Exploring Super-Convergence in Analog Hardware Acceleration Kit for In-memory Computing Design

Team: Mian Hamza, Alexander Fernandes, Jack Hu, Hung-Yang Chang

McGill UNIVERSITY

Introduction

Deep neural network (DNN) is a popular AI/ML algorithm used in many complex tasks such as image/speech recognition.

Training DNNs is computationally intensive & uses a lot of resources.

- This is because computation & memory endpoints are in different locations on a chip: data is transferred back and forth between storage and computation units frequently.
- This is known as the Von Neumann bottleneck, it prevents a system from achieving real-time & energy-efficient computation.

Goals

Mitigate Von Neumann bottleneck => Accelerate the training time

Solution is to implement a **hardware-software co-optimization** approach with **in-memory computing architectures** & the **super-convergence algorithm.**

Methodology

1. IBM Analog Hardware Acceleration Kit (*aihwkit*) Setup

- Emulates in-memory analog NNs.
- Implements Resistive Processing Units (RPU): analog weights that represent values & are updated locally during the weight update step.
- As shown in Figure 1, the network adheres to Pytorch's *Sequential* structure but the per-layer & optimizers are replaced with analog layers and analog optimizers.

$$\omega_{ij} \leftarrow \omega_{ij} + \Delta\omega_{min} \sum_{n=1}^{BL} A_i^n \wedge B_j^n$$

- Three Device configs: Gokmen, Semi-ideal, Normal Pytorch
  Gokmen: Includes cycle-to-cycle & device-to-device variation
  Semi-ideal: Same as Gokmen, but without cycle-to-cycle variation

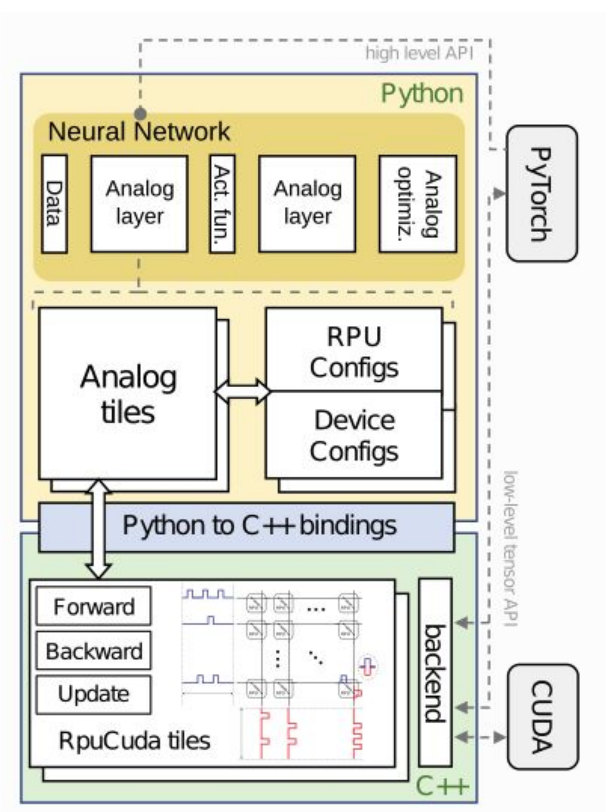

Figure 1: Architecture of aihwkit

2. Super Convergence Algorithm: **One-cycle CLR policy**

- The learning rate initially starts small to allow convergence in the correct direction in the latent space.
- As the network traverses the flat valley, the learning rate is large allowing for a faster progress through the valley.
- In the final training stages, when the network needs to converge to a local minimum the learning rate reduced to a small value.

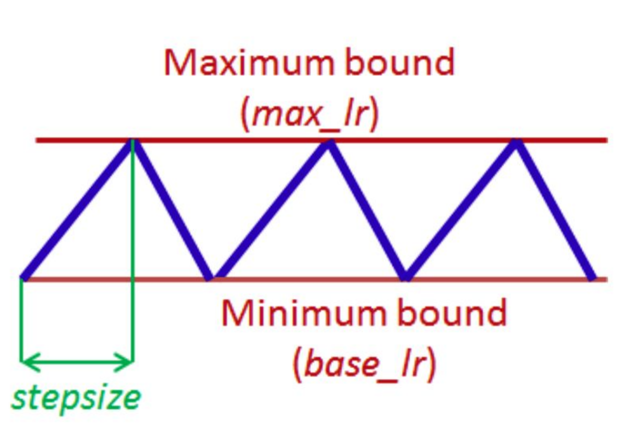

Figure 2: Cyclic Learning Rate

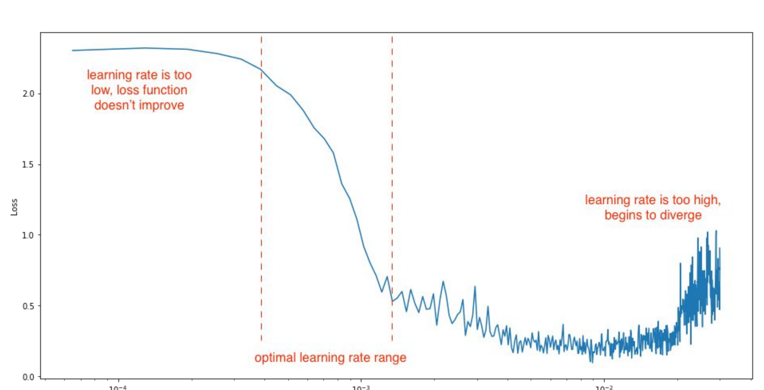

Figure 3: Learning Rate Range Test

Results

Dataset	Architecture	PL/Setting	CM/SS	WD	Batch Size	Epoch	Accuracy N (%)	Accuracy SI (%)	Accuracy GV (%)	
Cifar-10	ResNet-18	step/0.01/0.5/10	0.9	10^{-4}	512	100	80.01 ± 0.10	66.74 ± 0.52	54.32 ± 0.37	normal
Cifar-10	ResNet-18	step/0.01/0.5/10	0.9	10^{-4}	512	200	79.96 ± 0.09	66.75 ± 0.38	54.44 ± 0.55	normal
Cifar-10	ResNet-18	CLR/0.1-0.5/12	0.95-0.85/12	10^{-4}	512	25	83.88 ± 0.36	79.92 ± 0.27	58.83 ± 1.69	sc
Cifar-10	ResNet-18	CLR/0.1-0.5/23	0.95-0.85/23	10^{-4}	512	50	86.59 ± 0.31	82.98 ± 0.16	63.95 ± 0.88	sc
Cifar-10	VGG-8	step/0.01/0.1/10	0.9	10^{-4}	128	50	87.47 ± 0.12	84.20 ± 0.46	75.02 ± 0.68	normal
Cifar-10	VGG-8	CLR/0.01-0.15/12	0.95-0.85/12	10^{-4}	128	25	88.88 ± 0.18	86.16 ± 0.50	79.92 ± 0.99	sc
SVHN	VGG-8	step/0.01/0.1/10	0.9	5×10^{-4}	128	50	95.01 ± 0.01	93.89 ± 0.01	88.53 ± 0.28	normal
SVHN	VGG-8	CLR/0.01-0.15/12	0.95-0.85/12	5×10^{-4}	128	25	95.71 ± 0.04	93.04 ± 0.78	91.74 ± 0.08	sc
MNIST	LeNet	step/0.01/0.94/2	0.9	5×10^{-4}	512	50	84.44 ± 3.38	79.78 ± 0.02	82.99 ± 3.38	normal
MNIST	LeNet	step/0.01/0.94/2	0.9	5×10^{-4}	512	85	92.97 ± 0.34	92.02 ± 0.27	91.92 ± 0.29	normal
MNIST	LeNet	CLR/0.01-0.1/5	0.95-0.85/5	5×10^{-4}	512	12	98.59 ± 0.03	89.89 ± 0.38	90.07 ± 1.11	sc
MNIST	LeNet	CLR/0.01-0.1/12	0.95-0.85/12	5×10^{-4}	512	25	98.89 ± 0.06	93.74 ± 0.36	94.25 ± 0.14	sc

Table 1: Final accuracy & standard deviation for the specified datasets and architectures

The table 1 above shows the results for testing super convergence (rows with CLR) compared with normal training (rows with step).

- Networks are trained with the normal, semi-ideal, & Gokmen Vlasov configurations.
- Accuracies represent the average and standard deviation for the given epoch.

Across all datasets & architectures: super-convergence produced higher test accuracies compared to normal training for all device configuration comparisons. Shows that super convergence benefits CNNs in image classification and will improve results for semi-ideal & Gokmen Vlasov device configurations. Largest improvement (test accuracy) was with CIFAR-10 ResNet-18. Using the semi-ideal device, the accuracy improved by 16% when comparing CLR of 50 epochs to the regular stepLR of 200 epochs.

- Super-convergence can counter local minimum convergences by varying the learning rate between the minimum & maximum bounds, allowing for the optimizer to bypass local minimum convergences.

In SVHN and MNIST datasets, accuracies are lower with semi-ideal & Gokmen Vlasov device configurations and are improved after using super convergence. Figures 4 & 5 show how the use of CLR prevents the test accuracy and loss to converge. It bypasses the converged value around the 40-50th epoch for normal, semi-ideal, and Gokmen Vlasov device configurations.

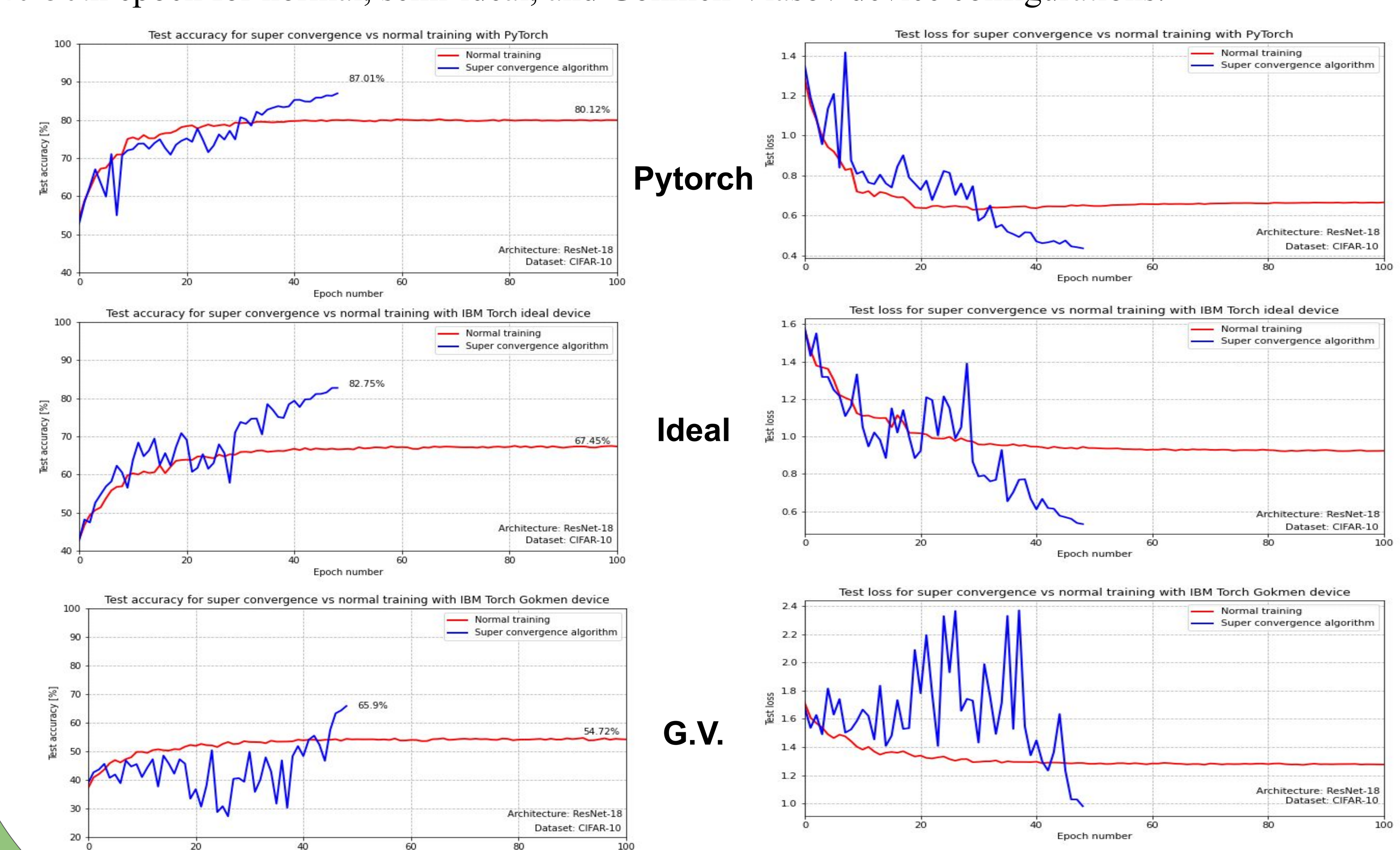

Figure 4 & 5: Test accuracy/loss for different device configurations in CIFAR-10 ResNet-18

Conclusion

- Compared **super convergence CLR** to **normal learning rate per step** on four separate dataset and architecture experiments.
- Observed that **super convergence improves the accuracy** for normal, semi-ideal, & Gokmen Vlasov device configurations.
- The largest noticeable improvement was for the semi-ideal device for CIFAR-10 ResNet-18 showing that after 50 epochs, the **test accuracy improved by 16%** (67% to 83%).
- When using super convergence methods, the trained neural networks will be **less likely to converge to local minimums**.
- Super convergence with analog hardware **will improve the speed and accuracy** of deep learning training for future studies.

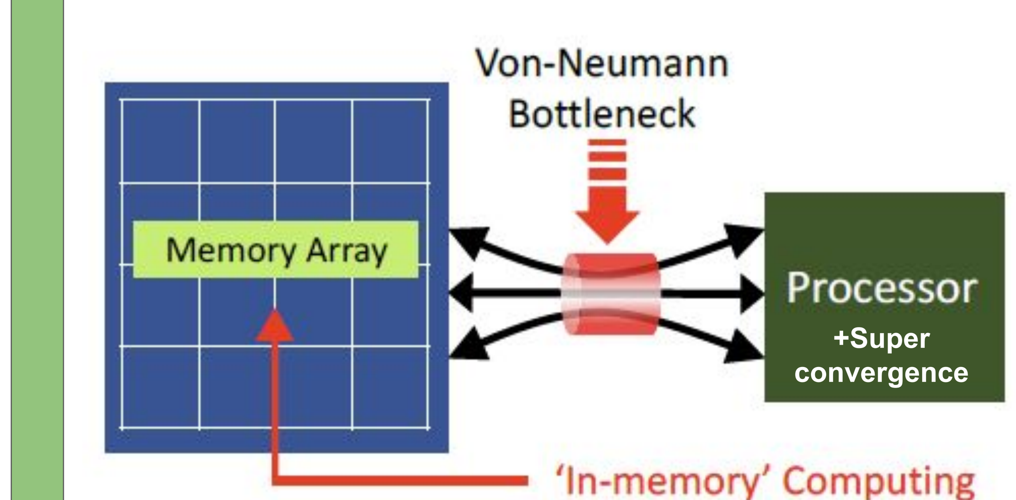

Figure 6: In Memory computing with Super Convergence

References

Ambrogio, S. *et al.* (2018). Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature*, **558**.

Gokmen, T. and Vlasov, Y. (2016). Acceleration of deep neural network training with resistive cross-point devices: Design considerations. *Frontiers in Neuroscience*, **10**, 333.

He, K. *et al.* (2015). Deep residual learning for image recognition.

Howard, J. (2021). Fast ai course. https://course.fast.ai/.

IBM (2020). Ibm analog hardware acceleration kit. https://github.com/IBM/aihwkit.

Ielmini, D. and Wong, H.-S. P. (2018). In-memory computing with resistive switching devices. *Nature Electronics*, **1**(6), 333–343.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Smith, L. N. (2017). Cyclical learning rates for training neural networks.

Smith, L. N. and Topin, N. (2018). Super-convergence: Very fast training of neural networks using large learning rates.

Y. LeCun, L. Bottou, Y. B. and Haffner., P. (1998). Gradient-based learning applied to document recognition.

Yuval Netzer, Tao Wang, A. C. A. B. B. W. and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.