuning process.

Also, this tendency is observed even in further tasks, such as semantic segmentation.

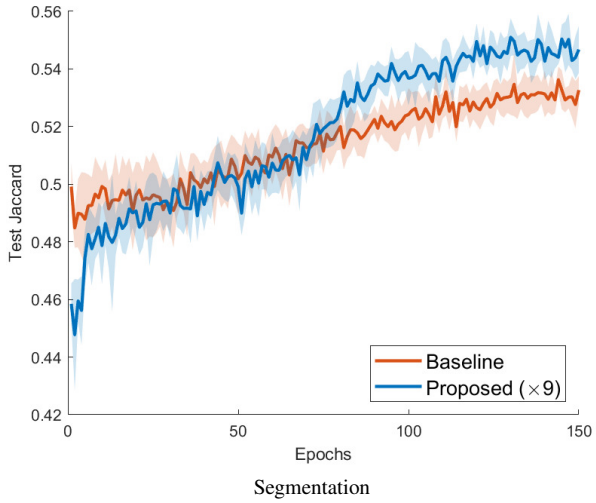


Figure 6. Tendency in test Jaccard Coefficient for semantic segmentation.

E.8. Incremental Learning vs. Forgetting

In practice, forecasting in an incremental learning setting may appear more suitable than reversing catastrophic forgetting. Specifically, this approach involves the training on sequentially increased datasets, such that each learned weight configuration is used as the initialization for the next. For instance, after training Θ^{S-1} on D^{S-1} from random initialization, these weights would serve as the starting point to train Θ^{S-2} on D^{S-2}. This process continues until Θ^0 is obtained, constructing a trajectory of weights that progressively encapsulate knowledge from smaller subsets of data. From this trajectory, one could then predict $\hat{\Theta}^{-1}$.

We initially considered this approach due to its conceptual alignment with forecasting weight evolution in a sequential learning framework. However, this method presents a significant drawback: since Θ^{S-1} is trained on the smallest subset of data, it inherently lacks comprehensive knowledge, serving as a suboptimal initialization for the next iteration. This limitation compounds with each step, leading to degraded performance in the learned weights. As a result, although weight prediction remains feasible within this framework, the reliance on Θ^{S-1} trained on smaller datasets makes the predicted weights inherently less effective than those trained on a full dataset.

Consequently, this limitation motivated us to adopt an alternative approach: reversing catastrophic forgetting. By starting with weights trained on the full dataset and sequentially removing subsets of information, we can preserve a richer representation of the original data, resulting in weight predictions that retain enhanced knowledge. This reverse approach not only mitigates the issues associated with reduced data but also allows us to construct more robust initializations for downstream tasks, ultimately leading to improved model performance and adaptability.

E.9. Applicability to Finetuning Rather Than Pretraining

As shown in our landscape visualization, the predicted KNOW is positioned closer to weights containing more knowledge. This suggests that KNOW prediction may also be applicable to the fine-tuning phase.

During fine-tuning, intentional forgetting can be induced through progressive downsampling. By reversing this trajectory, well-positioned weights can be obtained, potentially leading to improved performance when retraining from that point. In this paper, we focused on the applicability of KNOW prediction in the pretraining phase. We leave further exploration of its role in fine-tuning for future work.

References

[1] Boris Knyazev, Abhinav Moudgil, Guillaume Lajoie, Eugene Belilovsky, and Simon Lacoste-Julien. Accelerating training with neuron interaction and nowcasting networks. In *ICLR*, 2025. 7