Supplementary Materials for "Dynamic Hierarchical Mimicking Towards Consistent Optimization Objectives"

Duo Li Qifeng Chen
The Hong Kong University of Science and Technology
{duo.li@connect., cqf@}ust.hk

A. Architectural Design of Auxiliary Classifiers

Following descriptions in the main paper, we always attach two auxiliary branches on top of certain intermediate layers of the backbone networks. For brevity of clarification, we denote the main branch as $\mathcal{B}_0$ and the auxiliary branch close to (away from) the top-most classifier as $\mathcal{B}_1$ ($\mathcal{B}_2$). In the architecture engineering process, we heuristically follow three principles below: **(i)** building blocks in the auxiliary branches are the same as those in the original main branch for architectural identity; **(ii)** from the common input to the end of every branch, number of layers for down-sampling are kept the same, guaranteeing the uninterrupted coarse-to-fine information flow; **(iii)** $\mathcal{B}_1$ with broader pathway and $\mathcal{B}_2$ with shorter pathway are preferable in our design.

A.1. Various Networks on the CIFAR-100 dataset

We append two auxiliary branches to different popular networks with varied depths. Refer to Table 1, 2 and 3 for detailed architectural design of these auxiliary branches in ResNet [1], DenseNet [3] and WRN [4] respectively.

A.2. ResNet on the ImageNet dataset

We also append two auxiliary branches to certain locations of the ResNet [1] backbone for main experiments on the ImageNet dataset. For ablation study we further take into consideration a third branch connected to a shallower intermediate layer in ResNet-18 which is called $\mathcal{B}_3$ in accordance with the order of the subscript. Refer to Table 4 for full configurations including specific number of residual blocks and number of channels in each building block.

A.3. MobileNet on Re-ID datasets

For MobileNet used on the Re-ID tasks, we fork two auxiliary branches from the network stem, consisting of depthwise separable convolutions resembling the basic modules in the backbone. Refer to Table 5 for architectural details of both main and auxiliary branches.

B. Training Curves on the ImageNet dataset

We attach the training curves of representative ResNet-101 and ResNet-152 on ImageNet, as illustrated in Figure 1. Very deep ResNets with tens of millions of parameters are prone to over-fitting. We note that through our proposed Dynamic Hierarchical Mimicking, the training accuracy curve tends to be lower than both the plain one and Deeply Supervised Learning, but our methodology leads to substantial gain in the validation accuracy compared to the other two. We infer that our training scheme implicitly achieves strong regularization effect to enhance the generalization ability of deep convolutional neural networks.

C. Implicit Penalty on Inconsistent Gradients

The derivation process of Equation 11 in the main paper is presented here in detail. Similar analysis could be conducted on the paired branch ϕ_2.

$$
\begin{aligned}
& -\mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{\phi}_2^{(k)}(\boldsymbol{\theta}(\boldsymbol{x})+\boldsymbol{\xi})\log\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x})+\boldsymbol{\xi})] \\
= \;& -\mathbb{E}_{\boldsymbol{\xi}}[(\boldsymbol{\phi}_2^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))+\boldsymbol{\xi}^\top\nabla_{\boldsymbol{\theta}}\phi_2^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))+o(\sigma^2)) \\
& \qquad (\log\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))+\boldsymbol{\xi}^\top\frac{\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))}{\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))}+o(\sigma^2))] \\
& \qquad\qquad (\textit{Taylor Expansion}) \\
= \;& -[\boldsymbol{\phi}_2^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))\log\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x})) \\
& \quad +\mathbb{E}_{\boldsymbol{\xi}}(\log\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))\boldsymbol{\xi}^\top\nabla_{\boldsymbol{\theta}}\phi_2^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))) \\
& \quad +\mathbb{E}_{\boldsymbol{\xi}}(\boldsymbol{\phi}_2^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))\boldsymbol{\xi}^\top\frac{\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))}{\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))}) \\
& \quad +\sigma^2\frac{\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_2^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))}{\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))}+o(\sigma^2)] \\
\approx \;& -\boldsymbol{\phi}_2^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))\log\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x})) \\
& \quad -\sigma^2\frac{\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_2^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))\nabla_{\boldsymbol{\theta}}\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))}{\boldsymbol{\phi}_1^{(k)}(\boldsymbol{\theta}(\boldsymbol{x}))} \\
& \qquad\qquad (\textit{Note that } \mathbb{E}_{\boldsymbol{\xi}}\boldsymbol{\xi}^\top=0)
\end{aligned}
$$

layer name	output size	ResNet-32			ResNet-110			ResNet-1202		
		$\mathcal{B}_0$	$\mathcal{B}_2$	$\mathcal{B}_1$	$\mathcal{B}_0$	$\mathcal{B}_2$	$\mathcal{B}_1$	$\mathcal{B}_0$	$\mathcal{B}_2$	$\mathcal{B}_1$
conv1	32×32	3×3, 16			3×3, 16			3×3, 16		
conv2_x	32×32	$\begin{bmatrix} 3\times3, 16 \\ 3\times3, 16 \end{bmatrix}\times5$			$\begin{bmatrix} 3\times3, 16 \\ 3\times3, 16 \end{bmatrix}\times18$			$\begin{bmatrix} 3\times3, 16 \\ 3\times3, 16 \end{bmatrix}\times200$		
conv3_x	16×16	$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix}\times5$	$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix}\times5$		$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix}\times18$	$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix}\times9$		$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix}\times200$	$\begin{bmatrix} 3\times3, 32 \\ 3\times3, 32 \end{bmatrix}\times100$	
conv4_x	8×8	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times5$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times5$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times18$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times9$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times18$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times200$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times100$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times200$
classifier	1×1	average pool, 100-d fc, softmax								

Table 1: Architectures of the ResNet family with auxiliary branches for CIFAR-100. Residual blocks are shown in brackets with the numbers of blocks stacked. Downsampling is performed by conv3_1 and conv4_1 with a stride of 2.

layer name	output size	DenseNet (k=40, d=12)			DenseNet (k=100, d=12)		
		$\mathcal{B}_0$	$\mathcal{B}_2$	$\mathcal{B}_1$	$\mathcal{B}_0$	$\mathcal{B}_2$	$\mathcal{B}_1$
conv1	32×32	3×3, 2k			3×3, 2k		
conv2_x	32×32	[3×3, k] × 12			[3×3, k] × 32		
conv3_x	16×16	[3×3, k] × 12	[3×3, k] × 12		[3×3, k] × 32	[3×3, k] × 16	
conv4_x	8×8	[3×3, k] × 12	[3×3, k] × 6	[3×3, 3k] × 12	[3×3, k] × 32	[3×3, k] × 16	[3×3, 3k] × 32
classifier	1×1	average pool, 100-d fc, softmax					

Table 2: Architectures of the DenseNet family with auxiliary branches for CIFAR-100. Dense blocks are shown in brackets with the numbers of blocks stacked. Downsampling is performed by transition layers inserted between conv2_x, conv3_x and conv4_x with a stride of 2.

D. Effect of Bernoulli Sampling

In the main experiments, we keep using auxiliary classifiers forked from certain locations of the backbone network with a binary sampling strategy. Here as a justification for more complicated stochastic sampling methods, we use the CIFAR-100 dataset and the shallow ResNet-32 model as the test case. We maintain the original settings relevant to structures of auxiliary classifiers and collect cross-entropy losses from all of these classifiers. Then we stochastically discard some of these auxiliary branches depending on i.i.d. samples drawn from a multivariate Bernoulli distribution (each variate is associated with one auxiliary branch) with the probability of 0.5 when calculating mimicking losses at each training epoch. With the stochastically activated branches for interaction, much stronger regularization effect is achieved even using this small network. The ResNet-32 model trained with this Bernoulli sampling policy outperforms all of its counterparts in Table 1 of the main paper with the 27.002 ± 0.316 (mean $\pm$ std.) top-1 error.

E. Experiments on Corrupt Data

We further explore the flexibility of our method when applied to corrupt data [5], *i.e*. part of ground truth labels in the dataset are replaced with random labels. The best-performing WRN-28-10 architecture among our spectrum of experiments on CIFAR-100 is utilized as the testbed. We toggle the ratio of corruption from 0.2 to 0.5 and observe the corresponding performance change. When 20% training labels are corrupt, top-1 accuracy of the baseline model drops nearly 10 percent to 71.122 ± 0.269, while with our proposed training mechanism the trained model still struggles to preserve an accuracy of 74.528 ± 0.433, which is a more remarkable margin noticing that the performance improvement on clean data is just around 2%. Along with the corrupt ratio increasing to 50%, the performance of baseline model drops another 10 percent to 61.268 ± 0.311 while ours is 64.226 ± 0.300, maintaining a margin of around 3%. From Figure 2, we observe that training accuracy approximates to 100% even on corrupt data while the validation accuracy suffers a sharp decline which implies severe over-fitting problems. Intriguingly, our proposed hierarchical mimicking training mechanism achieves larger margin in this corrupt setting, demonstrating its powerful regularization effect of suppressing the random label disturbance.

F. Experiments Using WRN with Dropout

Reminiscent of the regularization efficiency of dropout layers in Wide Residual Networks [4], we extent our experiments on CIFAR-100 to WRN-28-10 equipped with dropout. There exists an evident decrease in top-1 error to 18.698 ± 0.154 compared with vanilla WRN-28-10. We apply our hierarchical mimicking method to the training procedure of WRN-28-10 (dropout=0.3), resulting in a further improvement by decreasing the top-1 error to 16.790 ± 0.110. We can conclude that our proposed method has no counteractive effect on previous popular regularization techniques, *e.g*. dropout and is complementary to them

layer name	output size	WRN-16-8			WRN-28-10		
		$\mathcal{B}_0$	$\mathcal{B}_2$	$\mathcal{B}_1$	$\mathcal{B}_0$	$\mathcal{B}_2$	$\mathcal{B}_1$
conv1	32×32	3×3, 16			3×3, 16		
conv2_x	32×32	[3×3, 16k; 3×3, 16k] ×2			[3×3, 16k; 3×3, 16k] ×4		
conv3_x	16×16	[3×3, 32k; 3×3, 32k] ×2	[3×3, 32k; 3×3, 32k] ×2		[3×3, 32k; 3×3, 32k] ×4	[3×3, 32k; 3×3, 32k] ×4	
conv4_x	8×8	[3×3, 64k; 3×3, 64k] ×2	[3×3, 64k; 3×3, 64k] ×1	[3×3, 128k; 3×3, 128k] ×2	[3×3, 64k; 3×3, 64k] ×4	[3×3, 64k; 3×3, 64k] ×2	[3×3, 128k; 3×3, 128k] ×4
classifier	1×1	average pool, 100-d fc, softmax					

Table 3: Architectures of the Wide Residual Network family with auxiliary branches for CIFAR-100. Residual blocks are shown in brackets with the numbers of blocks stacked. Downsampling is performed by conv3_1 and conv4_1 with a stride of 2.

layer name	output size	18-layer				50-layer			101-layer			152-layer		
		$\mathcal{B}_0$	$\mathcal{B}_3$	$\mathcal{B}_2$	$\mathcal{B}_1$	$\mathcal{B}_0$	$\mathcal{B}_2$	$\mathcal{B}_1$	$\mathcal{B}_0$	$\mathcal{B}_2$	$\mathcal{B}_1$	$\mathcal{B}_0$	$\mathcal{B}_2$	$\mathcal{B}_1$
conv1	112×112	7×7, 64, stride 2												
conv2_x	56×56	3×3 max pool, stride 2												
		[3×3, 64; 3×3, 64] ×2				[1×1, 64; 3×3, 64; 1×1, 256] ×3			[1×1, 64; 3×3, 64; 1×1, 256] ×3			[1×1, 64; 3×3, 64; 1×1, 256] ×3		
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×1			[1×1, 128; 3×3, 128; 1×1, 512] ×4			[1×1, 128; 3×3, 128; 1×1, 512] ×4			[1×1, 128; 3×3, 128; 1×1, 512] ×8		
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×1	[3×3, 256; 3×3, 256] ×1		[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×3		[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×12		[1×1, 256; 3×3, 256; 1×1, 1024] ×36	[1×1, 256; 3×3, 256; 1×1, 1024] ×18	
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×2	[3×3, 1024; 3×3, 1024] ×2	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×2	[1×1, 1024; 3×3, 1024; 1×1, 4096] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×2	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×2	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
classifier	1×1	average pool, 1000-d fc, softmax												

Table 4: Architectures of the ResNet family with auxiliary branches for ImageNet. Residual blocks are shown in brackets with the numbers of blocks stacked. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

towards achieving higher accuracy with powerful CNNs.

G. Comparison to Knowledge Transfer Research

Our knowledge matching loss is partially inspired by the line of Knowledge Transfer (KT) research but we shift its primary focus away from model compression in the conventional KT methods. The representative Dark Knowledge Distillation [2] requires a large teacher model to aid the optimization process of a small student model via offering informative hint in the form of probabilistic prediction output as the soft label. In this framework, aiming at easing the optimization difficulty of small networks, an available strong model is required beforehand. In contrast, we concentrate on developing deeply supervised training scheme and further boosting the optimization process of state-of-the-art CNNs instead of compact models. Moreover, unlike the teacher and student in the distillation procedure which are optimized sequentially without *straightforward* association during their separate training process, our training strategy drives all auxiliary branch classifiers together with the original classifier to be optimized simultaneously with a knowledge matching loss among them computed in an on-the-fly manner. Knowledge transfer process occurs in a more compact way within our proposed mechanism, which enables knowledge sharing across hierarchical layers in one single network, without the demand of an extra teacher model. Thus our knowledge integration learning scheme is ready to be deployed in the optimization process of any convolutional neural networks, both lightweight networks and heavy ones.

H. Visualization of Improved Representation Consistency

To visualize the improved intermediate features for demonstration, We select the side branch $\mathcal{B}_2$ and the main branch $\mathcal{B}_0$ of the ResNet-152 model, take the maximum from each 3×3 kernel of the middle layer in the residual blocks and normalize them across channels and filters. Then the correlation matrices are calculated between the corresponding convolutional layers from these two branches. Some representative comparisons are illustrated in Figure 3, in which our proposed method leads to clearly higher correlation values.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[2] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

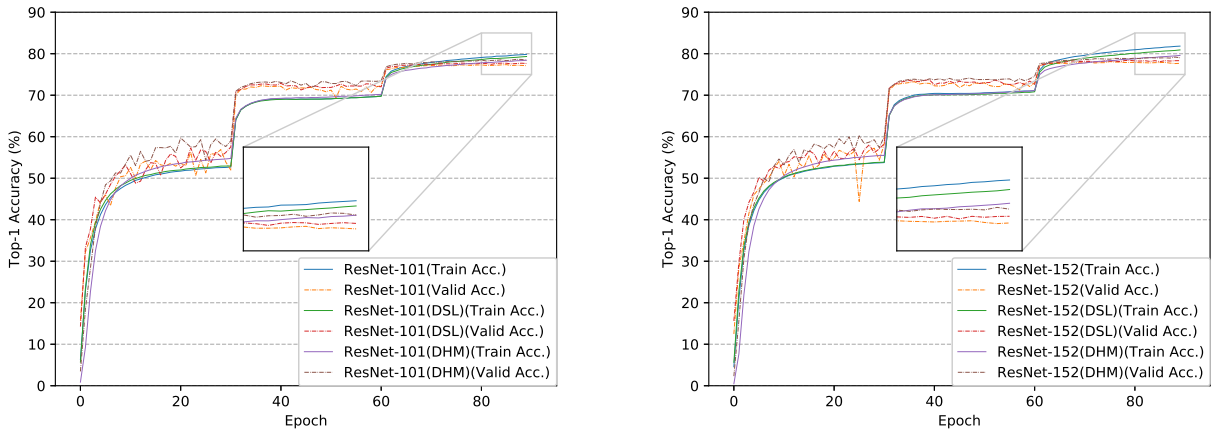


Figure 1: Curves of top-1 training (solid lines) and validation (dash lines) accuracy of ResNet-101 (left) and ResNet-152 (right) on the ImageNet dataset trained with different mechanism. The zoomed-in region shows that the model trained with our DHM method achieves the lowest training accuracy but the highest validation accuracy. Best viewed in color.

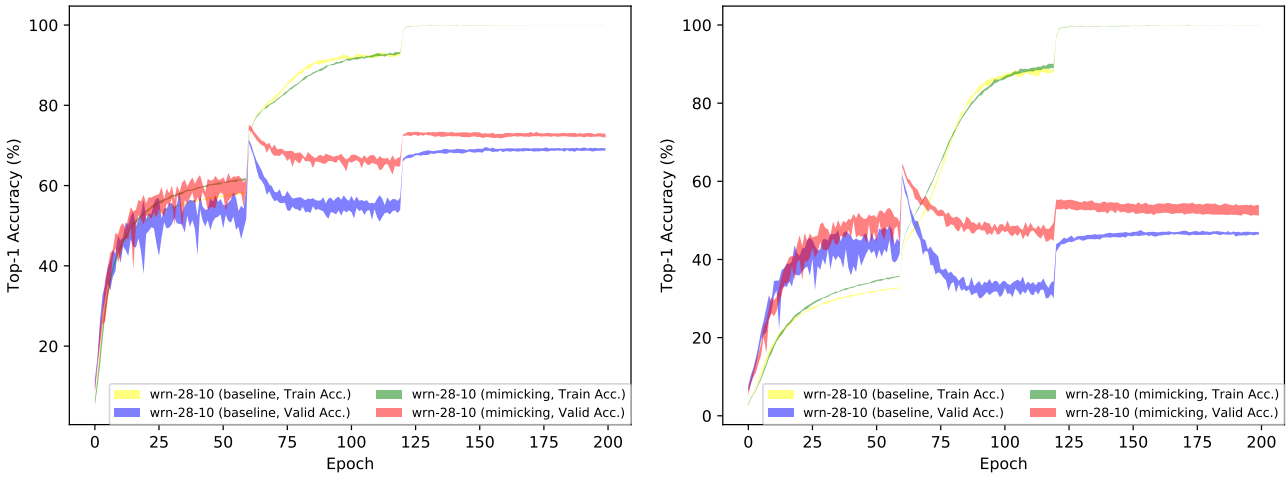
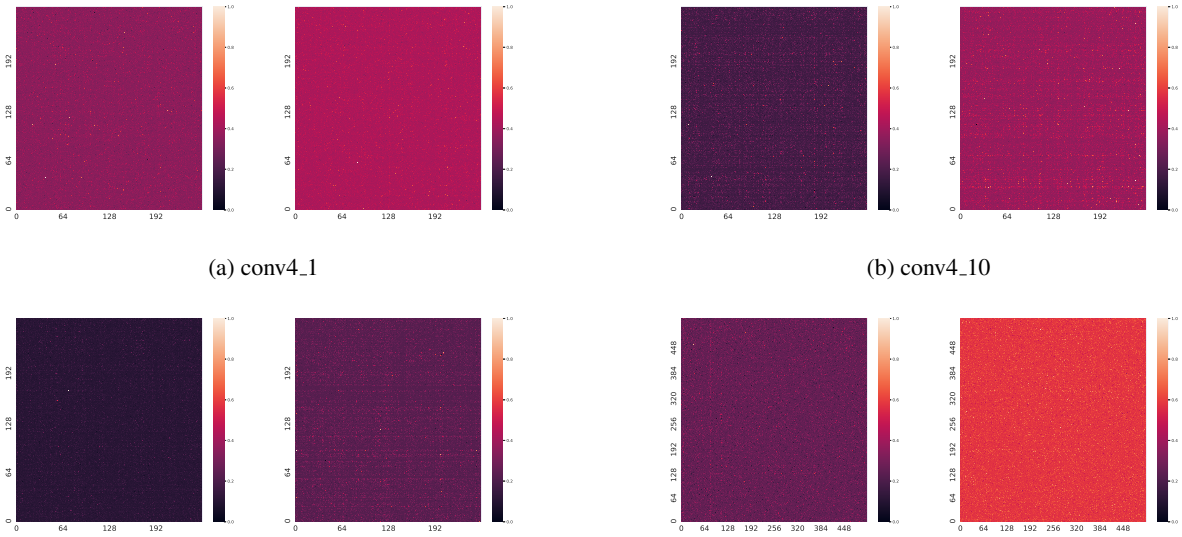


Figure 2: Curves of top-1 training and validation accuracy of WRN-28-10 on corrupt CIFAR-100 dataset with different training mechanism. 'baseline' denotes plain optimization scheme without auxiliary branches, 'mimicking' denotes our proposed methodology. The sub-figure in the left is obtained with the corresponding networks evaluated on the CIFAR-100 training set with a corrupt ratio of 0.2 while the one in the right with a corrupt ratio of 0.5. Results are bounded by the range of 5 successive runs. Best viewed in color.

[3] G. Huang, Z. Liu, L. v. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[4] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[5] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.



(a) conv4_1

(b) conv4_10

(c) conv4_17

(d) conv5_2

Figure 3: Correlation heatmaps of conv4_1, conv4_10, conv4_17 and conv5_2 in the ResNet-152 model. In each sub-figure, the left panel shows the result corresponding to the model trained through Deeply Supervised Learning, while the right panel shows the result corresponding to the model trained with our proposed Dynamic Hierarchical Mimicking strategy. The x-axis and y-axis represents input and output channel indices of a convolutional layer respectively.

$\mathcal{B}_0$	$\mathcal{B}_2$	$\mathcal{B}_1$
Conv(3, 32) / s2		
Conv(3, 32) dw / s1		
Conv(1, 64) / s1		
Conv(3, 64) dw / s2		
Conv(1, 128) / s1		
Conv(3, 128) dw / s1		
Conv(1, 128) / s1		
Conv(3, 128) dw / s2		
Conv(1, 256) / s1		
Conv(3, 256) dw / s1		
Conv(1, 256) / s1		
Conv(3, 256) dw / s2	Conv(3, 256) dw / s2	
Conv(1, 256) / s1	Conv(1, 256) / s1	
$5\times$ Conv(3, 512) dw / s1 Conv(1, 512) / s1	$3\times$ Conv(3, 512) dw / s1 Conv(1, 512) / s1	
Conv(3, 512) dw / s2	Conv(3, 512) dw / s2	Conv(3, 512) dw / s2
Conv(1, 1024) / s1	Conv(1, 1024) / s1	Conv(1, 2048) / s1
Conv(3, 1024) dw / s2	Conv(3, 1024) dw / s2	Conv(3, 2048) dw / s2
Conv(1, 1024) / s1	Conv(1, 1024) / s1	Conv(1, 2048) / s1
Avg Pool 7×7 / s1	Avg Pool 7×7 / s1	Avg Pool 7×7 / s1
FC 1024×1000 / s1	FC 1024×1000 / s1	FC 2048×1000 / s1
Softmax Classifier / s1	Softmax Classifier / s1	Softmax Classifier / s1

Table 5: Architecture of the MobileNet body with auxiliary branches used in person re-identification tasks. Conv(k, c) denotes convolutional filters with kernel size k and output channel c, 'dw' denotes depthwise convolution, s1 and s2 specify the stride in the corresponding layer.