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

Abstract

While the depth of modern Convolutional Neural Networks (CNNs) surpasses that of the pioneering networks with a significant margin, the traditional way of appending supervision only over the final classifier and progressively propagating gradient flow upstream remains the training mainstay. Seminal Deeply-Supervised Networks (DSN) were proposed to alleviate the difficulty of optimization arising from gradient flow through a long chain. However, it is still vulnerable to issues including interference to the hierarchical representation generation process and inconsistent optimization objectives, as illustrated theoretically and empirically in this paper. Complementary to previous training strategies, we propose Dynamic Hierarchical Mimicking, a generic feature learning mechanism, to advance CNN training with enhanced generalization ability. Partially inspired by DSN, we fork delicately designed side branches from the intermediate layers of a given neural network. Each branch can emerge from certain locations of the main branch dynamically, which not only retains representation rooted in the backbone network but also generates more diverse representations along its own pathway. We go one step further to promote multi-level interactions among different branches through an optimization formula with probabilistic prediction matching losses, thus guaranteeing a more robust optimization process and better representation ability. Experiments on both category and instance recognition tasks demonstrate the substantial improvements of our proposed method over its corresponding counterparts using diverse state-of-the-art CNN architectures. Code and models are publicly available at `https://github.com/d-li14/DHM`.

1. Introduction

Convolutional neural networks (CNNs) have become the mainstream models for tackling a wide array of computer vision problems such as image classification [3, 33, 8, 14], object detection [30, 29, 23] and semantic image segmentation [24, 43, 1]. The advent of AlexNet [18] that achieved groundbreaking results in the ImageNet Large Scale Visual Recognition Challenge 2012 [3] ignites the resurgence of deep CNN architectures. Recently along with the growing abundance of computational resources and development frameworks, a remarkable trend of modern CNN architectures is more and more convolutional layers are stacked upon, interweaved with indispensable non-linear activation layers and down-sampling layers. CNNs are now capable of mining intrinsic characteristics of the images as superhuman image descriptors with tens of thousands of parameters and engineered innovative connection topology [8, 14, 41]. Embracing these sophisticated CNNs as modeling tools, the past years witnessed an unprecedented achievement on a variety of visual recognition competitions [3, 5, 22]. However within the very deep network architectures, inappropriately designed blocks would impede the gradient flow across layers, consequently causing critical gradient vanishing or parameter redundancy problems [15, 14].

The aforementioned nuisances motivate us to ease the CNN model training and enhance generalization ability. One promising line of exploration lays emphasis on the intermediate feature representation and hidden layer supervision. In the Inception series [36, 37], auxiliary classifiers are connected to two intermediate layers. Although joint optimization of the weighted auxiliary probabilistic prediction together with the original one can combat the gradient vanishing problem as expected, gain in model performance is relatively minor (around 0.5% [36] or 0.4% [37]) as announced by the authors. Another contemporary work is DSN [20], which stacks a supervised MLP with very simple auxiliary classifiers, *i.e.*, SVM or Softmax on top of each hidden layer of the deep architecture. As [32] suggests, imposing a very discriminative hint for classification on intermediate layers might be too aggressive to achieve promising performance at the top-most classifier. Analogously, more recent MSDNet [13] empirically manifests that introducing early-exit classifiers to the intermediate layers will lead to accuracy degradation of the final classifier and assumes the phenomena is attributed to collapse of the progressive bottom-up feature generation process.

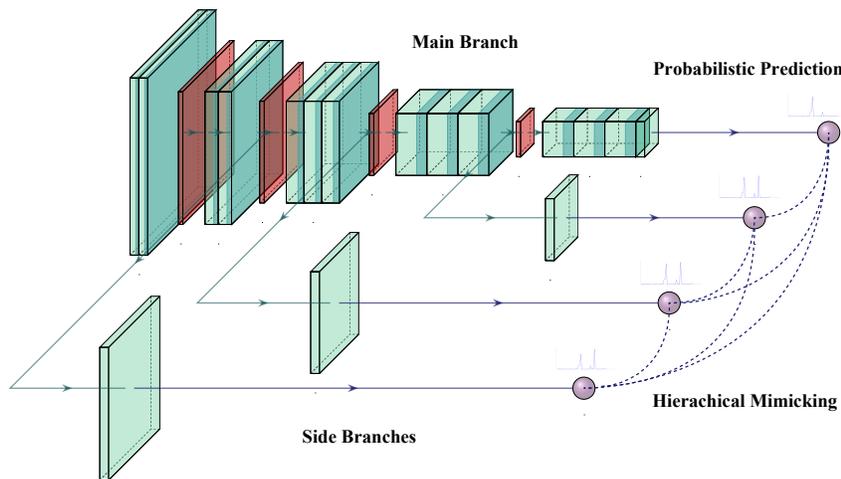Our reasoning arises from two critical standpoints pri-

Figure 1. Illustration of the Dynamic Hierarchical Mimicking mechanism. The proposed framework attaches three side branches to the main branch. In these branches, the green layers represent standard convolutional layers while the red ones represent downsampling layers. The purple dots at the end of existing classifiers represent the probabilistic prediction outputs. Bidirectional dashed lines densely connected to these dots represent the knowledge transfer process through pair-wise soft label mimicking. Best viewed in color.

marily. First, hierarchical root locations would endow different auxiliary classifiers with the ability to capture prediction representation with much more diversity, without interfering with the information flow of the main branch as long as the corresponding classifier is delicately architectural-engineered according to the location of the intermediate layer to which it attaches. Nevertheless, it brings limited benefit to improving model generalization and accuracy in a previously prevailing joint optimization scheme used in [20, 36]. Second, we blame the barrier to improving model generalization ability and accuracy on the insufficient collaboration of recognition knowledge extracted from diverse stages. It may become difficult that optimization directions of different auxiliary branches conform with each other, thus the gradients flowing upstream to their common stem can counteract and little positive optimization effects are imposed on the parameters of the most shallow layers shared by them all. In other words, within the training dynamics, potentially inconsistent optimization sub-objectives of different auxiliary classifiers can give rise to a suboptimal solution of the whole neural network.

Motivated by the issues above, we propose Dynamic Hierarchical Mimicking (DHM), a generic training framework amenable to any state-of-the-art CNN models, which noticeably improves the performance on supervised visual recognition tasks compared with the standard top-most supervised training as well as the deeply supervised training scheme. As illustrated in Figure 1, our mechanism is comprised of two components advancing the training procedure collaboratively. On the one hand, we attach carefully designed auxiliary branches to some intermediate nodes of the backbone network. For side branches, locations of corresponding root nodes are sampled from certain distributions (*e.g.* uniform discrete distribution). In principle, diverse auxiliary towers both inherit multi-level knowledge from

the backbone network and advance the inherited knowledge through staked modules to generate more diverse representation. On the other hand, with differentiable hierarchical predictions (*i.e.* probabilistic distribution output over training examples) extracted within a single CNN model at hand, it is naturally expected to enable comprehensive knowledge interactions. To this end, we propose a novel joint optimization formula containing a pairwise probabilistic distribution matching loss utilized between any two branches. This additional loss function enhances the opportunity of knowledge sharing and forces the optimization consistency across the whole network. Notably, we focus on improving the training procedure and discard all the side branches during inference, without introducing any computational overheads compared to the standard inference.

We evaluate our method on two challenging image classification datasets, including the large-scale ImageNet benchmark, as well as two widely-accepted person re-identification datasets using state-of-the-art CNN architectures. The presented results indicate that the deep CNN models trained with our proposed mechanism have significant accuracy and generalization ability improvements against their counterpart baseline models.

2. Related Work

We review some related approaches in prior literature, from which our method draws inspiration. We also analyze their differences from our mechanism.

Auxiliary Supervision. To accelerate convergence and combat gradient vanishing problem, supervision signals are delivered to hidden layers more straightforwardly through auxiliary towers built on top of these intermediate layers. Two concurrent works utilizing this advanced supervision methodology are GoogLeNet [36] and DSN [20] which are

benchmarked with primitive deep neural networks on the fundamental image classification tasks. Once published, this idea has been extended to various application fields to address edge detection [40], pose estimation [26], scene parsing [51], semantic segmentation [50] and other visual recognition tasks [21, 47, 25]. We conjecture that there still exists room for performance improvement through encouraging explicit knowledge interactions between each sampled pair of auxiliary classifiers and demonstrate the aforementioned conjecture with our proposed DHM training strategy.

Network Regularization. With the drive to suppress the over-fitting issue, Dropout [35] multiplies each hidden activation of a layer by a Bernoulli random variable during training which effectively impels hidden nodes to learn independent representations. Follow-up works validate the advantage of this idea, such as MaxOut [7], Drop-Connect [38], DropIn [34] and DropBlock [6]. Batch Normalization [16] unifies the layer-wise activation distribution to zero mean and unit variance which mitigates the need of Dropout in modern CNNs like ResNet and DenseNet. Stochastic Depth [15] shifts the focus from micro-architecture to macro- by stochastically discarding entire blocks to improve network resilience. FractalNet [19] develops Drop Path in a parallel fashion to discourage co-adaption of subnetworks in a group. Furthermore, over-fitting problems can be partially attributed to the inadequate available source of large-scale data. Data transformation techniques are widely applied to synthetically augment the original datasets, *e.g.*, reflectional padding, horizontal flipping, random cropping, color jittering, and linear interpolation [18, 36, 10, 4, 46]. Our method serves as an effective supplement to these existing methods, which behaves like a strong regularizer during the training process.

Knowledge Transfer. Our method also has a connection with the research field of Knowledge Transfer (KT). Top-performing deep CNN models suffering from intensive computational demands are hindered from being embedded into resource-aware applications. To narrow the gap between theoretical performance and real-world feasibility, Dark Knowledge Distillation [9] takes the probabilistic distribution prediction from a powerful but resource-hungry teacher model or an ensemble of teacher models as the soft target, to jointly regularize the optimization objective when training a smaller student model with given image samples and the corresponding one-hot labels. Intermediate feature maps are demonstrated to be effective hints to further advance the knowledge distillation process [32, 42, 45]. Extending the concept of knowledge distillation and its variants, Deep Mutual Learning [49] shows that the teacher model would benefit from the knowledge of the student model in turn, in contrast to the prevailing one-way teaching-learning mode. The newly established

idea was soon used in person re-identification tasks [49, 48]. Different from the method above in focus and formulation, our motivation is to solve the inherent deficiency hidden in the deeply-supervised training procedure, utilizing representation mimicking as a tool. Our proposed Dynamic Hierarchical Mimicking can be viewed as an internal knowledge transfer process limited in one single neural network among hierarchical auxiliary classifiers, which has never attracted enough attention from the research community. We also include a more comprehensive analysis of the differences between our method and KT in the supplementary materials.

3. Approach

In this section, we shed light on the intrinsic deficiency within the traditional hidden layer supervision and joint optimization scheme [20, 36]. Furthermore, we elaborate on the improved optimization objective of our proposed mechanism, highlighting its intuition and theoretical insights.

3.1. Analysis of Deep Supervision

Given a fully annotated dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \mid i = 1, 2, \ldots, N\}$ including N examples collected from K predefined classes where $\boldsymbol{x}_i$ is the ith training example and $y_i \in \{1, 2, \ldots, K\}$ is the corresponding ground truth label. Let $\boldsymbol{W}_m$ (m in subscript denotes main branch) be the learnable weight matrices of an L-layer deep neural network $\mathcal{N}$, $\boldsymbol{W}_m^l$ be the part of weights from the bottom layer up to the lth layer and $f_m(\boldsymbol{W}_m; \boldsymbol{x}_i)$ be the K-dimensional probability distribution prediction outputted from the network concerning the training sample $\boldsymbol{x}_i$. Then, the optimization objective of the standard training scheme with only top-most supervision can be defined as

$$\underset{\boldsymbol{W}_m}{\arg \min} \, \mathcal{L}_m(\boldsymbol{W}_m; \mathcal{D}) + \gamma \mathcal{R}(\boldsymbol{W}_m), \quad (1)$$

where $\mathcal{L}_m$ is the total loss over all training examples, $\mathcal{R}$ is the regularization term (usually ℓ_1 or ℓ_2 Frobenius norm) with γ as the positive weighting factor. Specifically, $\mathcal{L}_m$ is typically defined as the cross-entropy loss function

$$\mathcal{L}_m(\boldsymbol{W}_m; \mathcal{D}) = -\frac{1}{N} \sum_{i=1}^{N} \log f_m(\boldsymbol{W}_m; \boldsymbol{x}_i)^{(y_i)}. \quad (2)$$

Note that the regularization term $\mathcal{R}$ is an inherent attribute specific to the model structure which has no relation to supervision signals, for brevity and a better clarification of our main method, this term will be omitted in the following analysis. Now, Equation 1 can be reduced into

$$\underset{\boldsymbol{W}_m}{\arg \min} \, \mathcal{L}_m(\boldsymbol{W}_m; \mathcal{D}). \quad (3)$$

Besides one existing top-most classifier attached to the final fully-connected layer, Deeply-Supervised Nets [20]

append auxiliary classifiers over all hidden layers to create a more transparent learning process in which classification error information no longer needs to travel long distance over stacked modules to update the weights of a shallow layer. Instead, gradients could flow back through its nearest side branch in a much more straightforward manner. Let $\mathcal{W}_s = \{\boldsymbol{W}_s^l | \; l = 1, 2, \ldots, L - 1\}$ (s in subscript denotes side branch) be a set of weight matrices collected from auxiliary classifiers attached on top of intermediate layers where $\boldsymbol{W}_s^l$ denotes the weight matrix of the auxiliary classifier rooted in the lth hidden layer. Then, without loss of generality, the optimization objective of the training scheme with deep supervision can be expressed as

$$\underset{\boldsymbol{W}_m, \mathcal{W}_s}{\arg \min} \mathcal{L}_m(\boldsymbol{W}_m; \mathcal{D}) + \mathcal{L}_s(\boldsymbol{W}_m, \mathcal{W}_s; \mathcal{D}), \qquad (4)$$

where $\mathcal{L}_s$ is the weighted sum of losses from all auxiliary classifiers over all training examples with λ_l being the weighting factor of the lth auxiliary classifier. Namely, $\mathcal{L}_s$ is defined as

$$\mathcal{L}_s(\boldsymbol{W}_m, \mathcal{W}_s; \mathcal{D}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{L-1} \lambda_l \log f_s^l(\boldsymbol{W}_m^l, \boldsymbol{W}_s^l; \boldsymbol{x}_i)^{(y_i)},$$
$$(5)$$

where $f_s^l(\boldsymbol{W}_m^l, \boldsymbol{W}_s^l; \boldsymbol{x}_i)$ denotes the probabilistic prediction from the lth auxiliary classifier. Thus the optimization objective in the contemporary work, GoogLeNet [36], can be considered as a special case of Equation 4 which appends auxiliary towers selectively over two intermediate layers of its main branch. It is clear that through the newly introduced loss term in Equation 4, it allows the intermediate layers to gather gradients not only from the top-most supervision signal but also from the deep supervision signals, which is empirically demonstrated to be an effective method to combat the gradient vanishing problem and ensure faster convergence.

However, directly attaching simple classifiers to the hidden layers without thinking twice may lead to some performance decline when training very deep CNN models. Huang *et al*. [13] reported a series of similar experimental results and attributed the unsatisfactory performance to the intermediate auxiliary classifiers which interfere with the bottom-up feature generation process. As is known to all, deep neural networks have the capacity to represent hierarchical information as per depth, features learned in the shallow layers have fine spatial resolution but lack semantic meanings, thus their knowledge about category discrimination is much weaker in comparison to those of the deeper layers. Their hypothesis is further supported by the fact that accuracy degradation becomes more pronounced when auxiliary classifiers are attached to earlier layers. In this view, using a very discriminative supervision signal at intermediate layers might be too aggressive, it deviates the original shallow features optimized for short-term objectives.

That is, the directly attached auxiliary classifier may demand low-level features to be also discriminative about different categories resembling the high-level features. Without inevitably sacrificing the original coarse-level representation, the goal of precise visual recognition will be hard to achieve. Following promising methods to alleviate this issue [36, 50], we resort to auxiliary classifiers with rather complex structures. Specifically, each side branch consists of building blocks (*e.g.* residual blocks in ResNet) with the same type as the main branch. Furthermore, both auxiliary and the original branches keep the same down-sampling rate through its own pathway to the ending softmax classifier. We expect to maintain the progressive feature generation process in the main branch using these heuristic designing principles (we leave the architectural details regarding diverse networks in Section 4). Comparative experiments demonstrate that these well-designed auxiliary classifiers do facilitate the performance, though to a minor extent.

3.2. Dynamic Hierarchical Mimicking

As stated in the previous subsection, considering the locations and architecture design of auxiliary classifiers is essential to the whole network training. GoogLeNet [36] and MSDNet [13] also provide alternative solutions respectively. The former only attaches auxiliary towers to the endpoints of two relatively deep intermediate Inception blocks to ease the training, while the latter introduces horizontal and vertical connections, maintaining coarse-level information in the earlier layers and improving feature discrimination simultaneously. However, as clearly stated by the two, with their proposed methods, they have not achieved obvious improvement, *i.e.*, either around 0.5% or almost no change with a single auxiliary classifier, in their main classifier accuracy, compared with the standard training scheme.

We revisit the formulation of direct loss summation in Equation 4 and speculate that more intrinsic deficiency still lies in the optimization inconsistency among these added entries. Specifically, each item represents a sub-objective function of the corresponding auxiliary classifier. Discrepancy in their optimization directions could hamper the overall training procedure, leaving negative effects on the final model accuracy. Therefore, the more in-depth concern, which has been rarely explored in the related works [20, 36, 37, 13, 40, 26, 51] is the lack of essentially comprehensive interactions among the predication outputs from the auxiliary classifiers and the top-most classifier of the network. Denote the probabilistic representation information gathered by each branch as *knowledge*, our substantial research efforts have been invested in how to design a robust strategy that can facilitate aggregation of hierarchical knowledge extracted from classifiers located at different depths of the backbone network and relieve the optimization inconsistency among them.

Our core contribution is introducing a novel knowledge matching loss to regularize the training process towards optimization consistency efficiently and robustly. Based on the aforementioned analysis, we first attach delicately designed auxiliary classifiers to some particular intermediate layers of a given network. Locations of such intermediate layers are dynamically drawn from a given discrete probability distribution at each training epoch. Besides collecting the classification losses from auxiliary classifiers for straightforward optimization, we tend to focus more on their diverse representation in the pathway and meaningful probability distribution prediction outputs at the end. Once these knowledge from the auxiliary classifiers as well as the top-most classifier is generated, we use a pairwise probabilistic information mimicking strategy immediately, enabling on-the-fly comprehensive knowledge interactions and hierarchical information sharing. The objective function is yet another cross-entropy loss between the probabilistic prediction output of any two branches expected to be consistently optimized, partially resembling the KT procedure [9, 42, 49] in the form but also compactly combined with the DS methodology constrained in one single CNN model. Below, we describe the mathematical formulation of our proposed Dynamic Hierarchical Mimicking strategy.

Let $\mathcal{X}$ denote a probability space spreading over all the indices of intermediate layers and $\Phi \in \mathcal{X}$ be a dynamically sampled set containing indices of certain intermediate layers to which auxiliary classifiers are to be attached. Let $\tilde{\Phi} = \Phi \cup \{L\}$ and $\mathcal{W}_{\tilde{\Phi}} = \mathcal{W}_s \cup \{\boldsymbol{W}_m\}$ respectively denote the set of indices and weight matrices collected from all selected auxiliary classifiers and the top-most classifier. Let I_{Φ} be a binary indicator function as

$$I_{\Phi}(l) = \begin{cases} 1, & \text{if } l \in \Phi, \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where $I_{\Phi}(l) = 1, l = 1, 2, \ldots, L-1$ means there exists an activated auxiliary network classifier connected to the lth layer in the backbone. Then, following Equation 4, the optimization objective of our method is defined as

$$\underset{\boldsymbol{W}_m, \mathcal{W}_s}{\arg\min} \mathcal{L}_m(\boldsymbol{W}_m; \mathcal{D}) + \mathcal{L}_s(\mathcal{W}_{\tilde{\Phi}}; I_{\Phi}, \mathcal{D}) + \mathcal{L}_k(\mathcal{W}_{\tilde{\Phi}}; I_{\Phi}, \mathcal{D}), \tag{7}$$

where the second term samples some auxiliary classification losses from certain locations via the binary mask I_{Φ}, which is defined as

$$\mathcal{L}_s(\mathcal{W}_{\tilde{\Phi}}; I_{\Phi}, \mathcal{D}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{L-1} [I_{\Phi}(l)\lambda_l \tag{8}$$
$$\log f_s^l(\boldsymbol{W}_m, \boldsymbol{W}_s^l; \boldsymbol{x}_i)^{(y_i)}],$$

while the last term $\mathcal{L}_k$ (k in subscript denotes knowledge matching loss) represents pairwise probabilistic prediction

mimicking loss summed from all selected auxiliary classifiers in couples, which is defined as

$$\mathcal{L}_k(\mathcal{W}_{\tilde{\Phi}}; I_{\Phi}, \mathcal{D}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{p=1}^{L-1} \sum_{\substack{q=1 \\ q \neq p}}^{L-1} [I_{\Phi}(p) I_{\Phi}(q) \mu_{pq}$$
$$\sum_{k=1}^{K} f_s^p(\boldsymbol{W}_m^p, \boldsymbol{W}_s^p; \boldsymbol{x}_i)^{(k)} \log f_s^q(\boldsymbol{W}_m^q, \boldsymbol{W}_s^q; \boldsymbol{x}_i))^{(k)}], \tag{9}$$

where μ_{pq} is a positive coefficient indicating the confidence of the knowledge transfer process from the auxiliary classifier p to q. Empirically, we find that using the same weight (i.e., $\forall l, p, q$ setting $\lambda_l = 1, \mu_{pq} = 1$) for all entropy losses works well in practice, so we did not make much cumbersome exploration in tuning these weighting parameters. To stabilize the training process and avoid further regularization effect, we fork auxiliary classifiers from each node of the backbone network with probabilities toggling between zero and one in our main experiments, following a binary sampling strategy along the axis of network forward propagation. In addition, we also explore Bernoulli distributions $\mathcal{X}$ for comparison in the supplementary materials. As stated by Equation 9, the knowledge interaction process between any pair of activated network classifiers is expressed as a dual cross-entropy minimization process. This loss term can be optimized with an easily-implemented alternative of Kullback-Leibler divergence which differs from the original cross-entropy by nothing but a constant term. In principle, taking temporary probabilistic distribution prediction outputted from the network classifier p as a *fixed* soft label target, $\mathcal{L}_k$ forces the predication of network classifier q to become as similar as possible. In this way, knowledge currently learned by the network classifier p can be transferred to network classifier q as it accepts corresponding soft labels from classifier p as smoother hint for guidance. Therefore, by enabling dynamic knowledge mimicking among different combinations of network classifier cohorts in an on-the-fly fashion, our method can well enhance the capability of feature reuse across the whole network.

Theoretical Analysis. Through penalizing the distance between probabilistic distribution generated by different classifiers, the proposed mechanism behaves as a strong regularizer and improves model generalization ability. Without loss of generality, we select two matched branches for illustration. Let θ denote the network modules in their shared path, ϕ_1 and ϕ_2 denote the approximate functions in the separate branches. The total loss of one branch ϕ_1 can be represented as

$$-\mathbb{E}_{\boldsymbol{\xi}}\{\sum_{k=1}^{K} [(y^{(k)} + \phi_2^{(k)}(\theta(\boldsymbol{x}) + \boldsymbol{\xi})) \log \phi_1^{(k)}(\theta(\boldsymbol{x}) + \boldsymbol{\xi})]\}, \tag{10}$$

where $\theta(\boldsymbol{x})$ is the intermediate representation for input $\boldsymbol{x}$ with y as the corresponding label and $\boldsymbol{\xi}$ is the perturbation introduced by the randomness of data augmentation, with

zero expectation and variance σ. We lay analytic emphasis on the term of mimicking loss and derive that (refer to supplementary materials for the detailed derivation process)

$$-\mathbb{E}_{\boldsymbol{\xi}}(\phi_2^{(k)}(\boldsymbol{\theta}+\boldsymbol{\xi})\log\phi_1^{(k)}(\boldsymbol{\theta}+\boldsymbol{\xi})) \qquad (11)$$

$$\approx -\phi_2^{(k)}(\boldsymbol{\theta})\log\phi_1^{(k)}(\boldsymbol{\theta}) - \sigma^2\frac{\nabla_{\boldsymbol{\theta}}\phi_2^{(k)}(\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\phi_1^{(k)}(\boldsymbol{\theta})}{\phi_1^{(k)}(\boldsymbol{\theta})}.$$

In the approximation step, higher order infinitesimal of variable σ is omitted. The first term matches probabilistic predictions of the paired branches while the second term penalizes inconsistent gradients with respect to their shared parameters in the stem path, regularizing the overall training process robustly.

Theoretically and intuitively, aggregating the knowledge of multiple existing network classifiers in this novel way addressed the concern raised in the beginning of this subsection well.

4. Experiments

We conduct extensive experiments on several benchmark datasets to evaluate the effectiveness of our DHM method, *i.e.*, CIFAR-100 [17] and ILSVRC 2012 [3] datasets for image classification, Market-1501 [52] and DukeMTMC-reID [31] datasets for person re-identification. We follow the prevalent training scheme used in [18, 8, 44, 39, 14, 12, 2] for the single classifier based method and [20, 36, 37, 13] for the auxiliary classifier based ones separately. The experiments utilizing different training strategies are conducted with exactly the same setups for fair comparison, including data preprocessing, mini-batch sizes, training epochs and other relevant hyper-parameters for optimization.

4.1. Category Recognition

4.1.1 CIFAR-100

The CIFAR-100 dataset [17] is comprised of 50k training images and 10k test images, where each sample is a 32×32 colorful image within 100 categories. For data augmentation, we use the same data preprocessing method as [8, 20]. During training, images are zero-padded by 4 pixels on each side and then 32×32 regions are randomly cropped from padded images or their horizontal flips. The transformed image samples are finally normalized by subtracting their mean pixel value and dividing the standard deviation. During evaluation, error rates on the original test set are reported based on five successive runs with random seeds.

We apply different training strategies to three state-of-the-art CNN architectures for comprehensive comparisons, including ResNet [8], DenseNet [14] and WRN [44] with varied depths. For training, we use SGD with momentum as the default optimizer with initial learning rate as 0.1 and momentum as 0.9. We set the batch size as 128, the weight

Architecture	Method	Top-1 Error(%)
ResNet-32	baseline	30.306 ± 0.207
	DSL	30.174 ± 0.282
	DHM	$\mathbf{27.404 \pm 0.255}$
ResNet-110	baseline	28.598 ± 0.806
	DSL	28.352 ± 0.908
	DHM	$\mathbf{26.312 \pm 0.603}$
ResNet-1202	baseline	28.872 ± 1.218
	DSL	27.632 ± 0.923
	DHM	$\mathbf{24.860 \pm 1.546}$
DenseNet (d=40, k=12)	baseline	24.830 ± 0.116
	DSL	24.650 ± 0.414
	DHM	$\mathbf{23.112 \pm 0.247}$
DenseNet (d=100, k=12)	baseline	21.166 ± 0.259
	DSL	20.656 ± 0.274
	DHM	$\mathbf{19.912 \pm 0.201}$
WRN-16-8	baseline	20.488 ± 0.101
	DSL	20.052 ± 0.275
	DHM	$\mathbf{18.754 \pm 0.107}$
WRN-28-10	baseline	19.024 ± 0.125
	DSL	18.046 ± 0.264
	DHM	$\mathbf{17.200 \pm 0.224}$

Table 1. Top-1 error comparisons on CIFAR-100. Our results were obtained by computing mean and standard deviation over 5 runs (given in mean ± std. format in the table). The best result regarding each network architecture is highlighted in bold, baseline denotes model trained without adding auxiliary classifiers, DSL denotes model facilitated by Deeply Supervised Learning, DHM denotes model further boosted with Dynamic Hierarchical Mimicking, the same below in term of these notations.

decay as 0.0001 for all experiments. The learning rate annealing schedule follows the default settings of the original works proposing the corresponding network architectures respectively. We forks two carefully designed auxiliary classifiers before or after each down-sampling layer of these CNN architectures, *i.e.* after every residual stage for ResNet and WRN, after every transition layer for DenseNet. All auxiliary branches have the same heuristically designed macro-structure, *i.e.* stacked building blocks as the main branch, a global average pooling layer with a subsequent fully connected layer (refer to supplementary materials for architectural details).

Experimental results are summarized in Table 1 for clear comparison. Deep supervision consistently improves the model performance with carefully designed auxiliary classifiers, though restricted to relatively minor gain. Comparatively, our DHM method further brings considerable gain against the advanced deeply supervision scheme and more impressive gain against baseline across all network architectures. These experiments validate the effectiveness of our method, especially the vital importance of knowledge interaction process inside a single CNN model.

4.1.2 ImageNet

We also perform experiments on the large-scale ImageNet dataset [3], which is a much more challenging benchmark.

Architecture	Method	Top-1 / Top-5 Error(%)
ResNet-18	baseline	30.046 / 10.752
	DSL	29.728 / 10.450
	DHM	**28.714 / 9.940**
ResNet-50	baseline	23.990 / 7.166
	DSL	23.874 / 7.074
	DHM	**23.430 / 6.764**
ResNet-101	baseline	22.636 / 6.362
	DSL	22.260 / 6.128
	DHM	**21.348 / 5.684**
ResNet-152	baseline	21.894 / 5.886
	DSL	21.602 / 5.824
	DHM	**20.810 / 5.396**

Table 2. Top-1/Top-5 error comparisons on the ILSVRC 2012 validation set, with the single center crop testing method.

It consists of around 1.2 million training images and 50k validation images, labeled with 1,000 object classes. For training data processing, we use scale and aspect ratio augmentation and horizontal flipping as [36, 14]. Following common practice, top-1/top-5 error rates are reported on the validation set using single-crop testing.

We select the widely-used ResNet with varied depths as the backbone network for evaluation. We use the default input image resolution (224×224), batch size (256) , training epochs (90) and optimizer (SGD with momentum as 0.9 and weight decay as 0.0001) for training. The learning rate is initiated from 0.1 and decayed by the factor of 0.1 every 30 epochs. We choose the best model according to the validation accuracy among all training epochs since the baseline models are prone to over-fit along the overall training process. Noticing that our method behaves as a strong regularizer, scale augmentation is canceled for relatively shallow ResNet models to avoid excessive regularization effect when training with our hierarchical mimicking methodology. We attach two auxiliary classifiers after both the last and the penultimate group of residual blocks (*i.e.* denoted as `conv3_x` and `conv4_x` group in [8]). All auxiliary classifiers are constructed with sequential residual blocks, a global average pooling and a fully connected layer, though different in the depth and width of residual blocks (refer to supplementary materials for architectural details).

Experimental results are summarized in Table 2. Since ImageNet dataset is much more difficult than CIFAR-100, performance gain over the baseline model is *absolutely* small. Nevertheless, Deeply Supervised Learning method still boosts model accuracy and our Dynamic Hierarchical Mimicking mechanism further outperforms the deep supervision strategy with *relatively* considerable margins in top-1/top-5 accuracy. Even with the very deep architectures ResNet-101 and ResNet-152, substantial improvement of surpassing the baseline by over 1% in top-1 accuracy is achieved. Please refer to supplementary materials for their complete training curves.

Architecture	Method	Market-1501		DukeMTMC	
		mAP	R-1	mAP	R-1
ResNet-50 (w/ pretrain)	baseline	70.3	88.5	59.4	78.2
	DSL	72.0	88.2	60.5	78.8
	DHM	**76.7**	**90.3**	**65.4**	**81.1**
MobileNet (w/o pretrain)	baseline	55.6	78.2	45.7	69.0
	DSL	55.6	77.4	46.9	68.7
	DHM	**59.1**	**79.0**	**50.6**	**70.5**

Table 3. Rank-1 accuracy and mAP on the Market-1501 and DukeMTMC-reID datasets. R-1 denotes Rank-1 accuracy. w/ pretrain and w/o pretrain means with and without ImageNet pretrained weights loaded respectively.

4.2. Instance Recognition

We further conduct experiments on two popular person re-identification datasets to demonstrate the effectiveness of our method on this more challenging instance recognition problem. The Market-1501 [52] dataset has 32,668 bounding boxes drawn from 1,501 identities captured by 6 different cameras near the supermarket inside Tsinghua University, including 12,936 training images, 15,913 gallery images and 3,368 query images respectively detected by DPM [27]. The DukeMTMC-reID [31] dataset collected by 2 more cameras serves as one of the most challenging re-ID datasets to date, which contains 1,404 identities, 16,522 training examples, 17,661 gallery images and 2,228 queries.

We adopt the prevalent ResNet-50 [8] and the scalable MobileNet [11] as backbone networks and the simple cross-entropy as the loss function since our benchmark goal is to evaluate the newly proposed training mechanism. For the ResNet-50 backbone, training samples are resized slightly larger than the target size, then cropped to 256×128 regions and augmented with horizontal flipping and normalization. We set the batch size as 32 and train for 60 epochs with the AMSGRAD [28] optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$, weight decay=0.0005). The learning rate starts at 0.0003 and is divided by 10 every 20 epochs. Architectural design of auxiliary classifiers for ResNet-50 is totally identical to the one for experiments on ImageNet. Before standard training, ImageNet pre-trained weights of corresponding layers in the ResNet architecture are loaded, while all newly introduced layers for the re-ID model together with auxiliary classifiers without available pre-trained weights are trained for 10 epochs in advance with the pre-trained layers fixed. We also apply label smoothing during training since images in the re-ID datasets are not diverse enough. For the MobileNet backbone, we do not load the pre-trained weights, so we keep all the other settings the same but increase the initial learning rate to 0.001 and train longer for 90 epochs in total, decayed every 30 epochs with the same factor. We leave detailed structure of auxiliary classifiers for MobileNet in the supplementary materials. We report both mAP and Rank-1 accuracy under the single-query mode.

Side Branches	Top-1 / Top-5 Error(%)
$\mathcal{B}_0$	30.046 / 10.752
$\mathcal{B}_1$	29.276 / 10.272
$\mathcal{B}_2$	29.248 / 10.258
$\mathcal{B}_1$ & $\mathcal{B}_2$	**28.814 / 9.940**
$\mathcal{B}_1$ & $\mathcal{B}_2$ & $\mathcal{B}_3$	29.220 / 10.044

Table 4. Performance comparison of different configurations with respect to auxiliary branches.

From the result comparison in Table 3, we observe that deep supervision strategy leads to comparative performance in contrast to the baseline method, if not even worse. On the other hand, it is noteworthy that our hierarchical mimicking methodology outperforms the baseline with very compelling performance both on different datasets and with different backbones. Especially under the more comprehensive evaluation protocol of mAP, models trained with our proposed method achieve a margin of **over 6% mAP** on both datasets using the pre-trained ResNet-50 backbone.

4.3. Ablation Analysis

4.3.1 Knowledge Transfer Direction

From the view of Knowledge Transfer (KT), peer classifiers selected for mimicking can be deployed in a unidirectional or bidirectional mode. The unidirectional mode can include two specific configurations. One is the top-down configuration, in which only the probabilistic prediction from auxiliary classifiers connected to shallower layers is impelled to mimic that from deeper layers. The above situation reverses in the bottom-up configuration. However, inspired by [49], we heuristically prefer the bidirectional configuration which combines top-down and bottom-up mimicking directions. Actually, the bidirectional mode is the default choice in our main experiments. For rigorous verification, we also report the results of two configurations in the family of unidirectional mode using ResNet-18 model on the ImageNet dataset. Employing the bottom-up and top-down configuration respectively, there shows a slight decrease in top-1 error to 29.670 and 29.385 respectively, compared with the baseline of 30.046 and the DSL method of 29.728. This comparison preliminarily validates the regularization effect of our method as mathematically revealed in Equation 11. Recall that the resulting error of bidirectional mimicking in Table 2 is 28.814, we deduce that our adopted bidirectional mode further help to boost the knowledge interaction which is insufficient in the one-way skewed scenario above.

4.3.2 Design of Auxiliary Branches

Appropriate design of auxiliary classifiers is of vital importance to the final performance of deeply supervised learning and our proposed method. We perform experiments on the ImageNet dataset with ResNet-18 to analyze the influ-

Branch	Method	Top-1 / Top-5 Error(%)
$\mathcal{B}_0$	independent	30.046 / 10.752
	mimicking	28.814 / 9.940
$\mathcal{B}_1$	independent	27.988 / 9.560
	mimicking	27.626 / 9.176
$\mathcal{B}_2$	independent	31.458 / 11.522
	mimicking	29.544 / 10.370

Table 5. Influence of hierarchical mimicking on each auxiliary branch. Branches are evaluated in the same way as all the baseline experiments on ImageNet. 'independent' means individual classifier isolated from other branches, 'mimicking' means classifier trained together with peers using our DHM mechanism.

ence of various configurations related to auxiliary classifiers. We denote the main branch as $\mathcal{B}_0$ whose independent performance is identical to the baseline model. Auxiliary branches attached after the final and the penultimate group of residual blocks are denoted as $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively. We perform experiments by discarding one of the auxiliary branches of $\mathcal{B}_1$ and $\mathcal{B}_2$ or appending another auxiliary branch called $\mathcal{B}_3$ to shallower intermediate layers (refer to supplementary materials for details about its location and architecture). From Table 4, we notice that models trained with our proposed mechanism outperform the baseline model ($\mathcal{B}_0$) consistently, regardless of the number of auxiliary branches. One extra auxiliary classifier is sufficient to boost the performance by a non-negligible margin. Furthermore, we infer that the substantial gain of our proposed method does not arise from blindly increasing the model capacity via adding more auxiliary branches, since the triple-branch model starts to show a declining performance. Hence we adopt the double-branch model in our main experiments which achieves more satisfactory performance regarding both efficacy and efficiency. We also shed light on the influence of hierarchical mimicking on each auxiliary classifier. Towards this target, we first isolate each auxiliary branch from the main branch and train these classifiers separately. From the results shown in Table 5, it is obvious that all the auxiliary branches benefit from the regularization process within our proposed optimization mechanism compared to optimized independently.

5. Conclusion

In this paper, we propose a general-purpose optimization mechanism named DHM, which effectively and robustly facilitates the CNN training process without introducing computational cost during inference. Through delving into the training dynamics of deep supervision, a novel representation mimicking loss is considered to advance gradient consistency among optimization objectives of different delicately designed auxiliary branches. We theoretically and empirically demonstrate that this approach is beneficial to improving the accuracy and generalization ability of powerful neural networks on various visual recognition tasks.

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[2] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In *NIPS*. 2017.

[3] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[4] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

[5] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010.

[6] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *NeurIPS*. 2018.

[7] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *ICML*, 2013.

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

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

[10] Andrew G. Howard. Some improvements on deep convolutional neural network based image classification. *CoRR*, abs/1312.5402, 2013.

[11] 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. *CoRR*, abs/1704.04861, 2017.

[12] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[13] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018.

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

[15] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

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

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[19] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*, 2017.

[20] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, 2015.

[21] C. Li, M. Z. Zia, Q. Tran, X. Yu, G. D. Hager, and M. Chandraker. Deep supervision with intermediate concepts. *IEEE TPAMI*, 2019.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[25] Agata Mosinska, Pablo Márquez-Neila, Mateusz Koziński, and Pascal Fua. Beyond the pixel-wise loss for topology-aware delineation. In *CVPR*, 2018.

[26] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[27] D. Ramanan, P. Felzenszwalb, and D. McAllester. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[28] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *ICLR*, 2018.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*. 2015.

[31] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

[32] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[34] L. N. Smith, E. M. Hand, and T. Doster. Gradual dropin of layers to train very deep neural networks. In *CVPR*, 2016.

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[36] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[38] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using drop-connect. In *ICML*, 2013.

[39] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[40] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[41] Yibo Yang, Zhisheng Zhong, Tiancheng Shen, and Zhouchen Lin. Convolutional neural networks with alternately updated clique. In *CVPR*, 2018.

[42] J. Yim, D. Joo, J. Bae, and J. Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, 2017.

[43] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *CVPR*, 2017.

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

[45] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.

[46] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[47] Hanyuan Zhang, Hao Wu, Weiwei Sun, and Baihua Zheng. Deeptravel: A neural network based travel time estimation model with auxiliary supervision. In *IJCAI*, 2018.

[48] Xuan Zhang, Hao Luo, Xing Fan, Weilai Xiang, Yixiao Sun, Qiqi Xiao, Wei Jiang, Chi Zhang, and Jian Sun. Alignedreid: Surpassing human-level performance in person re-identification. *arXiv preprint arXiv:1711.08184*, 2017.

[49] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. Deep mutual learning. In *CVPR*, 2018.

[50] Zhenli Zhang, Xiangyu Zhang, Chao Peng, Xiangyang Xue, and Jian Sun. Exfuse: Enhancing feature fusion for semantic segmentation. In *ECCV*, 2018.

[51] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[52] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *ICCV*, 2015.