SUPPLEMENTARY MATERIAL OF "CONTEXTUAL CONVOLUTIONAL NETWORKS"

Shuxian Liang[1,2]*, **Xu Shen**[2], **Tongliang Liu**[3], **Xian-Sheng Hua**[1]†
[1]Zhejiang University, [2]Alibaba Cloud Computing Ltd.,
[3]Sydney AI Centre, The University of Sydney
shuxian.lsx@zju.edu.cn, shenxuustc@gmail.com,
tongliang.liu@sydney.edu.au, huaxiansheng@gmail.com

In this supplementary material, we provide detailed architecture specifications (§1), further experimental details (§2), acceleration approaches (§3), linear probe results (§4), more Grad-CAM visualization results (§5) and failure analysis (§6) for contextual CNNs.

1 DETAILED ARCHITECTURES

Table 1 presents a detailed architecture comparison between Contextual ConvNeXt-T, Contextual ResNet50 and their plain counterparts. For Contextual ConvNeXt-T/S/B, only one of every 3 blocks for each contextual stage ("*Stage2/3/4*") is substituted by the contextual convolution block. For Contextual ResNet50, one of every 2 blocks for each contextual stage is replaced by the contextual convolution block. The resulting contextual models share the same number of blocks and same number of channels as their plain counterparts at all stages.

2 DETAILED EXPERIMENTAL SETTINGS

2.1 CHOOSING THE NUMBERS OF CONSIDERED CLASSES (N_1, N_2, N_3 AND N_4)

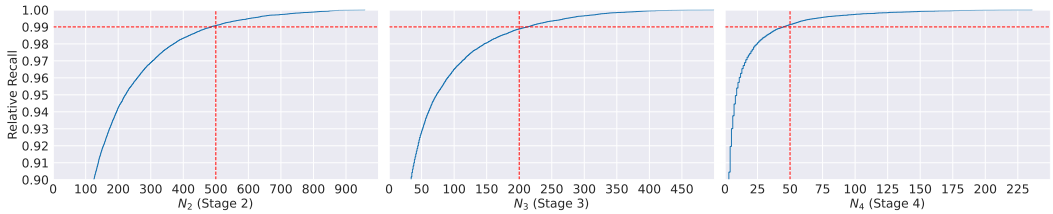

Figure 1: The relative recall curves of ground-truth classes for "*Stage2*", "*Stage3*" and "*Stage4*".

For "*Stage1*", "*Stage2*", "*Stage3*" and "*Stage4*" of Contextual CNN, the numbers of considered classes are denoted by N_1, N_2, N_3 and N_4, respectively. N_1 is always the number of classes of the dataset (e.g. $N_1 = 1000$ for ImageNet-1K), N_2, N_3 and N_4 are hyperparameters for the Contextual CNN. In what follows, taking Contextual ConvNeXt-T on ImageNet-1K for example, we introduce the strategy of choosing N_2, N_3 and N_4:

1. Following the training recipe of ConvNeXt-T, we train a naive Contextual ConvNeXt-T for a few epochs (e.g., 50) with $N_2 = 1000$, $N_3 = 1000$ and $N_4 = 1000$.
2. On the validation set, we plot the relative recall curve of ground-truth classes for stage i ($i = 2, 3, 4$) using the naive model. The recall is calculated using the classification scores s_i for the current stage (detailed in Section 3.2 of the main paper). The relative recall curve shows how the recall changes when the number of considered classes N_i varies.
3. The resulting curves are shown in Figure 1. One can observe that the naive model achieves a high recall of 99% when N_2 is around 500, N_3 is around 200, and N_4 is around 50. Based

*This work was done when the author was visiting Alibaba as a research intern.
†Corresponding author.

Table 1: Detailed architecture specifications for Contextual ConvNeXt-T, Contextual ResNet50 and their plain counterparts. The input image size is assumed as 224×224. "d7×7" denotes a 7×7 depthwise convolution. "c3×3" denotes a 3×3 contextual convolution. "cd7×7" denotes a 7×7 contextual depthwise convolution. The blocks in brown are contextual convolution blocks.

	output size	ConvNeXt-T	Contextual ConvNeXt-T	ResNet50	Contextual ResNet50
stem	56×56	4×4, 96, stride 4		7×7, 64, stride 2 3×3 max pool, stride 2	
stage1	56×56	$\begin{bmatrix} \text{d7×7, 96} \\ \text{1×1, 384} \\ \text{1×1, 96} \end{bmatrix} \times 3$		$\begin{bmatrix} \text{1×1, 64} \\ \text{3×3, 64} \\ \text{1×1, 256} \end{bmatrix} \times 3$	
stage2	28×28	$\begin{bmatrix} \text{d7×7, 192} \\ \text{1×1, 768} \\ \text{1×1, 192} \end{bmatrix} \times 3$	contextualizing ×1 $\left\{ \begin{bmatrix} \underline{\text{cd7×7, 192}} \\ \underline{\text{1×1, 768}} \\ \underline{\text{1×1, 192}} \end{bmatrix} \begin{bmatrix} \text{d7×7, 192} \\ \text{1×1, 768} \\ \text{1×1, 192} \end{bmatrix} \begin{bmatrix} \text{d7×7, 192} \\ \text{1×1, 768} \\ \text{1×1, 192} \end{bmatrix} \right\} \times 1$	$\begin{bmatrix} \text{1×1, 128} \\ \text{3×3, 128} \\ \text{1×1, 512} \end{bmatrix} \times 4$	contextualizing ×1 $\left\{ \begin{bmatrix} \underline{\text{1×1, 128}} \\ \underline{\text{c3×3, 128}} \\ \underline{\text{1×1, 512}} \end{bmatrix} \begin{bmatrix} \text{1×1, 128} \\ \text{3×3, 128} \\ \text{1×1, 512} \end{bmatrix} \right\} \times 2$
stage3	14×14	$\begin{bmatrix} \text{d7×7, 384} \\ \text{1×1, 1536} \\ \text{1×1, 384} \end{bmatrix} \times 9$	contextualizing ×1 $\left\{ \begin{bmatrix} \underline{\text{cd7×7, 384}} \\ \underline{\text{1×1, 1536}} \\ \underline{\text{1×1, 384}} \end{bmatrix} \begin{bmatrix} \text{d7×7, 384} \\ \text{1×1, 1536} \\ \text{1×1, 384} \end{bmatrix} \begin{bmatrix} \text{d7×7, 384} \\ \text{1×1, 1536} \\ \text{1×1, 384} \end{bmatrix} \right\} \times 3$	$\begin{bmatrix} \text{1×1, 256} \\ \text{3×3, 256} \\ \text{1×1, 1024} \end{bmatrix} \times 6$	contextualizing ×1 $\left\{ \begin{bmatrix} \underline{\text{1×1, 256}} \\ \underline{\text{c3×3, 256}} \\ \underline{\text{1×1, 1024}} \end{bmatrix} \begin{bmatrix} \text{1×1, 256} \\ \text{3×3, 256} \\ \text{1×1, 1024} \end{bmatrix} \right\} \times 3$
stage4	7×7	$\begin{bmatrix} \text{d7×7, 768} \\ \text{1×1, 3072} \\ \text{1×1, 768} \end{bmatrix} \times 3$	contextualizing ×1 $\left\{ \begin{bmatrix} \underline{\text{cd7×7, 768}} \\ \underline{\text{1×1, 3072}} \\ \underline{\text{1×1, 768}} \end{bmatrix} \begin{bmatrix} \text{d7×7, 768} \\ \text{1×1, 3072} \\ \text{1×1, 768} \end{bmatrix} \begin{bmatrix} \text{d7×7, 768} \\ \text{1×1, 3072} \\ \text{1×1, 768} \end{bmatrix} \right\} \times 1$	$\begin{bmatrix} \text{1×1, 512} \\ \text{3×3, 512} \\ \text{1×1, 2048} \end{bmatrix} \times 3$	contextualizing ×1 $\left\{ \begin{bmatrix} \underline{\text{1×1, 512}} \\ \underline{\text{c3×3, 512}} \\ \underline{\text{1×1, 2048}} \end{bmatrix} \begin{bmatrix} \text{1×1, 512} \\ \text{3×3, 512} \\ \text{1×1, 2048} \end{bmatrix} \begin{bmatrix} \underline{\text{1×1, 512}} \\ \underline{\text{c3×3, 512}} \\ \underline{\text{1×1, 2048}} \end{bmatrix} \right\} \times 1$

on this observation, we set $N_2 = 500$, $N_3 = 200$ and $N_4 = 50$ for Contextual ConvNeXt-T and train a new model from scratch with this setting for our experiments.

The above strategy can be easily extended to a new dataset or a new contextual model. For simplicity, we adopt the setting of Contextual ConvNeXt-T for all experiments on ImageNet-1K in this work, including the ones that use Contextual ConvNeXt-S/B and Contextual ResNet50. For the downstream experiments, we scale N_2/N_3/N_4 linearly according to the given N_1 (the number of classes for the datasets). For more details, please refer to §2.4 and §2.5.

2.2 Choosing the value of loss weight scalar (α)

The cross entropy loss $\mathcal{L}_i$ can be large in magnitude with a large number of considered classes N_i (refer to equation 1 of the main paper). As a result, losses of the earlier stages dominates the overall loss of Contextual CNN. To balance the losses of intermediate stages and the final classification layer,

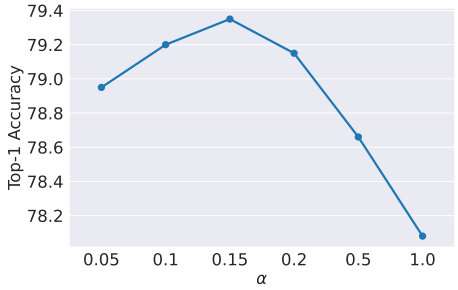

Figure 2: Top-1 accuracy as a function of the loss weight scalar α, measured on ImageNet with a Contextual ResNet50 model.

we adopt loss weight $\alpha < 1$ to downweighting the losses of earlier stages $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ (refer to equation 2 of the main paper). As shown in Figure 2, we have experimented with different values of α and found that $\alpha = 0.15$ leads to the best performance for Contextual ResNet50. We reuse this setting for all other experiments of this work.

2.3 Settings for Image Classification on ImageNet

For Contextual ConvNeXt-T/S/B, to compare with the state-of-the-arts methods, we follow the same training recipe as Liu et al. (2022). For clarity, we list the training settings in Table 2 (column 2). The settings are used for our results in Table 3 of the main paper. All Contextual ConvNeXt variants use the same setting, except that the stochastic depth rate is customized for each model.

For Contextual ResNet50, to accelerate the ablation study, we use simple augmentation and regularization strategies as shown in Table 2 (column 3). The settings are used for all our ablative experiments (Table 4-6 of the main paper). The accuracy of our baseline ResNet50 is slightly better than the official result of PyTorch (Paszke et al., 2019) (76.58% vs. 76.13%).

Table 2: ImageNet training settings. The recipe of training Contextual ConvNeXt-T/S/B from scratch is from Liu et al. (2022). The recipe of training Contextual ResNet50 from scratch is partially from Wightman et al. (2021). Multiple stochastic depth rates (i.e., 0.1/0.4/0.5) are for multiple variants (i.e., Contextual ConvNeXt-T/S/B), respectively.

settings	Contextual ConvNeXt-T/S/B	Contextual ResNet50
optimizer	AdamW	AdamW
base learning rate	4e-3	1e-3
weight decay	2e-5	1e-2
batch size	4096	1024
training epochs	300	150
learning rate schedule	cosine decay	cosine decay
warmup epochs	20	20
warmup schedule	linear	linear
randaugment (Cubuk et al., 2020)	(9, 0.5)	(9, 0.5)
mixup (Zhang et al., 2018)	0.8	None
cutmix (Yun et al., 2019)	1.0	None
random erasing (Zhong et al., 2020)	0.25	None
label smoothing (Szegedy et al., 2016)	0.1	None
stochastic depth (Huang et al., 2016)	0.1/0.4/0.5	None
layer scale (Touvron et al., 2021)	1e-6	None
EMA (Polyak & Juditsky, 1992)	0.9999	None

2.4 SETTINGS FOR VIDEO CLASSIFICATION ON KINETICS-400

For the experiments on Kinetics-400, we follow the training/testing settings used in TSM Lin et al. (2019). Specifically, the Contextual ResNet50 backbone is firstly pretrained on ImageNet using the recipe in Table 2 (column 3). Then, we fine-tune the pretrained backbone using the TSM framework. During fine-tuning, since Kinetics-400 has 400 video classes, we reinitialize the class embeddings for the 400 classes (with $d = 256$) and use the rest weights pretrained on ImageNet as network initializations. According to the strategy in §2.1, the numbers of considered classes for the four stages on Kinetics-400 are set as: $N_1 = 400$, $N_2 = 200$, $N_3 = 80$ and $N_4 = 20$.

2.5 SETTINGS FOR INSTANCE SEGMENTATION ON COCO

For the experiment on COCO, we use MMDetection (Chen et al., 2019) as codebase and follow the training/testing settings used in Swin Transformers (Liu et al., 2021) and ConvNeXts (Liu et al., 2022). We use multi-scale training, AdamW optimizer, and a $3\times$ schedule. Notably, since COCO is not annotated with image classes, we cancel all classification losses for the contextual backbone ($\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$ and $\mathcal{L}_4$), and reuse the class embeddings from ImageNet-1K (rather than reinitialize them as in Kinetics-400). During fine-tuning, the class embeddings are frozen and the rest of the contextual model is fine-tuned using the instance segmentation losses.

3 EFFICIENT BATCH COMPUTATION FOR CONTEXTUAL CONVOLUTIONS

One issue with the batch computation of contextual convolutions is that the *weight offsets* are adaptive for each image and therefore the contextual convolution weights have an extra dimension of batch size. For a contextual convolution layer, consider the input $\mathbf{X}$ (B, C, h, w) and weight $\mathbf{W}$ (B, C', C, k, k), where B is batch size, C/C' is input/output channel size, $h \times w$ is spatial size and $k \times k$ is kernel size. A naive solution is to perform convolution image by image, which is however slow when B grows large. To enjoy the advantage of parallel processing on GPUs, we reshape $\mathbf{X}$ to $(1, B \times C, h, w)$, reshape $\mathbf{W}$ to $(B \times C, C', k, k)$, then perform a group convolution with a group number of B following Ma et al. (2020).

Another issue with the batch computation of contextual convolutions is that the *sampling offsets* are adaptive for each image as well as each output position. To accelerate computation, we follow the two-step computation pipeline of deformable convolutions (Dai et al., 2017; Zhu et al., 2019). Specifically, we first sample the deformable input features ($C \times k \times k$) for each output position of each image in parallel, and second aggregate the features and perform batch matrix multiplication per convolution group.

To handle both *weight offsets* and *sampling offsets*, we combine the above two practices and found they work efficiently for contextual ResNets. However, for contextual ConvNeXts, depthwise convolutions[1] are adopted and the actual number of groups for contextual depthwise convolutions is as large as $B \times C$. Since the aforementioned computation pipeline performs batch matrix multiplication per group, the contextual depthwise convolutions get very slow in practice. To solve the problem, we propose an equivalent one-step computation pipeline for the contextual depthwise convolutions. Specifically, for each output position of each image, we accumulate the weighted values of $k \times k$ kernel positions for each channel ($C \times 1$). The accumulation step can be performed in parallel and still enjoy the acceleration of GPUs. Compared to the previous pipeline, the newly proposed pipeline generate the same output features while achieves $2\times$ faster inference and requires only $\frac{1}{k^2}$ of memory usage ($C \times k \times k \rightarrow C \times 1$). As a result, the Contextual ConvNexts shares highly competitive inference throughput as their plain counterparts (refer to Table 1 of the main paper).

4 LINEAR PROBE EVALUATION

This section compares the linear probe performance of Contextual ResNet50 and ResNet50 (He et al., 2016) on two widely-used downstream datasets, Stanford Cars (Krause et al., 2013) and CUB (Wah et al., 2011). The first dataset, Stanford Cars, is a fine-grained classification dataset with $8,144$ training images and $8,041$ testing images from 196 car classes. The second dataset, CUB, is another

[1] A depthwise convolution is typically taken as a group convolution with a group number of C.

fine-grained classification dataset with $5,994$ training images and $5,794$ testing images from 200 bird classes.

Settings. The pretrained weights for Contextual ResNet50 and ResNet50 are from the ablation models $a5$ and $a1$ in Table 4 of the main paper, respectively. For Contextual ResNet50, we freeze all pretrained layers (including both contextualizing layers and convolutional blocks), remove the original final classifying head and fine-tune a randomly-initialized fully-connected (FC) head over the backbone features. For both datasets, we fine-tune the FC head for $3,000$ iterations using an AdamW optimizer (Loshchilov & Hutter, 2017) with a learning rate of 0.001. The batch size is set as 512 and the weight decay is $1e-5$. The top-1 accuracy on a single crop of size 448×448 is reported.

Table 3: Linear probe results on Stanford Cars (Krause et al., 2013) and CUB (Wah et al., 2011).

backbone	Cars top-1 acc.	CUB top-1 acc.
ResNet50	50.20%	63.02%
Contextual ResNet50 (ours)	**55.82%**	**66.64%**

Results. On both Stanford Cars and CUB datasets, the proposed Contextual ResNet50 yields significantly better top-1 accuracy ($+5.63\%$/$+3.62\%$). This demonstrates that the learned features of the proposed method are robust and can be simply transferred to fine-grained downstream tasks.

5 More Grad-CAM Visualizations

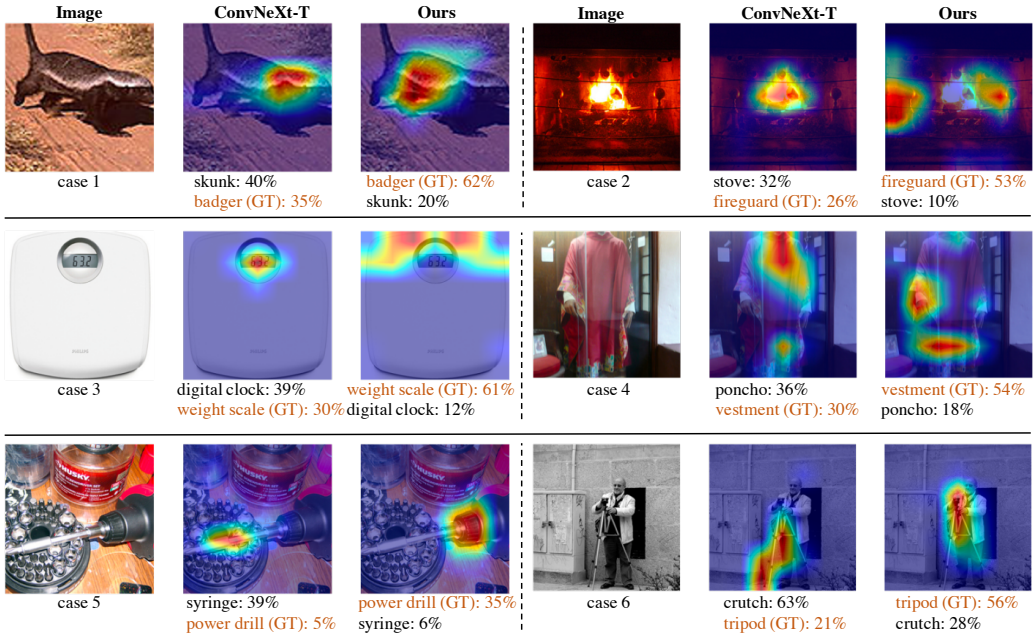

Figure 3: More Grad-CAM visualization results of ConvNeXt-T and Contextual ConvNeXt-T on ImageNet-1K. "GT" denotes the ground truth class of the image.

Figure 3 presents more Grad-CAM visualizations (Selvaraju et al., 2017) comparing ConvNeXt-T and the proposed Contextual ConvNeXt-T. For all cases, our method differentiates the ground-truth classes from their distractors by extracting features of discriminative patterns among them, instead of features of shared patterns. Specifically:

- For case 1, our method differentiates "badger" from "skunk" by extracting features of the body of the animal, instead of the head.

- For case 2, our method differentiates "fireguard" from "stove" by extracting features of the surroundings of the fire, instead of the fire itself.
- For case 3, our method differentiates "weight scale" from "digital clock" by extracting features of the shape of the object, instead of the digital screen.
- For case 4, our method differentiates "vestment" from "poncho" by extracting features of the sleeves and the decorations of the clothes, instead of the collar.
- For case 5, our method differentiates "power drill" from "syringe" by extracting features of the chuck, instead of the bit.
- For case 6, our method differentiates "tripod" from "crutch" by extracting features of the camera platform, instead of the leg.

These observations demonstrate that the newly proposed Contextual CNN can modulate themselves according to the contextual priors (a few most likely classes) for each image, and learn to generate more discriminative features with regard to these classes.

6 Failure Analysis

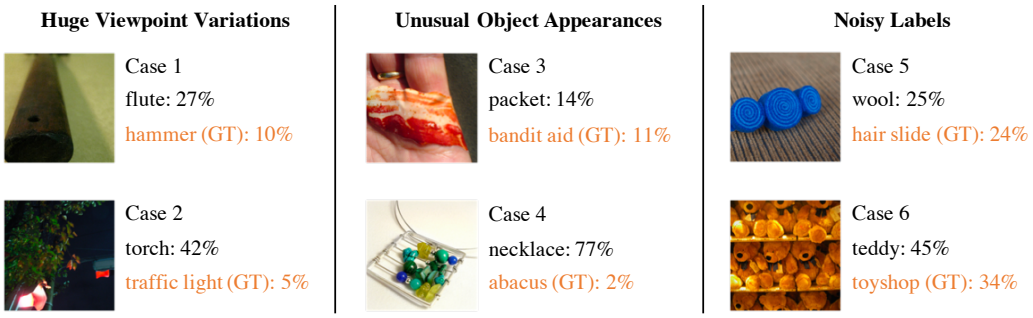

Figure 4: Visualization of failure cases of Contextual ConvNeXt-T on ImageNet-1K. "GT" denotes the ground truth class of the image.

Figure 4 presents some failure cases of the proposed Contextual ConvNeXt-T on ImageNet-1K. The failure cases are summarized into three kinds:

- Huge viewpoint variations. Both case 1 and case 2 show part of the objects ("hammer" and "traffic light") from the bottom of the objects. The viewpoints are very different from those of training, e.g., the frontal of the objects.
- Unusual object appearances. Case 3 shows an unusual bacon-like "bandit aid". Case 4 shows an unusual "abacus" that is made of decorative beads. These unusual object appearances confuse the proposed model.
- Noisy labels. Case 5 shows a "hair slide" that is simultaneously a "wool". Case 6 shows a scene of the "toyshop" which contains many "teddy" bears. These cases have noisy labels since each of them contain multiple correct classes but is annotated with only one label.

Similar failure cases are also observed in recent SOTA methods like Swin Transformers and ConvNeXt. These cases reveal the unsolved challenges of visual recognition and suggest the potential directions for future works.

References

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, 2020.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCVW*, pp. 554–561, 2013.

Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Ningning Ma, Xiangyu Zhang, Jiawei Huang, and Jian Sun. Weightnet: Revisiting the design space of weight networks. In *ECCV*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 1992.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *ICCV*, 2021.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

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

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020.

Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019.