# Paper Review:
# Masked Autoencoders (MAE) Are Scalable Vision Learners

He, Kaiming, et al. CVPR 2022

**Presenter : Chenwei Wu**

# Agenda today

- What is MAE? How MAE works?
- Comparison with BERT
- Intriguing Features Analysis
- Key Code Snippets Walkthrough
- Applications

[He et al, CVPR 2022]

# What is MAE?

- Very simple method, but highly effective

[Devlin et al, NAACL 2019] [He et al, CVPR 2022]

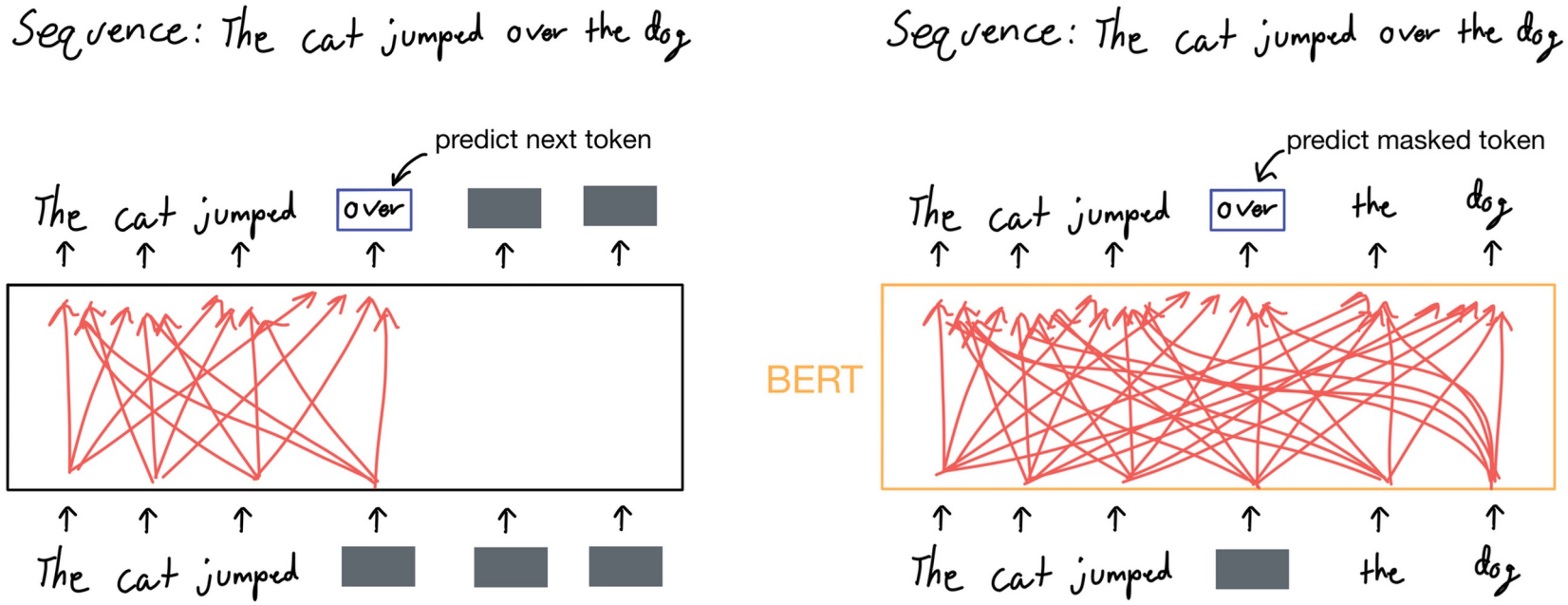
# What is MAE?

- Very simple method, but highly effective
- BERT-like algorithm, but with crucial design changes for vision
- BERT stands for Bidirectional Encoder Representations from Transformers

[Devlin et al, NAACL 2019] [He et al, CVPR 2022]

# What is MAE?

- Very simple method, but highly effective
- BERT-like algorithm, but with crucial design changes for vision

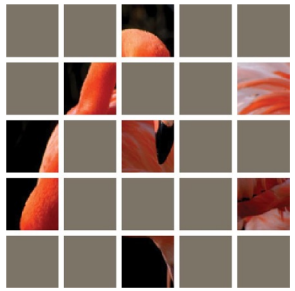


[Devlin et al, NAACL 2019] [He et al, CVPR 2022]

# What is MAE?

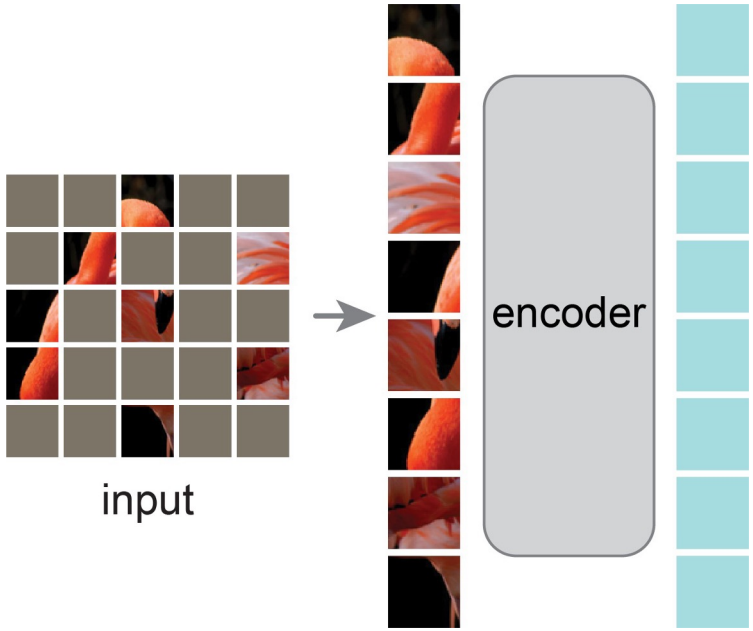
- Very simple method, but highly effective
- BERT-like algorithm, but with crucial design changes for vision
- Intriguing properties – better scalability and more from analysis

# How MAE Works?



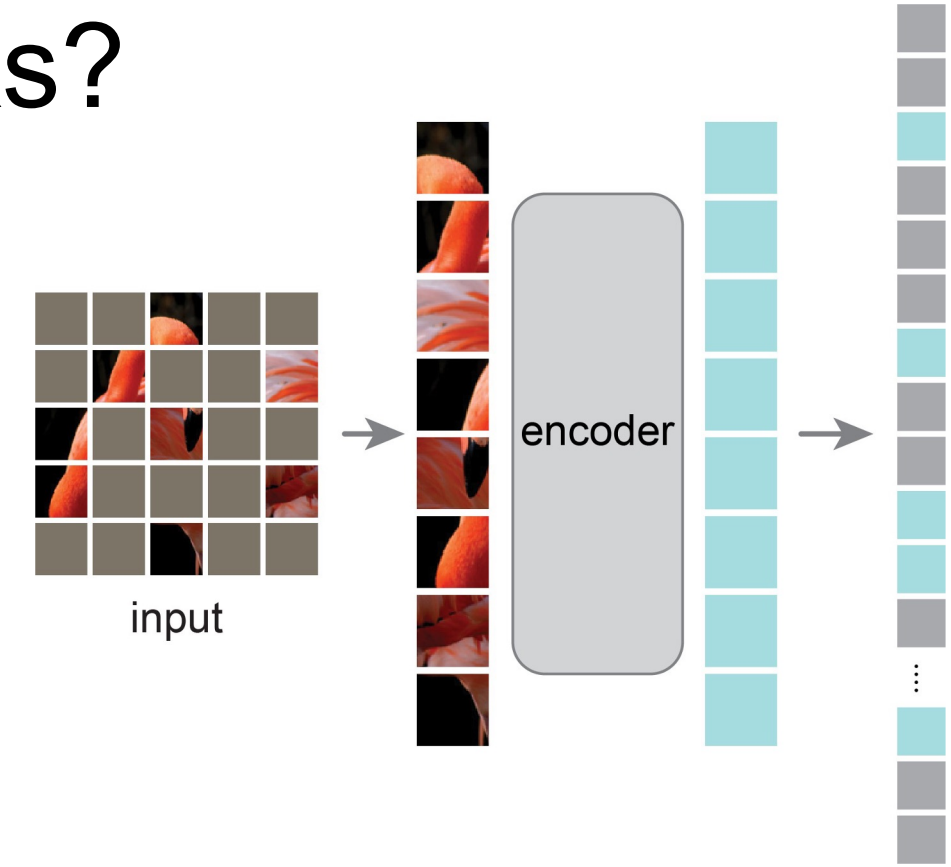
Random masking

# How MAE Works?




Encode visible patches

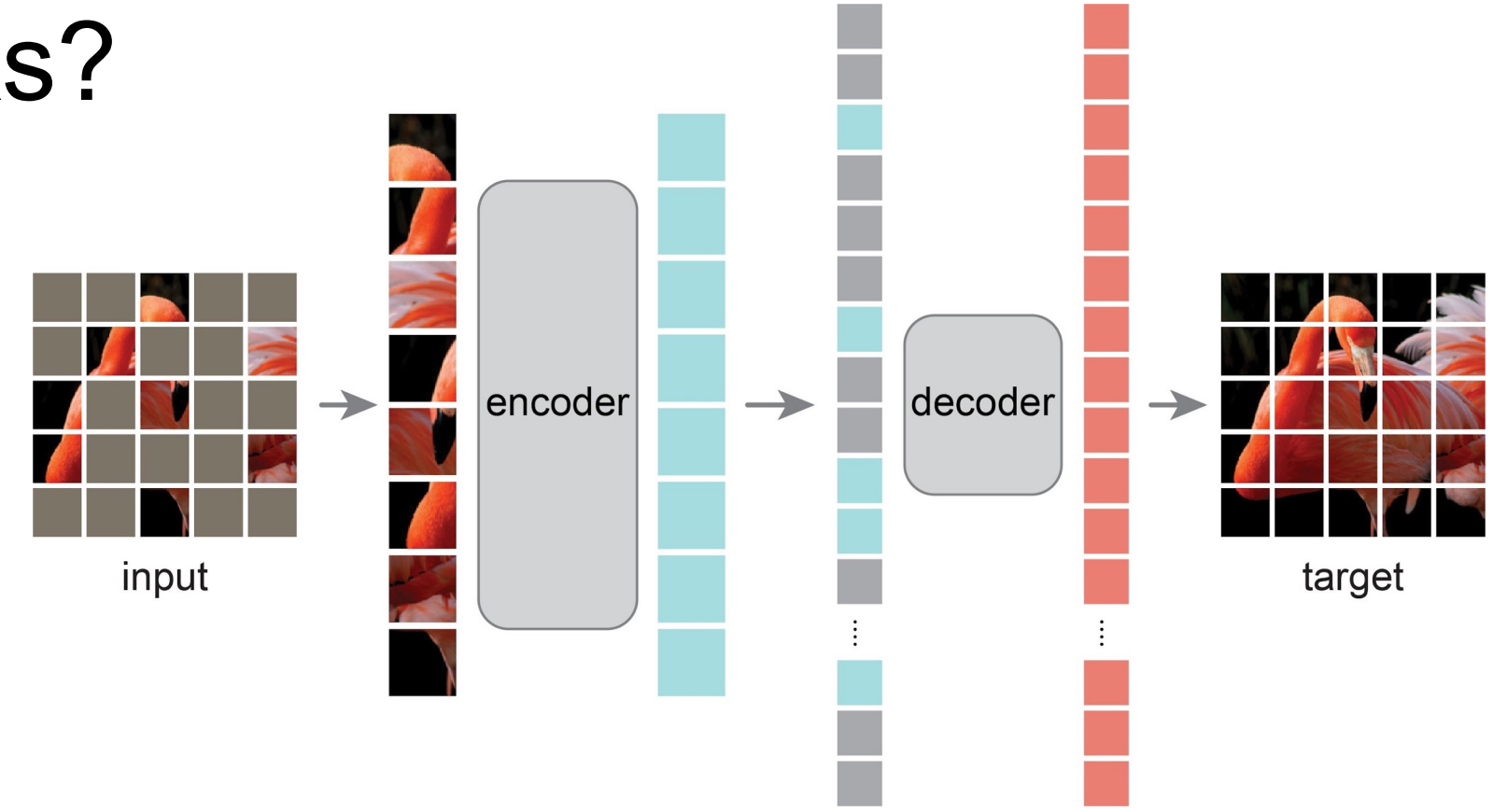
# How MAE Works?




Each mask token is a shared, learned vector that indicates the presence of a missing patch. Positional encodings are again applied to communicate to the decoder where the individual patches are located in the original image.

Add mask tokens

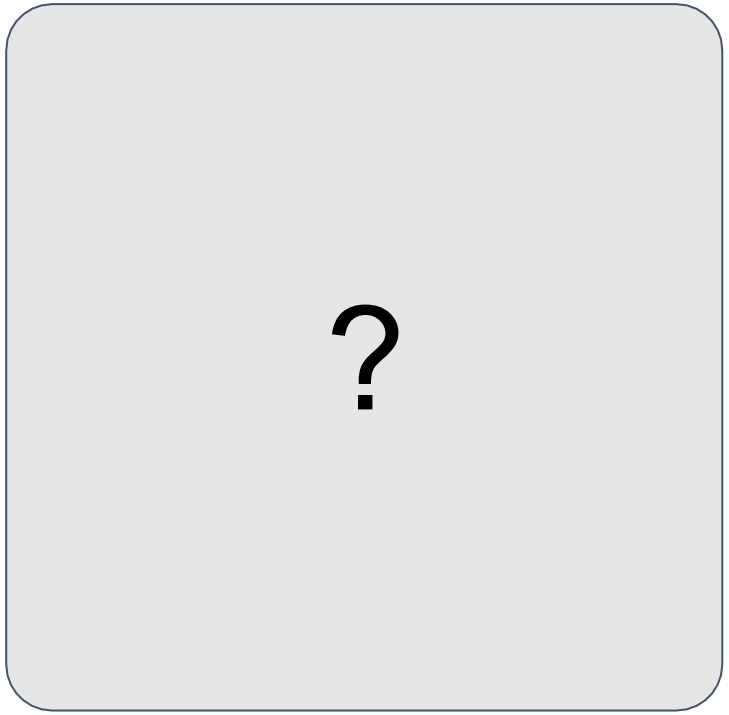
# How MAE Works?




Reconstruct (MSE)

# MAE Reconstruction Example

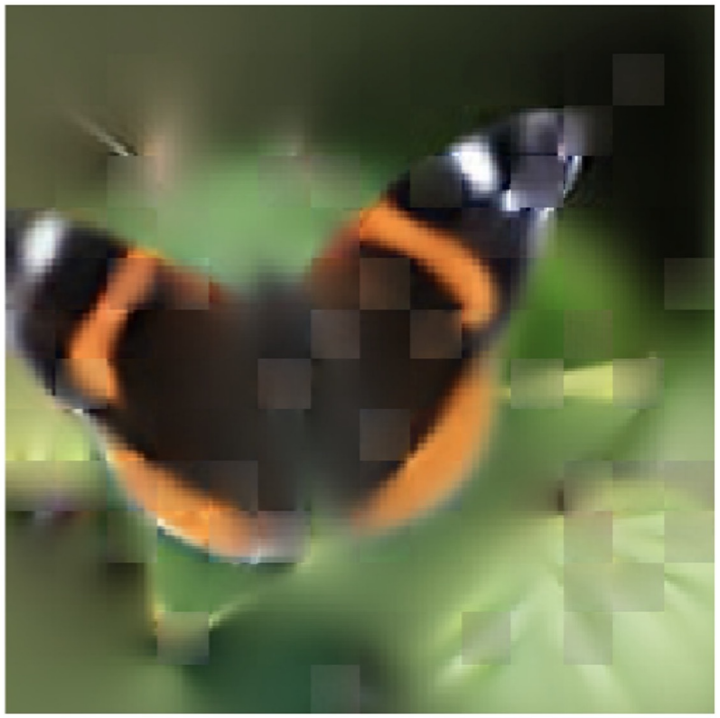
Masked input:
80%




You
guess?

# MAE Reconstruction Example

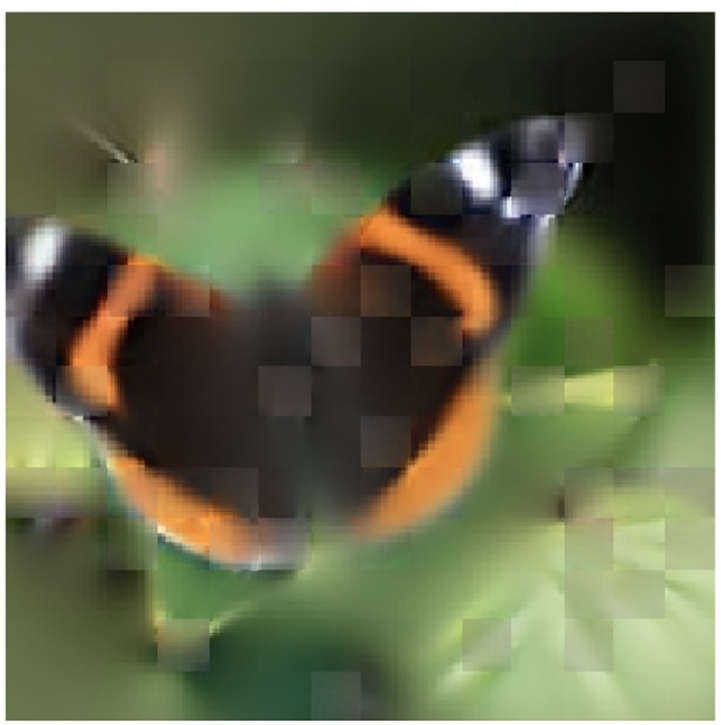
Masked input:
80%



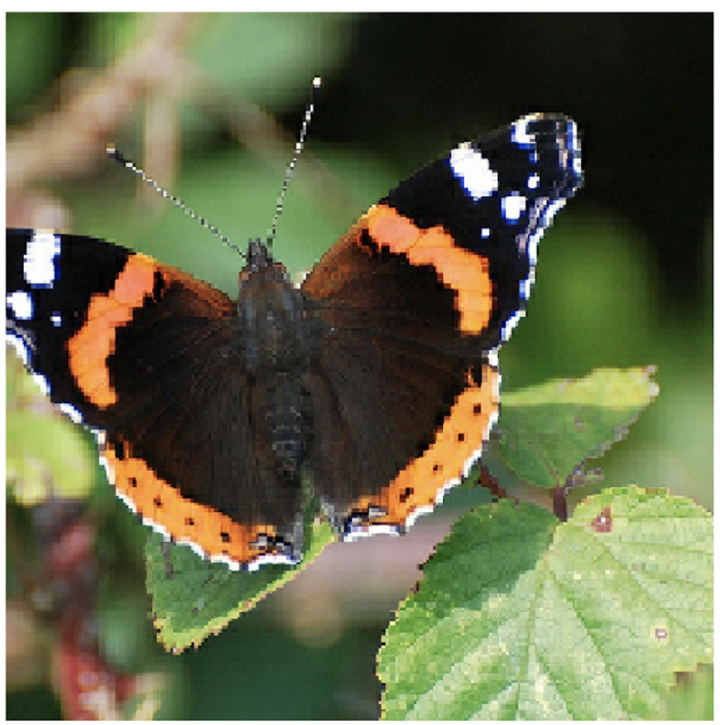
MAE's
guess

# MAE Reconstruction Example

Masked input: 80%

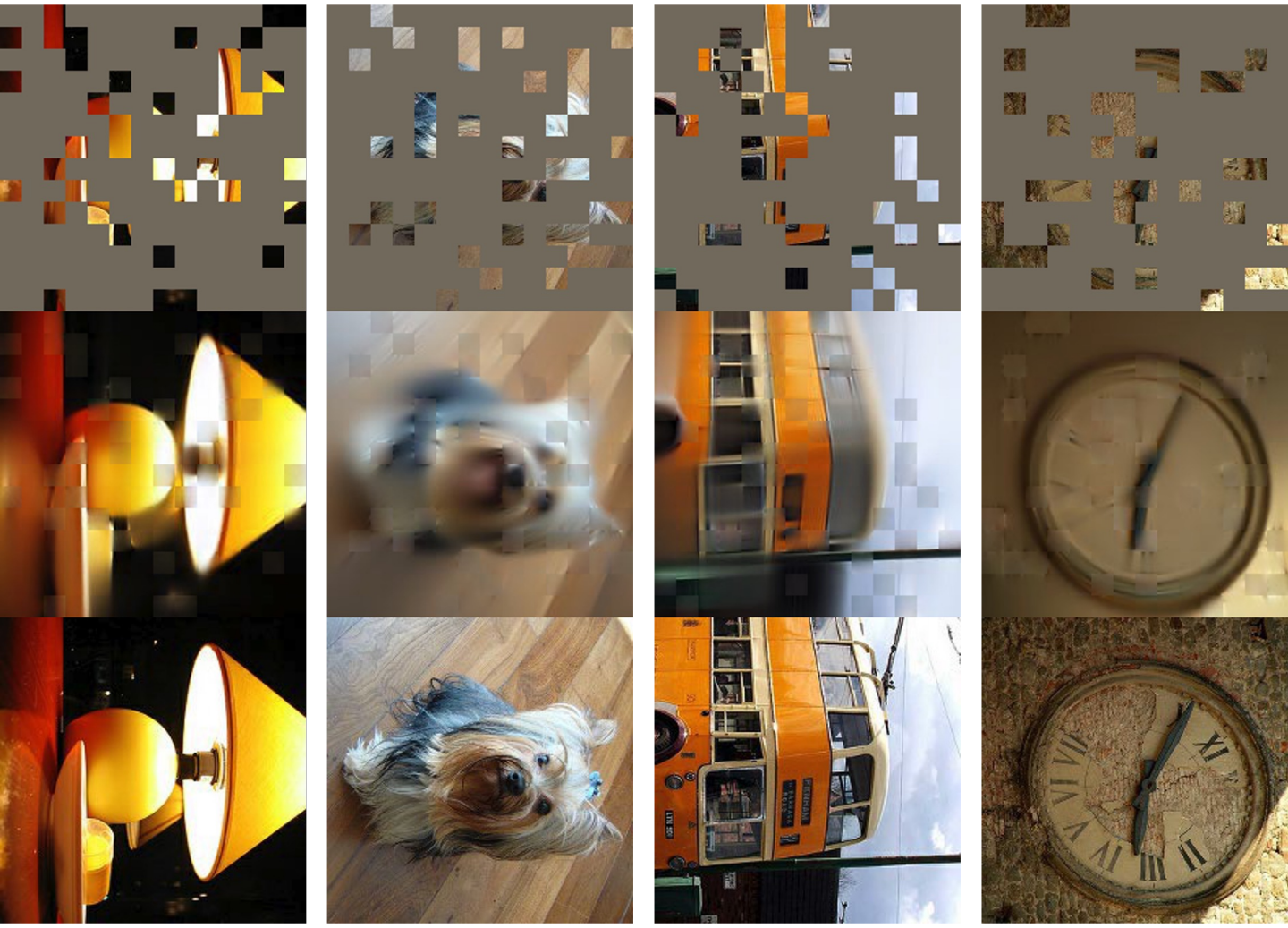


MAE's guess

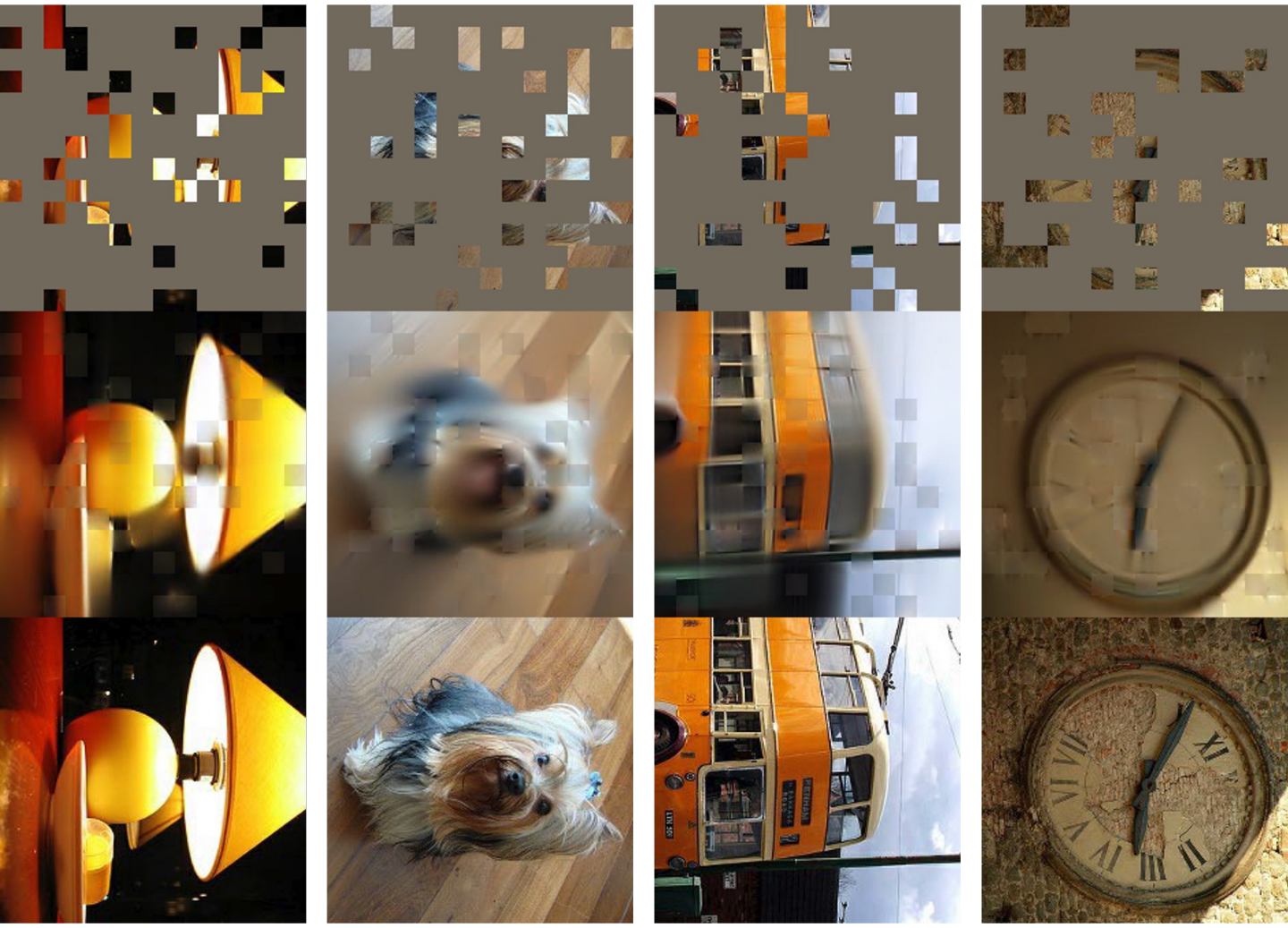


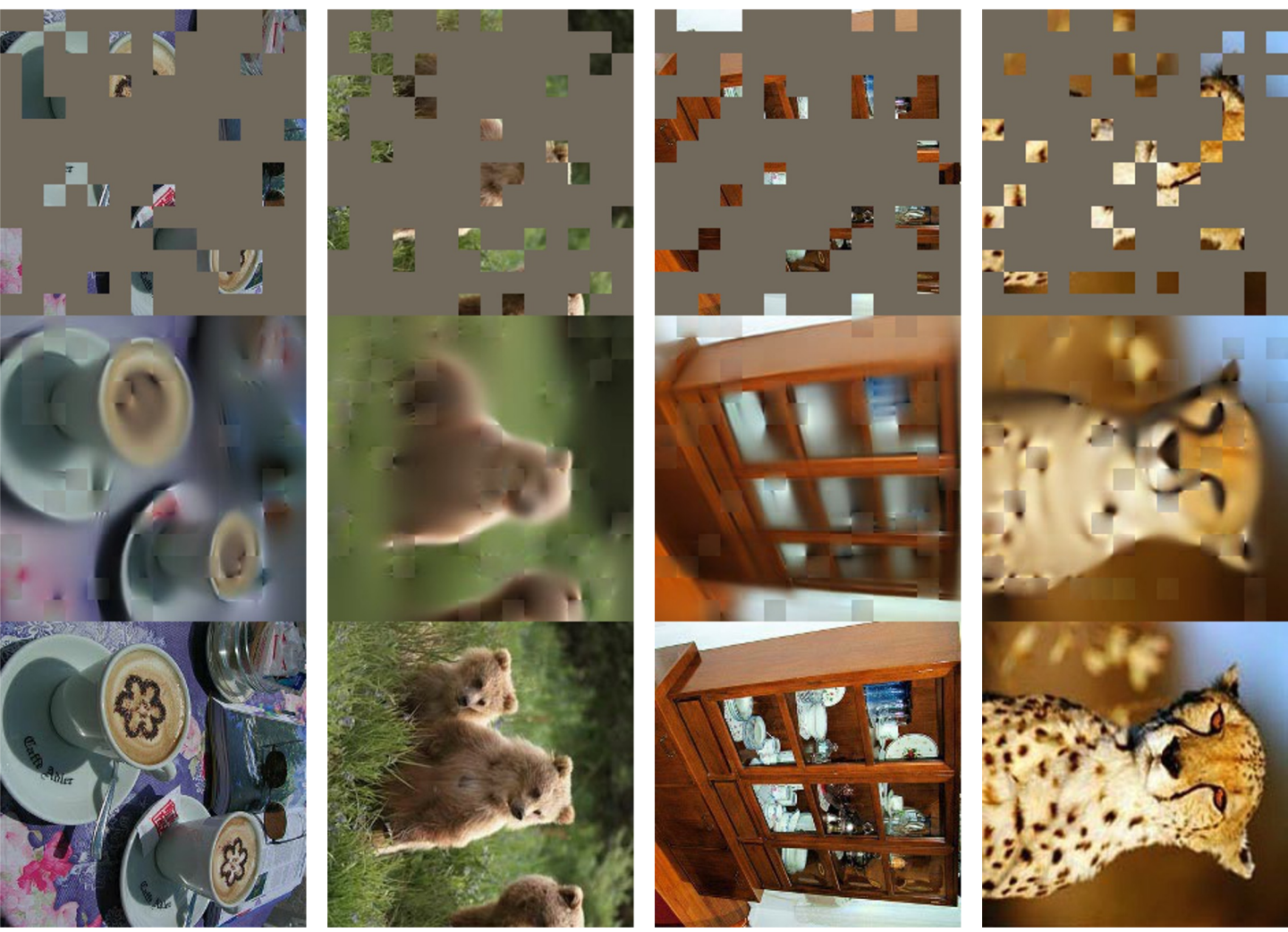
Ground truth

# ImageNet val set (unseen)

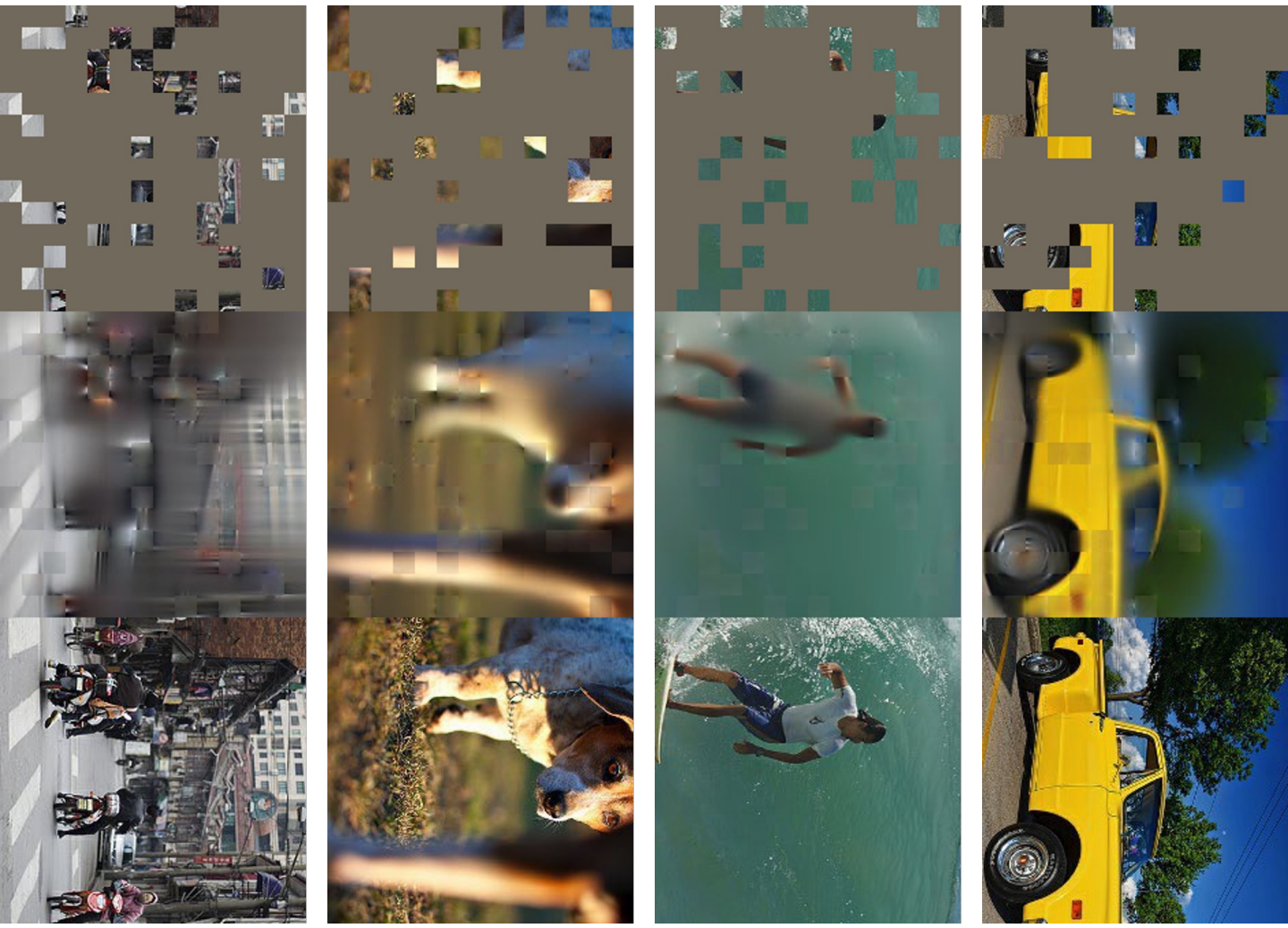


# ImageNet val set (unseen)





# COCO val set (unseen)



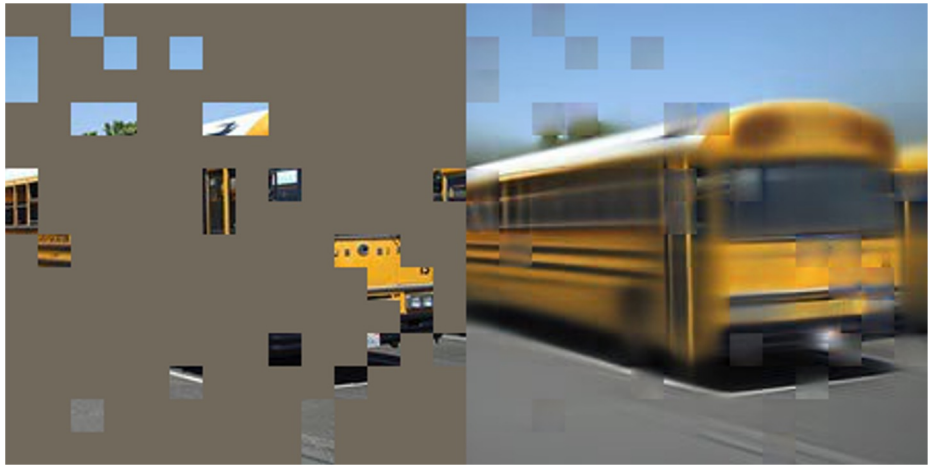

original

75%
mask

# MAE Can Generalize


original

75%
mask

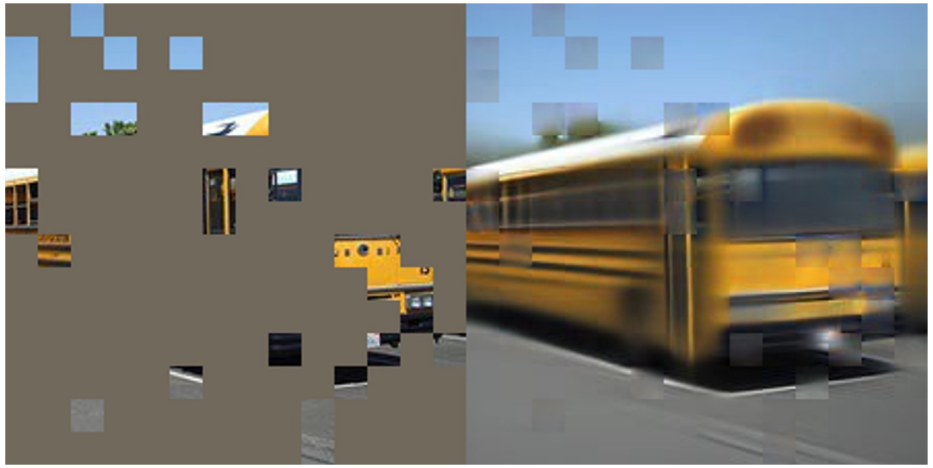


85%
mask

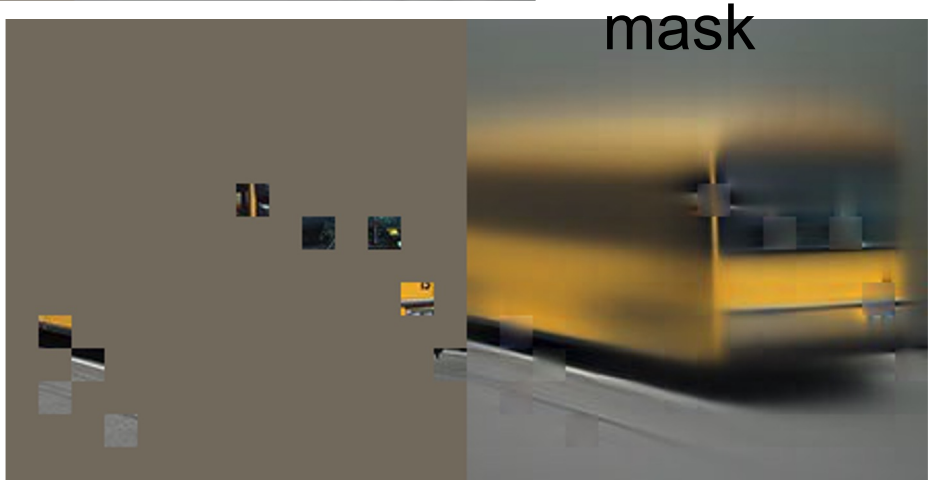
# MAE Can Generalize


original

75%
mask



85%
mask



95%
mask

# MAE Can Generalize

original

75%
mask

# MAE Can Generalize

original

75%
mask

85%
mask

# MAE Can Generalize

origina
l

75%
mask

85%
mask

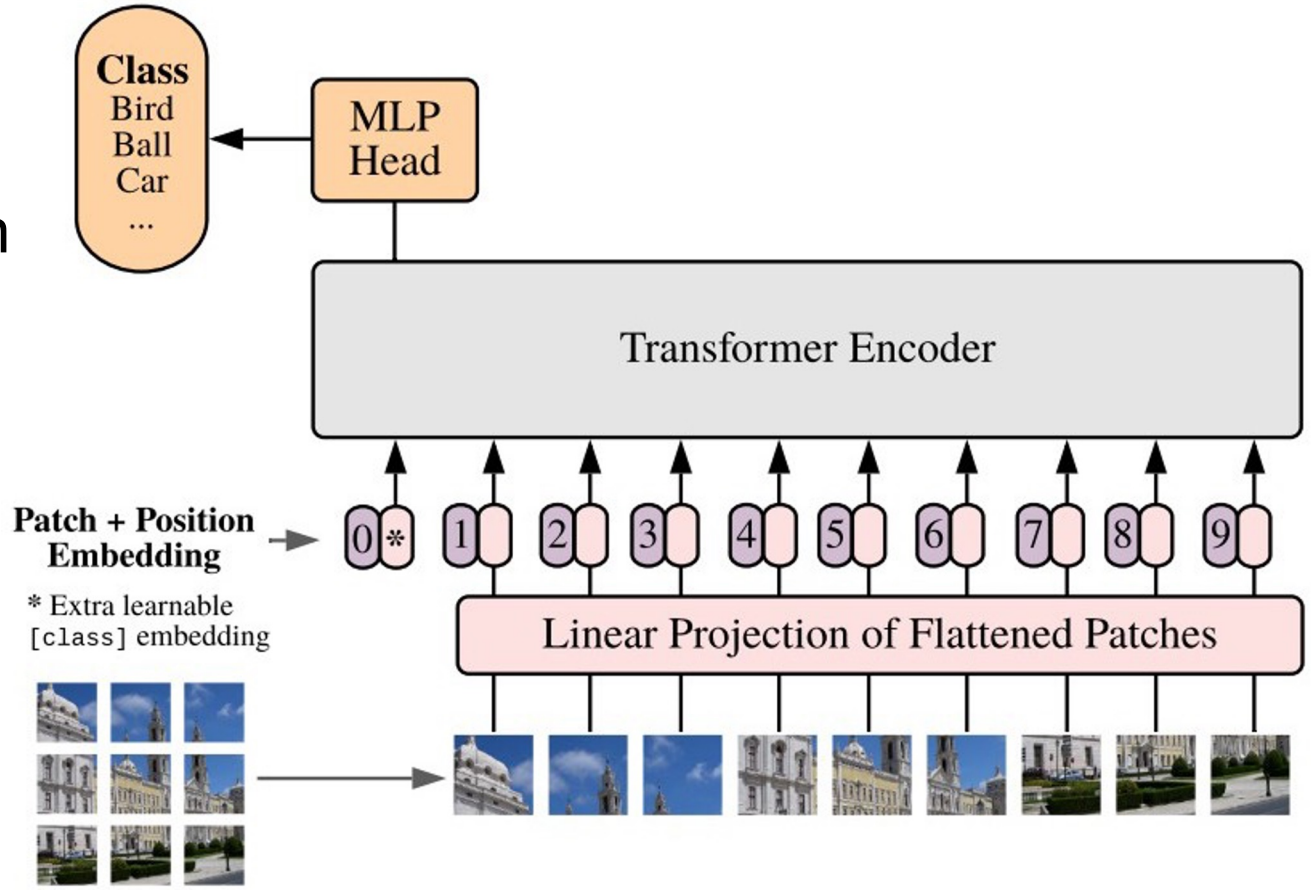
95%

# MAE Can Generalize

[Dosovitskiy et al, ICLR 2021]

# BERT-like: Transformers

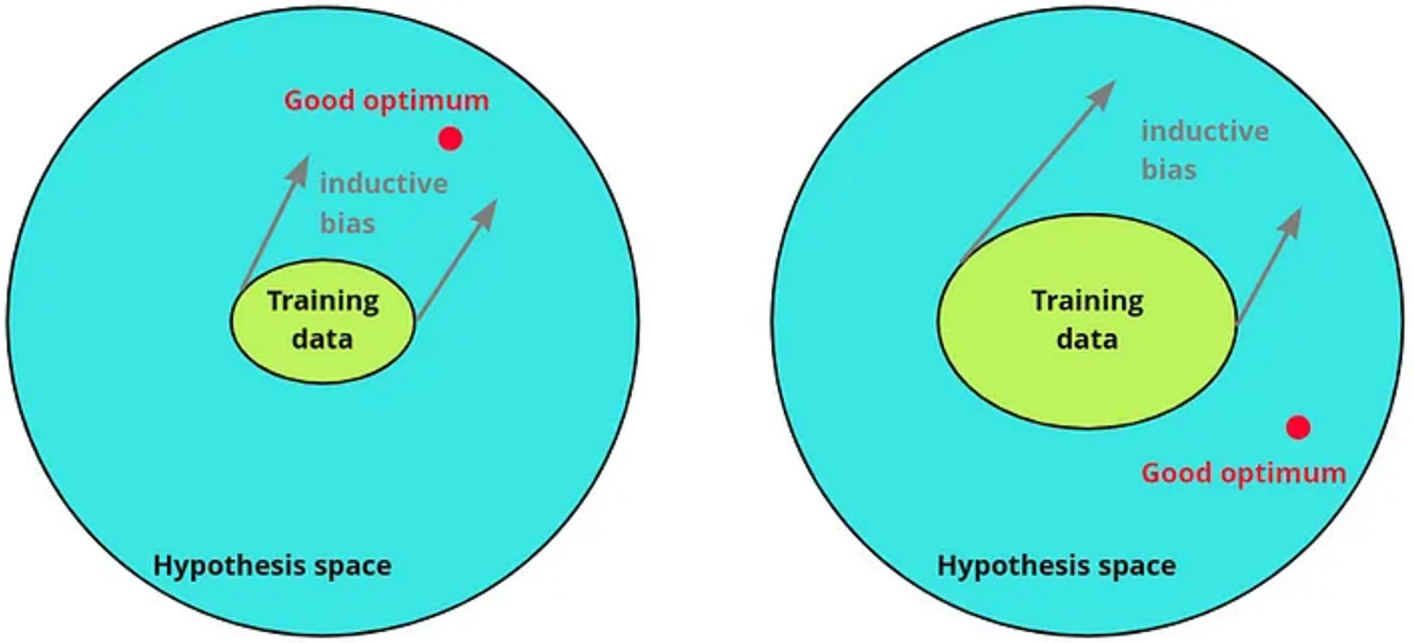
- Vision Transformer (ViT)
  - Less inductive bias
  - Non-overlapping tokenization
    - Easier for masked auto-encoding



[Dosovitskiy et al, ICLR 2021]

# BERT-like: Transformers

- Vision Transformer (ViT)
  - Less inductive bias

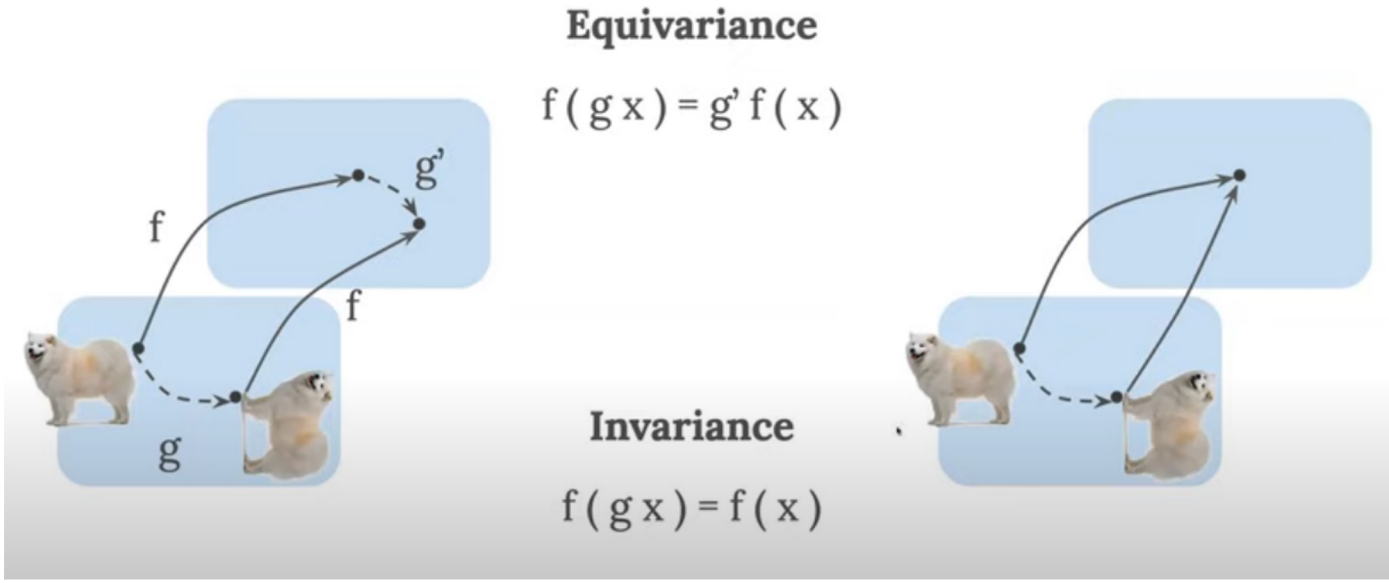



In a low data setting, right inductive bias may help to find good optimum, but in a rich data setting, it may lead to constrains that harm generalization

[Dosovitskiy et al, ICLR 2021]

# BERT-like: Transformers

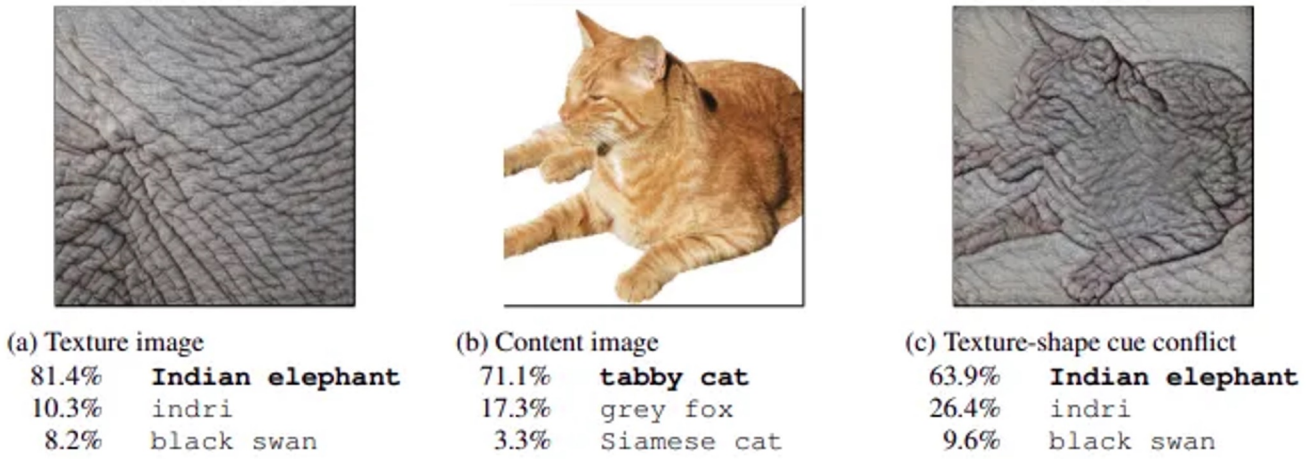
- Vision Transformer (ViT)
  - Less inductive bias
  - CNNs?
  - Locality implies that closely placed pixels are related to each other. (smaller filters)
  - Weight sharing implies searching for specific patterns.
    - Translation Invariance
    - (There are equivariant CNNs, a lot applied in physics)

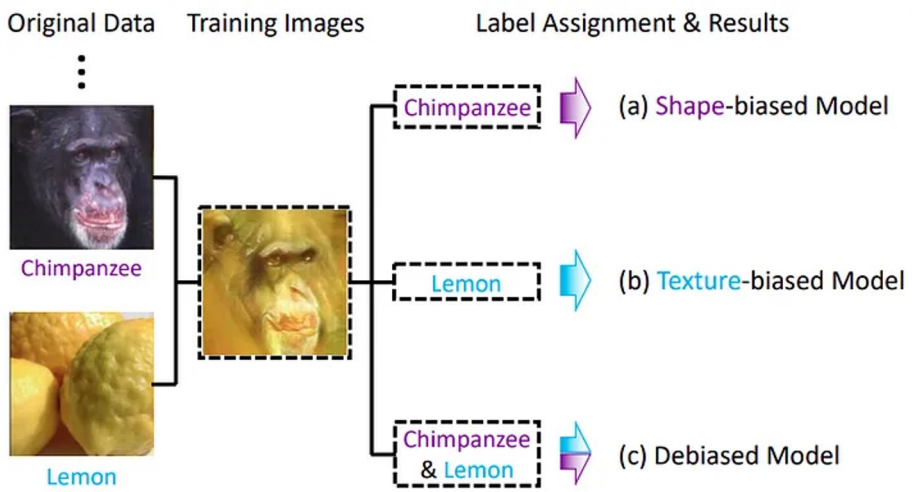


# BERT-like: Transformers

[Dosovitskiy et al, ICLR 2021] [Geirhos et al, ICLR 2019]

- Vision Transformer (ViT)
  - Less inductive bias
  - CNNs?
  - Other biases
  - Imagenet-Trained CNNS Are Biased Towards Textures (Geirhos et al, ICLR2019)

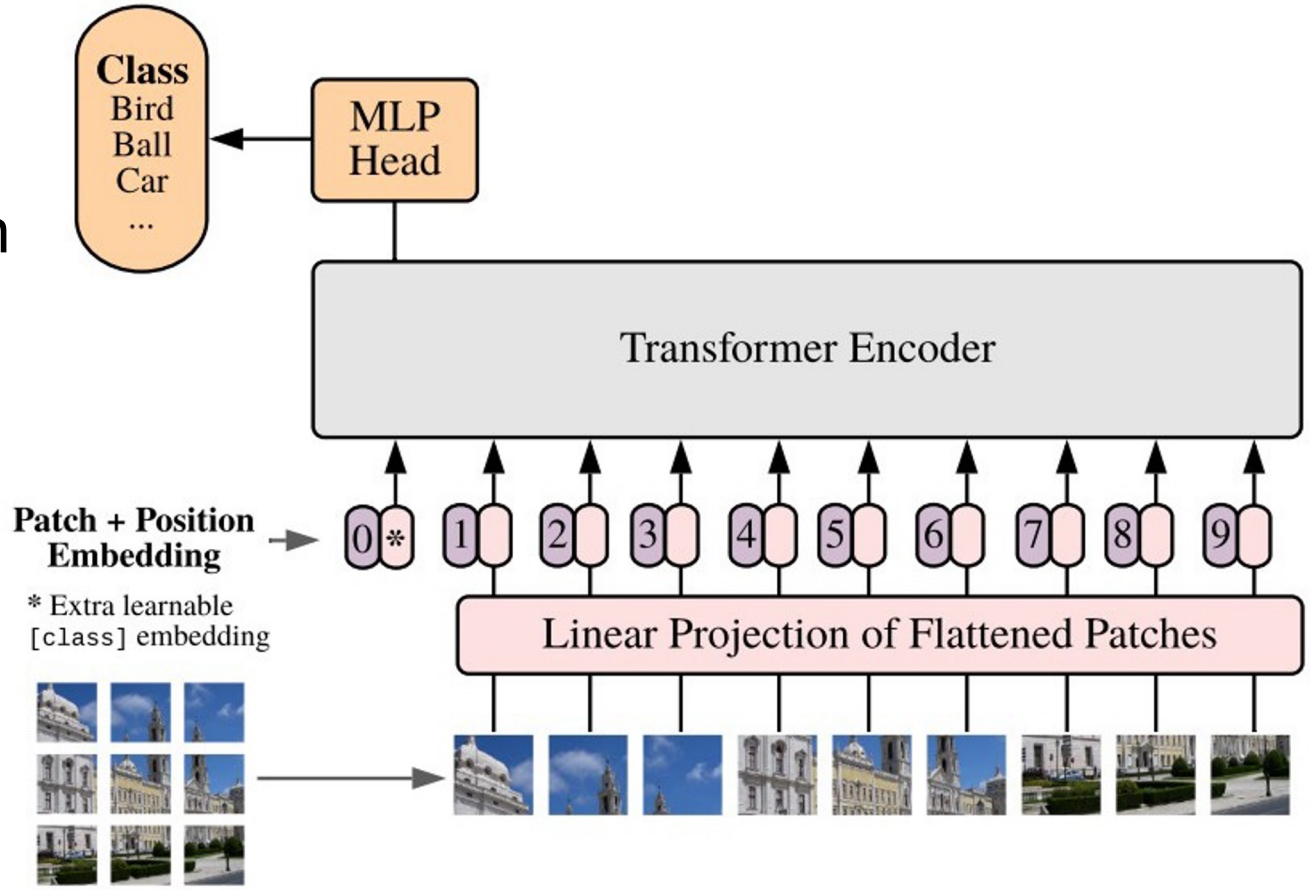


[Dosovitskiy et al, ICLR 2021]

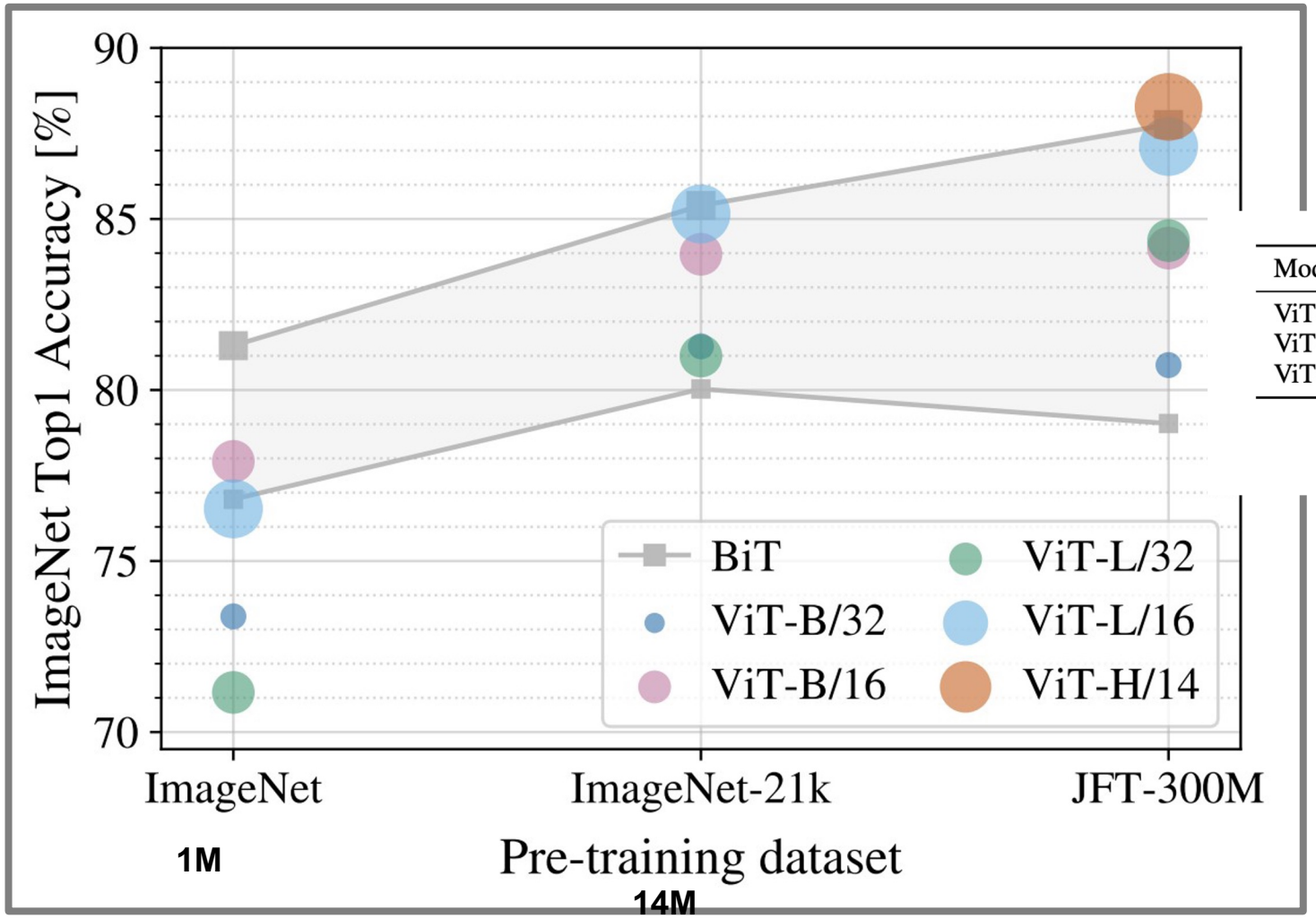
# BERT-like: Transformers

- Vision Transformer (ViT)
  - Less inductive bias
  - Non-overlapping tokenization
    - Easier for masked auto-encoding
- *Scalable*
  - with larger models
  - on larger datasets



[Dosovitskiy et al, ICLR 2021]

# BERT-like: Transformers




| Model | Layers | Hidden size $D$ | MLP size | Heads | Params |
|---|---|---|---|---|---|
| ViT-Base | 12 | 768 | 3072 | 12 | 86M |
| ViT-Large | 24 | 1024 | 4096 | 16 | 307M |
| ViT-Huge | 32 | 1280 | 5120 | 16 | 632M |

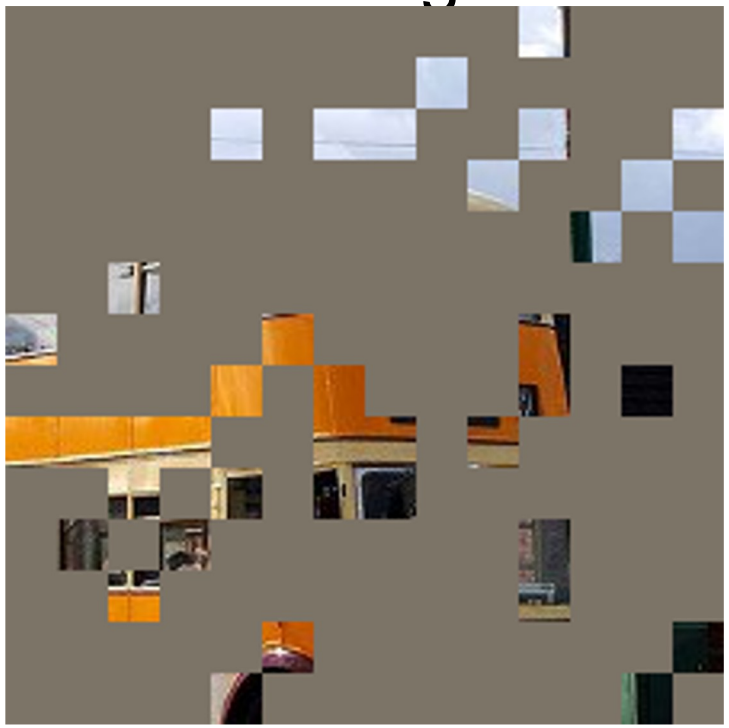
Table 1: Details of Vision Transformer model variants.

[Devlin et al, NAACL 2019]

# BERT-*unlike*: Mask Ratio

- BERT: 15% is enough to create a challenging task
- MAE: a high ratio of 75% - 80% is about optimal

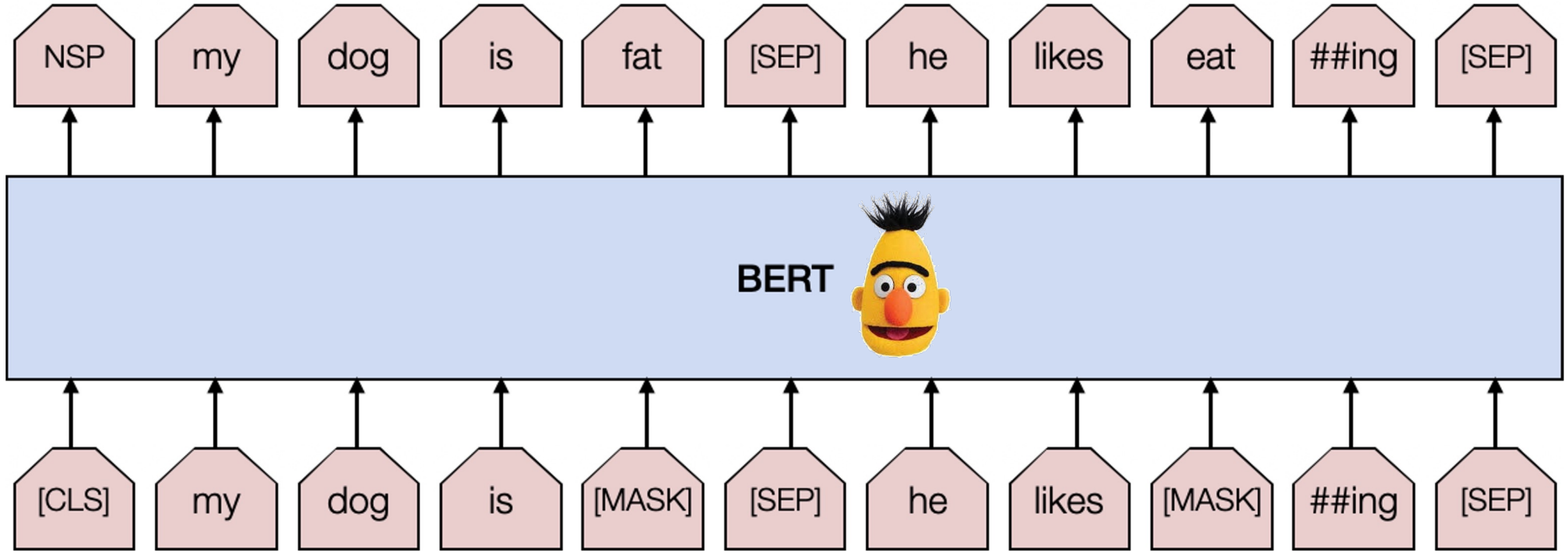
**Info density gap!**



[Devlin et al, NAACL 2019]

# BERT-*unlike*: Encoder-Decoder

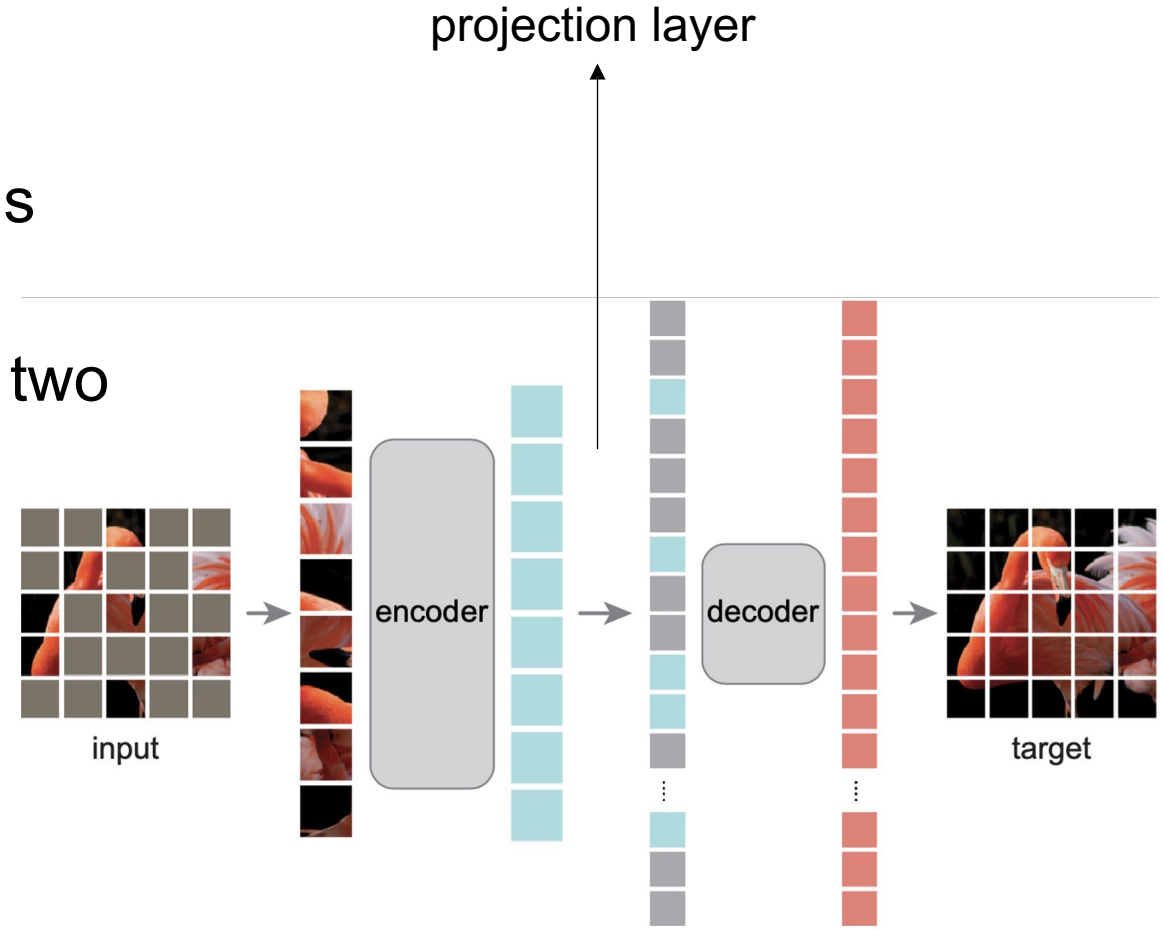
- BERT: encoder-*only* pre-training



[Devlin et al, NAACL 2019]

# BERT-*unlike*: Encoder-Decoder

- MAE:
  - *Large* encoder on *visible* tokens
  - Small decoder on *all* tokens
  - *Projection* layer to connect the two

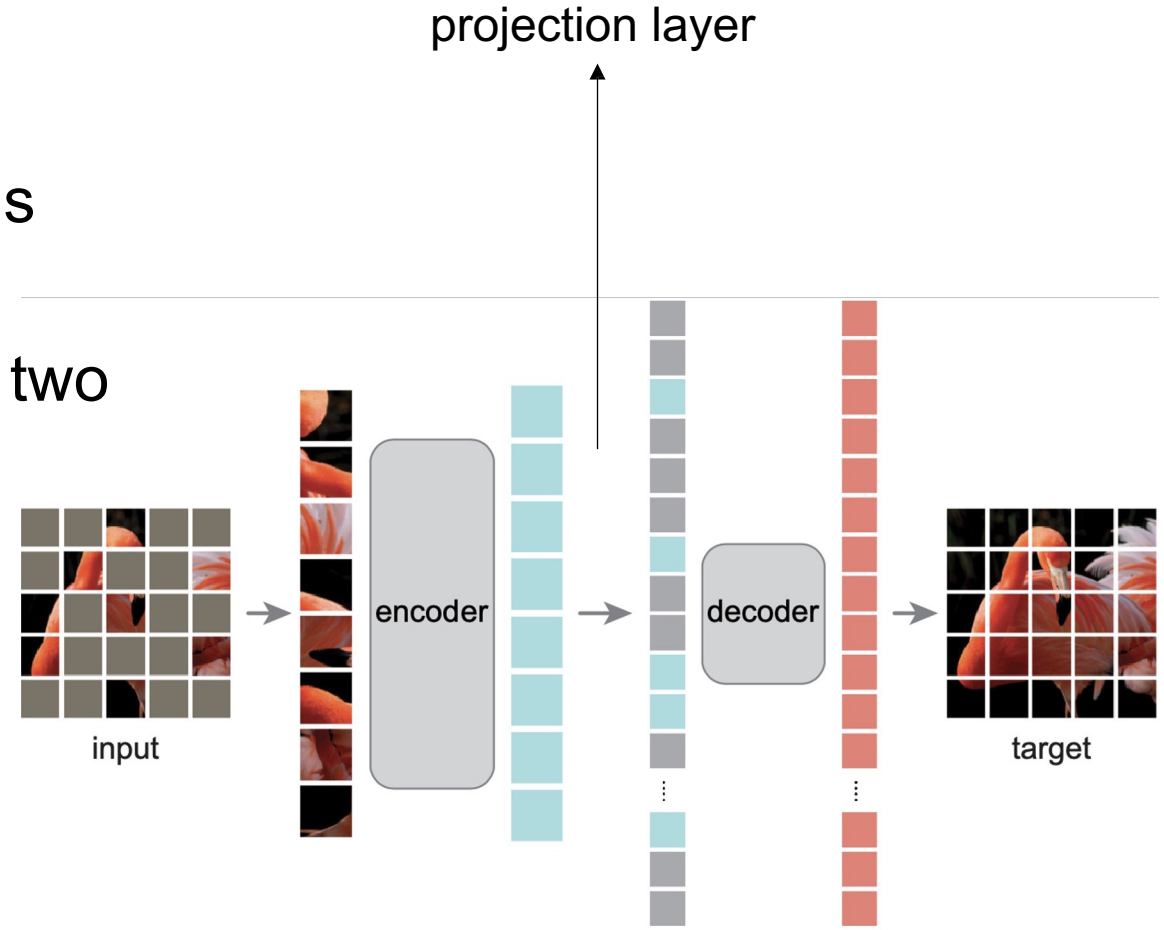


[Devlin et al, NAACL 2019]

# BERT-*unlike*: Encoder-Decoder

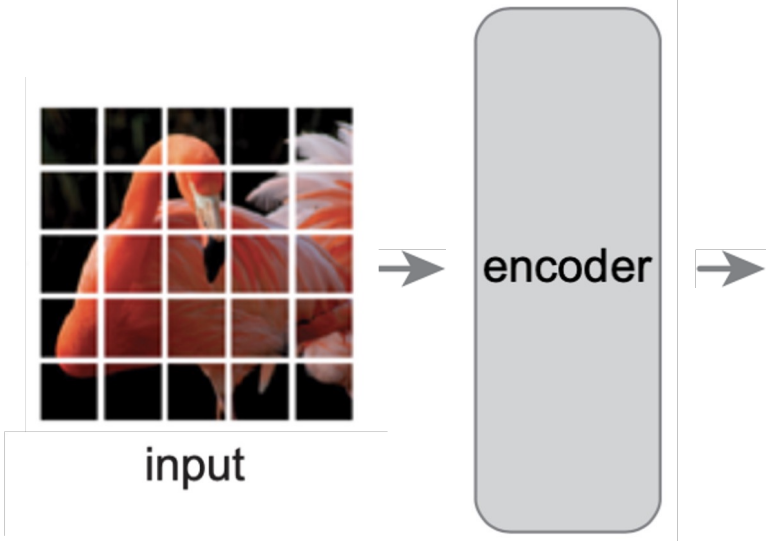
- MAE:
  - *Large* encoder on *visible* tokens
  - Small decoder on *all* tokens
  - *Projection* layer to connect the two

- Very efficient when coupled with high mask ratio (75%)



# MAE for Downstream Tasks: *Encoder Only*

- After MAE pre-training, just *throw away* the decoder
- Encoder is used for representations with *full-sequence* input
- Small Decoder!



# Experimental Protocols

- Pre-training dataset: ImageNet-1K
- Architecture: ViT-*Large* encoder, 512-dim decoder

| Model | Layers | Hidden size $D$ | MLP size | Heads | Params |
|---|---|---|---|---|---|
| ViT-Base | 12 | 768 | 3072 | 12 | 86M |
| ViT-Large | 24 | 1024 | 4096 | 16 | 307M |
| ViT-Huge | 32 | 1280 | 5120 | 16 | 632M |

Table 1: Details of Vision Transformer model variants.

# Experimental Protocols

- Pre-training dataset: ImageNet-1K
- Architecture: ViT-*Large* encoder, 512-dim decoder
- Transfer task: ImageNet-1K classification
  - "*ft*": end-to-end tuning with MAE as an initialization
  - "*lin*": linear probing, a single classifier on top of frozen encoder features

# Analysis: Decoder Size

- Encoder has 24-blocks, 1024-dimensional

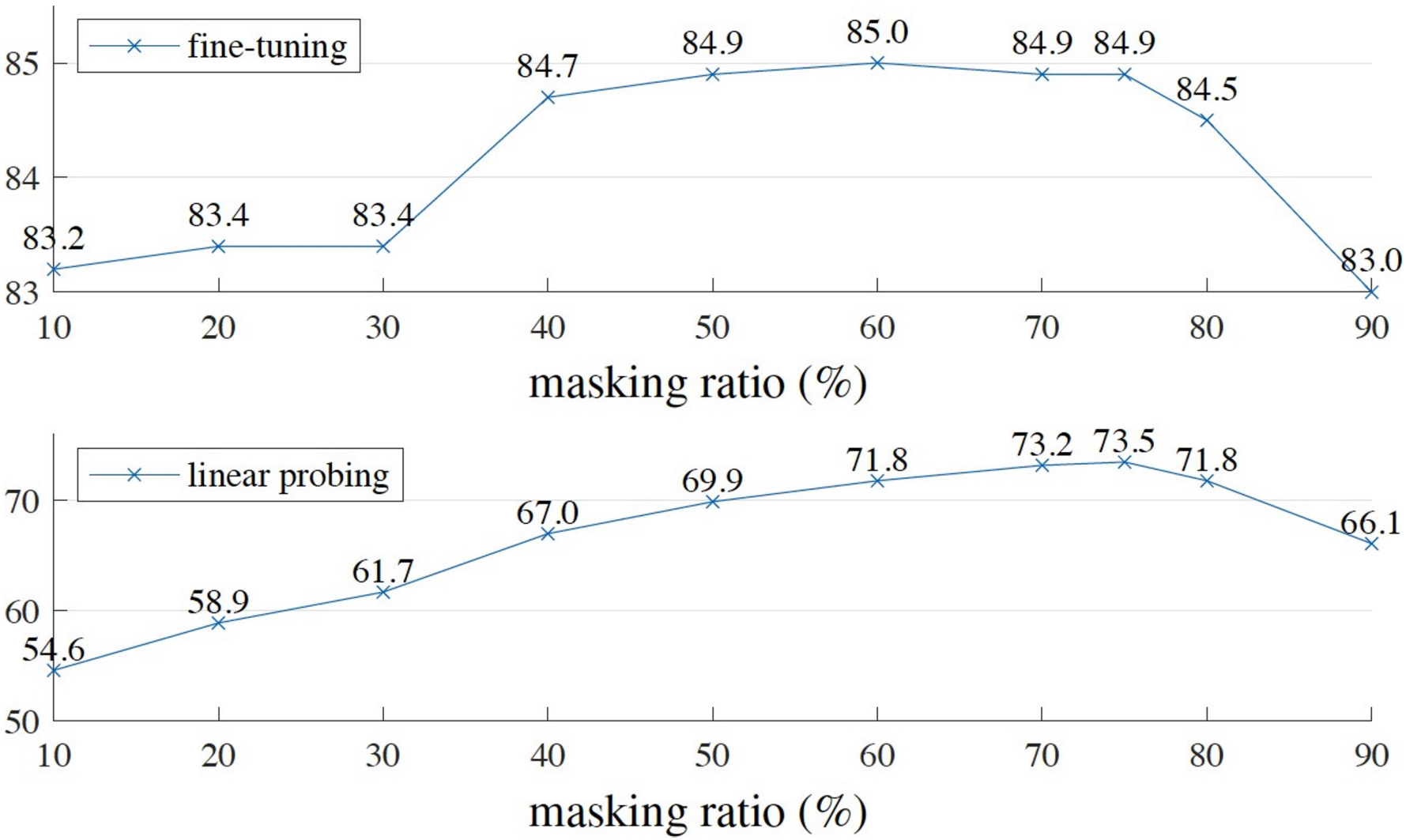
| blocks | ft | lin |
|---|---|---|
| 1 | 84.8 | 65.5 |
| 2 | **84.9** | 70.0 |
| 4 | **84.9** | 71.9 |
| 8 | **84.9** | **73.5** |
| 12 | 84.4 | 73.3 |

Decoder depth

| 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 |

Decoder width

# Analysis: Mask Ratio



# Analysis: Mask Token [M] in Encoder

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

- Encoder w/ [M] is default in BERT
- Big domain gap for linear probing
  - Pre-train sees 25% of the images only, while evaluation sees 100%

[Ramesh et al, ICML 2021] [Bao et al, ICLR 2022]

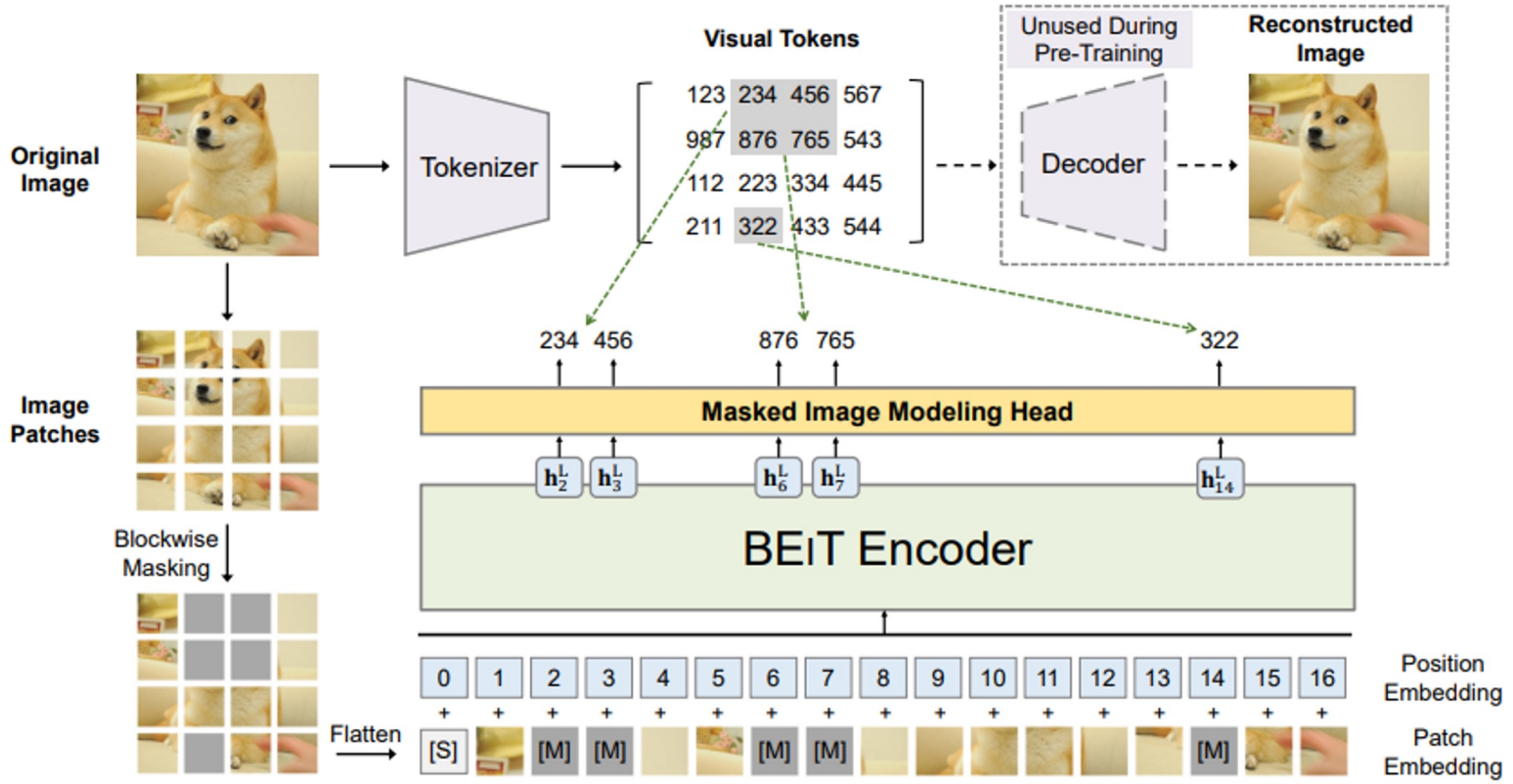
# Analysis: Reconstruction Target

| 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 |

- Pixels with normalization: per-patch -- minus *mean*, divide by *std*
- PCA: only low-frequency component is retained
- dVAE token: from DALLE, expensive to compute

[Ramesh et al, ICML 2021] [Bao et al, ICLR 2022]

# Analysis: Reconstruction Target




- dVAE token: from DALLE, expensive to compute (originally from Beit)

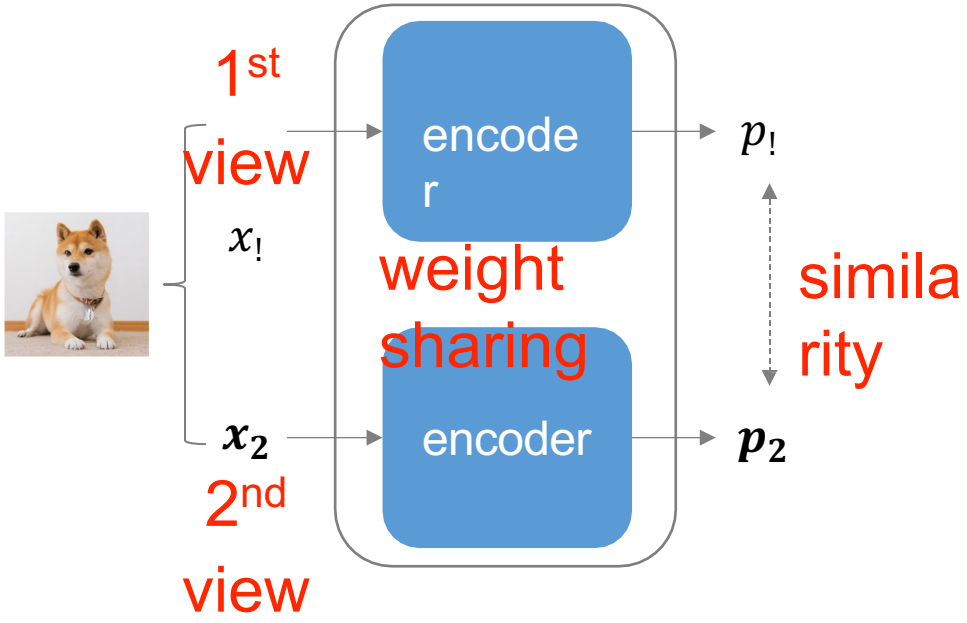
# Analysis: Augmentations

| 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 |

- MAE can work with minimal data augmentation

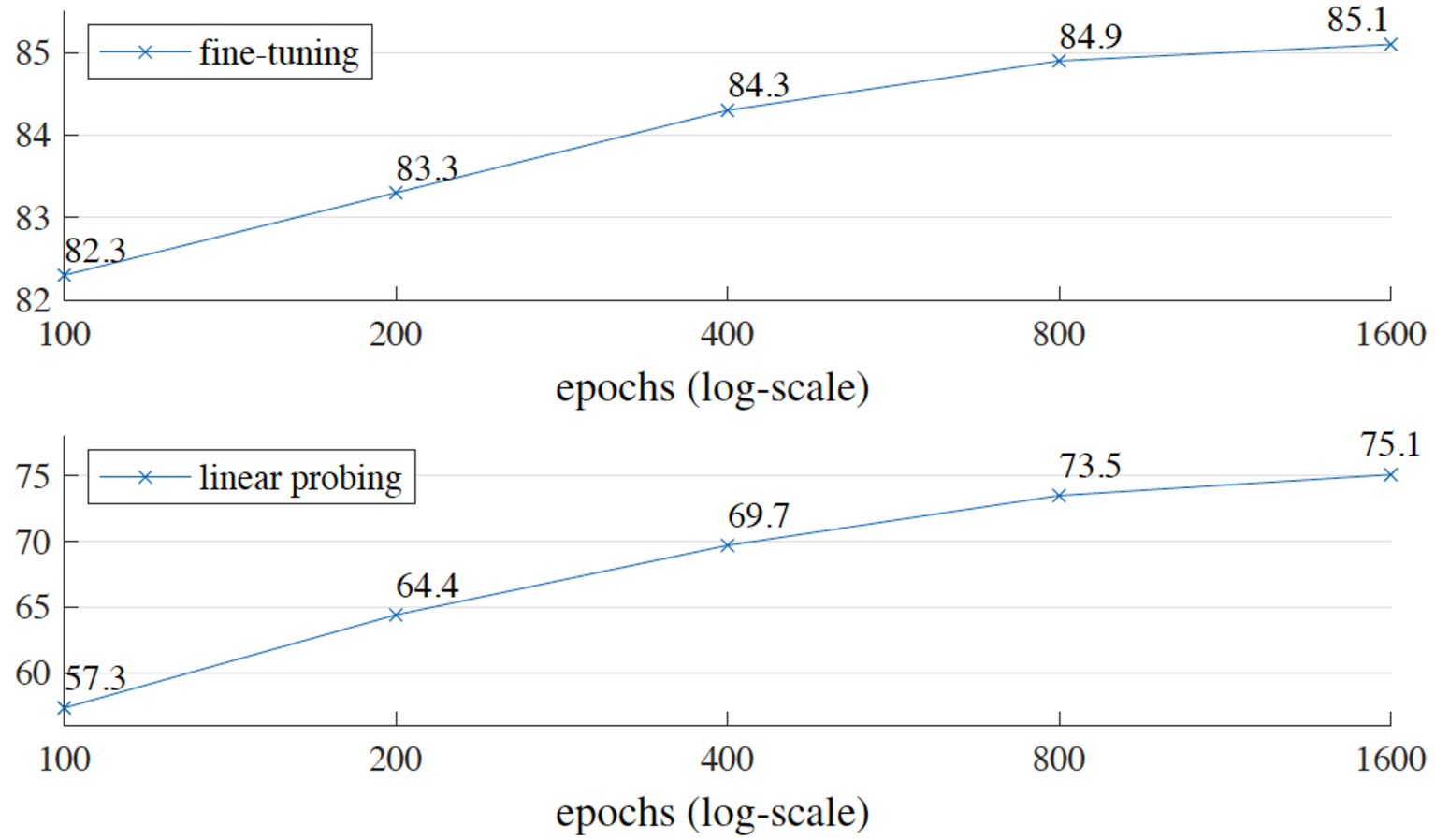
# Analysis: Augmentations

| 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 |



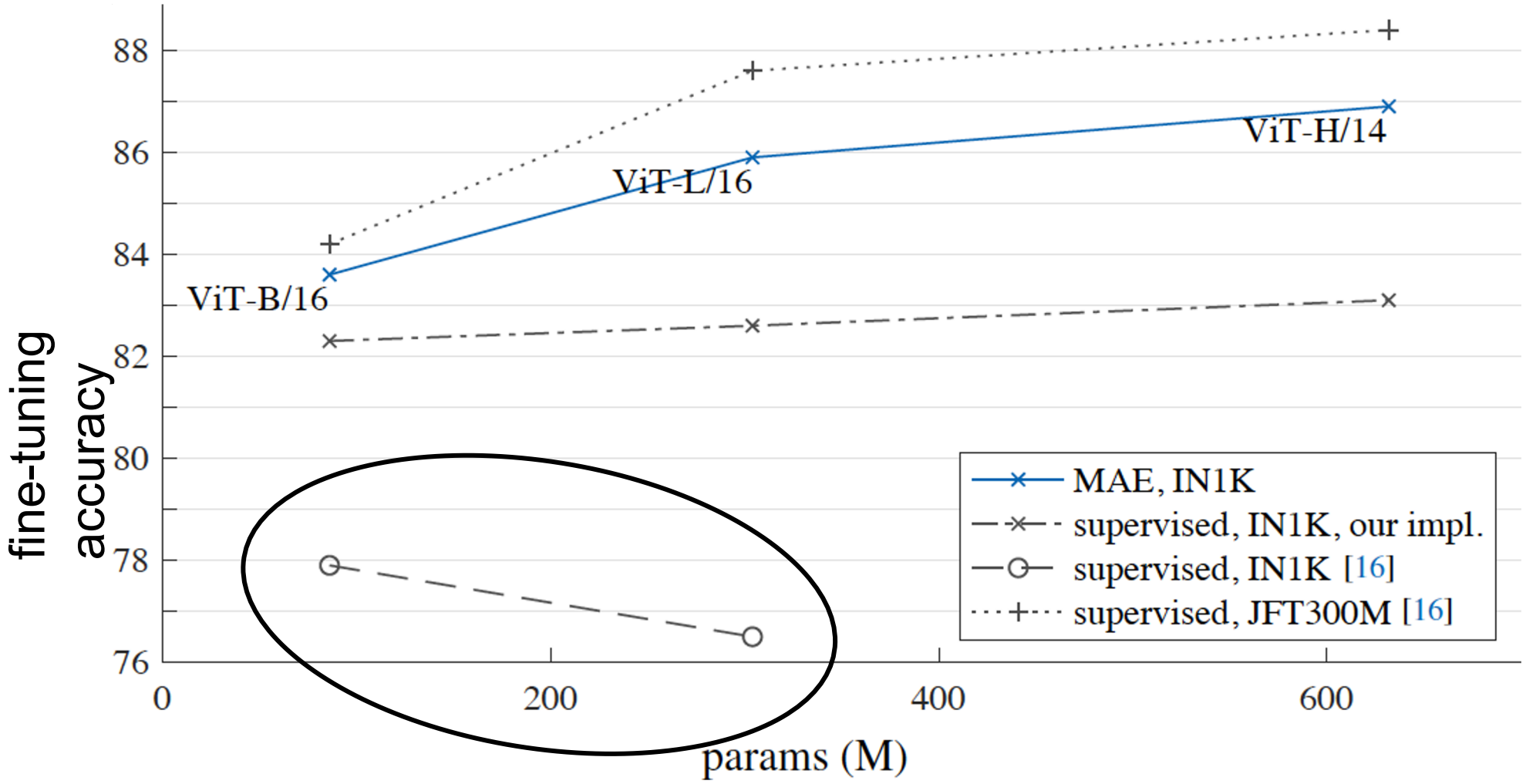

- MAE can work with minimal data augmentation
- SIMCLR augmentation is crucial (contrastive learning)

# Scalability: Longer Training

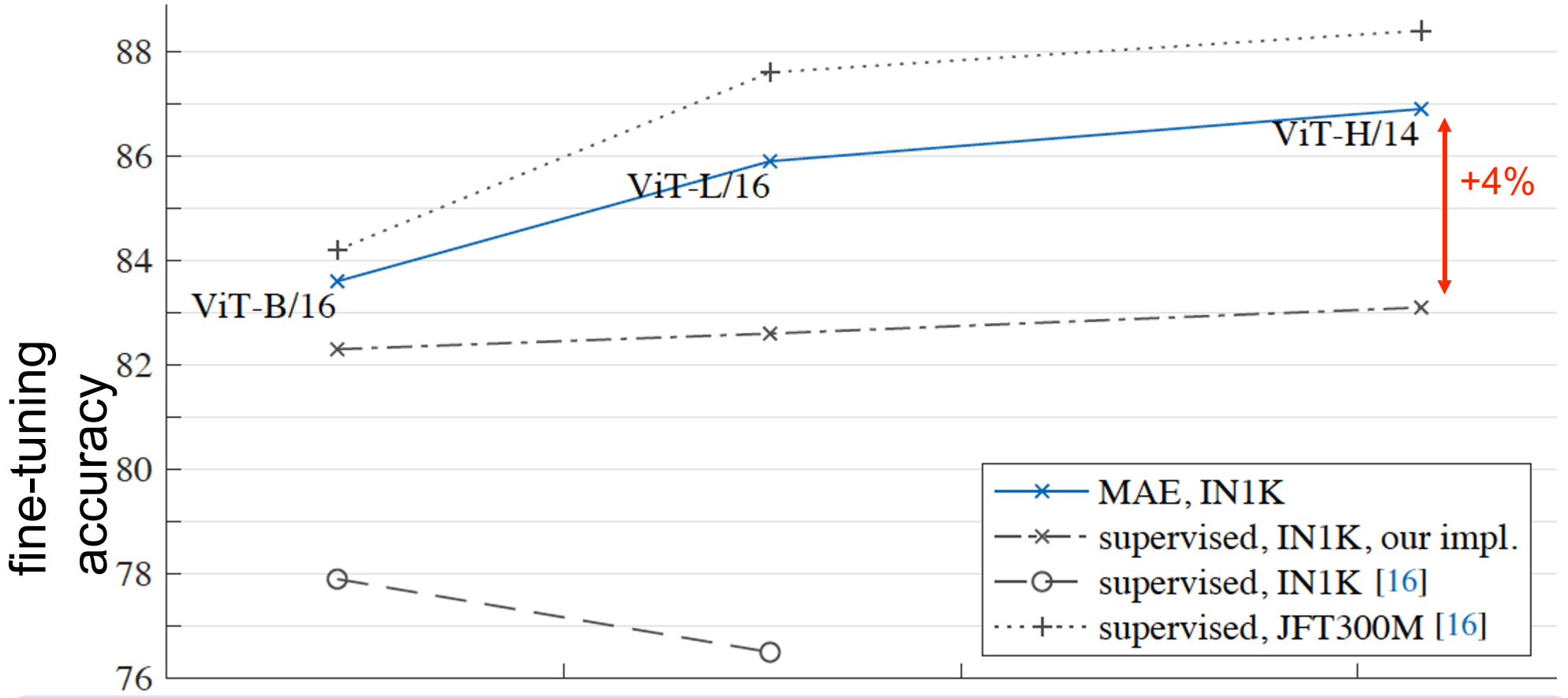



Wall-clock speed still efficient thanks to MAE design

# Scalability: Larger Models



# Scalability: Larger Models




new SOTA on ImageNet-1K (no extra data):

**87.8%**

# Scalability: Larger Models

| dataset | ViT-B | ViT-L | ViT-H | ViT-H$_{448}$ | prev best |
|---|---|---|---|---|---|
| iNat 2017 | 70.5 | 75.7 | 79.3 | **83.4** | 75.4 [50] |
| iNat 2018 | 75.4 | 80.1 | 83.0 | **86.8** | 81.2 [49] |
| iNat 2019 | 80.5 | 83.4 | 85.7 | **88.3** | 84.1 [49] |
| Places205 | 63.9 | 65.8 | 65.9 | **66.8** | 66.0 [19]† |
| Places365 | 57.9 | 59.4 | 59.8 | **60.3** | 58.0 [36]‡ |

new SOTA on 5 large-scale classification datasets

| dataset | ViT-B | ViT-L | ViT-H | ViT-H$_{448}$ | prev best |
|---|---|---|---|---|---|
| IN-Corruption ↓ [27] | 51.7 | 41.8 | **33.8** | 36.8 | 42.5 [32] |
| IN-Adversarial [28] | 35.9 | 57.1 | 68.2 | **76.7** | 35.8 [41] |
| IN-Rendition [26] | 48.3 | 59.9 | 64.4 | **66.5** | 48.7 [41] |
| IN-Sketch [60] | 34.5 | 45.3 | 49.6 | **50.9** | 36.0 [41] |

new SOTA on 4 ImageNet robust evaluations

# Scalability: Larger Models

| method | pre-train data | ViT-B | ViT-L |
|---|---|---|---|
| supervised | IN1K w/ labels | 47.9 | 49.3 |
| MoCo v3 | IN1K | 47.9 | 49.3 |
| BEiT | IN1K+DALLE | 49.8 | **53.3** |
| MAE | IN1K | **50.3** | **53.3** |

COCO detection: **+4.0%**

AP

| 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** |

ADE20K segmentation: **+3.7%**

mIOU

# Key Code Walkthrough

https://github.com/facebookresearch/mae/blob/main/models_mae.py

```
def patchify(self, imgs):
    """
    imgs: (N, 3, H, W)
    x: (N, L, patch_size**2 *3)
    """
    p = self.patch_embed.patch_size[0]
    assert imgs.shape[2] == imgs.shape[3] and imgs.shape[2] % p == 0

    h = w = imgs.shape[2] // p
    x = imgs.reshape(shape=(imgs.shape[0], 3, h, p, w, p))
    x = torch.einsum('nchpwq->nhwpqc', x)
    x = x.reshape(shape=(imgs.shape[0], h * w, p**2 * 3))
    return x
```

# Key Code Walkthrough

```python
def random_masking(self, x, mask_ratio):
    """
    Perform per-sample random masking by per-sample shuffling.
    Per-sample shuffling is done by argsort random noise.
    x: [N, L, D], sequence
    """
    N, L, D = x.shape  # batch, length, dim
    len_keep = int(L * (1 - mask_ratio))

    noise = torch.rand(N, L, device=x.device)  # noise in [0, 1]

    # sort noise for each sample
    ids_shuffle = torch.argsort(noise, dim=1)  # ascend: small is keep, large is remove
    ids_restore = torch.argsort(ids_shuffle, dim=1)

    # keep the first subset
    ids_keep = ids_shuffle[:, :len_keep]
    x_masked = torch.gather(x, dim=1, index=ids_keep.unsqueeze(-1).repeat(1, 1, D))

    # generate the binary mask: 0 is keep, 1 is remove
    mask = torch.ones([N, L], device=x.device)
    mask[:, :len_keep] = 0
    # unshuffle to get the binary mask
    mask = torch.gather(mask, dim=1, index=ids_restore)

    return x_masked, mask, ids_restore
```

```
>>> t = torch.tensor([[1, 2], [3, 4]])
>>> torch.gather(t, 1, torch.tensor([[0, 0], [1, 0]]))
tensor([[ 1,  1],
        [ 4,  3]])
```

**Equivalent to Sampling Without Replacement!**

# Key Code Walkthrough

```
def forward_encoder(self, x, mask_ratio):
    # embed patches
    x = self.patch_embed(x)

    # add pos embed w/o cls token
    x = x + self.pos_embed[:, 1:, :]

    # masking: length -> length * mask_ratio
    x, mask, ids_restore = self.random_masking(x, mask_ratio)

    # append cls token
    cls_token = self.cls_token + self.pos_embed[:, :1, :]
    cls_tokens = cls_token.expand(x.shape[0], -1, -1)
    x = torch.cat((cls_tokens, x), dim=1)

    # apply Transformer blocks
    for blk in self.blocks:
        x = blk(x)
    x = self.norm(x)

    return x, mask, ids_restore
```

# Key Code Walkthrough

```
def forward_decoder(self, x, ids_restore):
    # embed tokens
    x = self.decoder_embed(x)

    # append mask tokens to sequence
    mask_tokens = self.mask_token.repeat(x.shape[0], ids_restore.shape[1] + 1 - x.shape[1], 1)
    x_ = torch.cat([x[:, 1:, :], mask_tokens], dim=1)  # no cls token
    x_ = torch.gather(x_, dim=1, index=ids_restore.unsqueeze(-1).repeat(1, 1, x.shape[2]))  # unshuffle
    x = torch.cat([x[:, :1, :], x_], dim=1)  # append cls token

    # add pos embed
    x = x + self.decoder_pos_embed

    # apply Transformer blocks
    for blk in self.decoder_blocks:
        x = blk(x)
    x = self.decoder_norm(x)

    # predictor projection
    x = self.decoder_pred(x)

    # remove cls token
    x = x[:, 1:, :]

    return x
```

# Key Code Walkthrough

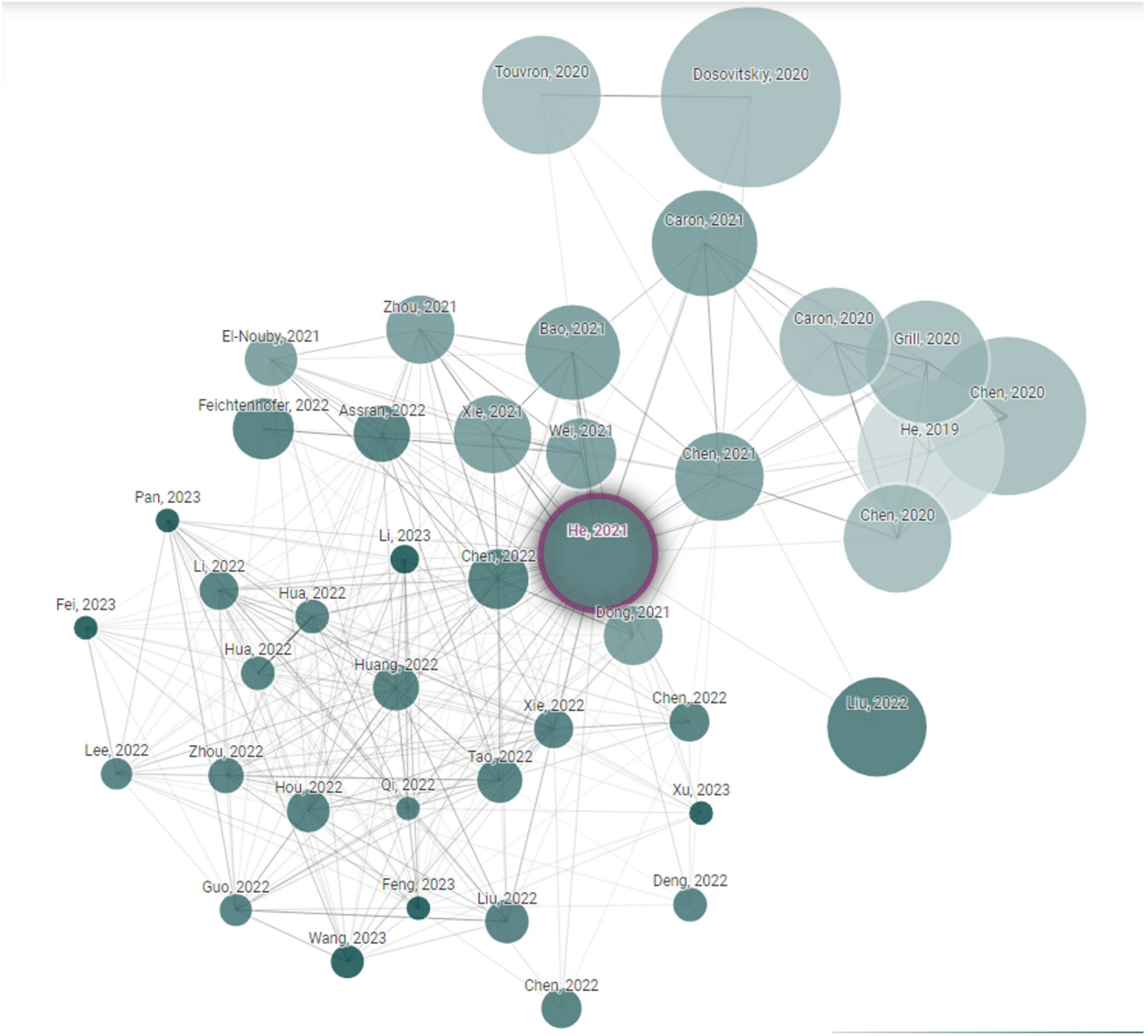
```
def forward(self, imgs, mask_ratio=0.75):
    latent, mask, ids_restore = self.forward_encoder(imgs, mask_ratio)
    pred = self.forward_decoder(latent, ids_restore)  # [N, L, p*p*3]
    loss = self.forward_loss(imgs, pred, mask)
    return loss, pred, mask
```

# Growth of MAEs

Diffusion Models as Masked Autoencoders

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

…



# MAE for all modalities?

Video-MAE

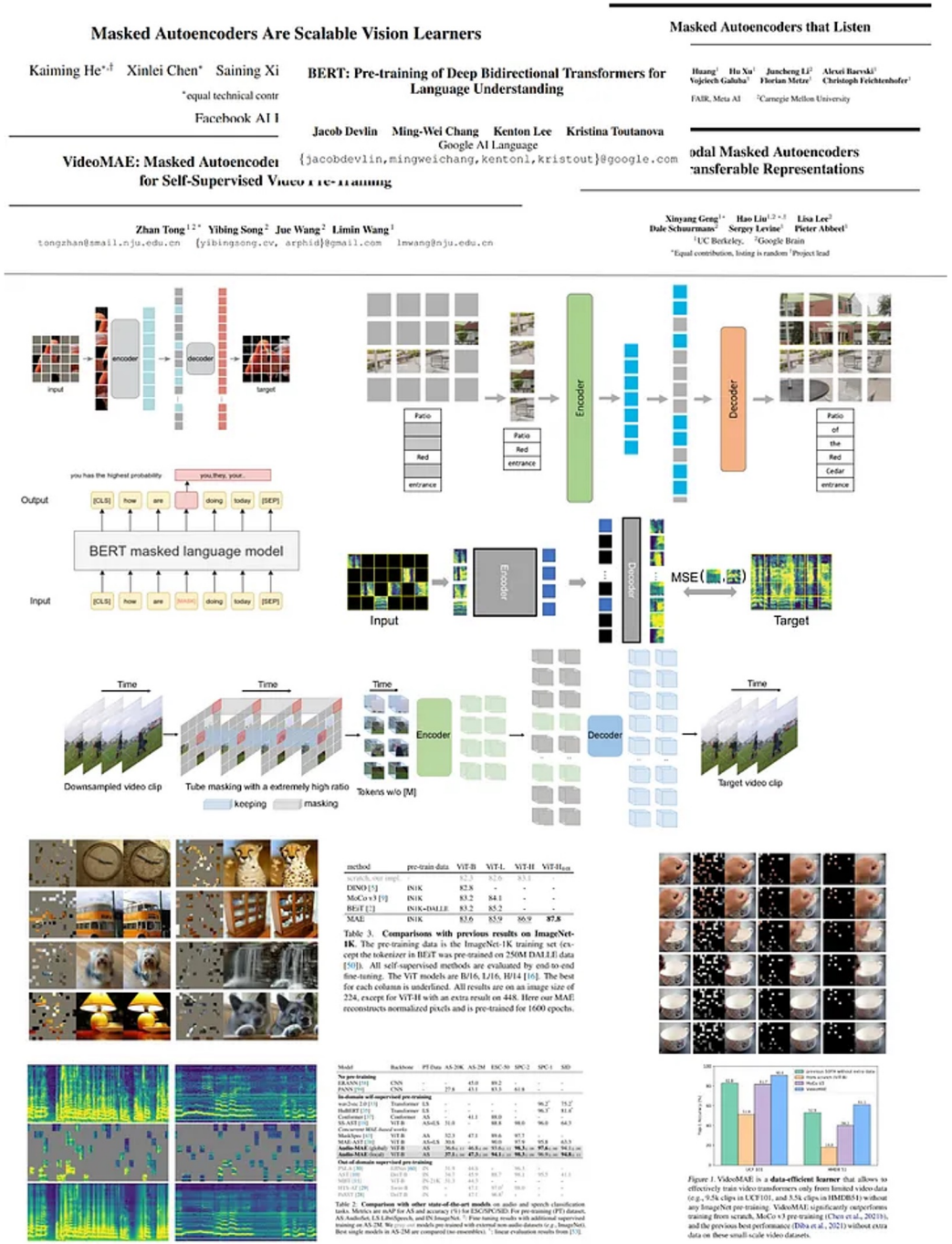
MAE itself (images)

MAE that Listen (Audio)

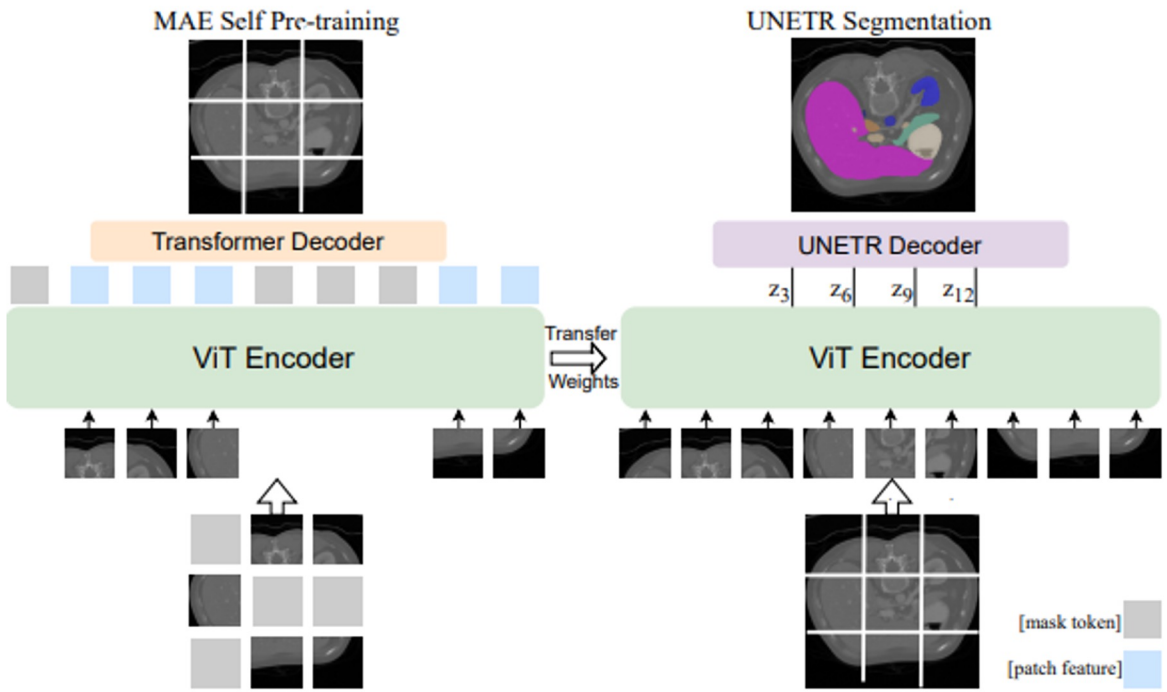
Bert (MAE for text)

Medical images? (2d, 3d)

Multimodal?



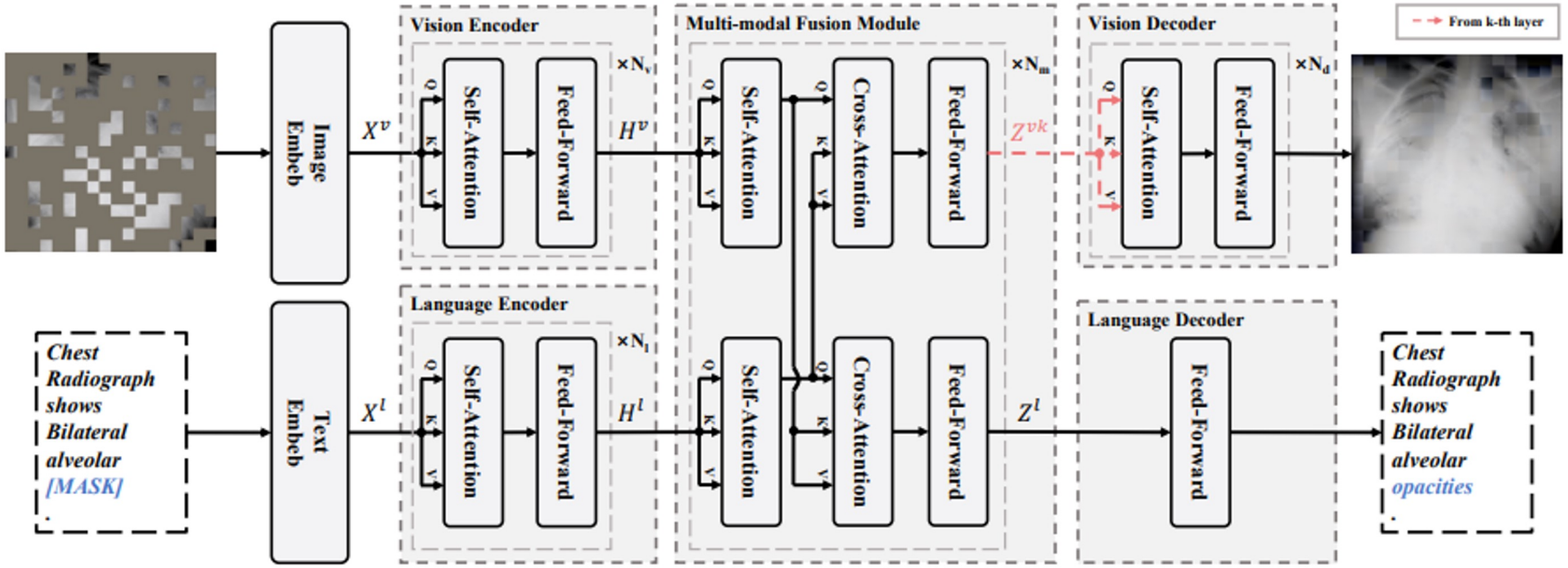
# What it means for us?




**Self Pre-training with Masked Autoencoders for Medical Image Classification and Segmentation, L Zhou et al 2022**

78.8% to 83.5% Dice score on BTCV Segmentation Challenge dataset

# What it means for us?




**Multi-Modal Masked Autoencoders for Medical Vision-and-Language Pre-Training, Z Chen et al 2022**

Quick QA:

Does MAE **Encoder** involves mask tokens in pretraining?

Quick QA:

Does MAE **Encoder** involves mask tokens in pretraining?

No

Quick QA:

Does MAE reconstruct each masked patch merely inferred from its adjacent neighbor patches?

Quick QA:

Does MAE reconstruct each masked patch merely inferred from its adjacent neighbor patches?

No