

2021 (december)

★ Masked Autoencoders - follows Bert approach; 75% of data masked.

Masked Autoencoders Are Scalable Vision Learners

Kaiming He^{*,†} Xinlei Chen* Saining Xie Yanghao Li Piotr Dollár Ross Girshick

*equal technical contribution

†project lead

Bert approach

Facebook AI Research (FAIR)

Abstract

This paper shows that masked autoencoders (MAE) are scalable self-supervised learners for computer vision. Our MAE approach is simple: we mask random patches of the input image and reconstruct the missing pixels. It is based on two core designs. First, we develop an asymmetric encoder-decoder architecture, with an encoder that operates only on the visible subset of patches (without mask tokens), along with a lightweight decoder that reconstructs the original image from the latent representation and mask tokens. Second, we find that masking a high proportion of the input image, e.g., 75%, yields a nontrivial and meaningful self-supervision task. Coupling these two designs enables us to train large models efficiently and effectively: we accelerate training (by 3x or more) and improve accuracy. Our scalable approach allows for learning high-capacity models that generalize well: e.g., a vanilla ViT-Huge model achieves the best accuracy (67.8%) among methods that use only ImageNet-1K data. Transfer performance in downstream tasks outperforms supervised pre-training and shows promising scaling behavior.

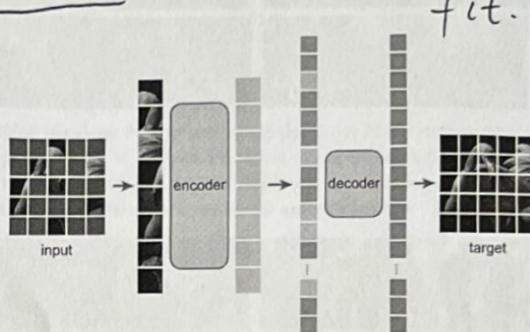


Figure 1. Our MAE architecture. During pre-training, a large random subset of image patches (e.g., 75%) is masked out. The encoder is applied to the small subset of visible patches. Mask tokens are introduced after the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

Some is bert

in vision [59, 46] preceded BERT. However, despite significant interest in this idea following the success of BERT, progress of autoencoding methods in vision lags behind NLP. We ask: what makes masked autoencoding different between vision and language? We attempt to answer this question from the following perspectives:

(i) Until recently, architectures were different. In vision, convolutional networks [34] were dominant over the last decade [33]. Convolutions typically operate on regular grids and it is not straightforward to integrate ‘indicators’ such as mask tokens [14] or positional embeddings [57] into convolutional networks. This architectural gap, however, has been addressed with the introduction of Vision Transformers (ViT) [16] and should no longer present an obstacle.

(ii) Information density is different between language and vision. Languages are human-generated signals that are highly semantic and information-dense. When training a model to predict only a few missing words per sentence, this task appears to induce sophisticated language understanding. Images, on the contrary, are natural signals with heavy spatial redundancy—e.g., a missing patch can be recovered from neighboring patches with little high-level un-

1. Introduction

Deep learning has witnessed an explosion of architectures of continuously growing capability and capacity [33, 25, 57]. Aided by the rapid gains in hardware, models today can easily overfit one million images [13] and begin to demand hundreds of millions of—often publicly inaccessible—labeled images [16].

This appetite for data has been successfully addressed in natural language processing (NLP) by self-supervised pre-training. The solutions, based on autoregressive language modeling in GPT [47, 48, 4] and masked autoencoding in BERT [14], are conceptually simple: they remove a portion of the data and learn to predict the removed content. These methods now enable training of generalizable NLP models containing over one hundred billion parameters [4].

The idea of masked autoencoders, a form of more general denoising autoencoders [58], is natural and applicable in computer vision as well. Indeed, closely related research

Asymmetric encoder - decoder

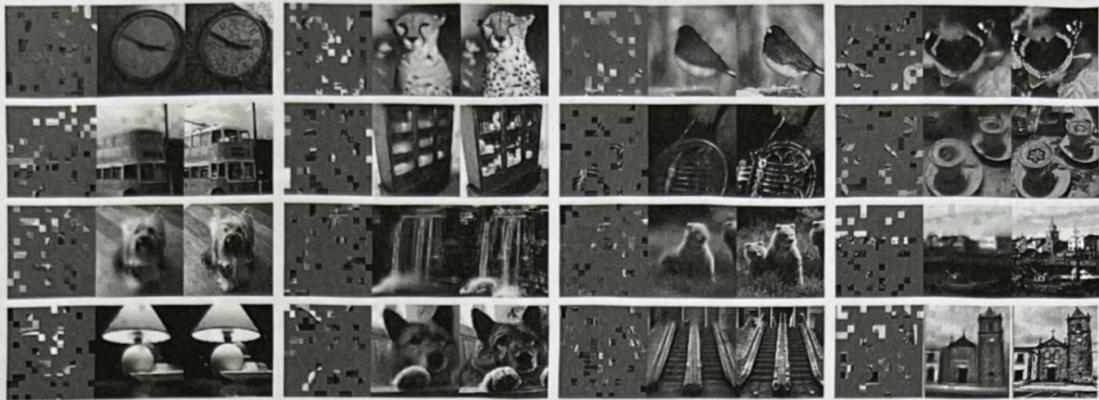


Figure 2. Example results on ImageNet validation images. For each triplet, we show the masked image (left), our MAE reconstruction[†] (middle), and the ground-truth (right). The masking ratio is 80%, leaving only 39 out of 196 patches. More examples are in the appendix.
[†]As no loss is computed on visible patches, the model output on visible patches is qualitatively worse. One can simply overlay the output with the visible patches to improve visual quality. We intentionally opt not to do this, so we can more comprehensively demonstrate the method's behavior.



Figure 3. Example results on COCO validation images, using an MAE trained on ImageNet (the same model weights as in Figure 2). Observe the reconstructions on the two right-most examples, which, although different from the ground truth, are semantically plausible.

derstanding of parts, objects, and scenes. To overcome this difference and encourage learning useful features, we show that a simple strategy works well in computer vision: masking a *very high* portion of random patches. This strategy largely reduces redundancy and creates a challenging self-supervised task that requires holistic understanding beyond low-level image statistics. To get a qualitative sense of our reconstruction task, see Figures 2 – 4.

(iii) The autoencoder's *decoder*, which maps the latent representation back to the input, plays a different role between reconstructing text and images. In vision, the decoder reconstructs *pixels*, hence its output is of a lower semantic level than common recognition tasks. This is in contrast to language, where the decoder predicts missing *words* that contain rich semantic information. While in BERT the decoder can be trivial (an MLP) [14], we found that for images, the decoder design plays a key role in determining the semantic level of the learned latent representations.

Driven by this analysis, we present a simple, effective, and scalable form of a masked autoencoder (MAE) for visual representation learning. Our MAE masks random patches from the input image and reconstructs the missing patches in the pixel space. It has an asymmetric encoder-decoder design. Our encoder operates only on the visible subset of patches (without mask tokens), and our decoder is

lightweight and reconstructs the input from the latent representation along with mask tokens (Figure 1). Shifting the mask tokens to the small decoder in our asymmetric encoder-decoder results in a large reduction in computation. Under this design, a very high masking ratio (e.g., 75%) can achieve a win-win scenario as it optimizes accuracy while allowing the encoder to process only a small portion (e.g., 25%) of patches. This can reduce overall pre-training time by 3× or more and likewise reduce memory consumption, enabling us to easily scale our MAE to large models.

Our MAE learns very high-capacity models that generalize well. With MAE pre-training, we can train data-hungry models like ViT-Large/-Huge [16] on ImageNet-1K with improved generalization performance. With a vanilla ViT-Huge model, we achieve 87.8% accuracy when fine-tuned on ImageNet-1K. This outperforms all previous results that use only ImageNet-1K data. We also evaluate transfer learning on object detection, instance segmentation, and semantic segmentation. In these tasks, our pre-training achieves better results than its supervised pre-training counterparts, and more importantly, we observe significant gains by scaling up models. These observations are aligned with those witnessed in self-supervised pre-training in NLP [14, 47, 48, 4] and we hope that they will enable our field to explore a similar trajectory.

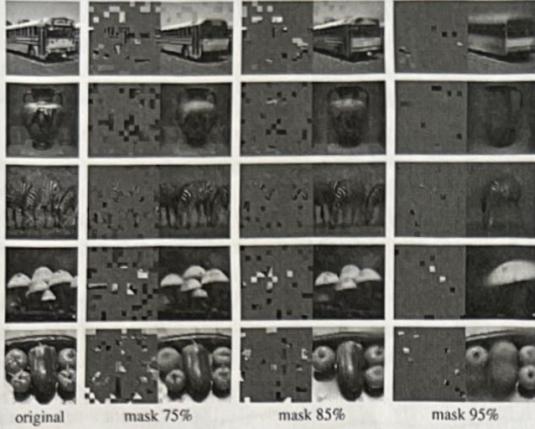


Figure 4. Reconstructions of ImageNet validation images using an MAE pre-trained with a masking ratio of 75% but applied on inputs with higher masking ratios. The predictions differ plausibly from the original images, showing that the method can generalize.

2. Related Work

Masked language modeling and its autoregressive counterparts, *e.g.*, BERT [14] and GPT [47, 48, 4], are highly successful methods for pre-training in NLP. These methods hold out a portion of the input sequence and train models to predict the missing content. These methods have been shown to scale excellently [4] and a large abundance of evidence indicates that these pre-trained representations generalize well to various downstream tasks.

Autoencoding is a classical method for learning representations. It has an encoder that maps an input to a latent representation and a decoder that reconstructs the input. For example, PCA and k-means are autoencoders [29]. Denoising autoencoders (DAE) [58] are a class of autoencoders that corrupt an input signal and learn to reconstruct the original, uncorrupted signal. A series of methods can be thought of as a generalized DAE under different corruptions, *e.g.*, masking pixels [59, 46, 6] or removing color channels [70]. Our MAE is a form of denoising autoencoding, but different from the classical DAE in numerous ways.

Masked image encoding methods learn representations from images corrupted by masking. The pioneering work of [59] presents masking as a noise type in DAE. Context Encoder [46] inpaints large missing regions using convolutional networks. Motivated by the success in NLP, related recent methods [6, 16, 2] are based on **Transformers** [57]. iGPT [6] operates on sequences of pixels and predicts unknown pixels. The ViT paper [16] studies masked patch prediction for self-supervised learning. Most recently, BEiT [2] proposes to predict discrete tokens [44, 50].

Self-supervised learning approaches have seen significant interest in computer vision, often focusing on different pretext tasks for pre-training [15, 61, 42, 70, 45, 17]. Recently, **contrastive learning** [3, 22] has been popular, *e.g.*, [62, 43, 23, 7], which models image similarity and dissimilarity (or only similarity [21, 8]) between two or more views. **Contrastive** and related methods strongly depend on **data augmentation** [7, 21, 8]. Autoencoding pursues a conceptually different direction, and it exhibits different behaviors as we will present.

3. Approach

Our masked autoencoder (MAE) is a simple autoencoding approach that reconstructs the original signal given its partial observation. Like all autoencoders, our approach has an encoder that maps the observed signal to a latent representation, and a decoder that reconstructs the original signal from the latent representation. Unlike classical autoencoders, we adopt an **asymmetric** design that allows the encoder to operate only on the **partial**, observed signal (without mask tokens) and a lightweight decoder that reconstructs the full signal from the latent representation and mask tokens. Figure 1 illustrates the idea, introduced next.

Masking. Following ViT [16], we divide an image into regular non-overlapping patches. Then we sample a subset of patches and **mask** (*i.e.*, remove) the remaining ones. Our sampling strategy is straightforward: we sample random patches without replacement, following a uniform distribution. We simply refer to this as “random sampling”.

Random sampling with a **high** masking ratio (*i.e.*, the ratio of removed patches) largely eliminates redundancy, thus creating a task that cannot be easily solved by extrapolation from visible neighboring patches (see Figures 2 – 4). The uniform distribution prevents a potential center bias (*i.e.*, more masked patches near the image center). Finally, the highly sparse input creates an opportunity for designing an efficient encoder, introduced next.

MAE encoder. Our encoder is a ViT [16] but applied only on **visible, unmasked patches**. Just as in a standard ViT, our encoder embeds patches by a linear projection with added positional embeddings, and then processes the resulting set via a series of Transformer blocks. However, our encoder only operates on a small subset (*e.g.*, 25%) of the full set. Masked patches are removed; no mask tokens are used. This allows us to train very large encoders with only a fraction of compute and memory. The full set is handled by a lightweight decoder, described next.

MAE decoder. The input to the MAE decoder is the full set of tokens consisting of (i) **encoded visible patches**, and (ii) **mask tokens**. See Figure 1. Each mask token [14] is a shared, learned vector that indicates the presence of a miss-

ing patch to be predicted. We add positional embeddings to all tokens in this full set; without this, mask tokens would have no information about their location in the image. The decoder has another series of Transformer blocks.

The MAE decoder is only used during pre-training to perform the image reconstruction task (only the encoder is used to produce image representations for recognition). Therefore, the decoder architecture can be flexibly designed in a manner that is *independent* of the encoder design. We experiment with very small decoders, narrower and shallower than the encoder. For example, our default decoder has <10% computation per token vs. the encoder. With this asymmetrical design, the full set of tokens are only processed by the lightweight decoder, which significantly reduces pre-training time.

Reconstruction target. Our MAE reconstructs the input by predicting the *pixel* values for each masked patch. Each element in the decoder’s output is a vector of pixel values representing a patch. The last layer of the decoder is a linear projection whose number of output channels equals the number of pixel values in a patch. The decoder’s output is reshaped to form a reconstructed image. Our loss function computes the mean squared error (MSE) between the reconstructed and original images in the pixel space. We compute the loss only on masked patches, similar to BERT [14].¹

We also study a variant whose reconstruction target is the normalized pixel values of each masked patch. Specifically, we compute the mean and standard deviation of all pixels in a patch and use them to normalize this patch. Using normalized pixels as the reconstruction target improves representation quality in our experiments.

Simple implementation. Our MAE pre-training can be implemented efficiently, and importantly, does not require any specialized sparse operations. First we generate a token for every input patch (by linear projection with an added positional embedding). Next we randomly shuffle the list of tokens and remove the last portion of the list, based on the masking ratio. This process produces a small subset of tokens for the encoder and is equivalent to sampling patches without replacement. After encoding, we append a list of mask tokens to the list of encoded patches, and unshuffle this full list (inverting the random shuffle operation) to align all tokens with their targets. The decoder is applied to this full list (with positional embeddings added). As noted, no sparse operations are needed. This simple implementation introduces negligible overhead as the shuffling and unshuffling operations are fast.

¹Computing the loss only on masked patches differs from traditional denoising autoencoders [58] that compute the loss on all pixels. This choice is purely result-driven: computing the loss on all pixels leads to a slight decrease in accuracy (*e.g.*, ~0.5%).

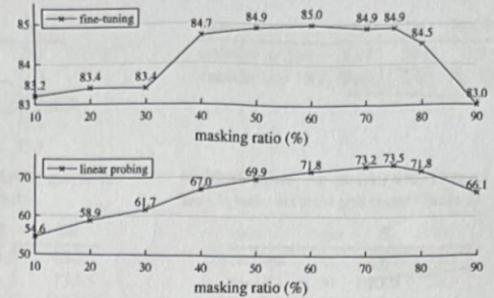


Figure 5. **Masking ratio.** A high masking ratio (75%) works well for both fine-tuning (top) and linear probing (bottom). The y-axes are ImageNet-1K validation accuracy (%) in all plots in this paper.

4. ImageNet Experiments

We do self-supervised pre-training on the ImageNet-1K (IN1K) [13] training set. Then we do supervised training to evaluate the representations with (i) end-to-end fine tuning or (ii) linear probing. We report top-1 validation accuracy of a single 224×224 crop. Details are in Appendix A.1.

Baseline: ViT-Large. We use ViT-Large (ViT-L/16) [16] as the backbone in our ablation study. ViT-L is very big (an order of magnitude bigger than ResNet-50 [25]) and tends to overfit. The following is a comparison between ViT-L trained from scratch vs. fine-tuned from our baseline MAE:

scratch, original [16]	scratch, our impl.	baseline MAE
76.5	82.5	84.9

We note that it is nontrivial to train *supervised* ViT-L from scratch and a good recipe with strong regularization is needed (82.5%, see Appendix A.2). Even so, our MAE pre-training contributes a big improvement. Here fine-tuning is only for 50 epochs (vs. 200 from scratch), implying that the fine-tuning accuracy heavily depends on pre-training.

50 epoch
fine-tune

4.1. Main Properties

We ablate our MAE using the default settings in Table 1 (see caption). Several intriguing properties are observed.

Masking ratio. Figure 5 shows the influence of the masking ratio. The optimal ratios are surprisingly high. The ratio of 75% is good for both linear probing and fine-tuning. This behavior is in contrast with BERT [14], whose typical masking ratio is 15%. Our masking ratios are also much higher than those in related works [6, 16, 2] in computer vision (20% to 50%).

The model infers missing patches to produce different, yet plausible, outputs (Figure 4). It makes sense of the gestalt of objects and scenes, which cannot be simply completed by extending lines or textures. We hypothesize that this reasoning-like behavior is linked to the learning of useful representations.

Figure 5 also shows that linear probing and fine-tuning results follow *different* trends. For linear probing, the ac-

blocks	ft	lin
1	84.8	65.5
2	84.9	70.0
4	84.9	71.9
8	84.9	73.5
12	84.4	73.3

(a) **Decoder depth.** A deep decoder can improve linear probing accuracy.

dim	ft	lin
128	84.9	69.1
256	84.8	71.3
512	84.9	73.5
768	84.4	73.1
1024	84.3	73.1

(b) **Decoder width.** The decoder can be narrower than the encoder (1024-d).

case	ft	lin	FLOPs
encoder w/ [M]	84.2	59.6	3.3×
encoder w/o [M]	84.9	73.5	1×

(c) **Mask token.** An encoder without mask tokens is more accurate and faster (Table 2).

case	ft	lin
pixel (w/o norm)	84.9	73.5
pixel (w/ norm)	85.4	73.9
PCA	84.6	72.3
dVAE token	85.3	71.6

(d) **Reconstruction target.** Pixels as reconstruction targets are effective.

case	ft	lin
none	84.0	65.7
crop, fixed size	84.7	73.1
crop, rand size	84.9	73.5
crop + color jit	84.3	71.9

(e) **Data augmentation.** Our MAE works with minimal or no augmentation.

case	ratio	ft	lin
random	75	84.9	73.5
block	50	83.9	72.3
block	75	82.8	63.9
grid	75	84.0	66.0

(f) **Mask sampling.** Random sampling works the best. See Figure 6 for visualizations.

Table 1. **MAE ablation experiments** with ViT-L/16 on ImageNet-1K. We report fine-tuning (ft) and linear probing (lin) accuracy (%). If not specified, the default is: the decoder has depth 8 and width 512, the reconstruction target is unnormalized pixels, the data augmentation is random resized cropping, the masking ratio is 75%, and the pre-training length is 800 epochs. Default settings are marked in gray.

curacy increases steadily with the masking ratio until the sweet point: the accuracy gap is up to ~20% (54.6% vs. 73.5%). For fine-tuning, the results are less sensitive to the ratios, and a wide range of masking ratios (40–80%) work well. All fine-tuning results in Figure 5 are better than training from scratch (82.5%).

Decoder design. Our MAE decoder can be flexibly designed, as studied in Table 1a and 1b.

Table 1a varies the decoder depth (number of Transformer blocks). A sufficiently deep decoder is important for linear probing. This can be explained by the gap between a pixel reconstruction task and a recognition task: the last several layers in an autoencoder are more specialized for reconstruction, but are less relevant for recognition. A reasonably deep decoder can account for the reconstruction specialization, leaving the latent representations at a more abstract level. This design can yield up to 8% improvement in linear probing (Table 1a, ‘lin’). However, if fine-tuning is used, the last layers of the encoder can be tuned to adapt to the recognition task. The decoder depth is less influential for improving fine-tuning (Table 1a, ‘ft’).

Interestingly, our MAE with a single-block decoder can perform strongly with fine-tuning (84.8%). Note that a single Transformer block is the minimal requirement to propagate information from visible tokens to mask tokens. Such a small decoder can further speed up training.

In Table 1b we study the decoder width (number of channels). We use 512-d by default, which performs well under fine-tuning and linear probing. A narrower decoder also works well with fine-tuning.

Overall, our default MAE decoder is lightweight. It has 8 blocks and a width of 512-d (gray in Table 1). It only has 9% FLOPs per token vs. ViT-L (24 blocks, 1024-d). As such, while the decoder processes all tokens, it is still a small fraction of the overall compute.

encoder	dec. depth	ft acc	hours	speedup
ViT-L, w/ [M]	8	84.2	42.4	-
ViT-L	8	84.9	15.4	2.8×
ViT-L	1	84.8	11.6	3.7×
ViT-H, w/ [M]	8	-	119.6†	-
ViT-H	8	85.8	34.5	3.5×
ViT-H	1	85.9	29.3	4.1×

Table 2. **Wall-clock time** of our MAE training (800 epochs), benchmarked in 128 TPU-v3 cores with TensorFlow. The speedup is relative to the entry whose encoder has mask tokens (gray). The decoder width is 512, and the mask ratio is 75%. †: This entry is estimated by training ten epochs.

Mask token. An important design of our MAE is to skip the mask token [M] in the encoder and apply it later in the lightweight decoder. Table 1c studies this design.

If the encoder uses mask tokens, it performs worse: its accuracy drops by 14% in linear probing. In this case, there is a gap between pre-training and deploying: this encoder has a large portion of mask tokens in its input in pre-training, which does not exist in uncorrupted images. This gap may degrade accuracy in deployment. By removing the mask token from the encoder, we constrain the encoder to always see real patches and thus improve accuracy.

Moreover, by skipping the mask token in the encoder, we greatly reduce training computation. In Table 1c, we reduce the overall training FLOPs by 3.3×. This leads to a 2.8× wall-clock speedup in our implementation (see Table 2). The wall-clock speedup is even bigger (3.5–4.1×), for a smaller decoder (1-block), a larger encoder (ViT-H), or both. Note that the speedup can be >4× for a masking ratio of 75%, partially because the self-attention complexity is quadratic. In addition, memory is greatly reduced, which can enable training even larger models or speeding up more by large-batch training. The time and memory efficiency makes our MAE favorable for training very large models.

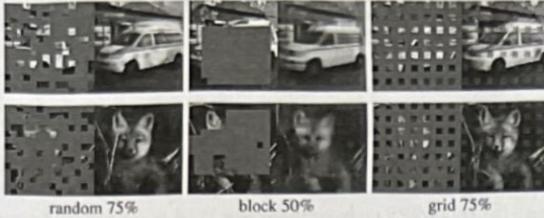


Figure 6. **Mask sampling strategies** determine the pretext task difficulty, influencing reconstruction quality and representations (Table 1f). Here each output is from an MAE trained with the specified masking strategy. Left: random sampling (our default). Middle: block-wise sampling [2] that removes large random blocks. Right: grid-wise sampling that keeps one of every four patches. Images are from the validation set.

Reconstruction target. We compare different reconstruction targets in Table 1d. Our results thus far are based on pixels without (per-patch) normalization. Using pixels with normalization improves accuracy. This per-patch normalization enhances the contrast locally. In another variant, we perform PCA in the patch space and use the largest PCA coefficients (96 here) as the target. Doing so degrades accuracy. Both experiments suggest that the high-frequency components are useful in our method.

We also compare an MAE variant that predicts *tokens*, the target used in BEiT [2]. Specifically for this variant, we use the DALLE pre-trained dVAE [50] as the tokenizer, following [2]. Here the MAE decoder predicts the token indices using cross-entropy loss. This tokenization improves fine-tuning accuracy by 0.4% vs. unnormalized pixels, but has no advantage vs. normalized pixels. It also reduces linear probing accuracy. In §5 we further show that tokenization is not necessary in transfer learning.

Our pixel-based MAE is much simpler than tokenization. The dVAE tokenizer requires one more pre-training stage, which may depend on extra data (250M images [50]). The dVAE encoder is a large convolutional network (40% FLOPs of ViT-L) and adds nontrivial overhead. Using pixels does not suffer from these problems.

Data augmentation. Table 1e studies the influence of data augmentation on our MAE pre-training.

Our MAE works well using cropping-only augmentation, either fixed-size or random-size (both having random horizontal flipping). Adding color jittering degrades the results and so we do not use it in other experiments.

Surprisingly, our MAE behaves decently even if using no data augmentation (only center-crop, no flipping). This property is dramatically different from contrastive learning and related methods [62, 23, 7, 21], which heavily rely on data augmentation. It was observed [21] that using cropping-only augmentation reduces the accuracy by 13%

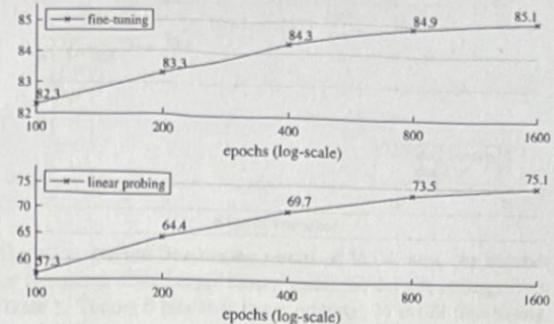


Figure 7. **Training schedules.** A longer training schedule gives a noticeable improvement. Here each point is a full training schedule. The model is ViT-L with the default setting in Table 1.

and 28% respectively for BYOL [21] and SimCLR [7]. In addition, there is no evidence that contrastive learning can work without augmentation: the two views of an image are the same and can easily satisfy a trivial solution.

In MAE, the role of data augmentation is mainly performed by random masking (ablated next). The masks are different for each iteration and so they generate new training samples regardless of data augmentation. The pretext task is made difficult by masking and requires less augmentation to regularize training.

Mask sampling strategy. In Table 1f we compare different mask sampling strategies, illustrated in Figure 6.

The block-wise masking strategy, proposed in [2], tends to remove large blocks (Figure 6 middle). Our MAE with block-wise masking works reasonably well at a ratio of 50%, but degrades at a ratio of 75%. This task is harder than that of random sampling, as a higher training loss is observed. The reconstruction is also blurrier.

We also study grid-wise sampling, which regularly keeps one of every four patches (Figure 6 right). This is an easier task and has lower training loss. The reconstruction is sharper. However, the representation quality is lower.

Simple random sampling works the best for our MAE. It allows for a higher masking ratio, which provides a greater speedup benefit while also enjoying good accuracy.

Training schedule. Our ablations thus far are based on 800-epoch pre-training. Figure 7 shows the influence of the training schedule length. The accuracy improves steadily with longer training. Indeed, we have not observed saturation of linear probing accuracy even at 1600 epochs. This behavior is unlike contrastive learning methods, e.g., MoCo v3 [9] saturates at 300 epochs for ViT-L. Note that the MAE encoder only sees 25% of patches per epoch, while in contrastive learning the encoder sees 200% (two-crop) or even more (multi-crop) patches per epoch.

different
mask sampling
by

all Epochs

method	pre-train data	ViT-B	ViT-L	ViT-H	ViT-H ₄₄₈
scratch, our impl.	-	82.3	82.6	83.1	-
DINO [5]	IN1K	82.8	-	-	-
MoCo v3 [9]	IN1K	83.2	84.1	-	-
BEiT [2]	IN1K+DALLE	83.2	85.2	-	-
MAE	IN1K	<u>83.6</u>	<u>85.9</u>	<u>86.9</u>	<u>87.8</u>

Table 3. Comparisons with previous results on ImageNet-1K. The pre-training data is the ImageNet-1K training set (except the tokenizer in BEiT was pre-trained on 250M DALLE data [50]). All self-supervised methods are evaluated by end-to-end fine-tuning. The ViT models are B/16, L/16, H/14 [16]. The best for each column is underlined. All results are on an image size of 224, except for ViT-H with an extra result on 448. Here our MAE reconstructs normalized pixels and is pre-trained for 1600 epochs.

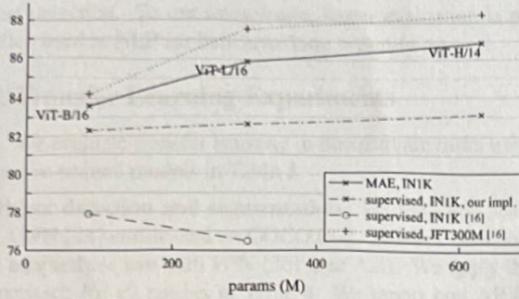


Figure 8. MAE pre-training vs. supervised pre-training, evaluated by fine-tuning in ImageNet-1K (224 size). We compare with the original ViT results [16] trained in IN1K or JFT300M.

4.2. Comparisons with Previous Results

Comparisons with self-supervised methods. In Table 3 we compare the fine-tuning results of self-supervised ViT models. For ViT-B, all methods perform closely. For ViT-L, the gaps among methods are bigger, suggesting that a challenge for bigger models is to reduce overfitting.

Our MAE can scale up easily and has shown steady improvement from bigger models. We obtain 86.9% accuracy using ViT-H (224 size). By fine-tuning with a 448 size, we achieve 87.8% accuracy, using only IN1K data. The previous best accuracy, among all methods using only IN1K data, is 87.1% (512 size) [67], based on advanced networks. We improve over the state-of-the-art by a nontrivial margin in the highly competitive benchmark of IN1K (no external data). Our result is based on *vanilla* ViT, and we expect advanced networks will perform better.

Comparing with BEiT [2], our MAE is *more accurate while being simpler and faster*. Our method reconstructs pixels, in contrast to BEiT that predicts tokens: BEiT reported a 1.8% degradation [2] when reconstructing pixels with ViT-B.² We do not need dVAE pre-training. Moreover, our MAE is considerably faster (3.5× per epoch) than BEiT, for the reason as studied in Table 1c.

²We observed the degradation also in BEiT with ViT-L: it produces 85.2% (tokens) and 83.5% (pixels), reproduced from the official code.

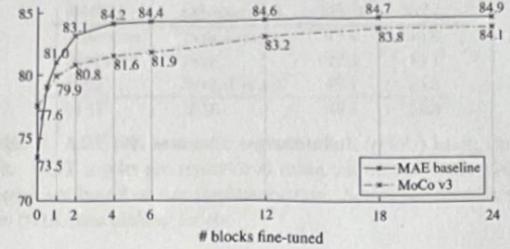


Figure 9. Partial fine-tuning results of ViT-L w.r.t. the number of fine-tuned Transformer blocks under the default settings from Table 1. Tuning 0 blocks is linear probing; 24 is full fine-tuning. Our MAE representations are less linearly separable, but are consistently better than MoCo v3 if one or more blocks are tuned.

The MAE models in Table 3 are pre-trained for 1600 epochs for better accuracy (Figure 7). Even so, our total pre-training time is *less* than the other methods when trained on the same hardware. For example, training ViT-L on 128 TPU-v3 cores, our MAE’s training time is 31 hours for 1600 epochs and MoCo v3’s is 36 hours for 300 epochs [9].

Comparisons with supervised pre-training. In the original ViT paper [16], ViT-L degrades when trained in IN1K. Our implementation of supervised training (see A.2) works better, but accuracy saturates. See Figure 8.

Our MAE pre-training, using only IN1K, can generalize better: the gain over training from scratch is bigger for higher-capacity models. It follows a trend similar to the JFT-300M supervised pre-training in [16]. This comparison shows that our MAE can help scale up model sizes.

4.3. Partial Fine-tuning

Table 1 shows that linear probing and fine-tuning results are largely *uncorrelated*. Linear probing has been a popular protocol in the past few years; however, it misses the opportunity of pursuing *strong but non-linear* features—which is indeed a strength of deep learning. As a middle ground, we study a *partial fine-tuning* protocol: fine-tune the last several layers while freezing the others. This protocol was also used in early works, e.g., [65, 70, 42].

Figure 9 shows the results. Notably, fine-tuning only *one* Transformer block boosts the accuracy significantly from 73.5% to 81.0%. Moreover, if we fine-tune only “half” of the last block (*i.e.*, its MLP sub-block), we can get 79.1%, much better than linear probing. This variant is essentially fine-tuning an MLP head. Fine-tuning a few blocks (*e.g.*, 4 or 6) can achieve accuracy close to full fine-tuning.

In Figure 9 we also compare with MoCo v3 [9], a contrastive method with ViT-L results available. MoCo v3 has higher linear probing accuracy; however, all of its partial fine-tuning results are worse than MAE. The gap is 2.6% when tuning 4 blocks. While the MAE representations are less linearly separable, they are stronger *non-linear* features and perform well when a non-linear head is tuned.

method	pre-train data	AP ^{box}		AP ^{mask}	
		ViT-B	ViT-L	ViT-B	ViT-L
supervised	IN1K w/ labels	47.9	49.3	42.9	43.9
MoCo v3	IN1K	47.9	49.3	42.7	44.0
BEiT	IN1K+DALLE	49.8	53.3	44.4	47.1
MAE	IN1K	50.3	53.3	44.9	47.2

Table 4. COCO object detection and segmentation using a ViT Mask R-CNN baseline. All entries are based on our implementation. Self-supervised entries use IN1K data *without* labels. Mask AP follows a similar trend as box AP.

These observations suggest that linear separability is not the sole metric for evaluating representation quality. It has also been observed (*e.g.*, [8]) that linear probing is not well correlated with transfer learning performance, *e.g.*, for object detection. To our knowledge, linear evaluation is not often used in NLP for benchmarking pre-training.

5. Transfer Learning Experiments

We evaluate transfer learning in downstream tasks using the pre-trained models in Table 3.

Object detection and segmentation. We fine-tune Mask R-CNN [24] end-to-end on COCO [37]. The ViT backbone is adapted for use with FPN [36] (see A.3). We apply this approach for all entries in Table 4. We report box AP for object detection and mask AP for instance segmentation.

Compared to supervised pre-training, our MAE performs better under all configurations (Table 4). With the smaller ViT-B, our MAE is 2.4 points higher than *supervised* pre-training (50.3 vs. 47.9, AP^{box}). More significantly, with the larger ViT-L, our MAE pre-training outperforms supervised pre-training by 4.0 points (53.3 vs. 49.3).

The *pixel-based* MAE is better than or on par with the *token-based* BEiT, while MAE is much simpler and faster. Both MAE and BEiT are better than MoCo v3 and MoCo v3 is on par with supervised pre-training.

Semantic segmentation. We experiment on ADE20K [72] using UperNet [63] (see A.4). Table 5 shows that our pre-training significantly improves results over *supervised* pre-training, *e.g.*, by 3.7 points for ViT-L. Our pixel-based MAE also outperforms the *token-based* BEiT. These observations are consistent with those in COCO.

Classification tasks. Table 6 studies transfer learning on the iNaturalists [56] and Places [71] tasks (see A.5). On iNat, our method shows strong scaling behavior: accuracy improves considerably with *bigger models*. Our results surpass the previous best results *by large margins*. On Places, our MAE outperforms the previous best results [19, 40], which were obtained via pre-training on billions of images.

Pixels vs. tokens. Table 7 compares pixels vs. tokens as the MAE reconstruction target. While using dVAE tokens is better than using *unnormalized pixels*, it is statistically similar to using *normalized pixels* across all cases we tested. It again shows that tokenization is not necessary for our MAE.

method	pre-train data	ViT-B	ViT-L
supervised	IN1K w/ labels	47.4	49.9
MoCo v3	IN1K	47.3	49.1
BEiT	IN1K+DALLE	47.1	53.3
MAE	IN1K	48.1	53.6

Table 5. ADE20K semantic segmentation (mIoU) using UperNet. BEiT results are reproduced using the official code. Other entries are based on our implementation. Self-supervised entries use IN1K data *without* labels.

dataset	ViT-B	ViT-L	ViT-H	ViT-H ₄₄₈	prev best
iNat 2017	70.5	75.7	79.3	83.4	75.4 [55]
iNat 2018	75.4	80.1	83.0	86.8	81.2 [54]
iNat 2019	80.5	83.4	85.7	88.3	84.1 [54]
Places205	63.9	65.8	65.9	66.8	66.0 [19] [†]
Places365	57.9	59.4	59.8	60.3	58.0 [40] [‡]

Table 6. Transfer learning accuracy on classification datasets, using MAE pre-trained on IN1K and then fine-tuned. We provide system-level comparisons with the previous best results.

[†]: pre-trained on 1 billion images. [‡]: pre-trained on 3.5 billion images.

	IN1K			COCO		ADE20K	
	ViT-B	ViT-L	ViT-H	ViT-B	ViT-L	ViT-B	ViT-L
pixel (w/o norm)	83.3	85.1	86.2	49.5	52.8	48.0	51.8
pixel (w/ norm)	83.6	85.9	86.9	50.3	53.3	48.1	53.6
dVAE token	83.6	85.7	86.9	50.3	53.2	48.1	53.4
△	0.0	-0.2	0.0	0.0	-0.1	0.0	-0.2

Table 7. Pixels vs. tokens as the MAE reconstruction target. △ is the difference between using dVAE tokens and using normalized pixels. The difference is statistically insignificant.

6. Discussion and Conclusion

Simple algorithms that scale well are the core of deep learning. In NLP, simple self-supervised learning methods (*e.g.*, [47, 14, 48, 4]) enable benefits from exponentially scaling models. In computer vision, practical pre-training paradigms are dominantly supervised (*e.g.* [33, 51, 25, 16]) despite progress in self-supervised learning. In this study, we observe on ImageNet and in transfer learning that an autoencoder—a simple self-supervised method similar to techniques in NLP—provides scalable benefits. Self-supervised learning in vision may now be embarking on a similar trajectory as in NLP.

On the other hand, we note that images and languages are *signals of a different nature* and this difference must be addressed carefully. Images are merely recorded light without a semantic decomposition into the visual analogue of words. Instead of attempting to remove objects, we remove random patches that most likely do *not* form a semantic segment. Likewise, our MAE reconstructs pixels, which are *not* semantic entities. Nevertheless, we observe (*e.g.*, Figure 4) that our MAE infers complex, holistic reconstructions, suggesting it has learned numerous visual concepts, *i.e.*, semantics. We hypothesize that this behavior occurs by way of a rich hidden representation inside the MAE. We hope this perspective will inspire future work.