Advanced Deep Learning

(2022. 11. 09)

BEiT: BERT Pre-Training of Image Transformers (2021)

통계데이터사이언스학과 통합과정 5학기 이승한

BEiT (2021)

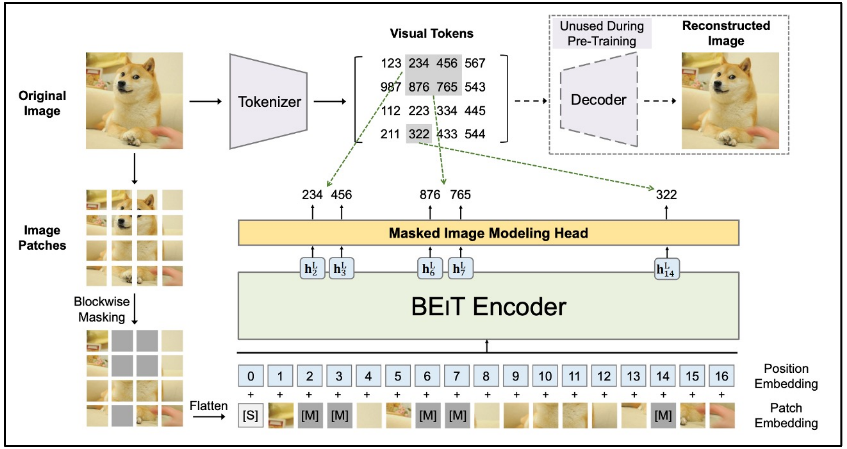
BEiT (2021)

1. **Introduction**
2. BEiT : Bidirectional Encoder representation from Image Transformers
    a. Two views of Representations
    b. Model Architecture
    c. Objective Function
3. Experiments

1. Introduction

BEiT = Bidirectional Encoder representation from Image Transformers

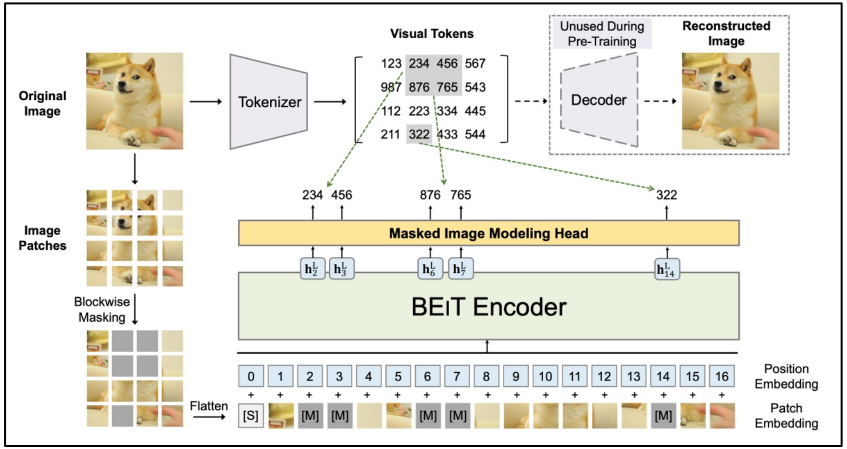
- **Self-supervised** vision representation model
- **MIM (Masked Image Modeling)** to pretrain ViT
  - MIM = recovering the masked image patches, based on encoding vectors
- Each Image has 2 views
  - view 1) **image patches** (ex. 16x16 pixels)
  - view 2) **visual tokens** (ex. discrete tokens)



1. Introduction

BEiT = Bidirectional Encoder representation from Image Transformers

- Procedure
  - step 1) tokenize image into **visual tokens**
  - step 2) randomly **mask** some image patches & fed them into backbone Transformer
- Objective : **recover the original visual tokens, based on corrupted image patches**
- Fine tune model params on downstream tasks



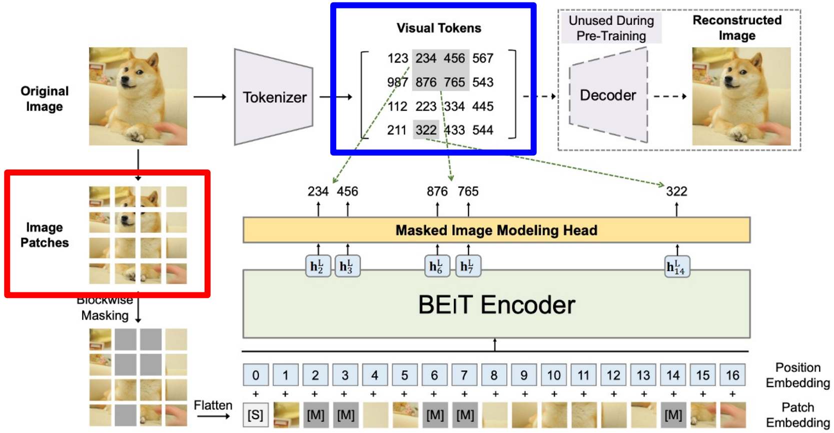
BEiT (2021)

BEiT (2021)

2. BEiT : Bidirectional Encoder representation from Image Transformers

a) 2 views of Representations

- (1) **image patch** (= serve as INPUT)
- (2) **visual tokens** (= serve as OUTPUT)



2. BEiT : Bidirectional Encoder representation from Image Transformers

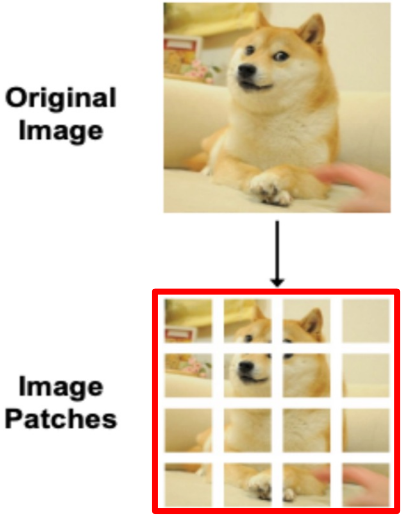
a) 2 views of Representations

- (1) **image patch** (= serve as INPUT)
- (2) visual tokens (= serve as OUTPUT)

image : split into a sequence of patches

- (from) image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$
- (to) $N = HW/P^2$ patches $\boldsymbol{x}^p \in \mathbb{R}^{N \times (P^2 C)}$

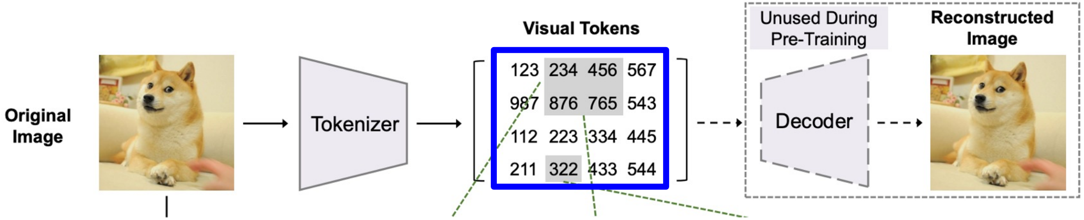
(P, P) = resolution of each patch



2. BEiT : Bidirectional Encoder representation from Image Transformers

a) 2 views of Representations

- (1) image patch (= serve as INPUT)
- (2) **visual tokens** (= serve as OUTPUT)



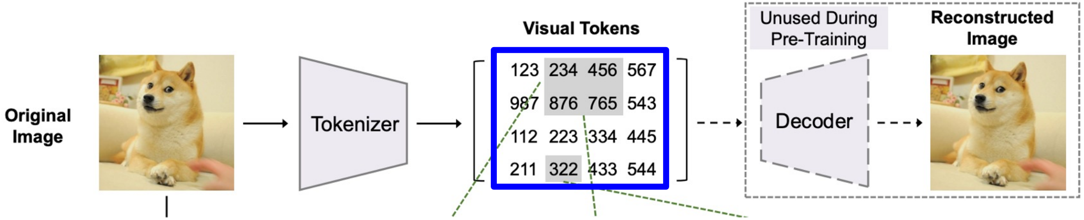
represent the image as a sequence of discrete tokens

(= obtained by an "image tokenizer")

2. BEiT : Bidirectional Encoder representation from Image Transformers

a) 2 views of Representations

- (1) image patch (= serve as INPUT)
- (2) **visual tokens** (= serve as OUTPUT)



represent the image as a sequence of discrete tokens

(= obtained by an "image tokenizer")

Tokenize ...

- (from) image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$
- (to) $\boldsymbol{z} = [z_1, \ldots, z_N] \in \mathcal{V}^{h \times w}$
  - where the vocabulary $\mathcal{V} = \{1, \ldots, |\mathcal{V}|\}$ contains discrete token indices

Details :

- # of visual tokens = # of image patches
- vocab size : $|\mathcal{V}| = 8192$

2. BEiT : Bidirectional Encoder representation from Image Transformers

a) 2 views of Representations

- (1) image patch (= serve as INPUT)
- (2) **visual tokens** (= serve as OUTPUT)

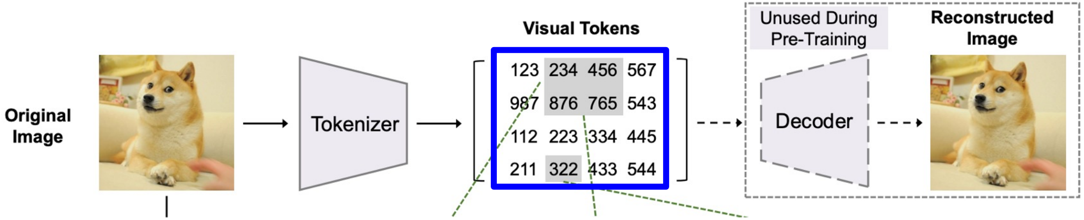
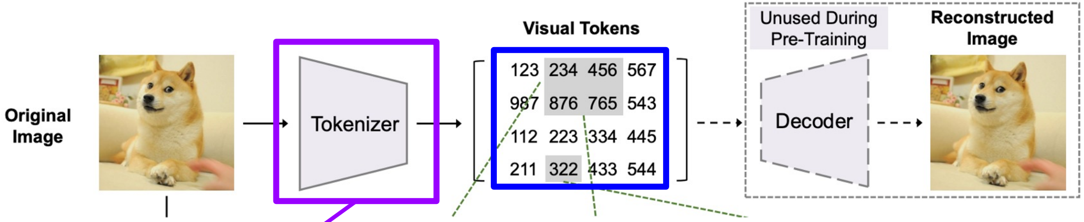


Image Tokenizer

- learned by discrete variational autoencoder (dVAE)
- two modules (during visual token learning)

2. BEiT : Bidirectional Encoder representation from Image Transformers



a) 2 views of Representations

- (1) image patch (= serve as INPUT)
- (2) **visual tokens** (= serve as OUTPUT)

Image Tokenizer

- learned by discrete variational autoencoder (dVAE)
- two modules (during visual token learning)

(1) tokenizer : $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$

- maps image pixels $\boldsymbol{x}$ into discrete tokens $\boldsymbol{z}$ (according to codebook (=vocab))
- uniform prior

2. BEiT : Bidirectional Encoder representation from Image Transformers

a) 2 views of Representations

- (1) image patch (= serve as INPUT)
- (2) **visual tokens** (= serve as OUTPUT)

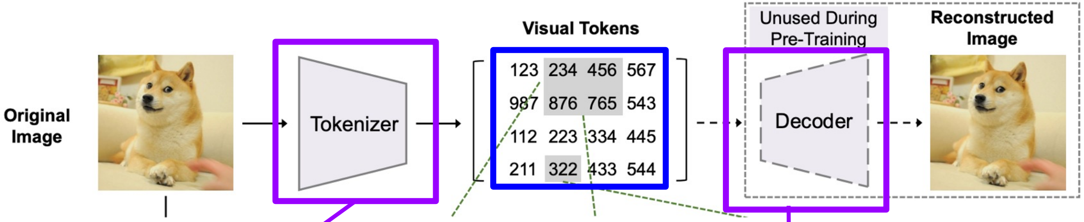


Image Tokenizer

- learned by discrete variational autoencoder (dVAE)
- two modules (during visual token learning)

(1) tokenizer : $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$

- maps image pixels $\boldsymbol{x}$ into discrete tokens $\boldsymbol{z}$
  (according to codebook (=vocab))
- uniform prior

(2) decoder : $p_\psi(\boldsymbol{x} \mid \boldsymbol{z})$

- reconstructs the mage $\boldsymbol{x}$ based on the visual tokens $\boldsymbol{z}$
- Reconstruction objective : $\mathbb{E}_{\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\psi(\boldsymbol{x} \mid \boldsymbol{z})\right]$
  (discrete? use **Gumbel Softmax Trick**)

2. BEiT : Bidirectional Encoder representation from Image Transformers

a) 2 views of Representations

- (1) image patch (= serve as INPUT)
- (2) **visual tokens** (= serve as OUTPUT)

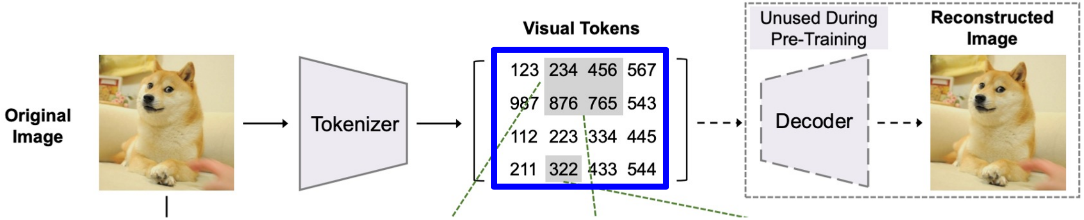
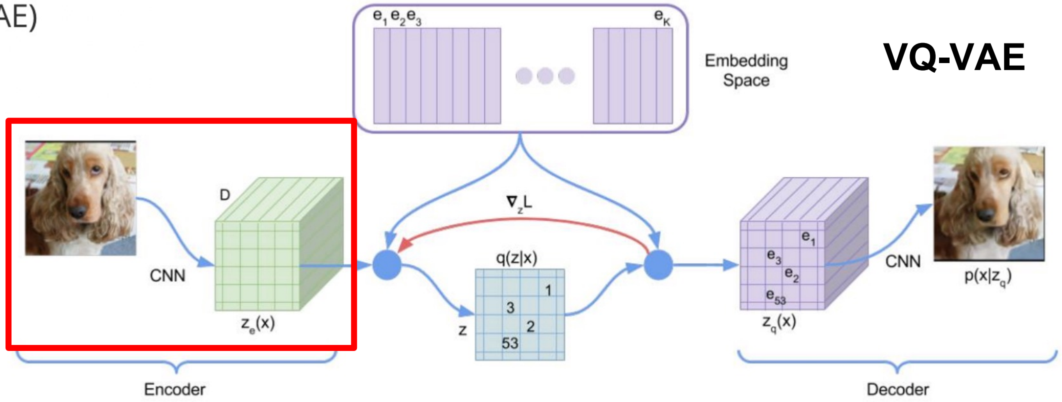


Image Tokenizer

- learned by discrete variational autoencoder (dVAE)
- two modules (during visual token learning)

VQ-VAE



2. BEiT : Bidirectional Encoder representation from Image Transformers

a) 2 views of Representations

- (1) image patch (= serve as INPUT)
- (2) **visual tokens** (= serve as OUTPUT)

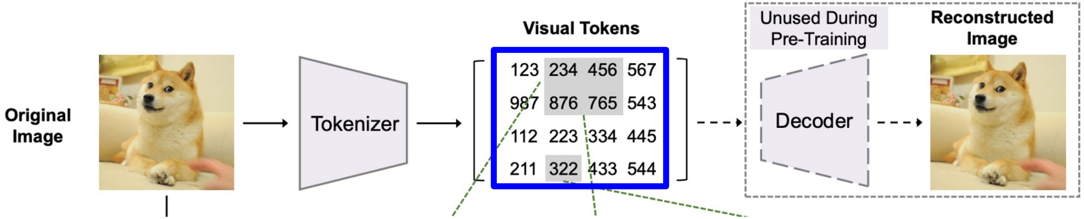
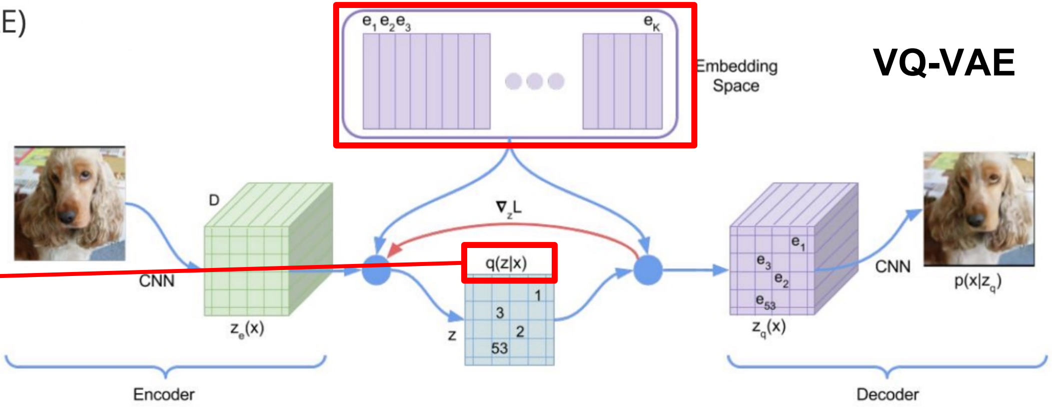


Image Tokenizer

- learned by discrete variational autoencoder (dVAE)
- two modules (during visual token learning)

Codebook containing discrete vectors

VQ-VAE



$$q(z=k|x) = \begin{cases} 1 & \text{for } \mathrm{k} = \mathrm{argmin}_j \|z_e(x) - e_j\|_2 \\ 0 & \text{otherwise} \end{cases}$$

2. BEiT : Bidirectional Encoder representation from Image Transformers

a) 2 views of Representations

- (1) image patch (= serve as INPUT)
- (2) **visual tokens** (= serve as OUTPUT)

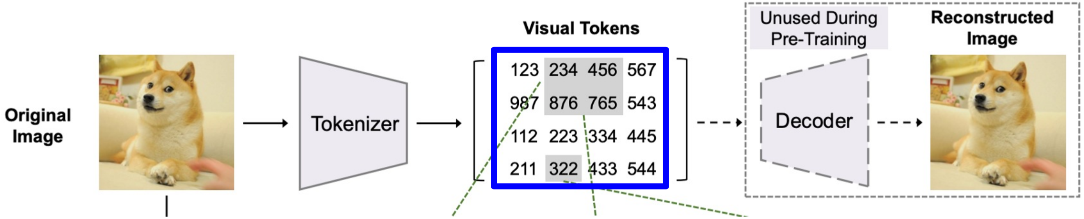
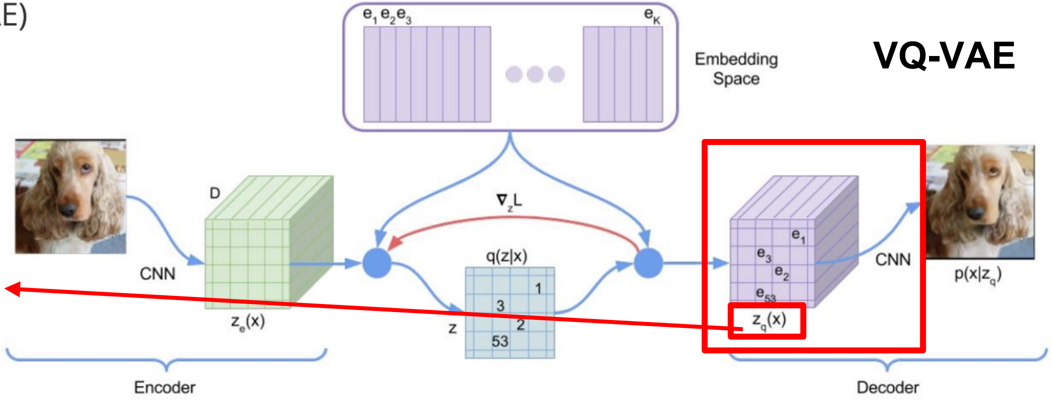


Image Tokenizer

- learned by discrete variational autoencoder (dVAE)
- two modules (during visual token learning)

$$\boxed{z_q(x) = e_k, \quad \text{where} \quad k = \operatorname{argmin}_j \|z_e(x) - e_j\|_2}$$



VQ-VAE

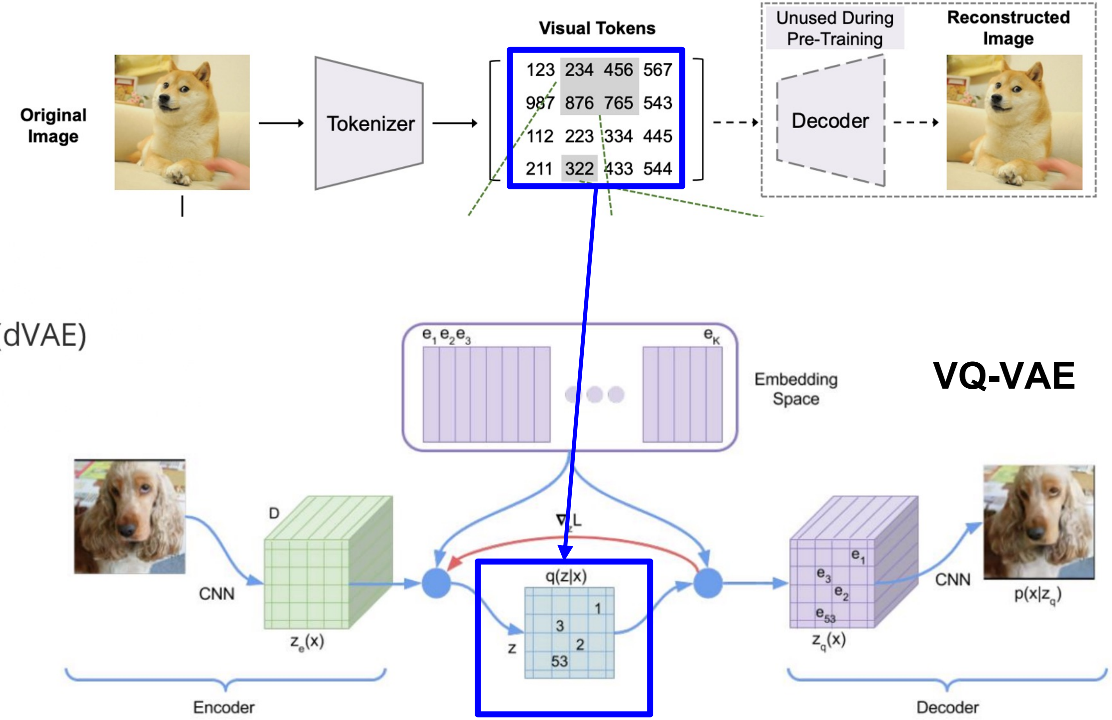
2. BEiT : Bidirectional Encoder representation from Image Transformers

a) 2 views of Representations

- (1) image patch (= serve as INPUT)
- (2) **visual tokens** (= serve as OUTPUT)

Image Tokenizer

- learned by discrete variational autoencoder (dVAE)
- two modules (during visual token learning)



2. BEiT : Bidirectional Encoder representation from Image Transformers

a) 2 views of Representations

- (1) image patch (= serve as INPUT)
- (2) **visual tokens** (= serve as OUTPUT)

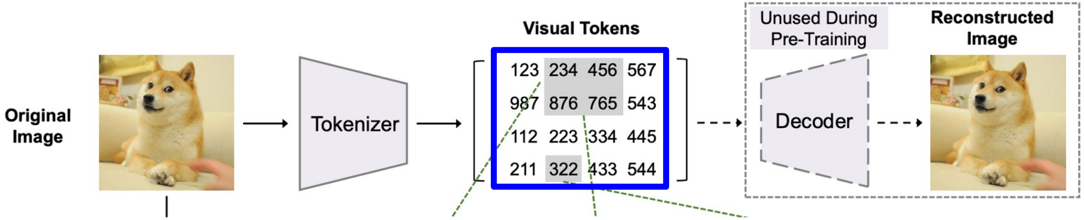


Image Tokenizer

- learned by discrete variational autoencoder (dVAE)
- two modules (during visual token learning)

Back-prop with **DISCRETE** variable?

→ use **"Gumbel Softmax Relaxation"** !

Train : "Softmax"

$$y_i = \frac{e^{\frac{g_i+\log\left(q\left(e_j|x\right)\right)}{\tau}}}{\sum_{j=1}^{k} e^{\frac{g_j+\log\left(q\left(e_j|x\right)\right)}{\tau}}} \rightarrow z = \sum_{j=1}^{k} y_j e_j$$

Test : "Argmax"

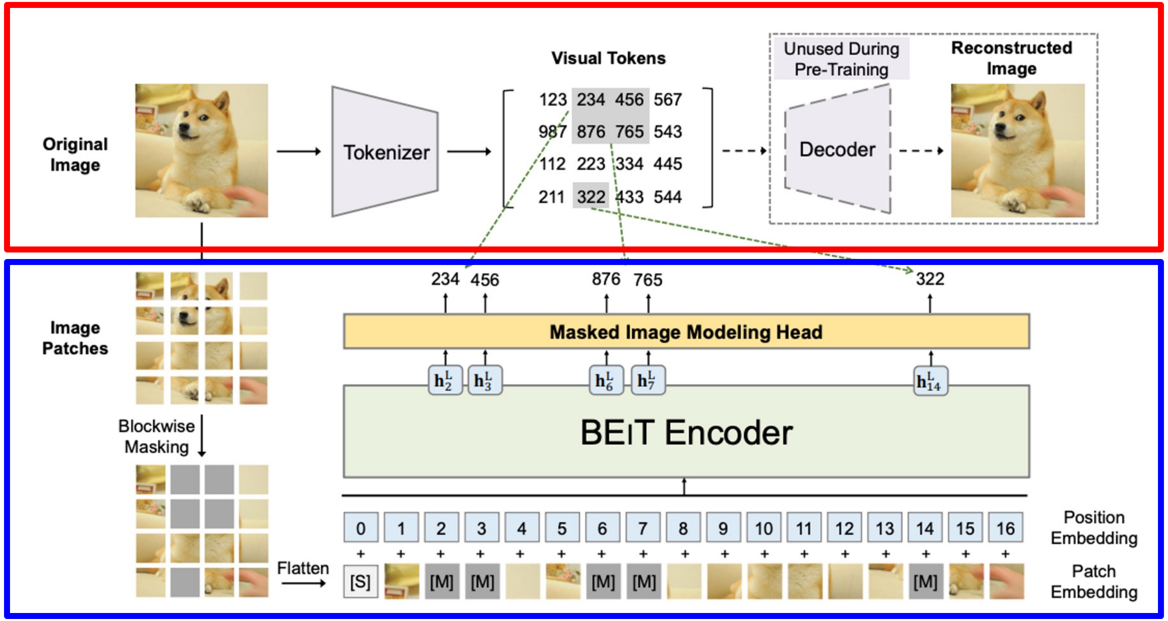
$$z = \text{codebook}[\arg\max_i [g_i + log(q(e_i|x))]]$$

BEiT (2021)

1. Introduction
2. **BEiT : Bidirectional Encoder representation from Image Transformers**
    a. Two views of Representations
    b. **Model Architecture**
    c. Objective Function
3. Experiments

2. BEiT : Bidirectional Encoder representation from Image Transformers

b) Model Architecture

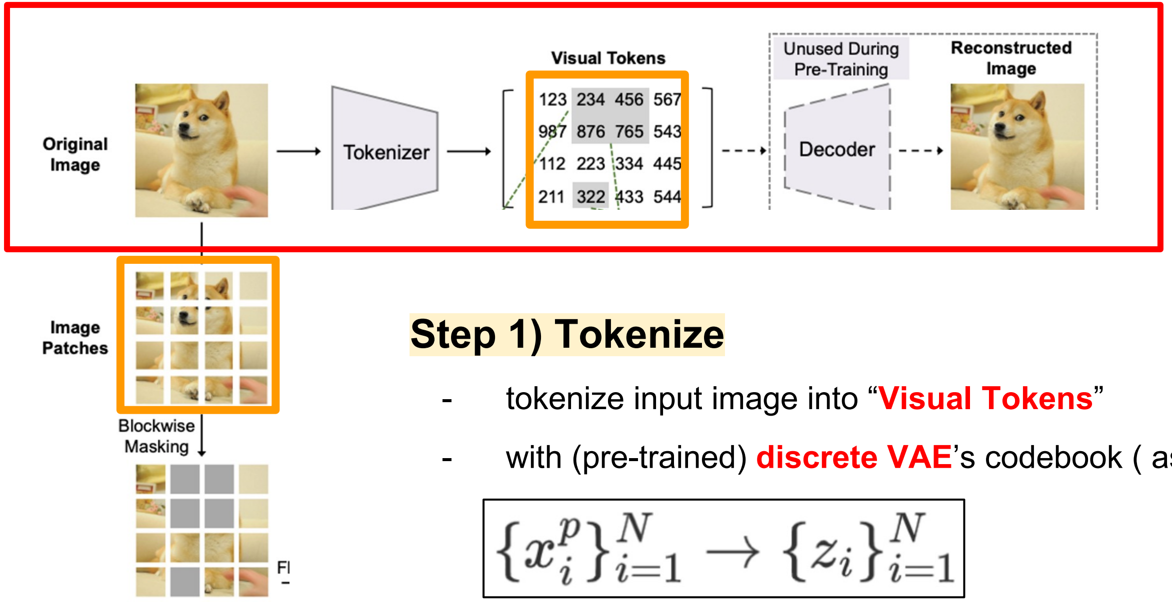


Discrete VAE

Masked Image Model

2. BEiT : Bidirectional Encoder representation from Image Transformers

b) Model Architecture



Discrete VAE

Step 1) Tokenize

- tokenize input image into "**Visual Tokens**"
- with (pre-trained) **discrete VAE**'s codebook (as DALL-E)

$$\{x_i^p\}_{i=1}^N \rightarrow \{z_i\}_{i=1}^N$$

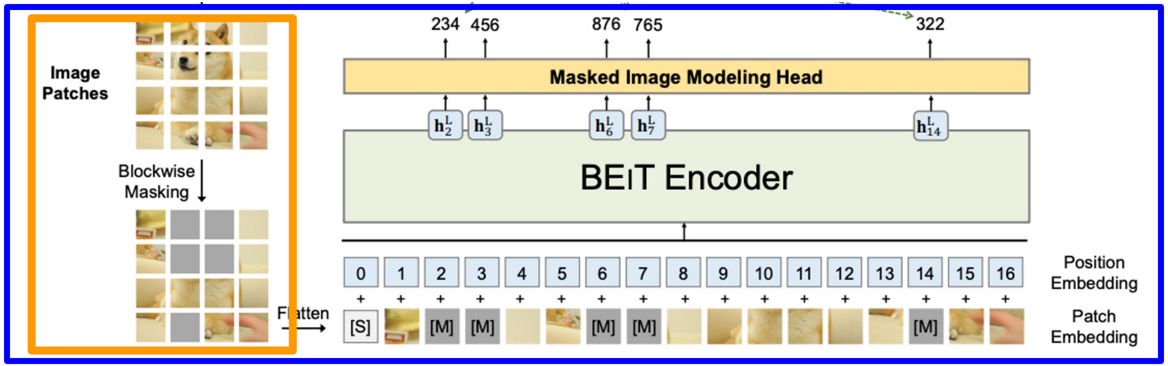
2. BEiT : Bidirectional Encoder representation from Image Transformers

b) Model Architecture

Step 2) Random Masking

- Blockwise Masking
- random masking 40% of patches

$$\mathcal{M} \in \{1, \ldots, N\}^{0.4N}$$
$$x^{\mathcal{M}} = \{x_i^p : i \notin \mathcal{M}\}_{i=1}^{N} \bigcup \{e_{[M]} : i \in \mathcal{M}\}_{i=1}^{N}$$



Masked Image Model

2. BEiT : Bidirectional Encoder representation from Image Transformers

b) Model Architecture

Step 2) Random Masking

- Blockwise Masking
- random masking 40% of patches

$$\mathcal{M} \in \{1, \ldots, N\}^{0.4N}$$
$$x^{\mathcal{M}} = \{x_i^p : i \notin \mathcal{M}\}_{i=1}^{N} \bigcup \{e_{[M]} : i \in \mathcal{M}\}_{i=1}^{N}$$

Algorithm 1 Blockwise Masking

Input: $N(= h \times w)$ image patches
Output: Masked positions $\mathcal{M}$
$\mathcal{M} \leftarrow \{\}$
repeat
  $s \leftarrow \text{Rand}(16, 0.4N - |\mathcal{M}|)$ ▷ *Block size*
  $r \leftarrow \text{Rand}(0.3, \frac{1}{0.3})$ ▷ *Aspect ratio of block*
  $a \leftarrow \sqrt{s \cdot r}; b \leftarrow \sqrt{s/r}$
  $t \leftarrow \text{Rand}(0, h - a)\ ; l \leftarrow \text{Rand}(0, w - b)$
  $\mathcal{M} \leftarrow \mathcal{M} \bigcup \{(i, j) : i \in [t, t + a), j \in [l, l + b)\}$
until $|\mathcal{M}| > 0.4N$ ▷ *Masking ratio is 40%*
return $\mathcal{M}$

(1) Sample “mask size” (at least 16)

(2) Sample “aspect ratio” (0.3:1 ~ 1:0.3)

(3) Compute height / width

(4) Compute Top.Left point of masking block

(5) Augment with Masking information

(6) if Masking ratio > 40%, STOP

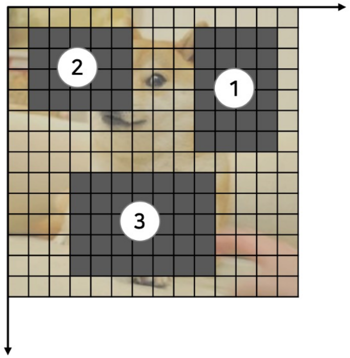
2. BEiT : Bidirectional Encoder representation from Image Transformers

b) Model Architecture

Step 2) Random Masking

- Blockwise Masking
- random masking 40% of patches

$$\mathcal{M} \in \{1, \ldots, N\}^{0.4N}$$
$$x^{\mathcal{M}} = \{x_i^p : i \notin \mathcal{M}\}_{i=1}^{N} \bigcup \{e_{[M]} : i \in \mathcal{M}\}_{i=1}^{N}$$



Mask ex 1) $s = 24, r = 1.5, a = 6, b = 4, |\mathcal{M}| = 24$

Mask ex 2) $s = 20, r = 0.8, a = 4, b = 5, |\mathcal{M}| = 44$

Mask ex 3) $s = 35, r = 0.7, a = 5, b = 7, |\mathcal{M}| = 79$

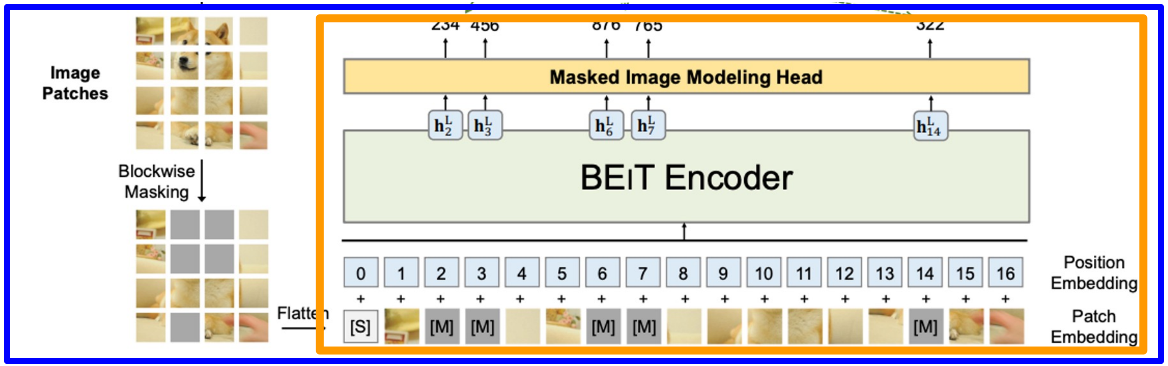
2. BEiT : Bidirectional Encoder representation from Image Transformers

b) Model Architecture

Step 3) Predict Visual Token

- predict the "masked image input"
- using ViT + Softmax CLS

$$p_{\mathrm{MIM}}\left(z' \mid x^{\mathcal{M}}\right) = \operatorname{softmax}_{z'}\left(\boldsymbol{W}_c \boldsymbol{h}_i^L + \boldsymbol{b}_c\right)$$



Masked Image Model

2. BEiT : Bidirectional Encoder representation from Image Transformers

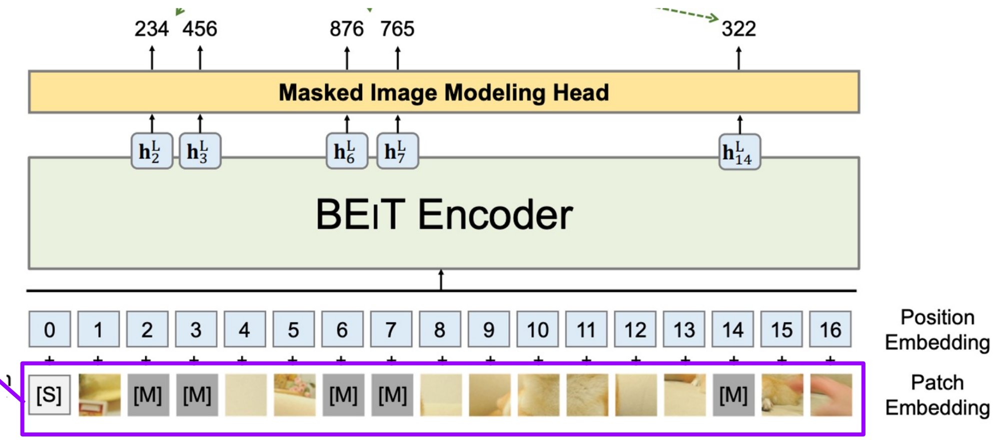
b) Model Architecture

Step 3) Predict Visual Token

a) Input (of Transformer) :

- sequence of image patches $\{x_i^p\}_{i=1}^N$

  (N = number of patches)



2. BEiT : Bidirectional Encoder representation from Image Transformers

b) Model Architecture

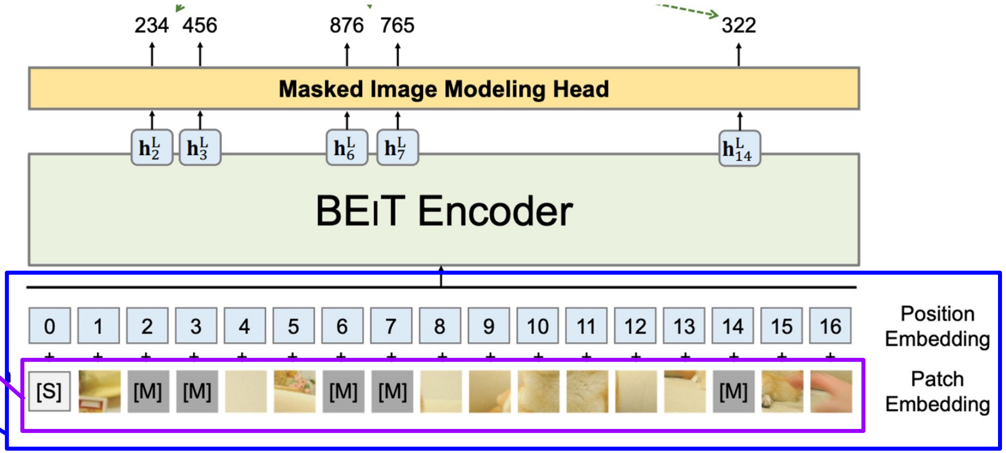
Step 3) Predict Visual Token

a) Input (of Transformer) :

- sequence of image patches $\{\boldsymbol{x}_i^p\}_{i=1}^N$
  (N = number of patches)

b) Embeddings :

- $\{\boldsymbol{x}_i^p\}_{i=1}^N$ are linearly projected to $\boldsymbol{E}\boldsymbol{x}_i^p$
  - where $\boldsymbol{E} \in \mathbb{R}^{(P^2C)\times D}$
- add learnable 1d positional embeddings : $\boldsymbol{E}_{\text{pos}} \in \mathbb{R}^{N\times D}$
- final output embedding : $\boldsymbol{H}_0 = [\boldsymbol{e}_{[\text{S}]}, \boldsymbol{E}\boldsymbol{x}_i^p, \ldots, \boldsymbol{E}\boldsymbol{x}_N^p] + \boldsymbol{E}_{\text{pos}}$



2. BEiT : Bidirectional Encoder representation from Image Transformers

b) Model Architecture

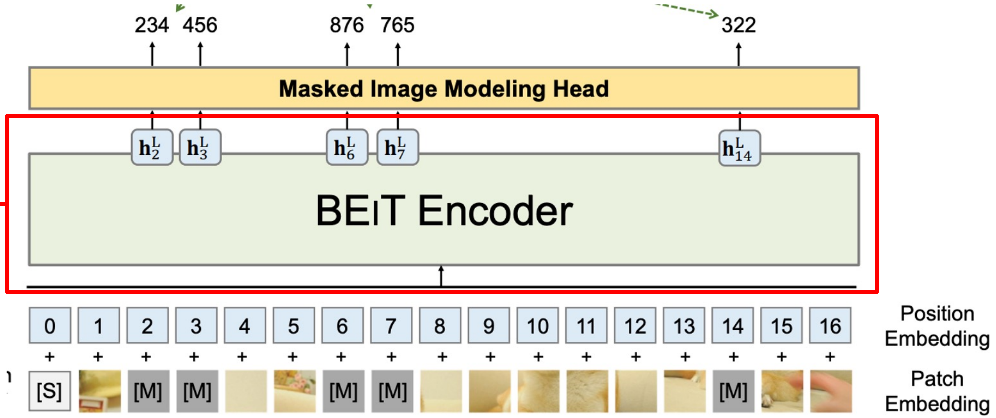
Step 3) Predict Visual Token

c) Encoder :

- contains L layers of Transformer blocks
  - $\boldsymbol{H}^l = \mathrm{Transformer}(\boldsymbol{H}^{l-1})$.
- **output vectors of the last layer :** $\boldsymbol{H}^L = \left[\boldsymbol{h}^L_{[s]}, \boldsymbol{h}^L_1, \ldots, \boldsymbol{h}^L_N\right]$

  ($\boldsymbol{h}^L_i$: vector of the i-th patch)

  → encoded representations for the image patches



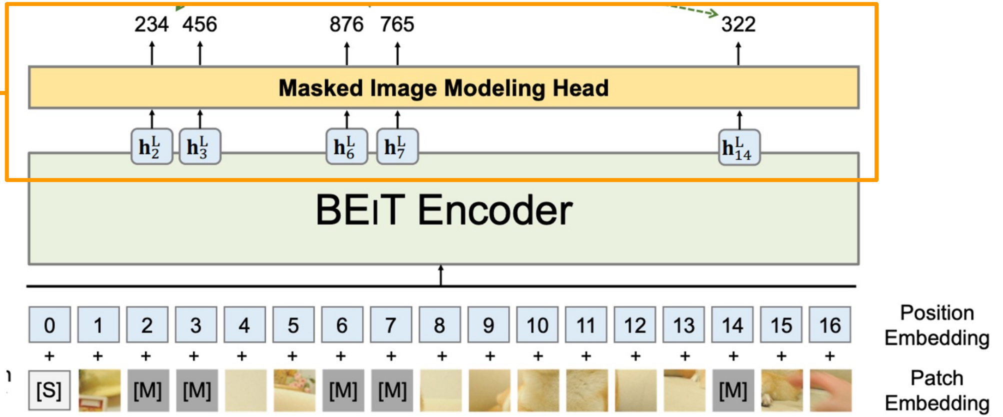
2. BEiT : Bidirectional Encoder representation from Image Transformers

b) Model Architecture

Step 3) Predict Visual Token

d) Classification (with softmax classifier)

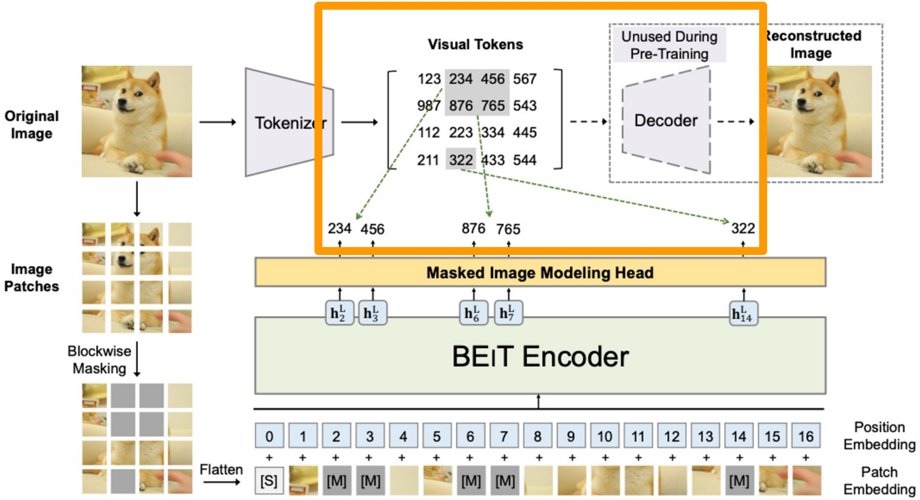
- classify for each masked position $\{\boldsymbol{h}_i^L : i \in \mathcal{M}\}_{i=1}^N$
- $p_{\text{MIM}}\left(z' \mid x^{\mathcal{M}}\right) = \text{softmax}_{z'}\left(\boldsymbol{W}_c \boldsymbol{h}_i^L + \boldsymbol{b}_c\right).$
  - $x^{\mathcal{M}}$: corrupted image
  - $\boldsymbol{W}_c \in \mathbb{R}^{|\mathcal{V}| \times D}$ and $\boldsymbol{b}_c \in \mathbb{R}^{|\mathcal{V}|}$



BEiT (2021)

2. BEiT : Bidirectional Encoder representation from Image Transformers

c) Objective Function



Maximize the log-likelihood of the correct **visual tokens**, given the **corrupted image**

$$\max \sum_{x \in \mathcal{D}} \mathbb{E}_{\mathcal{M}} \left[\sum_{i \in \mathcal{M}} \log p_{\mathrm{MIM}} \left(z_i \mid x^{\mathcal{M}} \right) \right]$$

BEiT (2021)

1. Introduction
2. BEiT : Bidirectional Encoder representation from Image Transformers
    a. Two views of Representations
    b. Model Architecture
    c. Objective Function
3. **Experiments**

3. Experiments

1) (Classification) Top-1 Accuracy

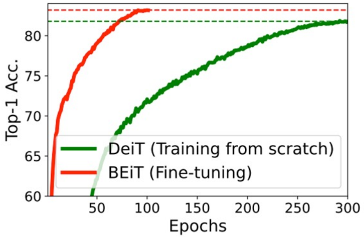


Table 2: Convergence curves of training DeiT from scratch and fine-tuning BEIT on ImageNet-1K.

Models	**Model Size**	**Resolution**	**ImageNet**
Training from scratch (i.e., random initialization)			
ViT_{384}-B [DBK+20]	86M	384^2	77.9
ViT_{384}-L [DBK+20]	307M	384^2	76.5
DeiT-B [TCD+20]	86M	224^2	81.8
DeiT_{384}-B [TCD+20]	86M	384^2	83.1
Supervised Pre-Training on ImageNet-22K (using labeled data)			
ViT_{384}-B [DBK+20]	86M	384^2	84.0
ViT_{384}-L [DBK+20]	307M	384^2	85.2
Self-Supervised Pre-Training on ImageNet-1K (without labeled data)			
iGPT-1.36B† [CRC+20]	1.36B	224^2	66.5
ViT_{384}-B-JFT300M‡ [DBK+20]	86M	384^2	79.9
MoCo v3-B [CXH21]	86M	224^2	83.2
MoCo v3-L [CXH21]	307M	224^2	84.1
DINO-B [CTM+21]	86M	224^2	82.8
BEIT-B (ours)	86M	224^2	83.2
BEIT_{384}-B (ours)	86M	384^2	84.6
BEIT-L (ours)	307M	224^2	85.2
BEIT_{384}-L (ours)	307M	384^2	**86.3**

Table 1: Top-1 accuracy on ImageNet-1K. We evaluate base- ("-B") and large-size ("-L") models at resolutions 224×224 and 384×384. †: iGPT-1.36B contains 1.36 billion parameters, while others are base-size models. ‡: ViT_{384}-B-JFT300M is pretrained with the "masked patch prediction" task on Google's in-house 300M images, while others use ImageNet.

3. Experiments

2) (Semantic Segmentation)

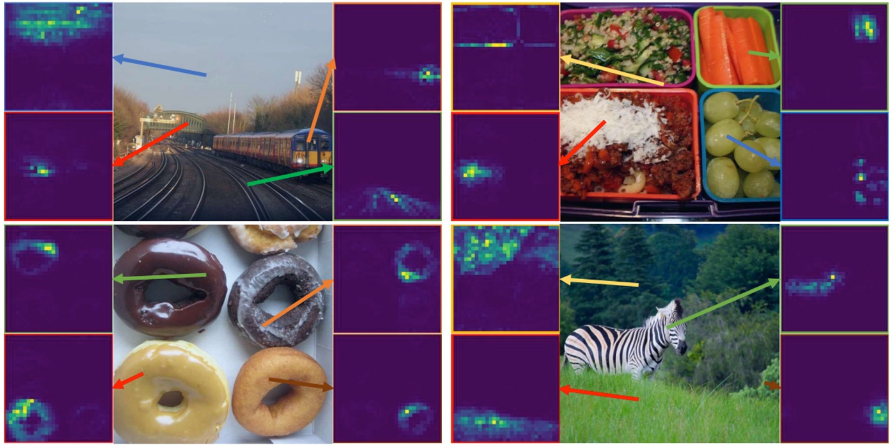


Figure 2: Self-attention map for different reference points. The self-attention mechanism in BEiT is able to separate objects, although self-supervised pre-training does not use manual annotations.

Models	ADE20K
Supervised Pre-Training on ImageNet	45.3
DINO [CTM+21]	44.1
BEiT (ours)	45.6
BEiT + Intermediate Fine-Tuning (ours)	**47.7**

Table 3: Results of semantic segmentation on ADE20K. We use SETR-PUP [ZLZ+20] as the task layer and report results of single-scale inference.

Thank You !