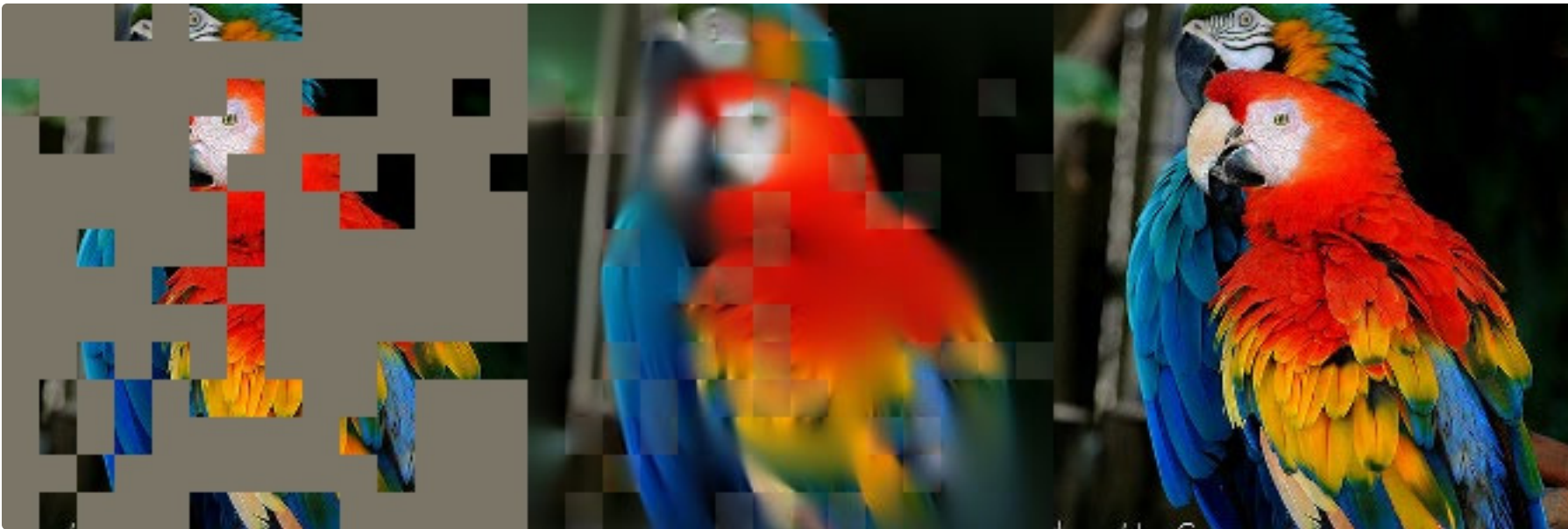


Masked Autoencoders Are Scalable Vision Learners

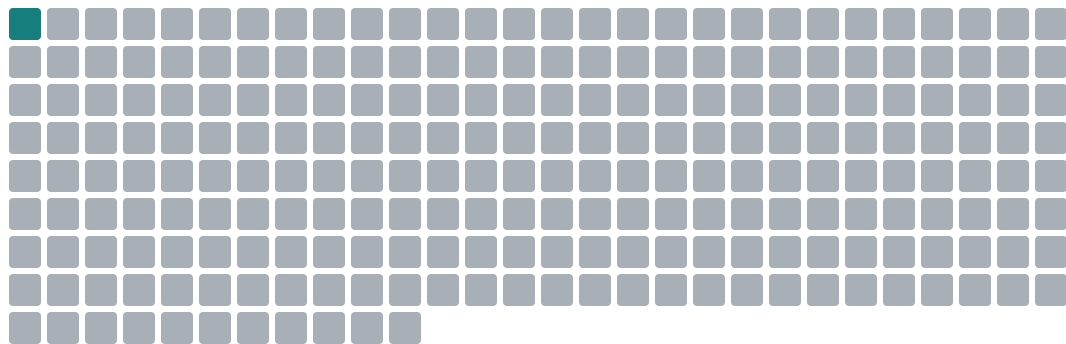
Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, Ross Girshick
Facebook AI Research, CVPR 2022



An ImageNet validation image with 75 percent of the patches removed, the MAE reconstruction, and the original. Figure 10 in the paper.

The problem, and an idea from language

Labels are the bottleneck



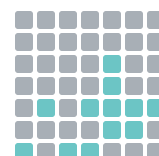
Each square is 1.28 million labeled images. Teal: ImageNet-1K, public. Gray: JFT-300M, Google-internal, used to pretrain ViT.

ViT-Large is big enough to memorize ImageNet. Trained from scratch on it, it overfits and lands below a ResNet. It only works well after pretraining on JFT-300M, which nobody else has. The question is whether ImageNet alone can be enough, without any extra labels.

Hide part of the data, predict it

The quick brown [] jumps over the [] dog.

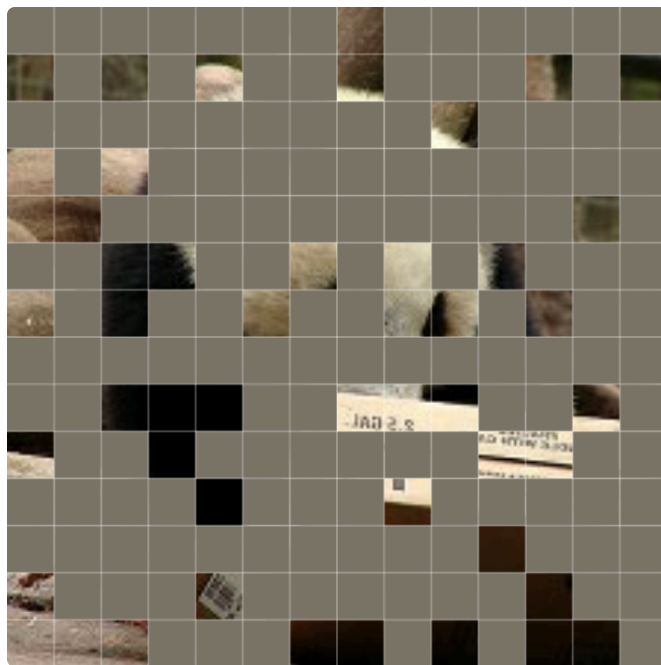
BERT: 15% of the words hidden, model predicts them. The target comes from the data i



The same idea for images: hide patches, predict them. People tried this before BERT. It never worked as well as in language. The paper asks why, and fixes it.

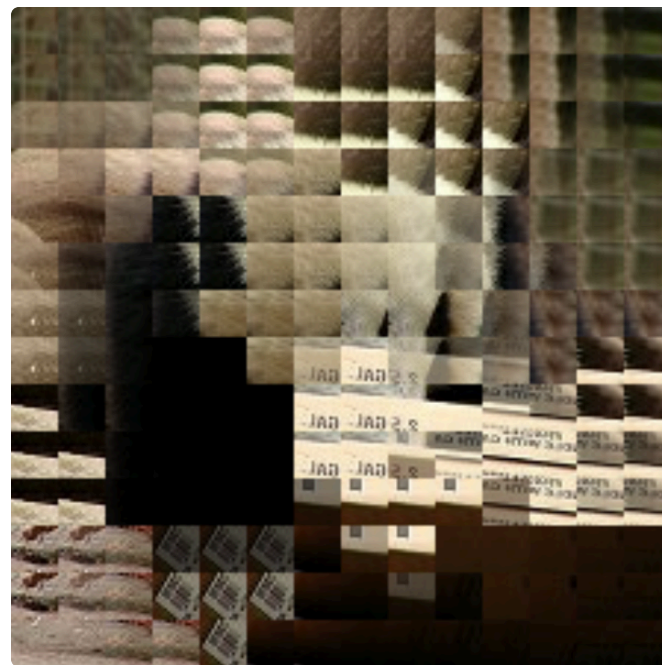
No one writes a label for a hidden word. Predicting it turns out to require understanding the sentence, and that understanding transfers to every downstream task. GPT and BERT scale to hundreds of billions of parameters this way.

Why hide 75 percent of the image?



what the network is given

Masking ratio 75 percent, patches chosen at random.



what averaging the visible neighbors recovers, no learning involved

Images are redundant. At BERT's 15 percent every hidden patch has visible neighbors, and simply averaging them already gives the picture back. A network trained on that task learns to average.

At 75 percent averaging gives a smear. To fill these gaps you have to know what a train looks like. That is the paper's argument for masking most of the image, and its explanation for why earlier work with 20 to 50 percent masking got weak features.

masking ratio

75 %

encoder cost vs. full image

24 % of FLOPs

patches the encoder sees

49 of 196

masked patches with no visible neighbor

47 of 147

Architecture

The image is split into a 14×14 grid, 196 patches. 147 of them are removed at random. The encoder, a normal ViT, gets only the 49 that remain.

The decoder gets the encoder output plus a mask token at each of the 147 missing positions, and predicts the missing pixels. The pixel error is the only training signal.

After pretraining the decoder is dropped. The encoder is used on full images for recognition.

Two choices make this work: the encoder never sees a mask token, which keeps it cheap, and the masking ratio is high, which keeps the task hard.

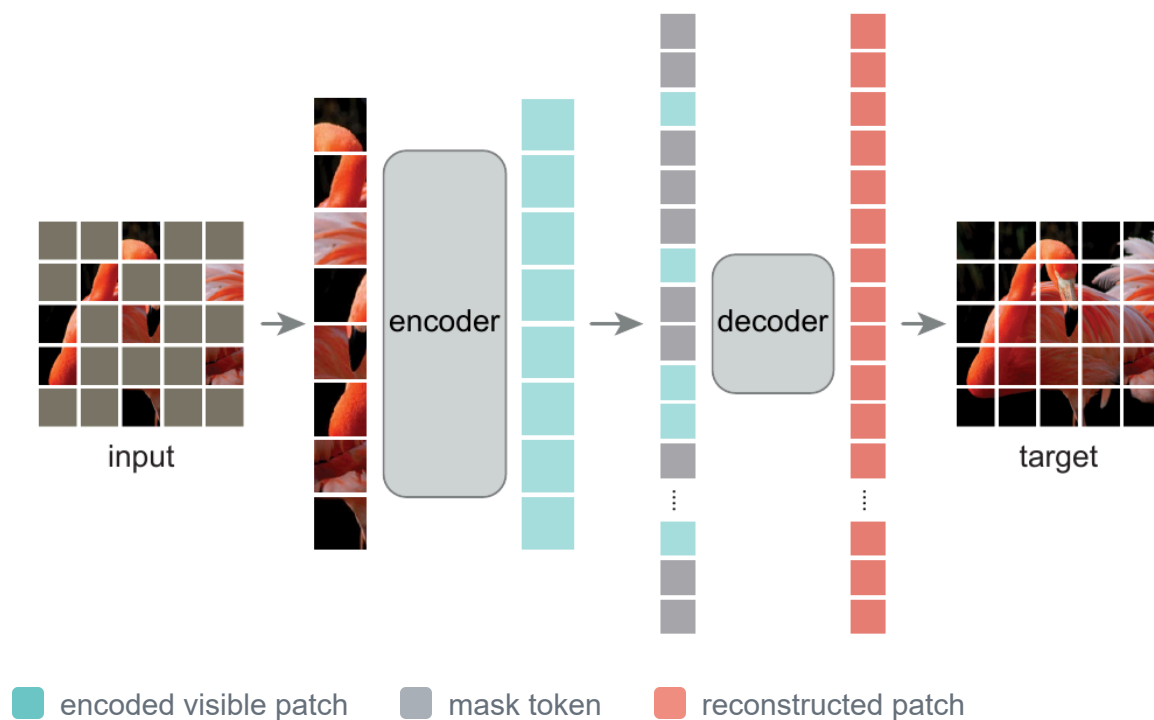


Figure 1 in the paper. These three colors are used the same way on every slide.

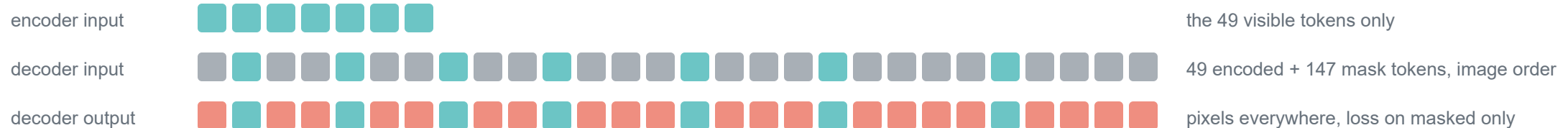
Encoder and decoder

Encoder: the large ViT, 24 blocks, width 1024

Only the 49 visible patches go in, each with its positional embedding. Attention compares every token with every other, so its cost grows with the square of the sequence length. A quarter of the tokens costs much less than a quarter of the compute.

Decoder: 8 blocks, width 512, under 10% of the encoder's compute per token

It turns the encoder output into pixels. If it were large it could do the understanding itself and the encoder could stay lazy. Kept small, it pushes the work into the encoder, which is the part we keep.



Implementation



the image as 4×4 patches, for illustration

0 Each patch becomes a token: a linear projection plus a positional embedding. 16 patches shown instead of 196.

Implementation



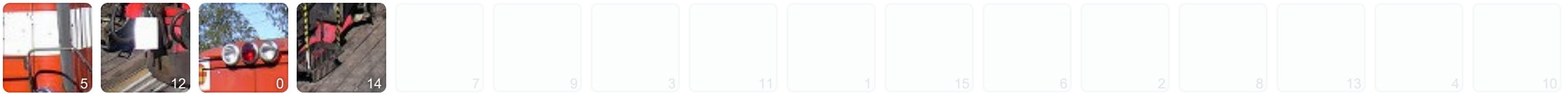
1 The tokens form a list, in natural order.

Implementation



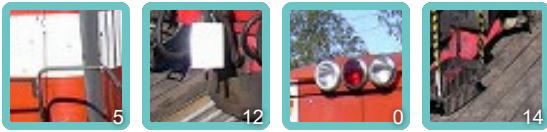
2 Shuffle the list randomly and remember the permutation.

Implementation



3 Drop the last 75 percent. The first quarter of a random permutation is a uniform sample without replacement.

Implementation



encoder, a standard ViT

4 Encode what is left. Four times fewer tokens, roughly sixteen times less attention compute.

Implementation



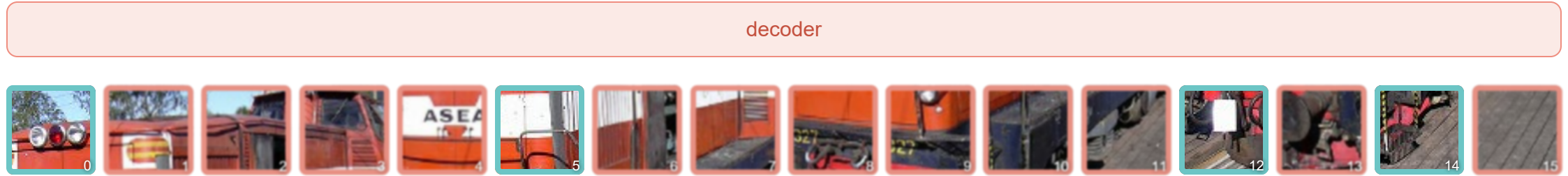
- 5 Append mask tokens, all copies of one learned vector, to get the full length back.

Implementation



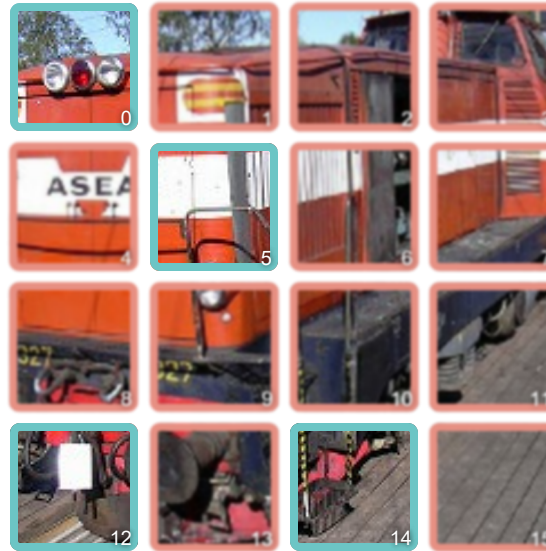
- 6 Unshuffle with the inverse permutation. Every token is back at its true position. Add positional embeddings.

Implementation



- 7 The decoder predicts a patch of pixels for every position. The loss is taken only on the masked ones, in salmon.

Implementation



reconstructed image

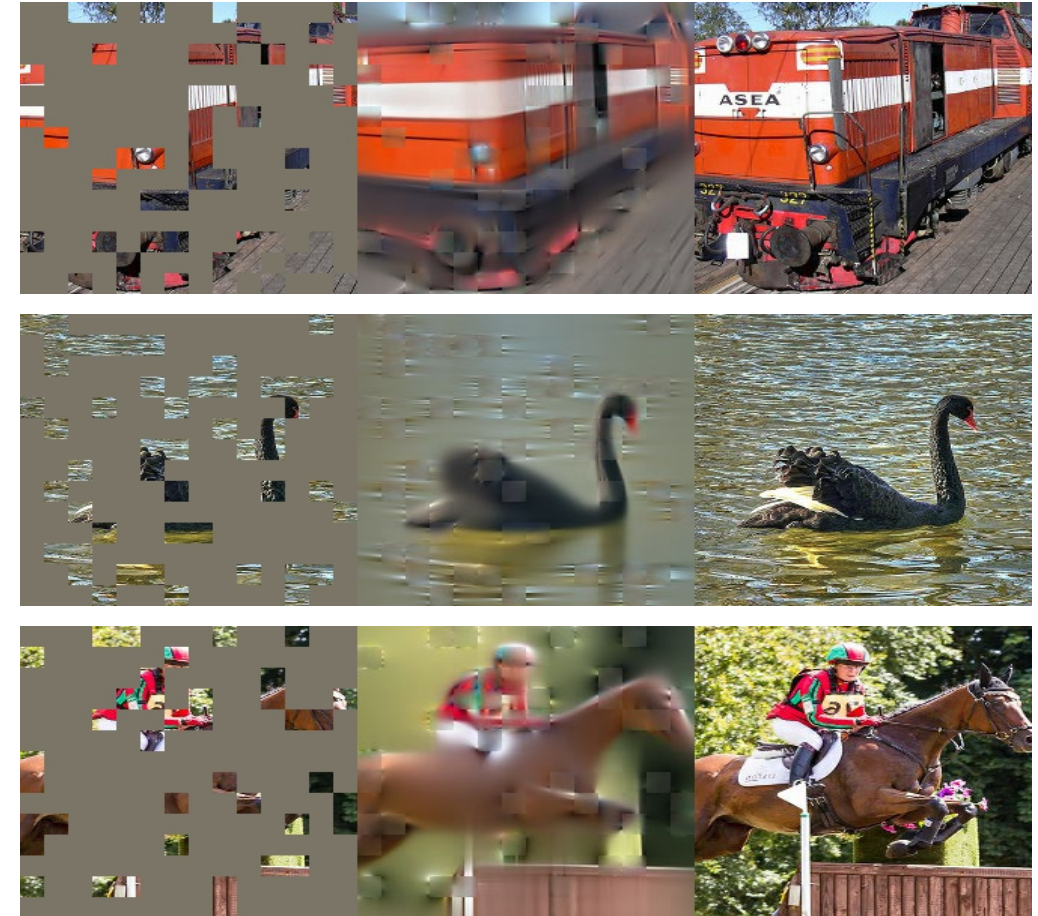
8 Reshape into an image.

Reconstruction target and loss

The decoder's last layer outputs $16 \times 16 \times 3$ values per token, which reshape into an image. The loss is the mean squared error against the original, computed on the masked patches only. Computing it on all patches costs about half a point of accuracy.

Normalizing each target patch by its own mean and standard deviation gives the best results in the paper.

Reconstructions are blurry, since squared error rewards the average, but they are structurally right. They stay plausible on COCO images the model never saw, which is the first sign that the encoder learned something general.



All at 75 percent: two ImageNet validation images (Figure 10) and a COCO image (Figure 3). Visible patches look worse in the output because no loss is applied to them; the authors show the raw output on purpose.

Pretraining, then fine-tuning

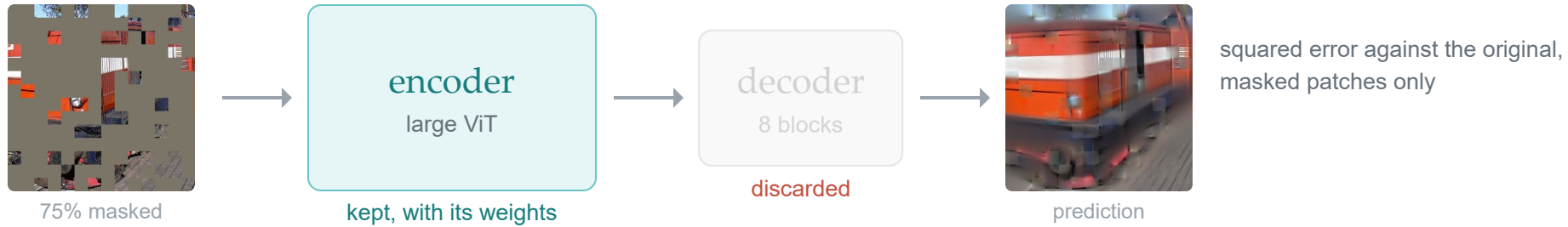
pretraining · 1.28 million ImageNet images · labels not used · 800 to 1600 epochs



- 0 Pretraining. Mask 75 percent, encode the visible patches, predict the missing pixels, take the squared error on the masked patches. Repeat over ImageNet for 800 to 1600 epochs without touching a label.

Pretraining, then fine-tuning

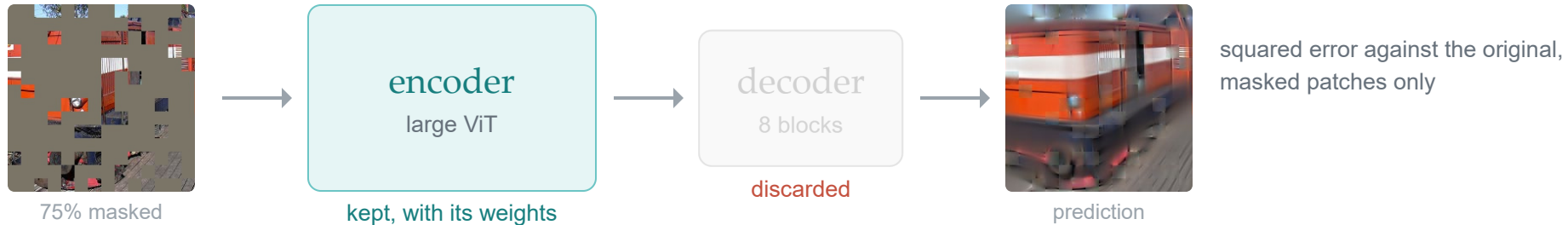
pretraining · 1.28 million ImageNet images · labels not used · 800 to 1600 epochs



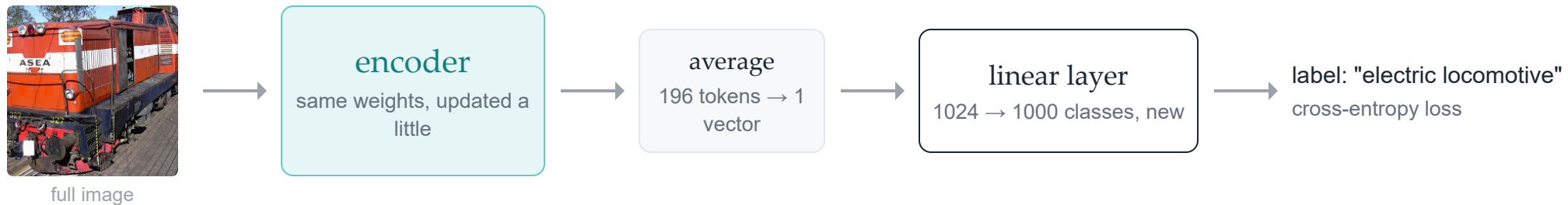
- 1 Afterwards the decoder is thrown away. The encoder is kept, with everything it learned.

Pretraining, then fine-tuning

pretraining · 1.28 million ImageNet images · labels not used · 800 to 1600 epochs



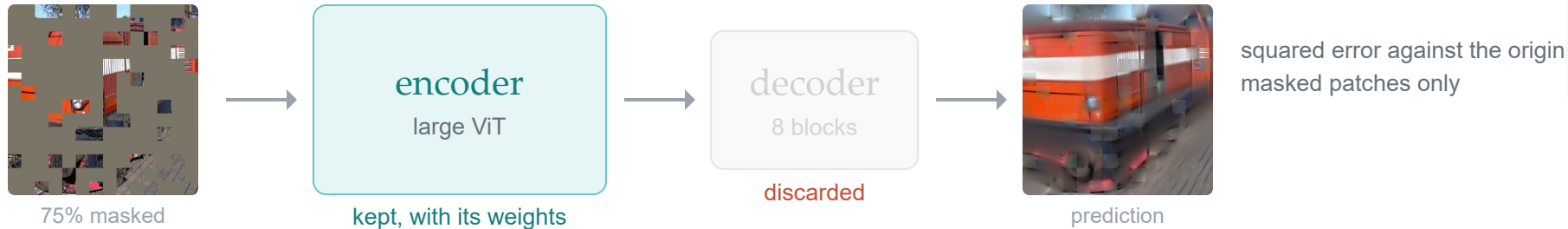
fine-tuning · the same images, now with their labels · 50 epochs



- 2 Fine-tuning. The same images, now with their labels. A full image goes through the encoder, the 196 output tokens are averaged, and a new linear layer maps that vector to the 1000 classes. Everything is trained with the classification loss: the new layer from scratch, the encoder only nudged. 50 epochs are enough.

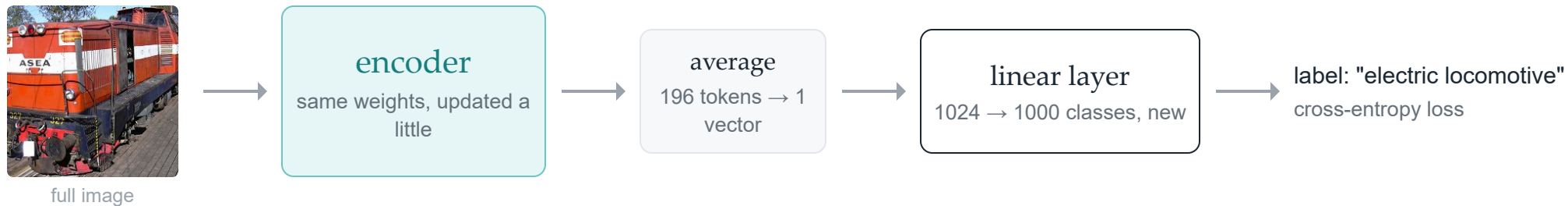
Pretraining, then fine-tuning

pretraining · 1.28 million ImageNet images · labels not used · 800 to 1600 epochs



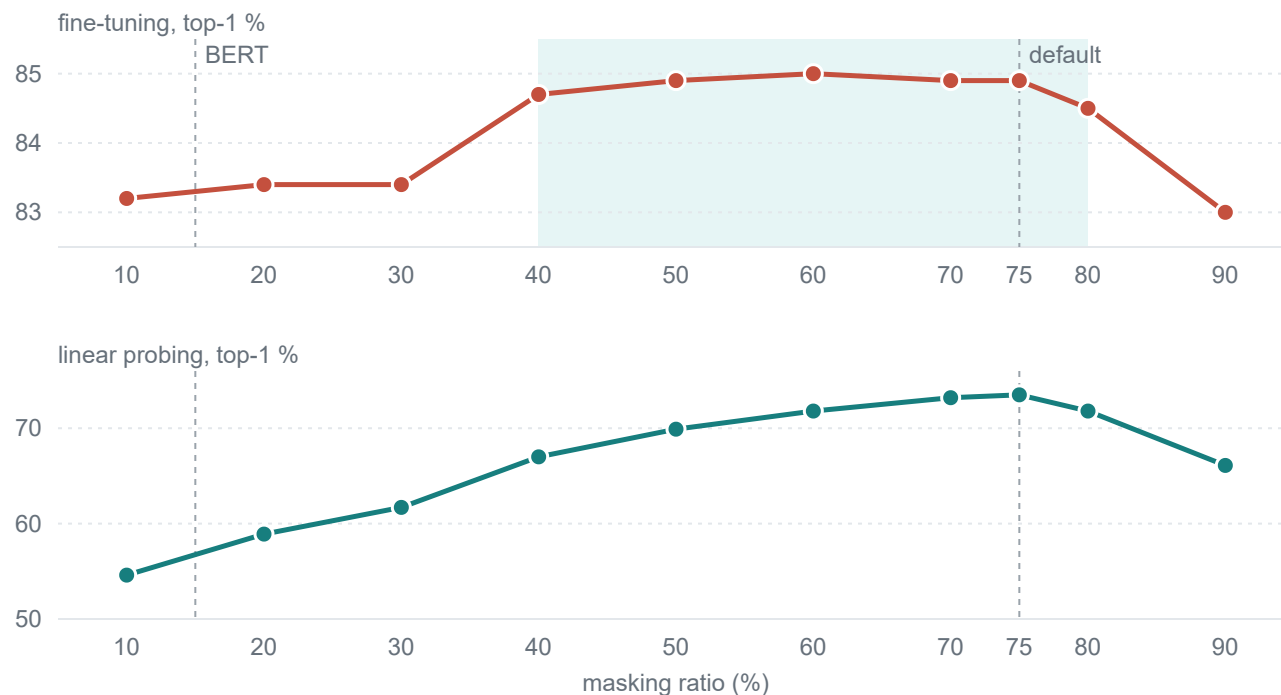
fine-tuning all weights are trained
linear probing encoder frozen, only the linear layer is trained

fine-tuning · the same images, now with their labels · 50 epochs



3 Two ways to evaluate, used throughout the paper. Fine-tuning trains everything: 84.9 percent for the large ViT, against 82.5 trained from scratch for 200 epochs. Linear probing freezes the encoder and trains only the linear layer, so it measures how linearly separable the frozen features are.

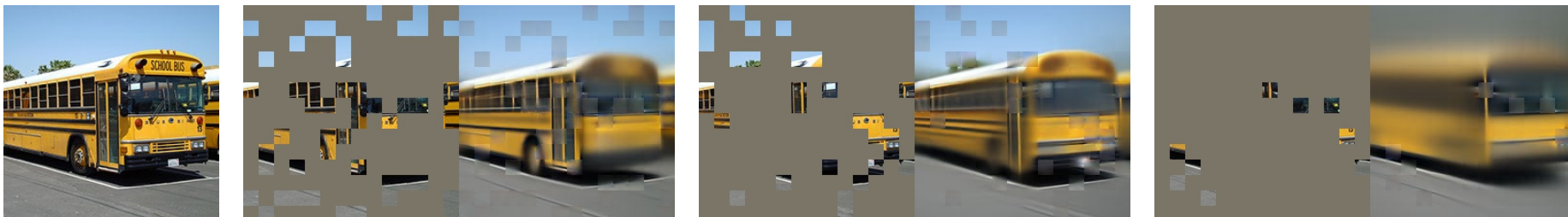
Masking ratio



Large ViT on ImageNet, 800 pretraining epochs per point, top-1 accuracy. Figure 5 in the paper.

Each point is a separate pretraining run with that ratio, then measured two ways. Fine-tuning: anything from 40 to 80 percent works, and every point is above the 82.5 percent of the same model trained from scratch.

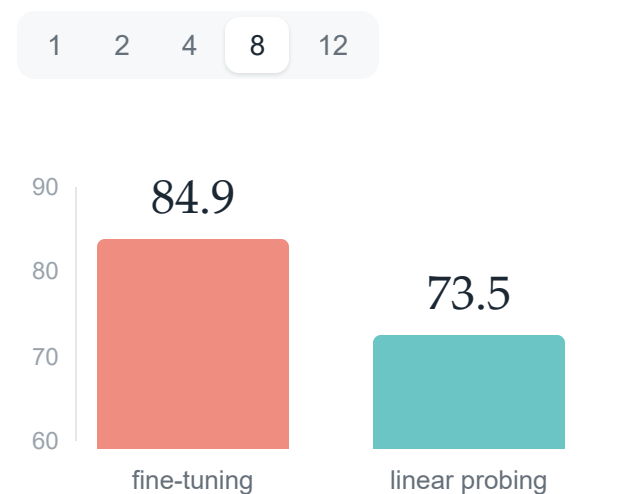
Linear probing keeps improving up to 75 percent, with almost twenty points between the two ends. BERT uses 15 percent. Earlier masked image methods used 20 to 50.



One model trained at 75 percent, tested with 75, 85 and 95 percent hidden. With nine visible patches it still draws a bus. Figure 4.

Decoder depth and mask tokens

Decoder depth in Transformer blocks, Table 1a. Both bars share the same axis.



Eight blocks, the default: 9 percent of ViT-L FLOPs per token, best linear probing.

Decoder depth matters for linear probing and barely for fine-tuning. A deeper decoder takes over the reconstruction-specific layers, so the encoder output stays more abstract. With fine-tuning the encoder's last blocks adapt anyway.

If the mask tokens are kept in the encoder instead of being dropped, linear probing falls by 14 points, because the encoder then trains on inputs full of placeholders it never sees at test time. It also costs 3.3 times the compute.

encoder	decoder blocks	fine-tune acc.	hours	speedup
Large, mask tokens kept in the encoder	8	84.2	42.4	—
Large	8	84.9	15.4	2.8×
Large	1	84.8	11.6	3.7×
Huge	8	85.8	34.5	3.5×
Huge	1	85.9	29.3	4.1×

Training time for 800 epochs on the same hardware. Large and Huge are the ViT sizes. Table 2.

Mask sampling and data augmentation

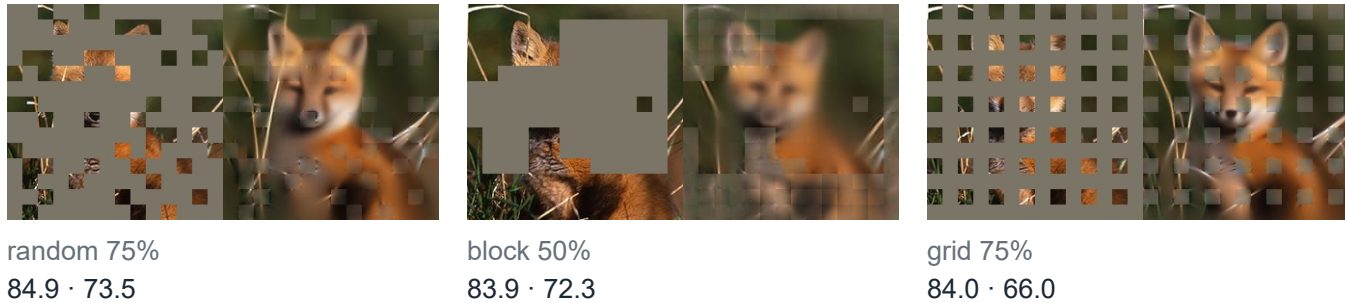


Figure 6 and Table 1f, numbers are fine-tuning · linear probing. Block masking at 75%: 82.8 · 63.9.

Block masking removes large regions. The task gets harder, the loss higher, the output blurrier, and at 75 percent it breaks down. Grid masking keeps one patch in four, which is easy: sharp reconstructions, worse features. Random sampling is the only one that tolerates a high ratio, and the high ratio is where the speed comes from.

augmentation	fine-tuning	linear probing
none, center crop only	84.0	65.7
crop, fixed size	84.7	73.1
crop, random size	84.9	73.5
crop + color jitter	84.3	71.9

Table 1e.

MAE barely needs augmentation. With no augmentation at all it loses less than a point in fine-tuning. Contrastive methods depend on it: SimCLR loses 28 points with cropping only, and with no augmentation its two views are identical and there is nothing to learn. In MAE, a new random mask every iteration already gives a new training sample from the same image.

What should the decoder predict?

The simplest choice is the pixels themselves, and that is what MAE does. The alternative, used by BEiT, a masked image method published a few months earlier, is to predict a code instead of pixels. A separate network first maps every patch to one of 8192 codes, a kind of visual vocabulary. The model then guesses the code of each hidden patch, the way BERT guesses a word. The argument was that codes are more meaningful than pixels.

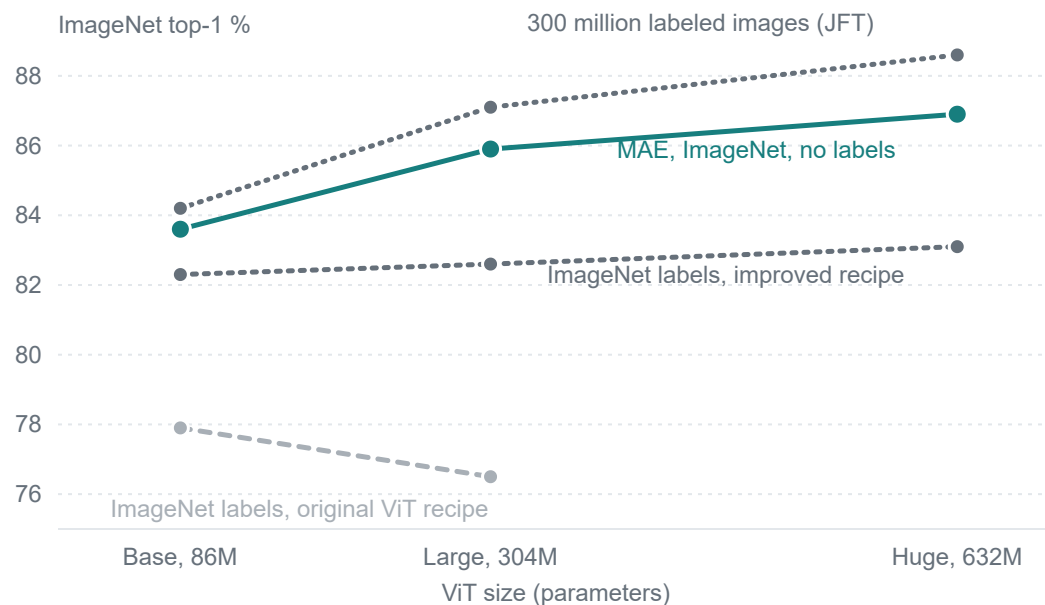
On MAE, codes turn out slightly better than raw pixels, no better than normalized pixels, and worse for linear probing. Predicting a compressed version of the patch is worse than predicting the full patch, so the fine detail is useful too.

The code network also has to be pretrained on 250 million extra images and adds 40 percent to training compute. Pixels are enough.

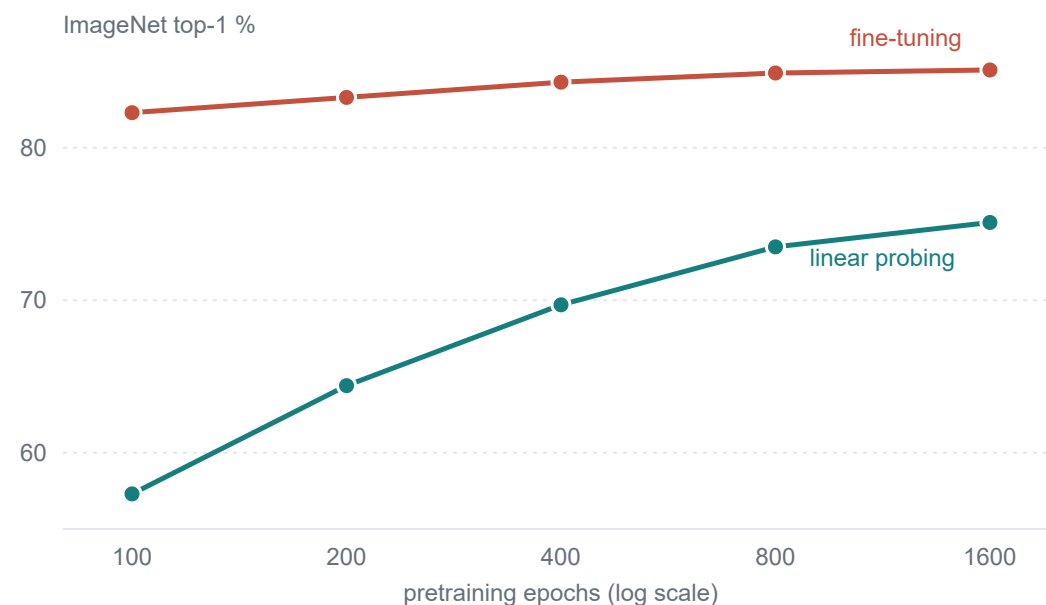
what the decoder predicts	fine-tuning	linear probing
raw pixels	84.9	73.5
pixels, normalized per patch	85.4	73.9
compressed patch (96 PCA components)	84.6	72.3
visual codes, as in BEiT	85.3	71.6

Table 1d. Normalizing per patch means subtracting the patch's own mean and dividing by its standard deviation before comparing.

Scaling



ImageNet accuracy after fine-tuning, for three ViT sizes. Figure 8 in the paper.



Accuracy of the large ViT against how long it was pretrained. Each point is a separate full run. Figure 7.

Trained with labels on ImageNet, a bigger ViT does not get better. Pretrained with MAE it does, and the gain per step in size matches what Google got from 300 million labeled images. The largest model, fine-tuned at higher resolution, reaches 87.8 percent, the best result at the time with no data outside ImageNet. Longer pretraining also keeps helping, with no flattening at 1600 epochs, where contrastive methods on ViT stop improving around 300. Since the encoder only sees a quarter of one image per step, 1600 MAE epochs still take less time than 300 contrastive ones.

Partial fine-tuning

Number of blocks fine-tuned, counted from the top of the large ViT. 0 is linear probing, 24 is full fine-tuning.

0 blocks tuned · linear probing, all 24 frozen



1 block tuned · the last block, in salmon

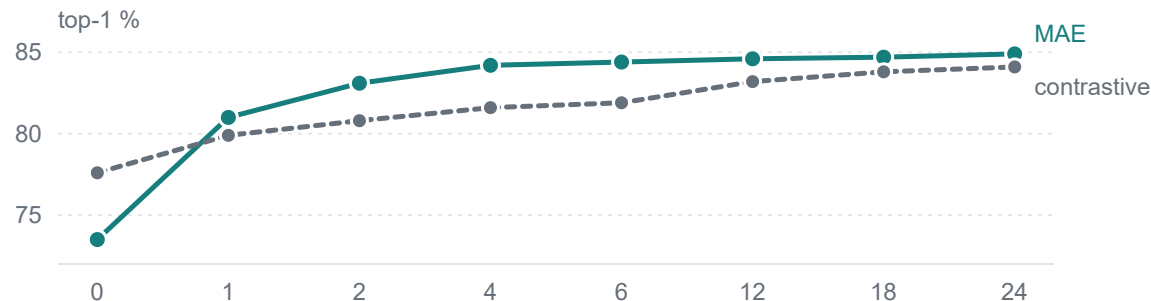


Figure 9.

The comparison is with MoCo v3, a contrastive method from the same family as SimCLR, trained on the same ViT. With the encoder frozen, its features are four points better. Tune a single block and MAE goes from 73.5 to 81.0 and stays ahead from there on. Tuning just the feed-forward half of the last block already gives 79.1.

So the MAE features are there, they are just not arranged so a single linear layer can separate them. The authors argue that linear probing, the usual test in contrastive work, misses exactly this, and suggest tuning the last few blocks as a fairer middle ground. In language modeling nobody evaluates with a frozen encoder anyway.

Transfer to other tasks

pretraining	detection, COCO Base	Large	segmentation, ADE20K Base	Large
ImageNet labels	47.9	49.3	47.4	49.9
contrastive (MoCo v3)	47.9	49.3	47.3	49.1
visual codes (BEiT)	49.8	53.3	47.1	53.3
MAE	50.3	53.3	48.1	53.6

Detection is box AP, segmentation is mean IoU; higher is better for both. Base and Large are the ViT sizes. Standard detection and segmentation heads, all methods fine-tuned the same way. Tables 4 and 5.

MAE, after ImageNet fine-tuning	Base	Large	Huge	Huge, 448 px	previous best
iNaturalist 2018, species classification	75.4	80.1	83.0	86.8	81.2
ImageNet-A, images that fool classifiers	35.9	57.1	68.2	76.7	35.8
ImageNet-R, paintings and sketches	48.3	59.9	64.4	66.5	48.7

The last two rows use the ImageNet-fine-tuned model as is, with no extra training. Tables 6 and 13.

Does the ImageNet gain carry over to other tasks? Yes, and it grows with model size. On detection, MAE is 2.4 points above training with labels for the base model and 4.0 for the large one. The contrastive method gives no gain at all here, even though it looked better under a linear probe. Segmentation shows the same pattern.

On fine-grained species classification the huge model beats the previous record by more than five points.

The robustness rows are the most striking. On the adversarial set, the huge MAE model scores 68.2 where the same model trained with labels scores 33.1.

Summary

What MAE does: hide 75 percent of the patches, run only the visible ones through the encoder, let a small decoder fill in the pixels, then throw the decoder away and keep the encoder.

Why it matters: it is simple and fast, it needs almost no augmentation, and it gets better as the model gets bigger and trains longer, which contrastive methods on ViT did not. With nothing but ImageNet, a huge ViT reaches 87.8 percent, and the gains carry over to detection and segmentation.

One more lesson: linear probing can mislead. MAE looks weak under a linear probe and strong as soon as a single block is tuned.

The patches we hide are not objects, and the pixels we predict mean nothing on their own. The encoder still ends up understanding the image, because there is no other way to fill in three quarters of it.

Paraphrasing the discussion section of the paper.

 see 25%  hide 75%  predict pixels

Thank you

Questions and discussion

He, Chen, Xie, Li, Dollár, Girshick. Masked Autoencoders Are Scalable Vision Learners. CVPR 2022. [arXiv:2111.06377](https://arxiv.org/abs/2111.06377).

Figures and tables in this talk are from the paper.