# SwinTransformer

# 論文ソース

- Swin Transformer: Hierarchical Vision Transformer using Shifted Windows
- 以下解説サイト
  - https://qiita.com/m_sugimura/items/139b182ee7c19c83e70a
  - https://zenn.dev/takoroy/articles/e8c9e2903096b8

# 概要

- 背景
  - 既存のVision Transformerを含め、TransformerをCVに適用するには以下の課題があった
    - 画像を特定サイズのパッチに切り出して入力情報としており、多様な解像度や物体スケールの変化に対応しきれない。
    - 入力データが大きい（おおよそ画像サイズの二乗）ため、計算量が大きすぎる。
  - これらに対応するため、Swin Transformerでは、以下の工夫を行っている。
    - Patch Merginという、Poolingのようにデータの縦横サイズを小さくする機構を導入。
    - Shifted windowsを用いて、局所的なAttentionを計算する。
- 概要
  - 画像分類、物体検出、セマンティックセグメンテーションなどのさまざまなVisionタスクでSOTA達成
  - CNNのような階層的なネットワーク構造により、階層的な表現の獲得を可能にしたVision TransformerであるSwin Transformerを提案
  - Self-Attentionの範囲を固定サイズのWindow内に限定することで、画像サイズに対して線形の計算量で収まるような高効率なネットワーク構造を実現
  - 計算量は以下

$$
\begin{aligned}
\Omega\left(\mathrm{MSA}\right) &= 4hwC^2 + 2\left(hw\right)^2 C \\
\Omega\left(\mathrm{W\text{-}MSA}\right) &= 4hwC^2 + 2M^2 hwC
\end{aligned}
\tag{1}
$$

# SwinTransformerの構造

- 以下の画像の左側のaが全体図

- 右のbはSwinTransformer2つ分の全体図
  - aを見ると分かる通りSwinTransformerは×2でセットで使っている
  - 最初のSwinTransformerではW-MSA(Window Multi head Self Attention)
  - 次のSwinTransformerではSW-MSA(Shifted Window Multi head Self Attention)
  - これを式にすると以下
  - LNはLayer Normalization, MLPは2層のNNで活性化関数はGELU

$$
\begin{aligned}
\hat{z}^{l} &= \text{W-MSA}\left(\text{LN}\left(z^{l-1}\right)\right) + z^{l-1} \\
z^{l} &= \text{MLP}\left(\text{LN}\left(\hat{z}^{l}\right)\right) + z^{l} \\
\hat{z}^{l+1} &= \text{SW-MSA}\left(\text{LN}\left(z^{l}\right)\right) + z^{l} \\
z^{l+1} &= \text{MLP}\left(\text{LN}\left(\hat{z}^{l+1}\right)\right) + \hat{z}^{l+1}
\end{aligned}
\tag{2}
$$

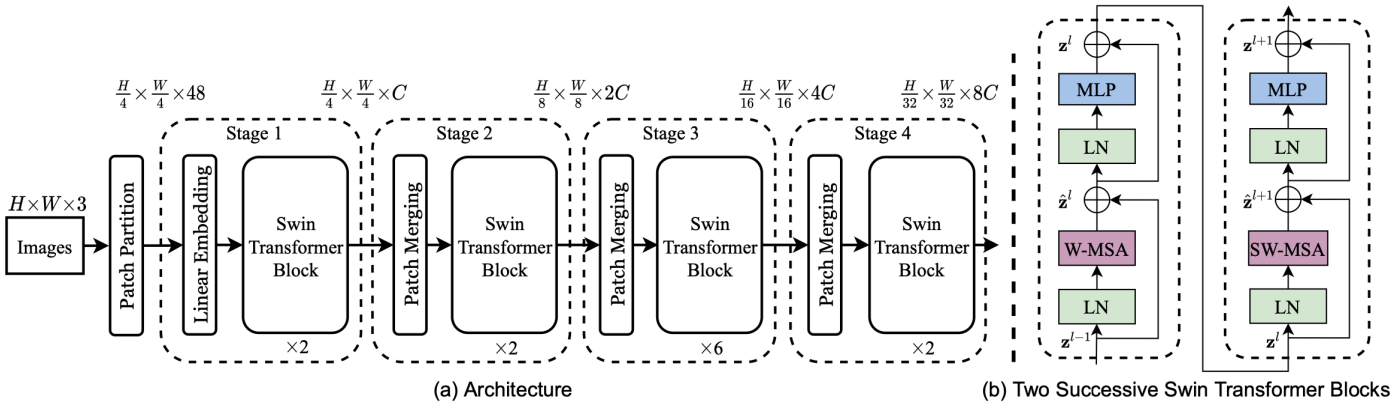



Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

- 1stage目
  - 入力データを、パッチ分割モジュールで分割(重なりのないパッチに分割される)
  - 各パッチをトークンとして扱い、RGB値をconcatしたものを特徴量とする。
  - 今回の実装では、パッチサイズは4 × 4であり各パッチの特徴量の次元は、4 × 4 × 3 = 48。
  - 最初のステージでは、この入力にlinear embedding layerでC次元に圧縮をかける
  - Swin Transformer Blockを2層適用(画像に×2とある)
    - Transformerのencoder（の1block分）がベース
    - Multi head self-attantionが、「Shift Window-based Multi head self-attantion」に改良
      - これがkey idea
- 2stage目以降
  - Patch Marging:HとWを2分の1倍にしてCを2倍に拡張する処理
  - 以下同じくSwin Transformer

## (S)W-MSA

- 下の画像は、$l$層でのW-MSAの説明、$l+1$層でのSW-MSAの説明(論文の画像だとエルとイチの違い分かりにくい)

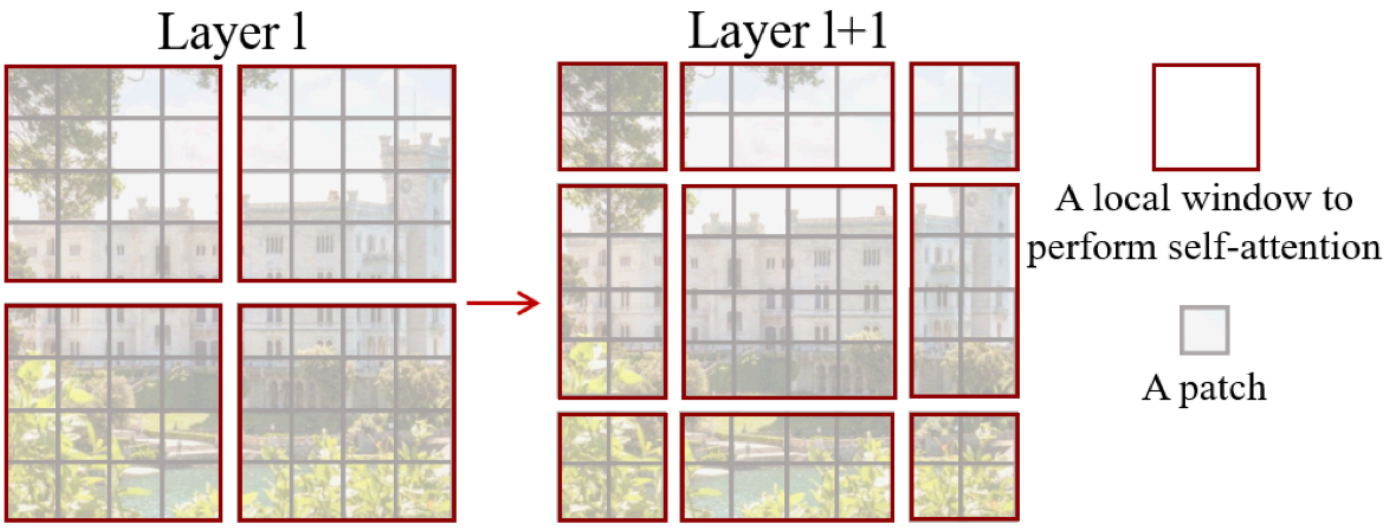



Figure 2. An illustration of the *shifted window* approach for computing self-attention in the proposed Swin Transformer architecture. In layer $l$ (left), a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l+1$ (right), the window partitioning is shifted, resulting in new windows. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer $l$, providing connections among them.

- 赤で分割したwindowごとにself attentionを計算するというアイディア
- $M^2$個のパッチをもった重なりのない(no-overlapping)windowに分割できる(画像は$M=4$)
- $l$層から$l+1$層にいくときに$\left(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor\right)$シフトする(画像は2,2)
- この状態では、元のWindowの大きさを保っているのは1つのみで、その他のWindowが8つでき、合計9個のWindowができることになる
- このままだと計算できないため、SW-MSAにおいては、cyclic shiftという工夫を行う
- 大きさを保っている唯一のwindowを左上に固定して考えるのが大事
- 下図のように、Windowをシフトしてはみ出した部分（薄く表示されている左上のA, B, C）をWindowの反対位置へ移動させる。これにより、Windowの数やWindowあたりのパッチ数は一定となる。標準的なMSAでよく使用されるマスクを設定してあげることで、無関係なパッチ間でAttentionが生じないようにできるので、複雑な実装を回避することができる。

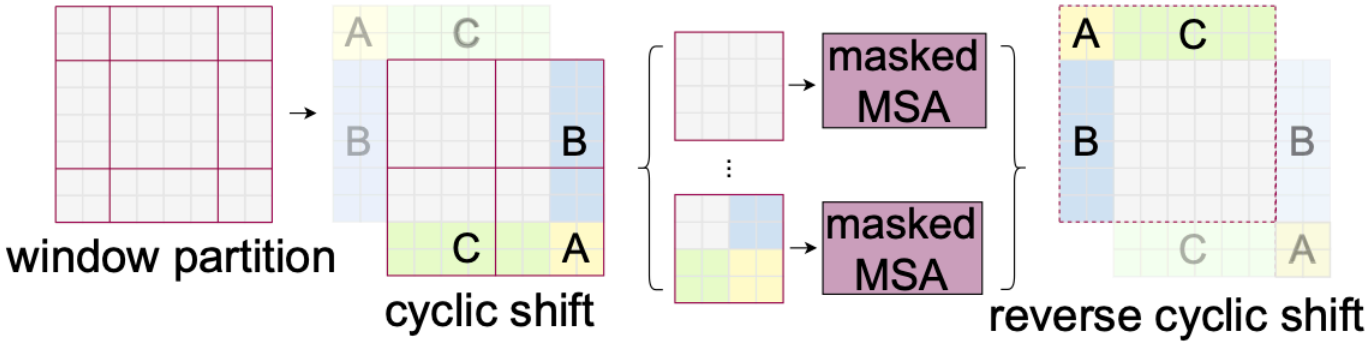
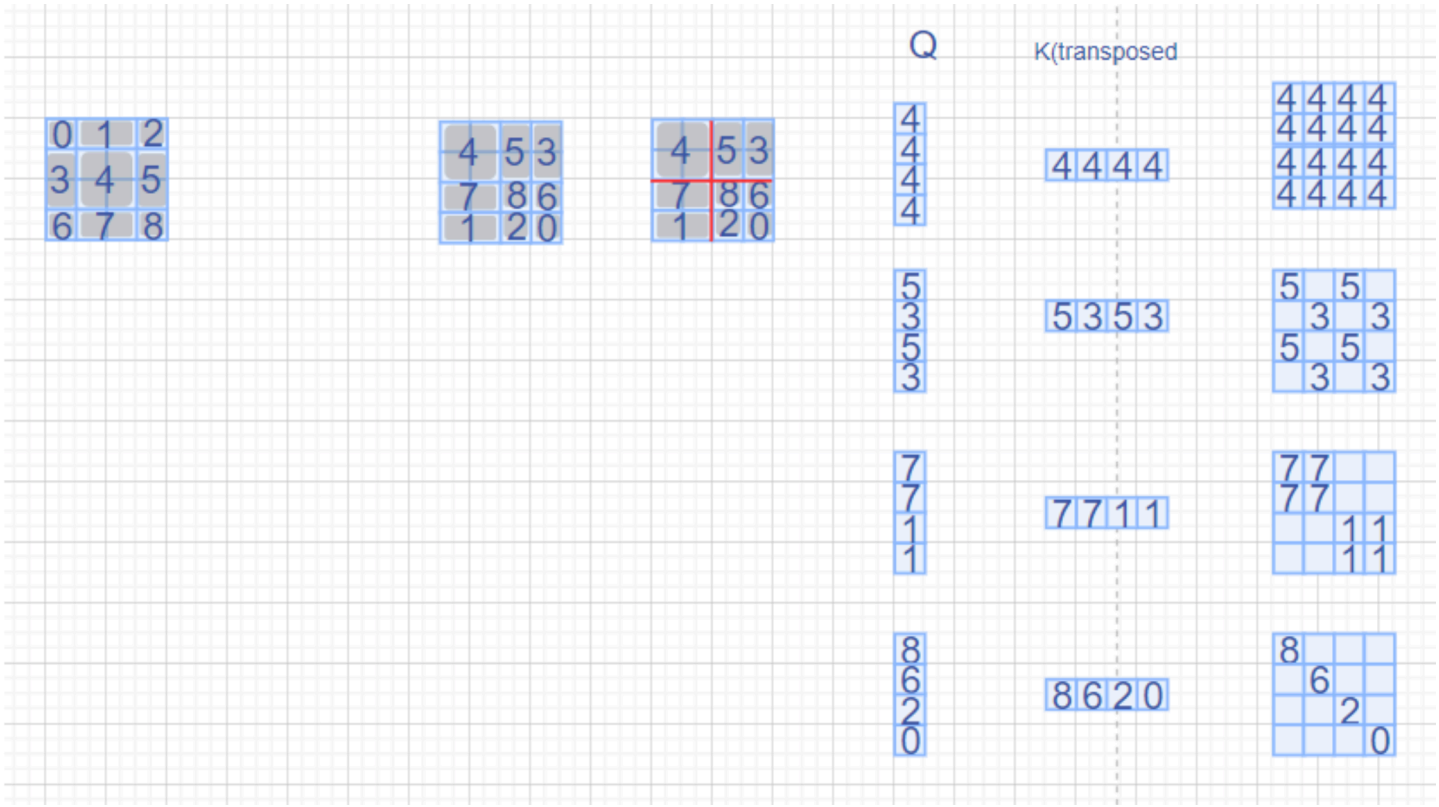



Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

- Cyclic shiftについて以下のgithubのissueにあった画像がわかりやすい




- AttentionにRelative position biasをつける

$$\mathrm{Attention}\left(Q,K,V\right)=\mathrm{SoftMax}\left(\frac{QK^T}{\sqrt{d}}+B\right)V$$