Object Classification with SOTA Models

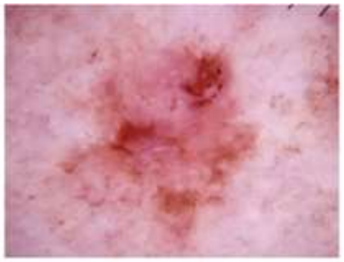
MSc. Nguyen Thanh An
VietAI Teaching Team

Content

- Object classification problem
- SOTA CNN models
  - VGG
  - Res-Net
  - Mobile-Net
  - Efficient-Net
- Attention Mechanism
- Vision Transformer
- Swin Transformer

Object Classification Problem

- Classification
  - Input: an image
  - Output: category/ID of the image
- Examples
  - Face recognition
  - Skin lesion classification
  - Fish classification
  - etc.



https://isysrg.com/2020/06/17/deep-ensemble-learning-for-skin-lesions-classification-with-convolutional-neural-network/

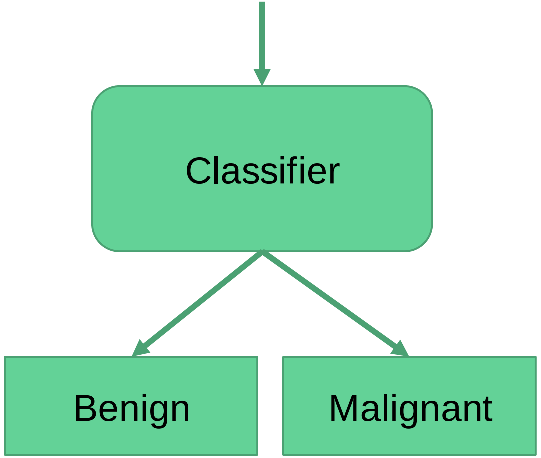


Image Representation

- Images are represented as matrices of pixels in computer.
    - Grayscale image ➔ pixel is a single integer
    - Color image ➔ pixel is a tuple of integers (Red, Green, Blue)
    - Other color spaces: HSV, CMYK, etc.
- Images are usually transformed to meet the structure of Machine Learning model input.
    - Flatten to a vector
    - Resize
    - Color space transformation

https://en.wikipedia.org/wiki/Lenna

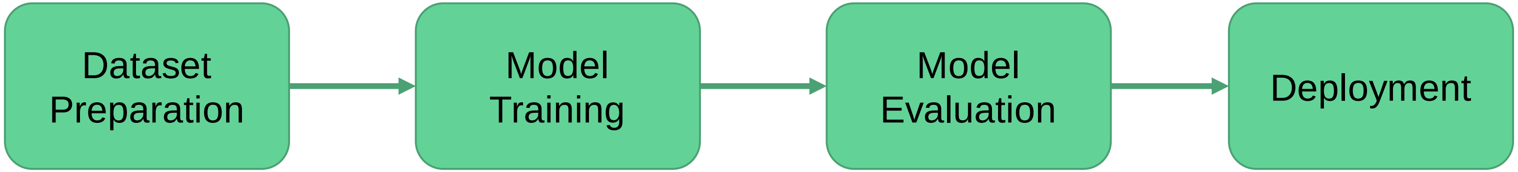
https://www.researchgate.net/figure/8-bit-256-x-256-Grayscale-Lena-Image_fig1_3935609

Classification Approaches

- Handcrafted features
  - Extract discriminative visual pattern in the image → SIFT, SURF, BLOB, HOG, etc.
  - Represent the image as a set of extracted features
  - Apply Machine Learning models to classify these features → SVM, Random Forest, etc.
- Feature-learning
  - Utilize raw images → matrices of pixels
  - Feed the image to a neural network → convolutional neural network
  - Obtain the prediction

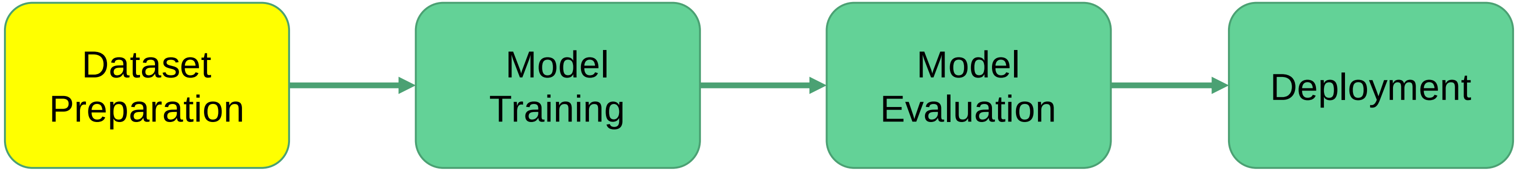
Feature-Learning Approach

- Deep Learning based models, especially convolutional neural network, are common applied.
- Deep Learning pipeline for object classification



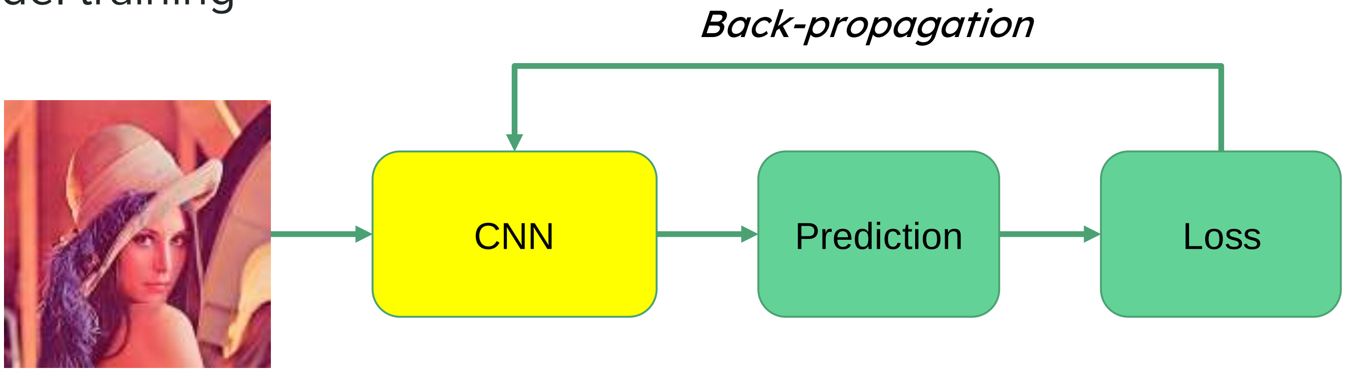
Feature-Learning Approach

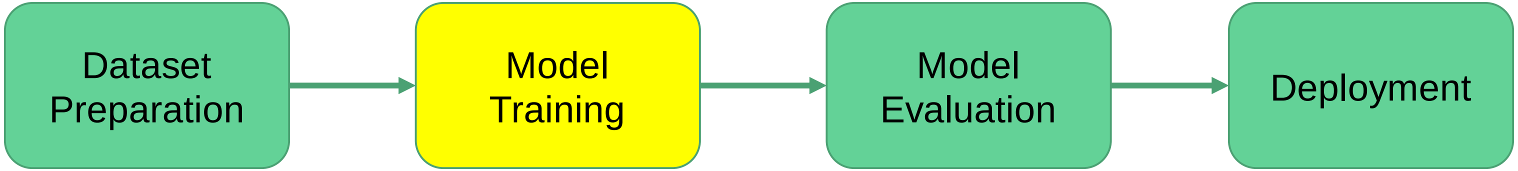
- Dataset Preparation:
  - Samples = {image, label}
  - Preprocessing: cropping, normalization, resizing, etc.



Feature-Learning Approach

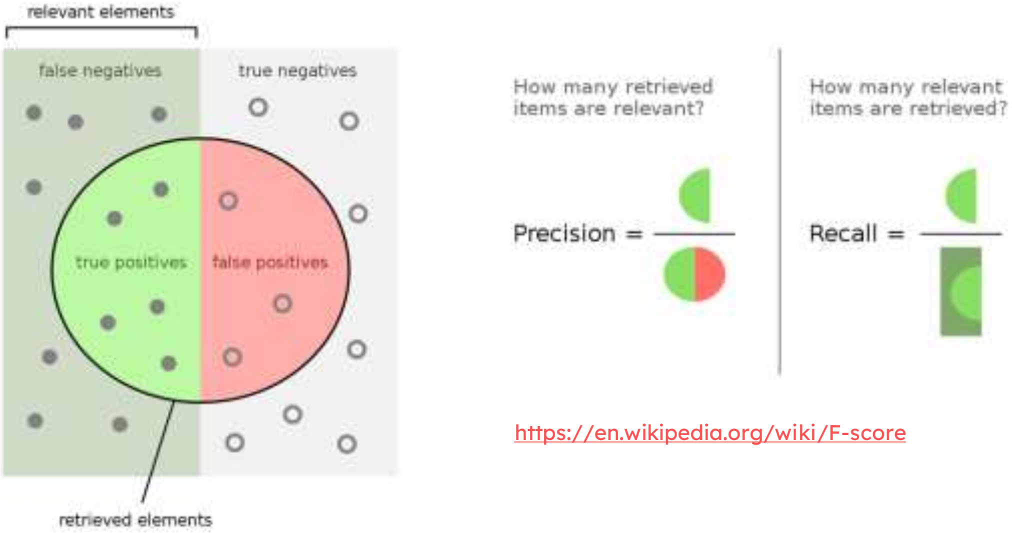
- Model training



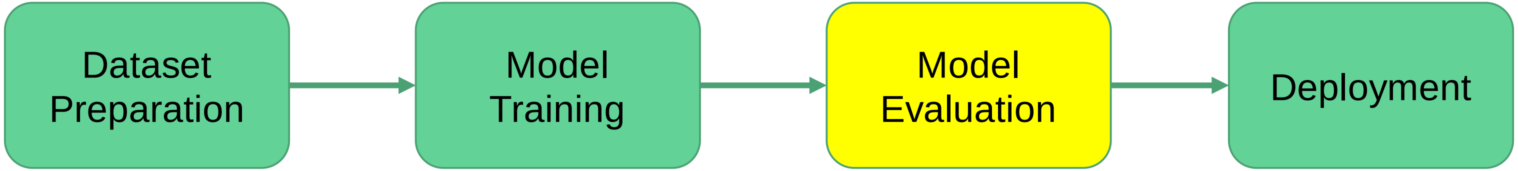


Feature-Learning Approach

- Model Evaluation: common metrics for classification
  - Precision
  - Recall
  - F1-score

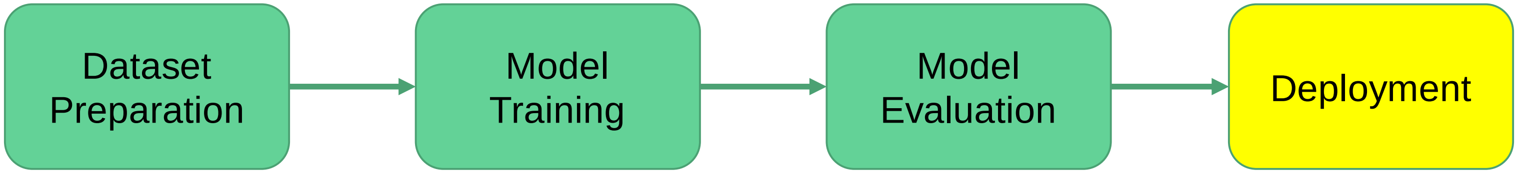


https://en.wikipedia.org/wiki/F-score



Feature-Learning Approach

- Deployment:
  - Implement the pretrained model (with pretrained weights)
  - Integrate into a software.
  - Programming languages and frameworks are carefully considered to optimize performance.



Convolutional Neural Network

- CNN → end-to-end model for image classification
- General architecture
    - Input layer
    - Convolutional layer + ReLU → Max-Pooling layer
    - …
    - Convolutional layer + ReLU → Max-Pooling layer
    - Flattenning
    - Dense layer + Sigmoid + Dropout
    - …
    - Dense layer + Sigmoid + Dropout
    - Softmax layer
    - Output

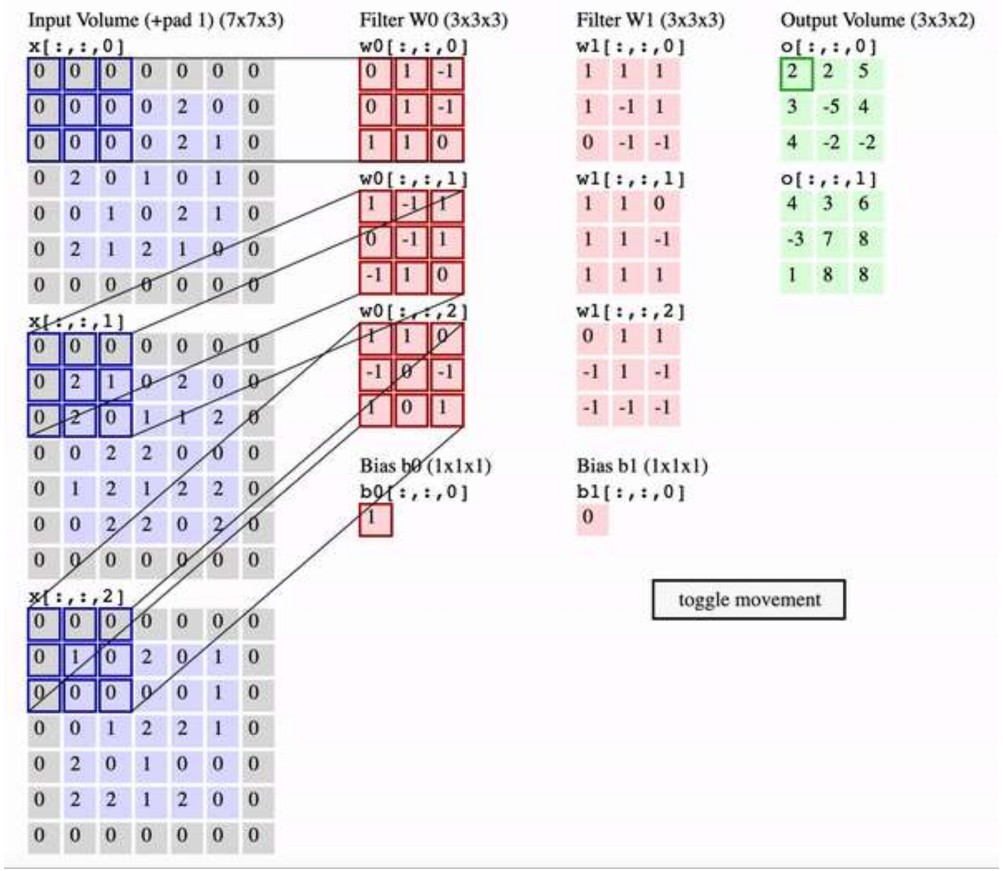
Feature extraction phase

Classification phase

Convolutional layer

Operation in a convolutional layer

Input tensor	[H, W, D]
Filter size	[H', W', D]
Filter banks	N
Bias size	N
Output	[H'', W'', N]

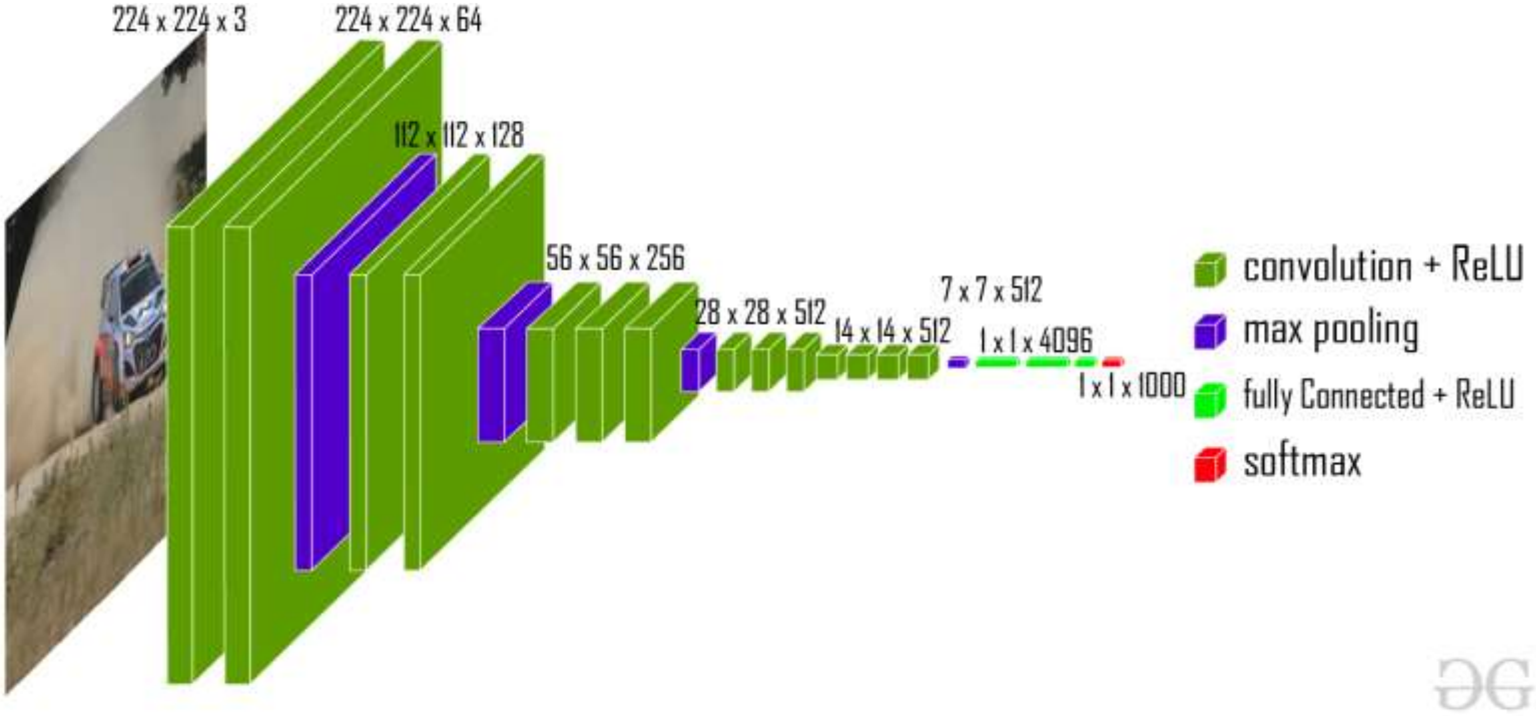


https://cs231n.github.io/convolutional-networks/#conv

History of CNNs

- History of CNNs with typical SOTA models
  - 2014 VGG → https://arxiv.org/abs/1409.1556
  - 2015 Res-Net → https://arxiv.org/abs/1512.03385
  - 2017 Mobile-Net → https://arxiv.org/abs/1704.04861
  - 2019 Efficient-Net → https://arxiv.org/abs/1905.11946

VGG-16

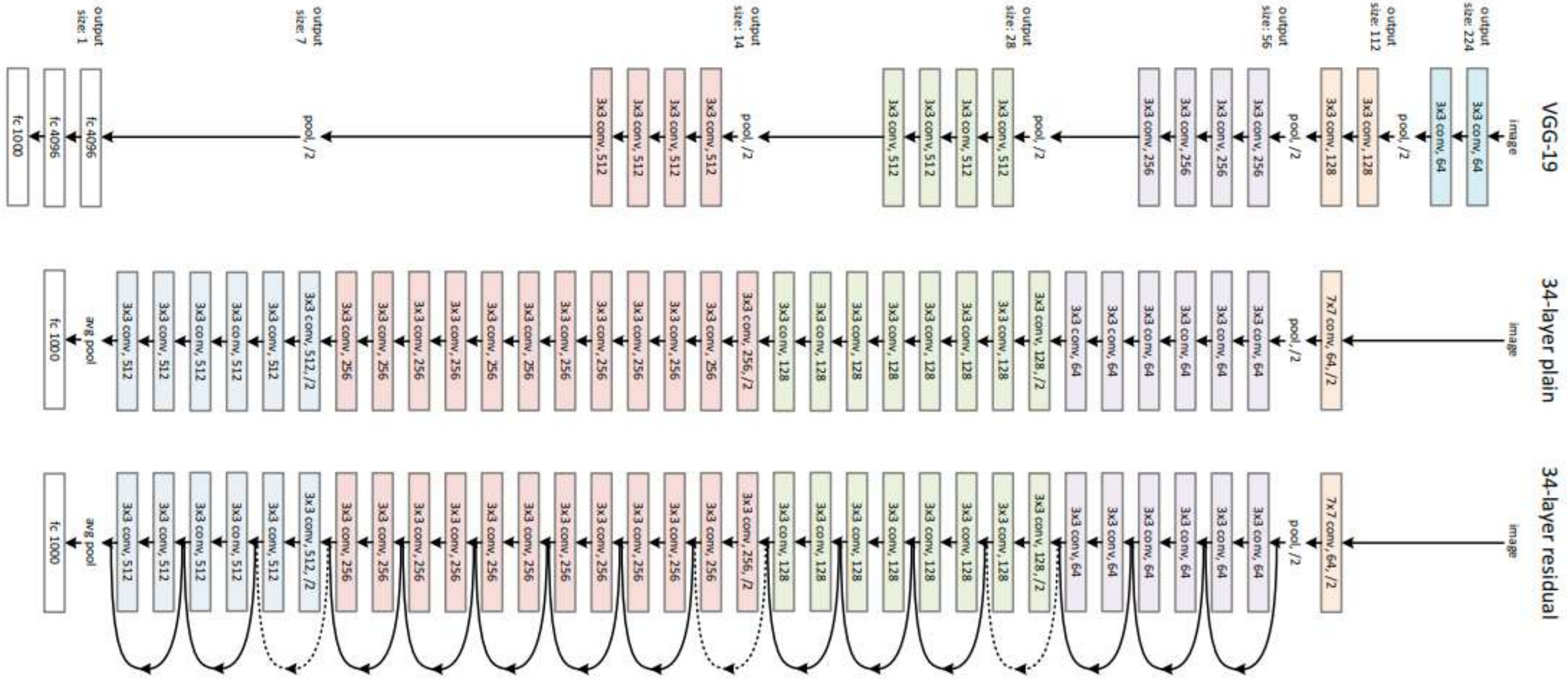


https://www.geeksforgeeks.org/vgg-16-cnn-model/

VGG

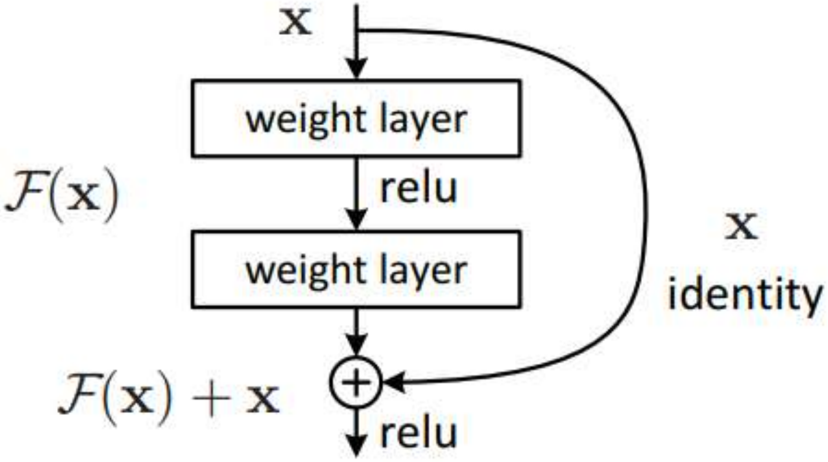
- Depth of a convolutional neural network takes effect of its classification accuracy.
- Shadow networks (with fewer layers) have a smaller number of parameters ➡ underfitting
- Deep networks (with more layers) have a larger number of parameters ➡ overfitting
- VGG's authors: replace a conv. layer with a large kernel size, i.e. 5x5, by a stack of ones with smaller kernel size, i.e. 3x3 ➡ maintain the same effect, but smaller number of parameters.

ResNet-34



https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/?ref=lbp

Residual Block

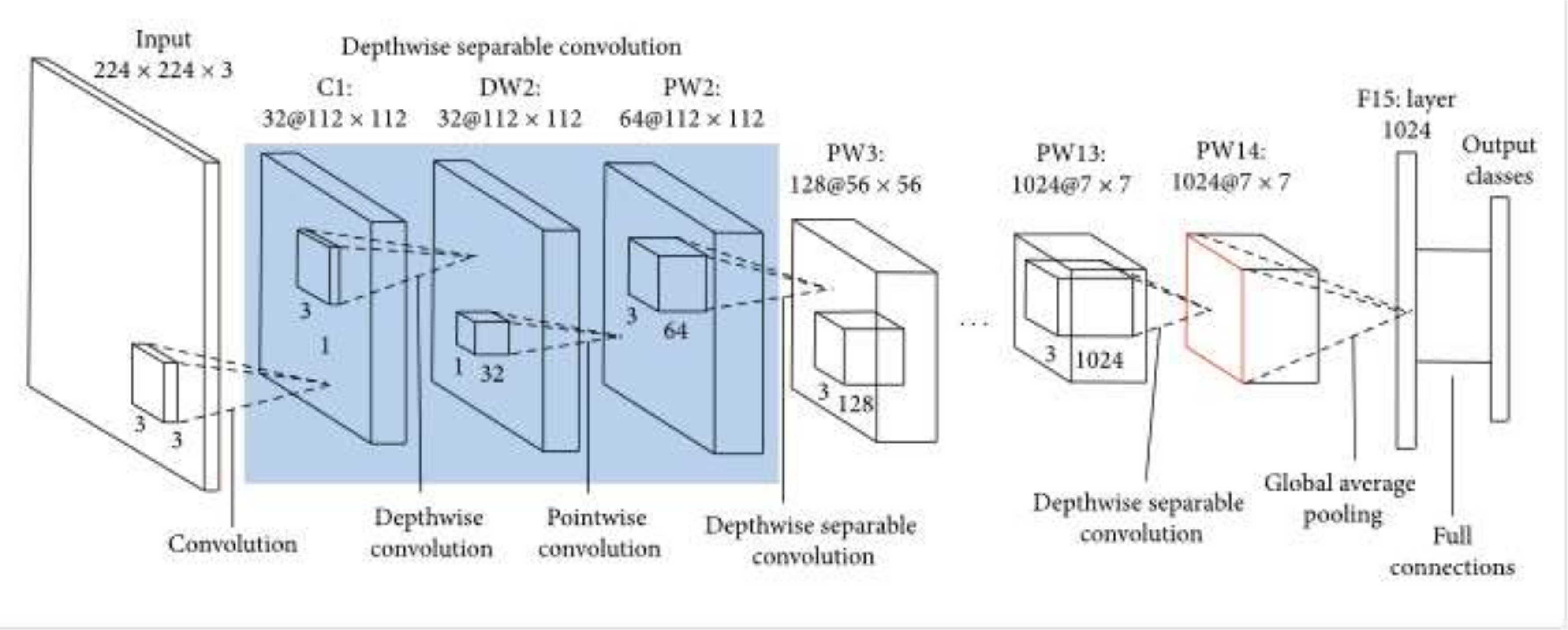


https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/?ref=lbp

Res-Net

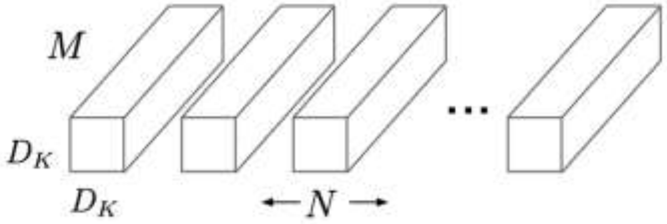
- If any layer hurts the performance of architecture, it will be skipped by regularization.
- Train a very deep neural network without vanishing/exploding gradient.
- The idea of residual blocks can be inherited and customized in later models.

Mobile-Net

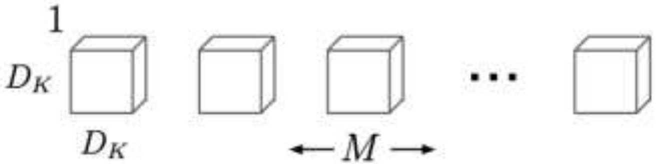


https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470

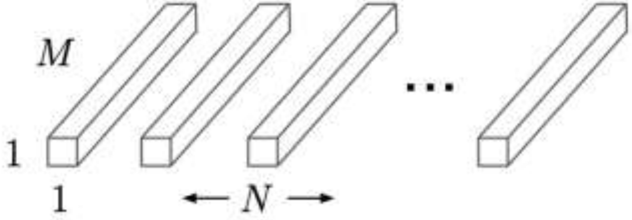
Mobile-Net



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



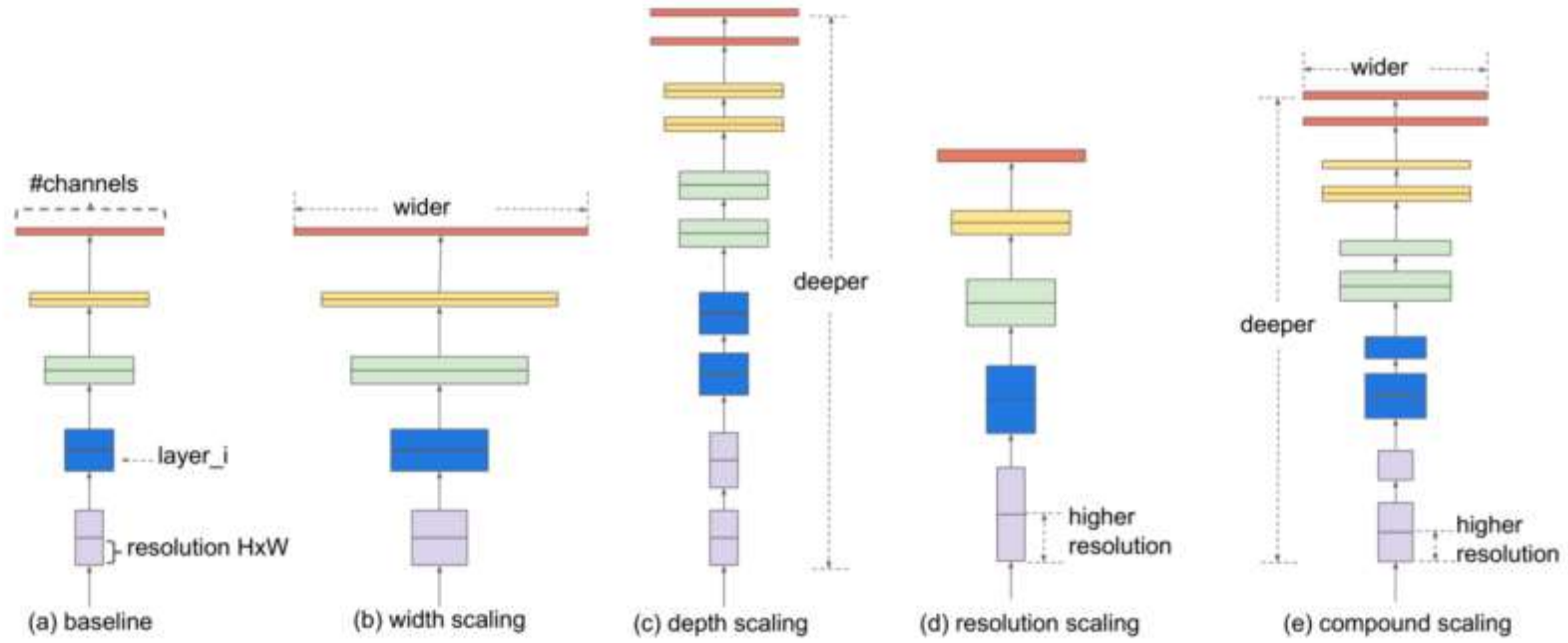
(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

https://arxiv.org/pdf/1704.04861.pdf

Mobile-Net

- Streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks
- Target to mobile and embed devices.
- Depth-wise convolutional filters reduce computation ➔ lightweight model.

Efficient-Net



https://arxiv.org/pdf/1905.11946.pdf

Efficient-Net

- EfficientNet's authors propose a systematic method for model scaling that helps improve accuracy and resource efficiency.
- Previous models are scaled randomly based on three major concepts: depth, width, and resolution ➔ manually tuning
- Scales each dimension with a certain fixed set of scaling coefficients

$$
\begin{aligned}
\text{depth: } & d = \alpha^{\phi} \\
\text{width: } & w = \beta^{\phi} \\
\text{resolution: } & r = \gamma^{\phi} \\
\text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\
& \alpha \geq 1, \beta \geq 1, \gamma \geq 1
\end{aligned}
$$

Efficient-Net

Table 1. **EfficientNet-B0 baseline network** – Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$. Notations are adopted from equation 2.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

https://arxiv.org/pdf/1905.11946.pdf

Efficient-Net

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

https://arxiv.org/pdf/1905.11946.pdf

Attention Mechanism

- Attention mechanism
  - Enhance important parts and
  - Fade out non-relevant information
- Simple example in NLP:
  - Self-attention compares every word in a sentence to each other
  - Reweighting the embeddings of each word to include contextual relevance
- E.g. : bank of the river
  - “River” changes the contextual meaning of “bank”
  - If we do not see “river”, then misunderstanding

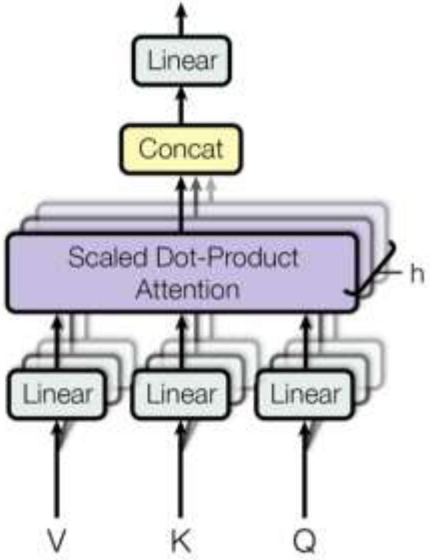
Basic Attention Module

- Basic structure of an attention module
  - Tensor x1 ➔ generates a “key” and a “value”
  - Tensor x2 ➔ generates a “query”
  - Output is the weighted sum of “value”
  - Weights are computed by compatibility function of “query” and the corresponding “key”

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-head Attention

- Multi-head attention:
  - runs an attention module several times and concatenate outputs.
  - jointly attend to information from different representation subspaces at different positions (not able for single-head attention)



https://paperswithcode.com/method/multi-head-attention

Convolutional Block Attention

- Convolutional Block Attention: emphasizes meaningful features along
  - the channel and
  - spatial axes
- Applicable at every convolutional block

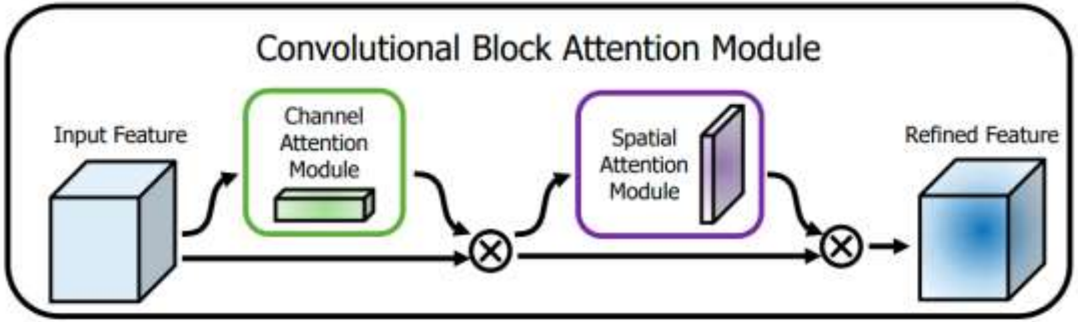


Fig. 1: **The overview of CBAM**. The module has two sequential sub-modules: *channel* and *spatial*. The intermediate feature map is adaptively refined through our module (CBAM) at every convolutional block of deep networks.

https://arxiv.org/pdf/1807.06521.pdf

Convolutional Block Attention

- Channel Attention Module (CAM):
  - input tensor ➔ 2 vectors (c × 1 × 1) (Global Average Pooling and Global Max Pooling)
  - output ➔ a fully connected layer + ReLU

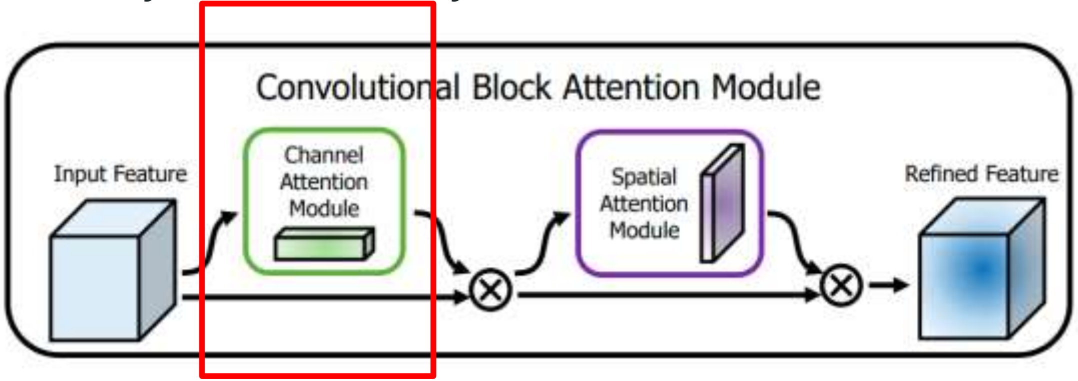


Fig. 1: **The overview of CBAM**. The module has two sequential sub-modules: *channel* and *spatial*. The intermediate feature map is adaptively refined through our module (CBAM) at every convolutional block of deep networks.

https://arxiv.org/pdf/1807.06521.pdf

Convolutional Block Attention

- Spatial Attention Module (SAM):
    - Channel Pool: apply Max Pooling and Average Pooling across the channels
        - $(c \times h \times w) \rightarrow (2 \times h \times w)$.
    - Conv. layer + BatchNorm + ReLU: $(1 \times h \times w)$
    - Sigmoid Activation layer

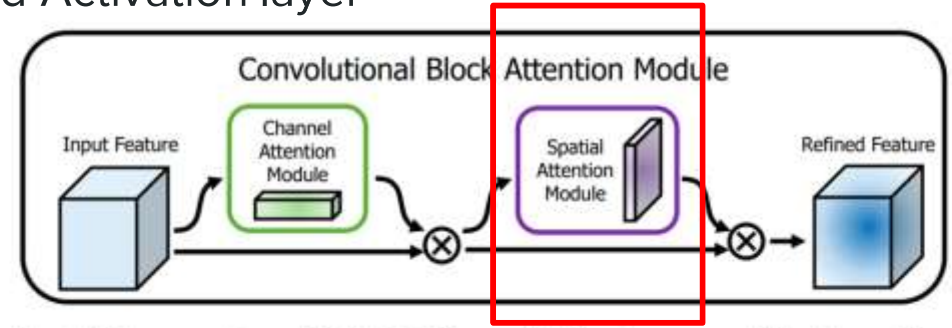


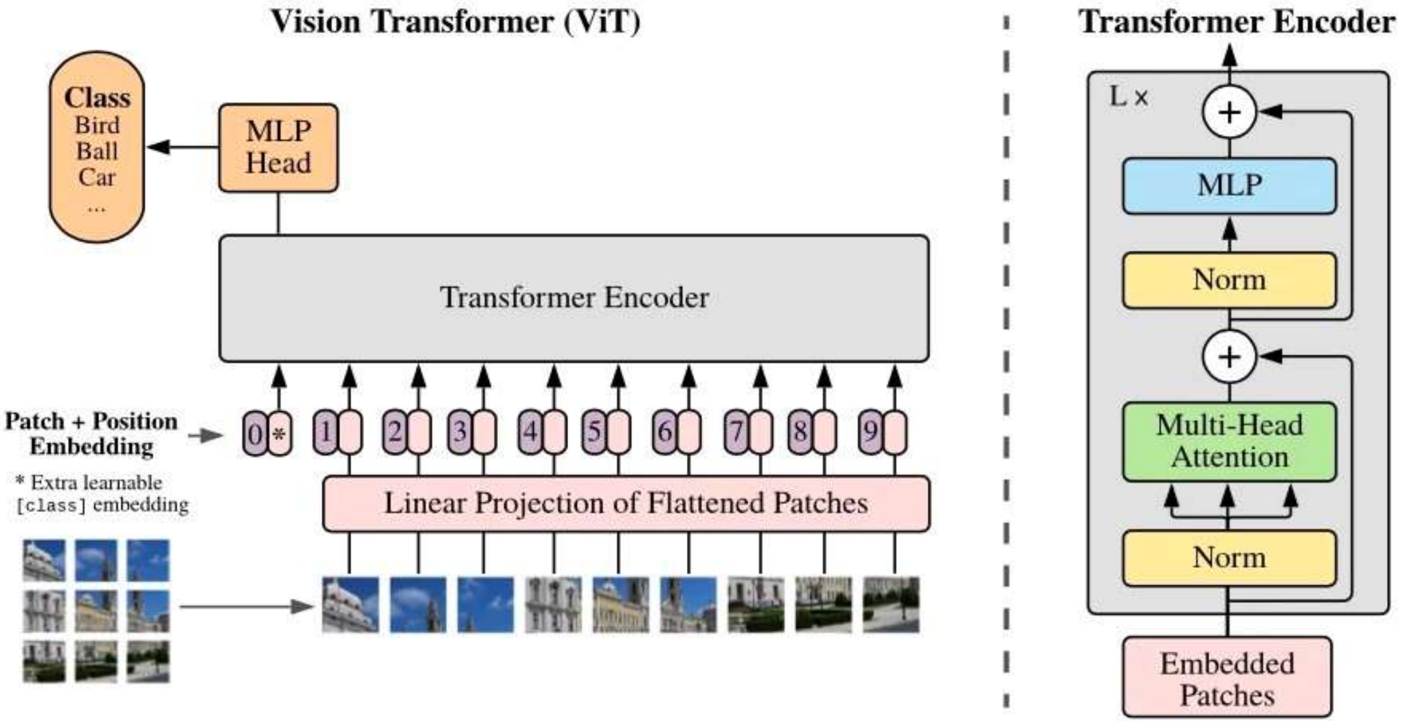
Fig. 1: **The overview of CBAM**. The module has two sequential sub-modules: *channel* and *spatial*. The intermediate feature map is adaptively refined through our module (CBAM) at every convolutional block of deep networks.

https://arxiv.org/pdf/1807.06521.pdf

Vision Transformer

- Vision Transformer is a recent advanced technique without reliance on convolutional neural networks.
  - Empirical results surpass CNNs'
  - Lower computational cost
- Main idea:
  - Divide images into patches encoded with position information
  - Handle encodings using attention mechanism
  - Classify output tensors using MLP.

Vision Transformer



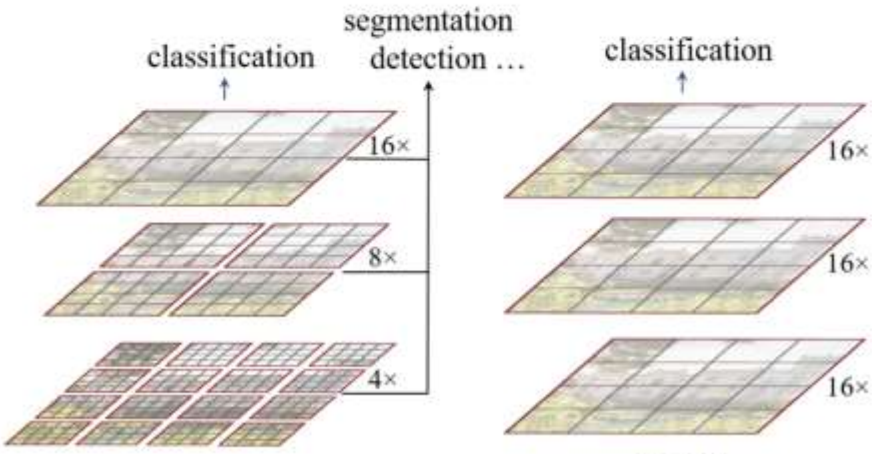
https://arxiv.org/pdf/2010.11929.pdf

Procedure of Vision Transformer

- Divide the input image into patches of a fixed size
- Flatten patches
- Generate feature embeddings with lower dimensionality from patches
- Attach order (position) of patches in feature embeddings
- Feed embeddings to a transformer encoder
- Train and evaluate the model in the whole dataset
- Tune model hyperparameters regarding to individual problems/cases

Swin Transformer

- Image resolution is a challenge of ViT, which is different from texts.
- Swin transformer is a hierarchical transformer whose representation is computed with Shifted windows.
- Shifted window based self-attention ➔ reduce computational cost.



(a) Swin Transformer (ours)   (b) ViT

https://arxiv.org/pdf/2103.14030.pdf

Swin Transformer

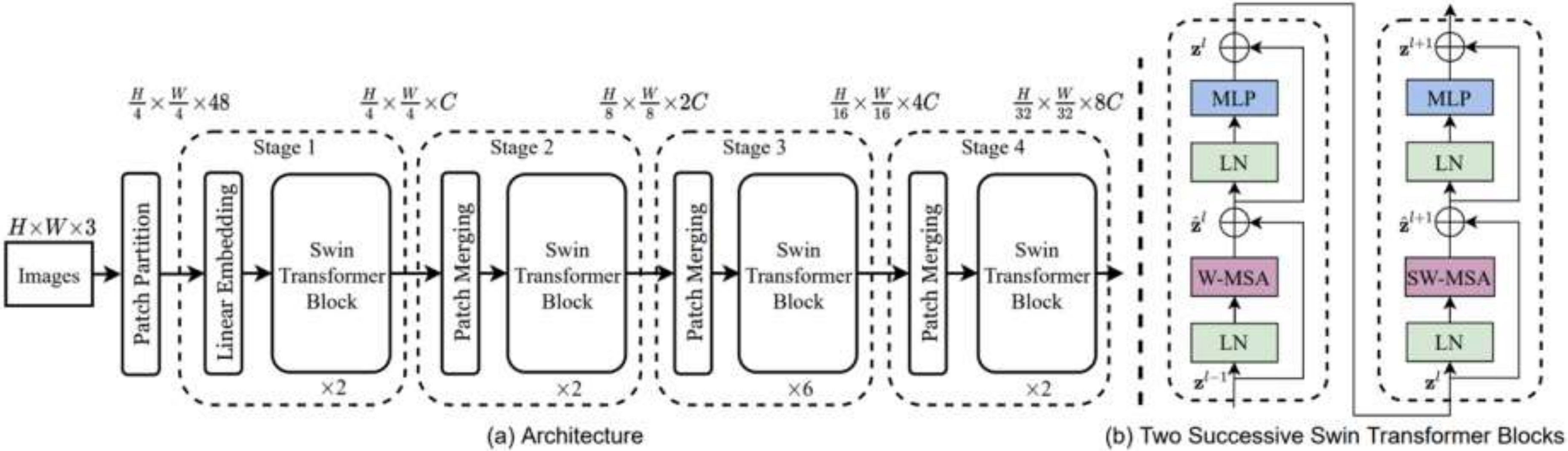


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

https://arxiv.org/pdf/2103.14030.pdf

Swin Transformer

- Images ➔ non-overlapping patches
- Linear projection(patches) ➔ features
- Swin transformer blocks
- Patch merging (neighbor patches of each group 2 x 2)
  ➔ patch size increases after each “stage”

Swin Transformer

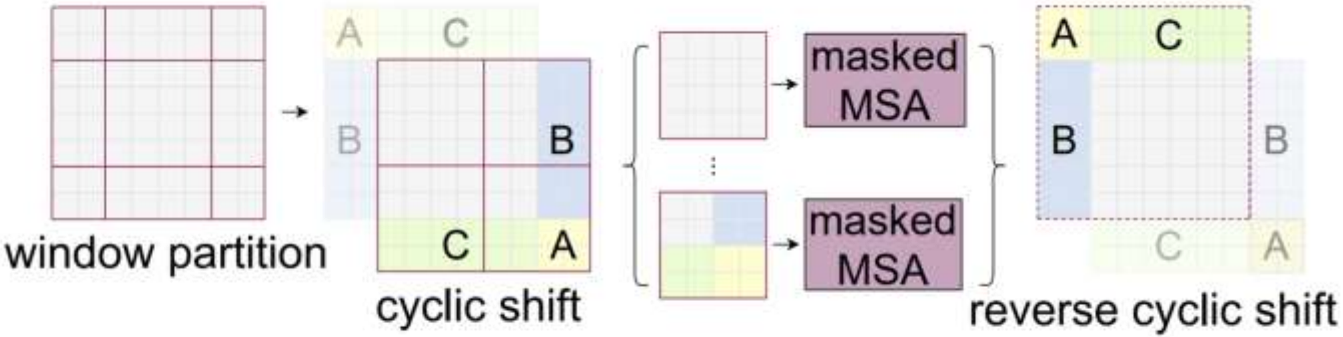
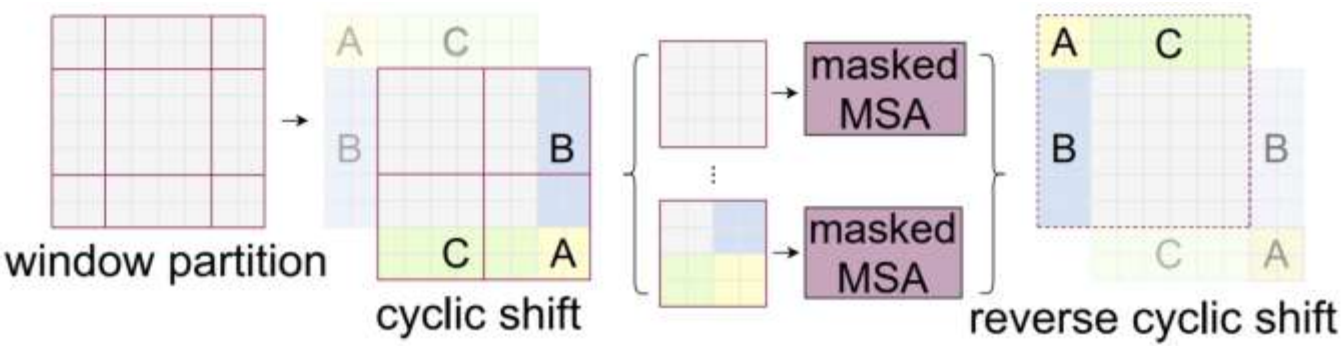


Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

https://arxiv.org/pdf/2103.14030.pdf

Swin Transformer

- Shift the windows cyclically ➔ windows with small size at borders and corners.
- Repeat the image periodically instead of zero-padding.
- “Padded regions” are handled using masks ➔ limit self-attention computation to within non-adjacent sub-windows.



https://arxiv.org/pdf/2103.14030.pdf