Appendix: A ConvNet for the 2020s

Zhuang Liu[1,2] Hanzi Mao[1] Chao-Yuan Wu[1] Christoph Feichtenhofer[1] Trevor Darrell[2] Saining Xie[1]
[1]Facebook AI Research (FAIR) [2]UC Berkeley

In this Appendix, we provide further experimental details (§A), robustness evaluation results (§B), more modernization experiment results (§C), and a detailed network specification (§D). We further benchmark model throughput on A100 GPUs (§E). Finally, we discuss the limitations (§F) and societal impact (§G) of our work. Our code is available at https://github.com/facebookresearch/ConvNeXt.

A. Experimental Settings

A.1. ImageNet (Pre-)training

We provide ConvNeXts' ImageNet-1K training and ImageNet-22K pre-training settings in Table 5. The settings are used for our main results in Table 1 (Section 3.1). All ConvNeXt variants use the same setting, except the stochastic depth rate is customized for model variants.

For experiments in "modernizing a ConvNet" (Section 2), we also use Table 5's setting for ImageNet-1K, except EMA is disabled, as we find using EMA severely hurts models with BatchNorm layers.

For isotropic ConvNeXts (Section 3.2), the setting for ImageNet-1K in Table A is also adopted, but warmup is extended to 50 epochs, and layer scale is disabled for isotropic ConvNeXt-S/B. The stochastic depth rates are 0.1/0.2/0.5 for isotropic ConvNeXt-S/B/L.

(pre-)training config	ConvNeXt-T/S/B/L ImageNet-1K 224^2	ConvNeXt-T/S/B/L/XL ImageNet-22K 224^2
weight init	trunc. normal (0.2)	trunc. normal (0.2)
optimizer	AdamW	AdamW
base learning rate	4e-3	4e-3
weight decay	0.05	0.05
optimizer momentum	β_1, β_2=0.9, 0.999	β_1, β_2=0.9, 0.999
batch size	4096	4096
training epochs	300	90
learning rate schedule	cosine decay	cosine decay
warmup epochs	20	5
warmup schedule	linear	linear
layer-wise lr decay [6, 12]	None	None
randaugment [14]	(9, 0.5)	(9, 0.5)
mixup [90]	0.8	0.8
cutmix [89]	1.0	1.0
random erasing [91]	0.25	0.25
label smoothing [69]	0.1	0.1
stochastic depth [37]	0.1/0.4/0.5/0.5	0.0/0.0/0.1/0.1/0.2
layer scale [74]	1e-6	1e-6
head init scale [74]	None	None
gradient clip	None	None
exp. mov. avg. (EMA) [51]	0.9999	None

Table 5. **ImageNet-1K/22K (pre-)training settings**. Multiple stochastic depth rates (e.g., 0.1/0.4/0.5/0.5) are for each model (e.g., ConvNeXt-T/S/B/L) respectively.

A.2. ImageNet Fine-tuning

We list the settings for fine-tuning on ImageNet-1K in Table 6. The fine-tuning starts from the final model weights obtained in pre-training, without using the EMA weights, even if in pre-training EMA is used and EMA accuracy is reported. This is because we do not observe improvement if we fine-tune with the EMA weights (consistent with observations in [73]). The only exception is ConvNeXt-L pre-trained on ImageNet-1K, where the model accuracy is significantly lower than the EMA accuracy due to overfitting, and we select its best EMA model during pre-training as the starting point for fine-tuning.

In fine-tuning, we use layer-wise learning rate decay [6, 12] with every 3 consecutive blocks forming a group. When the model is fine-tuned at 384^2 resolution, we use a crop ratio of 1.0 (i.e., no cropping) during testing following [2, 74, 80], instead of 0.875 at 224^2.

A.3. Downstream Tasks

For ADE20K and COCO experiments, we follow the training settings used in BEiT [6] and Swin [45]. We also use MMDetection [10] and MMSegmentation [13] toolboxes. We use the final model weights (instead of EMA weights) from ImageNet pre-training as network initializations.

We conduct a lightweight sweep for COCO experiments including learning rate {1e-4, 2e-4}, layer-wise learning rate decay [6] {0.7, 0.8, 0.9, 0.95}, and stochastic depth rate {0.3, 0.4, 0.5, 0.6, 0.7, 0.8}. We fine-tune the ImageNet-22K pre-trained Swin-B/L on COCO using the same sweep. We use the official code and pre-trained model weights [3].

The hyperparameters we sweep for ADE20K experiments include learning rate {8e-5, 1e-4}, layer-wise learning rate

pre-training config	ConvNeXt-B/L ImageNet-1K 224^2	ConvNeXt-T/S/B/L/XL ImageNet-22K 224^2
fine-tuning config	ImageNet-1K 384^2	ImageNet-1K 224^2 and 384^2
optimizer	AdamW	AdamW
base learning rate	5e-5	5e-5
weight decay	1e-8	1e-8
optimizer momentum	β_1, β_2=0.9, 0.999	β_1, β_2=0.9, 0.999
batch size	512	512
training epochs	30	30
learning rate schedule	cosine decay	cosine decay
layer-wise lr decay	0.7	0.8
warmup epochs	None	None
warmup schedule	N/A	N/A
randaugment	(9, 0.5)	(9, 0.5)
mixup	None	None
cutmix	None	None
random erasing	0.25	0.25
label smoothing	0.1	0.1
stochastic depth	0.8/0.95	0.0/0.1/0.2/0.3/0.4
layer scale	pre-trained	pre-trained
head init scale	0.001	0.001
gradient clip	None	None
exp. mov. avg. (EMA)	None	None(T-L)/0.9999(XL)

Table 6. **ImageNet-1K fine-tuning settings**. Multiple values (e.g., 0.8/0.95) are for each model (e.g., ConvNeXt-B/L) respectively.

decay {0.8, 0.9}, and stochastic depth rate {0.3, 0.4, 0.5}. We report validation mIoU results using multi-scale testing. Additional single-scale testing results are in Table 7.

backbone	input crop.	mIoU
ImageNet-1K pre-trained		
• ConvNeXt-T	512^2	46.0
• ConvNeXt-S	512^2	48.7
• ConvNeXt-B	512^2	49.1
ImageNet-22K pre-trained		
• ConvNeXt-B‡	640^2	52.6
• ConvNeXt-L‡	640^2	53.2
• ConvNeXt-XL‡	640^2	53.6

Table 7. **ADE20K validation results** with single-scale testing.

B. Robustness Evaluation

Additional robustness evaluation results for ConvNeXt models are presented in Table 8. We directly test our ImageNet-1K trained/fine-tuned classification models on several robustness benchmark datasets such as ImageNet-A [33], ImageNet-R [30], ImageNet-Sketch [78] and ImageNet-C/$\bar{\text{C}}$ [31, 48] datasets. We report mean corruption error (mCE) for ImageNet-C, corruption error for ImageNet-$\bar{\text{C}}$, and top-1 Accuracy for all other datasets.

ConvNeXt (in particular the large-scale model variants) exhibits promising robustness behaviors, outperforming state-of-the-art robust transformer models [47] on several benchmarks. With extra ImageNet-22K data, ConvNeXt-XL demonstrates strong domain generalization capabilities (*e.g.* achieving 69.3%/68.2%/55.0% accuracy on ImageNet-A/R/Sketch benchmarks, respectively). We note that these robustness evaluation results were acquired without using any specialized modules or additional fine-tuning procedures.

Model	Data/Size	FLOPs / Params	Clean	C (↓)	$\bar{\text{C}}$ (↓)	A	R	SK
ResNet-50	$1K/224^2$	4.1 / 25.6	76.1	76.7	57.7	0.0	36.1	24.1
Swin-T [45]	$1K/224^2$	4.5 / 28.3	81.2	62.0	-	21.6	41.3	29.1
RVT-S* [47]	$1K/224^2$	4.7 / 23.3	81.9	49.4	37.5	25.7	47.7	34.7
ConvNeXt-T	$1K/224^2$	4.5 / 28.6	82.1	53.2	40.0	24.2	47.2	33.8
Swin-B [45]	$1K/224^2$	15.4 / 87.8	83.4	54.4	-	35.8	46.6	32.4
RVT-B* [47]	$1K/224^2$	17.7 / 91.8	82.6	46.8	**30.8**	28.5	48.7	36.0
ConvNeXt-B	$1K/224^2$	15.4 / 88.6	**83.8**	**46.8**	34.4	**36.7**	**51.3**	**38.2**
ConvNeXt-B	$22K/384^2$	45.1 / 88.6	86.8	43.1	30.7	62.3	64.9	51.6
ConvNeXt-L	$22K/384^2$	101.0 / 197.8	87.5	40.2	29.9	65.5	66.7	52.8
ConvNeXt-XL	$22K/384^2$	179.0 / 350.2	**87.8**	**38.8**	**27.1**	**69.3**	**68.2**	**55.0**

Table 8. **Robustness evaluation of ConvNeXt**. We do not make use of any specialized modules or additional fine-tuning procedures.

C. Modernizing ResNets: detailed results

Here we provide detailed tabulated results for the *modernization* experiments, at both ResNet-50 / Swin-T and ResNet-200 / Swin-B regimes. The ImageNet-1K top-1 accuracies and FLOPs for each step are shown in Table 10 and 11. ResNet-50 regime experiments are run with 3 random seeds.

For ResNet-200, the initial number of blocks at each stage is (3, 24, 36, 3). We change it to Swin-B's (3, 3, 27, 3) at the step of changing stage ratio. This drastically reduces the FLOPs, so at the same time, we also increase the width from 64 to 84 to keep the FLOPs at a similar level. After the step of adopting depthwise convolutions, we further increase the width to 128 (same as Swin-B's) as a separate step.

The observations on the ResNet-200 regime are mostly consistent with those on ResNet-50 as described in the main paper. One interesting difference is that inverting dimensions brings a larger improvement at ResNet-200 regime than at ResNet-50 regime (+0.79% *vs*. +0.14%). The performance gained by increasing kernel size also seems to saturate at kernel size 5 instead of 7. Using fewer normalization layers also has a bigger gain compared with the ResNet-50 regime (+0.46% *vs*. +0.14%).

D. Detailed Architectures

We present a detailed architecture comparison between ResNet-50, ConvNeXt-T and Swin-T in Table 9. For differently sized ConvNeXts, only the number of blocks and the number of channels at each stage differ from ConvNeXt-T (see Section 3 for details). ConvNeXts enjoy the simplicity of standard ConvNets, but compete favorably with Swin Transformers in visual recognition.

	output size	• ResNet-50	• ConvNeXt-T	○ Swin-T
stem	56×56	7×7, 64, stride 2 3×3 max pool, stride 2	4×4, 96, stride 4	4×4, 96, stride 4
res2	56×56	[1×1, 64 3×3, 64 1×1, 256] × 3	[d7×7, 96 1×1, 384 1×1, 96] × 3	[1×1, 96×3 MSA, w7×7, H=3, rel. pos. 1×1, 96 1×1, 384 1×1, 96] × 2
res3	28×28	[1×1, 128 3×3, 128 1×1, 512] × 4	[d7×7, 192 1×1, 768 1×1, 192] × 3	[1×1, 192×3 MSA, w7×7, H=6, rel. pos. 1×1, 192 1×1, 768 1×1, 192] × 2
res4	14×14	[1×1, 256 3×3, 256 1×1, 1024] × 6	[d7×7, 384 1×1, 1536 1×1, 384] × 9	[1×1, 384×3 MSA, w7×7, H=12, rel. pos. 1×1, 384 1×1, 1536 1×1, 384] × 6
res5	7×7	[1×1, 512 3×3, 512 1×1, 2048] × 3	[d7×7, 768 1×1, 3072 1×1, 768] × 3	[1×1, 768×3 MSA, w7×7, H=24, rel. pos. 1×1, 768 1×1, 3072 1×1, 768] × 2
FLOPs		4.1×10^9	4.5×10^9	4.5×10^9
# params.		25.6×10^6	28.6×10^6	28.3×10^6

Table 9. **Detailed architecture specifications** for ResNet-50, ConvNeXt-T and Swin-T.

model	IN-1K acc.	GFLOPs
ResNet-50 (PyTorch [1])	76.13	4.09
ResNet-50 (enhanced recipe)	78.82 ± 0.07	4.09
stage ratio	79.36 ± 0.07	4.53
"patchify" stem	79.51 ± 0.18	4.42
depthwise conv	78.28 ± 0.08	2.35
increase width	80.50 ± 0.02	5.27
inverting dimensions	80.64 ± 0.03	4.64
move up depthwise conv	79.92 ± 0.08	4.07
kernel size → 5	80.35 ± 0.08	4.10
kernel size → 7	80.57 ± 0.14	4.15
kernel size → 9	80.57 ± 0.06	4.21
kernel size → 11	80.47 ± 0.11	4.29
ReLU → GELU	80.62 ± 0.14	4.15
fewer activations	81.27 ± 0.06	4.15
fewer norms	81.41 ± 0.09	4.15
BN → LN	81.47 ± 0.09	4.46
separate d.s. conv (ConvNeXt-T)	81.97 ± 0.06	4.49
Swin-T [45]	81.30	4.50

Table 10. **Detailed results for modernizing a ResNet-50.** Mean and standard deviation are obtained by training the network with three different random seeds.

model	IN-1K acc.	GFLOPs
ResNet-200 [29]	78.20	15.01
ResNet-200 (enhanced recipe)	81.14	15.01
stage ratio and increase width	81.33	14.52
"patchify" stem	81.59	14.38
depthwise conv	80.54	7.23
increase width	81.85	16.76
inverting dimensions	82.64	15.68
move up depthwise conv	82.04	14.63
kernel size → 5	82.32	14.70
kernel size → 7	82.30	14.81
kernel size → 9	82.27	14.95
kernel size → 11	82.18	15.13
ReLU → GELU	82.19	14.81
fewer activations	82.71	14.81
fewer norms	83.17	14.81
BN → LN	83.35	14.81
separate d.s. conv (ConvNeXt-B)	83.60	15.35
Swin-B [45]	83.50	15.43

Table 11. **Detailed results for modernizing a ResNet-200.**

E. Benchmarking on A100 GPUs

Following Swin Transformer [45], the ImageNet models' inference throughputs in Table 1 are benchmarked using a V100 GPU, where ConvNeXt is slightly faster in inference than Swin Transformer with a similar number of parameters. We now benchmark them on the more advanced A100 GPUs, which support the TensorFloat32 (TF32) tensor cores. We employ PyTorch [50] version 1.10 to use the latest "Channel Last" memory layout [22] for further speedup.

We present the results in Table 12. Swin Transformers and ConvNeXts both achieve faster inference throughput than V100 GPUs, but ConvNeXts' advantage is now significantly greater, sometimes *up to 49% faster*. This preliminary study shows promising signals that ConvNeXt, employed with standard ConvNet modules and simple in design, could be practically more efficient models on modern hardwares.

model	image size	FLOPs	throughput (image / s)	IN-1K / 22K trained, 1K acc.
○ Swin-T	224^2	4.5G	1325.6	81.3 / –
● ConvNeXt-T	224^2	4.5G	**1943.5** (+47%)	**82.1 / 82.9**
○ Swin-S	224^2	8.7G	857.3	83.0 / –
● ConvNeXt-S	224^2	8.7G	**1275.3** (+49%)	**83.1 / 84.6**
○ Swin-B	224^2	15.4G	662.8	83.5 / 85.2
● ConvNeXt-B	224^2	15.4G	**969.0** (+46%)	**83.8 / 85.8**
○ Swin-B	384^2	47.1G	242.5	84.5 / 86.4
● ConvNeXt-B	384^2	45.0G	**336.6** (+39%)	**85.1 / 86.8**
○ Swin-L	224^2	34.5G	435.9	– / 86.3
● ConvNeXt-L	224^2	34.4G	**611.5** (+40%)	**84.3 / 86.6**
○ Swin-L	384^2	103.9G	157.9	– / 87.3
● ConvNeXt-L	384^2	101.0G	**211.4** (+34%)	85.5 / **87.5**
● ConvNeXt-XL	224^2	60.9G	**424.4**	– / **87.0**
● ConvNeXt-XL	384^2	179.0G	**147.4**	– / **87.8**

Table 12. **Inference throughput comparisons on an A100 GPU.** Using TF32 data format and "channel last" memory layout, ConvNeXt enjoys up to ∼49% higher throughput compared with a Swin Transformer with similar FLOPs.

F. Limitations

We demonstrate ConvNeXt, a pure ConvNet model, can perform as good as a hierarchical vision Transformer on image classification, object detection, instance and semantic segmentation tasks. While our goal is to offer a broad range of evaluation tasks, we recognize computer vision applications are even more diverse. ConvNeXt may be more suited for certain tasks, while Transformers may be more flexible for others. A case in point is multi-modal learning, in which a cross-attention module may be preferable for modeling feature interactions across many modalities. Additionally, Transformers may be more flexible when used for tasks requiring discretized, sparse, or structured outputs. We believe the architecture choice should meet the needs of the task at hand while striving for simplicity.

G. Societal Impact

In the 2020s, research on visual representation learning began to place enormous demands on computing resources. While larger models and datasets improve performance across the board, they also introduce a slew of challenges. ViT, Swin, and ConvNeXt all perform best with their huge model variants. Investigating those model designs inevitably results in an increase in carbon emissions. One important direction, and a motivation for our paper, is to strive for simplicity — with more sophisticated modules, the network's design space expands enormously, obscuring critical components that contribute to the performance difference. Additionally, large models and datasets present issues in terms of model robustness and fairness. Further investigation on the robustness behavior of ConvNeXt vs. Transformer will be an interesting research direction. In terms of data, our findings indicate that ConvNeXt models benefit from pre-training on large-scale datasets. While our method makes use of the publicly available ImageNet-22K dataset, individuals may wish to acquire their own data for pre-training. A more circumspect and responsible approach to data selection is required to avoid potential concerns with data biases.