Learning to Rewind via Iterative Prediction of Past Weights for Unlearning

Jinhyeok Jang[1,2], Jaehong Kim[1], Chan−Hyun Youn[2]

[1] ETRI [2] KAIST

Introduction

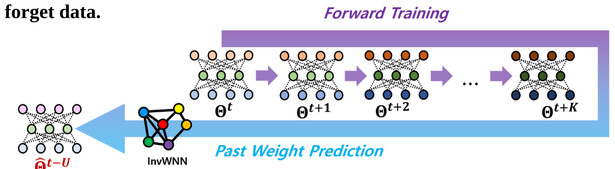
- There have been issues of Privacy/Copyright Infringement
- Machine Unlearning can be a post-solution of theses issues
- However, there are several limitations in prior works
  - Requiring the full training dataset (i.e., Fisher, SCRUB)
  - Resulting in Confusion (i.e., Random Label, Boundary Unlearning)
  - Inaccurate Rewinding (i.e., Gradient Ascent, Task Vector)

Related Works: Weight Prediction

- There have been some studies about weight-space learning, and there exist some DNN-based weight prediction works for acceleration of training process
  - Introspection, WNN, NiNo

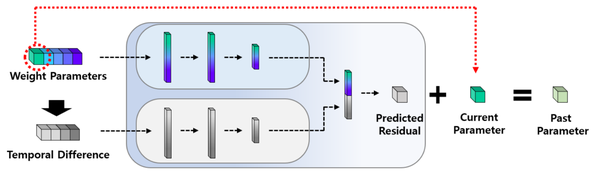
Proposed Strategy

- We repurpose weight prediction—from training acceleration to machine unlearning.
- We introduce a novel concept of past weight prediction using a pretrained DNN sub-module.
- Our method predicts past weights by leveraging the forward training trajectory on forget data.



Method: InvWNN

- We adopted the same architecture as the prior work (WNN), modifying only the layer widths.
- InvWNN was trained to predict weight from three epochs earlier using a forward trajectories.
- InvWNN performs predictions in a coordinate-wise (i.e., element-wise) manner.



Experiments: Standard Setting

- Setting
  - Pretrained Model: ResNet18 trained on Full CIFAR10
  - Removing the knowledge of randomly selected 50% training data
- Metrics
  - Remaining Accuracy (RA), Unlearning Accuracy (UA), Test Accuracy (TA), and Membership Inference Attack (MIA)

Table 1: Results of five trials for unlearning randomly selected 50% of CIFAR10. Note that better performance corresponds to a smaller gap with the retrained model.

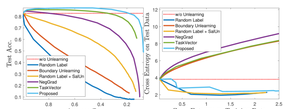
Method	Termination Condition: UA > 8.11%			
	RA (%)	UA (%)	TA (%)	MIA (%)
Pretrained	99.96±0.02	0.02±0.01	94.38±0.10	9.75±1.61
Retrain	99.92±0.02	8.11±0.19	91.45±0.32	29.07±0.42
Random Label	91.03±0.55	8.99±0.53	83.63±0.44	33.19±2.46
Boundary	91.24±0.21	8.42±0.16	83.72±0.10	28.51±3.97
SalUn	91.33±0.45	8.58±0.39	84.45±0.35	32.62±3.17
NegGrad	91.26±0.31	8.59±0.11	85.01±0.95	**28.56±9.60**
Task Vector	92.03±0.72	8.54±0.54	83.29±0.82	30.86±6.65
Proposed	**92.24±0.71**	8.31±0.24	**85.04±1.10**	31.94±4.56

Experiments: Side Effect Removal

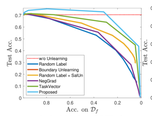
- For some scenarios, pretrained models implicitly learn some side effects such as:
  - Confusion caused by Label Noise
  - Intended misfunction due to Backdoor Attacks
  - Worse performance because of Overfitting
- Using ResNet18 and CIFAR10, we intentionally added and tried to remove these side effects via unlearning

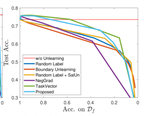
Experiments: Generalization

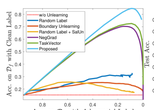
- We applied the label noise scenario to
  - Additional architectures: PVTv2 and ConvNext
  - Additional datasets: CIFAR100 and TinyImageNets
- Last, we applied our work to generative AI with diffusion models
  - Removing the knowledge about “4” from a diffusion model pretrained on MNIST

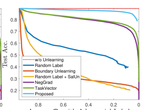

(a) Label Noise

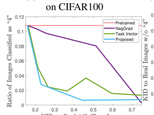
(b) Overfitting

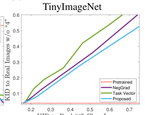

(a) Label Noise: PVTv2 on CIFAR100


(b) Label Noise: ConvNext on TinyImageNet


(c) Backdoor: Damage recovery

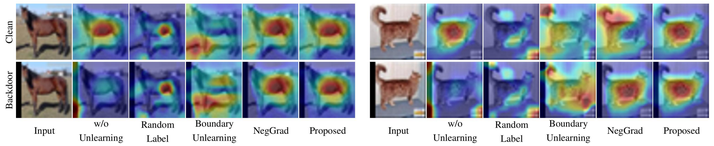

(d) Backdoor: Preserving test accuracy


(c) Diffusion: Ratio of images classified as “4”


(d) Diffusion: KID to all real images

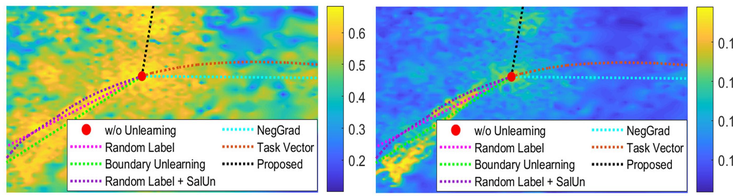
Visualization

- Class Activation Maps for backdoor attack scenario
  - Extracted for both clean images and backdoor images



- Generated images after unlearning
  - Red: “4”-like images
  - Yellow: Unnatural images

- Landscape visualization of Unlearning Trajectory
  - For the scenario of label noise, we extracted the unlearning trajectories and their backgrounds using exhaustive search and PCA



Test Accuracy (↑)

Accuracy on $\mathcal{D}_f$ with noised label (↓)

Conclusion

- We addressed key unlearning limitations:
  - Reliance on remaining data,
  - Risk of introducing incorrect knowledge,
  - Limited applicability, and
  - Inefficacy in complex search spaces.
- Our InvWNN predicts past weight using only forget data to mitigate these issues.
- Our implementation is available at https://github.com/jjh6297/InvWNN