Russian Doll Network: Learning Nested Networks for Sample-Adaptive Dynamic Inference

Borui Jiang
Wangxuan Institute of Computer Technology,
Center for Data Science,
Peking University
jbr@pku.edu.cn

Yadong Mu*
Wangxuan Institute of Computer Technology,
Peking University
myd@pku.edu.cn

Abstract

This work bridges recent advances in once-for-all (OFA) networks [1] and sample-adaptive dynamic networks. We propose a novel neural architecture dubbed as Russian doll network (RDN). Key differentiators of RDN are two-folds: first, a RDN topologically consists of a few nested sub-networks. Any smaller sub-network is completely embedded in all larger ones in a parameter-sharing manner. The computation flow of a RDN starts from the inner-most (and smallest) sub-network and sequentially executes larger ones according to the nesting order. A larger sub-network can re-use all intermediate features calculated at their inner sub-networks. This crucially ensures that each sub-network can conduct inference independently. Secondly, the nesting order of RDNs naturally plots the sequential neural path of a sample in the network. For an easy sample, much computation can be saved without much sacrifice of accuracy if an early-termination point can be intelligently determined. To this end, we formulate satisfying a specific accuracy-complexity tradeoff as a constrained optimization problem, solved via the Lagrangian multiplier theory. Comprehensive experiments of transforming several base models into RDN on ImageNet clearly demonstrate the superior accuracy-complexity balance of RDN.

1. Introduction

Modern deep neural networks have been firmly established as state of the art approaches in many computer vision and multimedia analysis tasks, such as image recognition [8, 28, 5], object detection [31] and vision-language learning [21]. The development of several key engineering techniques, particularly modular network design and residual learning, has spurred the explosive growing of very deep or dense models to obtain further performance gain.

*Corresponding author.

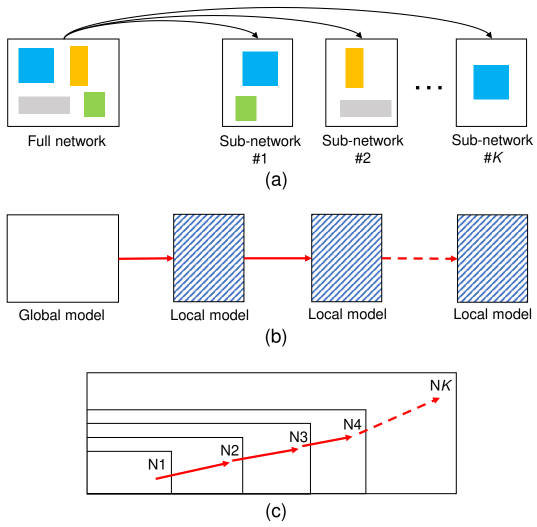


Figure 1. Conceptual comparison of Russian Doll Network (RDN) (in (c)) with representative once-for-all networks [1] (in (a)) and dynamic networks [25] (in (b)). Blocks with similar color (as in (a)) or filled pattern (as in (b)) imply parameter-sharing, which are jointly optimized during training.

Despite the remarkable new performance, large-sized deep networks have limited practical use owing to the intensive computations and memory footprint. Tremendous efforts have been devoted to learning slimmer and faster networks, such as using network compression [9, 32] and novel neural designs [22, 20]. In this work, we expose a novel network architecture, dubbed as *Russian Doll Network* (RDN), whose motivations are two-folds:

First, a deep model can be deployed on diverse hardware platforms with different requirement, ranging from low-cost smart-home sensors to dedicated Tensor Processing Unit (TPU). These platforms enforce different efficiency constraints and thus demand neural networks with varying complexities. Training an optimal model separately for each application scenario is seemingly possible yet economically forbidden. Recently, once-for-all networks [1] (see Figure 1(a)) devise multiple sub-networks with varying depths, widths, kernel sizes, and resolutions in a parameter-sharing manner. The training cost can be amortized by jointly optimizing all sub-networks in a single process. The resultant sub-networks can be flexibly deployed under diverse architectural configurations without re-training the models. Multiple sub-networks share similar architectures and context information, which makes the larger sub-networks efficiently deploy information and get faster and more accurate predictions.

Secondly, in image recognition or object detection, the difficulty level of confidently classifying an input significantly varies across different image or candidate box. Processing all samples equally is computationally inefficient since a lot of computations would be wasted on easy samples. [14] allocates proper weights for discriminating easy samples and hard samples during training. It is also crucial to treat easy samples during inference stage [3]. Researchers have recently proposed numerous methods for dynamic inference. Based on predictive inspection at some early network stage, a sample can choose different routes [24, 26], adaptively emphasize specific image subregion according to a learnable policy [25], or follows a gating mechanism for dynamically unrolling some neural units [7].

This paper introduces *Russian Doll Network* (RDN) as a new solution that bridges once-for-all networks and dynamic inference. Figure 1(c) shows the conceptual architecture of RDN, depicting a K-in-1 full network with recursively nested sub-networks $N_1, N_2, \ldots, N_K$. Crucially, we constrain RDNs to have strict sequential nesting order. To be specific, an inner sub-network in the sequence is fully embedded into any one larger than it, with all its parameters shared, which means the information flow from inner sub-networks to larger sub-networks, irreversibly. The computational flow always follows $N_1 \rightarrow N_2 \rightarrow \cdots \rightarrow N_K$. On the one hand, larger sub-networks have full access of the feature maps calculated at all previous sub-networks, but previous sub-networks are not affected by the larger sub-networks. This way, each sub-network acts as an uni-directional dependent classifier, and the residual network $N_i - N_{i-1}, i = 2 \ldots K$ aggregates all previous sub-networks $N_1, \ldots, N_{i-1}$ and progressively improves current sub-network. On the other hand, the sequential nesting order naturally enables dynamic inference. The calculation of a sample can early terminate once the confidence at some sub-network has been already sufficiently high. This will avoid unnecessary computations without heavy sacrifice of accuracy. In this work, we formulate the policy network to estimate the confidence distribution of samples, and propose a sample-adaptive inference as searching an optimal Lagrangian variable, which intelligently strives to achieve some pre-specified accuracy-complexity trade-off. The calculation of a sample can terminate by judging and weighing the user-given expected accuracy in order to save the computation complexity.

2. Related Works

Once-for-all network: Deep neural networks nowadays have wide deployment at diverse scenarios, which request different trade-off between latency and accuracy. Conventionally, deep networks are trained for special hardware platform, and need be re-configured and re-trained for any new deployment. For economic concern, training once and getting many diverse networks [30, 6] would alleviate the expensive cost of multiple deployments. Slimmable neural networks [29] privatize all batch normalization layers for each switch of channel width, producing varying-width networks. Once-for-all (OFA) networks [1] derive multiple networks along multiple dimensions (*e.g.*, resolution, channel width, depth etc.) and jointly train them via progressive shrinkage. In the model specialization stage, OFA-Net samples a subset of sub-networks to train an accuracy predictor and latency predictors. An optimal choice of sub-network will be thereby chosen for the target hardware and constraint.

Nested structure in deep networks: Numerous non-sequential structures in deep neural networks [13, 10, 23] have been recently explored for improved performance. Among them the most relevant work to ours is NestedNet [12]. Similar to our RDN, it is a resource-aware versatile architecture as the same network can meet diverse resource requirements. NestedNet is comprised of nested multiple levels of networks. Model parameters are shared across levels. However, NestedNet does not prune unnecessary connections between two levels of sub-networks, allowing high-level sub-network to update the features at lower level. This causes heavy parameter redundancy. In contrast, our proposed RDN only permits uni-directional information flow from inner-nested sub-network to large ones that it embeds, but not vice versa.

Dynamic deep networks: To avoid unnecessary computations, some methods have developed various sophisticated schemes for skipping part of the model. Examples include dropping some neural layers as piloted by a controller [27, 15] or conducting an early exit [11]. In image recognition task, image resolution occupies an important parts in both accuracy / complexity trade-off. GFNet [25] micro-form the traditional convolution neural networks and provides dynamic focusing to the enlarging image subregions. In this work, the proposed RDN naturally enforces a sequential nesting order of sub-networks. We adopt a Lagrangian multiple based policy for determining the optimal nested sub-network where a sample achieves desired level of accuracy / complexity trade-off.

3. Russian Doll Network

3.1. Overview

Here we formulate the construction of RDN as a generic network-transforming process. It aims to transform a pre-trained super-net into a new network which satisfies the primary requirement of RDNs, rather than learning RDNs from scratch. This way decouples structure configuration and parameter initialization, tending to lead to better accuracy as validated in our experiments. The construction of an RDN is comprised of four consecutive steps: adjusting the neuron connections that violate the nested structure, fine-tuning the new network as a multi-objective optimization, determining an optimal nesting order and eventually learning a policy that generates sample-adaptive neural route for pre-specified accuracy-complexity trade-off.

3.2. Transform Pre-trained Networks into RDN

In modern networks, channel-wise operators (variants ReLU, Sigmoid, Batch Normalization, depth-wise convolution etc.) need not any transform to become nested. But there are several popular neural operators in-proper to be transformed, including: 1) the output of L2 normalization, Softmax or Layer Normalization in some intermediate layers are tightly coupled. Remove or change a neuron would affect all others and violates the nesting requirement; 2) For a few popularly-adopted variants of convolution (such as grouped or depthwise separable convolution as used by MobileNeXt or MobileNetV2), they have already significantly-reduced parameters. Further transforming will not bring more model compression yet are prone to performance loss. We thus keep such operators unchanged.

Inspired by OFA-Net [1], we explore multiple dimensions to construct a nested network. In particular, we demonstrate the construction by operating on the dimension of network depth or channels in a convolution. Other dimensions are left for future exploration. Note the fully-connected (FC) layers can be treated as a special convolution with 1×1 receptive field.

Convolutional channel-wise nesting. Now let us elaborate on the surgery on vanilla convolutions. Let $\mathbf{X} \in \mathbb{R}^{c \times h \times w}$ be the input feature tensor of a vanilla convolutional layer, $\mathbf{W} \in \mathbb{R}^{c \times d \times k \times k}$ be the parameter matrix, where h, w define the spatial resolution and c, d are the counts of input / output channels respectively. Denote the convolutional output as $\mathbf{Y} = \mathbf{W} \otimes \mathbf{X} \in \mathbb{R}^{d \times h \times w}$. To make a convolution nested, assume its channels are split into k non-overlapping groups. Correspondingly, there are $k \times k$ sets of inter-group convolutional parameters. The main challenge of defining a nested convolution is tackling the connections that violate the nesting order (*e.g.*, red con-

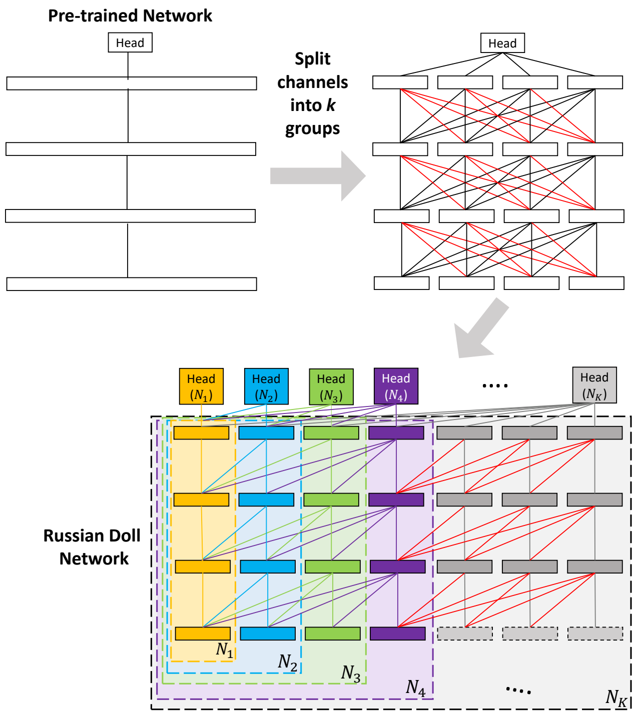


Figure 2. Learning a Russian Doll Network (RDN) can be accomplished by re-organizing and fine-tuning a pre-trained super-net. This figure overviews such a network surgery along the channel dimension. Each block represents a group of channels. The results RDN has a nesting structure as $N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_4 \rightarrow ... \rightarrow N_K$. We use different colors to imply which sub-network a connection or channel belongs to.

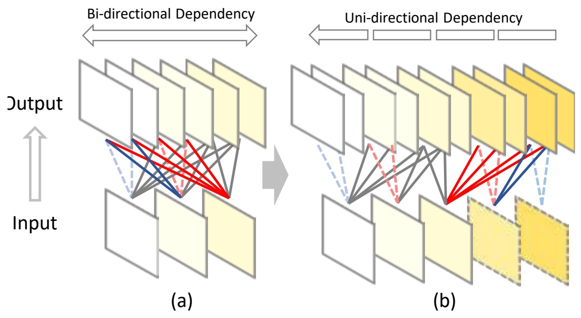


Figure 3. Illustration of transforming a vanilla convolution (a) with k groups into a nested one (b) with $K = 2k - 1$ groups. Connections that do not violate the nesting property are displayed in solid lines (*e.g.*, those in red and blue), otherwise shown in dashed lines. Note that the dependency is uni-directional for the groups in (b).

nections in Figure 2. Our proposed solution is introducing another $k - 1$ groups as the new destination of these connections, forming a total of $K = 2k - 1$ groups. This way eliminates all violating connections. For example, $k = 4$ and $K = 7$ in Figure 2.

Nested convolution can be regarded as group convolution + nested dependency among groups. Formally, we can define nested convolutions recurrently as below:

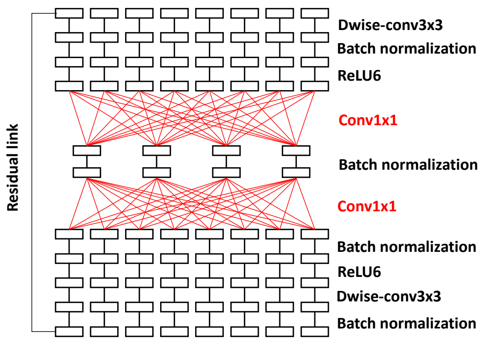
$$\mathbf{Y}_{i+1} = \begin{pmatrix} \mathbf{Y}_i \\ \mathbf{Y}_{i \to i+1} \end{pmatrix}, \quad 1 \le i \le K-1,$$
$$\mathbf{Y}_{i \to i+1} = \mathbf{W}_{i \to i+1}^{\mathsf{T}} \otimes \begin{pmatrix} \mathbf{X}_{q_1^i} \\ \dots \\ \mathbf{X}_{q_{n_i}^i} \end{pmatrix} + \mathbf{W}_{i+1}^{\mathsf{T}} \otimes \mathbf{X}_{i+1}, \tag{1}$$

where ( ), + denote tensor concatenation / addition, respectively. $\mathbf{q}^i \in \mathbb{R}^{n_i}$ is a sequence of indices, which defines the nested dependencies in group i. By the definition of nested convolution, $q_j^i \le i$ is a necessary condition. Feature $\mathbf{Y}_{i+1}$ concatenates two sources: one is the frozen $\mathbf{Y}_i$ as calculated by previous sub-networks in the nesting sequence, and the other is computed via vanilla convolution using both $\mathbf{X}_{q_1^i}, \dots, \mathbf{X}_{q_{n_i}^i}$ (parameterized by $\mathbf{W}_{i \to i+1}$) and $\mathbf{X}_{i+1}$ (controlled by $\mathbf{W}_{i+1}$). In this way it establishes nested dependency among sub-networks. In Figure 4, we show two popular base models, including MobileNeXt and ResNeXt. The operators which violate nesting manners are colored red.

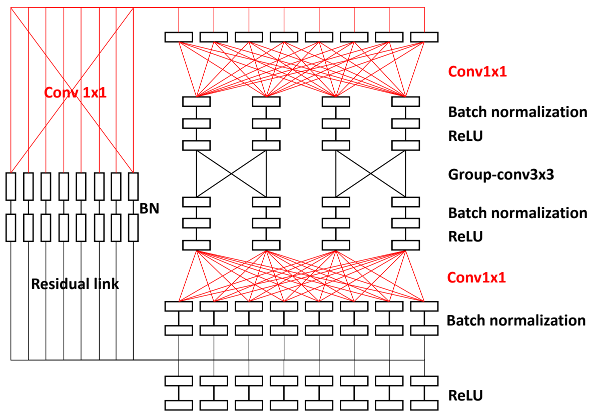
Network initialization is widely-known to be crucial for converging at high accuracy. We empirically find that training RDNs from scratch is not ideal, as later validated in experiments. Instead, we directly borrow the parameters in pre-trained super-net, namely using $\mathbf{W}$ to initialize all $\{\mathbf{W}_i\}$ and $\{\mathbf{W}_{i \to i+1}\}$. Figure 3 presents an example of transforming into a nested one. In addition, the supplemental materials describes more engineering implementation of above transform, including a greedy selection scheme that splits an input tensor into k groups and an inflation operation that solves the feature dimension mis-matching between nested convolution and other neural layers.

Depth-wise nesting. Nesting can be also accomplished along the dimension of network depth. Figure 5 shows an example of depth-wise split that divide a sub-network N_k to $N_{k,0}$ and $N_{k,1}$. For the two new sub-networks, $N_{k,0}$ is a relatively-shallow network with some additional light blocks for aligning spatial resolution and channel dimension between neural layers. The other $N_{k,1}$ reads the features outputted by $N_{k,0}$. These new sub-networks still have uni-directional dependency. Once the network $N_{k,0}$ finishes the computation, its features will not be updated during running $N_{k,1}$. Again, all blocks can re-use the parameters of the original N_k except for the newly-added light blocks.

Head design for sub-networks. We follow the common practice of stacking multiple MLP layers as the classifier's head. As shown in Figure 2, since the active feature channel varies for different nested sub-networks, the eventual flattened features for different heads may differ in length. To tackle this issue, we simply append an extra MLP layer in each head, aligning all the features to the same feature dimension. More importantly, as inspired by [25], the last layer of all heads are enforced to share parameters, which supposedly improves the robustness. Note that in RDN, the last layer is initialized by the classifier layer in the super-net.



(a) MobileNeXt Block



(b) ResNeXt Block

Figure 4. Two concrete examples (MobileNeXt and ResNeXt respectively) for transforming a pre-trained network into RDN. The surgical operations are mainly conducted on parameter-intensive convolutions (highlighted in red in the figures). Specially-designed convolutions (*e.g.*, depth-wise separable convolution in (a) or grouped convolution in (b) remain untouched, since their parameters are already economically used in the original network.

Greedy Select In Algorithm 1 we show the details of greedy select. The main purpose is getting an optimized permutation of filters which could reduce the differences between outputs of vanilla convolution and nested convolution. The whole algorithm is organized by selecting filters greedily and getting the target permutation step by step.

3.3. Learning Nesting Order

Previous works such as OFA-Net [1] adopt a large, searchable space for sub-spaces. For example, OFA-Net specializes a sub-network along four dimensions: depth,

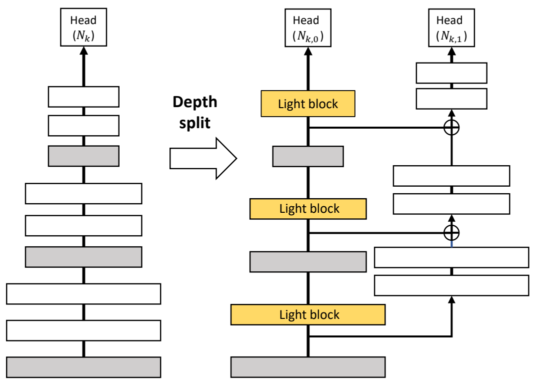


Figure 5. Our implementation of depth-wise nesting. On the left is a network with a classification head N_k. The right are two new sub-networks induced from N_k. Dependency of these two networks is still uni-directional.

Algorithm 1 Greedy Select

```
Input: convolutional parameters matrix, $\mathbf{W} \in \mathbb{R}^{c \times d \times k \times k}$;
    input tensor, $\mathbf{X} \in \mathbb{R}^{c \times h \times w}$; nested groups which is a
    divisor of both $c$ and $d$, $K \in \mathbb{N}^{+}$; input permutation of
    length $c$ from previous nested convolution, $\pi^{in}$;
Output: output permutation for the next layers, $\pi^{out}$;
 1: $sc \leftarrow \frac{c}{K}$, $sd \leftarrow \frac{d}{K}$
 2: for $g \in 1 \dots K-1$ do
 3:     $\pi \leftarrow$ a full permutation of length $d$
 4:     $\pi_g \leftarrow \pi^{in}_{g \times sc+1,\dots,c}$
 5:     for $t \in 1 \dots sd$ do
 6:         $\hat{b} \leftarrow \arg\min_b |\mathbf{W}^{\mathsf{T}}_{\pi_g \times b} \mathbf{X}_{\pi_g}|$, $b \in \pi$
 7:         $\pi \leftarrow \pi - \hat{b}$
 8:         $\pi^{out} \leftarrow \pi^{out} \cup \hat{b}$
 9:     end for
10: end for
11: $\pi^{out} \leftarrow \pi^{out} \cup \pi$
12: return $\pi^{out}$
```

width, kernel size and resolution. This brings tremendous memory for storing all of them and jointing training many sub-networks may cause interference with each other. The authors thus proposed to actively select a limited number of sub-networks occasionally during the training. Since the search space is not our main focus, this work adopts a relatively small space for sub-networks. As seen in Figure 6, we only allow the variations along network width or depth, establishing a 2-D grid of sub-networks. During training, a unique head is appended onto each sub-network. To strictly ensure a sequence of expanding nested sub-network, the transition between sub-networks are naturally defined by the network topology, as shown in the left panel of Figure 6. Once the optimization converges, all sub-networks are plotted with a complexity-accuracy frame. The optimality of

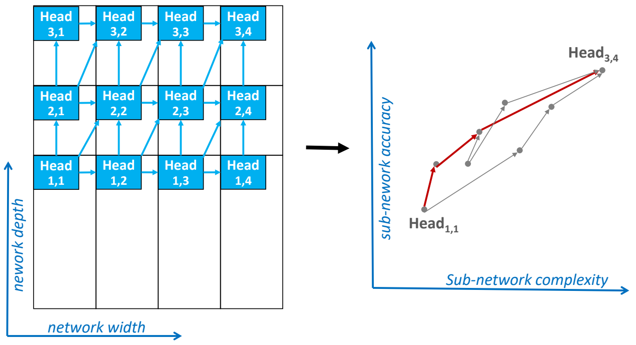


Figure 6. Learning nesting order. *Left*: topology of 2-D sub-networks along network depth and width; *Right*: plotting the learned sub-networks according to their network complexity and accuracy. The best nesting order is connected in red lines.

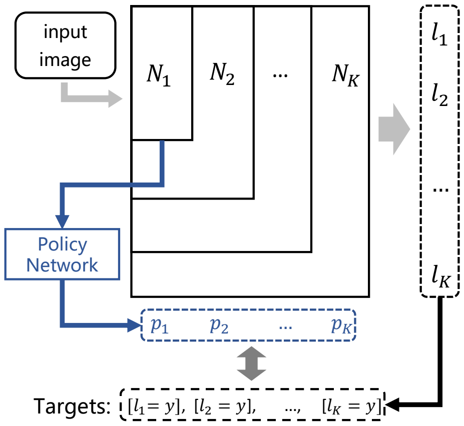


Figure 7. The training procedure of the policy network. Given an image, RDN with K sub-networks outputs a vector $(l_1, ..., l_K)$, which indicates K classification results of each sub-networks. Policy network estimates a vector $(p_1, ..., p_K)$ of k probabilities as described in Section 3.4. The bottom black dashed box indicates the adapting targets of policy network, in which y is the ground-truth label of given image. Note that "$[P]$" is the Iverson bracket, *i.e.*, 1 if P is true and 0 otherwise.

a nesting order is intuitively set as maximizing the integration under the curve, similar to the AUC (Area under the ROC Curve) metric widely used in image search. The optimal solution tends to be a convex hull of all sub-networks' plots. In addition, topological constraints should be satisfied. A pursued optimal nesting order is illustrated in the right panel of Figure 6.

3.4. Accuracy-Complexity Trade-off

Inspired by the DARTS algorithm in [3], we here propose a Lagrangian theory based scheme for elegantly searching a policy, which adaptively determines how far a sample will go in the learned nested RDN. In specific, we mis-use the notation K to denote the number of sub-networks in a learned RDN. N_K is the largest full model

Models	Avg Mem Used.(M)	MAdds.	Top-1 Acc.(%)	Top-5 Acc.(%)	Integrated Nets.
MobileNeXt-multi1.0	3.54	317M	74.02	91.65	1 full network
MobileNeXt-multi1.5	6.9	683M	76.64	93.10	1 full network
MobileNetV2-multi1.4	6.19	598M	74.16	91.63	1 full network
RDN-MobileNeXt-multi1.0	6.92	369M	74.17	91.68	7 sub-networks
RDN-MobileNeXt-multi1.5	14.40	789M	76.56	92.54	7 sub-networks
RDN-MobileNetV2-multi1.4	7.54	665M	73.96	91.67	7 sub-networks
RDN-MobileNeXt-multi1.0†	4.89	250M	74.15	91.56	7 sub-networks
RDN-MobileNeXt-multi1.5†	11.54	660M	**76.68**	**93.54**	7 sub-networks
RDN-MobileNetV2-multi1.4†	7.08	465M	74.03	91.65	7 sub-networks

Table 1. Main results of Russian Doll Network (RDN) and base models. "RDN-MobileNeXt", "RDN-MobileNetV2" are RDNs transformed from MobileNeXt, MobileNetV2 respectively. "multi1.0", "multi1.4", and "multi1.5" are variants definded by different paramters of multiplier as in [19]. "Avg Mem Used.(M)" are the average memory used during inference of whole network. In the top part, we show the results of base models. All of them are integrated by only 1 full network. In the middle part, the performances of the largest sub-networks in RDNs are compared with corresponding base models. In the bottom part, variants with a super-script † utilize our proposed dynamic inference. Number of parameters are slightly bigger than the basic RDN due to the additional policy network, and MAdds. are significantly saved because of the sample-adaptive dynamic inference.

and consumes most expensive computations. For a sub-network N_k, let the reward function $r_k \in [0,1]$ to indicate the percentage of saved computations in comparison with N_K. Clearly $r_K = 0$, and the sub-networks in the nesting order exhibit a decreasing r-value, namely $r_1 > r_2 > \ldots > r_K$.

Let f be a policy-induced classifier. $\Phi(f)$ is the averaged accuracy of classifier f. The optimization objective of accuracy-complexity trade-off can be written as below:

$$\begin{aligned} \underset{f}{\text{maximize}} \quad & R(f) = \mathbb{E}(r_{f(X)}[f(X) \text{ is correct}]) \\ \text{subject to} \quad & \Phi(f) \geq 1-\epsilon, \end{aligned} \tag{2}$$

where ϵ ($0 < 1-\epsilon \leq 1$) is a key hyper-parameter that decides the target accuracy level. The corresponding Lagrangian is:

$$L(f,\lambda) = R(f) + \lambda(\Phi(f) - 1 + \epsilon). \tag{3}$$

The optimization boils down to finding the Lagrangian variable λ that maximizes L. In practice, we pile up some Multi-Layer Perceptrons (MLPs) which reads the features of N_1 and plays as the policy net, seen in Figure 7. It is optimized to approximate $p_{Y|X}(v|x) = Pr(v \text{ is correct}|X = x)$. A one-pass binary search then finds λ^* for specific ϵ.

4. Experiments

In this section, we empirically evaluate the effectiveness of the proposed RDN and present ablation studies to corroborate various designs in our method. Firstly, we describe the experiment settings and critical implementation details. Secondly, we analyze the main results of all sub-networks in RDN, including the results when applying the sample-adaptive inference described in Section 3.4 during inference. Finally, we conduct a series of ablation experiments to show the superiority of our approach over variants.

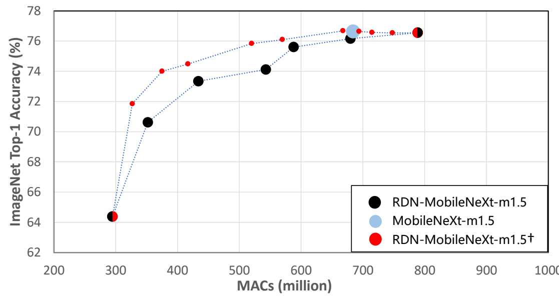


Figure 8. Performance of Russian Doll Network (RDN) transformed from MobileNeXt-multi1.5 base model. x-axis indicates the average MAdds during inference, and y-axis is the top-1 accuracy on ImageNet validation partition. In the figure, RDN-MobileNeXt-m1.5† (shown in red dots) is an enhanced RDN by applying sample-adaptive dynamic inference. Each red dot corresponds to the performance under a specific user-given λ.

4.1. Implementation Details

Dataset. Following previous work, we use ImageNet [18] for all of our classification experiments. The comparisons are based on accuracy versus various measures of resource usages such as the number of parameters, latency and multiply adds (MAdds) in mobile settings. We adpot the same data augmentation and pre-processing configurations as [2].

Setup. We adopt PyTorch toolbox [17] to implement all experiments. Code and pretrained models are mainly based on the implementation of public MobileNeXt-PyTorch [2]. If not mentioned, all the training and validation settings are the same as [2]. In specific, we train our models using distributed training setup on 8 NVIDIA V100 GPUs. As for both training-from-scratch or fine-tuning RDNs, we use the initial learning rate of 0.1, with batch size 1024 (128 images per GPU), and cosine learning rate scheduler [16] with decay rate of 1.0 and minimum learning rate of 1×10^{-5}. We

No.	Base model	Integrated Nets	Top-1 Acc.(%)
1	MnXt1.5	7 sub-networks	76.56
2	MBv21.4	7 sub-networks	74.16
3	MnXt1.5	14 sub-networks	75.44
4	MBv21.4	14 sub-networks	73.35

Table 2. Impact of depth-wise transformation in RDN. Both in MobileNeXt-multi1.5 and MobileNetV2-multi1.4, utilizing depth-wise transformation cause performance drop on ImageNet validation dataset by 1.12% and 0.81%, respectively.

No.	Pre-trained	Training Duration	Top-1 Acc.(%)
1	✓	200 epochs	74.17
2		200 epochs	70.35
3		400 epochs	71.03

Table 3. Impact of pre-trained base models of RDN-MobileNeXt-multi1.0. Note that the pre-trained MobileNeXt-multi1.0 is also trained for about 200 epochs under the same configuration as experiment 1 in this table.

use the standard SGD optimizer with Nesterov momentum 0.9 and weight decay 1×10^{-4}, and exponential moving average (EMA) with decay 0.9999. All convolutional layers use batch-normalization layers with an average decay of 0.99. The RDNs are further fine-tuned for 200 epochs after transformed from pre-trained super-models until otherwise clarified.

Backbones. MobileNetV2 [19] and MobileNeXt [2] are used as the pre-trained large networks to be converted into RDNs. After transforming them into nested architectures, additional MLP heads will integrate the features generated by each nested sub-networks. A shared classifier will output the classification results of all the sub-networks.

Nested transformation details. In our experiments, We adopt $k = 4$ by default, which totals 7 sub-networks in the final RDN. If a depth-wise transform is also utilized in RDN, we limit the transform to split the original sub-network into 2 new ones, which doubles the number of sub-networks to 14. A few shallow layers will be not involved in the nesting conversion, avoiding unnecessary performance drop. More concretely, the stem layer and the first two Sandglass Blocks in MobileNeXts, as well as the stem layer and the first two InvertedResidual Blocks in MobileNetV2s remain non-transformed. RDN transforms all the other layers which violate the nested property.

Policy network details. The policy network utilizes the features of the lightest sub-network in RDN. Given input samples, policy network could estimate the classification confidences of all the sub-networks in RDN. More concretely, policy network outputs binary predictions, indicating the estimations for top-1 classification precision of sub-networks. The $(\epsilon,\ \lambda^*)$ pairs are calculated on the training split of ImageNet [18], and used for sample-adaptive inference on the validation / test data.

4.2. Main Results

Table 1 presents key experimental results. For the notations, "RDN-MobileNeXt" is the RDN transformed from corresponding MobileNeXt. Likewise "RDN-MobileNetV2" is defined. "multi1.0", "multi1.4", and "multi1.5" correspond to three variants defined by a parameter of multipliers, full definition of which is found in [19]. The super-script † implies the use of our proposed dynamic inference. We have two major observations: 1) the RDNs demonstrate comparable performances, with slight increase of computation caused by the $k \to K$ group expansion and appending new network heads as described in Section 3.2, 2) our proposed dynamic inference can intelligently avoid unnecessary computation and save MAdds without sacrifice of performance. For example, the MAdds of RDN-MobileNeXt-multi1.0 and RDN-MobileNeXt-multi1.0† are 317M v.s. 250M respectively, with similar top-1 accuracy.

Depth-wise transformation. Transforming into depth-wise nested manners brings drop of performance but doubles the number of sub-networks. As shown in Table 2, after utilizing depth-wise transformation, both MobileNeXt-multi1.5 and MobileNetV2-multi1.4 would cause performance drop on ImageNet validation dataset by 1.12% and 0.81%, respectively.

Sample-adaptive dynamic inference. Figure 8 shows the accuracy-complexity trade-offs utilizing sample-adaptive dynamic inference. The RDN with sample-adaptive inference, termed RDN-MobileNeXt-m1.5† in Figure 8, clearly improves the top-1 accuracy, particularly at lower MACs.

4.3. Ablation study

Importance of pre-trained base models. In Table 3 we study the choice of utilizing pre-trained base models as well as random initialization. It can be seen that using pre-trained base models achieves better performance. Even after extending the training duration for the randomly-initialized network, there is still $\sim 3\%$ gap on Top-1 accuracy.

Importance of greedy select. In Table 4, we compare randomly or greedily split the channels into k groups when transforming vanilla convolutions. when utilizing greedy select strategy, performance of all the sub-networks in RDN will be improved, especially the lightest one. It is consistent with the insights in previous studies [1], [4] that parameters with better initialization and shared structures improve the generalization ability.

Joint optimization of multiple sub-networks in RDN. Table 5 investigates the mutual impact of multiple sub-networks in an RDN. We report the performances with 7

	select strategy	Top-1	Top-1(light)
RDN-MnXt1.5	greedy select	76.55	64.38
	random select	76.21	62.17

Table 4. Impact of greedy select strategy of RDN-MobileNeXt-multi1.5. "Top-1" indicates the top one accuracy on ImageNet validation dataset of the largest sub-network in RDN, and "Top-1(light)" indicates the accuracy of the lightest sub-network. Greedy select strategy significantly improves the performance of the lightest sub-net by $\sim 1.8\%$, and slightly effects the largest sub-network.

No.	Integrated Nets	Top-1 Acc.(%)
1	7 sub-networks	76.55
2	6 sub-networks	76.61
3	1 sub-network	76.12

Table 5. Impact of all the sub-networks in RDN. $No.1$ is vanilla RDN-MobileNeXt-multi1.5 model. $No.2$ shares the same structures with $No.1$, but is training without the lightest sub-network. Similarly, $No.3$ also shares RDN structures, and is only training with the largest sub-network.

(all), 6 (removing the lightest one), and 1 (only using the largest one) sub-networks. It is observed that jointly optimizing all sub-networks is apparently a better choice.

5. Conclusion

This work designs a novel Russian Doll Network (RDN) and present a method that transforms modern deep networks into RDNs. Additionally a dynamic inference scheme is proposed, targeting expedited computation with RDN. Our comprehensive evaluations clearly demonstrate the effectiveness of our proposed method.

Acknowledgement

This work is supported by National Natural Science Foundation of China (61772037) and Beijing Natural Science Foundation (Z190001).

References

[1] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020.

[2] Zhou Daquan, Qibin Hou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Rethinking bottleneck structure for efficient mobile network design. *arXiv preprint arXiv:2007.02269*, 2020.

[3] Jia Deng, Jonathan Krause, Alexander C Berg, and Li Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3450–3457, 2012.

[4] Biyi Fang, Xiao Zeng, and Mi Zhang. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. *CoRR*, abs/1810.10090, 2018.

[5] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xinyu Zhang, Ming-Hsuan Yang, and Philip H. S. Torr. Res2net: A new multi-scale backbone architecture. *CoRR*, abs/1904.01169, 2019.

[6] Luis Guerra, Bohan Zhuang, Ian Reid, and Tom Drummond. Switchable precision neural networks. *CoRR*, abs/2002.02815, 2020.

[7] Qiushan Guo, Zhipeng Yu, Yichao Wu, Ding Liang, Haoyu Qin, and Junjie Yan. Dynamic recursive neural network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2019.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision*, pages 1398–1406, 2017.

[10] Furong Huang, Jordan T. Ash, John Langford, and Robert E. Schapire. Learning deep resnet blocks sequentially using boosting theory. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 2063–2072, 2018.

[11] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *6th International Conference on Learning Representations*, 2018.

[12] Eunwoo Kim, Chanho Ahn, and Songhwai Oh. Nestednet: Learning nested sparse structures in deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8669–8678, 2018.

[13] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *5th International Conference on Learning Representations*, 2017.

[14] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):318–327, 2020.

[15] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 3675–3682, 2018.

[16] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[17] 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.

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy,

Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[19] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[20] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[21] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *International Conference on Computer Vision*, pages 7463–7472, 2019.

[22] Ke Sun, Mingjie Li, Dong Liu, and Jingdong Wang. IGCV3: interleaved low-rank group convolutions for efficient deep neural networks. In *British Machine Vision Conference*, 2018.

[23] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019.

[24] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. *CoRR*, abs/1709.01686, 2017.

[25] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, L. Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. *ArXiv*, abs/2010.05300, 2020.

[26] Wenhao Wu, Dongliang He, Xiao Tan, Shifeng Chen, Yi Yang, and Shilei Wen. Dynamic inference: A new approach toward efficient video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, Workshops*, pages 2890–2898, 2020.

[27] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogério Schmidt Feris. Blockdrop: Dynamic inference paths in residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018.

[28] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[29] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas S. Huang. Slimmable neural networks. In *7th International Conference on Learning Representations*, 2019.

[30] Tianyuan Zhang, Bichen Wu, Xin Wang, Joseph Gonzalez, and Kurt Keutzer. Domain-aware dynamic networks. *CoRR*, abs/1911.13237, 2019.

[31] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. *CoRR*, abs/2010.04159, 2020.

[32] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jin-Hui Zhu. Discrimination-aware channel pruning for deep neural networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, pages 883–894, 2018.