

Supplementary Material

Layer-wise Learning of CNNs by Self-tuning Learning Rate and Early Stopping at Each Layer

Melika Sadeghi Tabrizi¹
melikasadeghi16@ut.ac.ir

Ali Karimi¹
aliiikarimi@ut.ac.ir

Ahmad Kalhor¹
akalhor@ut.ac.ir

Babak Nadjar Araabi¹
araabi@ut.ac.ir

Mona Ahmadian²
m.ahmadian@surrey.ac.uk

¹ School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran

² University of Surrey, Guildford, UK

Abstract

In this supplementary material, there are graphs related to the results of Tables (2) and (3) of the main paper. These charts are not new results; they directly correspond to the findings reported in the mentioned tables. They show how the neural network training process has been using our proposed learning strategy. Additionally, a sample code of the implementation of our proposed learning strategy is attached to this file. This implementation uses the PyTorch library and Python programming language.

1 Introduction

This supplementary material provides additional figures and detailed analyses of experiments that complement the results presented in our paper, "Layer-wise Learning of CNNs by Self-tuning Learning Rate and Early Stopping at Each Layer." The following materials were prepared during the paper submission but were not included due to format and space constraints. The figures offer deeper insights into the comparative performance of different optimization strategies and learning rate adjustments in our proposed learning strategies. The content of this file does not feature any additional experiments beyond those presented in the original paper.

It includes graphs related to the experiments outlined in Tables 2 and 3 of the main paper, providing a more comprehensive understanding of the neural network training process with our proposed learning strategy. In all the figures of this report, column y introduces the value of Separation Index and column x shows the number of training epoch.

The values in Tables 2 and 3 were obtained from multiple experiments and then averaged. It should be noted that the graphs are related to one of these implementations. Therefore, the numbers reported in Tables 2 and 3 may not necessarily be equal to the values on the graph.

Dataset	Train Size (Per Class)	Test Size (Per Class)	Number of Classes
CIFAR-10	5,000	10,00	10
CIFAR-100	500	100	100
STL-10	500	800	10

Table 1: Comprehensive Dataset Specifications Used in the Experimental.

2 Comparing Impact of Parameters in Our Learning Strategy

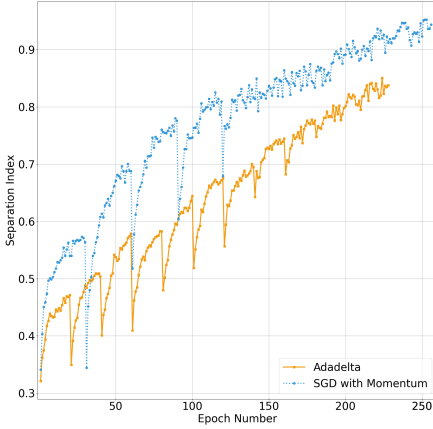
Optimizer				Loss Functions			
—		Maximum SI		—		Maximum SI	
Optimizer Name	CIFAR-10	STL-10	CIFAR-10	STL-10	Loss Function Name	CIFAR-10	STL-10
Adam 2	92.20±1.5	74.23±1.7	91.81	74.61	ArcFace	91.14±0.5	72.29±0.9
RMSProp 3	91.85±0.7	70.86±1.5	93.97	68.09	Contrastive	90.94±0.2	68.94±1.5
Adadelta 1	86.73±1.6	64.05±2.4	85.05	62.1	Cross-Entropy	90.92±1.0	72.09±1.7
Momentum	92.49±0.2	74.48±0.8	95.23	75.07	Triplet	92.49±0.2	74.47±0.8
			95.23	75.07			

Table 2: Comparison of various optimizers and loss functions for training on CIFAR-10 and STL-10 datasets using the VGG16 architecture

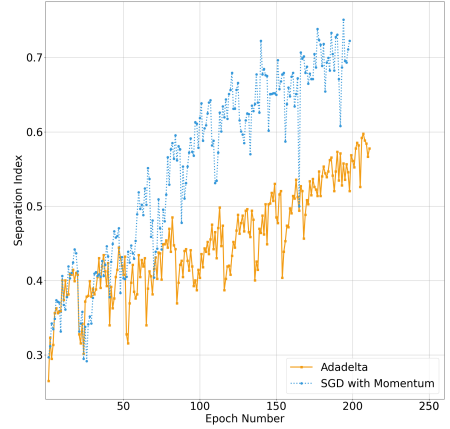
Learning Rate Schedulers				Stage Status & Stage Freezing Status			
—		SI		—		Maximum SI	
LR Scheduler	CIFAR-10	STL-10	CIFAR-10	STL-10	Stage and Freezing Status	CIFAR-10	STL-10
Incremental 5	86.73±0.1	70.55±0.2	87.78	68.57	Block-wise (Freeze) 8	82.08±0.4	53.16±3.6
Decremental 6	84.29±0.5	66.29±1.0	85.21	64.69	Layer-wise (Freeze) 7	76.73±0.6	58.27±2.7
Fixed 4	86.48±0.3	67.73±1.6	84.35	66.40	Block-wise (No Freeze) 8	90.48±0.3	74.88±0.5
Based on SI	92.49±0.2	74.47±0.8	95.23	75.07	Layer-wise (No Freeze) 7	92.49±0.2	75.48±0.8
			95.23	75.07			

Table 3: Comparison of various learning rate schedulers and stages status for training on CIFAR-10 and STL-10 datasets using the VGG16 architecture

3 Comparison of Optimizers in our layer-wise learning strategy

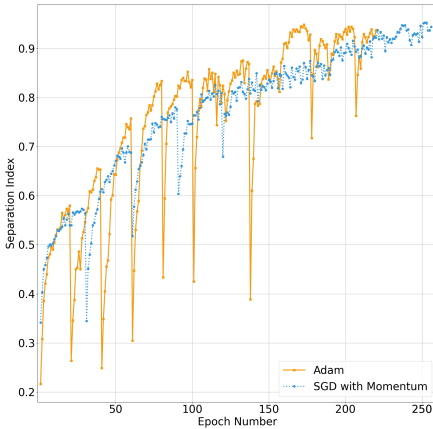


(Adadelta VS Momentum on CIFAR-10)

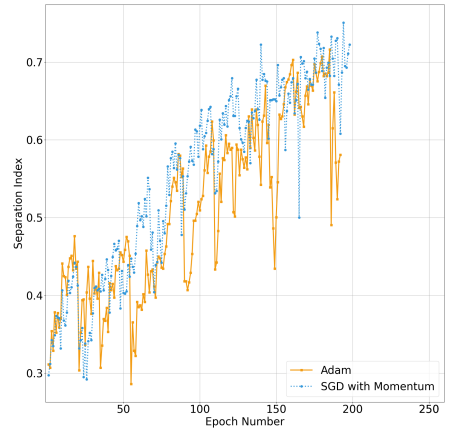


(Adadelta VS Momentum on STL-10)

Figure 1: SGD with Momentum outperforms Adadelta on both CIFAR-10 and STL-10 datasets, showing faster convergence, greater stability, and higher final Separation Index values

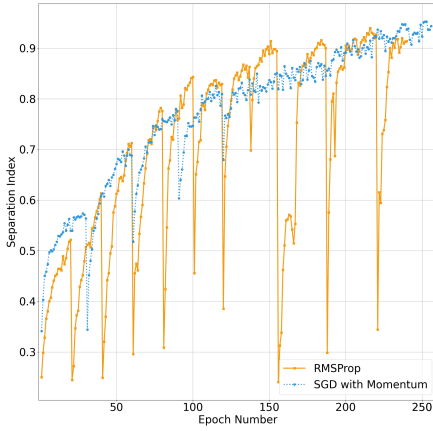


(Adam VS Momentum on CIFAR-10)

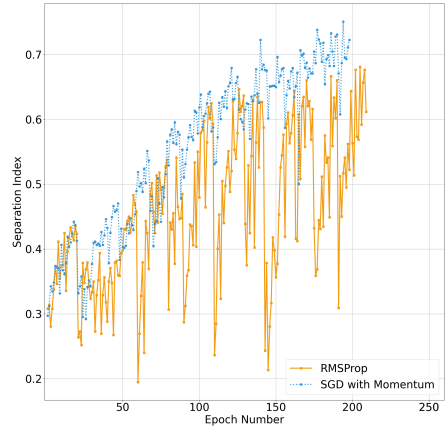


(Adam VS Momentum on STL-10)

Figure 2: SGD with Momentum has a more stable and consistent SI improvement, making it preferable for achieving steady training progress



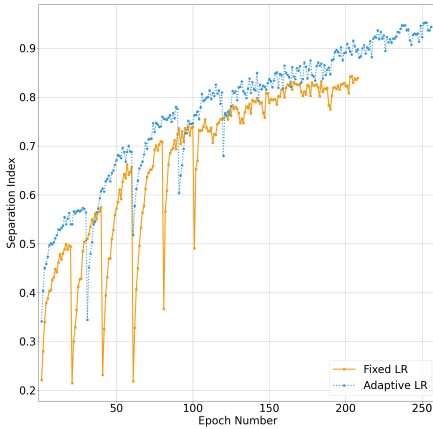
RMSProp VS Momentum on CIFAR-10



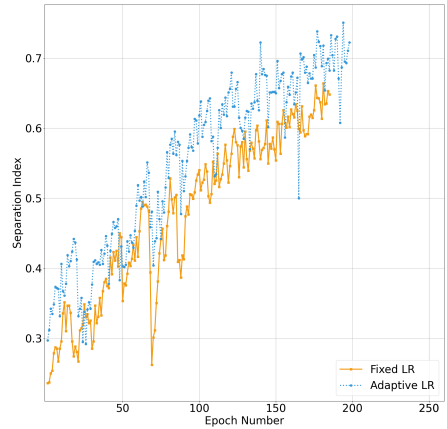
RMSProp VS Momentum on STL-10

Figure 3: RMSProp shows significant fluctuations and slower convergence compared to the smoother and more rapid improvement seen with SGD with Momentum.

4 Comparison of learning rate in our layer-wise learning strategy

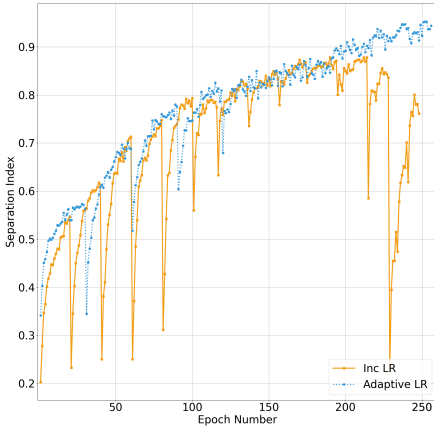


Fixed learning rate VS our adaptive learning rate on CIFAR-10

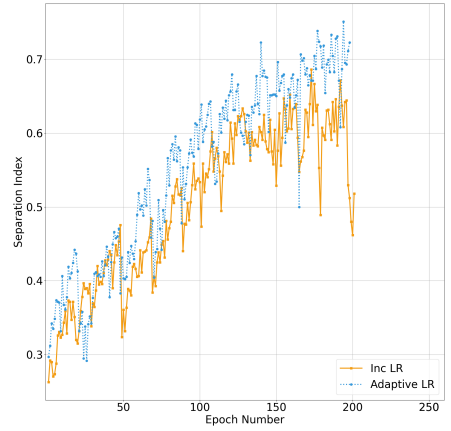


Fixed learning rate VS our adaptive learning rate on STL-10

Figure 4: Our Adaptive Learning Rate achieves higher and more stable SI values compared to Fixed Learning Rate, showing less fluctuation and faster improvement

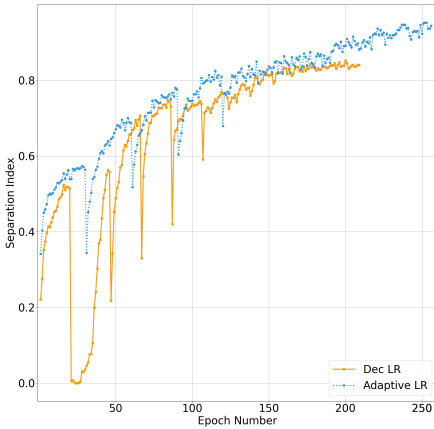


Incremental learning rate VS our adaptive learning rate on CIFAR-10

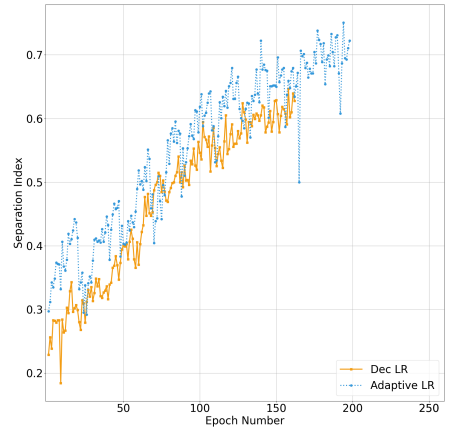


Incremental learning rate VS our adaptive learning rate on STL-10

Figure 5: Our Adaptive Learning Rate consistently outperforms Incremental Learning Rate, demonstrating higher stability and faster SI convergence



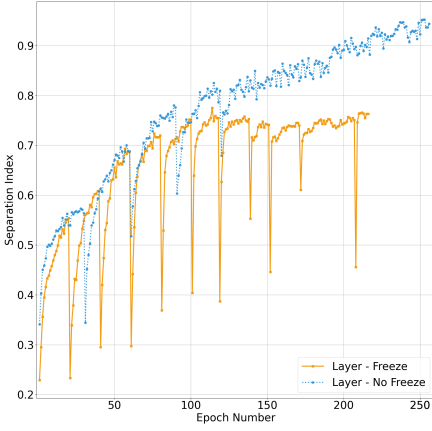
Decremental learning rate VS our adaptive learning rate on CIFAR-10



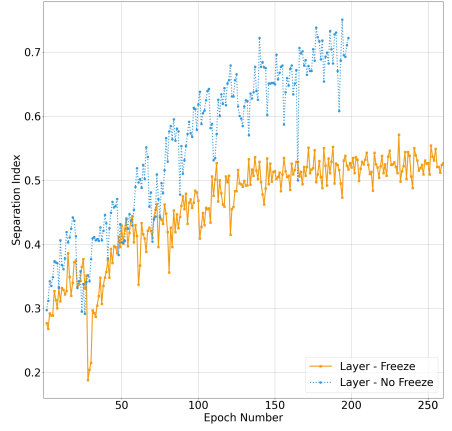
Decremental learning rate VS our adaptive learning rate on STL-10

Figure 6: Our Adaptive Learning Rate provides more stable and faster SI improvement compared to Decremental Learning Rate, with higher final SI values

5 The Impact of freezing in Our stage-wise Strategy

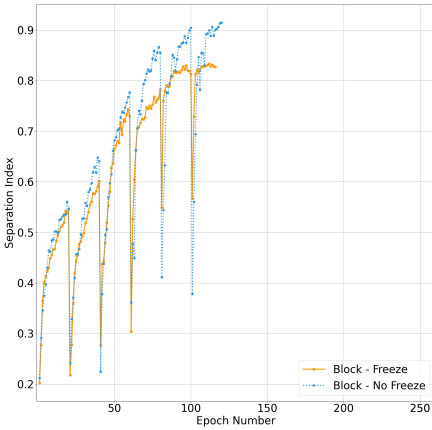


freezing VS not freezing layers in our layer-wise learning with the CIFAR-10 dataset

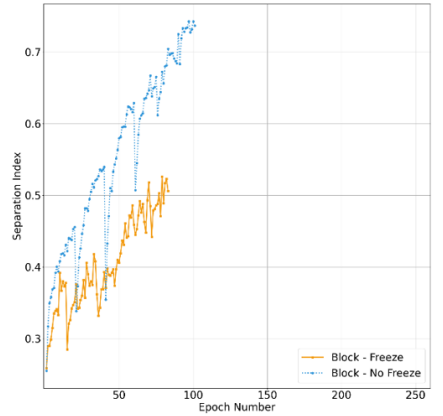


freezing VS not freezing layers in our layer-wise learning with the STL-10 dataset

Figure 7: Freezing layers during training results in slower and less stable SI improvements, while not freezing layers leads to faster and more consistent SI improvements with higher final SI values.



freezing VS not freezing blocks in our block-wise learning with the CIFAR-10 dataset



freezing VS not freezing blocks in our block-wise learning with the STL-10 dataset

Figure 8: Freezing blocks during training results in slower and less stable SI improvements, while not freezing blocks leads to faster and more consistent SI improvements with higher final SI values.