

DenseNets Reloaded: Paradigm Shift Beyond ResNets and ViTs

Donghyun Kim^{1*}, Byeongho Heo², and Dongyoon Han^{2*}

¹NAVER Cloud AI, ²NAVER AI Lab

Abstract. This paper revives Densely Connected Convolutional Networks (DenseNets) and reveals the underrated effectiveness over predominant ResNet-style architectures. We believe DenseNets’ potential was overlooked due to untouched training methods and traditional design elements not fully revealing their capabilities. Our pilot study shows dense connections through concatenation are strong, demonstrating that DenseNets can be revitalized to compete with modern architectures. We methodically refine suboptimal components - architectural adjustments, block redesign, and improved training recipes towards widening DenseNets and boosting memory efficiency while keeping concatenation shortcuts. Our models, employing simple architectural elements, ultimately surpass Swin Transformer, ConvNeXt, and DeiT-III — key architectures in the residual learning lineage. Furthermore, our models exhibit near state-of-the-art performance on ImageNet-1K, competing with the very recent models and downstream tasks, ADE20k semantic segmentation, and COCO object detection/instance segmentation. Finally, we provide empirical analyses that uncover the merits of the concatenation over additive shortcuts, steering a renewed preference towards DenseNet-style designs. Our code is available at <https://github.com/naver-ai/rdnet>.

1 Introduction

The “ImageNet moment” was sparked by the emergence of Convolutional Neural Networks (ConvNets), starting with the milestone AlexNet [38]. Subsequently, VGG [62] and GoogleNet [65] further highlighted the benefits of stacking multiple convolutional layers in ConvNets. In the same era, a monumental architecture ResNet [27] and its family [28, 87] stands out for introducing a groundbreaking concept - additive skip connections (also known as additive shortcuts or identity mapping [28]), which allowed for the stacking of up to 1,000 layers. The introduction of residual learning with it was a game-changer, diminishing the gradient vanishing problem by ensuring the input gradient always remained at one from the derivative of the identity mapping. This innovation sparked a series of successors, including the milestone ConvNets - EfficientNet [67] and ConvNeXt [48]; it paved the way for the next leap, such as Transformers [75], Vision Transformers (ViTs) [19], and Hierarchical ViTs [47], which accentuates the lasting influence of additive shortcuts.

* Equal contribution. Correspondence to Dongyoon Han.

In the early stage of this period governed by residual learning, Densely Connected Convolutional Networks (DenseNets [32]) introduced a novel approach: maintaining shortcut connections through feature concatenation instead of using additive shortcuts. This led to the concept of feature reuse [32], allowing more compact models and reducing overfitting through explicit supervision propagation to the early layers. DenseNets showcased efficiency and superior performance in tasks like semantic segmentation [36]. The evolution of architectural designs post-DenseNet appeared to challenge the dominance of ResNets but saw a decline in their popularity, shaded by the advantages of additive shortcuts.

Successors of DenseNets [40, 76, 79] revisited DenseNets to advance its design spirit but struggled against more dominant architectural trends again. We argue the potential of DenseNets still remained underexplored due to low accessibility, being gradually hindered by outdated training methods and the limitations of low-capacity components; they struggled to keep pace with the advancements in modern architectures that benefited from years of evolutionary refinements. Furthermore, we presume DenseNets requires an overhaul due to its limited applicability and memory challenges caused by increasing feature dimensionality. While the authors addressed memory concerns [32, 53], these issues continue to restrict the expansion of the architecture, particularly for width scaling. Despite the drawbacks, we conjecture the core design concept is still highly potent.

Bearing this in mind, this paper revitalizes DenseNets by highlighting the undervalued efficacy of concatenations. Through a comprehensive pilot study training with over 10k random networks across varied setups, we validate our claim that concatenation can surpass the additive shortcut. Afterward, we modernize DenseNet with a more memory-efficient design to widen it, abandoning ineffective components and enhancing architectural and block designs, while preserving the essence of dense connectivity via concatenation. We employ contemporary strategies that synergize with DenseNets as well. Our methodology eventually exceeds strong modern architectures [21, 25, 42, 45, 57, 97] and some milestones like Swin Transformer [47], ConvNeXt [48], and DeiT-III [71] in performance trade-offs on ImageNet-1K [59]. Our models demonstrate competitive performance on downstream tasks such as ADE20K semantic segmentation and COCO object detection/instance segmentation. Remarkably, our models do not exhibit slowdown or degradation as the input size increases. Ultimately, our empirical analyses shed light on the unique benefits of concatenation.

2 Related Work

Densely Connected Neural Networks (DenseNets) [32] pioneered dense connections within Convolutional Neural Networks (ConvNets) beyond additive shortcuts, highlighted by parameter efficiency and enhanced precisions. Building on this framework, some variants were proposed. PeleeNet [79] successfully proposed modifications to achieve real-time inference capabilities upon DenseNet. VovNet [40] departed from DenseNets’ dense feature reuse in favor of a sparser one-shot aggregation aimed at real-time object detection. CSPNet [76], by omit-

ting features and later concatenating them at cross-stage layers, reduces computational demands, barely affecting precision. DenseNets were further highlighted with the effectiveness on dense prediction tasks, for example, Jegou *et al.* [36] showed the effectiveness of DenseNets on semantic segmentation. MDU-Net [94] exploited the dense connectivity for enhanced biomedical image segmentation. DCCT [52] integrated dense connections into a Transformer architecture [75] to facilitate image dehazing. For video snapshot compressive imaging, EfficientSCI [78] also leveraged the benefits of dense connectivity. Wang *et al.* [82] utilized dense connections to improve the detection of small objects.

We believe these references demonstrate DenseNet-based designs’ potential, to our knowledge, but none recently challenged the ImageNet benchmarks using the principle of dense connections.

Modern architectures. DeiT [69] and AugReg [64] exhibited modernized training recipes [5, 67, 84] could replace massive training data for ViT [19] training. Descendant hierarchical ViTs [18, 47, 80, 81, 88, 89], which got closer to ConvNets, showed locality offers efficacy along with computational efficiency. Hybrid architectures [14, 25, 43, 74, 97] then explicitly equip convolutions for the locality. Ironically, incomers have become closer to ConvNets, aiming not to forsake the proven effectiveness of simple convolution, albeit using Transformers.

ConvNets [6, 27, 56, 67] initially predominated due to strong capability along with efficiency. Interestingly, advancements from the ViT side have also contributed to modernizing ConvNets; many recent architectures [23, 70, 73, 90] were inspired by ViT’s designs but armed with locality, demonstrating the continued high competitiveness of convolutions. Successors like RepLKNet [17] and SLaK [46] employed large-scale kernel convolutions built upon the predecessor’s legacy to emulate the globality of attention [19], offering to learn enhanced global representations. RevCol [7] introduced a new concept to mix multi-level features repeatedly through multiple columns. InceptionNeXt [91] adopted the inception module [65] inside ConvNeXt to show improved performance. HorNet [57] and MogaNet [42] both have presented remarkable performance by employing multiple gated convolution and multi-level features, respectively, which also took advantage of multi-scale features for globality.

Those architectures surpassed ViTs on ImageNet and dense prediction tasks as well, but similarities like using additive shortcuts and architectural complexity continue to restrict architectural diversity and innovation. Furthermore, network modernization methods [5, 48, 79, 84] have successfully revisited existing architectures but did not handle beyond baselines using additive shortcuts. This work follows a general direction but ensures our starts from a distinct baseline, acknowledging uncertainties about the effectiveness of existing roadmaps.

3 Methodology: Revitalizing DenseNets

This section starts with our conjecture that DenseNet may not fall behind modern architectures and proves it by substantiating revised DenseNet architectures. Based on our conjecture with our pilot experiments, we propose our methodology, which encompasses some modernized materials to revive DenseNet.

3.1 Preliminary

Motivation. ResNets [27] have renowned due to the simple formulation at l -th layer: $\mathbf{X}_{l+1} = \mathbf{X}_l + f(\mathbf{X}_l \mathbf{W})$, where the input \mathbf{X}_l and the weight \mathbf{W} . A pivotal element is the *residual connection* (i.e., additive shortcut, +), which facilitates modularized architectural designs as evidenced in Swin [47], ConvNeXt [48], and ViT [19]. While DenseNets [32] follow the formulation: $\mathbf{X}_{l+1} = [\mathbf{X}_l, f(\mathbf{X}_l \mathbf{W})]$ based on the *dense connection* through concatenation having explicit parameter efficiency. However, this formulation should constrain feature dimensionality due to memory concerns, making it challenging to scale in width.

DenseNets [32] initially outperformed ResNets [27] but failed to realize a complete paradigm shift, losing initial momentum due to the applicability. In particular, despite the efforts for memory [40,53], width scaling remains problematic for DenseNets, with wider models like DenseNet-161/-233 consuming more memory [32]. Nonetheless, inspired by the prior works [5,48,84] and motivated by successes in dense prediction tasks such as semantic segmentation [94], we believe DenseNets would outperform popular architectures and warrant further exploration of their potential: 1) feature concatenation merits strong capability; 2) the above concerns in DenseNets can be mitigated through strategic design.

Our conjecture. *Concatenation shortcut is an effective way of increasing rank.* Consider the layer output $f(\mathbf{X}\mathbf{W})$ with the weight $\mathbf{W} \in \mathbb{R}^{d_{in} \times d_{out}}$ and the input $\mathbf{X} \in \mathbb{R}^{N \times d_{in}}$ with a nonlinearity f , where we assume the number of instances $N \gg d_{in}$. As focusing on the matrix rank of f , $\text{rank}(f(\mathbf{X}\mathbf{W}))$ generally gets closer to d_{out} due to the nonlinearity when d_{in} is not that small [24]. Literature [10,22,24] manifested the layer \mathbf{W} with $d_{out} > d_{in}$ offers increased representational capacity. Intriguingly, DenseNets enjoy a similar aspect because we can decompose $\mathbf{W} = [\mathbf{W}_p, \mathbf{I}]$, where \mathbf{W}_p and \mathbf{I} denote the weights in the building block and concatenation. We further argue that increasing rank like this frequently would be more beneficial. The output dimension of \mathbf{W}_p is called *growth rate*.

A strategic design mitigates memory concerns. Consider the output of stacked layers $\mathbf{X}\mathbf{W}_1 f(\mathbf{W}_2)$, where the weights $\mathbf{W}_1 \in \mathbb{R}^{d_{in} \times d_r}$, $\mathbf{W}_2 \in \mathbb{R}^{d_r \times d_{out}}$, $d_r < d_{in}, d_{out}$, and a nonlinearity f after \mathbf{W}_2 . Likewise, the rank is likely preserved as d_r is not that small. This suggests that using intermediate dimension reducers like \mathbf{W}_1 (i.e., *transition layer*) may not impact the rank significantly. We argue a frequent application would effectively address memory concerns.

Pilot study. We conduct a pilot study to verify our conjecture by sampling over 15k networks on Tiny-ImageNet [39], where their shortcuts are either additive like ResNets [27] or concatenation in DenseNets [32]. We carefully control experiments regarding computational costs and involve diverse training setups to ensure a balanced and comprehensive comparison. Intriguingly, concatenation shortcuts all outperform additive ones with the averaged Tiny-ImageNet accuracy - **54.3±3.7 (concat) vs. 52.7±4.2 (add)**, thereby empirically supporting our claim. Detailed results and setups are provided in §5.1.

Table 1: ImageNet-1K performance progressions. Beginning from the baseline - DenseNet-201 [32], we report every performance change throughout progressions. We uphold DenseNet’s principle of feature reuse through concatenation as the core of the model progression. $+\alpha$ denotes a new element α was added to each prior model. Both enhancements in efficiency or accuracy are colored in **red**, while degradations are marked in **blue**.; GR denotes the growth rate, the amount of feature concatenation [32].

| Elements | Top-1 Acc (%) | Param (M) | FLOPs (G) | Lat (ms) (b1, Infer) | Lat (ms) (b128, Infer) | Lat (ms) (Train) | Mem (GB) (Train) |
|---|---------------|-------------|-------------|----------------------|------------------------|------------------|------------------|
| (a) DenseNet-201 [32] | 79.7 | 20.0 | 4.3 | 38.4 | 190 | 131 | 3.9 |
| (b) (a) + Wider & shallower | 79.5 (-0.2) | 21.8 (+1.8) | 11.1 (+6.8) | 8.5 (-29.9) | 170 (-20) | 85 (-46) | 3.2 (-0.7) |
| (c) (b) + Modernized blocks | 80.4 (+0.9) | 12.9 (-8.9) | 4.8 (-6.3) | 10.4 (+ 2.9) | 230 (+60) | 112 (+27) | 3.4 (+0.2) |
| (d) (c) + Channel dim \uparrow (GR \downarrow) | 80.8 (+0.4) | 19.9 (+7.0) | 4.7 (-0.1) | 11.8 (+ 1.4) | 184 (-46) | 88 (-24) | 3.1 (-0.3) |
| (e) (d) + Trans. layers \uparrow (GR \uparrow) | 82.3 (+1.5) | 21.2 (+1.3) | 5.0 (+0.3) | 11.0 (+ 0.8) | 183 (- 1) | 90 (+ 2) | 3.4 (+0.3) |
| (f) (e) + Patchification stem | 82.4 (+0.1) | 21.2 (-0.0) | 4.9 (-0.1) | 11.0 (- 0.0) | 179 (- 6) | 88 (- 2) | 3.2 (-0.2) |
| (g) (f) + Refined Trans. layers | 82.6 (+0.2) | 22.4 (+1.2) | 4.9 (+0.0) | 13.6 (+ 2.6) | 170 (- 9) | 97 (+ 9) | 3.1 (-0.1) |
| (h) (g) + Channel re-scaling | 82.8 (+0.2) | 23.9 (+1.5) | 5.0 (+0.1) | 14.0 (+ 0.4) | 175 (+ 5) | 99 (+ 2) | 3.1 (+0.0) |

3.2 Revitalizing DenseNets

We revisit DenseNets while maintaining its core principle via concatenation. Our strategy explores ways to widen DenseNets and identify effective elements. Elements that contribute to the performance improvements are detailed in Table 1.

Baseline. As the series of revisiting ResNets [5, 84] showed, refined training recipes bring significant improvements. Likewise, we train DenseNet-201 with a modern training setup, establishing it as our baseline. Following the well-explored setups [47, 48, 68, 69, 84], we include Label Smoothing [66], RandAugment [13], Random Erasing [16, 95], Mixup [93], Cutmix [92], and Stochastic Depth [33]; we use AdamW [50] with the cosine learning rate schedule [49] and linear warmup [20] with a popular large epochs training setup (300).

Going wider and shallower. DenseNets originally proposed exceedingly deep architectures (*e.g.*, DenseNet-265 [32]), which effectively showed the scalability. We argue that enhancing feature dimension through a high growth rate (GR) and increasing depth is hardly achieved simultaneously under resource constraints. Prior works [24, 40, 48, 56, 79] designed shallower networks to achieve efficiency, particularly latency. Inspired by this, we modify DenseNet to a favorable baseline accordingly; widening the network by augmenting GR while diminishing its depth. Specifically, we vastly increase GR - from 32 to 120 here - to achieve it; we adjust the number of blocks per stage, being reduced from (6, 12, 48, 32) to a much smaller (3, 3, 12, 3) for a depth adjustment. We do not shrink the depth as much to maintain minimal nonlinearity. Table 1(b) shows this strategic modification has led to notable latencies and memory efficiency - around 35% and 18% decreases in training speed and memory, respectively. The marked increase in GFLOPs to 11.1 will be adjusted through the later elements. Further study supports our decision - prioritizing width while balancing depth (see Table 8a).

Improved feature mixers. We employ the base block [48] for our feature mixer block, which has been extensively studied to reveal its effectiveness. Before using it, we should reevaluate the studies for our case because 1) DenseNets did not use

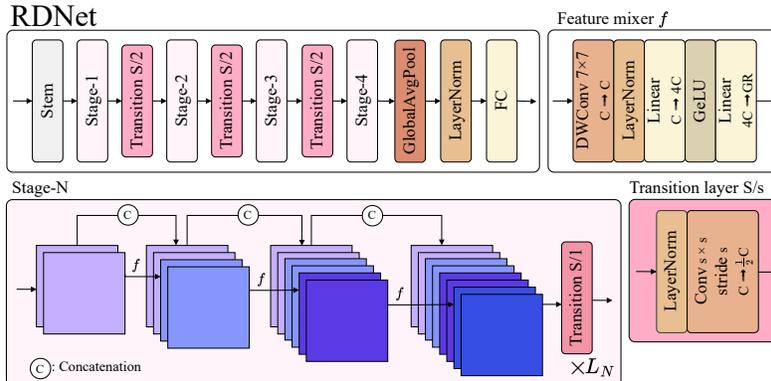


Fig. 1: Schematic illustration of RDNet. RDNet features a unique design distinguishing it from ResNet-style architectures, primarily due to the use of feature concatenation. We design four stages in RDNet across all scales, where each stage- N comprises $3L_N$ feature mixers and one additional transition layer. Feature mixer f denotes our building block combines previously concatenated features to compress them into GR-dimensional features for concatenation. The growth rate (GR) adjusts the amount of concatenated features and is predetermined for each stage. Transition layers for down-sampling are positioned after each stage as before. S and C denote stride and channel size. This figure illustratively sets GR to two.

additive shortcuts, and 2) the building block was originally designed to reduce dimensions successively. We find using the following setups still holds: using 1) Layer Normalization (LN) [2] instead of Batch Normalization (BN) [35]; 2) post-activation; 3) depthwise convolution [31] 4) fewer normalizations and activations; 5) a kernel size of 7. A unique aspect of our block is that the output channel (GR) is smaller than the input channel (C); mixed features are eventually more compressed features. As can be seen in Table 1(c), our design improves accuracy by a large margin (+0.9%p) while slightly increasing computational costs. We supplement factor analyses for our study here (see Table 8b).

Larger intermediate channel dimensions. A large input dimension for the depthwise convolution is crucial [60]. By adeptly modulating expansion ratio (ER) for inverted bottlenecks in the previous works [24, 48, 60, 67, 68] successfully achieved significant performance, by enlarging intermediate tensor size within the block beyond input dimensions (*e.g.*, ER was tuned to 6).

DenseNets similarly employed the ER concept; however, they distinctively applied it to the growth rate (GR) (*e.g.*, $ER=4\times GR$) rather than to the input dimension to reduce both input and output dimensions. We argue that this harms the capability of encoded features through the nonlinearity [24]. Thus, we reengineer the approach by directing ER proportional to the input dimension (*i.e.*, decoupling ER from GR). This change results in increased computational demands from a larger intermediate dimension; thus, halving GR (*e.g.*, from 120 to 60) manages these demands without compromising accuracy. Namely, we enrich the features before applying nonlinearity and further compress the

channels to control computational costs. Thereafter, we achieve both a faster training speed of 21% and 0.4%p improvement in accuracy shown in Table 1. Additionally, we conduct a factor analysis to ascertain whether reducing ER and increasing GR is preferable, or conversely, elevating ER and decreasing GR; Table 8c displays employing GR of 4 ultimately yields the optimal results.

More transition layers The transition layers [32] between stages are intended to reduce the number of channels. Due to the dense connections in every block, the intensified accumulation of features does not allow a high growth rate (GR). This gets worse as multiple blocks are stacked within a single stage, such as in the third stage, where numerous blocks are accumulated in a single stage with low GRs. We introduce a novel aspect using more transition layers to address it. To be specific, we propose to use a transition layer in a stage, not solely after each stage, but after every three blocks with a stride of 1. These transition layers do not concern downsampling but dimension reduction. This modification evidently reduces the computational costs substantially; therefore, we successfully increase overall GRs thanks to it¹. This is further supported by the results in Table 8e, which reveals using transition layers frequently often improves accuracy.

Additionally, we note that the models exhibit low parameter counts compared to their FLOPs. We remedy this by introducing variable GR at different stages (*e.g.*, 64, 104, 128, 192) instead of a uniform GR. Our further study in Table 8d suggests that a uniform growth rate (GR) compromises both accuracy and efficiency. Finally, Table 1(e) shows our design achieves significant accuracy improvements without greatly affecting computational costs.

Patchification stem Recent advancements revealed the effectiveness of using image patches as inputs within a stem [48, 57, 70]. We use the identical setup of a patch size 4 with a stride 4 as the patchification (LN [2] follows). Our empirical findings suggest that employing the patchification yields a notable acceleration in computational speed without loss of precision (see Table 1(f)).

Refined transition layers Another role of the transition layers was downsampling, and extra average poolings to downsample were adopted. We refine the transition layers, removing the average pooling and replacing the convolution by adjusting the kernel size and stride with the stride (LN replaces BN as well). Therefore, our transition layers play two additional roles: 1) dimension reduction, as aforementioned; 2) downsampling. Placing the transition layer after each stage exhibits +0.2%p gain, barely hurting efficiency (see Table 1(g)). For the dimension reduction ratio, we reexamine the impact, previously explored in [32]; Table 8f reconfirms 0.5 is optimal; higher transition ratios degrade precision.

Channel re-scaling. We investigate if channel re-scaling is required due to the diverse variance of concatenated features. We examine our proposed re-scaling approach, which has a similar formulation by merging the channel layer-scale [69] and an effective squeeze-excitation network [41]. Table 1(h) indicates it achieves a slight +0.2%p improvement, albeit with very minor inefficiency.

¹ Increase in GR aims to address the overall low GR in the baseline at an *architecture level*, whereas the abovementioned GR decrease was to boost ER on a *block level*.

3.3 Revitalized DenseNet (RDNet)

We finally introduce Revitalized DenseNet (dubbed RDNet), illustrated in Fig. 1. Our final model achieves both enhanced precision and efficiency, particularly enjoying significantly faster speed (see Table 1(h) vs. Table 1(a)). We construct the RDNet model family, aligned with the widely-adopted scales [27, 47, 48]. Our models distinctively include the Growth Rate - GR=(GR_1, GR_2, GR_3, GR_4), and the number of mixing blocks in each stage - B=(B_1, B_2, B_3, B_4), where we assign three blocks in each stage $B_n = 3L_n$. We summarize the configurations below:

- RDNet-T: GR = (64, 104, 128, 224), B = (3, 3, 12, 3)
- RDNet-S: GR = (64, 128, 128, 240), B = (3, 3, 21, 6)
- RDNet-B: GR = (96, 128, 168, 336), B = (3, 3, 21, 6)
- RDNet-L: GR = (128, 192, 256, 360), B = (3, 3, 24, 6)

4 Experiment

4.1 Image Classification

We evaluate our model family on ImageNet-1K [59]. Our models are trained following the training setups in Swin Transformer [47] and ConvNeXt [48] to ensure a fair comparison and not aimed to finetune the setups. The models are trained using AdamW [50] with a batch size of 512 and an initial learning rate of $1e-4$ for 300 epochs. As aforementioned in our baseline in §3.2, we employed identical data augmentations/regularization techniques to ConvNeXt’s; EMA is not used for our training. Comprehensive details of the training recipe are detailed in Appendix. We follow the standard evaluation protocols [27, 47, 48].

Our superiority is first underscored as compared with those of the current top-performing architectures [21, 25, 45, 57, 97]. We visualize the trade-off plots in Fig. 2 and detail the accuracies with diverse computational costs in Table 2. Ours show very competitive results compared with state-of-the-art models. Table 2 exhibits that while our models slightly fall behind in accuracy, they significantly make up with speed metrics. For example, RDNet-S can match with other lighter models such as SMT-S or MogaNet-S. Notably, ours do not require large memory usage as we aimed but achieve further efficiency.

We further exhibit a comparison with the popular models in Table 4. Ours surpass competitors by high precision, with decent memory usage and faster speeds. We further visualize trade-offs in Fig. 3, where RDNet demonstrates competitive performance even when juxtaposed with the milestone architectures.

4.2 Zero-shot Image Classification

We evaluate RDNet on ImageNet-1k zero-shot performance by training CLIP [55] to verify the applicability under a different training scheme. We follow the training protocol in ConvNeXt-OpenCLIP [11] using

Table 3: ImageNet-1K zero-shot classification results. Ours outperforms ConvNeXt further in efficiency.

| Models | Param | Top-1 | Top-5 |
|----------------|------------|-------------|-------------|
| ConvNeXt-B | 152 | 51.2 | 79.3 |
| RDNet-B | 134 | 53.0 | 81.0 |

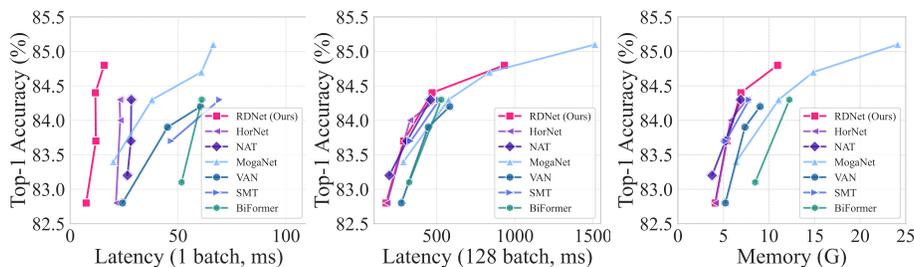


Fig. 2: ImageNet-1K performance trade-off among state-of-the-arts. We provide comparative visualizations between *state-of-the-art models*, which were known for top-performing models. It turns out that RDNet is highly competitive in practice in terms of model speed and memory consumption.

Table 2: ImageNet-1K comparison with the latest models. Fig. 2 visualized this table. We thoroughly compare our models against the latest architectures in practical latency and memory usage to demonstrate superiority. *bn* denotes latency (ms), measured with a batch size of n . *Mem* denotes the memory occupation (GB) measured with a batch size of 16. Interestingly, while our models slightly lag in accuracy, they significantly compensate with superior speed metrics.

| Model | Date | Param | FLOPs | Top-1 | b1 | b8 | b16 | b32 | b64 | b128 | Mem |
|------------------------------|--------------|-------|-------|-------|------|-------|-------|-------|-------|--------|------|
| RDNet-T | Ours | 24 | 5.0 | 82.8 | 7.4 | 13.4 | 24.6 | 45.7 | 88.9 | 175.2 | 4.1 |
| HorNet-T _{7×7} [57] | NeurIPS'2022 | 22 | 4.0 | 82.8 | 21.2 | 23.2 | 27.0 | 50.7 | 96.1 | 183.7 | 4.1 |
| VAN-B2 [21] | CVMJ'2023 | 27 | 5.0 | 82.8 | 24.2 | 28.0 | 39.0 | 75.4 | 144.4 | 274.8 | 5.2 |
| BiFormer-S [97] | CVPR'2023 | 26 | 4.5 | 83.8 | 51.6 | 50.5 | 50.9 | 86.8 | 167.5 | 197.2 | 8.5 |
| NAT-T [25] | CVPR'2023 | 28 | 4.3 | 83.2 | 26.5 | 28.0 | 33.2 | 53.0 | 102.2 | 335.3 | 3.8 |
| SMT-S [45] | ICCV'2023 | 21 | 4.7 | 83.7 | 46.9 | 48.2 | 55.8 | 96.0 | 176.5 | 335.3 | 5.3 |
| MogaNet-S [42] | ICLR'2024 | 25 | 5.0 | 83.4 | 20.0 | 22.4 | 40.9 | 77.2 | 147.4 | 288.1 | 6.4 |
| RDNet-S | Ours | 50 | 8.7 | 83.7 | 11.9 | 21.5 | 39.8 | 74.0 | 144.2 | 289.0 | 5.4 |
| HorNet-S _{7×7} [57] | NeurIPS'2022 | 50 | 8.8 | 84.0 | 23.2 | 25.7 | 46.0 | 88.3 | 171.4 | 328.9 | 5.7 |
| VAN-B3 [21] | CVMJ'2023 | 45 | 9.0 | 83.9 | 45.1 | 49.4 | 64.1 | 123.1 | 237.2 | 446.9 | 7.4 |
| BiFormer-B [97] | CVPR'2023 | 57 | 9.8 | 84.3 | 60.4 | 67.8 | 85.8 | 161.2 | 311.9 | 584.2 | 12.2 |
| NAT-S [25] | CVPR'2023 | 51 | 7.8 | 83.7 | 28.1 | 28.2 | 43.4 | 82.7 | 159.9 | 310.4 | 5.2 |
| SMT-B [45] | ICCV'2023 | 32 | 7.7 | 84.3 | 69.3 | 70.9 | 87.1 | 149.6 | 272.5 | 518.7 | 7.8 |
| MogaNet-B [42] | ICLR'2024 | 44 | 9.9 | 84.3 | 37.9 | 43.8 | 80.9 | 152.9 | 294.0 | 576.6 | 11.1 |
| RDNet-B | Ours | 87 | 15.4 | 84.4 | 11.7 | 32.2 | 61.4 | 116.6 | 233.7 | 471.6 | 6.9 |
| HorNet-B _{7×7} [57] | NeurIPS'2022 | 87 | 15.6 | 84.3 | 22.9 | 37.9 | 71.5 | 134.5 | 259.6 | 500.0 | 7.7 |
| VAN-B4 [21] | CVMJ'2023 | 60 | 12.2 | 84.2 | 60.4 | 67.8 | 85.8 | 161.2 | 311.9 | 584.2 | 9.0 |
| NAT-B [25] | CVPR'2023 | 90 | 13.7 | 84.3 | 28.3 | 33.5 | 43.4 | 82.7 | 159.9 | 310.4 | 5.2 |
| MogaNet-L [42] | ICLR'2024 | 83 | 15.9 | 84.7 | 60.8 | 64.9 | 118.5 | 224.3 | 429.2 | 838.6 | 14.9 |
| RDNet-L | Ours | 186 | 34.7 | 84.8 | 15.7 | 63.2 | 121.0 | 233.3 | 460.7 | 933.7 | 10.9 |
| MogaNet-XL [42] | ICLR'2024 | 181 | 34.5 | 85.1 | 66.3 | 112.3 | 207.5 | 394.0 | 771.9 | 1512.5 | 24.1 |

1.28B seen images from the aggregated set of CC3M [61], CC12M [9], and RedCaps [15]. We use the OpenCLIP codebase².

4.3 Semantic Segmentation

We employ ImageNet-1K pre-trained weights to perform semantic segmentation on the ADE20K [96] dataset using UperNet [86]. We use a learning rate of $8e-5$ with a weight decay of 0.03, and utilize stochastic depth rate 0.1, 0.2, and 0.3 for the RDNet-T, -S, and -B, respectively. The remainder of the training

² https://github.com/mlfoundations/open_clip

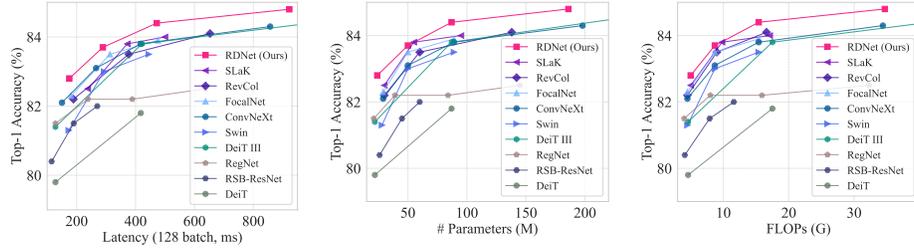


Fig. 3: ImageNet-1K performance trade-off among previous milestones. We provide comparative visualizations between previous architectures and our models. Notice that we also include speed comparisons to highlight actual differences in practice. Our models outperform the competing modern architectures revealing the potential of feature concatenation in designing networks.

Table 4: ImageNet-1K performance comparison with milestones. We report top-1 accuracy (%), parameter count (M), FLOPs (G), inference time (ms) for 128 images, and memory usage (GB) with a batch size of 16. All models are (pre-)trained on ImageNet-1k from scratch.

| Model | Res | Param | FLOPs | Lat | Mem | Top-1 | Model | Res | Param | FLOPs | Lat | Mem | Top-1 |
|----------------------|------------------|-------|-------|-----|-----|-------|----------------------|------------------|-------|-------|------|------|-------|
| RSB-ResNet50 [84] | 224 ² | 26 | 4.1 | 115 | 2.1 | 80.4 | RSB-ResNet152 [84] | 224 ² | 60 | 11.6 | 270 | 4.7 | 82.0 |
| RegNetY-4GF [56] | 224 ² | 21 | 4.0 | 128 | 2.7 | 81.5 | RegNetY-16GF [56] | 224 ² | 84 | 15.9 | 389 | 5.4 | 82.2 |
| DeiT-S [69] | 224 ² | 22 | 4.6 | 128 | 1.9 | 79.8 | DeiT-B [69] | 224 ² | 87 | 17.5 | 418 | 3.8 | 81.8 |
| CoaT-Lite-S [43] | 224 ² | 22 | 4.0 | 211 | 3.3 | 81.9 | Swin-B [47] | 224 ² | 89 | 15.4 | 445 | 5.4 | 83.5 |
| Swin-T [47] | 224 ² | 28 | 4.5 | 173 | 2.6 | 81.3 | PVTv2-B5 [81] | 224 ² | 82 | 11.8 | 414 | 7.0 | 83.8 |
| PVTv2-B2-Li [81] | 224 ² | 23 | 3.9 | 173 | 4.4 | 82.1 | ConvNeXt-B [48] | 224 ² | 89 | 15.4 | 417 | 5.4 | 83.8 |
| FocalNet-T [88] | 224 ² | 29 | 4.5 | 181 | 4.0 | 82.3 | CSWin-B [18] | 224 ² | 78 | 15.0 | 543 | 6.5 | 84.2 |
| ConvNeXt-T [48] | 224 ² | 29 | 4.5 | 150 | 2.7 | 82.1 | RepLkNet-31B [17] | 224 ² | 79 | 15.3 | 461 | 2.7 | 83.5 |
| CSWin-T [18] | 224 ² | 23 | 4.3 | 194 | 2.7 | 82.8 | DeiT III-B [71] | 224 ² | 87 | 17.5 | 422 | 4.0 | 83.8 |
| DeiT III-S [71] | 224 ² | 22 | 4.6 | 128 | 2.0 | 81.4 | FocalNet-B [88] | 224 ² | 89 | 15.4 | 476 | 6.1 | 83.9 |
| RevCol-T [7] | 224 ² | 30 | 4.5 | 189 | 2.0 | 82.2 | RevCol-B [7] | 224 ² | 138 | 16.6 | 653 | 3.5 | 84.1 |
| SLaK-T [46] | 224 ² | 30 | 5.0 | 238 | 3.3 | 82.5 | SLaK-B [46] | 224 ² | 95 | 17.1 | 558 | 6.9 | 84.0 |
| InceptionNeXt-T [91] | 224 ² | 28 | 4.2 | 132 | 3.3 | 82.3 | InceptionNeXt-B [91] | 224 ² | 87 | 14.9 | 405 | 6.1 | 84.0 |
| RDNet-T | 224 ² | 24 | 5.0 | 175 | 4.1 | 82.8 | RDNet-B | 224 ² | 87 | 15.4 | 472 | 6.9 | 84.4 |
| RSB-ResNet101 [84] | 224 ² | 45 | 7.9 | 190 | 3.9 | 81.5 | RegNetY-32GF [56] | 224 ² | 145 | 32.3 | 638 | 7.3 | 82.5 |
| RegNetY-8GF [56] | 224 ² | 39 | 8.0 | 238 | 4.0 | 82.2 | NFNet-F1 [6] | 320 ² | 133 | 35.5 | 421 | 5.9 | 84.7 |
| NFNet-F0 [6] | 224 ² | 71 | 12.4 | 235 | 3.5 | 83.6 | DeiT III-L [71] | 224 ² | 304 | 61.6 | 1375 | 10.5 | 84.9 |
| CoaT-Lite-M [43] | 224 ² | 45 | 9.8 | 396 | 5.5 | 83.6 | DeiT III-L [71] | 384 ² | 304 | 191.2 | 4586 | 28.1 | 85.8 |
| Swin-S [47] | 224 ² | 50 | 8.7 | 293 | 3.9 | 83.0 | ConvNeXt-L [48] | 224 ² | 198 | 34.4 | 857 | 8.6 | 84.3 |
| PVTv2-B4 [81] | 224 ² | 63 | 10.1 | 370 | 7.2 | 83.6 | ConvNeXt-L [48] | 384 ² | 198 | 101.1 | 2550 | 19.0 | 85.5 |
| ConvNeXt-S [48] | 224 ² | 50 | 8.7 | 266 | 4.0 | 83.1 | RDNet-L | 224 ² | 186 | 34.7 | 934 | 10.9 | 84.8 |
| CSWin-S [18] | 224 ² | 35 | 6.9 | 313 | 4.0 | 83.6 | RDNet-L | 384 ² | 186 | 101.9 | 2714 | 24.3 | 85.8 |
| FocalNet-S [88] | 224 ² | 50 | 8.7 | 313 | 4.6 | 83.5 | | | | | | | |
| RevCol-S [7] | 224 ² | 60 | 9.0 | 377 | 2.4 | 83.5 | | | | | | | |
| SLaK-S [46] | 224 ² | 55 | 9.8 | 372 | 5.0 | 83.8 | | | | | | | |
| InceptionNeXt-S [91] | 224 ² | 49 | 8.4 | 245 | 3.2 | 83.5 | | | | | | | |
| RDNet-S | 224 ² | 50 | 8.7 | 289 | 5.4 | 83.7 | | | | | | | |

settings follows ConvNeXt [48]. As demonstrated in Table 5, RDNet exhibits strong performance, which reveals the effectiveness on dense prediction tasks.

4.4 Object Detection

We evaluate object detection performance on COCO [44] using Mask-RCNN [26]. We use a learning rate of 3e-5 with a stochastic depth rate of 0.2. The remainder of the training settings follows ConvNeXt [48] again. As demonstrated in Table 6, RDNet exhibits competitive performance.

Table 5: ADE20K semantic segmentation results. All trained with the unified head UperNet (160K) on ADE20K. FLOPs (G) are measured at 512×2048 resolutions.

| Architecture | Crop | Param | FLOPs | mIoU ^{ss} | mIoU ^{ms} |
|-----------------|------------------------|------------|-------------|--------------------|--------------------|
| Swin-T [47] | 512 ² | 60 | 945 | 44.5 | 46.1 |
| ConvNeXt-T [48] | 512 ² | 60 | 939 | 46.0 | 46.7 |
| RevCol-T [7] | 512 ² | 60 | 937 | 47.4 | 47.8 |
| NAT-T [25] | 512 ² | 58 | 934 | 47.1 | 48.4 |
| RDNet-T | 512² | 58 | 961 | 47.6 | 48.6 |
| Swin-S [47] | 512 ² | 81 | 1038 | 47.6 | 49.5 |
| ConvNeXt-S [48] | 512 ² | 82 | 1027 | 48.7 | 49.6 |
| RevCol-S [7] | 512 ² | 90 | 1031 | 47.9 | 49.0 |
| NAT-S [25] | 512 ² | 82 | 1010 | 48.0 | 49.5 |
| RDNet-S | 512² | 86 | 1040 | 48.7 | 49.8 |
| Swin-B [47] | 512 ² | 121 | 1188 | 48.1 | 49.7 |
| ConvNeXt-B [48] | 512 ² | 122 | 1170 | 49.1 | 49.9 |
| DeiT III-B [71] | 512 ² | 128 | 1283 | 49.3 | 50.2 |
| RevCol-B [7] | 512 ² | 122 | 1169 | 49.0 | 50.1 |
| NAT-B [25] | 512 ² | 123 | 1137 | 48.5 | 49.7 |
| RDNet-B | 512² | 127 | 1187 | 49.6 | 50.5 |

5 Discussions

5.1 Pilot Study - Random Network Experiments

This study aims to reveal the effectiveness of dense connections over residual connections. We train tons of random networks across various scenarios, which include 1) multiple network scales; 2) multiple types of building blocks; 3) a range of network architectural elements; and 4) different training setups.

Parameter spaces and cost constraints. Table 7 (left) shows our parameter spaces for three individual scales, where $\text{RandNet}_{\mathcal{A},\mathcal{B},\mathcal{C}}$ are trained in. We diversify the search space with respect to the budgets, such as parameter count, FLOPs, and activation (reflecting memory consumption). We expand space from \mathcal{C} to \mathcal{D} by incorporating data augmentation, and further to \mathcal{E} with both data augmentation and a different optimizer [50]. Only randomly generated networks that meet the predefined budget are trained. We use the 90-epochs training setup [27] trained on Tiny-ImageNet [85]. For \mathcal{C}, \mathcal{E} spaces using data augmentation [13, 33, 66, 92, 93, 95], training is done for 180 epochs. Overall, the cumulative trained networks reach over 15k.

RandNet architecture. Based on [27], we stack random building blocks within the first stage. We generate random networks in the parameter space containing

Table 6: COCO object detection and segmentation results. We utilize MaskRCNN with 3x schedule. FLOPs (G) are calculated with image size (1280, 800). The result of Swin-T is from the official repository [1].

| Backbone | Param | FLOPs | AP ^{box} | AP ₅₀ ^{box} | AP ₇₅ ^{box} | AP ^{mask} | AP ₅₀ ^{mask} | AP ₇₅ ^{mask} |
|-----------------|------------|-------------|-------------------|---------------------------------|---------------------------------|--------------------|----------------------------------|----------------------------------|
| PVT-S [80] | 44M | 304G | 43.0 | 65.3 | 46.9 | 39.9 | 62.5 | 42.8 |
| Swin-T [47] | 48M | 267G | 46.0 | 68.1 | 50.3 | 41.6 | 65.1 | 44.9 |
| ConvNeXt-T [48] | 48M | 262G | 46.2 | 67.9 | 50.8 | 41.7 | 65.0 | 44.9 |
| RDNet-T | 43M | 278G | 47.3 | 68.5 | 51.9 | 42.2 | 64.6 | 44.8 |

Table 7: Random network experiments. We present our experimental setups (left) and results (right). Five parameter spaces guide random network generations for two distinct shortcuts. We sample random networks within each parameter space, ensuring similar computational costs. Each parameter space varies in 1) architectural elements - channel sizes, activations, normalizations, and convolution kernel sizes in $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}$; 2) data augmentations in \mathcal{D}, \mathcal{E} ; 3) optimizers in \mathcal{E} . \mathcal{D}, \mathcal{E} is based on the architectural space \mathcal{C} . $[a_1, \dots, a_n]$ and (a, b, c) denote a closed interval: a list of n elements and a range of elements from a to b with a step of c , respectively. All results are averaged.

| Parameter space | \mathcal{A} | \mathcal{B} | \mathcal{C} | Model | Skip | FLOPs | Param | Acts | Top-1 (%) |
|----------------------|-------------------------------------|---------------|---------------|--------------------------|--------|-----------------|-----------------|-----------------|----------------------------------|
| Param (xM) | $2 < x < 2.5$ | $4 < x < 5$ | $9 < x < 10$ | RandNet $_{\mathcal{A}}$ | add | 2.31 ± 0.11 | 2.27 ± 0.11 | 0.77 ± 0.13 | 45.6 ± 2.2 |
| FLOPs (xG) | $2 < x < 2.5$ | $4 < x < 5$ | $9 < x < 10$ | RandNet $_{\mathcal{A}}$ | concat | 2.15 ± 0.11 | 2.16 ± 0.11 | 0.61 ± 0.11 | 47.8 ± 2.1 |
| Depth | (3, 6, 1) | (3, 8, 1) | (3, 12, 1) | RandNet $_{\mathcal{B}}$ | add | 4.65 ± 0.24 | 4.56 ± 0.24 | 1.21 ± 0.24 | 49.7 ± 1.9 |
| Inter. channel dim | (32, 96, 8) | (64, 128, 8) | (64, 192, 8) | RandNet $_{\mathcal{B}}$ | concat | 4.37 ± 0.25 | 4.36 ± 0.25 | 0.91 ± 0.18 | 51.8 ± 1.8 |
| Output channel dim | (32, 96, 8) | (64, 128, 8) | (64, 192, 8) | RandNet $_{\mathcal{C}}$ | add | 9.62 ± 0.24 | 9.41 ± 0.23 | 2.01 ± 0.45 | 52.7 ± 2.2 |
| Activations | [ReLU, SiLU, Mish, GELU, LeakyReLU] | | | RandNet $_{\mathcal{C}}$ | concat | 9.46 ± 0.25 | 9.38 ± 0.25 | 1.37 ± 0.24 | 55.2 ± 1.1 |
| Normalization layers | [BatchNorm, LayerNorm] | | | RandNet $_{\mathcal{D}}$ | add | 9.63 ± 0.23 | 9.42 ± 0.22 | 2.05 ± 0.43 | 57.3 ± 1.4 |
| Kernel sizes | | [3, 5, 7, 9] | | RandNet $_{\mathcal{D}}$ | concat | 9.52 ± 0.26 | 9.42 ± 0.26 | 1.32 ± 0.24 | 58.1 ± 1.3 |
| Parameter space | | \mathcal{D} | \mathcal{E} | RandNet $_{\mathcal{E}}$ | add | 9.61 ± 0.23 | 9.41 ± 0.23 | 2.06 ± 0.44 | 58.1 ± 1.3 |
| Base space | | \mathcal{C} | \mathcal{C} | RandNet $_{\mathcal{E}}$ | concat | 9.52 ± 0.25 | 9.42 ± 0.25 | 1.31 ± 0.25 | 58.8 ± 1.5 |
| Optimizer | | - | AdamW | | | | | | |
| Data augmentation | | ✓ | ✓ | | | | | | |

(a) RandNet $_{\mathcal{A}}$

(b) RandNet $_{\mathcal{B}}$

(c) RandNet $_{\mathcal{C}}$

(d) RandNet $_{\mathcal{D}}$

(e) RandNet $_{\mathcal{E}}$

Fig. 4: Cumulative probability vs. error of trained models in Table 7 is visualized here following Radosavovic *et al.* [56]. Across all scales and settings, we observe concatenation-based models outperform those employing additive shortcuts.

diverse depths, widths, activations, normalizations, and kernel sizes to provide flexibility under constrained costs (see Table 7). Additionally, we diversify building blocks across all search spaces to conduct more extensive experiments. Three distinct architectural blocks - dubbed PreNorm, PostNorm, and PostNorm (w/o act) - are differentiated by the use of pre-activation and shortcut positions. PreNorm block adopts the pre-normalization [28, 32] precedes a skip connection. In contrast, two PostNorms enjoy post-normalization [27, 48]. PostNorm varies from PostNorm (w/o act) based on the activation function post-skip connection.

Result interpretation. Table 7 (right) exhibits that concatenation consistently outperforms additive shortcuts across all configurations. Furthermore, Fig. 4 demonstrates the superior capability of concatenation-based architectures.

5.2 Impact of Input Size on Performance

We provide compelling findings regarding versus input size. First, Fig 5 (left) shows RDNet enjoys strong adaptability to input size variations. Intriguingly, DenseNet161, even trained without strong data augmentations, still enjoys adaptability, surpassing DeiT-S trained with strong data augmentations. We attribute this to the effectiveness of dense connections.

Our finding further shows that, unlike width-oriented networks that slow with larger input sizes (due to the large intermediate tensors), our model’s

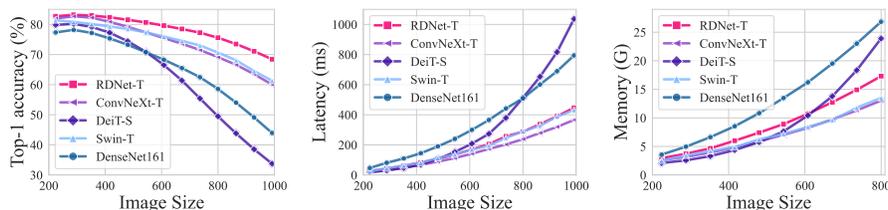


Fig. 5: Accuracy/latency/memory vs. resolution. RDNet enjoys resolution-robustness against various input image sizes to maintain accuracy. Furthermore, RDNet exhibits a similar latency/memory trend to ConvNeXt and Swin Transformer, maintaining minimal increase with larger images compared to DeiT-S and DenseNet161.

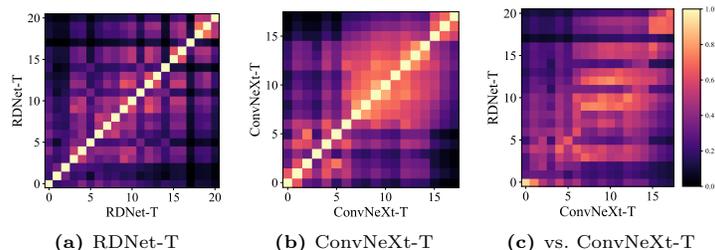


Fig. 6: CKA analysis. We compute CKA using features before passing through short-cuts (either concatenation or addition). This result suggests RDNet generates diverse features across all layers in comparison to ConvNeXt. The third column presents a direct comparison between RDNet and ConvNeXt. Overall, RDNet learns distinct features from what ConvNeXt does and more varied features.

optimized width avoids latency/memory loss. Fig 5 (middle, right) illustrates that RDNet compete with ConvNeXt and Swin Transformer, diverging from DenseNet161 [32] that gets slower and consumes more memory as image size grows. We note that larger scales (*e.g.*, -S, -B, and -L) all follow the same trend.

5.3 CKA analysis

We analyze the layer-specific features of RDNet compared to ConvNeXt using Centered Kernel Alignment (CKA) [37]. Fig. 6 displays RDNet learns distinct features at every layer, showcasing different patterns compared to ConvNeXt. In the third column, ConvNeXt and RDNet astonishingly learn different features when compared, highlighting the unique learning dynamics of each model.

5.4 Revisiting Stochastic Depth

Notably, DenseNets primitively did not employ Stochastic Depth [33] for model training due to sharing the similarity in connectivity patterns of networks. We posit that Stochastic Depth should not be overlooked; our results demonstrate a noticeable improvement when it is incorporated into our model, as illustrated

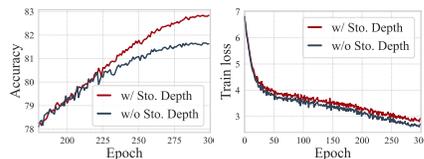


Fig. 7: Stochastic depth proves effective with dense connections. It still acts as a regularizer.

Table 8: Ablation study results are reported here with each ImageNet-1K accuracy (%) with parameter count (M) and FLOPS (G). Best options mark in gray .

| (a) Depth/width scaling | | | | (b) Block configuration | | | | (c) Expansion ratio (ER) | | | |
|-------------------------|-------|-------|-------|---|-------|-------|-------|--------------------------|-------|-------|-------|
| Depth | Param | FLOPs | Top-1 | Block conf | Param | FLOPs | Top-1 | ER | Param | FLOPs | Top-1 |
| 3, 3, 9, 3 | 23.3 | 5.0 | 82.5 | (a) RDNet-T | 23.9 | 5.0 | 82.8 | 1.0 | 24.4 | 5.0 | 82.1 |
| 3, 3, 12, 3 | 23.9 | 5.0 | 82.8 | (a) + more act | 23.9 | 5.0 | 81.9 | 2.0 | 23.8 | 5.0 | 82.6 |
| 3, 3, 15, 3 | 23.5 | 5.0 | 82.8 | (a) \leftrightarrow 3 \times 3 dwconv | 23.5 | 4.9 | 82.4 | 3.0 | 24.2 | 5.0 | 82.7 |
| 3, 3, 18, 6 | 50.3 | 8.7 | 83.5 | (a) \leftrightarrow 5 \times 5 dwconv | 23.7 | 5.0 | 82.5 | 4.0 | 23.9 | 5.0 | 82.8 |
| 3, 3, 21, 6 | 50.4 | 8.7 | 83.7 | (a) \leftrightarrow 9 \times 9 dwconv | 23.9 | 5.1 | 82.8 | 6.0 | 24.3 | 5.0 | 82.6 |
| 3, 3, 24, 6 | 49.9 | 8.7 | 83.6 | (a) \leftrightarrow 11 \times 11 dwconv | 24.4 | 5.2 | 82.7 | | | | |
| 3, 3, 18, 6 | 89.2 | 15.4 | 84.2 | (a) \leftrightarrow 13 \times 13 dwconv | 24.8 | 5.3 | 82.7 | | | | |
| 3, 3, 21, 6 | 86.2 | 15.4 | 84.4 | (b) (a) \leftrightarrow dwconv at last | 23.6 | 5.0 | 82.3 | | | | |
| 3, 3, 24, 6 | 87.4 | 15.4 | 84.2 | (b) + more act/norm | 23.6 | 5.0 | 81.1 | | | | |
| | | | | (b) LN \leftrightarrow BN | 23.6 | 5.0 | 82.2 | | | | |

| (d) Growth rate (GR) | | | | (e) Transition layer intervals | | | | (f) Transition ratio | | | |
|----------------------|-------|-------|-------|--------------------------------|-------|-------|-------|----------------------|-------|-------|-------|
| GR | Param | FLOPs | Top-1 | Interval | Param | FLOPs | Top-1 | Ratio | Param | FLOPs | Top-1 |
| 90, 90, 90, 90 | 13.2 | 5.0 | 81.6 | 2 | 24.3 | 5.0 | 82.7 | 0.3 | 24.4 | 5.0 | 82.6 |
| 120, 120, 120, 120 | 23.9 | 8.9 | 83.0 | 3 | 23.9 | 5.0 | 82.8 | 0.4 | 23.9 | 5.0 | 82.6 |
| 64, 104, 128, 224 | 23.9 | 5.0 | 82.8 | 4 | 24.1 | 5.0 | 82.6 | 0.5 | 23.9 | 5.0 | 82.8 |
| | | | | 6 | 23.7 | 5.0 | 82.1 | 0.6 | 23.8 | 5.0 | 82.5 |
| | | | | | | | | 0.7 | 23.6 | 5.0 | 82.3 |

in Fig. 7. We also observe a small stochastic depth ratio affects profoundly (see Table 9).

5.5 Ablation Studies

We gather all ablation studies in Table 8. Each table contains several models that are meticulously adjusted for almost equivalent computational costs with others to ensure a fair comparison of our specific focuses. Our methodology, in Section 3.2, methodically referenced each study.

6 Conclusion

In this paper, we have revisited the past success of DenseNet, which once outperformed ResNet in this era dominated by models using addition-based shortcuts, such as ResNet, ConvNeXt, and ViT. We have first rediscovered the potential of DenseNet, focusing on the underappreciated fact that DenseNet’s concatenation shortcuts surpass the expressivity of the convention of ResNet-style addition-based shortcuts through our pilot study. We then highlight the outdated training setups and classical macro-block designs that diminish DenseNet’s effectiveness against modernized architectures. By achieving our goal to widen DenseNet with modernized elements, we have proven that DenseNet’s foundational principles are competitive in achieving robust modeling performance on their own. Our models exhibit strong performance competitive to the latest modern architectures; the employment of diverse concatenated features has significantly enhanced performance in dense prediction tasks, showcasing an advantage overlooked in models utilizing addition shortcuts. We hope that our work sheds light on the advantages of using concatenations in network design, advocating for the consideration of DenseNet-style architectures alongside ResNet-style ones.

Table 9: Stochastic depth is compatible with dense connections.

| Ratio | Param | FLOPs | Top-1 |
|-------|-------|-------|-------|
| 0 | 23.9 | 5.0 | 81.6 |
| 0.05 | 23.9 | 5.0 | 82.5 |
| 0.10 | 23.9 | 5.0 | 82.6 |
| 0.15 | 23.9 | 5.0 | 82.8 |
| 0.20 | 23.9 | 5.0 | 82.6 |

Limitations. Our models have been scaled to a ‘-large’ level, but resource limitation prevents more extensions to upper scales such as ViT-G.

References

1. Github repository: Swin transformer for object detection, <https://github.com/SwinTransformer/Swin-Transformer-Object-Detection> 11, 22
2. Ba, J., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016) 6, 7
3. Bao, H., Dong, L., Wei, F.: Beit: Bert pre-training of image transformers. In: International Conference on Learning Representations (ICLR) (2022) 24, 27
4. Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., Tenenbaum, J., Katz, B.: Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Conference on Neural Information Processing Systems (NeurIPS). vol. 32 (2019) 21
5. Bello, I., Fedus, W., Du, X., Cubuk, E.D., Srinivas, A., Lin, T.Y., Shlens, J., Zoph, B.: Revisiting resnets: Improved training and scaling strategies. Conference on Neural Information Processing Systems (NeurIPS) 34, 22614–22627 (2021) 3, 4, 5
6. Brock, A., De, S., Smith, S.L., Simonyan, K.: High-performance large-scale image recognition without normalization. In: International Conference on Machine Learning (ICML). pp. 1059–1071 (2021) 3, 10
7. Cai, Y., Zhou, Y., Han, Q., Sun, J., Kong, X., Li, J., Zhang, X.: Reversible column networks. In: International Conference on Learning Representations (ICLR) (2023) 3, 10, 11
8. Cai, Z., Vasconcelos, N.: Cascade r-cnn: High-quality object detection and instance segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2019) 21, 24
9. Changpinyo, S., Sharma, P., Ding, N., Soricut, R.: Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In: IEEE Transactions on Computer Vision and Pattern Recognition (CVPR). pp. 3558–3568 (2021) 9
10. Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., Feng, J.: Dual path networks. Conference on Neural Information Processing Systems (NIPS) 30 (2017) 4
11. Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., Jitsev, J.: Reproducible scaling laws for contrastive language-image learning. In: IEEE Transactions on Computer Vision and Pattern Recognition (CVPR). pp. 2818–2829 (2023) 8
12. Clark, K., Luong, M.T., Le, Q.V., Manning, C.D.: Electra: Pre-training text encoders as discriminators rather than generators. In: International Conference on Learning Representations (ICLR) (2020) 24, 27
13. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: IEEE Transactions on Computer Vision and Pattern Recognition Workshop (CVPRW). pp. 702–703 (2020) 5, 11, 23, 27
14. Dai, Z., Liu, H., Le, Q.V., Tan, M.: Coatnet: Marrying convolution and attention for all data sizes. In: Conference on Neural Information Processing Systems (NeurIPS). pp. 3965–3977 (2021) 3

15. Desai, K., Kaul, G., Aysola, Z., Johnson, J.: Redcaps: Web-curated image-text data created by the people, for the people. arXiv preprint arXiv:2111.11431 (2021) [9](#)
16. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017) [5](#)
17. Ding, X., Zhang, X., Zhou, Y., Han, J., Ding, G., Sun, J.: Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In: IEEE Transactions on Computer Vision and Pattern Recognition (CVPR) (2022) [3](#), [10](#)
18. Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D., Guo, B.: Cswin transformer: A general vision transformer backbone with cross-shaped windows. In: IEEE Transactions on Computer Vision and Pattern Recognition (CVPR) (2022) [3](#), [10](#)
19. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (ICLR) (2020) [1](#), [3](#), [4](#)
20. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017) [5](#)
21. Guo, M.H., Lu, C.Z., Liu, Z.N., Cheng, M.M., Hu, S.M.: Visual attention network. Computational Visual Media (CVMJ) (2023) [2](#), [8](#), [9](#), [27](#)
22. Han, D., Kim, J., Kim, J.: Deep pyramidal residual networks. In: IEEE Transactions on Computer Vision and Pattern Recognition (CVPR). pp. 5927–5935 (2017) [4](#)
23. Han, D., Yoo, Y., Kim, B., Heo, B.: Learning features with parameter-free layers. arXiv preprint arXiv:2202.02777 (2022) [3](#)
24. Han, D., Yun, S., Heo, B., Yoo, Y.: Rethinking channel dimensions for efficient model design. In: IEEE Transactions on Computer Vision and Pattern Recognition (CVPR). pp. 732–741 (2021) [4](#), [5](#), [6](#)
25. Hassani, A., Walton, S., Li, J., Li, S., Shi, H.: Neighborhood attention transformer. In: IEEE Transactions on Computer Vision and Pattern Recognition (CVPR). pp. 6185–6194 (2023) [2](#), [3](#), [8](#), [9](#), [11](#), [22](#), [27](#)
26. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: International Conference on Computer Vision (ICCV) (2017) [10](#), [21](#), [24](#)
27. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Transactions on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016) [1](#), [3](#), [4](#), [8](#), [11](#), [12](#)
28. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. pp. 630–645. Springer (2016) [1](#), [12](#)
29. Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J., Gilmer, J.: The many faces of robustness: A critical analysis of out-of-distribution generalization. International Conference on Computer Vision (ICCV) (2021) [21](#)
30. Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., Song, D.: Natural adversarial examples. CVPR (2021) [21](#)
31. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) [6](#)

32. Huang, G., Liu, Z., Pleiss, G., Maaten, L.v.d., Weinberger, K.Q.: Convolutional networks with dense connectivity. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* **44**(12), 8704–8716 (2022) [2](#), [4](#), [5](#), [7](#), [12](#), [13](#)
33. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV* 14. pp. 646–661. Springer (2016) [5](#), [11](#), [13](#)
34. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: *European Conference on Computer Vision (ECCV)* (2016) [23](#), [24](#), [27](#)
35. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International Conference on Machine Learning (ICML)*. pp. 448–456. PMLR (2015) [6](#)
36. Jégou, S., Drozdal, M., Vazquez, D., Romero, A., Bengio, Y.: The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In: *IEEE Transactions on Computer Vision and Pattern Recognition Workshop (CVPRW)*. pp. 11–19 (2017) [2](#), [3](#)
37. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.: Similarity of neural network representations revisited. In: *International Conference on Machine Learning (ICML)*. pp. 3519–3529 (2019) [13](#)
38. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. *Communications of the ACM* **60**, 84 – 90 (2012) [1](#)
39. Le, Y., Yang, X.S.: Tiny imagenet visual recognition challenge (2015), <https://api.semanticscholar.org/CorpusID:16664790> [4](#)
40. Lee, Y., Hwang, J.w., Lee, S., Bae, Y., Park, J.: An energy and gpu-computation efficient backbone network for real-time object detection. In: *IEEE Transactions on Computer Vision and Pattern Recognition Workshop (CVPRW)*. pp. 752–760 (2019) [2](#), [4](#), [5](#)
41. Lee, Y., Park, J.: Centermask: Real-time anchor-free instance segmentation. In: *IEEE Transactions on Computer Vision and Pattern Recognition (CVPR)* (2020) [7](#)
42. Li, S., Wang, Z., Liu, Z., Tan, C., Lin, H., Wu, D., Chen, Z., Zheng, J., Li, S.Z.: Moganet: Multi-order gated aggregation network. In: *International Conference on Learning Representations (ICLR)* (2024), <https://openreview.net/forum?id=XhYWgjqCrV> [2](#), [3](#), [9](#), [27](#)
43. Li, Y., Chen, Y., Dai, X., Chen, D., Liu, M., Yuan, L., Liu, Z., Zhang, L., Vasconcelos, N.: Micronet: Improving image recognition with extremely low flops. In: *International Conference on Computer Vision (ICCV)*. pp. 468–477 (2021) [3](#), [10](#)
44. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: *European Conference on Computer Vision (ECCV)*. pp. 740–755. Springer (2014) [10](#)
45. Lin, W., Wu, Z., Chen, J., Huang, J., Jin, L.: Scale-aware modulation meet transformer. In: *International Conference on Computer Vision (ICCV)*. pp. 5992–6003 (10 2023) [2](#), [8](#), [9](#), [27](#)
46. Liu, S., Chen, T., Chen, X., Chen, X., Xiao, Q., Wu, B., Pechenizkiy, M., Mocanu, D.C., Wang, Z.: More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. In: *International Conference on Learning Representations (ICLR)* (2023) [3](#), [10](#), [22](#)
47. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: *International Conference on Computer Vision (ICCV)* (2021) [1](#), [2](#), [3](#), [4](#), [5](#), [8](#), [10](#), [11](#), [22](#)

48. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: *IEEE Transactions on Computer Vision and Pattern Recognition (CVPR)*. pp. 11976–11986 (2022) [1](#), [2](#), [3](#), [4](#), [5](#), [6](#), [7](#), [8](#), [10](#), [11](#), [12](#), [22](#), [24](#), [25](#)
49. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016) [5](#)
50. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: *International Conference on Learning Representations (ICLR)* (2019) [5](#), [8](#), [11](#)
51. Mingfei Ma, Vitaly Fedyunin, W.W.: Accelerating pytorch vision models with channels last on cpu (2022), <https://pytorch.org/blog/accelerating-pytorch-vision-models-with-channels-last-on-cpu/> [25](#)
52. Parihar, A.S., Java, A.: Densely connected convolutional transformer for single image dehazing. *Journal of Visual Communication and Image Representation* p. 103722 (2023) [3](#)
53. Pleiss, G., Chen, D., Huang, G., Li, T., van der Maaten, L., Weinberger, K.Q.: Memory-efficient implementation of densenets. arXiv preprint arXiv:1707.06990 (2017) [2](#), [4](#)
54. Polyak, B., Juditsky, A.B.: Acceleration of stochastic approximation by averaging. *Siam Journal on Control and Optimization* **30**, 838–855 (1992) [27](#)
55. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: *International Conference on Machine Learning (ICML)*. pp. 8748–8763. PMLR (2021) [8](#)
56. Radosavovic, I., Kosaraju, R.P., Girshick, R.B., He, K., Dollár, P.: Designing network design spaces. In: *IEEE Transactions on Computer Vision and Pattern Recognition (CVPR)*. pp. 10425–10433 (2020) [3](#), [5](#), [10](#), [12](#), [24](#), [25](#)
57. Rao, Y., Zhao, W., Tang, Y., Zhou, J., Lim, S.N., Lu, J.: Hornet: Efficient high-order spatial interactions with recursive gated convolutions. In: *Conference on Neural Information Processing Systems (NeurIPS)* (2022) [2](#), [3](#), [7](#), [8](#), [9](#), [22](#), [27](#)
58. Recht, B., Roelofs, R., Schmidt, L., Shankar, V.: Do imagenet classifiers generalize to imagenet? In: *International Conference on Machine Learning (ICML)* (2019) [21](#)
59. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)* **115**(3), 211–252 (2015) [2](#), [8](#)
60. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: *IEEE Transactions on Computer Vision and Pattern Recognition (CVPR)*. pp. 4510–4520 (2018) [6](#)
61. Sharma, P., Ding, N., Goodman, S., Soricut, R.: Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 2556–2565 (2018) [9](#)
62. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) [1](#)
63. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)* **15**(56), 1929–1958 (2014) [23](#)
64. Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., Beyer, L.: How to train your vit? data, augmentation, and regularization in vision transformers. arXiv preprint arXiv:2106.10270 (2021) [3](#)

65. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Transactions on Computer Vision and Pattern Recognition (CVPR). pp. 1–9 (2015) [1](#), [3](#)
66. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. IEEE Transactions on Computer Vision and Pattern Recognition (CVPR) pp. 2818–2826 (2016) [5](#), [11](#), [23](#), [27](#)
67. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning (ICML). pp. 6105–6114. PMLR (2019) [1](#), [3](#), [6](#)
68. Tan, M., Le, Q.V.: Efficientnetv2: Smaller models and faster training. In: International Conference on Machine Learning (ICML) (2021) [5](#), [6](#)
69. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jegou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning (ICML). pp. 10347–10357 (2021) [3](#), [5](#), [7](#), [10](#)
70. Touvron, H., Cord, M., El-Nouby, A., Bojanowski, P., Joulin, A., Synnaeve, G., Verbeek, J., J’egou, H.: Augmenting convolutional networks with attention-based aggregation. arXiv preprint arXiv:2112.13692 (2021) [3](#), [7](#)
71. Touvron, H., Cord, M., J’egou, H.: Deit iii: Revenge of the vit. In: European Conference on Computer Vision (ECCV) (2022) [2](#), [10](#), [11](#)
72. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., J’egou, H.: Going deeper with image transformers. International Conference on Computer Vision (ICCV) pp. 32–42 (2021) [27](#)
73. Trockman, A., Kolter, J.Z.: Patches are all you need? arXiv preprint arXiv:2201.09792 (2022) [3](#)
74. Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A.C., Li, Y.: Maxvit: Multi-axis vision transformer. In: European Conference on Computer Vision (ECCV) (2022) [3](#)
75. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Conference on Neural Information Processing Systems (NIPS) (2017) [1](#), [3](#)
76. Wang, C.Y., Liao, H.y., Wu, Y.H., Chen, P.Y., Hsieh, J.W., Yeh, I.H.: Cspnet: A new backbone that can enhance learning capability of cnn. In: IEEE Transactions on Computer Vision and Pattern Recognition (CVPR). pp. 1571–1580 (2020) [2](#)
77. Wang, H., Ge, S., Lipton, Z., Xing, E.P.: Learning robust global representations by penalizing local predictive power. In: Conference on Neural Information Processing Systems (NeurIPS). pp. 10506–10518 (2019) [21](#)
78. Wang, L., Cao, M., Yuan, X.: Efficientsci: Densely connected network with space-time factorization for large-scale video snapshot compressive imaging. In: IEEE Transactions on Computer Vision and Pattern Recognition (CVPR). pp. 18477–18486 (2023) [3](#)
79. Wang, R.J., Li, X., Ling, C.X.: Pelee: a real-time object detection system on mobile devices. In: Conference on Neural Information Processing Systems (NeurIPS). p. 1967–1976 (2018) [2](#), [3](#), [5](#)
80. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: International Conference on Computer Vision (ICCV). pp. 548–558 (2021) [3](#), [11](#)
81. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pvtv2: Improved baselines with pyramid vision transformer. Computational Visual Media (CVMJ) (2022) [3](#), [10](#), [22](#)

82. Wang, Z., Xie, K., Zhang, X.Y., Chen, H.Q., Wen, C., He, J.: Small-object detection based on yolo and dense block via image super-resolution. *IEEE Access* **9**, 56416–56429 (2021) [3](#)
83. Wightman, R.: Github repository: Pytorch image models, <https://github.com/huggingface/pytorch-image-models> [24](#)
84. Wightman, R., Touvron, H., Jégou, H.: Resnet strikes back: An improved training procedure in timm. <https://github.com/huggingface/pytorch-image-models> (2021) [3](#), [4](#), [5](#), [10](#)
85. Wu, K., Zhang, J., Peng, H., Liu, M., Xiao, B., Fu, J., Yuan, L.: Tinyvit: Fast pretraining distillation for small vision transformers. In: *European Conference on Computer Vision (ECCV)* (2022) [11](#)
86. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: *European Conference on Computer Vision (ECCV)*. Springer (2018) [9](#), [24](#)
87. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: *IEEE Transactions on Computer Vision and Pattern Recognition (CVPR)*. pp. 1492–1500 (2017) [1](#)
88. Yang, J., Li, C., Dai, X., Gao, J.: Focal modulation networks. In: *Conference on Neural Information Processing Systems (NeurIPS)* (2022) [3](#), [10](#), [22](#)
89. Yang, J., Li, C., Zhang, P., Dai, X., Xiao, B., Yuan, L., Gao, J.: Focal self-attention for local-global interactions in vision transformers. In: *Conference on Neural Information Processing Systems (NeurIPS)* (2021) [3](#)
90. Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., Yan, S.: Metaformer is actually what you need for vision. In: *IEEE Transactions on Computer Vision and Pattern Recognition (CVPR)*. pp. 10819–10829 (2022) [3](#)
91. Yu, W., Zhou, P., Yan, S., Wang, X.: Inceptionnext: When inception meets convnext. *arXiv preprint arXiv:2303.16900* (2023) [3](#), [10](#)
92. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: *International Conference on Computer Vision (ICCV)*. pp. 6023–6032 (2019) [5](#), [11](#), [23](#), [27](#)
93. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: *International Conference on Learning Representations (ICLR)* (2018) [5](#), [11](#), [23](#), [27](#)
94. Zhang, J., Jin, Y., Xu, J., Xu, X., Zhang, Y.: Mdu-net: Multi-scale densely connected u-net for biomedical image segmentation. *Health Information Science and Systems (HISS)* (2023) [3](#), [4](#)
95. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: *AAAI Conference on Artificial Intelligence (AAAI)*. pp. 13001–13008 (2020) [5](#), [11](#), [23](#), [27](#)
96. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision (IJCV)* **127**, 302–321 (2018) [9](#)
97. Zhu, L., Wang, X., Ke, Z., Zhang, W., Lau, R.: Biformer: Vision transformer with bi-level routing attention. *IEEE Transactions on Computer Vision and Pattern Recognition (CVPR)* (2023) [2](#), [3](#), [8](#), [9](#), [27](#)

Appendix

In this Appendix, we provide additional experiments and details to complement the main paper. The contents are as follows:

- §A benchmarks robustness of ImageNet-1K pre-trained models on the benchmarks, including out-of-distribution datasets including ImageNet-V2 [58], ObjectNet [4], ImageNet-A [30], ImageNet-Sketch [77], and ImageNet-R [29];
- §B reports further COCO object detection and instance segmentation results with Cascade Mask-RCNN [8];
- §C evaluates further ImageNet top-accuracy versus latency trade-offs on various testbeds, encompassing both PyTorch and TensorRT A100 inference, as well as CPU inference outcomes;
- §D provides more details, including specific results and setups of our pilot study described in §5.1;
- §E presents our experimental setups for ImageNet and downstream tasks training and evaluation setups.

A Robustness Evaluation

We further evaluate the robustness of our models using the ImageNet out-of-distribution (OOD) benchmarks - ImageNet-V2 [58], ImageNet-A [30], ImageNet-Sketch [77], ImageNet-R [29], and ObjectNet [4]. Table A shows our RDNet demonstrates superior robustness in comparison to the other models. We specifically select models (HorNet, SLaK, and NAT) having comparable ImageNet-1K accuracies to demonstrate the superior out-of-distribution (OOD) performance of our models compared with those. Nobaly, even when RDNet demonstrates lower accuracy on ImageNet-1K than competing models, RDNet achieves high OOD scores across various benchmarks.

B Object Detection with Cascade Mask-RCNN

An extension to the Mask-RCNN [26] results in Table 6 in the main paper, we employ the Cascade Mask-RCNN head [8] to evaluate our pre-trained models further. As demonstrated in Table B, RDNet exhibits competitive performance. Note that our models do not experience exhaustive fine-tuning of training hyperparameters for maximum precisions compared with ConvNeXt’s, which indicates additional potential for achieving higher accuracy.

C Further ImageNet Accuracy vs. Latency Trade-offs

To assess the practicality of our models, we measure speeds across diverse testbeds. We precisely measure inference speeds using the PyTorch framework on NVIDIA A100 GPU, Intel Xeon Gold 5120 CPU, and TensorRT Inference Engine on

Table A: Robustness evaluation. We compare the models evaluating the out-of-distribution (OOD) metrics ImageNet-V2/A/Sketch/ObjNet/R. We further average the OOD scores to show the averaged distribution shifts denoted by **Avg Shift**. Interestingly, even when RDNet demonstrates lower accuracy on ImageNet-1K compared to other networks, it consistently attains high OOD scores compared with other datasets.

| Model | Params | FLOPs | IN | Avg Shift | V2 | Obj | A | Sketch | R |
|-----------------|--------|-------|------|-----------|------|------|------|--------|------|
| Swin-T [47] | 28 | 4.5 | 81.3 | 38.9 | 69.7 | 33.1 | 21.1 | 29.3 | 41.5 |
| ConvNeXt-T [48] | 29 | 4.5 | 82.1 | 42.7 | 72.5 | 35.6 | 24.2 | 33.8 | 47.2 |
| HorNet-T [57] | 22 | 4.0 | 82.8 | 43.4 | 72.3 | 37.5 | 26.6 | 34.1 | 46.6 |
| SLaK-T [46] | 30 | 5.0 | 82.5 | 43.3 | 72.0 | 36.6 | 30.0 | 32.4 | 45.3 |
| NAT-T [25] | 28 | 4.3 | 83.2 | 44.0 | 72.2 | 37.8 | 33.0 | 31.9 | 44.9 |
| RDNet-T | 24 | 5.0 | 82.8 | 44.7 | 72.9 | 36.9 | 27.7 | 37.0 | 49.0 |
| Swin-S [47] | 50 | 8.7 | 83.0 | 43.8 | 72.0 | 36.8 | 32.5 | 32.3 | 45.2 |
| ConvNeXt-S [48] | 50 | 8.7 | 83.1 | 45.7 | 72.5 | 38.0 | 31.3 | 37.1 | 49.6 |
| HorNet-S [57] | 50 | 8.8 | 84.0 | 47.3 | 73.6 | 39.9 | 36.2 | 36.9 | 49.7 |
| SLaK-S [46] | 55 | 9.8 | 83.8 | 48.2 | 73.6 | 39.6 | 39.3 | 37.5 | 50.9 |
| NAT-S [25] | 51 | 7.8 | 83.7 | 46.4 | 73.2 | 39.9 | 37.4 | 34.3 | 47.3 |
| RDNet-S | 50 | 8.7 | 83.7 | 47.8 | 73.8 | 39.3 | 33.5 | 39.8 | 52.8 |
| Swin-B [47] | 88 | 15.4 | 83.5 | 44.9 | 72.4 | 37.6 | 35.4 | 32.7 | 46.5 |
| ConvNeXt-B [48] | 89 | 15.4 | 83.8 | 47.9 | 73.7 | 39.9 | 36.7 | 38.2 | 51.2 |
| HorNet-B [57] | 87 | 15.6 | 84.3 | 48.8 | 73.9 | 41.0 | 39.9 | 38.1 | 51.2 |
| SLaK-B [46] | 95 | 17.1 | 84.0 | 48.9 | 74.0 | 39.7 | 41.6 | 38.5 | 50.8 |
| NAT-B [25] | 90 | 13.7 | 84.3 | 48.5 | 74.1 | 40.7 | 41.4 | 36.6 | 49.7 |
| RDNet-B | 87 | 15.4 | 84.4 | 49.0 | 74.2 | 39.7 | 38.1 | 40.1 | 52.7 |
| ConvNeXt-L [48] | 198 | 34.4 | 84.3 | 49.9 | 74.2 | 40.6 | 41.3 | 40.1 | 53.5 |
| RDNet-L | 186 | 34.7 | 84.8 | 52.2 | 75.0 | 42.1 | 42.9 | 44.5 | 56.5 |

Table B: COCO object detection and segmentation results. We utilize Cascade Mask-RCNN with 3x schedule. FLOPs (G) are calculated with image size (1280, 800). The result of Swin-T is from the official repository [1].

| Backbone | Param | FLOPs | AP ^{box} | AP ₅₀ ^{box} | AP ₇₅ ^{box} | AP ^{mask} | AP ₅₀ ^{mask} | AP ₇₅ ^{mask} |
|-----------------|-------|-------|-------------------|---------------------------------|---------------------------------|--------------------|----------------------------------|----------------------------------|
| Swin-T [47] | 86M | 745G | 50.4 | 69.2 | 54.7 | 43.7 | 66.6 | 47.3 |
| PVTv2-B2 [81] | 83M | 788G | 51.1 | 69.8 | 55.3 | - | - | - |
| FocalNet-T [88] | 86M | 746G | 51.5 | 70.1 | 55.8 | - | - | - |
| ConvNeXt-T [48] | 86M | 741G | 50.4 | 69.1 | 54.8 | 43.7 | 66.5 | 47.3 |
| NAT-T [25] | 85M | 737G | 51.4 | 70.0 | 55.9 | 44.5 | 67.6 | 47.9 |
| RDNet-T | 81M | 757G | 51.6 | 70.5 | 56.0 | 44.6 | 67.9 | 48.3 |
| Swin-S [47] | 107M | 838G | 51.9 | 70.7 | 56.3 | 45.0 | 68.2 | 48.8 |
| ConvNeXt-S [48] | 108M | 827G | 51.9 | 70.8 | 56.5 | 45.0 | 68.4 | 49.1 |
| NAT-S [25] | 108M | 809G | 52.0 | 70.4 | 56.3 | 44.9 | 68.1 | 48.6 |
| RDNet-S | 108M | 832G | 52.3 | 70.8 | 56.6 | 45.3 | 68.5 | 49.3 |
| Swin-B [47] | 145M | 982G | 51.9 | 70.5 | 56.4 | 45.0 | 68.1 | 48.9 |
| ConvNeXt-B [48] | 146M | 964G | 52.7 | 71.3 | 57.2 | 45.6 | 68.9 | 49.5 |
| NAT-B [25] | 147M | 931G | 52.5 | 71.1 | 57.1 | 45.2 | 68.6 | 49.0 |
| RDNet-B | 144M | 971G | 52.8 | 71.4 | 57.0 | 45.6 | 68.9 | 49.5 |

NVIDIA A100 GPU. Our testing environment incorporates PyTorch version 1.13.1, CUDA version 11.6, and TensorRT version 8.5.3 for these experiments. **A** shows that our models consistently show superior accuracy vs. latency trade-offs across all evaluation setups. Note that (b) in **A** includes fewer models because

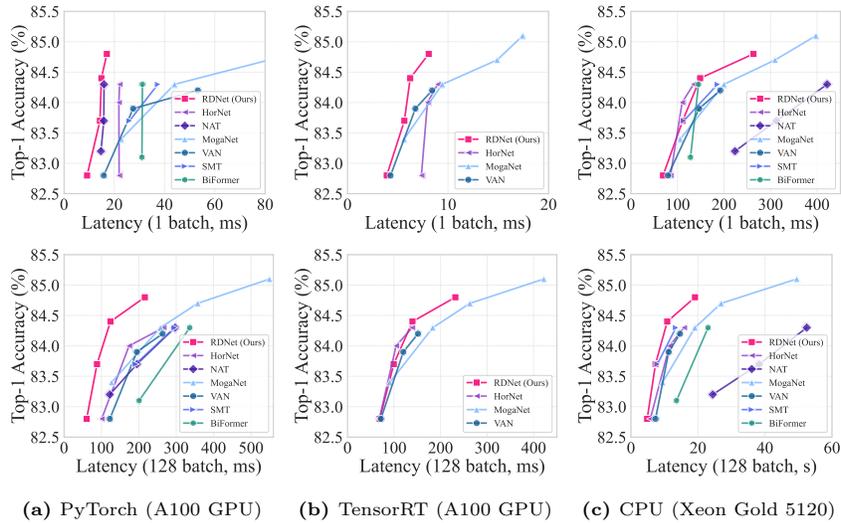


Fig. A: Further trade-offs in ImageNet-1K performance. We provide comparative visualizations with diverse environments between *state-of-the-art models*, which were known for top-performing models. It turns out that RDNet is highly competitive in practice in terms of model speed.

those that are challenging to convert to inference engines due to factors like the use of CUDA custom kernels are not evaluated. We report all the numbers in Table F.

D More Details of Our Pilot Study

We present individual RandNet experimental results performed under controlled setups. We conduct 200 experiments for each configuration of budget, block type, and skip connection type. Fig. B and Table C, concatenation outperforms addition across parameter spaces \mathcal{A} , \mathcal{B} , and \mathcal{C} . For the parameter spaces \mathcal{D} and \mathcal{E} using data augmentations, we use the following hyper-parameters for experiments. For RandAugment [13], we limit the magnitude values of {3, 5, 7}; for MixUp [93] and CutMix [92], we employ alpha values of {0.1, 0.3, 0.5} respectively; DropOut [63] ratio and Stochastic Depth [34] ratio are set to to {0.1, 0.2} and {0.05, 0.1}, respectively; label smoothing [66] is fixed to 0.1; and Random Erasing [95] is also fixed to 0.25.

Due to the diverse setups in each data argumentation, we conduct 600 experiments for each element. Additionally, even when switching the optimizer to AdamW, we train 600 random networks for every augmentation as well. Fig. C and Table D suggest that that data augmentation reduces the performance gap between additive and concatenative shortcuts, particularly noting that the element (*i.e.*, trained using stochastic depth with AdamW) benefit more from

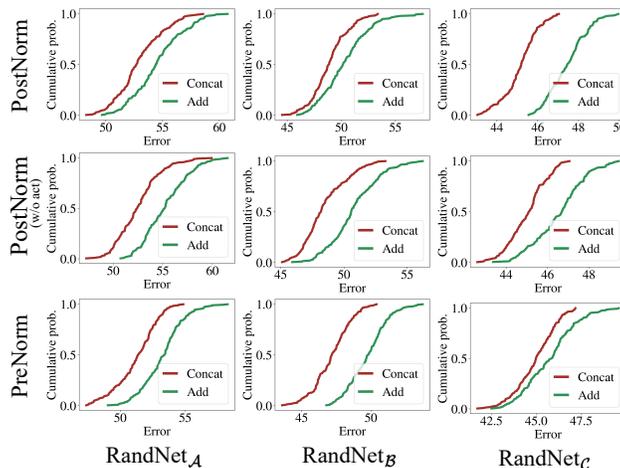


Fig. B: Cumulative probability vs. error of trained models in Table C is visualized here following Radosavovic *et al.* [56]. Each row demonstrates three block types, and each column exhibits parameter spaces \mathcal{A} , \mathcal{B} , and \mathcal{C} .

an additive shortcut. Nevertheless, we continue to observe that concatenation-based models dominate over advantage over additive shortcuts, even in AdamW training configurations.

E Experimental Settings

E.1 ImageNet Training

In Table E, we present the training configurations for RDNet on ImageNet-1K. Each variant of RDNet adheres to these settings, with the exception of the stochastic depth rate [34], which is tailored to each model variant. For fine-tuning, we group three consecutive feature mixer blocks for layer-wise learning rate decay [3, 12] akin to the approach taken in ConvNeXt. We use the timm python package [83] for model training.

E.2 Downstream Tasks

We adhere to the hyper-parameter sweep protocol outlined in [48] but sweep much lightly. For UperNet [86] training on ADE20K, we explore the following hyperparameters: learning rate $\{8e-5, 1e-3\}$, weight decay $\{0.01, 0.03, 0.05\}$. For Mask-RCNN [26] training on COCO, we explore hyperparameters such as learning rate $\{2e-3, 3e-3\}$, weight decay $\{0.05, 0.1\}$, stochastic depth $\{0.1, 0.2\}$. For Cascade Mask-RCNN [8] training on COCO, we explore hyperparameters such as learning rate $\{8e-5, 1e-4\}$, weight decay $\{0.05, 0.1\}$, stochastic depth $\{0.4, 0.5, 0.6\}$, and layer-wise learning rate decay $\{0.7, 0.8\}$. We note that our

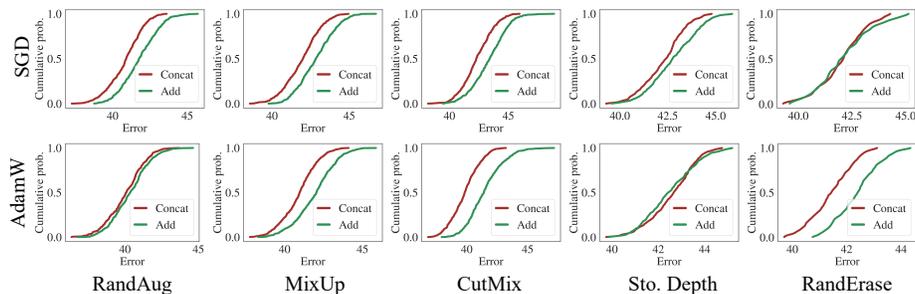


Fig. C: Cumulative probability vs. error of trained models in Table D is visualized here following Radosavovic *et al.* [56]. The figure corresponds to the manuscript’s parameter spaces \mathcal{D} and \mathcal{E} . Each row demonstrates optimizers, and each column exhibits augmentations.

search space is similar to or less extensive than the known search space in ConvNeXt [48] (*e.g.*, 6 (ours) vs. 12 (ConvNeXt) for ADE20K and 8/24 (ours) vs. 48 (ConvNeXt) for COCO experiments).

E.3 Benchmark Settings

We measure latency and memory on the V100 GPU utilizing PyTorch 1.13.1 and CUDA 11.6. In all measurements, we employ the channels-last memory format [51]. Memory is measured in the training phase with a batch size of 16.

Table C: Concatenation vs. addition. The table corresponds to the manuscript’s parameter spaces \mathcal{A} , \mathcal{B} , and \mathcal{C} . We sample 200 random networks within each parameter space, ensuring similar computational costs of FLOPs, the number of parameters (Param), and activations (Act), and individually train them on Tiny-ImageNet. All results are averaged along with the standard deviation (\pm std). Experimental results with higher accuracy are shaded in gray .

| Model | Skip type | Block type | FLOPs (G) | Param (M) | Act (M) | Top-1 (%) |
|----------------------|-----------|--------------------|-----------|-----------|-----------|-----------------|
| RandNet _A | Add | PostNorm | 2.29±0.12 | 2.25±0.12 | 0.77±0.13 | 45.4±2.2 |
| RandNet _A | Concat | PostNorm | 2.18±0.11 | 2.19±0.11 | 0.60±0.11 | 47.1±2.1 |
| RandNet _A | Add | PostNorm (w/o act) | 2.28±0.12 | 2.24±0.12 | 0.76±0.13 | 44.7±2.3 |
| RandNet _A | Concat | PostNorm (w/o act) | 2.19±0.12 | 2.20±0.12 | 0.62±0.11 | 47.5±2.2 |
| RandNet _A | Add | PreNorm | 2.36±0.08 | 2.32±0.08 | 0.80±0.12 | 46.7±1.6 |
| RandNet _A | Concat | PreNorm | 2.09±0.07 | 2.10±0.07 | 0.61±0.12 | 48.8±1.7 |
| RandNet _B | Add | PostNorm | 4.63±0.25 | 4.54±0.24 | 1.25±0.24 | 49.7±2.2 |
| RandNet _B | Concat | PostNorm | 4.41±0.27 | 4.40±0.27 | 0.88±0.17 | 51.1±1.7 |
| RandNet _B | Add | PostNorm (w/o act) | 4.58±0.25 | 4.49±0.25 | 1.17±0.24 | 49.2±2.0 |
| RandNet _B | Concat | PostNorm (w/o act) | 4.44±0.26 | 4.42±0.26 | 0.92±0.18 | 51.6±1.7 |
| RandNet _B | Add | PreNorm | 4.46±0.24 | 4.37±0.23 | 1.26±0.24 | 51.3±1.7 |
| RandNet _B | Concat | PreNorm | 4.43±0.25 | 4.42±0.25 | 0.86±0.17 | 52.2±1.4 |
| RandNet _C | Add | PostNorm | 9.67±0.23 | 9.46±0.23 | 2.02±0.46 | 51.4±2.2 |
| RandNet _C | Concat | PostNorm | 9.44±0.25 | 9.36±0.25 | 1.39±0.23 | 55.2±1.0 |
| RandNet _C | Add | PostNorm (w/o act) | 9.58±0.24 | 9.38±0.24 | 1.95±0.45 | 52.6±2.0 |
| RandNet _C | Concat | PostNorm (w/o act) | 9.44±0.25 | 9.36±0.24 | 1.43±0.23 | 55.2±1.1 |
| RandNet _C | Add | PreNorm | 9.55±0.21 | 9.34±0.21 | 2.11±0.42 | 54.5±1.6 |
| RandNet _C | Concat | PreNorm | 9.52±0.25 | 9.42±0.25 | 1.30±0.24 | 55.1±1.1 |

Table D: Concatenation vs. addition with data augmentations. The table corresponds to the manuscript’s parameter spaces \mathcal{D} and \mathcal{E} . We utilize the PreNorm block for augmentation experiments. We sample 600 random networks due to diverse degrees of data augmentations and report identically in Table C. Experimental results with higher accuracy are shaded in gray .

| Skip type | Augmentation | AdamW | FLOPs (G) | Param (M) | Act (M) | Top-1 (%) |
|-----------|--------------|-------|-----------|-----------|-----------|-----------------|
| Add | RandAug | | 9.65±0.23 | 9.44±0.22 | 2.05±0.43 | 58.2±1.3 |
| Concat | RandAug | | 9.50±0.26 | 9.40±0.25 | 1.33±0.24 | 59.2±1.2 |
| Add | MixUp | | 9.62±0.22 | 9.41±0.22 | 2.03±0.44 | 57.1±1.2 |
| Concat | MixUp | | 9.52±0.27 | 9.42±0.27 | 1.32±0.24 | 58.1±1.2 |
| Add | CutMix | | 9.64±0.22 | 9.43±0.22 | 2.06±0.43 | 56.6±1.5 |
| Concat | CutMix | | 9.51±0.27 | 9.42±0.27 | 1.33±0.24 | 57.7±1.3 |
| Add | Sto. Depth | | 9.61±0.23 | 9.40±0.23 | 2.05±0.44 | 57.2±1.3 |
| Concat | Sto. Depth | | 9.53±0.26 | 9.43±0.26 | 1.33±0.25 | 57.7±1.1 |
| Add | RandErase | | 9.58±0.22 | 9.38±0.22 | 2.07±0.44 | 57.9±1.3 |
| Concat | RandErase | | 9.53±0.25 | 9.43±0.25 | 1.26±0.23 | 58.0±1.1 |
| Add | RandAug | ✓ | 9.59±0.23 | 9.38±0.23 | 2.13±0.44 | 59.7±1.3 |
| Concat | RandAug | ✓ | 9.55±0.26 | 9.44±0.26 | 1.28±0.25 | 60.0±1.3 |
| Add | MixUp | ✓ | 9.61±0.23 | 9.40±0.22 | 2.05±0.43 | 58.1±1.3 |
| Concat | MixUp | ✓ | 9.49±0.24 | 9.39±0.24 | 1.31±0.23 | 59.1±1.2 |
| Add | CutMix | ✓ | 9.61±0.24 | 9.40±0.23 | 2.04±0.43 | 58.5±1.5 |
| Concat | CutMix | ✓ | 9.52±0.26 | 9.42±0.26 | 1.33±0.25 | 60.2±1.2 |
| Add | Sto. Depth | ✓ | 9.63±0.22 | 9.42±0.22 | 2.07±0.45 | 57.5±1.1 |
| Concat | Sto. Depth | ✓ | 9.53±0.26 | 9.43±0.25 | 1.30±0.24 | 57.4±1.0 |
| Add | RandErase | ✓ | 9.65±0.23 | 9.44±0.22 | 2.00±0.42 | 57.6±0.8 |
| Concat | RandErase | ✓ | 9.48±0.25 | 9.39±0.25 | 1.36±0.24 | 58.6±0.8 |

Table E: ImageNet-1K training settings. Most training setups are consistently used except for the multiple stochastic depth rates (e.g., 0.15/0.35/0.4/0.45) that regularize the corresponding models (e.g., RDNet-T/S/B/L), respectively.

| | RDNet-T/S/B/L (Pre-)Training | RDNet-L Fine-Tuning |
|---|---------------------------------|------------------------|
| image size | 224 | 384 |
| weight init | kaiming normal | pre-trained |
| optimizer | AdamW | AdamW |
| base learning rate | 1e-3 | 2e-5 |
| weight decay | 0.05 | 1e-8 |
| optimizer momentum (β_1, β_2) | 0.9, 0.999 | 0.9, 0.999 |
| batch size | 512 | 512 |
| training epochs | 300 | 30 |
| learning rate schedule | cosine decay | cosine decay |
| warmup epochs | 20 | 5 |
| warmup schedule | linear | linear |
| layer-wise lr decay [3, 12] | None | 0.7 |
| randaugment [13] | (9, 0.5) | (9, 0.5) |
| mixup [93] | 0.8 | 0.0 |
| cutmix [92] | 1.0 | 0.0 |
| random erasing [95] | 0.25 | 0.25 |
| label smoothing [66] | 0.1 | 0.1 |
| stochastic depth [34] | 0.15/0.35/0.4/0.5 | 0.6 |
| layer scale [72] | 1e-6 | pre-trained |
| head init scale [72] | None | 1e-3 |
| gradient clip | None | None |
| center crop percent | 0.9 | 1.0 |
| exp. mov. avg. (EMA) [54] | None | None |

Table F: ImageNet-1K comparison with the latest models. Fig. A visualized this table. We thoroughly compare our models against the latest architectures in practical latencies. *bn* denotes latency, measured with a batch size of *n*. Certain models were excluded from evaluation because their CUDA custom kernels were not compiled.

| Model | Date | Param FLOPs Top-1 | | | PyTorch (A100, ms) | | | | TensorRT (A100, ms) | | | | PyTorch (Xeon 5120, s) | | | |
|------------------------------|--------------|-------------------|------|------|--------------------|------|-------|-------|---------------------|------|-------|-------|------------------------|------|-------|-------|
| | | (M) | (G) | (%) | b1 | b8 | b32 | b128 | b1 | b8 | b32 | b128 | b1 | b8 | b32 | b128 |
| RDNet-T | Ours | 24 | 5.0 | 82.8 | 9.2 | 9.2 | 17.8 | 60.5 | 3.9 | 7.5 | 19.7 | 69.3 | 0.07 | 0.25 | 1.09 | 4.85 |
| HorNet-T _{7×7} [57] | NeurIPS'2022 | 22 | 4.0 | 82.8 | 21.9 | 21.9 | 27.8 | 100.7 | 7.4 | 10.6 | 22.8 | 68.4 | 0.09 | 0.21 | 0.89 | 5.79 |
| VAN-B2 [21] | CVMJ'2023 | 27 | 5.0 | 82.8 | 15.8 | 16.4 | 32.9 | 122.5 | 4.1 | 8.6 | 21.2 | 71.8 | 0.08 | 0.38 | 1.61 | 7.33 |
| BiFormer-S [97] | CVPR'2023 | 26 | 4.5 | 83.8 | 31.0 | 35.3 | 54.2 | 200.9 | - | - | - | - | 0.13 | 0.39 | 2.17 | 13.60 |
| NAT-T [25] | CVPR'2023 | 28 | 4.3 | 83.2 | 14.7 | 14.7 | 32.9 | 122.4 | - | - | - | - | 0.22 | 1.43 | 5.73 | 24.42 |
| SMT-S [45] | ICCV'2023 | 21 | 4.7 | 83.7 | 26.1 | 26.4 | 53.0 | 191.4 | - | - | - | - | 0.11 | 0.32 | 1.16 | 7.91 |
| MogaNet-S [42] | ICLR'2024 | 25 | 5.0 | 83.4 | 22.7 | 22.7 | 34.9 | 127.2 | 5.6 | 10.3 | 27.6 | 91.0 | 0.11 | 0.42 | 1.97 | 9.46 |
| RDNet-S | Ours | 50 | 8.7 | 83.7 | 14.3 | 14.4 | 26.4 | 88.3 | 5.7 | 10.8 | 29.3 | 99.4 | 0.11 | 0.38 | 1.73 | 7.34 |
| HorNet-S _{7×7} [57] | NeurIPS'2022 | 50 | 8.8 | 84.0 | 21.8 | 22.2 | 46.9 | 173.9 | 8.0 | 14.1 | 33.6 | 104.5 | 0.11 | 0.38 | 2.43 | 11.51 |
| VAN-B3 [21] | CVMJ'2023 | 45 | 9.0 | 83.9 | 27.5 | 32.8 | 50.4 | 194.8 | 6.8 | 13.7 | 34.5 | 119.9 | 0.15 | 0.59 | 2.50 | 11.27 |
| BiFormer-B [97] | CVPR'2023 | 57 | 9.8 | 84.3 | 31.2 | 31.1 | 89.3 | 336.3 | - | - | - | - | 0.15 | 0.78 | 4.90 | 23.03 |
| NAT-S [25] | CVPR'2023 | 51 | 7.8 | 83.7 | 15.8 | 20.6 | 51.9 | 194.9 | - | - | - | - | 0.31 | 2.10 | 8.78 | 38.27 |
| SMT-B [45] | ICCV'2023 | 32 | 7.7 | 84.3 | 37.4 | 39.1 | 81.2 | 295.6 | - | - | - | - | 0.19 | 0.60 | 2.24 | 13.41 |
| MogaNet-B [42] | ICLR'2024 | 44 | 9.9 | 84.3 | 44.0 | 47.4 | 70.6 | 258.0 | 9.4 | 19.0 | 53.4 | 183.4 | 0.20 | 0.77 | 4.08 | 19.10 |
| RDNet-B | Ours | 87 | 15.4 | 84.4 | 14.9 | 15.1 | 36.0 | 124.2 | 6.2 | 13.6 | 39.5 | 139.8 | 0.15 | 0.58 | 2.52 | 10.84 |
| HorNet-B _{7×7} [57] | NeurIPS'2022 | 87 | 15.6 | 84.3 | 22.1 | 22.3 | 68.9 | 266.1 | 9.0 | 16.5 | 41.8 | 139.1 | 0.13 | 0.57 | 3.43 | 15.86 |
| VAN-B4 [21] | CVMJ'2023 | 60 | 12.2 | 84.2 | 53.3 | 53.6 | 80.7 | 263.6 | 8.4 | 17.0 | 45.1 | 151.6 | 0.19 | 0.70 | 3.26 | 14.65 |
| NAT-B [25] | CVPR'2023 | 90 | 13.7 | 84.3 | 15.9 | 22.5 | 88.6 | 296.9 | - | - | - | - | 0.42 | 3.03 | 12.33 | 52.47 |
| MogaNet-L [42] | ICLR'2024 | 83 | 15.9 | 84.7 | 81.6 | 90.3 | 98.8 | 357.7 | 14.9 | 28.7 | 77.4 | 263.2 | 0.31 | 1.08 | 51.56 | 26.93 |
| RDNet-L | Ours | 186 | 34.7 | 84.8 | 17.0 | 17.3 | 59.8 | 216.1 | 8.1 | 20.4 | 62.9 | 231.8 | 0.26 | 1.15 | 4.72 | 19.13 |
| MogaNet-XL [42] | ICLR'2024 | 181 | 34.5 | 85.1 | 82.3 | 93.3 | 146.3 | 549.5 | 17.4 | 38.6 | 115.4 | 421.9 | 0.40 | 2.57 | 10.27 | 49.49 |