# SPARSE AND HIERARCHICAL MASKED MODELING FOR CONVOLUTIONAL REPRESENTATION LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

This paper presents a simple yet powerful framework to pre-train convolutional network (convnet) with *Spar*se mas*K*ed modeling. *SparK* addresses key challenges in applying transformer-specialized masked modeling to convolutional models: (i) convolution operation cannot handle irregular, random-masked input; (ii) the single-scale nature of existing masked modeling is inconsistent with convnet's hierarchical structure. For (i), we sparsely gather all unmasked pixels to a sparse image and use sparse convolution for feature encoding. For the latter, we develop a hierarchical encoder-decoder to reconstruct from multi-scale encoded features to fully exploit the advantage of hierarchy. As the first hierarchical masked modeling method designed for convnets, SparK exploits their untapped potential: on three downstream tasks, SparK surpasses both state-of-the-art contrastive learning and *transformer-based* masked modeling by similarly large margins (around +1.0%). Improvements on object detection and instance segmentation are more substantial (>1.0%), verifying the strong transferability of features learned by SparK. We also demonstrate its favorable scaling behavior by observing more gains on larger models. Taken all results together, SparK suggests a promising future of generative pre-training on convnets. Codes and models will be made publicly available soon.

## 1 INTRODUCTION

Learning expressive representations from limited supervision is an enduring research topic. In computer vision, many well-established paradigms have been developed to create supervision without labels, which basically fall into two categories (Chen et al., 2020a): generative and discriminative. The *discriminative* ones, represented by contrastive learning (Chen et al., 2020a; He et al., 2020; Grill et al., 2020; Chen & He, 2021; Caron et al., 2020; Zbontar et al., 2021), used to lie at the center of visual self-supervised learning due to their well-proven representation properties (*e.g.*, good linear separability) and compelling downstream performance. It is the introduction of masked modeling (Devlin et al., 2018; Clark et al., 2020) together with transformer architecture (Vaswani et al., 2017) from natural language processing (NLP) that rekindled public interest in *generative* visual pre-training. These approaches (Bao et al., 2021; He et al., 2021; Xie et al., 2021; Fang et al., 2022) outperform their contrastive and supervised counterparts on *vision transformers* (Dosovitskiy et al., 2020; Touvron et al., 2021; Liu et al., 2021), thereby showing great promise. Curiously, the success has not yet spread to *convolutional networks*, especially since some advanced contrastive methods still remain state-of-the-art (Caron et al., 2020; 2021; Grill et al., 2020; Tian et al., 2020; Chen et al., 2021). We therefore ask: *what makes it difficult to apply masked modeling to convnets?*

In essence, the difficulty is rooted in the *gap* between NLP and computer vision in terms of data processing (He et al., 2021; Devlin et al., 2018; Yang et al., 2019): transformer processes text as a variable-length sequence of words (well-defined semantic units), while convnet has to extract semantics from objects of different sizes (like "units" at different scales). This big difference gives rise to the following challenges: **(i) Removing information of masked parts is difficult for convnet.** In vision transformer, an input image is divided to non-overlapping patches. Simply dropping masked patches or replacing them with mask tokens removes the information. This ease relies on the transformer being able to handle *irregular (variable-length)* and *non-overlapping* patches, thus cannot be achieved on convnets as they not only operate on *regular* grids, but also perform sliding window *with overlapping*. One may zero-out all masked pixels and feed the image into convnet. This, however, leads to a severe data distribution shift (shown in figure 1) and cannot be an ideal solution.

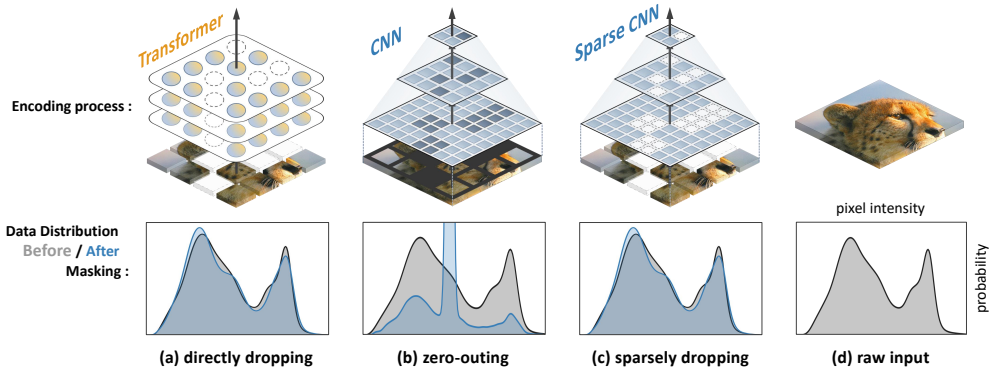


Figure 1: **Different masking strategies** with pixel intensity histograms plotted before (in gray) and after (blue) masking. (b) is a straightforward idea to apply masked modeling to convnets, which results in a distribution shift. (a) corresponds to MAE (He et al., 2021) that has no such side effect thanks to the transformer's ability to process variable-length input. We propose (c) to adapt convnet to irregular masked input without data distribution shift.

**(ii) Single-scale algorithm cannot learn multi-scale (hierarchical) convolutional features.** Multi-scale structure has long been a gold-standard in computer vision. It allows visual processing systems like SIFT/SURF descriptors (Lowe, 1999; Bay et al., 2006) and pyramid networks (He et al., 2015; Lin et al., 2017; Liu et al., 2016) to cope with changes in object scale. In contrast, masked modeling from NLP typically works in a single-scale manner, since the decoding process is based only on the features at the end of encoder. Applying it directly on convnets will miss the advantage of hierarchy.

In this paper, we develop a methodology called **Spar**se mas**K**ed modeling with hierarchy (SparK) to clear all the hurdles above. We first mask images in a patch-wise manner. Observing the sparse nature of point clouds coincides with those masked patches, we then sparsely gather unmasked patches to a sparse image, and use sparse convolutions for encoding. This strategy accurately eliminates the information from masked parts whilst allowing convnets to easily handle *irregular* masked images. For decoding, we fill in empty positions on the multi-scale encoded features with mask embeddings, and feed them into a multi-scale decoder. This way, we can embrace the advantage of feature hierarchy. Taken together, SparK makes masked modeling well suited for any convolutional models.

Evaluation is performed on standard downstream tasks including classification, object detection, and instance segmentation. We observe that both classical (He et al., 2016) and modern (Liu et al., 2022) convnets benefit a lot (by around +1.0%) from SparK, and with more performance gains on larger models that demonstrates scalability. On all downstream tasks, convnet-based SparK outperforms both (i) state-of-the-art contrastive learning and (ii) *transformer-based* masked pre-training by similarly large margins (around +1.0%), which shows great promise in generative pre-training for convnets. The improvements on COCO tasks are more significant than those on ImageNet, indicating the representations learned by SparK are highly transferable. To summarize, SparK provides:

- A first hierarchical masked modeling method for convnets, overcoming their inability to handle irregular masked inputs and fully exploiting their advantage of multi-scale structure.
- A leap in convnet's performance across downstream tasks, enabling convnets to surpass self-supervised vision transformers with similar model capacity.
- An insight into designing generative pre-training for convnets, and a promise of extending the success of transformer's pretrain-finetune paradigm to convnets.

The computer vision community has recently paid more attention to research on vision transformers, while convnets no longer appear in the spotlight (Liu et al., 2021; Bao et al., 2021; He et al., 2021). Nonetheless, convolutional neural networks, which condense the essence (scale- and translation-equivariant; locality; weight-sharing; hardware-friendly) of many classical vision processing systems (Lowe, 1999; Csurka et al., 2004; Jégou et al., 2010; Farabet et al., 2010), still play an irreplaceable role in tackling a wide range of real-world visual tasks that are more challenging or structural than classification (Liu et al., 2017; Jaderberg et al., 2015; Liu et al., 2022). We hope SparK's excellent performance compared to transformer-based pre-training will prompt us to revisit convnets as generic backbones for computer vision, and inspire more future research in exploiting their full potential.

# 2 RELATED WORK

## 2.1 HIERARCHICAL VISUAL PROCESSING SYSTEMS

**Hierarchical structure** is acknowledged as a gold-standard for visual representation systems. Many fundamental handcrafted feature descriptors (Lowe, 1999; Bay et al., 2006; Rublee et al., 2011) extract multi-scale visual representations via scale-space extremum on feature pyramid (say, octave). The crux behind this hierarchical design is to extract scale-invariant (or equivariant) features, thus, allows the system to cope with varying object sizes (scales). Widely used in visual tasks (Yang et al., 2009; Felzenszwalb et al., 2008), these descriptors also motivate the design principles of convolutional networks (He et al., 2016; Tan & Le, 2019; Liu et al., 2022). Some recent arts also elaborately design hierarchical modules that allow the information aggregation at different granularities to better tackle detection and segmentation tasks using convnets (Long et al., 2015; Lin et al., 2017; Liu et al., 2016).

## 2.2 RECENT PROGRESS ON VISUAL SELF-SUPERVISED LEARNING

**Recently, the contrastive learning** formulates self-supervise learning as instance classification (Van den Oord et al., 2018; He et al., 2020; Chen et al., 2020a). Efforts have been made (Grill et al., 2020; Caron et al., 2020; Chen & He, 2021) to overcome the core issue of mode collapse. Moco (He et al., 2020; Chen et al., 2020b) employs a momentum-updated offline bank to store sufficient negative samples. BYOL (Grill et al., 2020) and SimSiam (Chen & He, 2021) explore stop gradient and asymmetric heads, where they find these techniques helpful to get rid of the reliance on negative examples. More advanced methods are developed since then (Tian et al., 2020; Zbontar et al., 2021; Chen et al., 2021; Caron et al., 2021), and this line of work dominated the area of visual unsupervised learning until masked generative pre-training along with the vision transformer came into view.

**Masked image modeling,** inspired by the recent success of masked language modeling in natural language processing (NLP) (Devlin et al., 2018; Liu et al., 2019), has attracted growing interest for visual pre-training. The pioneering work Bao et al. (2021) pre-trains vision transformers by learning to predict token indices of masked patches. He et al. (2021) ingeniously takes advantage of transformer's ability to handle variable-length inputs and implements an efficient and scalable method. Both He et al. (2021) and Xie et al. (2021) regress raw RGBs to simplify the pre-training, while Wei et al. (2022) selects HoG (Dalal & Triggs, 2005) as targets due to their rich semantics. Gao et al. (2022) designs a transformer with a heavier patchifier to perform masked modeling. So far, enormous studies have successfully verified the efficacy of these algorithms on vision transformers (Zhou et al., 2021; Chen et al., 2022; Fang et al., 2022). However, on the other hand, their methodology is almost the same as that in NLP (Devlin et al., 2018; Liu et al., 2019), and is therefore difficult to be used for hierarchical convolutional models – on convnets, contrastive learning still remains state-of-the-art.

## 2.3 SPARSE CONVOLUTION FOR VISUAL REPRESENTATION

Convolution is widely used in 2D computer vision (Dalal & Triggs, 2005; He et al., 2016), which typically performs sliding window on regular grids (pixels). When facing with 3D point clouds, this operator quickly becomes unaffordable due to the cubic increasing number of grids (voxels). Considering point clouds are highly sparse and irregular, one can skip all empty voxels for speed. This motivates the **sparse convolution** (sparseconv) (Liu et al., 2015), which is heavily used in modern convnets for 3D visual tasks (Zhou & Tuzel, 2018; Sindagi et al., 2019). In practice, there are two different sparseconvs. One is vanilla sparseconv that has the equivalent effect to the ordinary "dense" convolution but with a faster speed. The other is submanifold sparseconv, which has a different effect from the ordinary: it only computes when the convolution kernel covers a non-empty position, thus keeps the sparsity during convolution and is faster than the former (Graham & van der Maaten, 2017).

# 3 APPROACH

Illustrated in figure 2, our SparK framework aims to pre-train a convolutional network encoder via hierarchical masked image modeling – masking a portion of image and learning to recover it. We are going to detail SparK by introducing a sparse masking strategy (section 3.1), a hierarchical encoder-decoder architecture (section 3.2), and a target of pre-training task (section 3.3).

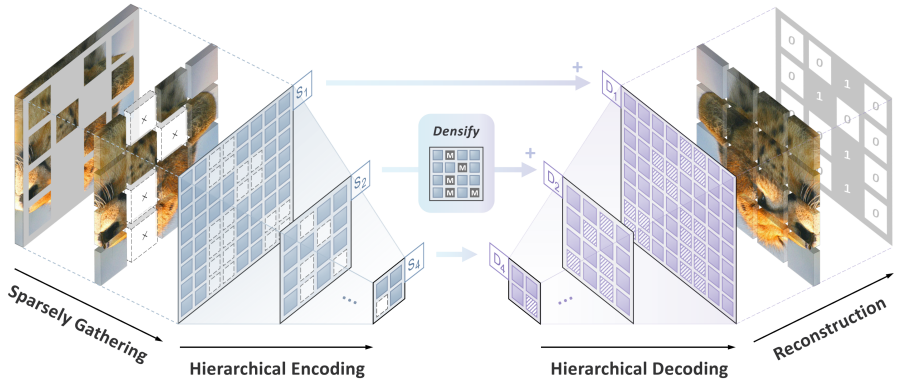


Figure 2: **Sparse masked modeling with hierarchy.** To adapt convolution to irregular masked input, visible patches are gathered into a sparse image and encoded by sparse convolution. To pre-train a hierarchical encoder, we employ a UNet-style architecture to decode multi-scale sparse feature maps, where all empty positions are filled with mask embedding. This "densifying" is necessary to reconstruct a dense image. Only the regression loss on masked patches will be optimized. After pre-training, only the encoder is used for downstream tasks.

### 3.1 Sparsely Gathering Unmasked Patches

We use patch-wise masking following the convention of masked image modeling. An image $x$ is divided into non-overlapping square patches, each of which will be masked independently with a given probability called mask ratio. The key to masked image modeling is how to eliminate the information from masked parts.

**Previous transformer-based work** either removes masked patches, or replaces them with a mask token to eliminate information. These strategies are *only* applicable to vision transformer since the transformer is able to handle *irregular* (variable-length) input, and it operates on *non-overlapping* patches. So, simply corrupting a patch (by removing or replacing) can avoid any information leakage from masked pixels. But this cannot be used directly on convnets which operate on regular grids with overlapping. A straightforward way is to set all masked pixels to zero to erase information. Although applicable to convnets, this solution has three evident shortcomings: (i) computing convolution on masked regions is redundant; (ii) it disturbs the distribution of pixels, as illustrated in figure 1; (iii) patterns on mask maps will vanish after several convolutional layers. We examine and explain this problem in figure 3, similar to the case in Graham & van der Maaten (2017), where we compare this zero-outing with our sparse masking strategy (introduced next). The problem is particularly acute when using modern deep convnets because of the large number of successive convolutional blocks.

**To overcome the problems, we propose to sparsely gather the unmasked patches** into a sparse image, and then use **sparse convolutions**[1] to encode it. This strategy: (i) ensures no information is leaked; (ii) is well adapted to convnet architectures; (iii) is efficient as sparse convolution computes only at visible places; (iv) solves the issues of pixel distribution shift as well as the "mask pattern vanishing". As shown in figure 3, sparse convolution will skip all masked positions on sparse feature maps, and only computes at unmasked points. In other words, all empty positions remain empty after sparse convolution, and vice versa. This helps to prevent the shape of the mask pattern from changing with convolution, thus ensures a consistent masking effect and ratio throughout all convolution layers.

### 3.2 Hierarchical Encoding and Decoding

**By "hierarchical" encoding,** we mean the encoder will generate a set of feature maps with different resolutions, namely different *scales*. Taking a ResNet-style model for example, it typically contains 4 stages each with a series of convolutional blocks and a downsampling module. The feature resolution is downsampled by a factor of 2 after every stage. For an image shaped as $H \times W$, a sparse ResNet-50 produces feature maps at 4 scales with resolutions of $\frac{H}{4} \times \frac{W}{4}$, $\frac{H}{8} \times \frac{W}{8}$, $\frac{H}{16} \times \frac{W}{16}$, and $\frac{H}{32} \times \frac{W}{32}$. Let $S_1$, $S_2$, $S_3$, and $S_4$ be the sparse features, respectively. They will be used to decode.

[1] By "sparse convolution", we mean the *submanifold sparse convolution* that computes only when the kernel center covers a non-empty element. Please refer to Graham & van der Maaten (2017) for more details.

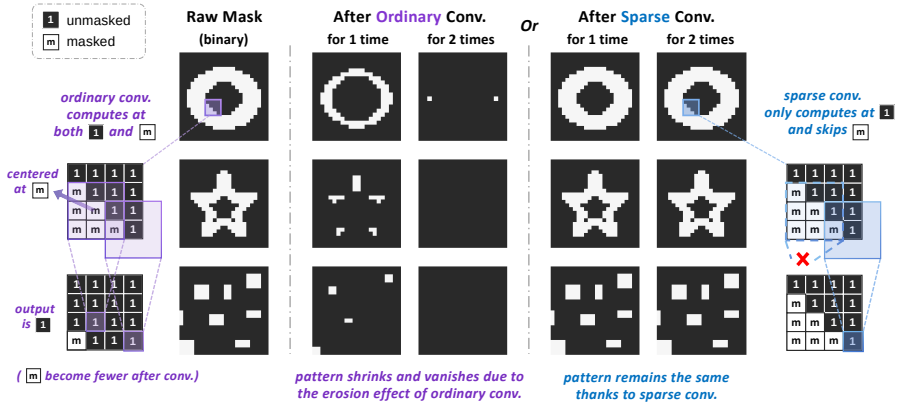


Figure 3: **Using sparse convolution to address "mask pattern vanishing" issue.** Three examples of mask are shown. As in left, when computing ordinary "dense" convolution centered at a masked position, the result would be non-zero if the filter covers any non-zero (unmasked) points. Repeating this process will *erode* masked regions (in white) and *dilate* unmasked ones (black), leading to the vanishing problem. We use sparse convolution to overcome this undesired property, which skips all masked positions and only computes at unmasked points.

**Overall, the decoder** follows the design of UNet Ronneberger et al. (2015). We use a relatively light decoder that contains three successive blocks $\{\mathcal{B}_3, \mathcal{B}_2, \mathcal{B}_1\}$ with upsampling. To reconstruct a dense image, we need to *fill in* the empty positions on sparse feature maps before feeding them to the decoder. This is called "**densifying**". Taking the smallest sparse feature $S_4$ as example, all empty positions (inactive sites) on $S_4$ are filled with a mask embedding $\texttt{[M}_4\texttt{]}$ to get a dense feature $S'_4$. A projection layer $\phi_4$ is applied then, in case encoder and decoder have different network widths:

$$D_4 = \phi_4(S'_4). \tag{1}$$

So $D_4$ is the input of decoder's first block $\mathcal{B}_3$. It has the same resolution of $S_4$ with $\frac{H}{32} \times \frac{W}{32}$. Similarly, we can get $D_3$, $D_2$, and $D_1$ (with shapes of $\frac{H}{16} \times \frac{W}{16}$, $\frac{H}{8} \times \frac{W}{8}$, $\frac{H}{4} \times \frac{W}{4}$) via:

$$D_i = \mathcal{B}_i(D_{i+1}) \, + \, \phi_i(S'_i) \qquad (\forall\, i \in \{3, 2, 1\}). \tag{2}$$

Note that four different mask embeddings $\texttt{[M}_{4\sim1}\texttt{]}$ and projection layers $\phi_{4\sim1}$ are required: they belong to each of the scales. The final output of decoder is $D_1$, with a shape of $\frac{H}{4} \times \frac{W}{4}$.

### 3.3 Optimization Target and Transferring to Downstream

**To reconstruct an image from $D_1$,** a head module $h$ is needed, which should include two more upsampling layers to reach the original resolution of input $H \times W$. As for the reconstruction target, we choose per-patch normalized pixels as targets with an $L^2$-loss, and calculate errors only on masked positions. These designs have been proven to facilitate models to learn more informative features in He et al. (2021), and are also verified by ablation study later in section 4.5.

**After pre-training,** we discard the decoder and only use the encoder for downstream tasks. When fine-tuning, the pre-trained sparse encoder can be directly generalized to dense images without any tuning, due to the fact that dense input is a special case of the sparse, where every position is active.

## 4 Empirical Results

### 4.1 Simple Implementation of Pre-training

**Components.** SparK can use any convolutional network as the encoder by converting all its layers (Conv2d, BatchNorm, MaxPooling, *etc.*) to sparse form. After encoding, mask embeddings $\texttt{[M}_{4\sim1}\texttt{]}$ are required to fill in sparse feature maps with different resolutions. These embeddings are implemented as random-initialized learnable feature vectors. For decoding, we use a lightweight UNet decoder (Ronneberger et al., 2015) with three blocks. See Appendix A for more detailed structure. Positional embeddings are not used since convnet already encodes the spatial information.

**Hyperparameters.** For simplicity, SparK uses the **same** pre-training hyperparameters for all model architectures (ResNets, ConvNeXts, *etc.*) and model sizes, even though tuning each would improve more at face value. We pre-train models with 1.28 million unlabeled images from ImageNet (Deng et al., 2009) training set for 1600 epochs by default. Only minimal data augmentation is required (a random horizontal flipping and a random resize cropping to $224 \times 224$). We use the same masking patch size (32) and ratio (60%) as in SimMIM (Xie et al., 2021). We train with the LAMB optimizer (You et al., 2019), a batch size of 4096, and a cosine-annealing learning rate $lr = 0.0002 \times batchsize/256$ with linear warmup. Following the advanced work (Caron et al., 2021; Zhou et al., 2021), the weight decay also follows a cosine schedule from 0.04 to 0.2.

### 4.2 IMAGENET EVALUATION

**Performance comparison with self-supervised transformers.** We first validate SparK framework on ImageNet (Deng et al., 2009). We experiment SparK with the pure convolutional model, ConvNeXt (Liu et al., 2022), and make comparisons between it and self-supervised pre-trained transformers of Dosovitskiy et al. (2020); Liu et al. (2021) in table 1. Smaller models ({ViT,ConvNeXt}-S, Swin-T) and the bigger ones ({ViT, Swin, ConvNeXt}-B) are compared separately.

By comparing the results vertically in table 1, one can find the convolutional models, with SparK, overwhelmingly outperform all transformer-based pre-training methods by large margins (+0.8~2.7), though SparK neither employs external models (DALL-E dVAE (Ramesh et al., 2021)), nor profits from advanced (MIM+CL) pre-training. This is indeed a surprising result since the transformers are well-known data-hungry models with much less inductive bias than convolutional nets, and therefore are considered to benefit more from large-scale self-supervised training. The result here conveys a new message: convnets may have much more potential than expected, and their capability in visual representation may not be inferior to that of transformers. The key depends on how to use powerful pre-training algorithms (*e.g.*, SparK – masked modeling) to turn this *potential* into *capability*.

**Efficiency.** Similar to MAE (He et al., 2021), SparK has high encoding efficiency among all methods in table 1, especially compared to contrastive learning that processes two or more images in a forward pass. For instance, DINO and iBOT by default (Caron et al., 2021; Zhou et al., 2021) use multi-crop with 2 global crops of $224 \times 224$ and 10 locals of $96 \times 96$, leading to $2 + 10\,(96/224)^2 \approx 3.8$ times the cost of single image encoding. In contrast, SparK requires only 40% of the overhead thanks to the sparsity of masked input: 60% of patches are masked, and sparse convolution only needs to process the rest. This efficiency allows SparK to scale-up as easily as MAE, demonstrated in section 4.4.

Table 1: **Comparing self-supervised transformers and convnets with SparK pre-training on ImageNet.** All methods fine-tune with the **resolution of 224**. Top-1 accuracy on ImageNet-val is reported. "Extra model" indicates whether an algorithm uses DALL-E's dVAE (trained on 250 million extra data) in pre-training. Entries with † are quoted from Zhou et al. (2021) or Li et al. (2022). ‡ is our re-implementation using the official codes.

| Pre-train (PT) | PT task | Enc. cost | Extra model | Small Backbone Arch. | Small Backbone Acc. | Base Backbone Arch. | Base Backbone Acc. |
|---|---|---|---|---|---|---|---|
| *Vision Transformer Backbone* | | | | | | | |
| MoCov3 (Chen et al., 2021) | CL | 5.0× | | ViT-S | 81.4 | ViT-B | 83.2 |
| DINO (Caron et al., 2021) | CL | 9.5× | | ViT-S | 82.0 | ViT-B | 82.8 |
| BEiT (Bao et al., 2021) | MIM | 2.5× | ✓ | ViT-S | 81.4† | ViT-B | 83.2 |
| CIM (Fang et al., 2022) | MIM | 2.5× | ✓ | ViT-S | 81.6 | ViT-B | 83.3 |
| CAE (Chen et al., 2022) | MIM | 2.5× | ✓ | ViT-S | 81.8 | ViT-B | 83.6 |
| MAE (He et al., 2021) | MIM | 0.6× | | ViT-S | 81.5‡ | ViT-B | 83.6 |
| SimMIM (Xie et al., 2021) | MIM | 2.5× | | ViT-S | 81.7 | ViT-B | 83.8 |
| SimMIM (Xie et al., 2021) | MIM | 2.5× | | Swin-T | 82.2† | Swin-B | 84.0 |
| iBOT (Zhou et al., 2021) | MIM+CL | 9.5× | | ViT-S | 82.3 | ViT-B | 84.0 |
| *Convolutional Backbone* | | | | | | | |
| SparK (ours) | MIM | 1× | | ConvX-S | **84.1** | ConvX-B | **84.8** |

[2]"Effective epoch" takes into account the total amount of images processed in pre-training. For instance, a typical contrastive learning encodes two images per forward pass, so the effective epoch is twice the literal value.

Table 2: **Thoroughly comparing convnets and transformers on three downstream tasks.** For ImageNet, all methods fine-tune with 224 resolution. For COCO, two commonly-used configurations are compared separately for fairness. Average precision (AP) of detection box and segmentation mask on `val2017` are reported. "$AP_{75}$" denotes AP with $IOU = 75\%$. SparK is emphasized as the only method pre-training *convolutional* models.

| Pre-train (PT) | Arch. | PT task | Eff.[2] epoch | Cls. (Acc.) | Det. $AP^{bb}$ | Det. $AP^{bb}_{75}$ | Seg. $AP^{mk}$ | Seg. $AP^{mk}_{75}$ |
|---|---|---|---|---|---|---|---|---|
| *Small backbones with 3× Cascade RCNN (Cai & Vasconcelos, 2018) and FPN Lin et al. (2017)* | | | | | | | | |
| Supervised (Zhou et al., 2021) | ViT-S | Sup. | 300 | 79.9 | 46.2 | 49.6 | 40.1 | 42.8 |
| Supervised (Liu et al., 2022) | ConvX-S | Sup. | 300 | 83.1 | 51.9 | 56.5 | 45.0 | 49.1 |
| iBOT (Zhou et al., 2021) | ViT-S | M+C | 3200 | 82.3 | 49.9 | 53.9 | 43.1 | 66.4 |
| SparK (ours) | ConvX-S | MIM | 1600 | **84.1** | **52.8** | **57.4** | **45.4** | **49.6** |
| *Base backbones with 3× Mask-RCNN (He et al., 2017) and FPN* | | | | | | | | |
| MoCov3 (Chen et al., 2021) | ViT-B | CL | 1600 | 83.2 | 47.9 | – | 42.7 | – |
| BEiT (Bao et al., 2021) | ViT-B | MIM | 800 | 83.2 | 49.8 | – | 44.4 | – |
| MAE (He et al., 2021) | ViT-B | MIM | 1600 | 83.6 | 50.3 | – | 44.9 | – |
| SimMIM (Xie et al., 2021) | Swin-B | MIM | 800 | 84.0 | 50.4 | 55.5 | 44.4 | 47.9 |
| Spark (ours) | ConvX-B | MIM | 1600 | **84.8** | **51.2** | **56.1** | **45.1** | **48.9** |

### 4.3 Transferring to Downstream Tasks

Previous results on ImageNet classification have exposed the potential of SparK pre-training. In this part we further evaluate the representation quality on fundamental downstream tasks, including object detection and instance segmentation on COCO (Lin et al., 2014). These tasks are challenging, serving as professional feature evaluators because they place higher demands than classification: models need to predict not only *what*, but also *where* the objects (instances) are. Here, we consider two different settings: comparison with self-supervised vision transformers, and then with convolutional networks.

**Performance *vs.* self-supervised transformers.** In table 2, we comprehensively compare the fine-tuning results on *three* downstream tasks: image classification (Cls.), object detection (Det.), and instance segmentation (Seg.). SparK, as the only *convolutional* pre-training method, perform consistently well in all three downstream tasks, surpassing all *vision-transformer-based* counterparts by substantial margins. Even when compared to the strongest SimMIM (Xie et al., 2021) with swin-transformer, SparK still yields superior results by $+0.8$, $+0.8$, $+0.7$ on three tasks respectively. With the base-sized backbone, all transformer-based masked modeling methods perform closely, while SparK presents a leap in performance. These observations are consistent with those in section 4.2 and once again validate the effectiveness of generative pre-training on hierarchical convnets.

**Performance *vs.* self-supervised convnets.** We then compare SparK to state-of-the-art convolutional contrastive learning methods. In table 3, all contrastive methods are comparable or slightly worse than supervised pre-training. While SparK, the first *generative* pre-training method for hierarchical

Table 3: **ResNet-50 results on downstream tasks.** We thoroughly compare SparK with contrastive learning algorithms. All methods pre-train with 224 resolution. For ImageNet, the same training recipe from Wightman et al. (2021) with 224 fine-tuning is used. For COCO, Mask-RCNN ResNet50-FPN is equally fine-tuned on `train2017` for 12 or 24 epochs (1× or 2×), with average precision of detection box and segmentation mask on `val2017` reported. SparK is highlighted as the only *generative* method evaluating Mask-RCNN R50-FPN.

| Pre-train (ResNet-50) | Pre-train task | Eff. epoch | Cls. (Acc.) | Det. (Box) 1× | Det. (Box) 2× | Seg. (Mask) 1× | Seg. (Mask) 2× |
|---|---|---|---|---|---|---|---|
| Supervised | – | – | 79.8 | 38.9 | 41.3 | 35.4 | 37.3 |
| SimSiam (Chen & He, 2021) | Contrastive | 800 | 79.1 | – | – | – | – |
| MoCo (He et al., 2020) | Contrastive | 800 | – | 38.5 | 40.8 | 35.1 | 36.9 |
| MoCov2 (Chen et al., 2020b) | Contrastive | 1600 | 79.8 | 40.4 | 41.7 | 36.4 | 37.6 |
| SimCLR (Chen et al., 2020a) | Contrastive | 4000 | 80.0 | – | – | – | – |
| InfoMin (Tian et al., 2020) | Contrastive | 800 | – | 40.6 | 42.5 | 36.7 | 38.4 |
| BYOL (Grill et al., 2020) | Contrastive | 1600 | 80.0 | 40.4 | 42.3 | 37.2 | 38.3 |
| SwAV (Caron et al., 2020) | Contrastive | 1200 | 80.1 | – | 42.3 | – | 38.2 |
| Spark (ours) | Generative | 1600 | **80.6** | **41.6** | **43.4** | **37.7** | **39.4** |

convnets, performs significantly better than them across all downstream tasks by $+0.5$~$1.2$ points. In particular, SparK does not rely on sophisticated augmentations which have proven to be essential for contrastive learning (Chen et al., 2020a; Tian et al., 2020). We attribute these superior results above to the fact that generative pre-training (SparK) inherently provides more supervisory signals than discriminative methods. It optimizes a reconstruction loss, a form of regression loss, which is considered to be more dense and localized than contrastive loss.

**Feature transferability.** Another intriguing phenomenon in table 3 is that the improvements from SparK on COCO are more significant than those on ImageNet. Notice that all methods pre-train on ImageNet with 224 resolution before transferring to COCO, and there are several key differences between the two datasets: (i) image resolution of COCO is much higher; (ii) most images in ImageNet are object-centric, while COCO images usually contain multiple disorganized objects. This *domain gap* poses a challenge for transfer learning, and the *performance gap* between SparK and contrastive learning suggests that models can learn more transferable features through SparK. Taken all results above together, a promising future of generative pre-training on convnets has been initially shown.

### 4.4 Scaling-up SparK

In this experiment, we gradually scale-up the model size and test SparK's fine-tuning performance. Results are reported in table 4, where we quote the accuracy from Wightman et al. (2021) (the strongest ResNet baselines) and Liu et al. (2022) (a modern convolutional architecture ConvNeXt) as "Baseline Acc.". We refer to the hyperparameters in Fang et al. (2022) and Liu et al. (2022) for fine-tuning. Comparing the results vertically in table 4, one can observe that with SparK pre-training, *four* models with smaller sizes (ResNet-101, ResNet-152, ConvNeXt-S, ConvNeXt-B) achieve performance on par with their non-pretrained versions of larger sizes. Such a qualitative leap indicates that SparK can push the convnet to the "next level" in terms of representation capability. Comparing horizontally, SparK greatly improves all supervised baselines by +0.8~1.5, verifying the superiority of self-supervised learned representation. Overall, the results demonstrate a favorable scaling ability of SparK as larger models benefit more. These steady gains across classical and modern architectures make us believe SparK can boost many other state-of-the-art convnets, like VAN (Guo et al., 2022).

Table 4: **Scaling up SparK with the model size.** Top-1 validation accuracy on ImageNet is reported. Absolute improvements over baselines are listed as $\Delta$. The last column highlights if a pre-trained model with a smaller size (*e.g.*, 84.1 of ConvNeXt-S) reaches the baseline performance of a larger size (*e.g.*, 83.8 of ConvNeXt-B).

| Architecture | #Para. (M) | FLOPs (G) | Baseline Acc. (%) | Spark (ours) Acc. (%) | $\Delta$ (%) | Reach the next level |
|---|---|---|---|---|---|---|
| *Basic Architecture* | | | | | | |
| ResNet-50 | 25.6 | 4.1 | 79.8 | 80.6 | +0.8 | |
| ResNet-101 | 44.5 | 7.9 | 81.3 | 82.2 | +0.9 | ✓ |
| ResNet-152 | 60.2 | 11.6 | 81.8 | 82.7 | +0.9 | ✓ |
| ResNet-200 | 64.7 | 15.1 | 82.1 | 83.1 | +1.0 | |
| *Modern Architecture* | | | | | | |
| ConvNext-Small | 50.0 | 8.7 | 83.1 | 84.1 | +1.0 | ✓ |
| ConvNext-Base | 89.0 | 15.4 | 83.8 | 84.8 | +1.0 | ✓ |
| ConvNext-Large | 198 | 34.4 | 84.3 | 85.4 | +1.1 | |

### 4.5 Ablation Study

In this study, we gradually ablate the components in SparK framework and evaluate the corresponding performance respectively. ImageNet fine-tuning results of each SparK's variants are listed in table 5.

**Core designs.** We first remove the two most important designs in SparK: sparse masking strategy and the hierarchical architecture. By replacing our sparse strategy with the zero-outing discussed in section 3.1, we observe a noticeable performance degradation in row 3 of table 5 that almost reaches the supervised baseline. This suggests that the issues raised by zero-outing (data distribution shift in figure 1 and mask pattern vanish in figure 3) can lead to ineffective pre-training. We then remove the hierarchical design (row 4), which results in a *single-scale* masked modeling that is commonly used for transformers (Devlin et al., 2018; Bao et al., 2021; He et al., 2021; Xie et al., 2021). It only uses

Table 5: **Ablation study on the importance of each components in SparK**. Experiments are based on ConvNeXt-Small, with ImageNet validation accuracy reported. Our default setting is in row 2. Differences are highlighted in blue. "APE" denotes whether to use absolute positional embedding during the pre-training.

| | Method | Masking | Hierarchy | APE | Loss | Epoch | Acc. | Δ |
|---|---|---|---|---|---|---|---|---|
| 1 | Supervised | | | | | | 83.1 | -1.0 |
| 2 | SparK (ours) | sparse | ✓ | ✗ | masked only | 1600 | 84.1 | 0.0 |
| 3 | zero-outing | zero-outing | ✓ | ✗ | masked only | 1600 | 83.2 | -0.9 |
| 4 | w/o hierarchy | sparse | ✗ | ✗ | masked only | 1600 | 83.6 | -0.5 |
| 5 | w/ APE | sparse | ✓ | ✓ | masked only | 1600 | 83.9 | -0.2 |
| 6 | w/ more loss | sparse | ✓ | ✗ | all | 1600 | 83.3 | -0.8 |
| 7 | pre-train less | sparse | ✓ | ✗ | masked only | 800 | 83.7 | -0.4 |

the features at the end of encoder to reconstruct. This modification is shown to impair the fine-tuning performance as well. In sum, both sparse strategy and hierarchy play key roles in SparK pre-training.

**Other components.** In addition, we find adding absolute positional embeddings (row 5) is practically useless for learning convolutional representations. We also observe calculating loss values only on masked patches gives higher accuracy (row 6), which is consistent with He et al. (2021). Finally and reasonably, our SparK benefits from longer pre-training as verified in row 7.

### 4.6 Visualization

We visualize some reconstruction results to check how the model performs in pre-training. From figure 4 we can see that the model is able to make different but plausible predictions on masked regions (*e.g.*, in the 2-nd column). In the 4-th and 6-th columns, the model can almost recover the original appearance of red fruits from the very small portion of the exposed edges. The clear texture in the 3-rd column also shows the model can capture the visual signals with medium or high frequencies.

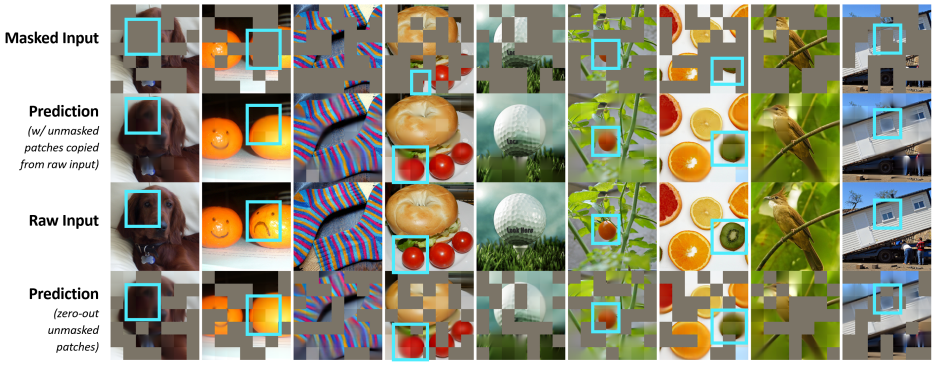


Figure 4: **Reconstruction examples by a pre-trained ConvNeXt-Base with a mask ratio of** 60%**.** Images are randomly selected from ImageNet validation set. Several interesting regions are highlighted.

## 5 Conclusion

The NLP community has witnessed the rise and proliferation of masked modeling on transformers, followed by recent efforts to validate this paradigm on vision transformers. However, generalizing this method to convolutional networks (convnets) is a wonderful, but unrealized vision. Observing the sparse nature of point clouds coincides with those masked patches, we propose to sparsely gather unmasked patches into a sparse image, and use sparse convolution to encode them. We also employ a hierarchical decoder to make full use of the multi-scale encoded features. The resulting framework, called SparK, makes masked modeling well suited for convnets, and shows superior downstream performance compared with state-of-the-art contrastive learning and transformer-based pre-training. Through these results, a promising future of generative pre-training on convnets has been initially shown. We hope our exploration and discovery will inspire more future work to exploit the potential of convnets (maybe by generative pre-training), and prompt us to refocus attention back to them.

## REFERENCES


Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pp. 404–417. Springer, 2006.

Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6154–6162, 2018.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning. *arXiv preprint arXiv:2202.03026*, 2022.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15750–15758, 2021.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9640–9649, 2021.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pp. 1–2. Prague, 2004.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pp. 886–893. Ieee, 2005.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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. *arXiv preprint arXiv:2010.11929*, 2020.

Yuxin Fang, Li Dong, Hangbo Bao, Xinggang Wang, and Furu Wei. Corrupted image modeling for self-supervised visual pre-training. *arXiv preprint arXiv:2202.03382*, 2022.

Clément Farabet, Berin Martini, Polina Akselrod, Selçuk Talay, Yann LeCun, and Eugenio Culurciello. Hardware accelerated convolutional neural networks for synthetic vision systems. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 257–260. IEEE, 2010.

Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE conference on computer vision and pattern recognition*, pp. 1–8. Ieee, 2008.

Peng Gao, Teli Ma, Hongsheng Li, Jifeng Dai, and Yu Qiao. Convmae: Masked convolution meets masked autoencoders. *arXiv preprint arXiv:2205.03892*, 2022.

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

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33: 21271–21284, 2020.

Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *arXiv preprint arXiv:2202.09741*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9): 1904–1916, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3304–3311. IEEE, 2010.

Xiang Li, Wenhai Wang, Lingfeng Yang, and Jian Yang. Uniform masking: Enabling mae pre-training for pyramid-based vision transformers with locality. *arXiv preprint arXiv:2205.10063*, 2022.

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.

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

Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 806–814, 2015.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3000–3009, 2017.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

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.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pp. 1150–1157. Ieee, 1999.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pp. 2564–2571. Ieee, 2011.

Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7276–7282. IEEE, 2019.

Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *Advances in Neural Information Processing Systems*, 33:6827–6839, 2020.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv e-prints*, pp. arXiv–1807, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14668–14678, 2022.

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

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. *arXiv preprint arXiv:2111.09886*, 2021.

Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *2009 IEEE Conference on computer vision and pattern recognition*, pp. 1794–1801. IEEE, 2009.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pp. 12310–12320. PMLR, 2021.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021.

Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4490–4499, 2018.

## A DETAILS ON DECODER ARCHITECTURE

The same lightweight decoder of `LightDecoder(768, 32)` is used equally for all pre-trained convnets, including all ResNets and ConvNeXts. Its pytorch implementation is as follows.

```
import math
import torch.nn as nn


class UNetBlock2x(nn.Module):
   def __init__(self, cin, cout):
      super().__init__()
      self.b = nn.Sequential(
         nn.Conv2d(cin, cin, kernel_size=3, stride=1, padding=1,
             bias=False),
         nn.BatchNorm2d(cin), nn.ReLU6(inplace=True),
         nn.Conv2d(cin, cout, kernel_size=3, stride=1, padding=1,
             bias=False),
         nn.BatchNorm2d(cout),
      )

   def forward(self, x):
      return self.b(x)


class DecoderConv(nn.Module):
   def __init__(self, cin, cout):
      super().__init__()
      self.up = nn.ConvTranspose2d(cin, cin, kernel_size=4, stride=2,
          padding=1, bias=True)

      self.conv = UNetBlock2x(cin, cout)

   def forward(self, x):
      x = self.up(x)
      return self.conv(x)


class LightDecoder(nn.Module):
   def __init__(self, decoder_fea_dim, upsample_ratio):
      super().__init__()
      self.fea_dim = decoder_fea_dim

      n = round(math.log2(upsample_ratio))
      channels = [self.fea_dim // 2 ** i for i in range(n + 1)]
      self.dec = nn.ModuleList([
         DecoderConv(cin, cout) for (cin, cout) in zip(channels[:-1],
             channels[1:])
      ])
      self.proj = nn.Conv2d(channels[-1], 3, kernel_size=1, stride=1,
          bias=True)

   def forward(self, to_dec):
      x = 0
      for i, d in enumerate(self.dec):
         if i < len(to_dec) and to_dec[i] is not None:
            x = x + to_dec[i]
         x = self.dec[i](x)
      return self.proj(x)
```