FEATURE-LEVEL AUGMENTATION TO IMPROVE ROBUSTNESS OF DEEP NEURAL NETWORKS TO AFFINE TRANSFORMATIONS

Adrian Sandru   *Mariana-Iuliana Georgescu*   *Radu Tudor Ionescu*

University of Bucharest, Romania
SecurifAI, Romania

ABSTRACT

Recent studies revealed that convolutional neural networks do not generalize well to small image transformations, e.g. rotations by a few degrees or translations of a few pixels. To improve the robustness to such transformations, we propose to introduce data augmentation at intermediate layers of the neural architecture, in addition to the common data augmentation applied on the input images. By introducing small perturbations to activation maps (features) at various levels, we develop the capacity of the neural network to cope with such transformations. We conduct experiments on three image classification benchmarks (Tiny ImageNet, Caltech-256 and Food-101), considering two different convolutional architectures (ResNet-18 and DenseNet-121). When compared with two state-of-the-art stabilization methods, the empirical results show that our approach consistently attains the best trade-off between accuracy and mean flip rate.

Index Terms— Deep learning, data augmentation, convolutional neural networks, robustness to affine transformations.

1. INTRODUCTION

A series of recent studies [1, 2, 3, 4, 5, 6, 7, 8] showed that convolutional neural networks (CNNs) are not properly equipped to deal with small image perturbations. Indeed, it appears that a subtle affine transformation, e.g. a rotation by a few degrees or a translation of a few pixels, can alter the model's decision towards making a wrong prediction. The problem is illustrated by the example shown in Figure 1, where a deep neural model is no longer able to predict the correct class upon downscaling the input image by a factor of 0.9. To increase the robustness to such small perturbations, researchers [1, 2, 3, 4] proposed various approaches ranging from architectural changes [2, 3] and training strategy updates [4] to input data augmentations [9, 10]. However, to the best of our knowledge, none of the previous works tried to apply augmentations at the intermediate layers of the neural network. We conjecture that introducing feature-level augmentations improves the robustness of deep CNNs to affine transformations. To this end, we present an augmentation technique that randomly selects a convolutional layer at each

Fig. 1: An image of a humming bird from Caltech-256 that is wrongly classified as grasshopper (with a probability of 0.22) by a ResNet-18 model, after downscaling it by a factor of 0.9. Best viewed in color.

mini-batch and applies independent affine transformations (translation, rotation, scaling) on each activation map from the selected layer.

To demonstrate the practical utility of our approach, we conduct experiments with ResNet-18 [11] and DenseNet-121 [12] on three benchmark data sets, namely Tiny ImageNet [13], Caltech-256 [14] and Food-101 [15]. Importantly, we show that feature-level augmentation helps even when the models are trained with standard data augmentation. When compared with two state-of-the-art methods [2, 3], the empirical results show that our approach attains the best trade-off between accuracy and mean flip rate. To the best of our knowledge, this is the first time such a comparison is made.

Contribution. In summary, our contribution is twofold:

- We introduce a novel method based on feature-level augmentation to increase the robustness of deep neural networks to affine transformations.

- We conduct an empirical evaluation study to compare state-of-the-art methods addressing the robustness problem among themselves as well as with our approach.

2. RELATED WORK

In literature, there are several approaches towards improving the robustness of deep neural networks to image perturbations. A popular and natural strategy, that proved to work sufficiently well, is to train the network using augmented images,

as suggested in [1, 9, 10]. The intuition behind this approach is to train the model on a wider domain, which can become more similar to the test data. The experiments conducted by the authors suggest that the robustness and performance improve on all scenarios. Due to its popularity, we apply image augmentation to the baseline models employed in our experiments. An extended solution of using augmented images is mentioned in [4], where the authors employed an additional loss function in order to stabilize the output features of the network in such a manner that a strongly perturbed image should have a similar outcome to the original one. Data augmentation methods are considered to be constrained by the photographers' bias [1]. Thus, the model may only learn to generalize to images from a certain (biased) distribution.

One of the recent architectural changes leading to improvements in the stability of CNNs was introduced by Zhang [3]. The observations of the author centered on the fact that modern CNNs are not shift invariant. Thus, small linear changes applied on the input image may have a negative impact on the final result. The source of this issue is considered to be represented by the downsampling process that usually occurs inside neural networks through pooling operations, which breaks the shift-equivariance. A straightforward solution to solve this issue is to avoid subsampling [1], but this comes with a great computational burden. Therefore, Zhang [3] provides adjustments to conventional operations by including a blur kernel in order to reduce the judder caused by downsampling. We conduct experiments showing that this method can deteriorate the quality of the features and negatively impact the final accuracy of the model, despite improving its stability.

Another new architectural design was proposed in [2], where the authors addressed the downsampling issue by proposing a pooling operation that considers all possible grids and selects the component with the highest norm. Their approach was benchmarked against circular shifts of the input, which do not naturally occur in practical scenarios. Hence, we extend their evaluation to generic (non-circular) affine transformations and compare their approach to our own procedure, showcasing the superiority of our method.

Although researchers explored multiple methods to improve the stability of neural models to image perturbations, it seems that there is no technique that can guarantee a never-failing solution. Hence, we consider that addressing the stability problem with better solutions is of great interest to the computer vision community. To the best of our knowledge, we are the first to propose feature-level augmentation as an enhancement to the stability of neural networks.

3. METHOD

The proposed method consists of extending the conventional input augmentation procedure by applying it at the feature level. In all our experiments, we include the conventional input augmentation, which is based on randomly shifting the images on both axes with values between -15 and $+15$ pixels, rotating them by -15 to $+15$ degrees, or rescaling them with a factor between 0.4 and 1.15, following [16]. On top of this, we introduce feature-level augmentation (FLA).

We underline that CNNs are usually composed of multiple blocks intercalated with downscaling (pooling) operations. Starting from this observation, we capture the features between two randomly selected consecutive blocks, augment the activation maps provided by the first selected block, and train the network from the second block to the output using an augmented version of the activation maps.

The augmentation that we propose to employ on the features is based on shifting, rotating and rescaling the activation maps. Due to the fact that one element of a feature map, somewhere deep in the network, is actually the result of processing multiple pixels from the corresponding location in the input, our augmentation procedure should take into account the depth where it is applied, so that the receptive field is not severely affected. Thus, the maximum translation value is scaled accordingly, starting from 15 pixels near the input, gradually going down to 1 pixel for the high-level layers. For rotations, we consider a random value between -15 and $+15$ degrees, while for the scaling operation, we employ a random resize factor between 0.85 and 1.15. In order to avoid noisy features at the beginning, we activate our augmentation procedure only after two training epochs. Afterwards, we apply feature-level augmentation with a probability of 0.5 on each mini-batch. To increase variety, a feature map has an equal probability of being transformed with any FLA operation. Hence, combining FLA operations is possible.

4. EXPERIMENTS

4.1. Data Sets

Tiny ImageNet. The Tiny ImageNet data set is a subset of ImageNet [13] containing 120,000 natural images belonging to 200 classes. The resolution of each image is 64×64 pixels. The training set contains 100,000 images, while the validation and test sets contain 10,000 samples each.

Caltech-256. The Caltech-256 [14] data set contains 30,607 images of 256 categories. For each object category, we divide the images into 40 for training, 20 for validation, leaving the remainder (at least 20 samples from each category) for the testing stage.

Food-101. The Food-101 [15] data set is formed of 101,000 images of 101 food types. The original split contains 750 training images and 250 test images for each category. We keep 250 training samples per category to validate the models, leaving us with 500 samples per category for training.

4.2. Evaluation Setup

Evaluation measures. For the evaluation, we first employ the accuracy between the ground-truth labels and the predictions of the neural models. Following [3], we also use the mean

Table 1: Accuracy scores and mFR values (in %) for translation, rotation and scaling operations on the Tiny ImageNet [13], Caltech-256 [14] and Food-101 [15] data sets. Reported results represent the average and the standard deviation over 5 runs. ↑ indicates higher values are better. ↓ indicates lower values are better. Best scores on each data set are highlighted in bold.

Data set	Model	Method	Accuracy ↑	mFR Translate ↓	mFR Rotate ↓	mFR Scale ↓	Trade-off ↑
Tiny ImageNet	ResNet-18	baseline	71.50 ± 0.20	12.94 ± 0.14	15.71 ± 0.40	22.29 ± 0.21	54.52
		BlurPool-3 [3]	71.16 ± 0.26	$\mathbf{10.61 \pm 0.33}$	14.98 ± 0.20	21.28 ± 0.25	55.53
		APS [2]	70.30 ± 0.22	15.32 ± 0.38	22.91 ± 0.23	27.86 ± 0.35	48.27
		FLA (ours)	$\mathbf{71.76 \pm 0.16}$	12.00 ± 0.14	$\mathbf{14.42 \pm 0.15}$	$\mathbf{21.11 \pm 0.15}$	$\mathbf{55.91}$
	DenseNet-121	baseline	76.50 ± 0.37	9.11 ± 0.08	14.19 ± 0.33	18.62 ± 0.19	62.52
		BlurPool-3 [3]	76.57 ± 0.24	$\mathbf{7.70 \pm 0.28}$	13.07 ± 0.39	17.68 ± 0.30	63.75
		APS [2]	76.00 ± 0.21	10.03 ± 0.09	17.77 ± 1.01	21.29 ± 0.26	59.63
		FLA (ours)	$\mathbf{76.60 \pm 0.44}$	8.54 ± 0.18	$\mathbf{12.59 \pm 0.20}$	$\mathbf{17.27 \pm 0.25}$	$\mathbf{63.80}$
Caltech-256	ResNet-18	baseline	$\mathbf{78.96 \pm 0.30}$	5.82 ± 0.15	6.16 ± 0.13	10.07 ± 0.09	71.61
		BlurPool-3 [3]	76.92 ± 0.26	$\mathbf{4.61 \pm 0.14}$	$\mathbf{4.83 \pm 0.16}$	$\mathbf{8.40 \pm 0.19}$	70.97
		APS [2]	78.12 ± 0.40	8.79 ± 0.15	9.86 ± 0.11	13.87 ± 0.10	67.28
		FLA (ours)	78.91 ± 0.06	4.99 ± 0.12	5.45 ± 0.07	9.24 ± 0.05	$\mathbf{72.35}$
	DenseNet-121	baseline	$\mathbf{83.98 \pm 0.14}$	4.00 ± 0.13	4.28 ± 0.04	7.11 ± 0.03	78.85
		BlurPool-3 [3]	82.91 ± 0.32	$\mathbf{3.26 \pm 0.32}$	$\mathbf{3.52 \pm 0.07}$	$\mathbf{6.17 \pm 0.05}$	78.59
		APS [2]	83.53 ± 0.11	4.57 ± 0.08	5.57 ± 0.05	8.63 ± 0.09	77.27
		FLA (ours)	83.54 ± 0.27	3.32 ± 0.11	3.67 ± 0.08	6.31 ± 0.13	$\mathbf{79.10}$
Food-101	ResNet-18	baseline	$\mathbf{76.29 \pm 0.41}$	6.41 ± 0.03	7.36 ± 0.06	12.39 ± 0.05	67.57
		BlurPool-3 [3]	74.38 ± 0.24	$\mathbf{4.77 \pm 0.11}$	$\mathbf{5.71 \pm 0.09}$	$\mathbf{9.90 \pm 0.09}$	67.58
		APS [2]	76.03 ± 0.20	8.77 ± 0.21	11.41 ± 0.21	16.29 ± 0.08	63.87
		FLA (ours)	76.28 ± 0.33	5.69 ± 0.02	6.63 ± 0.02	11.41 ± 0.04	$\mathbf{68.37}$
	DenseNet-121	baseline	$\mathbf{83.26 \pm 0.20}$	4.08 ± 0.07	4.85 ± 0.08	8.31 ± 0.08	77.51
		BlurPool-3 [3]	81.94 ± 0.26	$\mathbf{3.26 \pm 0.05}$	$\mathbf{4.00 \pm 0.03}$	$\mathbf{7.05 \pm 0.09}$	77.17
		APS [2]	82.77 ± 0.16	4.69 ± 0.04	6.19 ± 0.08	10.20 ± 0.06	75.74
		FLA (ours)	82.86 ± 0.20	3.46 ± 0.07	4.21 ± 0.10	7.58 ± 0.09	$\mathbf{77.78}$

flip rate (mFR) to measure the stability of the models to affine transformations (scaling, rotation and translation). The mFR is measured by how often the predicted label changes, on average, in images with consecutive perturbations. We run each neural model for 5 times, reporting the average scores and the corresponding standard deviations.

Following [16], we create the perturbed version of each test set. We perturb the original test data with rotation, scaling and translation operations. We perform each operation individually on each sample. We rotate the images with angles starting from -15 to $+15$ degrees, using a step of 1 degree. We translate the images by up to 20 pixels in each direction, using a step of 1 pixel. We scale the samples by a scaling factor between 0.4 and 1.15, using a step of 0.025.

In order to quantify the trade-off between the accuracy level of a model and its stability to affine transformations, we define the *trade-off* T as:

$$T = \text{accuracy} - \text{average}_{op}(\text{mFR}_{op}), \qquad (1)$$

where $op \in [\text{rotate}, \text{scale}, \text{translate}]$. A higher value for T represents a better trade-off.

Baselines. As neural models, we choose two very widely used architectures, namely ResNet-18 [11] and DenseNet-121 [12]. We train the models using common augmentations

applied on the input images, such as random shift, scaling and rotation. This represents our first baseline. In addition, we consider two state-of-the-art methods as baselines for improving the stability of these models, namely anti-aliasing (Blur-Pool) [3] and adaptive polyphase sampling (APS) [2]. For the BlurPool [3] method, we used the Triangle-3 filter, which obtains the best trade-off between accuracy and stability to affine transformations. We apply the same input augmentations for all baseline models, as well as for our own models based on feature-level augmentation (FLA).

Hyperparameter tuning. We train each model for a maximum of 100 epochs, halting the training when the value of the validation loss does not decrease for 10 consecutive epochs. We set the batch size to 16 samples and the learning rate to $5 \cdot 10^{-4}$. We optimize the models using Adam [17], keeping the default values for the parameters of Adam.

4.3. Results

We present the results on the Tiny ImageNet [13], Caltech-256 [14] and Food-101 [15] data sets in Table 1. As mentioned earlier, we compare our approach with a baseline based on standard data augmentation as well as two state-of-the-art methods, namely APS [2] and BlurPool [3].

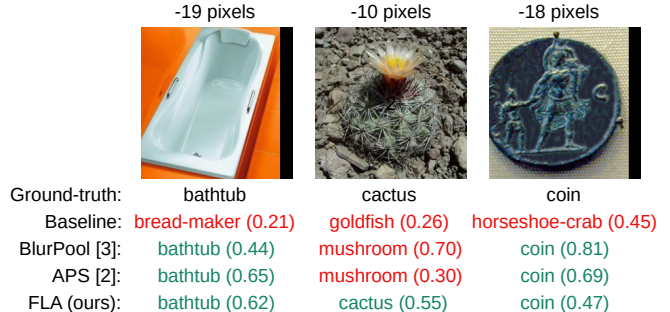First, we observe that the APS [2] method does not sur-
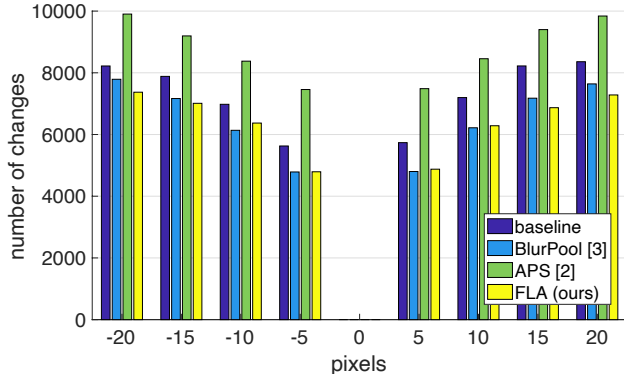
Fig. 2: The number of times the labels predicted by a ResNet-18 model having augmented images as input are different from the labels predicted for the original images (translated with 0 pixels) from Caltech-256. The images are translated from -20 pixels to $+20$ pixels. The ResNet-18 model is trained using various methods addressing the stability problem: input data augmentation (baseline), APS [2], Blur-Pool [3] and FLA (ours). Best viewed in color.

Fig. 3: Images perturbed with various translations with ground-truth and predicted labels (by the baseline, BlurPool, APS and FLA methods). Best viewed in color.

pass the accuracy level obtained by the baseline, regardless of the architecture or the data set. It also does not increase the models' stability to affine transformations, even obtaining worse results than the baseline in terms of mFR. The results of APS demonstrate that the 100% stability to circular shift claimed by Chaman et al. [2] does not increase the network's stability to generic affine transformations.

The BlurPool [3] method increases the models' stability to affine transformations, performing better than the baseline in terms of mFR. However, it also decreases the accuracy of the model in most cases. The only case when Blur-Pool increases the accuracy level is on Tiny ImageNet for the DenseNet-121 architecture (the baseline reaches an accuracy of 76.50%, while BlurPool reaches a higher accuracy of 76.57%). With one exception (the Tiny ImageNet data set), the models trained with the BlurPool method obtain lower trade-off indices compared to the baseline models.

Different from APS [2] and BlurPool [3], our method consistently attains the best trade-off between accuracy and stability to affine transformations across all models and data sets, as shown in Table 1. On Tiny ImageNet, we obtain the highest accuracy score and the lowest mFR for rotation and scaling, regardless of the network architecture. On Caltech-256, our method obtains the closest accuracy scores to the baselines, while also increasing the models' stability in terms of all mFR scores. Since our method increases the stability of ResNet-18 during multiple runs on Caltech-256, it seems to have an interesting effect on the standard deviation of the reported accuracy, reducing it from 0.30 to 0.06. On the Food-101 benchmark, our approach applied on the ResNet-18 architecture attains an accuracy level on par with the baseline ($76.28\% \pm 0.33$ vs. $76.29\% \pm 0.41$), but the stability of our approach to affine transformations is significantly higher compared to the stability of the baseline (5.69% vs. 6.41%, 6.63% vs. 7.36%, 11.41% vs. 12.39% in terms of the mFR for the translation, rotation and scaling operations, respectively).

In Figure 2, we illustrate how many times the labels predicted by a ResNet-18 model having translated images as input are different from the labels predicted for the original images (translated with 0 pixels). We observe that our method has a significantly smaller number of instabilities (different predictions) than the baseline and APS [2] methods, while the BlurPool [3] method attains similar results to our method. We also observe that the number of label differences increases with the magnitude of the translation operation for all methods, but the pace seems comparatively slower for our method.

In Figure 3, we show a couple of images from Caltech-256 that are misclassified by the baseline. We observe that BlurPool and APS induce correct labels for two out of three samples, while FLA is able to correct all labels.

5. CONCLUSION

In this paper, we have proposed a novel method to address the stability of CNNs against affine perturbations applied on the input images. To improve stability, our method relies on feature-level augmentation. In addition, we are the first who have conducted comparative experiments to assess the performance of state-of-the-art methods [2, 3] addressing the stability problem.

Although there is a recent trend towards focusing on vision transformers [18, 19, 20, 21], our study involves only convolutional architectures. However, the recent work of Liu et al. [22] shows that, upon making proper adjustments, CNNs can obtain comparable results to vision transformers. We thus believe that research related to CNN architectures, such as our own, is still valuable to the computer vision community.

In future work, we aim to study the effect of applying our approach at inference time. This might further improve the stability of neural models, but the stability gains have to be put in balance with the inevitable slowdown in terms of computational time. At the moment, our technique does not affect the inference time at all.

6. REFERENCES

[1] Aharon Azulay and Yair Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?," *Journal of Machine Learning Research*, vol. 20, pp. 1–25, 2019.

[2] Anadi Chaman and Ivan Dokmanic, "Truly Shift-Invariant Convolutional Neural Networks," in *Proceedings of CVPR*, 2021, pp. 3773–3783.

[3] Richard Zhang, "Making Convolutional Networks Shift-Invariant Again," in *Proceedings of ICML*, 2019, vol. 97, pp. 7324–7334.

[4] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow, "Improving the Robustness of Deep Neural Networks via Stability Training," in *Proceedings of CVPR*, 2016, pp. 4480–4488.

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of ICLR*, 2015.

[6] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard, "DeepFool: a simple and accurate method to fool deep neural networks," in *Proceedings of CVPR*, 2016, pp. 2574–2582.

[7] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, "Intriguing properties of neural networks," in *Proceedings of ICLR*, 2014.

[8] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami, "Practical black-box attacks against machine learning," in *Proceedings of ASIA CCS*, 2017, pp. 506–519.

[9] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S. Ecker, Matthias Bethge, and Wieland Brendel, "Benchmarking robustness in object detection: Autonomous driving when winter is coming," *arXiv preprint arXiv:1907.07484*, 2020.

[10] Georg Volk, Stefan Müller, Alexander von Bernuth, Dennis Hospach, and Oliver Bringmann, "Towards Robust CNN-based Object Detection through Augmentation with Synthetic Rain Variations," in *Proceedings of ITSC*, 2019, pp. 285–292.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of CVPR*, 2016, pp. 770–778.

[12] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of CVPR*, 2017, pp. 2261–2269.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[14] Gregory Griffin, Alex Holub, and Pietro Perona, "Caltech-256 Object Category Dataset," Tech. Rep., California Institute of Technology, 2007.

[15] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool, "Food-101 – Mining Discriminative Components with Random Forests," in *Proceedings of ECCV*, 2014, pp. 446–461.

[16] Dan Hendrycks and Thomas Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *Proceedings of ICLR*, 2019.

[17] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proceedings of ICLR*, 2015.

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proceedings of ICLR*, 2021.

[19] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah, "Transformers in Vision: A Survey," *arXiv preprint arXiv:2101.01169*, 2021.

[20] Nicolae-Catalin Ristea, Andreea-Iuliana Miron, Olivian Savencu, Mariana-Iuliana Georgescu, Nicolae Verga, Fahad Shahbaz Khan, and Radu Tudor Ionescu, "CyTran: Cycle-Consistent Transformers for Non-Contrast to Contrast CT Translation," *arXiv preprint arXiv:2110.06400*, 2021.

[21] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang, "CvT: Introducing Convolutions to Vision Transformers," *arXiv preprint arXiv:2103.15808*, 2021.

[22] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie, "A ConvNet for the 2020s," *arXiv preprint arXiv:2201.03545*, 2022.