LST-Net: Learning a Convolutional Neural Network with a Learnable Sparse Transform

Lida Li[1[0000−0001−9386−194X]⋆], Kun Wang[2,1[0000−0001−5954−8036]⋆], Shuai Li[1,3[0000−0003−0760−5267]], Xiangchu Feng[2[0000−0002−3463−2060]], and Lei Zhang[1,3[0000−0002−2078−4215]⋆⋆]

[1] Dept. of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[2] School of Mathematics and Statistics, Xidian University
[3] DAMO Academy, Alibaba Group
{cslli, csshuaili, cslzhang}@comp.polyu.edu.hk, kwang96@stu.xidian.edu.cn,
xcfeng@mail.xidian.edu.cn

In this supplementary material, we provide more details about:

1. Experimental settings;
2. The LST-Net structures *w.r.t.* existing CNN architectures (e.g., ResNet, VGG, AlexNet) on different datasets;
3. Extra experimental results of LST-Net for large-scale scene recognition.

1 Experimental settings

All experiments are conducted using an 8-way NVIDIA Tesla P100 GPU server with 2 Intel Xeon Gold 6136 CPUs and 128G RAM. **CIFAR-10 and CIFAR-100 datasets**. Standard data augmentation strategies [7,3] were adopted in training, including random horizontal flip, padding of four extra pixels on each side, random crop, etc. Each model was trained for 160 epochs. We used SGD with a mini-batch of 128 samples for optimization. Weight decay and momentum were set to 5×10^{-4} and 0.9, respectively. Learning rate started at 0.1, and was reduced by a factor of 10 after $32K$ and $48K$ iterations. One GPU card was used to train LST-Nets constructed *w.r.t.* ResNet-20 and ResNet-56 architectures; two GPU cards were employed for 110- and 164-layer LST-Nets; four GPU cards were adopted to train LST-Nets built regarding to other architectures. We did not use any SyncBN layers.
ImageNet LSVRC2012 dataset. By default, we follow the settings in [4,1] to compare different methods on the validation set (no test labels are released). SGD with a mini-batch of 256 samples was used for optimization. Weight decay was set to 1×10^{-4} and momentum to 0.9. We trained each model from scratch for 90 epochs. Learning rate started at 0.1, and was reduced by a factor of 10 for every 30 epochs. We employed four GPU cards to train LST-Net constructed *w.r.t.* ResNet-18, ResNet-34, ShiftNet, AlexNet and MobileNet V2. Eight GPU cards were all used to train other models.

⋆ The first two authors contribute equally in this work.

⋆⋆ Corresponding author. This work is supported by HK RGC General Research Fund (PolyU 152216/18E).

ImageNet-C dataset. We used the ImageNet-C dataset to study the robustness of those models trained on ImageNet. No fine-tuning was conducted for test on ImageNet-C.
Places365-Standard dataset. We reused the same training settings on ImageNet. We report the best Top-5 test accuracy achieved by ten-crop estimation for each model.

2 Detailed structures of LST-Net

We can construct our LST-Nets *w.r.t.* existing CNN architectures (e.g., ResNet, VGG and AlexNet, etc.) by replacing their main building blocks, such as conventional Conv2d layers or featured bottlenecks, with our proposed LST-I or LST-II bottleneck. For each existing CNN architecture, we closely followed its instantiation on different datasets to construct our corresponding LST-Net.

LST-Net *w.r.t.* **ResNet on CIFAR-10/100**. Table I shows the structures of LST-Net *w.r.t.* ResNet on CIFAR-10/100. We substituted each basic bottleneck of ResNet with a pair of LST-I or LST-II bottlenecks as there are two Conv2d operations in each original bottleneck. To keep the same classifier (the last FC layer), we inserted a PWConv before GAP (please refer to the third last row of Table Ib) so that C_{in} of FC remains 64.

LST-Net *w.r.t.* **ResNet on ImageNet**. Table II shows the architectures of LST-Net *w.r.t.* ResNet on ImageNet. For shallow models, such as ResNet-18 and ResNet-34, we built up LST-Net for ImageNet in the same way as that for CIFAR-10/100. For deep models, such as ResNet-50 and ResNet-101, we did not introduce extra PWConv before the GAP layer. We employed LST-II bottlenecks at each stage of conv2_x∼conv5_x with comparable number of parameters and computational cost.

LST-Net *w.r.t.* **WRN on CIFAR-10/100**. Table III presents the details of LST-Net constructed *w.r.t.* WRN on CIFAR-10/100. We adopted LST-II bottlenecks for construction. Following WRN, we enlarged the core channels of each LST-II bottleneck, *i.e.*, C_{r_out}, for a few times according to the pre-defined width multiplier.

LST-Net *w.r.t.* **WRN on ImageNet**. LST-II bottlenecks are adopted to construct LST-Net *w.r.t.* WRN on ImageNet. Following WRN, we enlarged the core channels of each LST-II bottleneck, *i.e.*, C_{r_out}, for a few times according to the pre-defined width multiplier. Thus, it has very similar structure to the one built up *w.r.t.* ResNet on ImageNet.

LST-Net *w.r.t.* **VGG on ImageNet**. Table IV presents the LST-Nets constructed *w.r.t.* VGG on ImageNet using two distinct classifiers. As VGG has a larger spatial size at various layers than that of the corresponding layers in ResNet, we adopt LST-I bottleneck for VGG to save overhead. LST-Net (FC) adopts the same classifier as the standard VGG, *i.e.* three FC layers. In contrast, classifier of LST-Net (GAP) is similar to that of ResNet.

LST-Net *w.r.t.* **AlexNet on ImageNet**. Table V presents the LST-Net constructed *w.r.t.* AlexNet on ImageNet. For the same reason as that of VGG, we employed LST-I bottleneck. LST-Net (FC) has the same classifier as the original AlexNet. In contrast, LST-Net (GAP) takes the same classifier structure as ResNet.

LST-Net *w.r.t.* **ShiftNet on ImageNet**. Table VI shows the LST-Net constructed *w.r.t.* ShiftNet on ImageNet. We employ LST-I bottleneck for the same reason as VGG and AlexNet. Besides, we set $a = 2$ for all bottlenecks. Following ShiftNet, we set the base width to 32 for LST-Net (A) and half the number for LST-Net (B) and LST-Net (C). We reduced a few bottlenecks at each stage to match its original expansion rate.

LST-Net *w.r.t.* **MobileNet V2 on ImageNet**. Table VII shows the structure of LST-Net built up *w.r.t.* MobileNet V2. To adapt LST-I bottleneck to the Inverted Residual bottleneck, we replaced the Inverted Residual bottlenecks in MobileNet V2 with modified LST-I bottlenecks and reused the original settings, including kernel size, stride, expansion rate $\mathcal{E}$, number of bottlenecks, etc. We made three changes for the modified version of LST-I bottlenecks: (1) we replaced each ReLU in the original LST-I bottleneck by ReLU6 and the ReLU-ST activation scheme was adapted to ReLU6-ST, where we set $\tau = 1 \times 10^{-8}$ in ST to take care of the need for a linear transform; (2) we removed PWConv and BN in channel-wise transform T_c when the expansion rate $\mathcal{E} = 1$; (3) we removed the downsample operator D and element-wise plus when $\mathcal{E} > 1$ while stride>1 or $C_{in} \neq C_{out}$. Fig. 3 illustrates the LST-I bottlenecks corresponding to MobileNet V2 bottlenecks. Batch size, initial learning rate and weight decay are set to 256, 0.05 and 5×10^{-4}, respectively. We adopted a cosine learning rate decay strategy and trained our model for 150 epochs.

Finally, we present the convergence curves of LST-Net on ImageNet in terms of Top-1 and Top-5 error rates. Figs. 1 and 2 compare the convergence curves of ResNet-18, ResNet-50 and their corresponding LST-Nets. One can see that our LST-Nets achieve lower error rates during the entire training process.

3 Extra experimental results of LST-Net for large-scale scene recognition

We evaluate LST-Net for large-scale scene recognition on Places365-Standard dataset [10]. We build up LST-Nets *w.r.t.* ResNet [1], AlexNet [6] and 11-layer VGG [8] for fair comparison. We compare LST-Net with its counterpart networks. Table VIII presents Top-5 accuracies obtained using ten-crop estimation.

One can see that LST-Net surpasses its counterparts. This validates that LST-Net is also effective for the large-scale scene recognition task. In particular, an 18-layer LST-Net can even surpass ResNet-50 by 1.27% while saving nearly 70% of the total parameters and 64% of the total FLOPs. Meanwhile, LST-Net under ResNet-50 architecture achieves the best performance on Places365-Standard dataset, 0.96% higher than its closest follower, CBAM-50. Besides, by replacing the last linear layers of AlexNet by GAP, the accuracy drops significantly, while LST-Net(GAP) is robust in this case. AlexNet (BN) only slightly improves AlexNet [5], while LST-Net (FC) built up *w.r.t.* AlexNet, also depicting BN and FC, improves much AlexNet (BN). Similarly, the accuracy drops by nearly 1% when the last linear layers of VGG is replaced by GAP, while LST-NET (GAP) constructed regarding to VGG is also robust in the same case. And LST-Net (FC) built up *w.r.t.* VGG improves VGG (BN) by 0.24%.

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks.In: Proc. ECCV. Springer (2016)
2. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proc. CVPR (2018)
3. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks withstochastic depth. In: Proc. ECCV. Springer (2016)
4. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training byreducing internal covariate shift. In: Proc. ICML (2015)
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proc. NeurIPS (2012)
6. Krizhevsky, A.: One weird trick for parallelizing convolutional neural networks. arXiv preprint arXiv:1404.5997 (2014)
7. Lin, M., Chen, Q., Yan, S.: Network in network. In: Proc. ICLR (2014)
8. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale imagerecognition. In: Proc. ICLR (2015)
9. Woo, S., Park, J., Lee, J.Y., So Kweon, I.: CBAM: Convolutional block attentionmodule. In: Proc. ECCV (2018)
10. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 millionimage database for scene recognition. IEEE Trans. Pattern Anal. Mach. Intell.40(6),1452–1464 (2018)

Table I: LST-Net constructed *w.r.t.* ResNet on CIFAR-10/100. Please refer to Table 1 and Table 2(a) in our paper.

(a) LST-I

Type/Stride	C_{in}	$a^2 \times C_s$	C_{out}	Repeat 20	56	110	164
Conv3x3/1	3	N.A.	16	1			
LST-I/1	16	64	16	5	17	35	53
LST-I/2	16	128	32	1			
LST-I/1	32			5	17	35	53
LST-I/2	32	256	64	1			
LST-I/1	64			5	17	35	53
GAP	64	N.A.	64	1			
FC	64	N.A.	10/100	1			

(b) LST-II (by default)

Type/Stride	C_{in}	C_{r_out}	C_{out}	Repeat 20	56	110	164
Conv3x3/1	3	N.A.	16	1			
LST-I/1	16	16	64	1			
	64			5	17	35	53
LST-II/2	64	32	128	1			
LST-II/1	128			5	17	35	53
LST-II/2	128	64	256	1			
LST-II/1	256			5	17	35	53
Conv1x1/1	256	N.A.	64	1			
GAP	64	N.A.	64	1			
FC	64	N.A.	10/100	1			

Table II: LST-Net constructed *w.r.t.* ResNet on ImageNet and Places365-Standard. Please refer to Table 3(a), Table 4 and Table 5(a) in our paper.

(a) 18 and 34 layers.

Name	Type/Stride	C_{in}	C_{r_out}	C_{out}	Repeat 18	Repeat 34
conv1	Conv7×7/2	3	N.A.	64	1	1
conv2_x	MaxPool3×3/2	64	N.A.	64	1	1
	LST-II/2	64	64	256	1	1
	LST-II/1	256			1	5
conv3_x	LST-II/2	256	128	512	1	1
	LST-II/1	512			1	7
conv4_x	LST-II/2	512	256	1024	1	1
	LST-II/1	1024			1	11
conv5_x	LST-II/2	1024	512	2048	1	1
	LST-II/1	2048			1	5
	Conv1x1/1	2048	N.A.	512	1	1
	GAP	512	N.A.	512	1	1
	FC	512	N.A.	$365/1K$	1	1

(b) 50 and 101 layers.

Name	Type/Stride	C_{in}	C_{r_out}	C_{out}	Repeat 50	Repeat 101
conv1	Conv7×7/2	3	N.A.	64	1	1
conv2_x	MaxPool3×3/2	64	N.A.	64	1	1
	LST-II/2	64	64	256	1	1
	LST-II/1	256			9	9
conv3_x	LST-II/2	256	128	512	1	1
	LST-II/1	512			13	13
conv4_x	LST-II/2	512	256	1024	1	1
	LST-II/1	1024			20	77
conv5_x	LST-II/2	1024	512	2048	1	1
	LST-II/1	2048			9	9
	GAP	2048	N.A.	2048	1	1
	FC	2048	N.A.	$365/1K$	1	1

Table III: LST-Net constructed *w.r.t.* WRN on CIFAR-10/100. Please refer to Table 2(b) in our paper.

(a) width multiplier = 8

Type/Stride	C_{in}	C_{r_out}	C_{out}	Repeat 16	Repeat 22	Repeat 28	Repeat 40
Conv3x3/1	3	N.A.	16	1			
LST-II/1	16	128	512	1			
	512			1	2	3	5
LST-II/2	512	256	1024	1			
LST-II/1	1024			1	2	3	5
LST-II/2	1024	512	2048	1			
LST-II/1	2048			1	2	3	5
Conv1x1/1	2048	N.A.	512	1			
GAP	512	N.A.	512	1			
FC	512	N.A.	10/100	1			

(b) width multiplier = 10

Type/Stride	C_{in}	C_{r_out}	C_{out}	Repeat 16	Repeat 22	Repeat 28	Repeat 40
Conv3x3/1	3	N.A.	16	1			
LST-II/1	16	160	640	1			
	640			1	2	3	5
LST-II/2	640	320	1280	1			
LST-II/1	1280			1	2	3	5
LST-II/2	1280	640	2560	1			
LST-II/1	2560			1	2	3	5
Conv1x1/1	2560	N.A.	640	1			
GAP	640	N.A.	640	1			
FC	640	N.A.	10/100	1			

Table IV: LST-Net constructed *w.r.t.* VGG on ImageNet and Places365-Standard. Please refer to Table 3(c) and Table 5(b) in our paper.

(a) LST-Net (FC)

Type/Stride	C_{in}	$a^2 \times C_s$	C_{out}	Repeat
Conv3×3/1	3	N.A.	64	1
MaxPool2×2/2	64	N.A.	64	1
LST-I/1	64	512	128	
MaxPool2×2/2	128	N.A.	128	1
LST-I/1	128	1024	256	
LST-I/1	256			
MaxPool2×2/2	256	N.A.	256	1
LST-I/1	256	2048	512	
LST-I/1	512			
MaxPool2×2/2	512	N.A.	512	1
LST-I/1	512	2048	512	2
FC	25088	N.A.	4096	1
FC	4096	N.A.	4096	
FC	4096	N.A.	$365/1K$	

(b) LST-Net (GAP)

Type/Stride	C_{in}	$a^2 \times C_s$	C_{out}	Repeat
Conv3×3/1	3	N.A.	64	1
MaxPool2×2/2	64	N.A.	64	1
LST-I/1	64	512	128	
MaxPool2×2/2	128	N.A.	128	1
LST-I/1	128	1024	256	
LST-I/1	256			
MaxPool2×2/2	256	N.A.	256	1
LST-I/1	256	2048	512	
LST-I/1	512			
MaxPool2×2/2	512	N.A.	512	1
LST-I/1	512	2048	512	2
GAP	512	N.A.	512	1
FC	512	N.A.	$365/1K$	1

Table V: LST-Net constructed *w.r.t.* AlexNet on ImageNet and Places365-Standard. Please refer to Table 3(c) and Table 5(b) in our paper.

(a) LST-Net (FC)

Type/Stride	C_{in}	$a^2 \times C_s$	C_{out}	Repeat
Conv11×11/4	3	N.A.	64	1
MaxPool3×3/2	64	N.A.	64	1
LST-I/1	64	768	192	
MaxPool3×3/2	192	N.A.	192	1
LST-I/1	192	1536	384	
MaxPool3×3/2	384	N.A.	384	1
LST-I/1	384	1024	256	
LST-I/1	256			
FC	9216	N.A.	4096	1
FC	4096	N.A.	4096	
FC	4096	N.A.	$365/1K$	

(b) LST-Net (GAP)

Type/Stride	C_{in}	$a^2 \times C_s$	C_{out}	Repeat
Conv11×11/4	3	N.A.	64	1
MaxPool3×3/2	64	N.A.	64	1
LST-I/1	64	768	192	
MaxPool3×3/2	192	N.A.	192	1
LST-I/1	192	1536	384	
MaxPool3×3/2	384	N.A.	384	1
LST-I/1	384	1024	256	
LST-I/1	256			
GAP	512	N.A.	512	1
FC	512	N.A.	$365/1K$	1

Table VI: LST-Net constructed *w.r.t.* ShiftNet on ImageNet. Please refer to Table 3(c) in our paper.

(a) LST-Net (A)

Type/Stride	C_{in}	$a^2 \times C_s$	C_{out}	Repeat
Conv7×7/2	3	N.A.	32	1
LST-I 5×5/2	32	128	32	1
LST-I 5×5/1				4
LST-I 5×5/2	32	256	64	1
LST-I 5×5/1	64			2
LST-I 3×3/2	64	512	128	1
LST-I 3×3/1	128			
LST-I 3×3/2	128	1024	256	1
LST-I 3×3/1	256			
GAP	256	N.A.	256	1
FC	256	N.A.	$1K$	1

(b) LST-Net (B)

Type/Stride	C_{in}	$a^2 \times C_s$	C_{out}	Repeat
Conv7×7/2	3	N.A.	16	1
LST-I 5×5/2	16	64	16	1
LST-I 5×5/1				4
LST-I 5×5/2	16	128	32	1
LST-I 5×5/1	32			2
LST-I 3×3/2	32	256	64	1
LST-I 3×3/1	64			
LST-I 3×3/2	64	512	128	1
LST-I 3×3/1	128			
GAP	128	N.A.	128	1
FC	128	N.A.	$1K$	1

(c) LST-Net (C)

Type/Stride	C_{in}	$a^2 \times C_s$	C_{out}	Repeat
Conv7×7/2	3	N.A.	16	1
LST-I 5×5/2	16	64	16	1
LST-I 5×5/1				
LST-I 5×5/2	16	128	32	1
LST-I 5×5/1	32			
LST-I 3×3/2	32	256	64	1
LST-I 3×3/1	64			
LST-I 3×3/2	64	512	128	1
LST-I 3×3/1	128			
GAP	128	N.A.	128	1
FC	128	N.A.	$1K$	1

Table VII: LST-Net constructed *w.r.t.* MobileNet V2 on ImageNet. Please refer to Table 3(c) in our paper.

Type/Stride	C_{in}	$a^2 \times C_s$ ($\mathcal{E}$)	C_{out}	Repeat
Conv3x3/1	3	N.A.	16	1
Modified LST-I/1	16	16 (1)	16	1
Modified LST-I/2 Modified LST-I/1	16 24	96 (6) 144 (6)	24	1 1
Modified LST-I/2 Modified LST-I/1	24 32	144 (6) 192 (6)	32	1 2
Modified LST-I/1	32 64	192 (6) 384 (6)	64	1 3
Modified LST-I/2 Modified LST-I/1	64 96	384 (6) 576 (6)	96	1 2
Modified LST-I/2 Modified LST-I/1	96 160	576 (6) 960 (6)	160	1 2
Modified LST-I/1	160	960 (6)	320	1
PWConv	320	N.A.	1280	1
GAP	1280	N.A.	1280	1
FC	1280	N.A.	$1K$	1

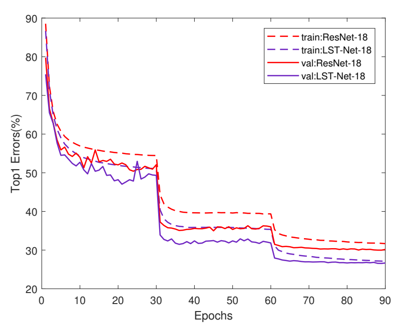
Table VIII: Results on Places365-Standard dataset.

(a) ResNet family.

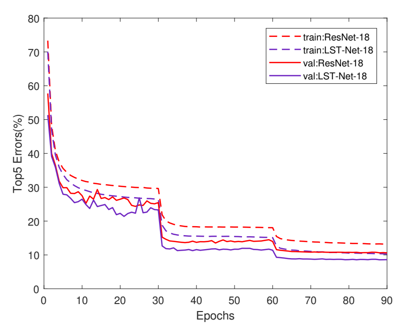
Model	Param/FLOPs	Top-5 Acc. (%)
ResNet-50 [1]	25.24M/4.09G	85.08
SENet-50 [2]	26.77M/4.09G	85.86
CBAM-50 [9]	26.79M/4.09G	86.22
LST-Net (ResNet-18)	**7.71M/1.48G**	**86.35**
LST-Net (ResNet-34)	**13.50M/2.56G**	**86.94**
LST-Net (ResNet-50)	**23.01M/4.05G**	**87.18**

(b) AlexNet and VGG.

Model	Param/FLOPs	Top-5 Acc. (%)
AlexNet [6]	58.50M/0.71G	82.89
AlexNet (BN)	58.50M/0.71G	82.98
AlexNet (GAP)	2.56M/0.66G	77.89
LST-Net (FC)	**57.70M/0.64G**	**83.99**
LST-Net (GAP)	**2.09M/0.62G**	**82.95**
VGG [8]	130.26M/7.61G	84.91
VGG (BN)	130.26M/7.61G	85.09
VGG (GAP)	9.73M/7.49G	83.95
LST-Net (FC)	**127.15M/6.01G**	**85.33**
LST-Net (GAP)	**6.30M/5.89G**	**85.12**

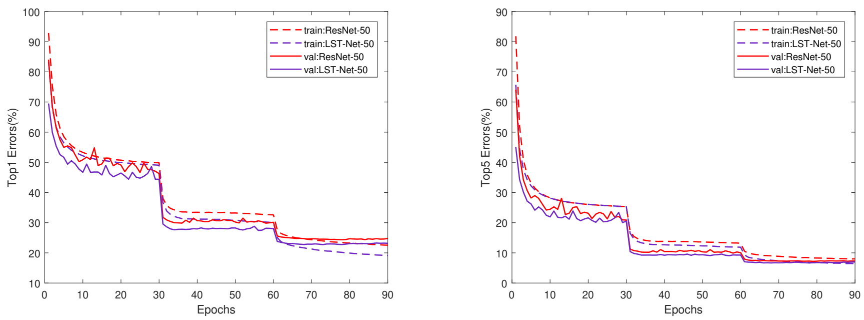


(a)



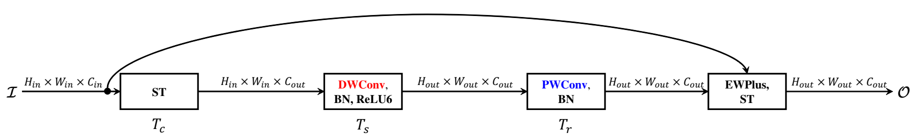
(b)

Fig. 1: Convergence curves of ResNet-18 and our LST-Net on ImageNet: (a) Top-1 error rates, and (b) Top-5 error rates.

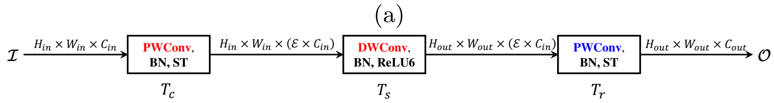


(a) (b)

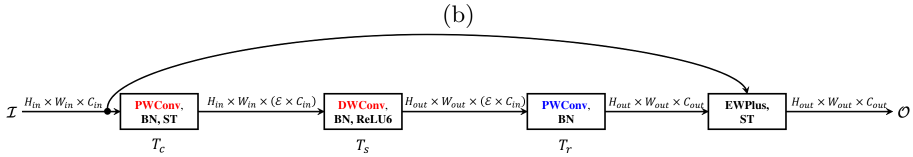
Fig. 2: Convergence curves of ResNet-50 and our LST-Net on ImageNet: (a) Top-1 error rates, and (b) Top-5 error rates.



(a)



(b)



(c)

Fig. 3: Illustration of LST-I bottleneck *w.r.t.* the Inverted Residual bottleneck in MobileNet V2. (a): $\mathcal{E} = 1$; (b): $\mathcal{E} > 1$ while stride>1 or $C_{in} \neq C_{out}$; (c): $\mathcal{E} > 1$ while stride=1 and $C_{in} = C_{out}$. EWPlus means element-wise plus. PW-Conv/DWConv in red font indicates initialization with 2D-DCT while blue font suggests random initialization.