The Heterogeneity Hypothesis: Finding Layer-Wise Dissimilated Network Architecture

Yawei Li¹, Wen Li², Martin Danelljan¹, Kai Zhang¹,
Shuhang Gu³, Luc Van Gool^{1,4}, Radu Timofte¹

¹Computer Vision Lab, ETH Zürich, ²UESTC, ³The University of Sydney, ⁴KU Leuven
{yawei.li, martin.danelljan, vangool, radu.timofte}@vision.ee.ethz.ch,
{liwenbnu, cskazhang, shuhangu}@gmail.com

Abstract

In this paper, we tackle the problem of convolutional neural network design. Instead of focusing on the overall architecture design, we investigate a design space that is usually overlooked, *i.e.* adjusting the channel configurations of predefined networks. We find that this adjustment can be achieved by pruning widened baseline networks and leads to superior performance. Based on that, we articulate the “heterogeneity hypothesis”: with the same training protocol, there exists a layer-wise dissimilated network architecture (LW-DNA) that can outperform the original network with regular channel configurations under lower level of model complexity.

The LW-DNA models are identified without added computational cost and training time compared with the original network. This constraint leads to controlled experiment which directs the focus to the importance of layer-wise specific channel configurations. Multiple sources of hints relate the benefits of LW-DNA models to overfitting, *i.e.* the relative relationship between model complexity and dataset size. Experiments are conducted on various networks and datasets for image classification, visual tracking and image restoration. The resultant LW-DNA models consistently outperform the compared baseline models. Code will be available at https://github.com/ofsoundof/Heterogeneity_Hypothesis.

1 Introduction

Since the advent of deep learning era, convolutional neural network (CNN) [17] design has replaced the role of feature design in various computer vision tasks. Recently, neural network design has also evolved from manual design [39, 10, 14] to neural architecture search (NAS) [28, 40] and semi-automation [42, 12, 36]. State-of-the-art network designs focus on discovering the overall network architecture with regularly repeated convolutional layers. The repetition continues until the next stage of the network, *e.g.* the pooling operation. This has been the golden standard of current CNN design and been confirmed by some research. For example, Ma *et al.* mentioned that a network should have equal channel width [31]. But their analysis is limited to minimizing the memory access cost given the FLOPs for a single pointwise convolution.

The motivation of this paper kind of contradicts the previous design heuristics. In this paper, we investigate a design space that is usually overlooked and not fully explored, *i.e.* adjusting the layer-wise channel configurations. We try to answer three questions: i) whether there exists a layer-wise specific channel configuration that can outperform the original one; ii) how to identify it efficiently; iii) if there exists such an advantageous channel configuration, why it can beat the regular configuration.

Hints from network pruning. Recent network pruning methods shed lights on the existence of the advantageous layer-wise specific networks [29, 25]. Those methods can result in pruned networks with layer-wise specific channel configurations. Some works [29] report accuracy gain of the pruned

network over the width-scaled version of ResNet and MobileNets [10, 13, 38]. However, since the advantageous networks are identified in a network compression sense, thus with accuracy drop compared with the uncompressed network, it still remains unknown whether there exists a layer-wise specific network that can compete with the original one. A recent work [25] reports accuracy gain over uncompressed MobileNets on Tiny-ImageNet. Yet, further investigation on larger dataset is not conducted. Moreover, the pruned network are usually derived with training protocols different from that used for the baseline network, *e.g.* additional searching stage, larger batch size, or prolonged fine-tuning stage. It remains unknown how the layer-wise specific channel configurations benefit the network. Based on the discussion, we state the heterogeneity hypothesis as follows.

The Heterogeneity Hypothesis: *For a CNN, when trained with exactly the same training protocol (e.g. number of epochs, batch size, learning rate schedule), there exist a layer-wise dissimilated network architecture (LW-DNA) that can outperform the original network with regular layer-wise channel configurations under lower level of model complexity in term of FLOPs and parameters.*

To be specific, we aim at adjusting the channel numbers of the convolutional layers in predefined CNNs. The other layer configurations such as kernel size and stride are not changed. Formally, consider an L -layer CNN $f(x; \theta, \mathbf{c})$, where $\mathbf{c} = (c_1, c_2, \dots, c_L)$ is the channel configuration of all of the convolutional layers, θ denotes the parameters in the network, and x is the input of the network. The heterogeneity hypothesis implies that there is a new channel configuration $\mathbf{c}' = (c'_1, c'_2, \dots, c'_L)$ such that the new architecture $f'(x; \theta', \mathbf{c}')$ performs no worse than the original one. After the adjustment, the channel configurations c'_l could be either larger or smaller than the original c_l .

The focus of this paper is network architectures. Thus, to validate the advantage of LW-DNA models, controlled experiments should be conducted to exclude the influence of other factors other than network architecture. To do that, in addition to controlling influencing factors such as the optimizer and the training protocol, the identification procedure of LW-DNA should also be minimized to avoid additional computational cost and training time. This constraint leads to the following problem.

Problem Statement: *If the heterogeneity hypothesis is valid, how can we efficiently and reliably find a LW-DNA model for a CNN without any added computational cost and training time?*

Efficiency and reliability are the core constraints of the problem and should be kept in mind when designing the adjustment algorithm and criterion. An efficient adjustment algorithm that can discover LW-DNA models without added computational cost and training time should be developed. And a reliable adjustment criterion which can be robustly applied to different baseline network architectures and datasets should be devised. On the one hand, adjusting the channel configuration of a network involves a pruning criterion that diminishes c_l and a growing criterion that enlarges c_l . But there is no reliable criterion to decide where to grow a network merely based on the information from the predefined architecture. The lack of such a growing criterion makes the problem seemingly intractable. On the other hand, growing from a smaller architecture has the same effect of pruning a larger one. Thus, the lack of a proper growing criterion is circumvented by starting from a larger version of a network. In short, the adjustment problem in a smaller space is cast into a pruning problem in a larger space. Efficient pruning algorithm and reliable pruning criterion are developed.

Identifying LW-DNA.

The larger space is derived by widening the baseline network. A channel configuration residing in the space is identified by an efficient pruning algorithm. The recent developed DHP hypernetwork [25] is utilized as the identifying agent. With the agent, the less significant channels in the convolutional layers of the widened network are greedily pruned. This is done by removing the corresponding elements of the input latent vectors of the hypernetwork according to the vectors' gradients at initialization [22]. The pruning criterion, *i.e.* pruning wrt gradients at initialization, minimizes the pruning procedure to the computation of only one random batch. This solves the stated problem, *i.e.* identifying a better channel configuration without added computational cost and training time. More details and rationales about the choice of the LW-DNA identifying procedure are given in Sec. 3. In short, the identifying procedure proceeds as follows.

1. Reparameterize the widened baseline network with hypernetworks. The outputs of the hypernetworks act as the weight parameters of the baseline network.
2. Compute the gradients of the hypernetwork input, *i.e.* the latent vectors, with one random batch.
3. Prune the latent vectors greedily according to the magnitude of their gradients.
4. Compute the weight parameters with the pruned latent vectors.

5. Train the resultant network from scratch with the same training protocol as the baseline network. The hypernetworks are removed after the computation of the pruned weight parameters. The computed weight parameters are used as the trainable parameters during the training stage.

Explaining the benefits of LW-DNA. We try to identify LW-DNA versions of various state-of-the-art networks on three vision tasks including image classification [10, 14, 13, 38, 12, 40, 41, 36], image restoration [19, 26, 44, 37], and visual tracking [3]. Interestingly, the identified LW-DNA models are consistently better than the baseline even with lower level of model complexity in terms of FLOPs and number of parameters. We try to explain the phenomenon from several perspectives.

1. The CNNs are redundant. Thus, it is possible to shoot a layer-wise specific channel configuration comparable with the baseline under lower model complexity budgets.
2. Some layers of the LW-DNA model have more channels than the baseline. There is a tendency that the lower layers are strengthened with more channels. It might be those layers that play the essential role in improving the network accuracy.
3. The accuracy gain of LW-DNA model might be related to model overfitting. We derive this conjecture from several observations. **I.** By comparing the training and testing curve of LW-DNA model and the baseline in Fig. 2, we find that towards the end of the training, the identified LW-DNA models shoot a higher training error but lower testing error, *i.e.* improved generalization. This phenomenon is consistent across different datasets. This also matches the observations from the pioneering unstructured pruning, *i.e.* optimal brain damage and optimal brain surgeon that try to improve network generalization [18, 9]. **II.** The accuracy gain of LW-DNA model is larger on smaller datasets (Tiny-ImageNet) which is easier to be overfitted compared with larger datasets (ImageNet). **III.** On the same dataset (ImageNet), it is easier to identify a LW-DNA model version for larger networks (*i.e.* ResNet50) than for smaller networks (*i.e.* MobileNetV3).

Contributions. The contributions of this paper are summarized as follows.

1. We demonstrate that it is possible to identify a superior version of a network by only adjusting the channel configuration of the network. This validates the possibility of network pruning working as a complementary searching mechanism to semi- or fully automated neural architecture search.
2. We propose a method that can identify an LW-DNA candidate network without added computational cost and training time compared with the baseline network.
3. We try to explain the reason for the improved performance of LW-DNA by observing the experimental results.

2 Related Work

The lottery ticket hypothesis. The heterogeneity hypothesis is reminiscent of the lottery ticket hypothesis [6], which addresses the existence of sparse subnetworks that can match the test accuracy of randomly-initialized dense networks. The winning ticket is identified by greedily pruning single elements of weight parameters with smallest magnitude. The unstructured pruning breaks the dynamical isometry in the network [21]. The core problem is the trainability of the sparse subnetworks and the gradient flow in the subnetworks [21]. On the contrary, the heterogeneity hypothesis focuses on adjusting the channel configuration of the network. Since the weight elements of an entire channel is pruned together, there is no irregular kernel in the pruned network. Gradient flow is no longer a problem in this scenario.

NAS. NAS automatizes neural network design by searching in the design space [45, 27, 35]. Earlier works consume lots of computation [45, 27]. Recent developments accelerate the searching procedure by introducing differentiability into the optimization process [28]. After the searching stage, the derived cells are repeated to construct the final network. Thus, the final network still has a regular architecture. In this paper, we try to adjust the channel configuration of the network, which can be regarded as a complementary method to NAS.

Hypernetworks Hypernetworks are actually a kind of reparameterization of the backbone network [7]. Hypernetworks generates the weight parameters of the backbone network. The input of hypernetworks can be either static or dependent on the feature maps of the backbone network. In this sense, hypernetworks fall in the paradigm of meta learning. Recent developments bring hypernetworks to network pruning [29, 25]. Earlier hypernetwork designs are just a stack of two linear layers. Thus, the outputs are fixed, which should be cropped before being used as weights of the backbone network. The recent DHP hypernetwork [25] can adapt the outputs according to

length of the input latent vectors. This design naturally suits the task of network pruning. This is one of the reason why we select the DHP hypernetwork as the pruning agent.

Network pruning. Network pruning removes unimportant weight parameters in the network [18, 9, 8, 23, 24, 22, 25]. Among them, the most relevant are the DHP hypernetworks mentioned above and single-shot pruning [20, 21]. Since we want to purely investigate the importance of the architecture of the identified network, the other factors such as training protocol should be excluded. The pruning procedure should also be simplified as much as possible. Inspired by [22, 21], the widened network is pruned at initialization according to gradients. The pruning procedure only needs one random batch.

3 Methodology

3.1 Problem formulation

Configuration space and configuration vector. Consider an L -layer CNN whose channels are summarized as a configuration vector $\mathbf{c} = (c_1, c_2, \dots, c_L)$ in a configuration space $\mathbb{E}$ (See Fig. 1). The dimension of the configuration vectors depends on the number of convolutional layers in the network. State-of-the-art network architecture search methods identify an overall network architecture with fixed configuration vectors. The configuration vectors are regular and dependent on each other in the sense that most of the elements are repeated. For example, for the image classification networks, the golden standard is to repeat the building block with the same configuration until the reduction of spatial dimension of the feature map. Some efficient designs for mobile devices introduce a width multiplier $\alpha < 1$ to adapt to constrained resource requirements, which results in a scaled configuration vector $\alpha\mathbf{c} = (\alpha c_1, \alpha c_2, \dots, \alpha c_L)$.

Configuration vector adjustment. Since the configuration vector is manually fixed, it is not guaranteed to be optimal. In this paper, we explore in the new design space, *i.e.* the configuration space. The aim is to demonstrate that there is an irregular configuration vector $\mathbf{c}'$ that can compete with the original one under reduced model complexity. An algorithm needs to adjust the elements of the configuration vector $\mathbf{c}$ while controlling the model complexity. This operation is actually searching in the neighborhood of $\mathbb{N}(\mathbf{c}) \subset \mathbb{E}$ of the vector. After the adjustment, an element c_l can be either enlarged or diminished, which corresponds to growing or pruning the l -th layer of the network. Current research focuses on how to prune a network because it is straightforward to develop a pruning criterion based on the predefined architecture. But a pruning algorithm can only explore a subspace $\mathbb{S}(\mathbf{c}) = \{\mathbf{x} \in \mathbb{N}(\mathbf{c}) | x_l \leq c_l\}$ of the neighborhood. The lack of a convincing growing criterion makes the configuration vector adjustment problem seemingly intractable.

We circumvent this problem by introducing a larger searching space which is obtained by widening the width of the network with a width multiplier $\beta > 1$. The new searching space $\mathbb{H}$ is a hyperrectangle delimited by the zero vector $\mathbf{0}$ and the up-scaled configuration vector $\beta\mathbf{c}$ in the high-dimensional space, *i.e.* $\mathbb{H}(\mathbf{0}, \beta\mathbf{c}) = \{\mathbf{x} \in \mathbb{E} | 0 \leq x_l \leq \beta c_l\} \subset \mathbb{E}$. The searching algorithm then starts from the up-scaled vector $\beta\mathbf{c}$ and reduces the value of its l -th elements greedily according to the significance of the channels in the corresponding convolutional layer. This process at its core is pruning the widened networks. Thus, in this paper, the identification of the irregular configuration vector, *i.e.* the corresponding LW-DNA models is cast as a pruning problem in the larger space. As explained in Sec. 1, the pruning procedure consists of five steps, *i.e.* i) constructing hypernetworks for the widened baseline network, ii) computing the gradients of the latent vectors, iii) pruning the latent vectors according to the gradients, iv) computing the pruned weights parameters, and v) training the resultant network from scratch. And in the following, we explain some of the key steps in detail.

Figure 1: Illustration of the configuration space. The proposed method identifies layer-specific channel configurations within the enlarged and constrained subspace $\mathbb{C}(\rho\mathbf{c}, \beta\mathbf{c})$. Compared with searching within the constrained neighborhood $\mathbb{S}(\mathbf{c})$ of $\mathbf{c}$, the enlarged configuration space makes it possible to develop straightforward pruning criterion.

3.2 LW-DNA identifying agent

Recent research on network pruning reveals the fact that automatic network pruning can be regarded as fine-grained architecture search methods [30, 29, 25]. Inspired by that, the automatic network pruner is utilized as the identifying agent of LW-DNA. In this paper, the employed agent is the DHP hypernetwork that is tailored to the problem of network pruning considering its several merits [25]. The DHP hypernetworks bring network pruning into a latent space. The pruning of a channel is equivalent to deleting a single element of the latent vector. It provides a straightforward extension of single-shot pruning [22] to channel pruning (See Subsec 3.3). The latent vector sharing mechanism makes it possible to deal with various state-of-the-art networks.

As in Subsec. 3.1, the L -layer CNN is widened and brought to a larger configuration space $\mathbb{H}(\mathbf{0}, \beta\mathbf{c})$. Consider the l -th convolutional layer of the CNN. The dimension of the weight parameter is $\beta c_l \times \beta c_{l-1} \times w \times h$, where βc_l , βc_{l-1} , and $w \times h$ denotes the output channel, input channel, and kernel size of the layer. For the simplicity of notation, let $n = \beta c_l$ and $c = \beta c_{l-1}$ in the following. A latent vector $\mathbf{z}$ is attached to every layer of the CNN and for the l -th layer $\mathbf{z}^l \in \mathbb{R}^n$. The latent vector controls the output channel of the layer and removing an element of the latent vector is equivalent to pruning an output channel of the layer. Since the output channel of the current layer acts as the input channel of the next layer, the latent vectors are shared between consecutive layers. The hypernetwork takes as input the latent vector of the previous layer and the current layer. It computes a latent matrix, *i.e.* $\mathbf{Z}^l = \mathbf{z}^l \cdot \mathbf{z}^{l-1T}$. Then every element of the latent matrix is transformed to a vector by two consecutive linear operations, *i.e.*

$$\mathbf{O}_{i,j}^l = \mathbf{w}_2^l \cdot (\mathbf{Z}_{i,j}^l \mathbf{w}_1^l), \quad (1)$$

where $\mathbf{w}_1^l \in \mathbb{R}^m$ and $\mathbf{w}_2^l \in \mathbb{R}^{wh \times m}$. Note that $\mathbf{w}_1^l$ and $\mathbf{w}_2^l$ are unique for each element $\mathbf{Z}_{i,j}^l$ and for the simplicity of notation the subscript i,j is omitted. The output could be assembled into a 3D tensor $\mathbf{O}^l \in \mathbb{R}^{n \times c \times wh}$ which can be used as the weight parameter of the convolutional layer. The latent matrix acts as a handle for pruning. For example, if a single element of the latent vector $\mathbf{z}^l$ is nullified, the entire row of the latent matrix is masked out. As a result, the corresponding output channel is removed, thus achieving the effect of network pruning. For a better understanding, the demo code of the hypernetwork is attached in the supplementary.

3.3 Pruning criterion

Instead of using proximal gradient descent (PGD) to sparsify the latent vectors, we propose to prune the latent vectors according to their gradients right after the initialization of the hypernetwork. That is, the pruning criterion is defined as $\mathcal{C} = |\partial \mathcal{L} / \partial \mathbf{z}|$, where $\mathcal{L}$ is the loss function of the task, $|\cdot|$ takes the absolute value. Elements with small criterion values are pruned. The choice of the pruning criterion is based on the following considerations.

Pruning at initialization. The solution to the proximal operator with ℓ_1 regularization, *i.e.* the soft-thresholding operator, shows that PGD tends to diminish the elements of the latent vectors with the approximately consistent speed. As a results, the final magnitude of the elements has some kind of relationship with the initial magnitude. That is, if the initialization of an element is large, it is likely that the final magnitude is still relatively large and the element is kept after pruning. This is confirmed by the experiment. The distribution of the latent vectors during the PGD optimization is shown in the supplementary. As shown there, the final distribution is related to the initialization. Thus, it becomes reasonable to prune the latent vectors at initialization.

Pruning according to gradients. Proximal gradient 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 account for more FLOPs but less parameters compared with those in the higher layers. At initialization of the hypernetwork, the range of the gradients of the latent vectors are relatively balanced across the layers. Thus, gradients are chosen as the pruning criterion.

Justification of the collaboration between hypernetwork and pruning at initialization. Pruning at initialization is inspired by single-shot pruning of weight elements [22]. But the pruning criterion, *i.e.* the normalized gradient magnitude, is single element oriented. Structured pruning aims at nullifying an entire filter. It remains to be explored how to transform the heuristics of unstructured single-shot pruning to structured pruning. The hypernetworks provide such a connection. By resorting

to hypernetworks, the pruning procedure is transformed into the latent space. Deleting an element of the latent vector is equivalent to pruning a channel in the network. Thus, pruning the latent vectors according to their gradients is a natural transferring of the single-shot pruning heuristics.

Additional benefit. The additional benefit is that this pruning procedure is simple enough without sophisticated designs. Because only one random batch from the dataset is needed. Thus, exactly the same training protocol can be utilized for the baseline network and the identified LW-DNA model. This consistence makes it possible to identify the importance of the architecture of LW-DNA models while controlling the other factors.

Thus, the pruning procedure first computes the gradients of the latent vectors with respect to only one random batch. The derived gradients are used as the pruning metric. All of the latent vectors are greedily and jointly pruned according to the magnitude of the gradients.

3.4 Knowledge distillation

For image classification, besides the cross-entropy loss function, a distillation term is also used, *i.e.*

$$\mathcal{L} = (1 - \alpha)\mathcal{L}_{ce}(\mathbf{y}, \sigma(\mathbf{z}_s)) + 2\alpha T^2 \mathcal{L}_{ce}(\sigma(\mathbf{z}_s/T), \sigma(\mathbf{z}_t/T)), \quad (2)$$

where $\mathcal{L}_{ce}(\cdot)$ is the cross-entropy loss function, $\sigma(\cdot)$ is the softmax function, $\mathbf{y}$ is the class label, $\mathbf{z}_s$ and $\mathbf{z}_t$ are the logit outputs of LW-DNA model and the teacher [11]. We use fixed parameters $\alpha = 0.4$ and $T = 4$. The teacher is the pretrained widened version of the baseline network. Knowledge distillation is not used for experiments on ImageNet because the execution of the teacher network in this case also consumes considerable time and GPU resources.

3.5 Constraining model complexity

Model complexity is measured in terms of FLOP and parameter count. The target is to shoot a model that has both fewer FLOPs and parameters while achieving improved accuracy. Yet, the two metrics are not always consistent with each other. For example, when the FLOPs target is set, a parameter over-pruned model might be observed in some of the experiments, which could lead to inferior performance. Thus, a new hyper-parameter ρ is introduced which controls the minimum percentage of remaining channels in convolutional layers. In this way, the search space $\mathbb{C}(\rho\mathbf{c}, \beta\mathbf{c})$ is a confined subspace of the original search space $\mathbb{H}(\mathbf{0}, \beta\mathbf{c})$, *i.e.* $\mathbb{C}(\rho\mathbf{c}, \beta\mathbf{c}) = \{\mathbf{x} \in \mathbb{E} | \rho c_l \leq x_l \leq \beta c_l\} \subset \mathbb{H}(\mathbf{0}, \beta\mathbf{c})$. For image classification networks, a similar hyper-parameter τ for the final linear layers is also introduced. The two hyper-parameters ρ and τ are termed convolutional percentage and linear percentage in this paper, respectively. During the pruning, the FLOP budget is fixed. By tuning the hyper-parameters ρ and τ , the algorithm is able to find networks with the same FLOPs but varying parameter budgets.

3.6 Difference from pruning works

This paper is different from the previous pruning works in the following aspects. **Aim.** The aim of this paper is to identify a version of predefined network with improved accuracy and slightly reduced model complexity. Previous pruning works aim at improving the efficiency of networks and they bring accuracy drop for the pruned networks. **Method.** Pruning criterion for unstructured pruning is transferred to structured pruning by the collaboration of hypernetworks and gradient criterion. There is no computational overhead for the channel pruning method used in this paper. **Interpretation.** This paper tries to interpret where the benefit of the slightly reduced models comes from, which is not done by recent pruning works.

4 Experimental Results

The experimental results are shown in this section. We try to identify LW-DNA for various state-of-the-art networks including ResNet [10], RegNet [36], MobileNets [13, 38, 12], EfficientNet [41], MnasNet [40], DenseNet [14], SRResNet [19], EDSR [26], DnCNN [44], and U-Net [37]. The identified LW-DNA model and the baseline network are trained with exactly the same training protocol. The details of the training protocol for different tasks are given in the supplementary. For image classification on CIFAR [16] and Tiny-ImageNet[4], knowledge distillation [11] is also used for some of the experiments (Baseline KD, DHP KD [25], and LW-DNA model).

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.

Dataset	Network	Method	Top-1 Error (%)	FLOPs [G] / Ratio (%)	Params [M] / Ratio (%)
ImageNet [4]	ResNet50 [10]	Baseline	23.28	4.1177 / 100.0	25.557 / 100.0
		LW-DNA	23.00	3.7307 / 90.60	23.741 / 92.90
	RegNet [36]	Baseline	23.05	4.0005 / 100.0	22.118 / 100.0
		LW-DNA	22.74	3.8199 / 95.49	15.285 / 69.10
	MobileNetV3 [12] small	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 [13]	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 [38]	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
	MnasNet [40]	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 [36] 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
	EfficientNet [41]	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 [14]	Baseline	26.00	0.2901 / 100.0	1.100 / 100.0
		Baseline KD	22.84	0.2901 / 100.0	1.100 / 100.0
CIFAR10 [16]	ResNet56 [10]	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

Figure 2: Training and testing log of the LW-DNA models and the baseline models.

Image classification. The results of image classification networks are compared in Table 1. A complete version of the results is given in the supplementary. We have several key observations from the results. **I.** The identified LW-DNA models outperform the original network (denoted as Baseline or Baseline KD when knowledge distillation is used) with lower model complexity in terms of both FLOPs and number of parameters. This is a direct support for the Heterogeneity Hypothesis. **II.** The accuracy of the baseline network can be improved by knowledge distillation. Yet, the improved baseline still performs worse than LW-DNA. This shows the robustness of LW-DNA, *i.e.* not affected by a specific training technique. **III.** The improvement of LW-DNA scales up to large-scale datasets, *i.e.* ImageNet. For the ImageNet experiment, we set $\rho = 0.4$ and $\tau = 0.45$ by the ablation study on Tiny-ImageNet shown in the supplementary. This hyper-parameter combination works well across the three investigated networks. The success on ImageNet and the robustness of the hyper-parameters imply the wide existence of LW-DNA models and the ease of finding them.

The benefits of LW-DNA models are analyzed by several observations of the experimental results. **I.** The percentage of remaining channels is shown in Fig. 3. Some layers of the LW-DNA networks are strengthened. This might contribute to the improved performance of LW-DNA. **II.** As shown in Fig. 2, towards the end of the training, the LW-DNA models shoot a lower test error with increased training error. The improved generalization on the test set comes with reduced model complexity and

Figure 3: Percentage of remaining output channels of LW-DNA models over the baseline network.

lower training accuracy. This phenomenon is consistent with the pioneering unstructured pruning methods [18, 9] that try to balance model complexity and overfitting. The same phenomenon on both unstructured pruning and structured pruning points to a common underlying factor. **III.** The accuracy gain of LW-DNA on Tiny-ImageNet is larger than ImageNet. As known, smaller datasets are easier to be overfitted. **IV.** On ImageNet, it is easier to identify LW-DNA models for ResNet50 and RegNet than MobileNetV3. Since the larger models ResNet50 and RegNet contain more redundancy, it is easier for them to overfit the dataset. Based on the above observations, we conjecture that the improvement of LW-DNA model might be related to model overfitting.

Visual tracking. To validate the generalization ability of the identified LW-DNA, we apply the LW-DNA and baseline version of ResNet50 to visual tracking. State-of-the-art tracking workflow DiMP [3] is used as the test bed. For a fair comparison, the LW-DNA and the baseline are trained with the same protocol. They are first pretrained on ImageNet then finetuned following the DiMP workflow. In Table 2, the networks are compared on two datasets, *i.e.* TrackingNet [33] and LaSOT [5]. On the smaller dataset TrackingNet, LW-DNA version slightly beats the baseline while on the larger dataset LaSOT, LW-DNA outperforms the baseline elegantly. The success plot on LaSOT is shown in Fig. 4. As shown there, DiMP-LW-DNA is consistently better than DiMP-Baseline and other state-of-the-art tracking methods across the range of overlap threshold. In conclusion, the results show that **the benefits of LW-DNA can be transferred to other vision tasks.**

Image Restoration. Table 3 shows the results on super-resolution networks. For EDSR, the LW-DNA models perform better than the baseline but with significant reduction of model complexity. On the large test dataset Urban100 and DIV2K, the LW-DNA model of EDSR leads to nearly 0.1dB PSNR gain over the baseline. For SRResNet, LW-DNA achieves slightly reduction of model complexity without drop of PSNR. More results on image denoising are shown in the supplementary. In conclusion, the results **validate the existence of LW-DNA models for low-level vision networks.**

Table 2: Tracking test results. DiMP-LW-DNA and DiMP-Baseline use the identified LW-DNA and baseline version of ResNet50, respectively.

Metric	DiMP-Baseline	DiMP-LW-DNA
TrackingNet [33]		
Precision	68.06	68.27
Norm. Prec. (%)	79.70	79.64
Success (AUC) (%)	73.77	73.83
LaSOT [5]		
Precision	54.97	57.30
Norm. Prec. (%)	63.70	65.82
Success (AUC) (%)	55.87	57.43

Figure 4: Success plot on the LaSOT dataset for visual tracking.

Table 3: Results on single image super-resolution networks. The upscaling factor is $\times 4$.

Network	Method	PSNR [dB]					FLOPs [G] / Ratio (%)	Params [M] / Ratio (%)
		Set5 [2]	Set14 [43]	B100 [32]	Urban100 [15]	DIV2K [1]		
SRResNet [19]	Baseline	32.02	28.50	27.52	25.88	28.84	32.81 / 100.0	1.53 / 100.0
	LW-DNA	32.07	28.51	27.52	25.88	28.85	28.79 / 87.75	1.36 / 88.43
EDSR [26]	Baseline	32.10	28.55	27.55	26.02	28.93	90.37 / 100.0	3.70 / 100.0
	LW-DNA	32.13	28.61	27.59	26.09	28.99	55.44 / 61.34	2.84 / 76.94

5 Conclusion

In this paper, we state the heterogeneity hypothesis which in essence is the existence of LW-DNA models for predefined network architecture. We try to validate the hypothesis by empirical studies. In

order to single out the importance of the network architecture, the training protocol is kept the same for the baseline and the LW-DNA models. This is achieved by converting the problem of identifying LW-DNA to a pruning problem and designing an efficient pruning algorithm. The experiments on various network architectures and vision tasks demonstrate the benefits of the identified LW-DNA models. In addition, by observing the experimental results, we conjecture that the advantage of the LW-DNA model might be related to model overfitting.

References

- [1] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *Proc. CVPRW*, July 2017.
- [2] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *Proc. BMVC*, 2012.
- [3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proc. ICCV*, pages 6182–6191, 2019.
- [4] 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.
- [5] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. LaSOT: A high-quality benchmark for large-scale single object tracking. In *Proc. CVPR*, pages 5374–5383, 2019.
- [6] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.
- [7] David Ha, Andrew Dai, and Quoc V Le. HyperNetworks. In *Proc. ICLR*, 2017.
- [8] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *Proc. ICLR*, 2015.
- [9] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Proc. NeurIPS*, pages 164–171, 1993.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778, 2016.
- [11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- [12] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proc. ICCV*, pages 1314–1324, 2019.
- [13] 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.
- [14] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. CVPR*, pages 2261–2269, 2017.
- [15] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proc. CVPR*, pages 5197–5206, 2015.
- [16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- [17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [18] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Proc. NeurIPS*, pages 598–605, 1990.
- [19] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proc. CVPR*, pages 105–114, 2017.
- [20] Dongwoo Lee, Haesol Park, In Kyu Park, and Kyoung Mu Lee. Joint blind motion deblurring and depth estimation of light field. In *Proc. ECCV*, pages 288–303, 2018.
- [21] Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip HS Torr. A signal propagation perspective for pruning neural networks at initialization. *arXiv preprint arXiv:1906.06307*, 2019.
- [22] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. SNIP: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

- [23] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *Proc. ICLR*, 2017.
- [24] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proc. CVPR*, 2020.
- [25] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. DHP: Differentiable meta pruning via hypernetworks. *arXiv preprint arXiv:2003.13683*, 2020.
- [26] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proc. CVPRW*, pages 1132–1140, 2017.
- [27] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proc. ECCV*, September 2018.
- [28] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *Proc. ICLR*, 2019.
- [29] 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.
- [30] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *Proc. ICLR*, 2019.
- [31] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet V2: Practical guidelines for efficient cnn architecture design. In *Proc ECCV*, pages 116–131, 2018.
- [32] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*, volume 2, pages 416–423, July 2001.
- [33] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proc. ECCV*, pages 300–317, 2018.
- [34] 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.
- [35] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *Proc. ICML*, pages 4095–4104, 2018.
- [36] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. *arXiv preprint arXiv:2003.13678*, 2020.
- [37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, pages 234–241. Springer, 2015.
- [38] 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.
- [39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [40] 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.
- [41] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.
- [42] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. NetAdapt: Platform-aware neural network adaptation for mobile applications. In *Proc. ECCV*, pages 285–300, 2018.
- [43] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, pages 711–730. Springer, 2010.
- [44] 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.
- [45] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *Proc. ICLR*, 2017.

A Training protocol

The code is implemented in PyTorch [34]. 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.

Table 4: 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.

Dataset	Network	Method	Top-1 Error (%)	FLOPs [G] / Ratio (%)	Params [M] / Ratio (%)
ImageNet	ResNet50 [10]	Baseline	23.28	4.1177 / 100.0	25.557 / 100.0
		LW-DNA	23.00	3.7307 / 90.60	23.741 / 92.90
	RegNet [36] 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 [12] small	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 [13]	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 [38]	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 [12] 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 [12] 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 [40]	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 [36] 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 [36] 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 [36] 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 [36] 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 [41]	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 [14]	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 [14]	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 [10]	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

A.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[16] 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 channel-wise mean and standard deviation of the the training set [10, 14]. 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.

A.2 Visual tracking

For visual tracking, we follow the training protocol in [3]. To compare the baseline network and the LW-DNA models, the backbone network is initialized with the weights of ResNet50 and LW-DNA respectively. Then the same training and testing protocol is applied. The results are denoted by DiMP-Baseline and DiMP-LW-DNA respectively.

A.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. Each epoch 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.

B 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))
```

Table 5: Compression results on image denoising networks. The noise level σ is 70.

Network	Method	PSNR [dB]		FLOPs [G] / Ratio (%)	Params [M] / Ratio (%)
		BSD68	DIV2K		
DnCNN [44]	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 [37]	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 6: Ablation study of the hyper-parameters ρ and τ on MobileNetV1.

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

```
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.

C 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 4. Fig. 2 shows more results on the training and testing log of different models. Fig. 3 shows the percentage of remaining channels of more LW-DNA models.

Denoising

Image denoising results is shown in Table 5. The identified LW-DNA models perform no worse than the baseline network.

Ablation study on Tiny-ImageNet

An ablation study of the hyper-parameters ρ and τ is shown in Table 6. 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 6 are better than Baseline KD in Table 4, 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. 5. 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.

Figure 5: The distribution of the latent vectors in MobileNetV2 during the proximal gradient optimization of DHP.

Figure 6: Training and testing log of the LW-DNA models and the baseline models.

Figure 7: Percentage of remaining output channels of LW-DNA models over the baseline network