ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders

Sanghyun Woo[1*] Shoubhik Debnath[2] Ronghang Hu[2]
Xinlei Chen[2] Zhuang Liu[2] In So Kweon[1] Saining Xie[3†]
[1]KAIST [2] Meta AI, FAIR [3]New York University

Code: https://github.com/facebookresearch/ConvNeXt-V2

Abstract

Driven by improved architectures and better representation learning frameworks, the field of visual recognition has enjoyed rapid modernization and performance boost in the early 2020s. For example, modern ConvNets, represented by ConvNeXt [52], have demonstrated strong performance in various scenarios. While these models were originally designed for supervised learning with ImageNet labels, they can also potentially benefit from self-supervised learning techniques such as masked autoencoders (MAE) [31]. However, we found that simply combining these two approaches leads to subpar performance. In this paper, we propose a fully convolutional masked autoencoder framework and a new Global Response Normalization (GRN) layer that can be added to the ConvNeXt architecture to enhance inter-channel feature competition. This co-design of self-supervised learning techniques and architectural improvement results in a new model family called ConvNeXt V2, which significantly improves the performance of pure ConvNets on various recognition benchmarks, including ImageNet classification, COCO detection, and ADE20K segmentation. We also provide pre-trained ConvNeXt V2 models of various sizes, ranging from an efficient 3.7M-parameter Atto model with 76.7% top-1 accuracy on ImageNet, to a 650M Huge model that achieves a state-of-the-art 88.9% accuracy using only public training data.

1. Introduction

Building on research breakthroughs in earlier decades [34,44,47,60,68], the field of visual recognition has ushered in a new era of large-scale visual representation learning. Pre-trained, large-scale vision models have become essential tools for feature learning and enabling a wide range of vision applications. The performance of a visual representation learning system is largely influenced by three main factors: the neural network architecture chosen, the method used for training the network, and the data used for training. In the field of visual recognition, progress in each of these areas contributes to overall improvements in performance.

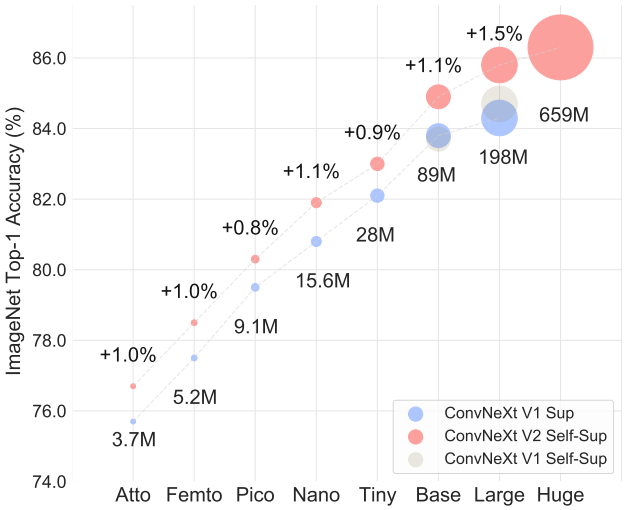

Figure 1. **ConvNeXt V2 model scaling**. The ConvNeXt V2 model, which has been pre-trained using our fully convolutional masked autoencoder framework, performs significantly better than the previous version across a wide range of model sizes.

Innovation in neural network architecture design has consistently played a major role in the field of representation learning. Convolutional neural network architectures (ConvNets) [34, 44, 47] have had a significant impact on computer vision research by allowing for the use of generic feature learning methods for a variety of visual recognition tasks [25, 33], rather than relying on manual feature engineering. In recent years, the transformer architecture [68], originally developed for natural language processing, has also gained popularity due to its strong scaling behavior with respect to model and dataset size [21]. More recently, ConvNeXt [52] architecture has modernized traditional ConvNets and demonstrated that pure convolutional models could also be scalable architectures. However, the most common method for exploring the design space for neural network architectures is still through benchmarking *supervised learning* performance on ImageNet.

[*] Work done during an internship at FAIR.
[†] Corresponding author.

In a separate line of research, the focus of visual representation learning has been shifting from supervised learning with labels to self-supervised pre-training with pretext objectives. Among many different self-supervised algorithms, masked autoencoders (MAE) [31] have recently brought success in masked language modeling to the vision domain and quickly become a popular approach for visual representation learning. However, a common practice in self-supervised learning is to use a *predetermined* architecture designed for supervised learning, and assume the design is fixed. For instance, MAE was developed using the vision transformer [21] architecture.

It is possible to combine the design elements of architectures and self-supervised learning frameworks, but doing so may present challenges when using ConvNeXt with masked autoencoders. One issue is that MAE has a specific encode-decoder design that is optimized for the sequence processing capabilities of transformers, which allows the compute-heavy encoder to focus on visible patches and thus reduce the pre-training cost. This design may not be compatible with standard ConvNets, which use dense sliding windows. Additionally, if the relationship between the architecture and the training objective is not taken into consideration, it may be unclear whether optimal performance can be achieved. In fact, previous research has shown that training ConvNets with mask-based self-supervised learning can be difficult [43], and empirical evidence suggests that transformers and ConvNets may have different feature learning behaviors that can affect representation quality.

To this end, we propose to *co-design* the network architecture and the masked autoencoder under the same framework, with the aim of making mask-based self-supervised learning effective for ConvNeXt models and achieving results similar to those obtained using transformers.

In designing the masked autoencoder, we treat the masked input as a set of sparse patches and use sparse convolutions [28] to process only the visible parts. The idea is inspired by the use of sparse convolutions in processing large-scale 3D point clouds [15, 76]. In practice, we can implement ConvNeXt with sparse convolutions, and at fine-tuning, the weights are converted back to standard, dense layers without requiring special handling. To further improve the pre-training efficiency, we replace the transformer decoder with a single ConvNeXt block, making the entire design fully convolutional. We have observed mixed results with these changes: the learned features are useful and improve upon the baseline results, but the fine-tuning performance is still not as good as the transformer-based model.

We then conduct a feature space analysis of different training configurations for ConvNeXt. We identify a potential issue of feature collapse at the MLP layer when training ConvNeXt directly on masked input. To address this issue, we propose adding a Global Response Normalization layer to enhance inter-channel feature competition. This change is most effective when the model is pre-trained with masked autoencoders, suggesting that reusing a fixed architecture design from supervised learning may be suboptimal.

In summary, we introduce ConvNeXt V2 which demonstrates improved performance when used in conjunction with masked autoencoders. We have found that this model significantly improves the performance of pure ConvNets across various downstream tasks, including ImageNet classification [60], COCO object detection [49] and ADE20K segmentation [81]. The ConvNeXt V2 models can be used in a variety of compute regimes and includes models of varying complexity: from an efficient 3.7M-parameter *Atto* model that achieves 76.7% top-1 accuracy on ImageNet, to a 650M *Huge* model that reaches a state-of-the-art 88.9% accuracy when using IN-22K labels.

2. Related Work

ConvNets. The design of ConvNets, which were first introduced in the 1980s [46] and trained using back-propagation, has undergone numerous improvements in terms of optimization, accuracy, and efficiency over the years [35, 36, 39, 44, 58, 61, 63, 75]. These innovations have mainly been discovered through the use of supervised training on the ImageNet dataset. In recent years, some efforts have been made to perform architecture search using self-supervised pre-text tasks such as rotation prediction and colorization, as in the case of UnNAS [50]. Recently, ConvNeXt [52] conducted a comprehensive review of the design space and demonstrated pure ConvNets can be as scalable as the vision transformers [21, 51], which have become the dominant architecture in many applications. ConvNeXt has particularly excelled in scenarios requiring lower complexity [7, 70, 71]. Our ConvNeXt V2 model, which is powered by self-supervised learning, provides a simple way to upgrade existing models and achieve a significant boost in performance across a wide range of use cases.

Masked Autoencoders. Masked image modeling, represented by masked autoencoders [31], is one of the latest self-supervised learning strategies. As a neural network pre-training framework, masked autoencoders have shown a broad impact on visual recognition. However, original masked autoencoders are not directly applicable to ConvNets due to their asymmetric encoder-decoder design. Alternative frameworks such as [3, 77] have attempted to adapt the approach for use with ConvNets, but with mixed results. MCMAE [23] uses a few convolutional blocks as input tokenizers. To the best of our knowledge, there are no pre-trained models that show self-supervised learning can improve upon the best ConvNeXt supervised results.

3. Fully Convolutional Masked Autoencoder

Our approach is conceptually simple and runs in a fully convolutional manner. The learning signals are generated by randomly *masking* the raw input visuals with a high masking ratio and letting the model *predict* the missing parts given the remaining context. Our framework is illustrated in Figure 2, and we will now describe its main components in more detail.

Masking. We use a random masking strategy with a masking ratio of 0.6. As the convolutional model has a hierarchical design, where the features are downsampled in different stages, the mask is generated in the last stage and upsampled recursively up to the finest resolution. To implement this in practice, we randomly remove 60% of the 32×32 patches from the original input image. We use minimal data augmentation, only including random resized cropping.

Encoder design. We use ConvNeXt [52] model as the encoder in our approach. One challenge in making masked image modeling effective is preventing the model from learning shortcuts that allow it to copy and paste information from the masked regions. This is relatively easy to prevent in transformer-based models, which can leave the visible patches as the only input to the encoder. However, it is more difficult to achieve this with ConvNets, as the 2D image structure must be preserved. While naive solutions involve introducing learnable masked tokens in the input side [3, 77], these approaches decrease the efficiency of pre-training and result in a train and test time inconsistency, as there are no mask tokens at test time. This becomes especially problematic when the masking ratio is high.

To tackle this issue, our new insight is to view the masked image from a "sparse data perspective", which was inspired by learning on sparse point clouds in 3D tasks [15, 76]. Our key observation is that the masked image can be represented as a 2D sparse array of pixels. Based on this insight, it is natural to incorporate sparse convolution into our framework to facilitate pre-training of the masked autoencoder. In practice, during pre-training, we propose to convert the standard convolution layer in the encoder with the submanifold sparse convolution, which enables the model to operate *only* on the visible data points [15, 27, 28]. We note that the sparse convolution layers can be converted back to standard convolution at the fine-tuning stage without requiring additional handling. As an alternative, it is also possible to apply a binary masking operation before and after the dense convolution operation. This operation has numerically the same effect as sparse convolutions, is theoretically more computationally intensive, but can be more friendly on AI accelerators like TPU.

Decoder design. We use a lightweight, plain ConvNeXt block as the decoder. This forms an asymmetric encoder-decoder architecture overall, as the encoder is heavier and has a hierarchy. We also considered more complex decoders such as hierarchical decoders [48, 59] or transformers [21, 31], but the simpler single ConvNeXt block decoder performed well in terms of fine-tuning accuracy and reduced pre-training time considerably, demonstrated in Table 1. We set the dimension of the decoder to 512.

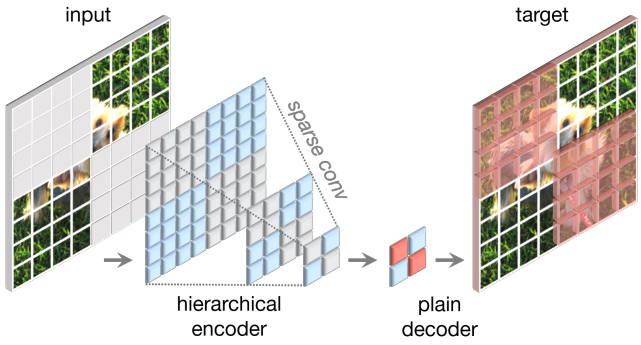

Figure 2. **Our FCMAE framework.** We introduce a fully convolutional masked autoencoder (FCMAE). It consists of a sparse convolution-based ConvNeXt encoder and a lightweight ConvNeXt block decoder. Overall, the architecture of our autoencoder is asymmetric. The encoder processes only the visible pixels, and the decoder reconstructs the image using the encoded pixels and mask tokens. The loss is calculated only on the masked region.

Reconstruction target. We compute the mean squared error (MSE) between the reconstructed and target images. Similar to MAE [31], the target is a patch-wise normalized image of the original input, and the loss is applied only on the masked patches.

FCMAE. We now present a **F**ully **C**onvolutional **M**asked **A**uto**E**ncoder (FCMAE) by combining the proposals described above. To evaluate the effectiveness of this framework, we use the ConvNeXt-Base model as the encoder and conduct a series of ablation studies. Throughout the paper, we focus on the end-to-end fine-tuning performance becuase of its practical relevance in transfer learning, and use that to assess the quality of the learned representation.

We pre-train and fine-tune using the ImageNet-1K (IN-1K) dataset for 800 and 100 epochs, respectively, and report the top-1 IN-1K validation accuracy for a single 224×224 center crop. Additional details about the experimental setup can be found in the appendix.

To understand the impact of using sparse convolution in our FCMAE framework, we first investigate how it affects the quality of the learned representation during masked image pre-training. Our empirical findings show that it is essential to prevent information leakage from the masked region in order to achieve good results.

w/o Sparse conv.	w/ Sparse conv.
79.3	83.7

dec. type	ft	hours	speedup
UNet w/ skip	**83.7**	12.9	-
UNet w/o skip	83.5	12.9	-
Transformer [31]	83.4	8.5	1.5×
ConvNeXt block	**83.7**	**7.7**	**1.7**×

(a) **Decoder design**. A simple convolutional block outperforms more complex decoder designs.

blocks	ft
1	**83.7**
2	83.5
4	**83.7**
8	83.6
12	83.3

(b) **Decoder depth**. A single block yields competitive fine-tuning performance.

dim	ft
128	83.5
256	**83.7**
512	**83.7**
768	83.6
1024	83.5

(c) **Decoder width**. A decoder width of 256 or 512 achieves the best performance.

Table 1. **MAE decoder ablation experiments** with ConvNeXt-Base on ImageNet-1K. We report fine-tuning (ft) accuracy (%). The pre-training schedule is 800 epochs. In the decoder design exploration, the wall-clock time is benchmarked on a 256-core TPU-v3 pod using JAX. The speedup is relative to the UNet decoder baseline. Our final design choices employed in the paper are marked in gray.

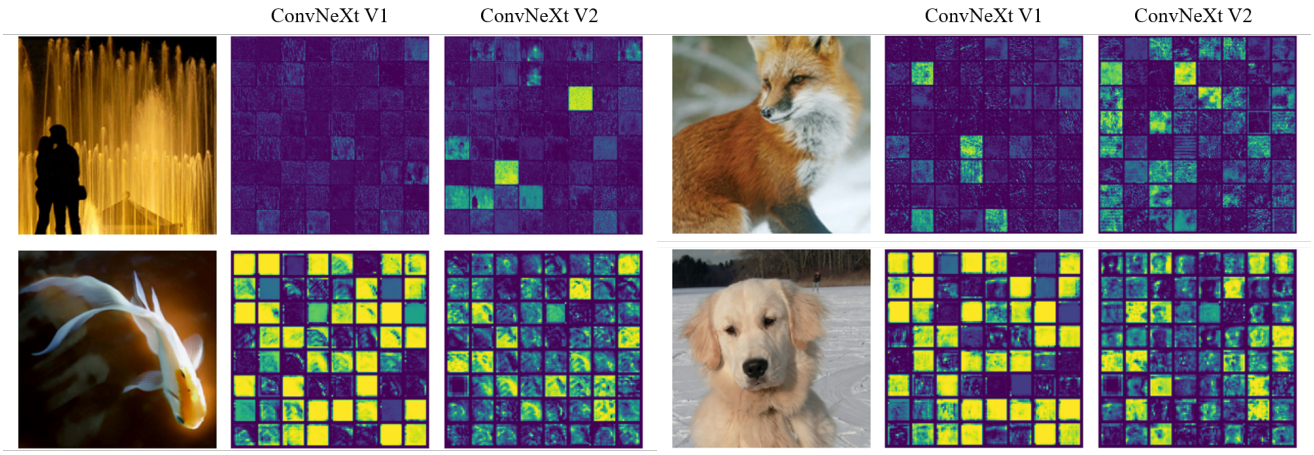

Figure 3. **Feature activation visualization.** We visualize the activation map for each feature channel in small squares. For clarity, we display 64 channels in each visualization. The ConvNeXt V1 model suffers from a feature collapse issue, which is characterized by the presence of redundant activations (dead or saturated neurons) across channels. To fix this problem, we introduce a new method to promote feature diversity during training: the global response normalization (GRN) layer. This technique is applied to high-dimensional features in every block, leading to the development of the ConvNeXt V2 architecture.

Next, we compare our self-supervised approach to supervised learning. Specifically, we obtain two baseline experimental results: the supervised 100 epoch baseline using the same recipe and the 300 epoch supervised training baseline provided in the original ConvNeXt paper [52]. We find that our FCMAE pre-training provides better initialization than the random baseline (*i.e.*, 82.7 → 83.7), but it still needs to catch up to the best performance obtained in the original supervised setup.

Sup, 100ep	Sup, 300ep. [52]	FCMAE
82.7	83.8	83.7

This is in contrast to the recent success of masked image modeling using transformer-based models [3, 31, 77], where the pre-trained models significantly outperform the supervised counterparts. This motivates us to investigate the unique challenges faced by the ConvNeXt encoder during masked autoencoder pre-training, which we discuss next.

4. Global Response Normalization

In this section, we introduce a new Global Response Normalization (GRN) technique to make FCMAE pre-training more effective in conjunction with the ConvNeXt architecture. We first motivate our approach through both qualitative and quantitative feature analyses.

Feature collapse. To gain more insight into the learning behavior, we first perform qualitative analysis in the feature space. We visualize the activations of a FCMAE pre-trained ConvNeXt-Base model and notice an intriguing "feature collapse" phenomenon: there are many dead or saturated feature maps and the activation becomes redundant across channels. We show some of the visualizations in Figure 3. This behavior was mainly observed in the dimension-expansion MLP layers in a ConvNeXt block [52].

Feature cosine distance analysis. To further validate our observation quantitatively, we perform a feature cosine distance analysis. Given an activation tensor $X \in R^{H \times W \times C}$,

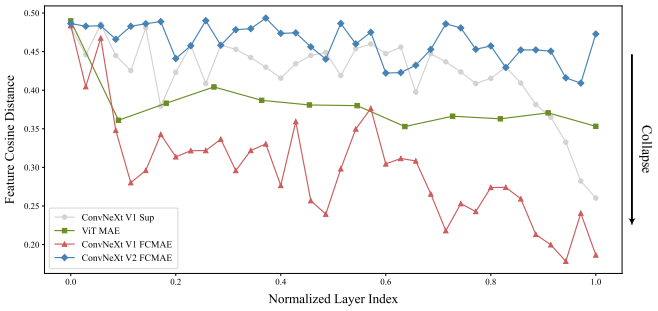

Figure 4. **Feature cosine distance analysis**. As the number of total layers varies for different architectures, we plot the distance values against the normalized layer indexes. We observe that the ConvNeXt V1 FCMAE pre-trained model exhibits severe feature collapse behavior. The supervised model also shows a reduction in feature diversity, but only in the final layers. This decrease in diversity in the supervised model is likely due to the use of the cross-entropy loss, which encourages the model to focus on class-discriminative features while suppressing the others.

$X_i \in R^{H \times W}$ is the feature map of the i-th channel. We reshape it as a HW dimensional vector and compute the average pair-wise cosine distance across the channels by $\frac{1}{C^2}\sum_i^C \sum_j^C \frac{1-cos(X_i,X_j)}{2}$. A higher distance value indicates more diverse features, while a lower value indicates feature redundancy.

To perform this analysis, we randomly select 1,000 images from different classes in the ImageNet-1K validation set and extract the high-dimensional features from each layer of different models, including the FCMAE models, the ConvNeXt supervised model [52] and the MAE pre-trained ViT model [31]. We then compute the distance per layer for each image and average the values across all images. The results are plotted in Figure 4. The FCMAE pre-trained ConvNeXt model exhibits a clear tendency towards feature collapse, consistent with our observations from the previous activation visualizations. This motivates us to consider ways to diversify the features during learning and prevent feature collapse.

Approach. There are many mechanisms in the brain that promote neuron diversity. For example, lateral inhibition [6, 30] can help to sharpen the response of the activated neuron and increase the contrast and selectivity of individual neurons to the stimulus while also increasing the diversity of responses across the population of neurons. In deep learning, this form of lateral inhibition can be implemented by response normalization [45]. In this work, we introduce a new response normalization layer called global response normalization (GRN), which aims to increase the contrast and selectivity of channels. Given an input feature, $X \in R^{H \times W \times C}$, the proposed GRN unit consists of three steps: 1) global feature aggregation, 2) feature normalization, and 3) feature calibration.

Algorithm 1 Pseudocode of GRN in a PyTorch-like style.

```
# gamma, beta: learnable affine transform parameters
# X: input of shape (N,H,W,C)

gx = torch.norm(X, p=2, dim=(1,2), keepdim=True)
nx = gx / (gx.mean(dim=-1, keepdim=True)+1e-6)
return gamma * (X * nx) + beta + X
```

First, we aggregate a spatial feature map X_i into a vector gx with a global function $\mathcal{G}(\cdot)$:

$$\mathcal{G}(X) := X \in \mathcal{R}^{H \times W \times C} \rightarrow gx \in \mathcal{R}^C. \quad (1)$$

This can be viewed as a simple pooling layer. We experimented with different functions in Table 2a. Interestingly, global average pooling, a widely used feature aggregator [37, 72], did not perform well in our case. Instead, we found that using norm-based feature aggregation, specifically, using L2-norm, resulted in better performance. This gives us a set of aggregated values $\mathcal{G}(X) = gx = \{||X_1||, ||X_2||, \ldots, ||X_C||\} \in \mathcal{R}^C$ where $\mathcal{G}(X)_i = ||X_i||$ is a scalar that aggregates the statistics of the i-th channel.

Next, we apply a response normalization function $\mathcal{N}(\cdot)$ to the aggregated values. Concretely, we use a standard divisive normalization as follows,

$$\mathcal{N}(||X_i||) := ||X_i|| \in \mathcal{R} \rightarrow \frac{||X_i||}{\sum_{j=1,\ldots,C} ||X_j||} \in \mathcal{R}, \quad (2)$$

where $||X_i||$ is the L2-norm of the i-th channel. [1] Intuitively, for the i-th channel, Eqn. 2 computes its *relative importance* compared to all the other channels. Similar to other forms of normalization [42, 45, 68], this step creates a feature competition across channels by mutual inhibition. In Table 2b, we also examine the use of other normalization functions and find that the simple divisive normalization works best, though standardization $(||X_i|| - \mu)/\sigma$ yields similar results when applied to the same L2-norm aggregated values.

Finally, we calibrate the original input responses using the computed feature normalization scores:

$$X_i = X_i * \mathcal{N}(\mathcal{G}(X)_i) \in \mathcal{R}^{H \times W} \quad (3)$$

The core GRN unit is very easy to implement, requiring only three lines of code, and has no learnable parameters. The pseudo-code for the GRN unit is in Algorithm 1.

To ease optimization, we add two additional learnable parameters, γ and β, and initialize them to zero. We also add a residual connection between the input and output of the GRN layer. The resulting final GRN block is $X_i = \gamma * X_i * \mathcal{N}(\mathcal{G}(X)_i) + \beta + X_i$. This setup allows a GRN layer

[1] To account for the increased number of channels at deeper layers, in practice, we also scale the normalized value by the channel count C.

case	ft
g.avg.	83.7
L1	84.3
L2	**84.6**

(a) **Global aggregation** $G(\cdot)$. L2 Norm-based aggregation function produces the best result.

case	ft
$(\|\|X_i\|\| - \mu)/\sigma$	84.5
$1/\sum \|\|X_i\|\|$	83.8
$\|\|X_i\|\|/\sum \|\|X_i\|\|$	**84.6**

(b) **Normalization operator,** $N(\cdot)$. Divisive normalization is an effective channel importance calibrator.

case	ft
w/o skip	84.0
w/ skip	**84.6**

(c) **Residual connection** helps with GRN optimization and leads to better performance.

case	ft
Baseline	83.7
LRN [45]	83.2
BN [41]	80.5
LN [2]	83.8
GRN	**84.6**

(d) **Feature normalization**. GRN outperforms other normalizations through global contrasting.

case	ft	#param
Baseline	83.7	89M
SE [37]	84.4	109M
CBAM [72]	84.5	109M
GRN	**84.6**	**89M**

(e) **Feature re-weighting**. GRN does effective and efficient feature re-weighting without parameter overhead.

case	ft
Baseline	83.7
drop at ft.	78.8
add at ft.	80.6
both	**84.6**

(f) **GRN in pre-training/fine-tuning**. To be effective, GRN should be used in both stages.

Table 2. **GRN ablations** with ConvNeXt-Base. We report fine-tuning accuracy on ImageNet-1K. Our final proposal is marked in gray.

to initially perform an identity function and gradually adapt during training. The importance of residual connection is demonstrated in Table 2c.

ConvNeXt V2. We incorporate the GRN layer into the original ConvNeXt block, as illustrated in Figure 5. We empirically found that LayerScale [65] becomes unnecessary when GRN is applied and can be removed. Using this new block design, we create various models with varying efficiency and capacity, which we refer to as the ConvNeXt V2 model family. These models range from lightweight (*e.g.* Atto [70]) to compute-intensive (*e.g.* Huge) ones. Detailed model configurations can be found in the appendix.

Impact of GRN. We now pre-train ConvNeXt V2 using the FCMAE framework and evaluate the impact of GRN. From visualization in Figure 3 and cosine distance analysis in Figure 4, we can observe that ConvNeXt V2 effectively mitigates the feature collapse issue. The cosine distance values are consistently high, indicating that feature diversity is maintained across layers. This behavior is similar to that of the MAE pre-trained ViT model [31]. Overall, this suggests that ConvNeXt V2 learning behavior can resemble ViT, under a similar masked image pre-training framework.

Next, we evaluate the fine-tuning performance.

V1 + Sup, 300ep.	V1 + FCMAE	V2 + FCMAE
83.8	83.7	84.6

When equipped with GRN, the FCMAE pre-trained model can significantly outperform the 300 epoch supervised counterpart. GRN improves the representation quality by enhancing the feature diversity, which was absent in the V1 model but has proven crucial for masked-based pre-training. Note this improvement is achieved *without* adding additional parameter overhead or increased FLOPS.[2]

Relation to feature normalization methods. Can other normalization layers [2,41,45,67,73] perform as well as the

[2]The additional affine parameters γ/β are negligible.

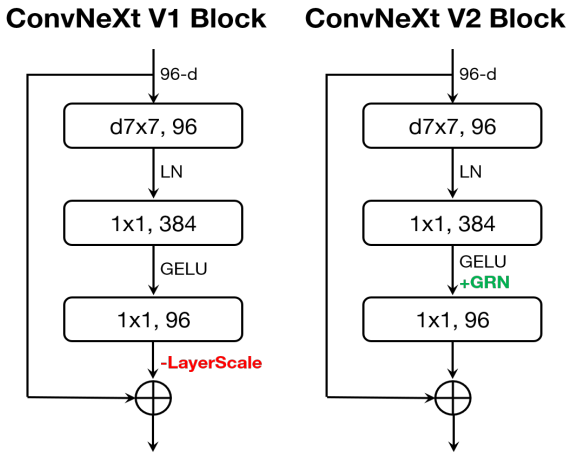

Figure 5. **ConvNeXt Block Designs**. In ConvNeXt V2, we add the GRN layer after the dimension-expansion MLP layer and drop LayerScale [65] as it becomes redundant.

global response normalization (GRN) layer? In Table 2d, we compare GRN with the three widely used normalization layers: Local Response Normalization (LRN) [45], Batch Normalization (BN) [41], and Layer Normalization (LN) [2]. We observe that only GRN can significantly outperform the supervised baseline. LRN lacks global context as it only contrasts channels within nearby neighbors. BN normalizes spatially along the batch axis, which is unsuitable for masked inputs. LN implicitly encourages feature competition through global mean and variance standardization but does not work as well as GRN.

Relation to feature gating methods. Another way to enhance competition across neurons is to use dynamic feature gating methods [37, 56, 69, 72, 78]. In Table 2e, we compare our GRN with two classic gating layers: squeeze-and-excite (SE) [37] and convolutional block attention module (CBAM) [72]. SE focuses on channel gating, while CBAM focuses on spatial gating. Both modules can increase the

Backbone	Method	#param	FLOPs	Val acc.
ConvNeXt V1-B	Supervised	89M	15.4G	83.8
ConvNeXt V1-B	FCMAE	89M	15.4G	83.7
ConvNeXt V2-B	Supervised	89M	15.4G	84.3 (+0.5)
ConvNeXt V2-B	FCMAE	89M	15.4G	**84.6** (+0.8)
ConvNeXt V1-L	Supervised	198M	34.4G	84.3
ConvNeXt V1-L	FCMAE	198M	34.4G	84.4
ConvNeXt V2-L	Supervised	198M	34.4G	84.5 (+0.2)
ConvNeXt V2-L	FCMAE	198M	34.4G	**85.6** (+1.3)

Table 3. **Co-design matters**. When the architecture and the learning framework are co-designed and used together, masked image pre-training becomes effective for ConvNeXt. We report the fine-tuning performance from 800 epoch FCMAE pre-trained models. The relative improvement is bigger with a larger model.

contrast of individual channels, similar to what GRN does. GRN is much simpler and more efficient as it does not require additional parameter layers (such as MLPs).

The role of GRN in pre-training/fine-tuning. Finally, we examine the importance of GRN in pre-training and fine-tuning. We present results in Table 2f where we either remove GRN from fine-tuning or add newly initialized GRN only at the time of fine-tuning. Either way, we observe a significant performance degradation, suggesting that keeping GRN in both pre-training and fine-tuning is important.

5. ImageNet Experiments

In this section, we present and analyze two key proposals, the FCMAE *pre-training framework* and ConvNeXt V2 *architecture*, which are co-designed to make masked-based self-supervised pre-training successful. We show these designs synergize well and provide a strong foundation for scaling the model to various sizes. Additionally, we compare our approach to previous masked image modeling approaches through experiments. Furthermore, we show that our largest ConvNeXt V2 Huge model, which has been pre-trained using the FCMAE framework and fine-tuned on the ImageNet-22K dataset, can achieve a new state-of-the-art of 88.9% top-1 accuracy on the ImageNet-1K dataset, using only publicly available data.

Co-design matters. In this paper, we conduct a unique study that involves *co-designing* both the self-supervised learning framework (FCMAE) and the model architecture improvement (GRN layer), through an empirical study of their learning behavior. The results presented in Table 3 demonstrate the importance of this approach.

We found that using the FCMAE framework without modifying the model architecture has a limited impact on representation learning quality. Similarly, the new GRN layer has a rather small effect on performance under the supervised setup. However, the combination of the two results in a significant improvement in fine-tuning performance. This supports the idea that both the model and learning framework should be considered together, particularly when it comes to self-supervised learning.

Backbone	Method	#param	PT epoch	FT acc.
ViT-B	BEiT	88M	800	83.2
ViT-B	MAE	88M	1600	83.6
Swin-B	SimMIM	88M	800	84.0
ConvNeXt V2-B	FCMAE	89M	800	84.6
ConvNeXt V2-B	FCMAE	89M	1600	**84.9**
ViT-L	BEiT	307M	800	85.2
ViT-L	MAE	307M	1600	85.9
Swin-L	SimMIM	197M	800	85.4
ConvNeXt V2-L	FCMAE	198M	800	85.6
ConvNeXt V2-L	FCMAE	198M	1600	**85.8**
ViT-H	MAE	632M	1600	86.9
Swin V2-H	SimMIM	658M	800	85.7
ConvNeXt V2-H	FCMAE	659M	800	85.8
ConvNeXt V2-H	FCMAE	659M	1600	**86.3**

Table 4. **Comparisons with previous masked image modeling approaches**. The pre-training data is the IN-1K training set. All self-supervised methods are benchmarked by the end-to-end fine-tuning performance with an image size of 224. We underline the highest accuracy for each model size and bold our best results.

Model scaling. In this study, we evaluated a range of 8 models with different sizes, from a low-capacity 3.7M Atto model to a high-capacity 650M Huge model. We pre-trained these models using the proposed FCMAE framework and compared the fine-tuning results to the fully supervised counterparts.

The results, shown in Figure 1, demonstrate strong model scaling behavior, with consistently improved performance over the supervised baseline across all model sizes. This is the first time the benefit of masked image modeling has been demonstrated in such a broad model spectrum, both in terms of effectiveness and efficiency. The complete tabulated results can be found in the appendix.

Comparisons with previous methods. We compare our approach to previous masked auto-encoder methods [3, 31, 77], which were all designed for transformer-based models. The results are summarized in Table 4. Our framework outperforms the Swin transformer pre-trained with SimMIM [77] *across all model sizes*. Compared to the plain ViT pre-trained with MAE [31], our approach performs similarly up to the Large model regime, despite using much fewer parameters (198M vs 307M). However, in the huge model regime, our approach slightly lagged behind. This might be because a huge ViT model can benefit more from self-supervised pre-training. As we will see next, the gap might be closed with additional intermediate fine-tuning.

ImageNet-22K intermediate fine-tuning. We also present ImageNet-22K intermediate fine-tuning results [3]. The training process involves three steps: 1) FCMAE pre-

Type	Backbone	size	#param	FLOPS	Val acc.
Conv	Efficient V2-XL	480^2	208M	94.0G	87.3
	ConvNeXt V1-XL	384^2	350M	179.0G	87.8
Hybrid	CoAtNet-4	512^2	275M	360.9G	88.1
	MaxViT-XL	384^2	475M	293.7G	88.5
	MaxViT-XL	512^2	475M	535.2G	88.7
Trans	MViTV2-H	384^2	667M	388.5G	88.6
	MViTV2-H	512^2	667M	763.5G	88.8
Conv	ConvNeXt V2-H	384^2	659M	337.9G	88.7
	ConvNeXt V2-H	512^2	659M	600.7G	**88.9**

Table 5. **ImageNet-1K fine-tuning results using IN-21K labels**. The ConvNeXt V2 Huge model equipped with the FCMAE pre-training outperforms other architectures and sets a new state-of-the-art accuracy of 88.9% among methods using public data only.

training, 2) ImageNet-22K fine-tuning, and 3) ImageNet-1K fine-tuning. We use 384^2 resolution images for pre-training and fine-tuning [38]. We compare our results to the state-of-the-art architecture designs, including convolution-based [52, 64], transformer-based [22], and hybrid designs [20,66]. All these results were trained with ImageNet-22K supervised labels. The results are summarized in Table 5. Our method, using a convolution-based architecture, sets a new state-of-the-art accuracy using publicly available data only (*i.e*. ImageNet-1K and ImageNet-22K).

6. Transfer Learning Experiments

We now benchmark the transfer learning performance. First, we evaluate the impact of our co-design, *i.e*. comparing ConvNeXt V1 + supervised *vs.* ConvNeXt V2 + FCMAE. We also directly compare our approach with Swin transformer models pre-trained with SimMIM [77]. The training and testing details are provided in the appendix.

Object detection and segmentation on COCO. We fine-tune Mask R-CNN [33] on the COCO dataset [49] and report the detection mAP$^{\text{box}}$ and the segmentation mAP$^{\text{mask}}$ on the COCO val2017 set. The results are shown in Table 6. We see a gradual improvement as our proposals are applied. From V1 to V2, the GRN layer is newly introduced and enhances performance. Upon this, the model further benefits from better initialization when moving from supervised to FCMAE-based self-supervised learning. The best performances are achieved when both are applied together. Additionally, our final proposal, ConvNeXt V2 pre-trained on FCMAE, outperforms the Swin transformer counterparts across all model sizes, with the largest gap achieved in the huge model regime.

Semantic segmentation on ADE20K. To summarize, we conduct experiments on the ADE20K [82] semantic segmentation task using the UperNet framework [74]. Our results show a similar trend to the object detection experiments, and our final model significantly improves over the V1 supervised counterparts. It also performs on par with the Swin transformer in the base and large model regimes but outperforms Swin in the huge model regime.

Backbone	Method	FLOPS	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
ConvNeXt V1-B	Supervised	486G	50.3	71.6	56.1	44.9	68.5	48.8
ConvNeXt V2-B	Supervised	486G	51.0	72.4	56.6	45.6	69.5	49.7
Swin-B	SimMIM	497G	52.3	–	–	–	–	–
ConvNeXt V2-B	FCMAE	486G	**52.9**	72.6	58.9	**46.6**	70.0	51.1
ConvNeXt V1-L	Supervised	875G	50.6	71.5	56.3	45.1	68.7	49.2
ConvNeXt V2-L	Supervised	875G	51.5	72.5	57.3	45.8	69.4	49.9
Swin-L	SimMIM	904G	53.8	–	–	–	–	–
ConvNeXt V2-L	FCMAE	875G	**54.4**	73.9	60.4	**47.7**	71.4	52.3
Swin V2-H	SimMIM	–	54.4	–	–	–	–	–
ConvNeXt V2-H	FCMAE	2525G	**55.7**	75.2	61.8	**48.9**	72.8	53.6

Table 6. **COCO object detection and instance segmentation results** using Mask-RCNN. FLOPS are calculated with image size (1280, 800). Swins' results are from [77]. All COCO fine-tuning experiments rely on ImageNet-1K pre-trained models.

Backbone	Method	input	mIoU	#param	FLOPS
ConvNeXt V1-B	Supervised	512^2	49.9	122M	1170G
ConvNeXt V2-B	Supervised	512^2	50.5	122M	1170G
Swin-B	SimMIM	512^2	**52.8**	121M	1181G
ConvNeXt V2-B	FCMAE	512^2	52.1	122M	1170G
ConvNeXt V1-L	Supervised	512^2	50.5	235M	1573G
ConvNeXt V2-L	Supervised	512^2	51.6	235M	1573G
Swin-L	SimMIM	512^2	53.5	234M	1601G
ConvNeXt V2-L	FCMAE	512^2	**53.7**	235M	1573G
Swin V2-H	SimMIM	512^2	54.2	–	–
ConvNeXt V2-H	FCMAE	512^2	**55.0**	707M	3272G
ConvNeXt V2-H	FCMAE, *22K ft*	640^2	**57.0**	707M	5113G

Table 7. **ADE20K semantic segmentation results** using UPerNet. Swins' results are from [77]. FLOPS are based on input sizes of (2048, 512) or (2560, 640). All ADE20K fine-tuning experiments rely on ImageNet-1K pre-trained model except FCMAE, *22K ft*, in which case the ImageNet-1K pre-training is followed by ImageNet-22K supervised fine-tuning.

7. Conclusion

In this paper, we introduce a new ConvNet model family called ConvNeXt V2 that covers a broader range of complexity. While the architecture has minimal changes, it is specifically designed to be more suitable for self-supervised learning. Using our fully convolutional masked autoencoder pre-training, we can significantly improve the performance of pure ConvNets across various downstream tasks, including ImageNet classification, COCO object detection, and ADE20K segmentation.

Acknowledgments. We thank Ross Wightman for the initial design of the small-compute ConvNeXt model variants and the associated training recipe. We also appreciate the helpful discussions and feedback provided by Kaiming He.

Appendix

This appendix provides implementation details, including model configurations, pre-training and fine-tuning recipes, and sparse and dense encoding methods for FCMAE pre-training (see §A). In §B, we present complete fine-tuning accuracy comparisons between ConvNeXt V1 and V2 on ImageNet 1K and 22K. In §C, we perform analyses on the efficiency of sparse encoding and general feature analysis using the class selectivity index. Finally, in §D, we conduct additional ablation studies on the masking ratio and GRN component analysis. We also compare FCMAE (masked image modeling) with MoCo V3 (contrastive learning).

A. Implementation Details

A.1. ConvNeXt V2 model configurations

The basic models, *i.e.*, Tiny (28M), Base (89M) and Large (198M), follow the same configurations of the stage, block (B), and channel (C) settings of the ConvNeXt V1 [52].

- ConvNeXt V2-T: C=96, B=(3, 3, 9, 3)
- ConvNeXt V2-B: C=128, B=(3, 3, 27, 3)
- ConvNeXt V2-L: C=192, B=(3, 3, 27, 3)

Given the same definitions above, we scale the model to provide a broad model size spectrum, targeting versatile scenarios. First, to obtain efficient models, we scale down as follows:

- ConvNeXt V2-A: C=40, B=(2, 2, 6, 2)
- ConvNeXt V2-F: C=48, B=(2, 2, 6, 2)
- ConvNeXt V2-P: C=64, B=(2, 2, 6, 2)
- ConvNeXt V2-N: C=80, B=(2, 2, 8, 2)

A, F, P, N denote Atto (3.7M), Femto (5.2M), Pico (9.1M), and Nano (15.6M) models designed originally in [70]. Next, to introduce the large-capacity variant, we scale up as follows:

- ConvNeXt V2-H: C=352, B=(3, 3, 27, 3)

H denotes Huge (659M) model, which is newly presented in this work.

A.2. ImageNet Experiments

Pre-training All models share the same pre-training setup, as noted in Table 8. We use the linear *lr* scaling rule [26]: *lr* = *base_lr*×batchsize / 256.

ImageNet-1K fine-tuning As the learning capacity varies by model size, we adopt different fine-tuning recipes for each model. We summarize them in Table 9, 10 and 11. We see longer fine-tuning epochs help small models. We adopt two different learning-rate layer decay strategies in this work: group-wise [52], where we treat three sequential layers as a single "layer" and use the same decaying value for them, and the layer-wise [3], where we assign a distinct value for each layer, both following the standard decaying rule. The default is a layer-wise strategy, but we apply the group-wise decaying strategy to Base and Large models.

ImageNet-22K intermediate fine-tuning We conduct ImageNet-22K intermediate fine-tuning with the FCMAE-pretrained ConvNeXt models. We use nano, tiny, base, large, and huge models. The setups are summarized in Table 12 and 13. Similarly, using larger layer-wise learning rate decay values for small models is helpful.

Sparse encoding implementations. We propose two possible implementations to enable FCMAE pre-training: 1) sparse encoding using sparse convolution [15, 27, 28] supported by external libraries [15, 18], and 2) simulating sparse encoding with the masked dense convolution, which can be easily implemented by applying binary masks *before and after* the standard convolution operation. As they produce numerically identical outputs, both can be adopted depending on different use cases. In this work, we adopt sparse encoding on the GPU environment, where we use MinkowskiEngine library [15] and PyTorch framework [57]; we use dense masked conv based encoding on TPU accelerators using Jax [5]. The experiments in the main paper are all conducted on TPU (v3-256) pods and we release a PyTorch reproduction.

A.3. Object detection and segmentation on COCO

For COCO experiments, we use the MMDetection [10] toolbox and the final model weights from ImageNet-1K pre-training as network initializations. All models are trained with a 3x schedule (36 epochs) and a batch size of 32. We utilize an AdamW optimizer [54] with a learning rate of 1e-4, a weight decay of 0.05 and sweep layer-wise learning rate decay in {0.9, 0.95}, stochastic depth rate in {0.2, 0.3, 0.4, 0.5}. We employ a large-scale jittering augmentation [24] (1024×1024 resolution, scale range [0.1, 2.0]). We use single-scale testing with soft-NMS [4] during inference.

A.4. Semantic segmentation in ADE20K

For ADE20K experiments, we use the MMSegmentation [17] toolbox. We use an AdamW optimizer [54] with the following hyperparameters: a weight decay of 0.05, a batch size of 16 and sweep layer-wise decay rate {0.8, 0.9}, learning rate {1e-4, 2e-4, 3e-4}, stochastic depth rate {0.1, 0.2, 0.3, 0.4}. All models are trained for 160K iterations with

config	value
optimizer	AdamW [54]
base learning rate	1.5e-4
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.95 [11]
batch size	4096
learning rate schedule	cosine decay [53]
warmup epochs [26]	40
training epochs	800 or 1600
augmentation	RandomResizedCrop

Table 8. **Pre-training setting.**

config	value
optimizer	AdamW
base learning rate	2e-4
weight decay	0.05 (F), 0.3 (A/P/N)
optimizer momentum	β_1, β_2=0.9, 0.999
layer-wise lr decay [3, 16]	0.9
batch size	1024
learning rate schedule	cosine decay
warmup epochs	0
training epochs	600
augmentation	RandAug (9, 0.5) [19]
label smoothing [62]	0.2
mixup [80]	0.0 (A), 0.3 (F/P), 0.5 (N)
cutmix [79]	0.0 (A), 0.3 (F/P), 0.5 (N)
drop path [40]	0.1 (A/N), 0.0 (F/P),
head init [52]	0.001
ema	0.9999

Table 9. **End-to-end IN-1K fine-tuning setting for Atto (A), Femto (F), Pico (P) and Nano (N) models.**

config	value
optimizer	AdamW
base learning rate	8e-4
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.999
layer-wise lr decay [3, 16]	0.9
batch size	1024
learning rate schedule	cosine decay
warmup epochs	40
training epochs	300
augmentation	RandAug (9, 0.5) [19]
label smoothing [62]	0.1
mixup [80]	0.8
cutmix [79]	1.0
drop path [40]	0.2
head init [52]	0.001
ema	0.9999

Table 10. **End-to-end IN-1K fine-tuning setting for Tiny model.**

config	value
optimizer	AdamW
base learning rate	6.25e-3 (B/L), 1.25e-3 (H)
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.999
layer-wise lr decay [3, 16]	0.6 (B/L), 0.75 (H)
batch size	1024
learning rate schedule	cosine decay
warmup epochs	20 (B/L), 10 (H)
training epochs	100 (B/L), 50 (H)
augmentation	RandAug (9, 0.5) [19]
label smoothing [62]	0.1
mixup [80]	0.8
cutmix [79]	1.0
drop path [40]	0.1 (B), 0.2 (L), 0.3 (H)
head init [52]	0.001
ema	0.9999

Table 11. **End-to-end IN-1K fine-tuning setting for Base (B), Large (L), and Huge (H) models.**

config	value
optimizer	AdamW
base learning rate	2.5e-4
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.999
layer-wise lr decay [3, 16]	0.8 (B/L/H), 0.9 (N/T)
batch size	4096
learning rate schedule	cosine decay
warmup epochs	5
training epochs	90
augmentation	RandAug (9, 0.5) [19]
label smoothing [62]	0.1
mixup [80]	0.8
cutmix [79]	1.0
drop path [40]	0.(N/T), 0.1 (B/L), 0.3 (H)
head init [52]	0.001
ema	None

Table 12. **End-to-end IN-22K intermediate fine-tuning settings.**

config	value
optimizer	AdamW
base learning rate	2.5e-5
weight decay	1e-8
optimizer momentum	β_1, β_2=0.9, 0.999
layer-wise lr decay [3, 16]	0.8 (B/L), 0.85 (H), 0.9 (N/T)
batch size	512
learning rate schedule	cosine decay
warmup epochs	None
training epochs	30 (B/L/H), 90 (N/T)
augmentation	RandAug (9, 0.5) [19]
label smoothing [62]	0.1
mixup [80]	None
cutmix [79]	None
drop path [40]	0.1(N/T), 0.2 (B), 0.3 (L), 0.5(H)
head init [52]	0.001
ema	0.9999 (N/T/B/L), None (H)

Table 13. **End-to-end IN-1K fine-tuning settings (after IN-22K intermediate fine-tuning).**

an input resolution of 512×512. In inference, a multi-scale test using resolutions that are [0.75,0.875,1.0,1.125,1.25] of 512×2048 is employed.

Similar to [77], we initialized the segmentation models using model weights after supervised fine-tuning on ImageNet-1K, as we found its performance superior to using the self-supervised pre-trained weights directly.

B. Complete comparisons with V1

In Tables 14 and 15, we present detailed experiment-level comparisons between ConvNeXt V1 [52, 70] and V2. In particular, Table 14 shows ImageNet-1K fine-tuning results using eight models: Atto, Femto, Nano, Pico, Tiny, Base, Large, and Huge, which range from low-compute (Atto, 3.7M) to large-capacity models (Huge, 660M). We see a consistent and significant improvement across all models. The best performance is achieved when the architecture is upgraded from V1 to V2 and the self-supervised learning framework FCMAE is used, demonstrating the effectiveness of the co-design. In Table 15, we present ImageNet-22K intermediate fine-tuning results. The pre-training and fine-tuning process consists of three steps: 1) FCMAE pre-training, 2) ImageNet-22K fine-tuning, and 3) ImageNet-1K fine-tuning. Here, we focus on five V2 mod-

Backbone	Method	#param	FLOPs	Val acc.
ConvNeXt V1-A	Supervised	3.7M	0.55G	75.7
ConvNeXt V2-A	Supervised	3.7M	0.55G	76.2 (+0.5)
ConvNeXt V2-A	FCMAE	3.7M	0.55G	**76.7** (+1.0)
ConvNeXt V1-F	Supervised	5.2M	0.78G	77.5
ConvNeXt V2-F	Supervised	5.2M	0.78G	78.0 (+0.5)
ConvNeXt V2-F	FCMAE	5.2M	0.78G	**78.5** (+1.0)
ConvNeXt V1-P	Supervised	9.1M	1.37G	79.5
ConvNeXt V2-P	Supervised	9.1M	1.37G	79.7 (+0.2)
ConvNeXt V2-P	FCMAE	9.1M	1.37G	**80.3** (+0.8)
ConvNeXt V1-N	Supervised	15.6M	2.45G	80.8
ConvNeXt V2-N	Supervised	15.6M	2.45G	81.2 (+0.4)
ConvNeXt V2-N	FCMAE	15.6M	2.45G	**81.9** (+1.1)
ConvNeXt V1-T	Supervised	28.6M	4.47G	82.1
ConvNeXt V2-T	Supervised	28.6M	4.47G	82.5 (+0.4)
ConvNeXt V2-T	FCMAE	28.6M	4.47G	**83.0** (+0.9)
ConvNeXt V1-B	Supervised	89M	15.4G	83.8
ConvNeXt V1-B	FCMAE	89M	15.4G	83.7
ConvNeXt V2-B	Supervised	89M	15.4G	84.3 (+0.5)
ConvNeXt V2-B	FCMAE	89M	15.4G	**84.9** (+1.1)
ConvNeXt V1-L	Supervised	198M	34.4G	84.3
ConvNeXt V1-L	FCMAE	198M	34.4G	84.4
ConvNeXt V2-L	Supervised	198M	34.4G	84.5 (+0.2)
ConvNeXt V2-L	FCMAE	198M	34.4G	**85.8** (+1.5)
ConvNeXt V2-H	FCMAE	660M	115G	**86.3**

Table 14. **ImageNet-1K fine-tuning results** with a single 224×224 crop. The improvement over the V1 supervised model is shown in parentheses.

Backbone	image size	#param	FLOPs	Val acc.
ConvNeXt V2-N	224^2	15.6M	2.45G	**82.1**
ConvNeXt V2-N	384^2	15.6M	7.21G	**83.4**
ConvNeXt V1-T	224^2	28.6M	4.47G	82.9
ConvNeXt V2-T	224^2	28.6M	4.47G	**83.9**(+1.0)
ConvNeXt V1-T	384^2	28.6M	13.1G	84.1
ConvNeXt V2-T	384^2	28.6M	13.1G	**85.1**(+1.0)
ConvNeXt V1-B	224^2	89M	15.4G	85.8
ConvNeXt V2-B	224^2	89M	15.4G	**86.8**(+1.0)
ConvNeXt V1-B	384^2	89M	45.2G	86.8
ConvNeXt V2-B	384^2	89M	45.2G	**87.7**(+0.9)
ConvNeXt V1-L	224^2	198M	34.4G	86.6
ConvNeXt V2-L	224^2	198M	34.4G	**87.3**(+0.7)
ConvNeXt V1-L	384^2	198M	101.1G	87.5
ConvNeXt V2-L	384^2	198M	101.1G	**88.2**(+0.7)
ConvNeXt V1-XL	224^2	350M	60.9G	87.0
ConvNeXt V1-XL	384^2	350M	179.0G	87.8
ConvNeXt V2-H	384^2	660M	337.9G	**88.7**
ConvNeXt V2-H	512^2	660M	600.8G	**88.9**

Table 15. **ImageNet-22K intermediate fine-tuning results** with a single 224×224 crop. The improvement over the V1 supervised model is shown in parentheses.

els: Nano, Tiny, Base, Large and Huge. We see consistent improvement over the V1 counterparts. In particular, the

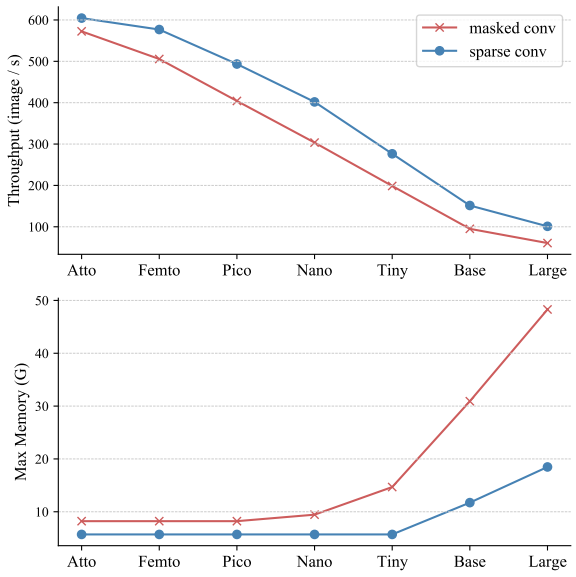

Figure 6. **Sparse encoding efficiency.** Under the pre-training setup, we measure the training throughput (image/s) and max GPU memory usage (G). The per GPU batch size is 64, and the throughput values are measured using 20 forward and backward steps. Our results show that the sparse convolution-based encoder allows for improved pre-training efficiency compared to the dense masked convolution-based counterpart.

V2 Base (86.8%/87.7%) and Large (87.3%/88.2%) models outperform the next-level model sizes of V1, which are the Large (86.6%/87.5%) and XLarge (87.0%/87.8%) models. The V2 Huge model also achieves a new state-of-the-art with a performance of 88.9%. Our proposal demonstrates that pure convolutional models can also be strong and scalable vision learners with mask-based pre-training.

C. Further Analyses

Sparse encoding efficiency. One of the key design choices in our FCMAE framework is the use of sparse convolution [15, 27, 28] during pre-training. The primary purpose is to block the flow of information from the masked region and facilitate masked autoencoder pre-training. As a byproduct, it also offers improved computational and memory efficiency during pre-training, as the kernels only apply to the visible pixels. However, we note that the sparse convolution libraries [15, 18] are not highly optimized for modern hardware, and the efficiency achieved usually depends on the frameworks [1, 5, 57] used in practice.

To better understand the actual pre-training efficiency achieved using sparse convolution, we conducted benchmark experiments using a controlled setup with Minkowski Engine v0.5.4 [15] and PyTorch [57]. We simulated the pre-training masked input (image size 224×224, masking ratio

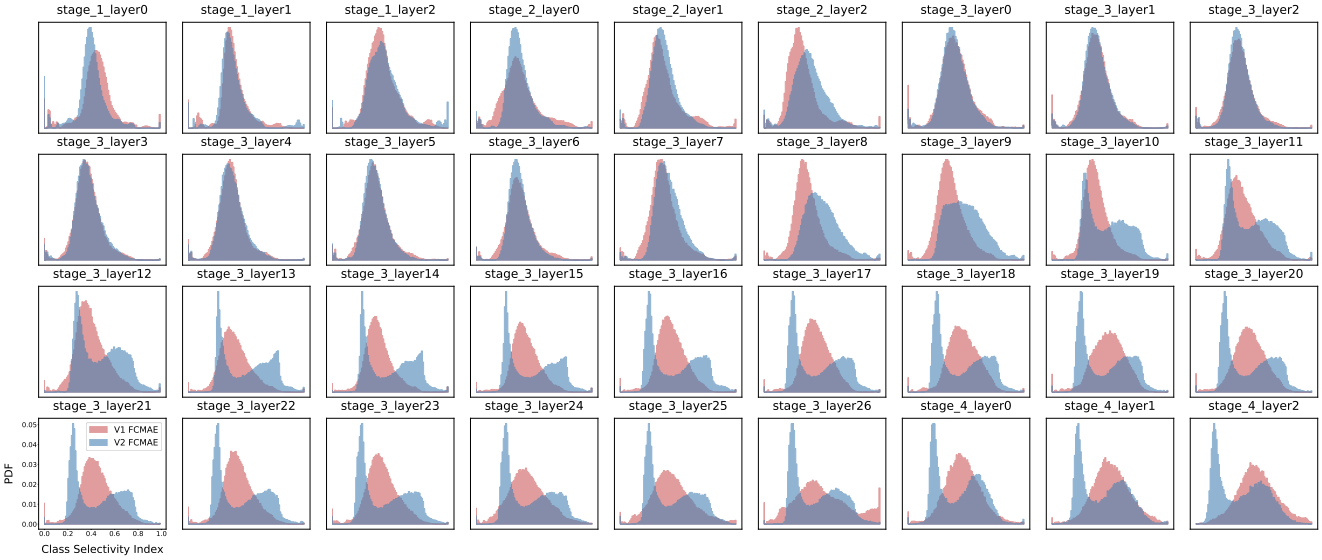

Figure 7. **Class selectivity index distribution.** The x-axis and y-axis show the class selectivity index and its density (PDF), respectively. Using the ImageNet-1K validation dataset, we calculated the class selectivity index distribution of both FCMAE pre-trained ConvNeXt V1 (red) and V2 (blue). While they tend to match closely in the early stages, the distribution becomes different in the deep layers. V2 tends to include more class-generic features in the later stages.

0.6, mask size 32×32) and compared the training throughput (image/s) and max GPU memory usage (G) between the sparse convolution-based and dense masked convolution-based encoders. While the results may vary depending on the experimental environment (we used PyTorch V1.8.0, CUDA 11.1, CuDNN 8.2, and NVIDIA RTX A6000 GPU), we observed a moderate increase in pre-training efficiency, with an average of 1.3× increase in throughput and a 2× decrease in max memory usage across the models. The gap becomes more salient as the model size increases.

Class Selectivity Index. FCMAE pre-trained ConvNeXt V2 has a distinctive feature characteristic compared to V1. We conducted a class selectivity index analysis on the FCMAE pre-trained weights for ConvNeXt V1 and V2 to understand this. The class selectivity index is a metric that measures the difference between the highest class-conditional mean activity and all other class-conditional mean activities. The final normalized value lies between 0 and 1, with 1 indicating that a filter activates only for a single class and 0 indicating that the filter activates uniformly for all classes. In Figure 7, we plot the class selectivity index distribution for all intermediate layers in the model, using the output of every residual block. The distribution is closely matched between V1 and V2 in the early stages, but they begin to diverge in the deep layers, such as stage 3 layer 12. As the layer becomes deeper, the plot shows that V2 (bimodal) tends to include more class-generic features than V1 (unimodal). Since class-agnostic features are more transferrable [55], this leads to better fine-tuning performance in downstream tasks. We leave more explorations as a future study.

case	GRN functions: aggregation	GRN functions: normalization	Val acc.
base	-	-	83.7
(a) X	-	-	83.9
(b) $X * \mathcal{G}(X)$	✓	-	83.9
(c) $X * \mathcal{N}(X)$	-	✓	unstable
(d) $X * \mathcal{N}(\mathcal{G}(X))$	✓	✓	84.6

Table 16. **GRN component analysis**. We report the fine-tuning performance after the 800 epoch FCMAE pre-training. Here, affine parameters and residual connection are omitted for clarity. The base denotes the ConvNeXt V1 fine-tuning performance. The **aggregation** and **normalization** are spatial L2-norm pooling and channel-wise divisive normalization, respectively. Case-(a) indicates a simple baseline of channel-wise scaling and shifting (with affine parameters) without explicit feature normalization.

D. Additional Experiments

GRN component analysis. The proposed Global Response Normalization (GRN) consists of three steps: global feature aggregation, feature normalization, and feature calibration. The main paper demonstrates that the combination of L2-norm based aggregation and divisive normalization works well in practice. Table 16 verifies the individual contribution of these components using ConvNeXt V2-Base as the encoder. When either component is dropped, performance significantly decreases, and the training becomes unstable if

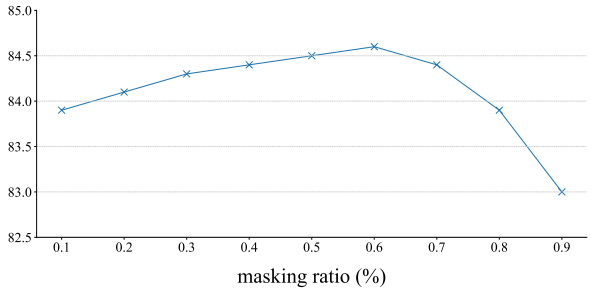

Figure 8. **Masking ratio.** We observe that a masking ratio of 0.6 provides the best result. The y-axis is ImageNet-1K accuracy (%).

feature normalization is not preceded by global aggregation. This supports the idea that both operations work together to make GRN effective.

Masking ratios. We conduct a hyper-parameter analysis on the masking ratio for a mask size of 32×32. The results, shown in Figure 8, suggest that a masking ratio in the range of 0.5 to 0.7 produces the best results, with a masking ratio of 0.6 providing the highest performance. The model's performance declines at the two extremes of either removing or leaving 90% of the input information, although it is more robust when more information is retained.

Comparison with contrastive SSL. In this work, we compare the performance of the two dominant self-supervised learning (SSL) approaches: contrastive learning [8, 9, 12–14, 29, 32] and masked image modeling [3, 31, 77]. Specifically, we compare the end-to-end fine-tuning performance of MoCoV3 [14], the current state-of-the-art contrastive learning method, with our proposed FCMAE framework using the same ConvNeXt V2-Base as the encoder. We follow the default pre-training and fine-tuning recipes for each approach and present the results below.

Sup, 300ep.	MoCo V3	FCMAE
84.3	83.7	84.9

We use the 300-epoch supervised learning baseline as a reference. The above table shows that FCMAE leads to better representation quality than MoCo V3 and also outperforms the supervised baseline. This is consistent with the recent observations that masked image modeling offers superior results over contrastive learning-based SSL for end-to-end fine-tuning. In this work, this success was also made possible with pure ConvNets.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *Operating Systems Design and Implementation*, 2016.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Hangbo Bao, Li Dong, and Furu Wei. BEiT: BERT pre-training of image transformers. In *ICLR*, 2022.

[4] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *ICCV*, 2017.

[5] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[6] Fergus W Campbell and John G Robson. Application of fourier analysis to the visibility of gratings. *The Journal of physiology*, 1968.

[7] Thomas Capelle. Finding the New Resnet18. https://wandb.ai/fastai/fine_tune_timm/reports/Finding-the-New-Resnet18--VmlldzoyMDI0MjU3, 2022.

[8] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[10] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv:1906.07155*, 2019.

[11] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pre-training from pixels. In *ICML*, 2020.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[13] Xinlei Chen and Kaiming He. Exploring simple Siamese representation learning. In *CVPR*, 2021.

[14] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised Vision Transformers. In *ICCV*, 2021.

[15] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019.

[16] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

[17] MMSegmentation contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

[18] Spconv Contributors. Spconv: Spatially sparse convolution library. https://github.com/traveller59/spconv, 2022.

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

[20] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. In *NeurIPS*, 2021.

[21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[22] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *CVPR*, 2021.

[23] Peng Gao, Teli Ma, Hongsheng Li, Jifeng Dai, and Yu Qiao. MCMAE: Masked Convolution Meets Masked Autoencoders. In *NeurIPS*, 2022.

[24] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, 2021.

[25] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[26] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv:1706.02677*, 2017.

[27] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018.

[28] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[29] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *NeurIPS*, 2020.

[30] H K Hartline, Henry G Wagner, and Floyd Ratliff. Inhibition in the eye of limulus. *The Journal of general physiology*, 1956.

[31] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[32] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[33] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

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

[36] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.

[37] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[38] Ronghang Hu, Shoubhik Debnath, Saining Xie, and Xinlei Chen. Exploring long-sequence masked autoencoders. *arXiv preprint arXiv:2210.07224*, 2022.

[39] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

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

[41] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[42] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.

[43] Li Jing, Jiachen Zhu, and Yann LeCun. Masked siamese convnets. *arXiv preprint arXiv:2206.07700*, 2022.

[44] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 2017.

[46] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[47] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[48] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[49] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[50] Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. Are labels necessary for neural architecture search? In *ECCV*, 2020.

[51] 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.

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

[53] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[54] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[55] Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. In *ICLR*, 2018.

[56] Jongchan Park, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Bam: Bottleneck attention module. In *BMVC*, 2018.

[57] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NeurIPS*, 2017.

[58] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020.

[59] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[60] 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. *IJCV*, 2015.

[61] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

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

[63] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[64] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *ICML*, 2021.

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

[66] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *ECCV*, 2022.

[67] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016.

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[69] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. In *CVPR*, 2020.

[70] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

[71] Ross Wightman and Jeremy Howard. Which image models are best? https://www.kaggle.com/code/jhoward/which-image-models-are-best, 2022.

[72] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV*, 2018.

[73] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

[74] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018.

[75] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[76] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pretraining for 3d point cloud understanding. In *ECCV*, 2020.

[77] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *CVPR*, 2022.

[78] Zongxin Yang, Linchao Zhu, Yu Wu, and Yi Yang. Gated channel transformation for visual recognition. In *CVPR*, 2020.

[79] 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.

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

[81] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using Places database. In *NeurIPS*, 2014.

[82] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. In *IJCV*, 2019.