Supplementary Material for "The Heterogeneity Hypothesis: Finding Layer-Wise Differentiated Network Architectures"

Yawei Li[1], Wen Li[2], Martin Danelljan[1], Kai Zhang[1], Shuhang Gu[3], Luc Van Gool[1,4], Radu Timofte[1]
[1]Computer Vision Lab, ETH Zürich, [2]UESTC, [3]The University of Sydney, [4]KU Leuven
{yawei.li, martin.danelljan, kaizhang, vangool, radu.timofte}@vision.ee.ethz.ch
{liwenbnu,shuhanggu}@gmail.com

In this supplementary material, we first provide more details about the network shrinkage algorithm in Sec. 1. Then we describe the training protocols used for training networks on different tasks including image classification, visual tracking, and image restoration in Sec. 2. For a better understanding of the hypernetworks, the demo code is provided in Sec. 3. Finally, more experimental results are given in Sec. 4.

1. More Details

1.1. Handling different layers

In addition to the normal convolutional layers, the hypernetwork can be applied to other layers including depth-wise convolution and batch normalization. Special consideration needs to be made for depth-wise convolution. For depth-wise convolution, the dimensionality of the weight parameter along the input channel is 1. Thus, the latent vector controlling the input channel has only one element and shrinkage of this latent vector is avoided during the network shrinkage phase. For the normalization layers like BatchNorm, they are forced to have the same number of output channels as their preceding convolutional layers. The linear layers are pruned along with the preceding convolutional layer such that the same number of channels are removed.

1.2. Pruning according to gradients

The proximal gradient descent (PGD) algorithm in DHP [8] prunes the latent vectors according to the magnitude of their elements. Yet, the problem of PGD algorithm is that it might result in unbalanced pruning. For example, in Table 1, for the DHP results on MobileNetV2, the number of parameters increases despite the reduction of FLOPs. This is because larger percentage of channels in the lower layers are pruned, which accounts for more FLOPs but less parameters compared with those in the higher layers. On the contrary, at initialization of the hypernetwork, the range of the gradients of the latent vectors are relatively balanced across the layers. Thus, gradient magnitude of the latent vectors are chosen as the pruning criterion.

2. Training Protocol

The code is implemented in PyTorch [10]. For ImageNet experiments, the networks are trained with 4 Nvidia V100 GPUs. For the other experiments, the training is conducted on Nvidia TITAN Xp GPUs. The training protocols for different tasks are explained as follows.

2.1. Image classification

ImageNet

The ImageNet2012 dataset has 1000 classes. The training set contains 1.2 million images while the test set contains 50,000 image with 50 image for every class. Standard image normalization and data augmentation method are used. The training continues for 150 epochs. The initial learning rate is 0.05. Cosine learning rate decay is used. The weight decay factor is set to $4e^{-5}$. SGD optimizer is used during the training. The batch size is 256.

Tiny-ImageNet

Tiny-Imagenet is a simplified version of ImageNet2012. It has 200 classes. Each class has 500 training images and 50 validation images. And the resolution of the images is 64×64. The images are normalized with channel-wise mean and standard deviation. Horizontal flip is used to augment the dataset. The networks are trained for 220 epochs with SGD and an initial learning rate of 0.1. The learning rate is decayed by a factor of 10 at Epoch 200, Epoch 205, Epoch 210, and Epoch 215. The momentum of SGD is 0.9. Weight decay factor is set to 0.0001. The batch size is 64.

CIFAR

CIFAR [7] dataset contains two datasets *i.e.* CIFAR10 and CIFAR100. CIFAR10 contains 10 different classes. The training subset and testing subset of the the dataset contain 50,000 and 10,000 images with resolution 32×32, respectively. CIFAR100 is the same as CIFAR10 except that it has 100 classes. All of the images are normalized using

Dataset	Network	Method	Top-1 Error (%)	FLOPs [G] / Ratio (%)	Params [M] / Ratio (%)
ImageNet [2]	ResNet50 [3]	Baseline	23.28	4.1177 / 100.0	25.557 / 100.0
		MutualNet [16]	21.40	4.1177 / 100.0	25.557 / 100.0
		LW-DNA	23.00	3.7307 / 90.60	23.741 / 92.90
		MetaPruning [9]	23.80	3.0000 / 72.86	–
		AutoSlim [17]	24.00	3.0000 / 72.86	23.100 / 90.39
	RegNet [11] X-4.0GF	Baseline	23.05	4.0005 / 100.0	22.118 / 100.0
		LW-DNA	22.74	3.8199 / 95.49	15.285 / 69.10
	MobileNetV3 small [4]	Baseline	34.91	0.0612 / 100.0	3.108 / 100.0
		LW-DNA	34.84	0.0605 / 98.86	3.049 / 98.11
Tiny-ImageNet	MobileNetV1 [5]	Baseline	51.87	0.0478 / 100.0	3.412 / 100.0
		Baseline KD	48.00	0.0478 / 100.0	3.412 / 100.0
		DHP KD	46.70	0.0474 / 99.16	2.267 / 66.43
		LW-DNA	46.44	0.0460 / 96.23	1.265 / 37.08
	MobileNetV2 [13]	Baseline	44.38	0.0930 / 100.0	2.480 / 100.0
		Baseline KD	41.25	0.0930 / 100.0	2.480 / 100.0
		DHP KD	41.06	0.0896 / 96.34	2.662 / 107.34
		LW-DNA	40.74	0.0872 / 93.76	2.230 / 89.90
	MobileNetV3 [4] large	Baseline	45.53	0.0860 / 100.0	4.121 / 100.0
		Baseline KD	38.21	0.0860 / 100.0	4.121 / 100.0
		DHP KD	38.14	0.0856 / 99.53	3.561 / 86.42
		LW-DNA	37.45	0.0797 / 92.67	3.561 / 86.43
	MobileNetV3 [4] small	Baseline	47.55	0.0207 / 100.0	2.083 / 100.0
		Baseline KD	41.52	0.0207 / 100.0	2.083 / 100.0
		DHP KD	41.46	0.0192 / 92.75	1.078 / 51.76
		LW-DNA	41.35	0.0178 / 85.99	1.799 / 86.36
	MnasNet [14]	Baseline	51.79	0.0271 / 100.0	3.359 / 100.0
		Baseline KD	48.17	0.0271 / 100.0	3.359 / 100.0
		DHP KD	48.10	0.0264 / 97.42	2.512 / 74.79
		LW-DNA	46.85	0.0250 / 92.25	1.258 / 37.45
CIFAR100	RegNet [11] Y-200MF	Baseline	21.94	0.2259 / 100.0	2.831 / 100.0
		Baseline KD	19.87	0.2259 / 100.0	2.831 / 100.0
		LW-DNA	19.87	0.2095 / 92.74	1.524 / 53.85
	RegNet [11] Y-400MF	Baseline	21.65	0.4585 / 100.0	3.947 / 100.0
		Baseline KD	18.71	0.4585 / 100.0	3.947 / 100.0
		LW-DNA	18.65	0.4468 / 97.45	2.466 / 62.48
	RegNet [11] X-200MF	Baseline	23.62	0.2255 / 100.0	2.353 / 100.0
		Baseline KD	21.38	0.2255 / 100.0	2.353 / 100.0
		LW-DNA	21.19	0.2075 / 92.02	1.239 / 52.68
	RegNet [11] X-400MF	Baseline	21.75	0.4698 / 100.0	4.810 / 100.0
		Baseline KD	19.06	0.4698 / 100.0	4.810 / 100.0
		LW-DNA	18.81	0.4610 / 98.13	4.404 / 91.56
	EfficientNet [15]	Baseline	20.74	0.4161 / 100.0	4.136 / 100.0
		Baseline KD	19.73	0.4161 / 100.0	4.136 / 100.0
		LW-DNA	19.54	0.3850 / 92.53	2.121 / 51.28
	DenseNet40 [6]	Baseline	26.00	0.2901 / 100.0	1.100 / 100.0
		Baseline KD	22.84	0.2901 / 100.0	1.100 / 100.0
		LW-DNA	22.46	0.2638 / 90.93	1.016 / 92.35
CIFAR10	DenseNet40 [6]	Baseline	5.50	0.2901 / 100.0	1.059 / 100.0
		Baseline KD	4.88	0.2901 / 100.0	1.059 / 100.0
		LW-DNA	4.87	0.2632 / 90.73	0.963 / 90.87
	ResNet56 [3]	Baseline	5.74	0.1274 / 100.0	0.856 / 100.0
		Baseline KD	5.73	0.1274 / 100.0	0.856 / 100.0
		LW-DNA	5.49	0.1262 / 99.06	0.536 / 62.62

Table 1: Image classification results. Baseline and Baseline KD denote the original network trained without and with knowledge distillation respectively. DHP-KD is the DHP version trained with knowledge distillation.

channel-wise mean and standard deviation of the the training set [3, 6]. Standard data augmentation is also applied. Both of the baseline and the LW-DNA networks are trained for 300 epochs with SGD optimizer and an initial learning rate of 0.1. The learning rate is decayed by 10 after 50% and 75% of the epochs. The momentum of SGD is 0.9. Weight decay factor is set to 0.0001. The batch size is 64.

2.2. Visual tracking

For visual tracking, we follow the training protocol in [1]. To compare the baseline network and the LW-DNA models, the backbone network is initialized with the weights of ResNet50 and LW-DNA trained for this paper, respectively. Then the same training and testing protocol is applied. The results are denoted by DiMP-Baseline and DiMP-LW-DNA respectively.

2.3. Image restoration

Super-resolution
DIV2K dataset is used to train image super-resolution networks. The dataset contains 800 training images, 100 validation images, and 100 test images. The full resolution images are cropped into 480×480 subimages with overlap 240. There are 32208 subimages in total. For EDSR, the size of the extracted low-resolution input patch is 48×48 while for SRResNet the size is 24×24. The batch size is 16. Adam optimizer is used for the training. Default hyper-parameters are used for Adam optimizer. The weight decay factor is 0.0001. The networks are trained for 300 epochs. The learning rate starts from 0.0001 and decays by 10 after 200 epochs.

A simplified version of EDSR is used in order to speed up the training of EDSR. The original EDSR network contains 32 residual blocks and each convolutional layer has 256 channels. The simplified version has 8 residual blocks and with 128 channels for each convolutional layers.

Denoising
For image denoising, the images in DIV2K dataset are converted to gray images. For DnCNN the patch size of the input image is 64×64 and the batch size is 64. For UNet, the patch size and the batch size are 128×128 and 16, respectively. Gaussian noise is added to degrade the input patches on the fly with noise level $\sigma = 70$. Adam optimizer is used to train the network. The weight decay factor is 0.0001. The networks are trained for 60 epochs and each epoch contains 10,000 iterations. So in total, the training continues for 600k iterations. The learning rate starts with 0.0001 and decays by 10 at Epoch 40.

3. Demo code of hypernetworks

Listing 1: Demo code of the utilized hypernetworks.

```
import torch
z_o = torch.randn(n)
z_i = torch.randn(c)
w_1 = torch.randn(n, c, m)
w_2 = torch.randn(n, c, w*h, m)
o = torch.matmul(z_o.unsqueeze(-1),
        z_i.unsqueeze(0))
o = o.unsqueeze(-1) * w_1
o = torch.matmul(w_2, o.unsqueeze(-1))
```

For a better understanding, the demo code of the utilized hypernetworks is shown in the code Listing 1. The main part of code only contains 3 lines.

4. More Experimental Results

Full list of image classification results
Due to the lack of space, only a part of the results on image classification is shown in the main paper. The full list of image classification results is shown in Table 1. Fig. 1 shows more results on the training and testing log of different models. Fig. 2 shows the percentage of remaining channels of more LW-DNA models.

Denoising
Image denoising results are shown in Table 2. The identified LW-DNA models perform no worse than the baseline network with reduced number of parameters and FLOPs.

Ablation study on Tiny-ImageNet
An ablation study of the hyper-parameters ρ and τ is shown in Table 3. The experiments are conducted for MobileNetV1 on Tiny-ImageNet. The FLOPs budget is fixed for the experiments. Two conclusions can be drawn from the result. **I.** By increasing the hyper-parameters ρ and τ, the model complexity is also increased. And the accuracy of the network is also improved. **II.** All of the results in Table 3 are better than Baseline KD in Table 1, which shows the robustness of ρ and τ. Based on the experience on Tiny-ImageNet, we set $\rho = 0.4$ and $\tau = 0.45$ for ImageNet experiments. Quite surprising, this combination works well across the three investigated networks (ResNet50, RegNet, and MobileNetV3).

Distribution of latent vectors
The distribution of the latent vectors in MobileNetV2 during the DHP proximal gradient optimization is shown in Fig. 3. The distribution of the latent vectors at the end the optimization is related to the initial distribution to some extent. This phenomenon inspires us to pruning the latent vectors at initialization.

References

[1] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proc. ICCV*, pages 6182–6191, 2019. 3

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. CVPR*, pages 248–255. IEEE, 2009. 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778, 2016. 2

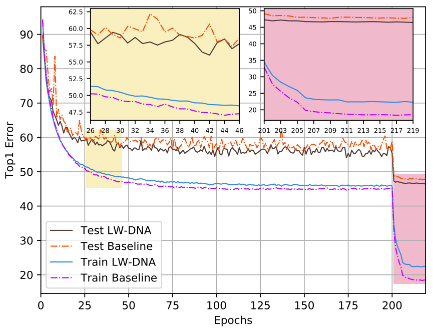
[4] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu,

Network	Method	PSNR $[dB]$		FLOPs $[G]$ / Ratio (%)	Params $[M]$ / Ratio (%)
		BSD68	DIV2K		
DnCNN [18]	Baseline	24.9	26.7	9.10 / 100.0	0.56 / 100.0
	LW-DNA	24.9	26.7	5.43 / 59.64	0.33 / 59.47
U-Net [12]	Baseline	25.2	27.2	3.41 / 100.0	7.76 / 100.0
	LW-DNA	25.2	27.2	3.26 / 95.60	5.86 / 75.57

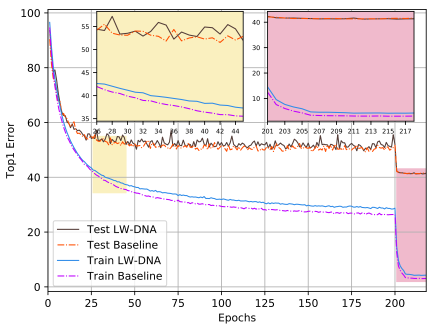
Table 2: Compression results on image denoising networks. The noise level σ is 70.

ρ	τ	Top-1	FLOPs [G]	Params [M]
0.1	0.4	47.02	0.046	0.948
0.1	0.45	46.66	0.046	0.986
0.2	0.4	46.94	0.0459	1.210
0.2	0.45	46.44	0.046	1.265

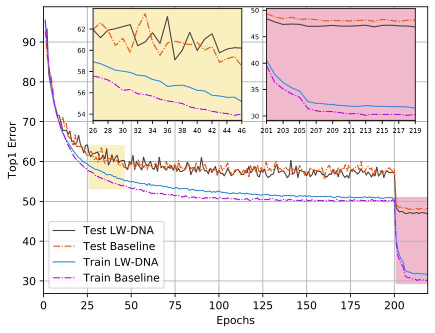
Table 3: Ablation study of the hyper-parameters ρ and τ on MobileNetV1.



(a) MobileNetV2.



(b) MobileNetV3 small.



(c) MNASNet.

Figure 1: Training and testing log of the LW-DNA models and the baseline models.

Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proc. ICCV*, pages 1314–1324, 2019. 2

[5] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2

[6] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. CVPR*, pages 2261–2269, 2017. 2

[7] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 1

[8] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. DHP: Differentiable meta pruning via hypernetworks. *arXiv preprint arXiv:2003.13683*, 2020. 1

[9] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Tim Kwang-Ting Cheng, and Jian Sun. MetaPruning: Meta learning for automatic neural network channel pruning. In *Proc. ICCV*, 2019. 2

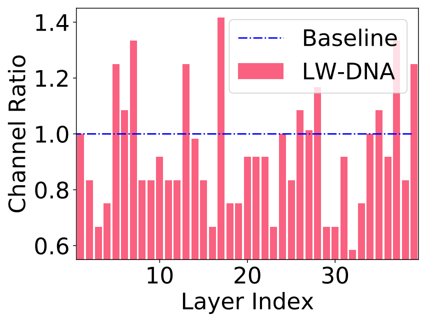
[10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017. 1

[11] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. *arXiv preprint arXiv:2003.13678*, 2020. 2

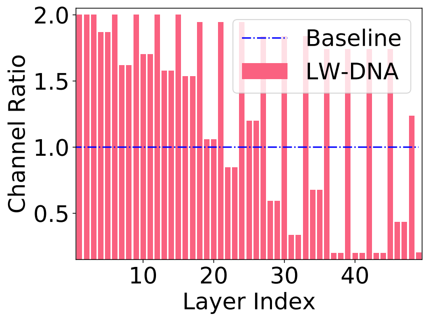
[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, pages 234–241. Springer, 2015. 4

[13] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proc. CVPR*, pages 4510–4520, 2018. 2

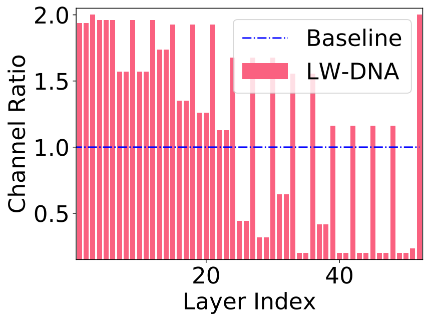
[14] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proc. CVPR*, pages 2820–2828, 2019. 2



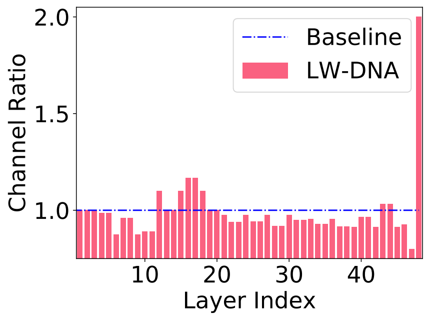
(a) DenseNet, CIFAR100.



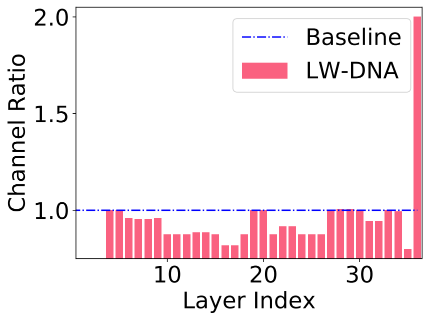
(b) EfficientNet, Tiny-ImageNet.



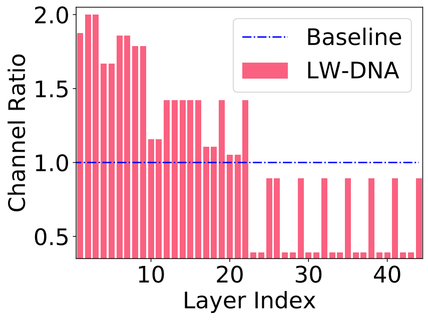
(c) MNASNet, Tiny-ImageNet.



(d) MobileNetV3-large, Tiny-ImageNet.



(e) MobileNetV3-small, Tiny-ImageNet.



(f) RegNet 200MF, Tiny-ImageNet.

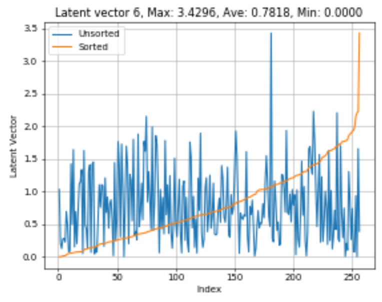
Figure 2: Percentage of remaining output channels of LW-DNA models over the baseline network

[15] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 2

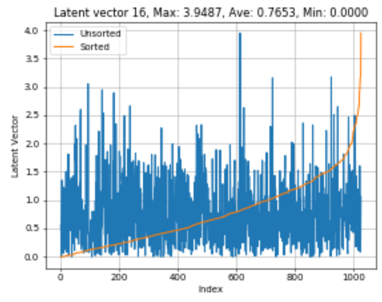
[16] Taojiannan Yang, Sijie Zhu, Chen Chen, Shen Yan, Mi Zhang, and Andrew Willis. MutualNet: Adaptive convnet via mutual learning from network width and resolution. 2020. 2

[17] Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019. 2

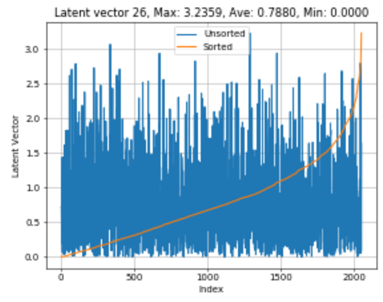
[18] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. *IEEE TIP*, 26(7):3142–3155, 2017. 4



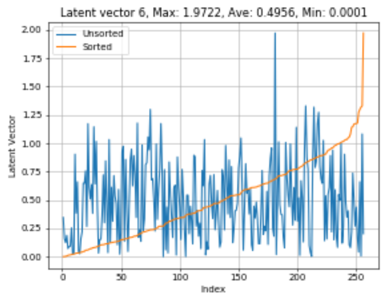
Layer 6, Epoch 1



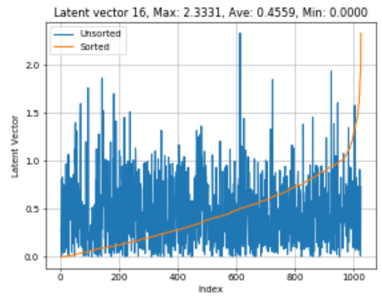
Layer 16, Epoch 1



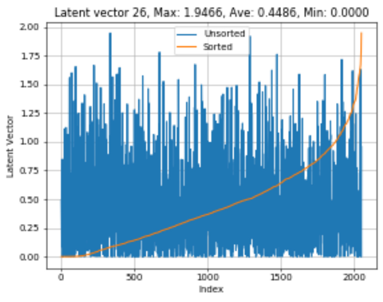
Layer 26, Epoch 1



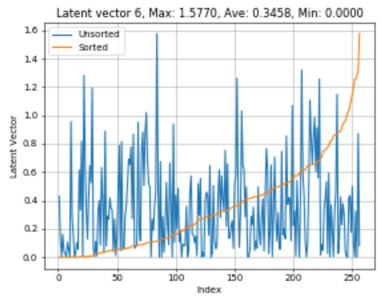
Layer 6, Epoch 4



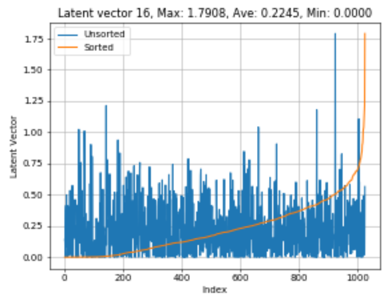
Layer 16, Epoch 4



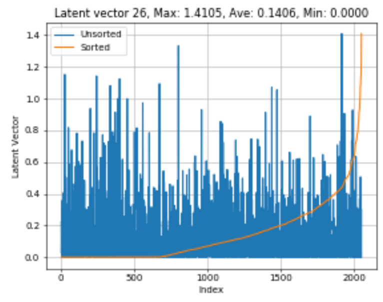
Layer 26, Epoch 4



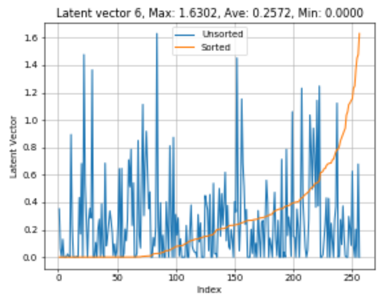
Layer 6, Epoch 10



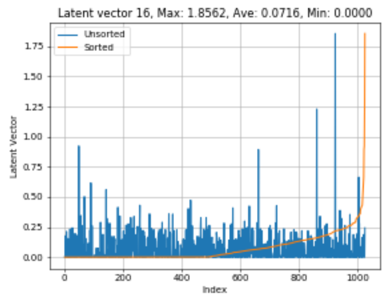
Layer 16, Epoch 10



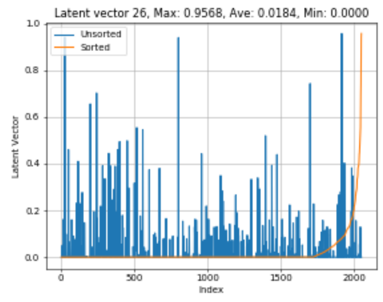
Layer 26, Epoch 10



Layer 6, Epoch 17



Layer 16, Epoch 17



Layer 26, Epoch 17

Figure 3: The distribution of the latent vectors in MobileNetV2 during the proximal gradient optimization of DHP.