University of Rajshahi

Assignments of Computer Vision

Course Name: Computer Vision
Course Code: CSE M 2151
Date: November 16, 2024

Prepared by
Md. Meem Mursalin Chowdhury
ID: 1810276103
Year: Masters, Semester: 2nd
Session: 2017-18

Submitted to
Sangeeta Biswas
Associate Professor
Dept. of Computer Science and Engineering,
University of Rajshahi

Dept. of Computer Science and Engineering,
University of Rajshahi

Contents

1 Assignment-1

Question: Comparison among different CNN model architecture.

Solution: The Comparison among CNN Architectures is given below-

VGG

Who Made It: Simonyan and Zisserman (2014).
Why Made It: To deepen networks for improved feature learning.
Why Needed: Shallow networks failed to extract deep hierarchical features.
Why It Is Called: Named after the "Very Deep Convolutional Networks" paper.
No. of Parameters: 138M (VGG-16).
No. of Total Layers: 16 (VGG-16) or 19 (VGG-19).
No. of Feature Layers: Sequential convolutional layers.
No. of Head Layers: 3 Fully Connected layers.
Type of Layer Used: 3x3 Convolutions, Fully Connected.
Type of Convolution Layer Used: Regular convolutions.
Type of Connection Used: Sequential.
Other Features: High parameter count, simple sequential structure, widely used for benchmarking.

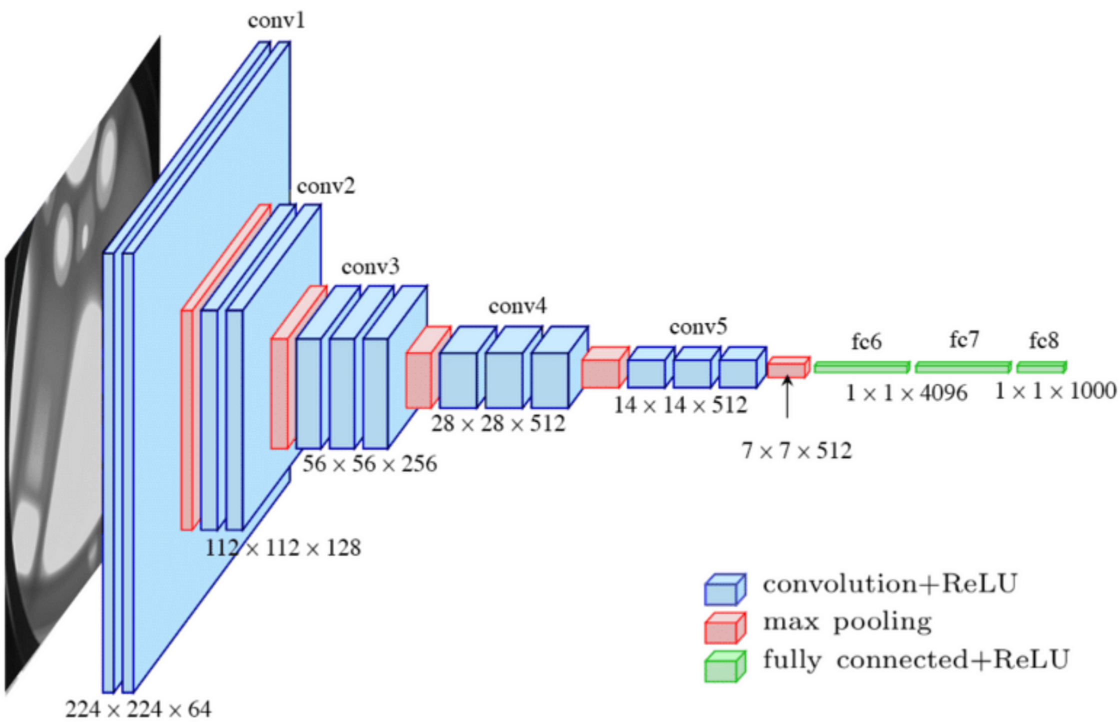

Figure 1: VGG16 Architecture

Inception

Who Made It: Google (2015).
Why Made It: To improve computational efficiency and feature extraction using multiple kernel sizes.
Why Needed: Traditional convolutional networks were computationally expensive.
Why It Is Called: Named after the "Inception Module."
No. of Parameters: 5M (Inception v1) to 23M (Inception v4).
No. of Total Layers: 22 (Inception v1) to 55 (Inception v4).
No. of Feature Layers: Multiple kernel-size feature extraction layers.
No. of Head Layers: Global average pooling + Dense.
Type of Layer Used: Inception Modules (multi-branch convolutions).
Type of Convolution Layer Used: Regular, 1x1, 3x3 convolutions in branches.
Type of Connection Used: Multi-branch (Inception Modules).
Other Features: Efficient multi-scale feature extraction, modular design.

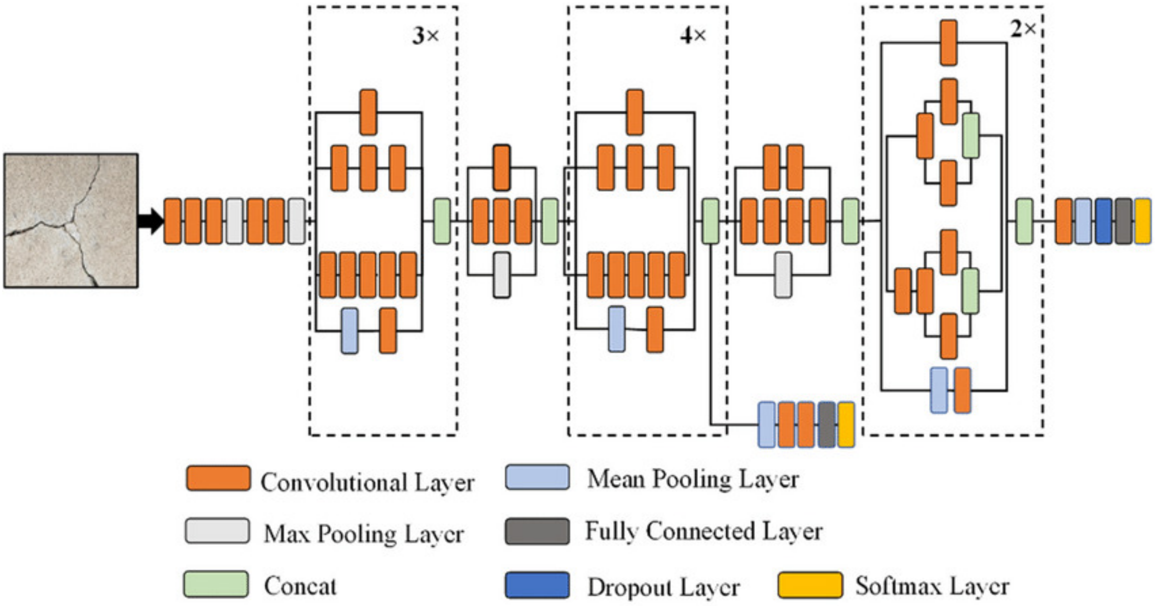

Figure 2: InceptionV3 Architecture

ResNet

Who Made It: Microsoft (2015).
Why Made It: To solve vanishing gradient issues and enable very deep networks.
Why Needed: Training deep networks without degradation was difficult.
Why It Is Called: Named for the residual learning mechanism.
No. of Parameters: 11M (ResNet-18) to 60M+ (ResNet-152).
No. of Total Layers: 18 to 152+.
No. of Feature Layers: Residual blocks.
No. of Head Layers: Fully connected with softmax.
Type of Layer Used: Residual Blocks.
Type of Convolution Layer Used: Regular convolutions + Identity mapping.
Type of Connection Used: Shortcut (Residual Connections).
Other Features: Extremely deep, avoids degradation via residuals.

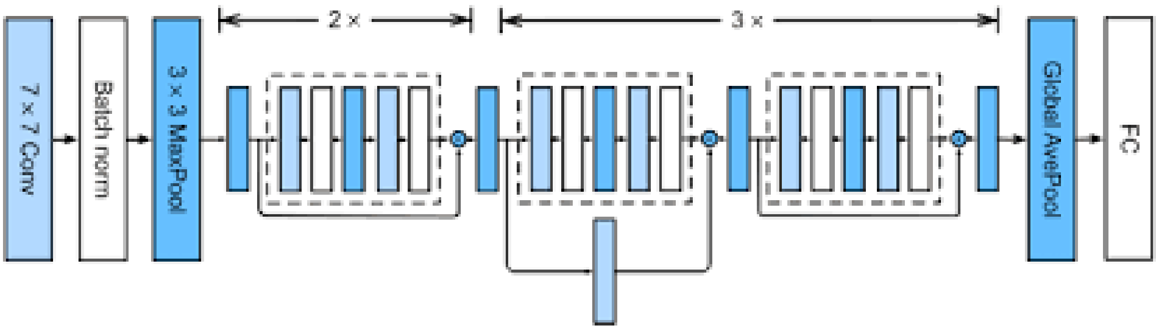

Figure 3: ResNet Architecture

EfficientNet

Who Made It: Google (2019).
Why Made It: To balance accuracy and efficiency via neural architecture search (NAS).
Why Needed: To create models with better performance across different resource constraints.
Why It Is Called: Efficient due to compound scaling.
No. of Parameters: 5M (EfficientNet-B0) to 66M (EfficientNet-B7).
No. of Total Layers: 18 to 66 depending on model variant.
No. of Feature Layers: Scaled feature blocks.
No. of Head Layers: Fully connected with softmax.
Type of Layer Used: Swish activation, MBConv blocks.
Type of Convolution Layer Used: Regular + Mobile Inverted Bottleneck.
Type of Connection Used: Compound Scaling connections.
Other Features: Scalable and optimized for performance and efficiency.

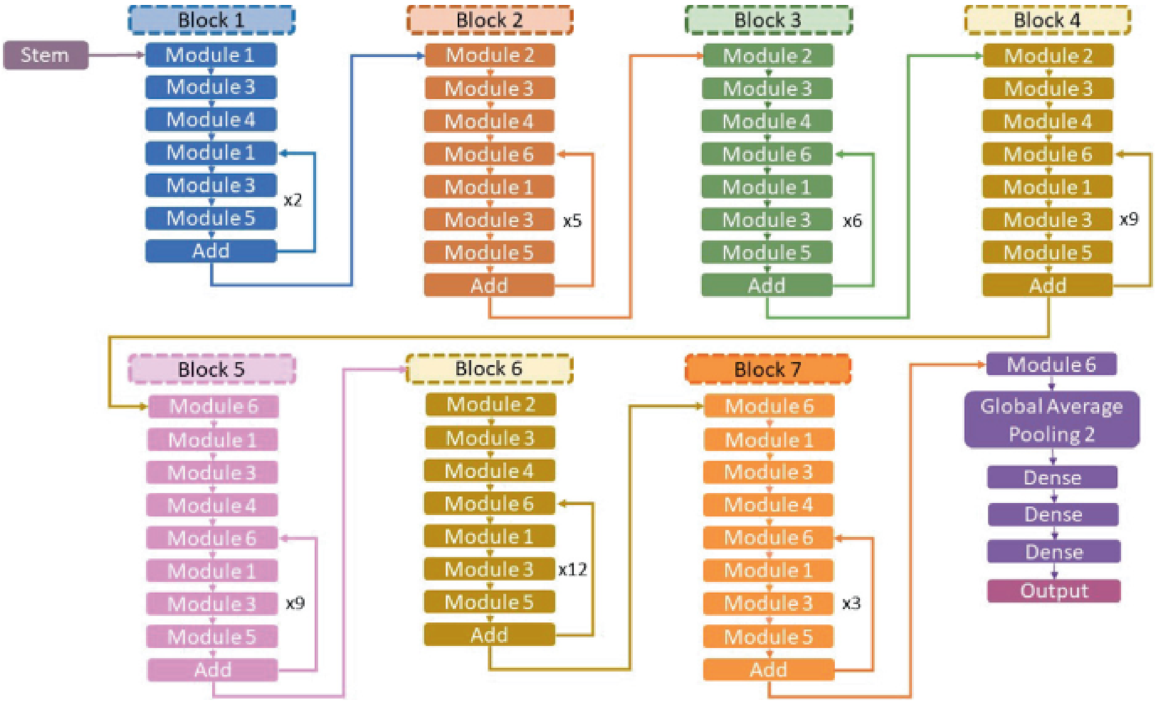

Figure 4: EfficientNet B7 Architecture

MobileNet

Who Made It: Google (2017).
Why Made It: To reduce computational cost for mobile devices.
Why Needed: To deploy efficient models for mobile/embedded systems.
Why It Is Called: Mobile-focused efficiency.
No. of Parameters: 4.2M (MobileNet v1).
No. of Total Layers: 28 (MobileNet v1).
No. of Feature Layers: Depthwise separable convolutions.
No. of Head Layers: Global average pooling + Dense.
Type of Layer Used: Depthwise Separable Convolutions.
Type of Convolution Layer Used: Depthwise Separable.
Type of Connection Used: Lightweight connections.
Other Features: Highly efficient for mobile devices.

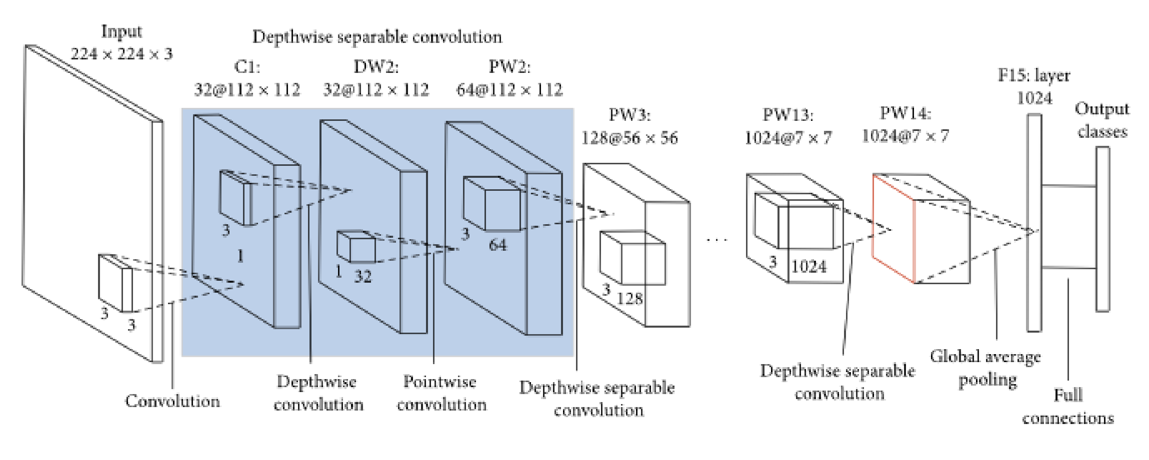

Figure 5: MobileNet Architecture

XceptionNet

Who Made It: Google (2017).
Why Made It: To extend Inception by using depthwise separable convolutions.
Why Needed: To simplify and enhance InceptionNet's design.
Why It Is Called: Extended depthwise separable convolutions.
No. of Parameters: 22.8M (Xception).
No. of Total Layers: 36 (Xception).
No. of Feature Layers: Depthwise separable convolutions.
No. of Head Layers: Global average pooling + Dense.
Type of Layer Used: Depthwise Separable Convolutions.
Type of Convolution Layer Used: Depthwise Separable.
Type of Connection Used: Lightweight connections.
Other Features: Simplifies Inception, uses extreme separation of convolutions.

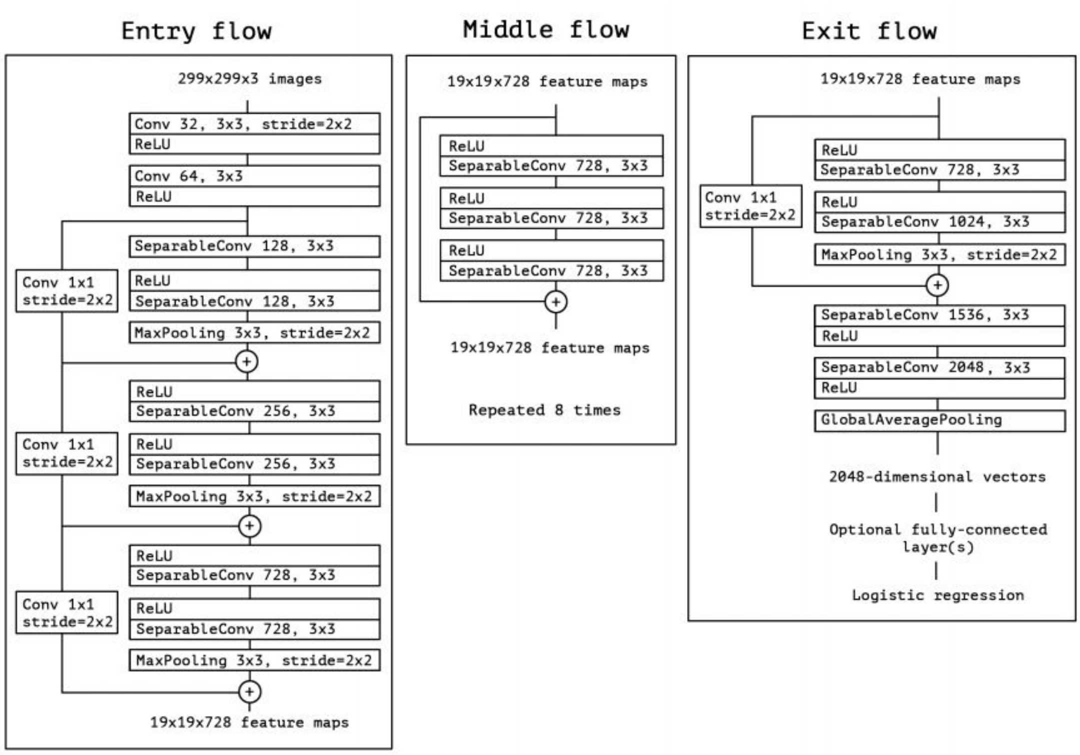

Figure 6: XceptionNet Architecture

DenseNet

Who Made It: Huang et al. (2017).
Why Made It: To improve feature propagation and reduce redundancy via dense connections.
Why Needed: To encourage feature reuse and improve parameter efficiency.
Why It Is Called: Dense connectivity between layers.
No. of Parameters: 8M (DenseNet-121).
No. of Total Layers: 121, 169, 201, or 264 depending on variant.
No. of Feature Layers: Dense blocks.
No. of Head Layers: Global average pooling + Dense.
Type of Layer Used: Dense Blocks, Transition Layers.
Type of Convolution Layer Used: Regular convolutions with dense connectivity.
Type of Connection Used: Dense Connections.
Other Features: High feature reuse, efficient gradient flow, reduced parameter count.

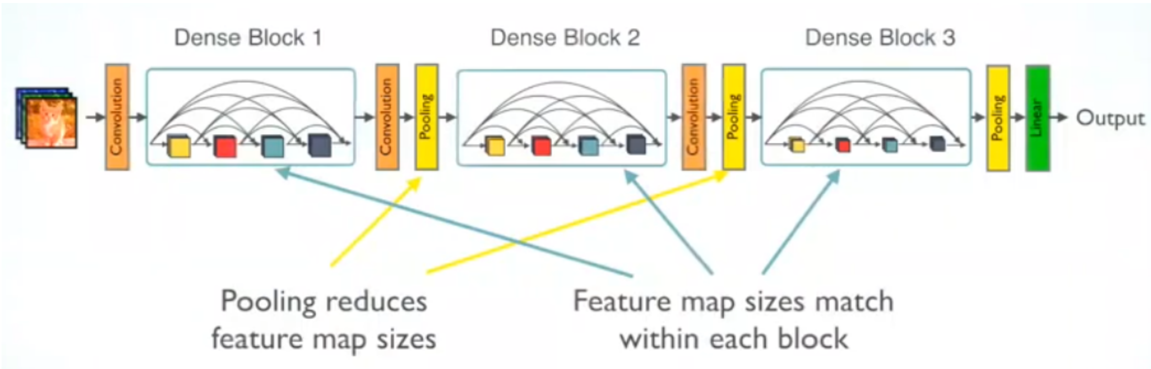

Figure 7: DenseNet Architecture