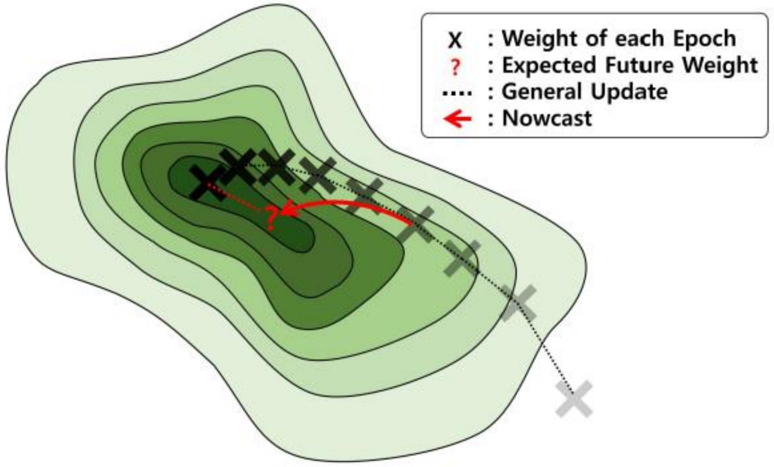
Learning to Boost Training by Periodic Nowcasting Near Future Weights

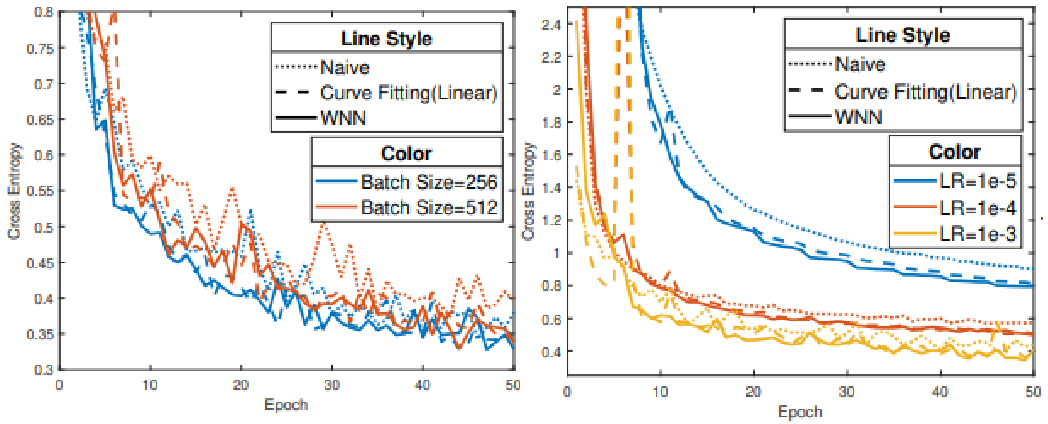
Jinhyeok Jang, Woo-han Yun, Won Hwa Kim,
Youngwoo Yoon, Jaehong Kim, Jaeyeon Lee,
ByungOk Han

Weight Prediction

- **Can we bypass the training process and directly arrive at future weights?**



Description of Optimization as Contour



General Tendency of Loss graph

Our Strategy

- 1) Learning-based regression model
- 2) Element-wise independent forecasting.
- 3) Separate forecasting for each mathematical operation,
- 4) Periodic short-term nowcasting per every 5 epochs,
- 5) Predicting residual between the future and the current weights
- 6) Applying a forecast network trained on Image Classification Task to various tasks

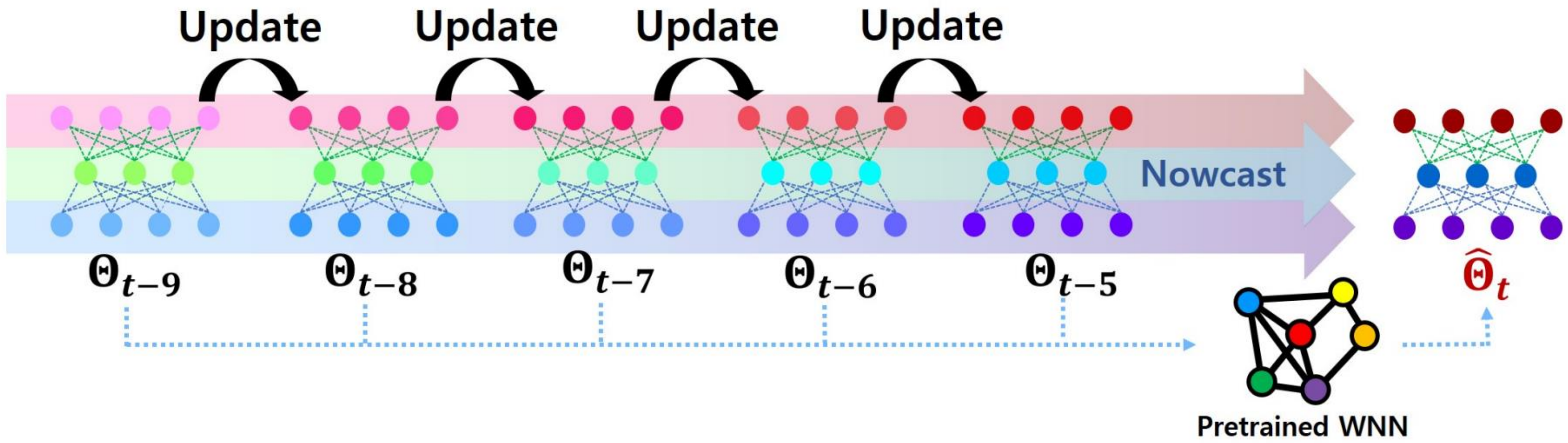


Figure 4. Conceptual view of short-term prediction with the proposed WNN.

Weight Nowcaster Network (WNN)

- For weight forecast, two things were considered:
  1) **Accurate Regression**
  2) **Fast Process**: ~5,000 parameters size of WNN

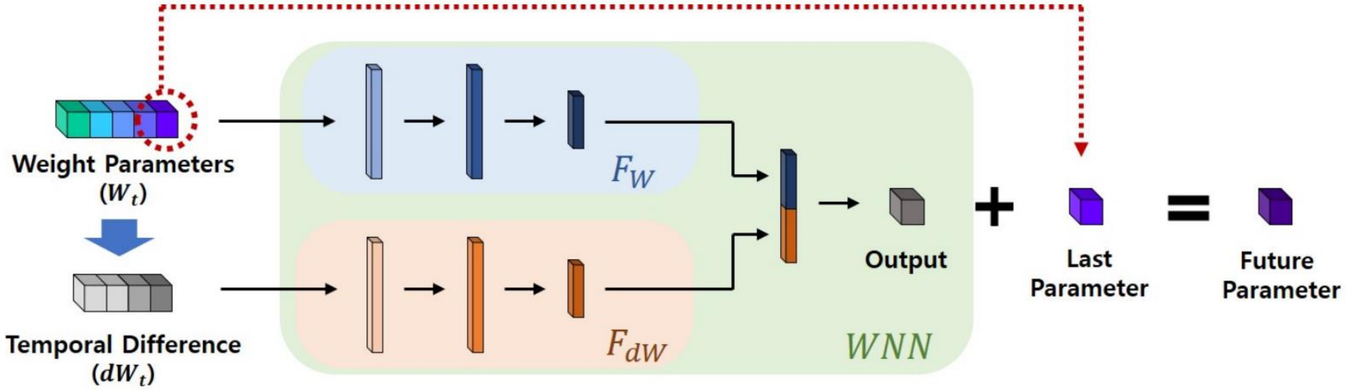


Figure 5. **Weight Nowcaster Network Architecture**. The WNN is composed of simple two-stream networks that use fully-connected layers and an activation network. Feature vectors from those two networks are unified to a feature vector and it is passed through a fully-connected layer. The predicted future weight parameters are obtained by adding outputs and input weight parameters.

Data Collection

- Architecture: LeNet, VGG, ResNet, MobileNetV2, ShuffleNetV2, DenseNet
- Dataset: MNIST, CIFAR10
- Optimizer: Adam
- Training Data: Weights of 30,000 epochs (~1.8e+10 parameters)
- Validation: 2,200 epochs of ShuffleNetV2 and ResNet32

Experiments I

- Application to Vanilla CNN + CIFAR10
- Comparisons with
  1) Various Curve Fitting Models
  2) Various Acceleration Methods

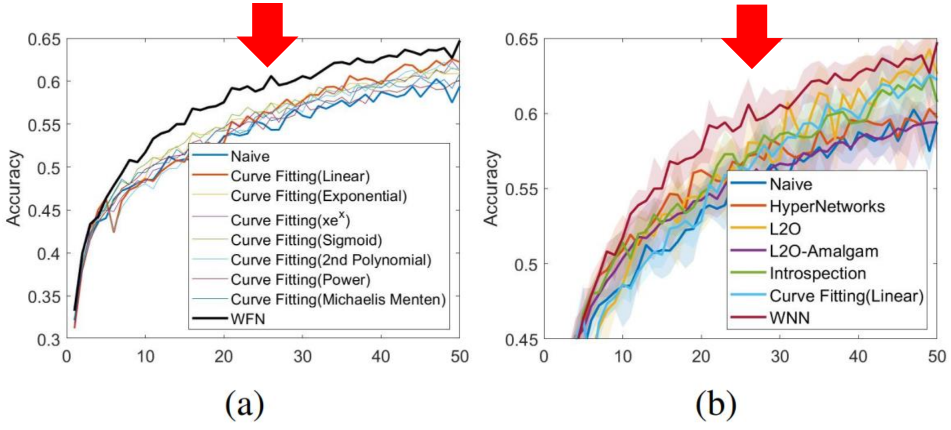


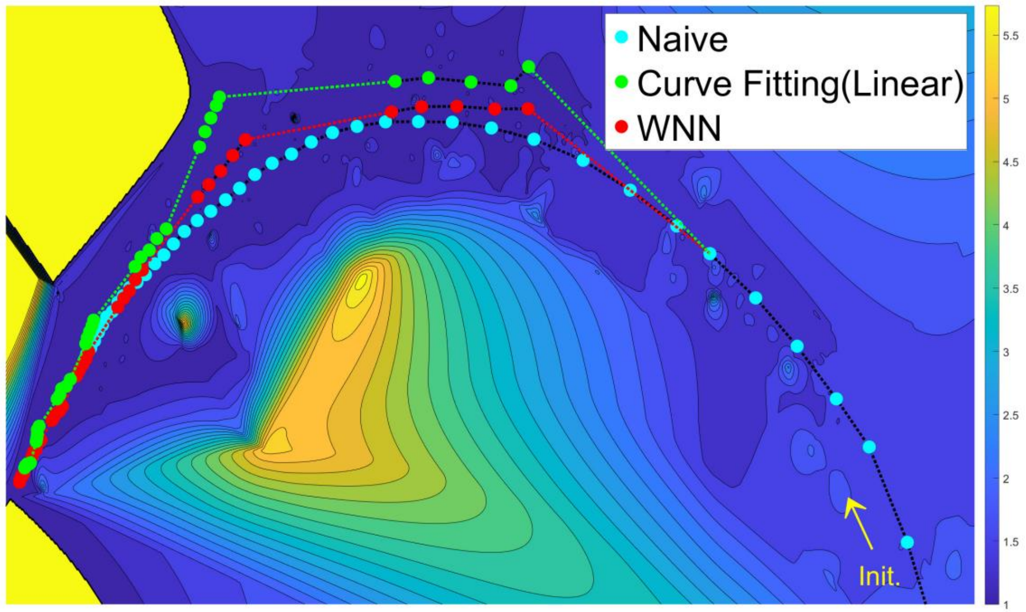
Figure 5. Comparisons of validation accuracy (a) of the proposed and curve fitting models, (b) of the proposed with previous methods (HyperNetworks, L2O, L2O-Amalgam, and Introspection). The shading represents the variation in validation accuracy of five trials.

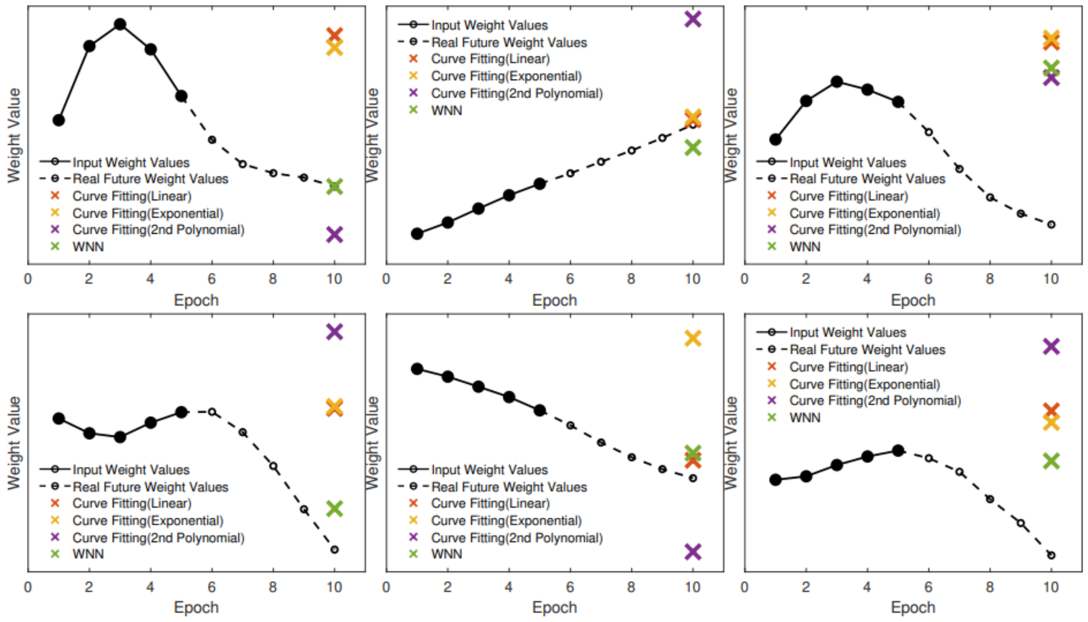
Table 4. Time cost comparisons to train a Vanilla CNN by using various recent methods on the CIFAR10 dataset. NVIDIA TITAN Xp GPU was used to estimate time cost. "Converge" is the time to reach a validation accuracy of 59%

Method	Update (sec/batch)	meta-learning (hour)	forecasting (sec)	Converge (sec)	Speed Up
Naive Training	0.0245	-	-	52.82	×1.00
HyperNetworks	0.0267	-	-	51.02	×1.04
L2O	0.0290	20 (per task)	-	44.05	×1.20
L2O-Amalgam	0.0291	6.35 (only once)	-	67.02	×0.79
Introspection	0.0245	0.03 (only once)	0.015	43.23	×1.22
CF (Linear)	0.0245	-	0.015	39.71	×1.33
CF (Exponential)	0.0245	-	3.32	61.94	×0.85
WNN	0.0245	0.08 (only once)	0.015	**25.27**	×2.09

Experiments II

- Application to Vanilla CNN and CIFAR10
- Comparisons with Linear Curve Fitting Model





Experiments III

- **Further Tasks**

(i) **ImageNet Classification.** ImageNet (Deng et al., 2009) is a widely-used large-scale dataset with 1.3 million images from 1,000 classes. We trained the MobileNetV2 using the Adam with exponential decay.

(ii) **Image Segmentation.** DeepLabV3+ (Chen et al., 2018) with the MobileNetV2 backbone was trained on the PASCAL VOC 2012 dataset (Everingham et al., 2015) with the SGD, cosine decay, and cross entropy.

(iii) **Pose Estimation.** We adopted OpenPose (Cao et al., 2017) with an ImageNet-pre-trained VGG19 backbone on the MS COCO 2016 (Lin et al., 2014) that consists of over 100K persons' keypoints.

(iv) **Language Modeling.** The goal is to predict a masked word from a sequence of words, *i.e.*, text. The universal BERT (Dehghani et al., 2019) on the WikiText-2 dataset which consists of over 2 million words was trained with the Adam, the masked LM loss, and the penalized confidence (Pereyra et al., 2017).

(v) **Reinforcement Learning.** We applied WNN to the Pendulum problem, which is a famous reinforcement learning problem using the DDPG (Deep Deterministic Policy Gradient) (Lillicrap et al., 2015). Two networks, an actor and a critic, were trained using the Adam for 200 episodes.

(vi) **Transfer Learning on Attention Model.** PVTv2-B0 (Wang et al., 2022) with ImageNet-pre-trained weights was trained on CIFAR100 for 50 epochs using the Adam, the warm-up scheduling, and decay.

(vii) **Diffusion Model.** We validated on the Denoising Diffusion Implicit Model (DDIM) (Song et al., 2020) on the Oxford Flowers dataset (Nilsback & Zisserman, 2008). The model was trained to minimize $\ell 1$ loss using the AdamW with the learning rate decay from $1e$-3 for 60 epochs. We evaluated it based on KID (Kernel Inception Distance) metric (Bińkowski et al., 2018).

Table 5. Experimental comparisons on various tasks with naive training, Introspection, a linear curve fitting, and the proposed WNN. WNN consistently outperforms the other methods on a variety of tasks.

Task (Metric)	Method	Best	Converge (epoch/episode)	Reach	Speed Up	Task (Metric)	Method	Best	Converge (epoch/episode)	Reach	Speed Up
ImageNet Classification (Val Acc↑)	Naive	69.40%	26	68.00%	× 1.00	RL (Episode Reward↑)	Naive	-139.10	116	-200.00	× 1.00
	Introspection	70.13%	24	68.00%	× 1.08		Introspection	-128.70	107	-200.00	× 1.08
	CF (Linear)	68.19%	41	68.00%	× 0.63		CF (Linear)	-138.79	111	-200.00	× 1.05
	WNN	70.57%	21	68.00%	× **1.24**		WNN	-123.55	88	-200.00	× **1.32**
Image Segmentation (Val Jaccard↑)	Naive	53.99%	55	48.00%	× 1.00	Transfer Learning on Attention Model (Val Acc↑)	Naive	83.22%	17	80.00%	× 1.00
	Introspection	53.97%	46	48.00%	× **1.20**		Introspection	81.52%	38	80.00%	× 0.45
	CF (Linear)	53.79%	50	48.00%	× 1.10		CF (Linear)	82.51%	14	80.00%	× **1.21**
	WNN	53.81%	45	48.00%	× **1.20**		WNN	83.09%	14	80.00%	× **1.21**
Pose Estimation (Val Loss↓)	Naive	623.54	43	673.00	× 1.00	Diffusion Model (Val KID↓)	Naive	0.1918	32	0.2000	× 1.00
	Introspection	627.17	46	673.00	× 0.94		Introspection	0.1786	36	0.2000	× 0.89
	CF (Linear)	614.80	30	673.00	× **1.43**		CF (Linear)	0.1878	54	0.2000	× 0.59
	WNN	615.79	30	673.00	× **1.43**		WNN	0.1732	26	0.2000	× **1.23**
Language Modeling (Val Perplexity↓)	Naive	31.25	55	100.00	× 1.00	Average	Naive	N/A	N/A	N/A	× 1.00
	Introspection	39.93	56	100.00	× 0.98		Introspection	N/A	N/A	N/A	× 0.94
	CF (Linear)	42.67	58	100.00	× 0.95		CF (Linear)	N/A	N/A	N/A	× 0.99
	WNN	31.26	48	100.00	× **1.15**		WNN	N/A	N/A	N/A	× **1.25**

Code Availability

- https://github.com/jjh6297/WNN

Weight Nowcasting Network (WNN)

Code for ["Learning to Boost Training by Periodic Nowcasting Near Future Weights"]

dependency

| Library | Known Working | Known Not Working |
| tensorflow | 2.3.0, 2.9.0 | <= 2.0 |

Usage

WNN can be easily used as a callback function extending tf.keras.callbacks.Callback:

```
import tensorflow as tf
import tensorflow.keras
from WNN import *

.
.
.

model.fit(..., callbacks=[WeightForecasting()])
```

Pre-trained Weights

Pre-trained weights of WNN are included. 'NWNN_XXX_13.h5 ' in this repo are the pre-trained weights for each mathematical operation type (Conv, FC, Bias).

Thanks