

# トランسفォーマーによる画像や音声の処理

本谷 秀堅

名古屋工業大学

# Vision Transformer

# ViT: Vision Transformer

画像を AI にとって都合の良いベクトル（トークン）で表現したい

- 文章：各単語をトークンで表現
- 画像は？

# 復習：画像データ

## 画像データ：画素値の集積



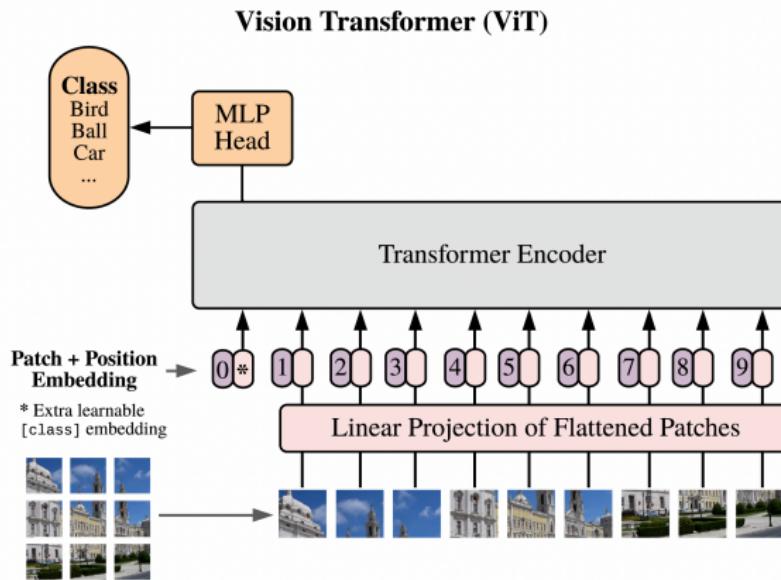
157	153	174	168	150	152	129	151	172	163	155	156
156	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	54	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	197	251	237	299	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	103	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	236	73	1	81	47	0	4	217	264	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	12	96	218

157	153	174	168	150	152	129	151	172	161	155	156
156	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	54	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	103	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	4	217	265	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

- カラー画像であれば各ピクセルは3次元ベクトル (RGB: Red/Green/Blue) で表現されている
  - 画素ごとにトークンを求めるのは非現実的（画素数多過ぎ）

# 画像をトランスフォーマーで

- 画像を「パッチ」に分割
- それぞれのパッチを「単語」のように扱う
  - $P \times P$  の正方形 (多くの場合  $P = 16$ )
  - 元画像が  $H \times W$  だったときパッチは  $N = HW/P^2$  枚



# トークン化の意味

パッチは、そのままでもベクトルで表現できる

- 元画像が  $C$  チャンネルだとする
  - 白黒画像 :  $C = 1$
  - カラー画像 :  $C = 3$

パッチのサイズを  $P \times P$  とする。画像中の  $i$  番目のパッチの  $j$  番目の画素値を  $x_{ij} \in \mathbb{R}^C$  で表すとき、次式でパッチ画像をベクトル表現できる：

$$\mathbf{x}_i = [\mathbf{x}_{i1}^\top, \mathbf{x}_{i2}^\top, \dots, \mathbf{x}_{iJ}^\top]^\top \in \mathbb{R}^{P^2 C}$$

ただし、 $J = P^2$ 。例えば  $C = 3$  のとき、 $\mathbf{x}_i = [r_i, g_i, b_i]^\top \in \mathbb{R}^3$  など。

- 重要：このままでは AI に向き（次ページ）

# トークン化の意味

例題  $C = 1, P = 16$  とする。黒地に上から  $i$  番目の行に幅 1 ピクセルの白の水平線が描かれたパッチ画像を  $x_i$  で表す。黒地部分の画素値は 0 で、白地部分の画素値は 1 とする。例えば  $x_2$  は次式のような  $P^2 = 256$  次元の二値ベクトルである。

$$\boldsymbol{x}_2 = \underbrace{[0, 0, \dots, 0]}_{16}, \underbrace{1, 1, \dots, 1}_{16}, \underbrace{0, 0, \dots, 0}_{16 \times 14}]^\top$$

$\boldsymbol{x}_i^\top \boldsymbol{x}_j$  ( $i \neq j$ ) を求めよ。

この例では  $\boldsymbol{x}_i$  ( $i = 1, 2, \dots, P$ ) は互いに直交している。それぞれのベクトルの向きは、水平線の位置を一切反映していない。

- パッチ画像の画素値を並べたベクトル表現,  $\boldsymbol{x}$ , は「パターンの類似度」評価に適していない

# トランسفォーマーによる画像処理

トランسفォーマーへの入力準備

① 画像を  $P \times P$  のパッチに切り分ける

②  $i$  番目のパッチ:  $\mathbf{x}_i \in \mathbb{R}^{P^2 C}$  ( $i = 1, 2, \dots, N$ )

- $P = 16, C = 3$  のときは 768 次元

③ 線形変換する:  $\left(\tilde{\mathbf{z}}_i^{(0)}\right)^\top = \mathbf{x}_i^\top \mathbf{E}, \quad \mathbf{E} \in \mathbb{R}^{P^2 C \times D}$

- 例えば  $D = 768$

④ 各パッチの位置ベクトルを足す:  $\mathbf{z}_i^{(0)} = \tilde{\mathbf{z}}_i^{(0)} + \mathbf{r}_i$

⑤ 追加でひとつ Class トークンを追加する:  $\mathbf{z}_0^{(0)}$

⑥ それぞれをレイヤ正規化する

# トランسفォーマーによる画像処理

## トランسفォーマー層での処理

- ① クエリ、キー、バリューそれぞれのベクトルへと変換

$$\mathbf{q}_i^\top = z_i^{(0)} \mathbf{W}^{(q)}, \quad \mathbf{k}_i^\top = z_i^{(0)} \mathbf{W}^{(k)}, \quad \mathbf{v}_i^\top = z_i^{(0)} \mathbf{W}^{(v)}$$

- ② 各ヘッドで注意（アテンション）の強さの計算

$$a_{ij} = \text{SM}\left(\frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{D^k}}\right)$$

- ③ 各ヘッドでトークンの更新

$$\mathbf{h}_i = \sum_j a_{ij} \mathbf{v}_j$$

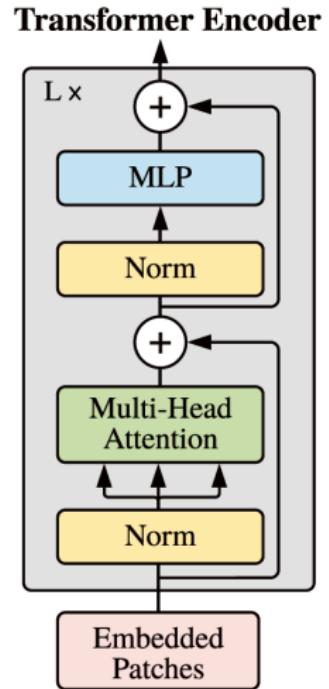
- ④ マルチヘッドの統合

$$\tilde{\mathbf{z}}_i^\top = [\mathbf{h}_i^\top(1), \mathbf{h}_i^\top(2), \dots, \mathbf{h}_i^\top(H)] \mathbf{W}^{(o)}$$

- ⑤ 残差接続： $\mathbf{z}'_i = \tilde{\mathbf{z}}_i + \mathbf{z}_i^{(0)}$

# トランسفォーマーによる画像処理

- ① レイヤ正規化:  $\text{LN}(z')$
- ② MLP を適用 :  $z^{(1)} = \text{MLP}(\text{LN}(z')) + z'$
- ③ 出力 :  $y = \text{LN}(z^{(1)})$



# トランスフォーマーによる画像識別器の構築

Class トークンを使って入力画像が何の画像かを識別する

- 画像中の対象を強く注意することを期待する



- CNN の受容野を思い出す
- ヘッドごとに注意する領域が異なる / 層の深さにも依存
- 必ずしも常に対象のみに注意するわけではなさそう

[1]: Dosovitskiy, AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE, 2021

# CNN vs トランスフォーマー

画像に適しているアーキテクチャは CNN ではなかったか

- CNN: アーキテクチャが画像の有する性質を反映していた
  - 畳み込みフィルタによるピクセル間の近接関係を反映
  - 並進同変性
- トランスフォーマー
  - 画像処理に適した構造を持たない

# CNN vs トランスフォーマー

CNN と Transformer：ネットワークの規模を概ね揃えて、同じデータベースを使って学習して性能比較

文献 [1] によると：

- 学習データ数が少ないとき：トランスフォーマーは過学習して汎化性能は劣る
- 学習データ数が大規模なとき：トランスフォーマーが CNN を凌駕

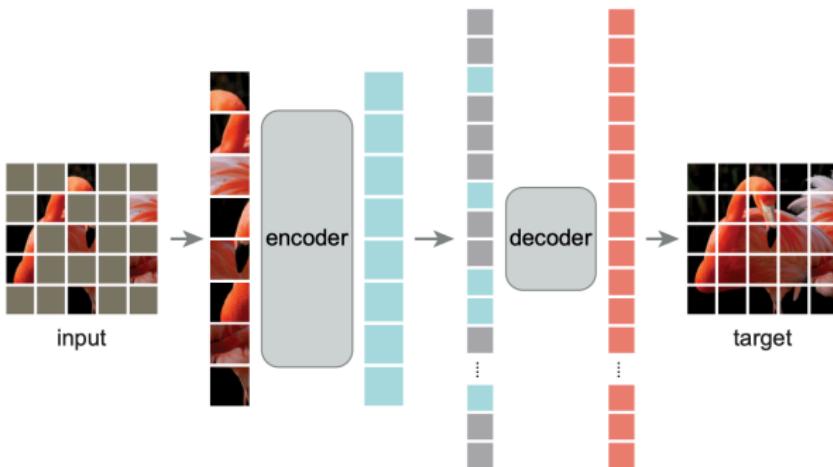
「大規模」とは…

- JFT-300M: 3億3千枚の画像・1万8千クラスのラベル
- ImageNet-21k: 1400万枚の画像・2万1千クラスのラベル
- ImageNet: 130万枚の画像・1千クラスのラベル

[1]: Dosovitskiy, AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE, 2021

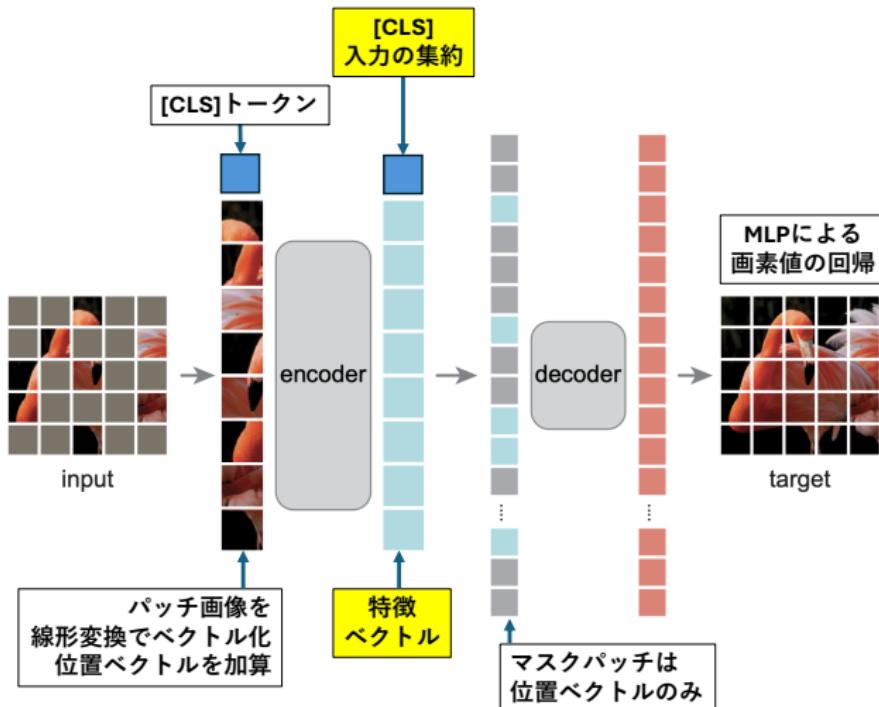
# MAE: Masked Auto Encoder

- マスクされた部分の画素値を予測
- マスクされた部分はエンコード対象から除外（軽量）
- エンコーダより小さいデコーダ

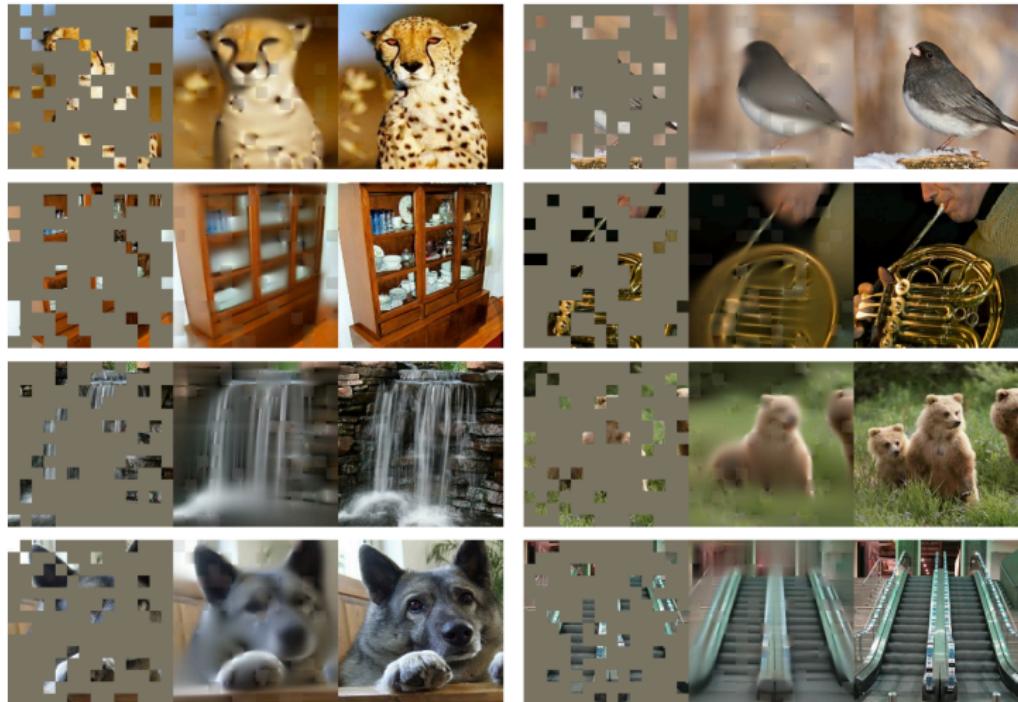


# MAE: Masked Auto Encoder

- 学習後に使うのはエンコーダのみ



# MAE: Masked Auto Encoder



# MAE: Masked Auto Encoder

文脈的に「正しい」予測



# MAE: Masked Auto Encoder

[CLS] トークンを使って線形識別 (ImageNet-1K)

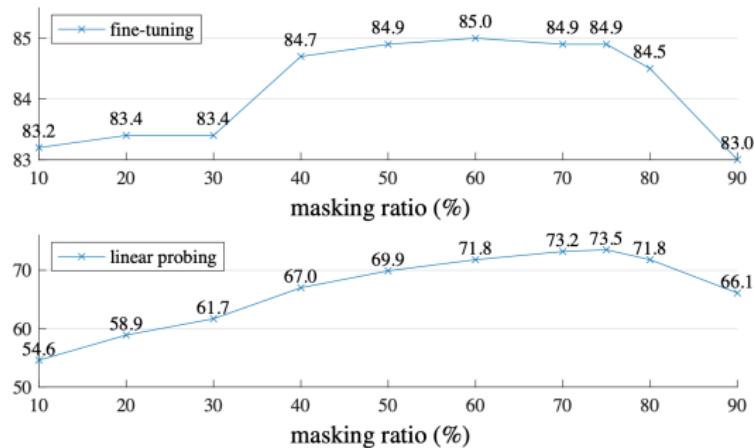


Figure 5. **Masking ratio**. A high masking ratio (75%) works well for both fine-tuning (top) and linear probing (bottom). The y-axes are ImageNet-1K validation accuracy (%) in all plots in this paper.

全体の 75%をマスクして学習したときが最良  
画像の冗長性を排除して学習するにはマスクを大きくしたほうが良い