Learning from Oblivion: Predicting Knowledge-Overflowed Weights via Retrodiction of Forgetting

Jinhyeok Jang, Jaehong Kim, Jung Uk Kim

Highlight

CVPR JUNE 3-7, 2026 DENVER COLORADO

Motivation

- **Our work is motivated by the following fundamental questions:**

> *(i) What defines a better pre-trained weight?*
> *(ii) If practical constraints exist, how can we obtain the better weights?*

- **For the first question, we draw inspiration from "Scaling Law."**
  - **More training data generally leads to better model weights.**
  - **However, obtaining additional training data is challenging.**

Hypothesis

- **Our Hypothesis**

Weights trained as if on more training data can be predicted by reversing the forgetting trajectory.

Preliminary

Consider a dataset D^0 and a subset $D^1 \subset D^0$. Let Θ^0 be the weights obtained from pre-training on D^0. When fine-tuning Θ^0 on D^1, the resulting weights Θ^1 exhibit a loss of knowledge about $D^0 \setminus D^1$ due to forgetting.

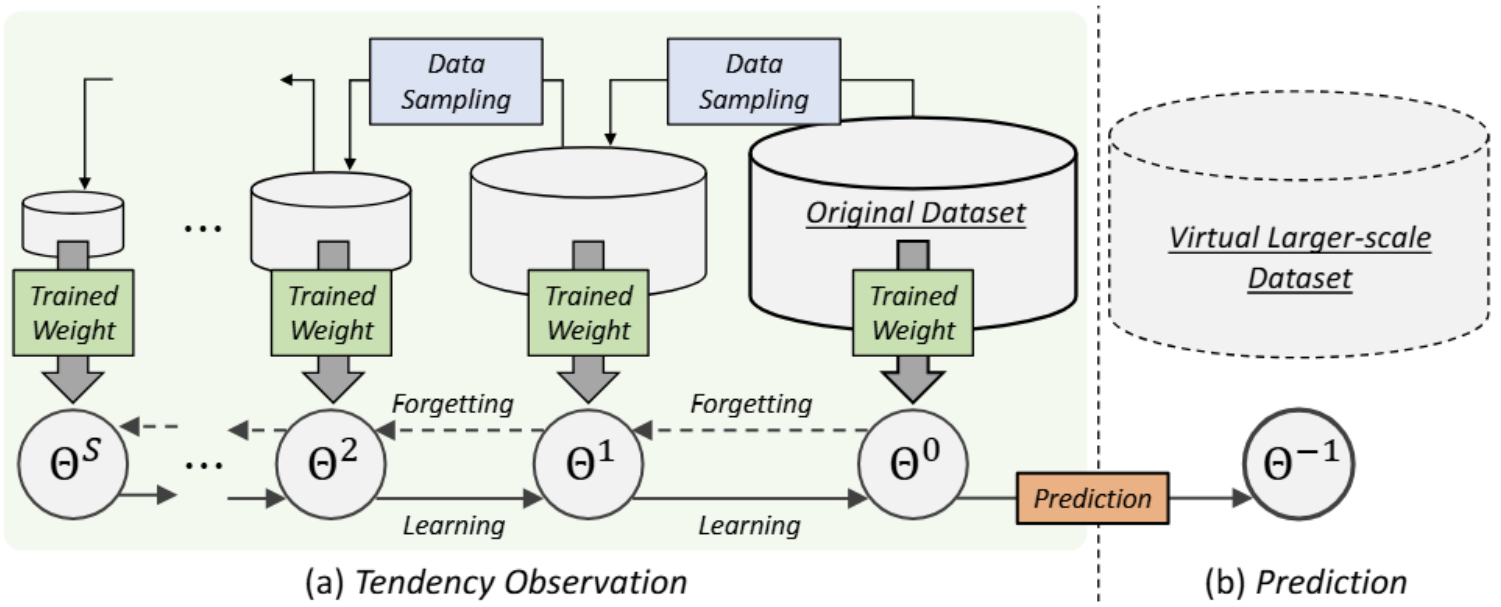
Hypothesis

Given Θ^0 trained on D^0, there exist ideal weights Θ^{-1} satisfying the following conditions:

1. Θ^{-1} is trained on D^{-1}, where $D^0 \subset D^{-1}$.
2. Fine-tuning Θ^{-1} on D^0 results in Θ^0.

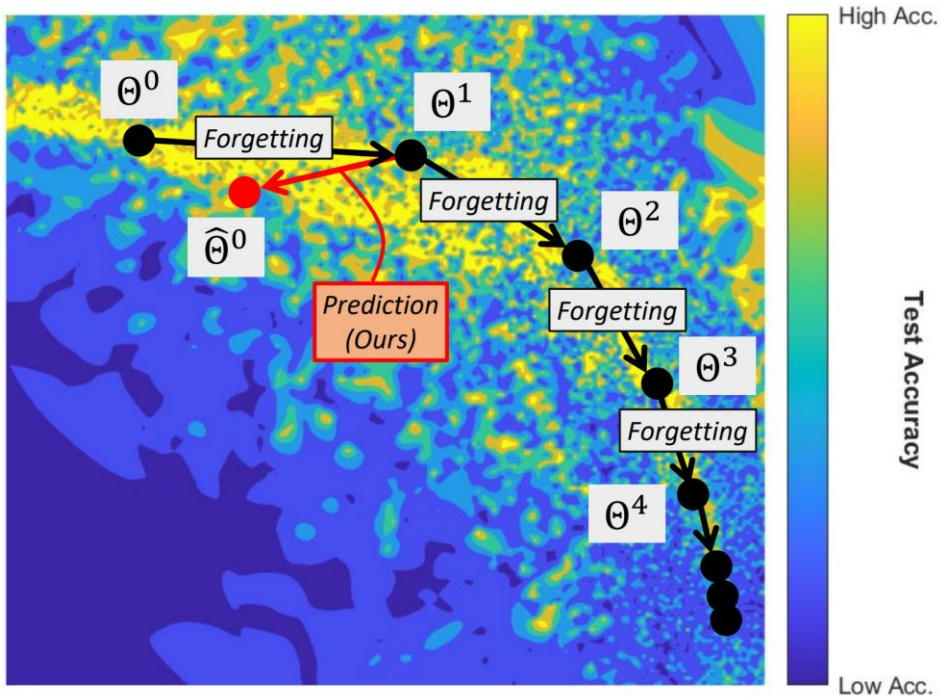
Extension to Progressive Forgetting

Given a sequence of datasets $D^S \subset D^{S-1} \subset \cdots \subset D^1 \subset D^0$ constructed with a sampling ratio $r \in [0, 1]$, we obtain a corresponding sequence of weights $[\Theta^S, \Theta^{S-1}, \ldots, \Theta^0]$, where each Θ^i is obtained by fine-tuning Θ^{i-1} on D^i.



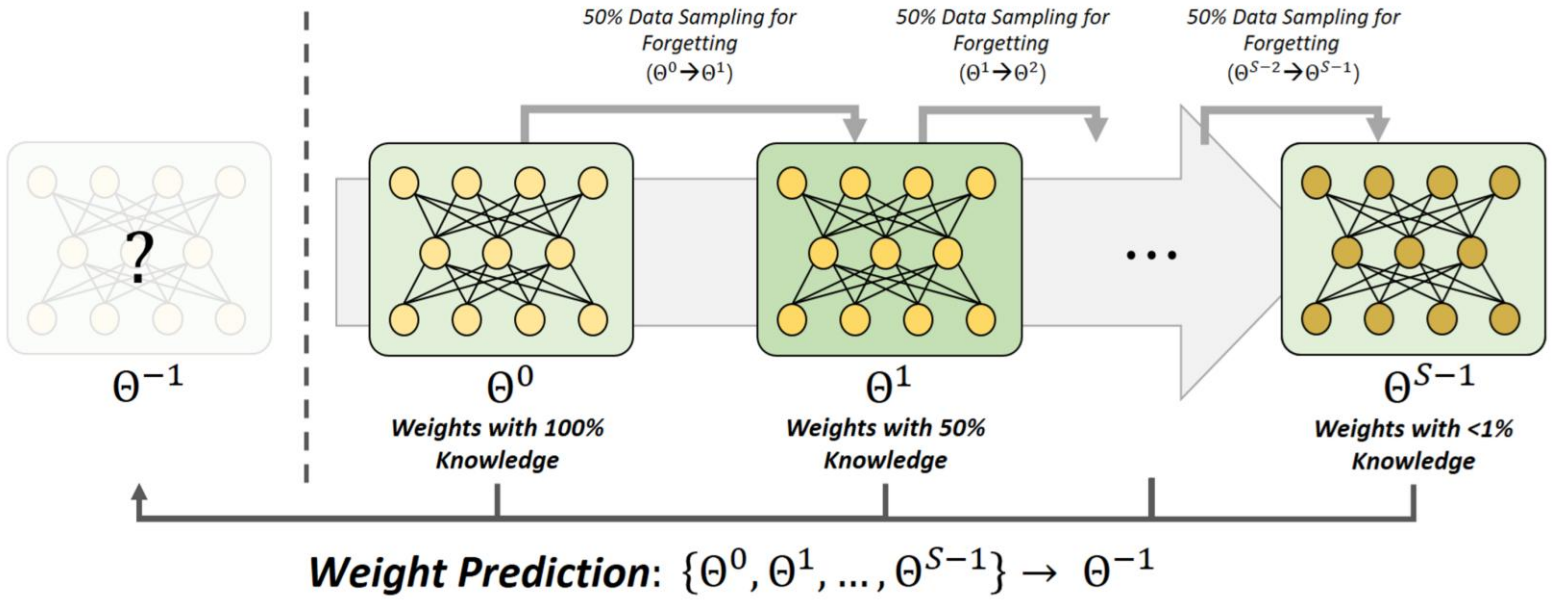
Observation

- **Visualizing the Forgetting Trajectory as a Loss Landscape**
  - **Dataset:** CIFAR10
  - **Method:**
    1) **Base trajectory collection**
       Fine-tune the model on progressively downsampled datasets.
    2) **Neighborhood exploration**
       Perform exhaustive fine-tuning with random noise injection.
    3) **Trajectory visualization**
       Project the collected weight trajectories onto a PCA space.
- **The forgetting trajectory exhibits a pattern in the weight space.**



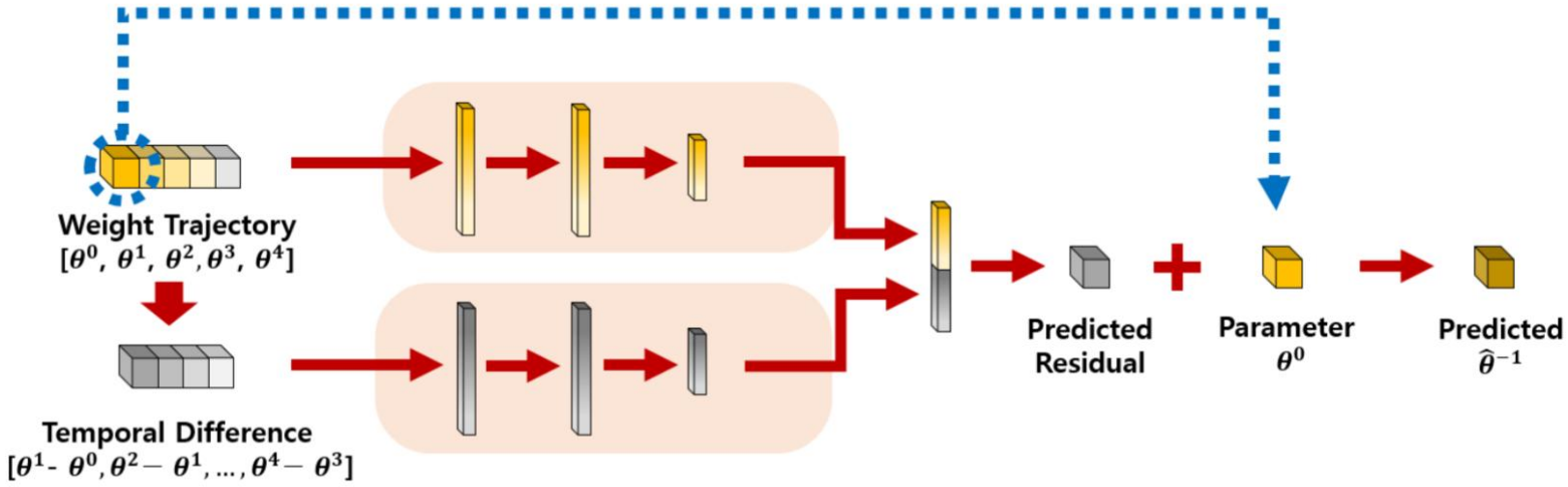
Knowledge-Overflowed Weight Prediction

- **We establish three steps:**
  1) **Pretraining on the full dataset**
  2) **Progressive fine-tuning on sequential subsets *(intentional forgetting)***
  3) **Predicting weights trained as if on more data by reversing the forgetting trajectory and extrapolation**



KNOW Nowcaster (KNOWN)

- **We meta-train a dedicated hypernetwork, called KNOWN.**
  1) **Trajectory Collection:** Collect weight trajectories across diverse settings(e.g., architectures, datasets, and training recipes)
  2) **KNOWN Meta-Training:** Train KNOWN as a lightweight meta-learned predictor
  3) **After meta-training, KNOWN requires no additional data collection or meta-learning.**



Experiments

- **KNOW prediction is effective across diverse settings.**
  - **Weight-space prediction methods (e.g., curve fitting and task arithmetic) can improve performance.**
  - **Our meta-learned KNOWN provides the largest performance gain.**

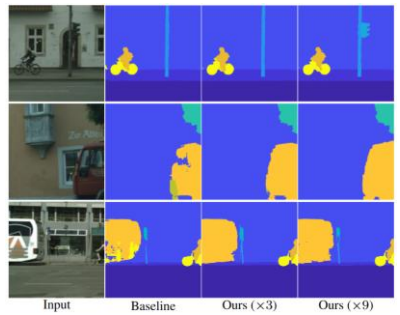
Pred. ($\times n$)	Methods		Amount of pre-training Data (%) 100%	50%	25%	12.5%	6.25%
$\times 1$	Naïve Transfer (Baseline)		92.40 ± 0.11	92.08 ± 0.18	91.90 ± 0.24	91.49 ± 0.22	91.51 ± 0.12
	Incremental Pretraining (Baseline)		92.29 ± 0.10	91.32 ± 0.14	90.79 ± 0.48	90.07 ± 0.10	89.51 ± 0.08
$\times 2$	KNOW[§]	LinearFit	92.70 ± 0.16	92.28 ± 0.16	91.89 ± 0.24	91.42 ± 0.15	91.37 ± 0.16
		LogFit	92.83 ± 0.25	92.41 ± 0.07	91.89 ± 0.28	91.39 ± 0.17	91.33 ± 0.24
		ExpFit	79.58 ± 0.41	86.79 ± 0.23	89.26 ± 0.04	89.85 ± 0.08	90.60 ± 0.16
		TaskVector [24]	92.69 ± 0.09	92.39 ± 0.12	92.22 ± 0.25	91.45 ± 0.05	91.46 ± 0.08
		ConsensusTA[†] [55]	92.69 ± 0.09	92.39 ± 0.12	92.22 ± 0.25	91.45 ± 0.05	91.46 ± 0.08
		MagMax[‡] [44]	92.69 ± 0.09	92.39 ± 0.12	92.22 ± 0.25	91.45 ± 0.05	91.46 ± 0.08
		TSV [11]	92.82 ± 0.14	92.36 ± 0.16	92.19 ± 0.11	91.64 ± 0.19	91.74 ± 0.25
		KNOWN	$\mathbf{93.00 \pm 0.11}$	$\mathbf{92.58 \pm 0.14}$	$\mathbf{92.29 \pm 0.04}$	$\mathbf{92.11 \pm 0.10}$	$\mathbf{91.90 \pm 0.13}$
$\times 4$	KNOW[§]	LinearFit	92.40 ± 0.08	91.99 ± 0.22	91.63 ± 0.14	90.52 ± 0.12	90.64 ± 0.09
		LogFit	92.82 ± 0.10	92.33 ± 0.13	91.87 ± 0.13	91.05 ± 0.15	91.08 ± 0.23
		ExpFit	72.70 ± 0.64	80.75 ± 0.55	85.16 ± 0.04	87.46 ± 0.27	88.80 ± 0.29
		TaskVector [24]	92.70 ± 0.09	92.48 ± 0.10	92.19 ± 0.12	91.45 ± 0.14	91.36 ± 0.18
		ConsensusTA[‡] [55]	92.70 ± 0.09	92.48 ± 0.10	92.19 ± 0.12	91.45 ± 0.14	91.36 ± 0.18
		MagMax [44]	92.44 ± 0.28	92.19 ± 0.22	92.12 ± 0.19	91.29 ± 0.16	91.15 ± 0.14
		TSV [11]	92.74 ± 0.13	92.40 ± 0.16	92.17 ± 0.18	91.64 ± 0.20	91.63 ± 0.16
		KNOWN	$\mathbf{93.27 \pm 0.09}$	$\mathbf{92.62 \pm 0.25}$	$\mathbf{92.88 \pm 0.11}$	$\mathbf{92.40 \pm 0.06}$	$\mathbf{91.98 \pm 0.14}$
$\times 8$	KNOW[§]	LinearFit	92.13 ± 0.10	91.71 ± 0.22	90.77 ± 0.18	90.15 ± 0.40	90.42 ± 0.44
		LogFit	92.65 ± 0.10	91.93 ± 0.25	91.59 ± 0.05	90.49 ± 0.25	90.68 ± 0.24
		ExpFit	66.54 ± 0.29	73.89 ± 0.91	81.05 ± 0.14	84.34 ± 0.56	85.16 ± 0.45
		TaskVector [24]	92.65 ± 0.13	92.24 ± 0.15	92.03 ± 0.15	91.13 ± 0.21	91.28 ± 0.34
		ConsensusTA [55]	92.43 ± 0.14	92.02 ± 0.12	91.93 ± 0.17	91.11 ± 0.29	90.63 ± 0.32
		MagMax [44]	92.39 ± 0.20	92.16 ± 0.10	92.07 ± 0.09	91.43 ± 0.20	90.85 ± 0.33
		TSV [11]	92.72 ± 0.11	92.37 ± 0.15	92.25 ± 0.15	91.51 ± 0.17	91.48 ± 0.15
		KNOWN	$\mathbf{93.55 \pm 0.05}$	$\mathbf{93.11 \pm 0.19}$	$\mathbf{92.92 \pm 0.15}$	$\mathbf{92.22 \pm 0.37}$	$\mathbf{92.07 \pm 0.18}$

<CIFAR100→CIFAR10>

Pred. ($\times n$)	Methods		Downstream Dataset $\mathcal{D}$ (ImageNet (Pre-train) $\rightarrow \mathcal{D}$) 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
$\times 3$	KNOW	LogFit	81.73 ± 0.37 (-0.30)	77.32 ± 0.14 $(+1.15)$	88.49 ± 0.68 $(+0.37)$	70.45 ± 0.71 (-0.04)	88.24 ± 0.20 $(+0.26)$
		TaskVector [24]	82.15 ± 0.29 $(+0.12)$	77.49 ± 0.21 $(+1.32)$	88.31 ± 0.24 $(+0.19)$	71.00 ± 0.25 $(+0.51)$	88.18 ± 0.42 $(+0.20)$
		TSV [11]	82.14 ± 0.46 $(+0.11)$	76.06 ± 0.17 (-0.11)	$\mathbf{88.78 \pm 0.12}$ $(+0.80)$	70.84 ± 0.48 $(+0.35)$	86.37 ± 0.37 (-1.61)
		KNOWN	$\mathbf{82.46 \pm 0.20}$ $(+0.43)$	$\mathbf{77.53 \pm 0.11}$ $(+1.36)$	88.57 ± 0.18 $(+0.45)$	$\mathbf{71.18 \pm 0.45}$ $(+0.69)$	$\mathbf{88.65 \pm 0.69}$ $(+0.67)$
$\times 9$	KNOW	LogFit	81.75 ± 0.35 (-0.28)	77.29 ± 0.14 $(+1.12)$	88.42 ± 0.30 $(+0.30)$	71.09 ± 0.65 $(+0.60)$	88.43 ± 0.57 $(+0.45)$
		TaskVector [24]	82.12 ± 0.45 $(+0.09)$	77.29 ± 0.23 $(+1.12)$	88.18 ± 0.17 $(+0.06)$	70.38 ± 0.34 (-0.11)	88.37 ± 0.43 $(+0.39)$
		MagMax [44]	82.27 ± 0.24 $(+0.24)$	75.98 ± 0.18 (-0.19)	88.61 ± 0.31 $(+0.49)$	70.95 ± 0.39 $(+0.46)$	86.51 ± 0.76 (-1.47)
		TSV [11]	82.31 ± 0.20 $(+0.28)$	76.18 ± 0.27 $(+0.01)$	$\mathbf{88.85 \pm 0.09}$ $(+0.73)$	70.79 ± 0.32 $(+0.30)$	86.47 ± 0.56 (-1.51)
		KNOWN	$\mathbf{82.33 \pm 0.15}$ $(+0.30)$	$\mathbf{77.58 \pm 0.12}$ $(+1.41)$	$\mathbf{88.85 \pm 0.23}$ $(+0.73)$	$\mathbf{71.30 \pm 0.28}$ $(+0.81)$	$\mathbf{88.53 \pm 0.27}$ $(+0.55)$

<ImageNet→CIFAR100/TinyImageNet/Cars196/CUB/Flowers>

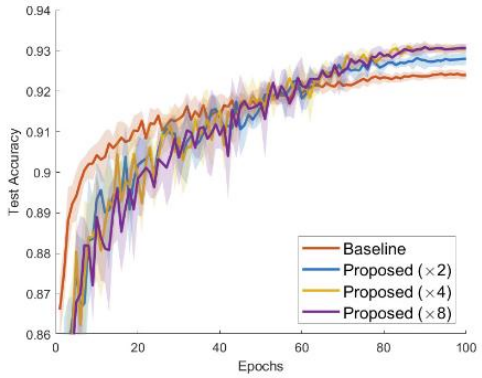
Pred.($\times n$)	Methods		mIoU (%)
$\times 1$	Naïve Transfer (Baseline)		68.52 ± 1.34
$\times 3$	KNOW	LogFit	68.79 ± 0.68 $(+0.27)$
		TaskVector [24]	68.83 ± 1.71 $(+0.31)$
		TSV [11]	68.64 ± 0.71 $(+0.12)$
		KNOWN	$\mathbf{69.00 \pm 1.04}$ $(+0.48)$
$\times 9$	KNOW	LogFit	69.16 ± 1.95 $(+0.64)$
		TaskVector [24]	67.99 ± 0.96 (-0.53)
		MagMax [44]	70.04 ± 0.81 $(+1.52)$
		TSV [11]	68.98 ± 1.06 $(+0.46)$
		KNOWN	$\mathbf{71.22 \pm 0.82}$ $(+2.70)$



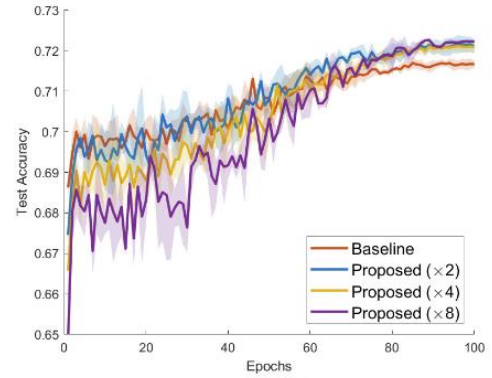
<ImageNet→CityScapes>

Ablation

- KNOWN progresses slowly in the early phase due to prediction error, but reaches higher final performance.
- Longer history improves convergence.
  - However, longer history requires more training data.
- KNOWN achieves a good balance between data efficiency and training efficiency.



CIFAR10



CIFAR100

	Pred.$(\times 2)$	**Pred.**$(\times 4)$	**Pred.**$(\times 8)$
$S = 2$ (Task Vector)	92.69 ± 0.09	92.70 ± 0.09	92.65 ± 0.13
$S = 3$ (KNOWN)	$\mathbf{93.01 \pm 0.16}$	93.04 ± 0.06	92.72 ± 0.10
$S = 4$ (KNOWN)	92.97 ± 0.16	$\underline{93.10 \pm 0.13}$	$\underline{92.89 \pm 0.16}$
$S = 5$ (KNOWN)	$\underline{93.00 \pm 0.11}$	$\mathbf{93.27 \pm 0.09}$	$\mathbf{93.55 \pm 0.05}$

Methods	**Dataset Amount (%)**	**Training Time (s)**	**Inference Time (s)**	**Accuracy (%)**
Naïve Transfer	50	3,800	N/A	92.08
(Baseline)	100	7,600	N/A	92.40
	KNOWN (×2)	50	7,372	3.01	92.58
KNOW	KNOWN (×4)	50	7,372	6.02	<u>92.62</u>
	KNOWN (×8)	50	7,372	9.03	**93.11**

Summary

- We introduce **KNOW prediction**, which predicts weights trained as if on more data by reversing forgetting trajectories.
- We observe that progressive forgetting exhibits a pattern in weight space.
- We propose **KNOWN**, a lightweight hypernetwork for efficient weight prediction across diverse settings.
- Our findings show that forgetting dynamics can serve as a useful signal for data- and training-efficient model improvement.

Day3 Afternoon
Contact: jangjh6297@gmail.com

Thanks

Learning from Oblivion:
Predicting Knowledge-Overflowed Weights via Retrodiction of Forgetting