Comparative Analysis of Deep Learning Architectures for Transfer Learning Applications

1. VGG-19 (Transfer Learning)

1.1. Original Paper Reference

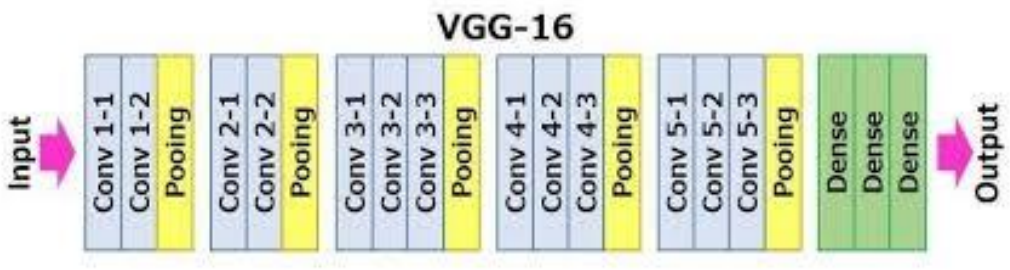
Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. University of Oxford (Visual Geometry Group – VGG).

1.2. Core Architectural Principles

- Utilizes small 3×3 convolutional filters exclusively.
- Implements a deep, sequential architecture comprising 19 weight layers.
- Employs stacked small kernels to increase network depth and non-linearity while maintaining a manageable parameter count per layer.
- Prioritizes architectural simplicity and depth over heterogeneous filter design.

1.3. Detailed Architecture Specification

- **Input Layer:** Accepts RGB imagery of dimensions 224 × 224 × 3.
- **Convolutional Blocks 1 & 2:** Each block contains two convolutional layers (with 64 and 128 filters, respectively) followed by a 2×2 max-pooling operation.
- **Convolutional Block 3:** Contains four convolutional layers (256 filters each) followed by max-pooling.
- **Convolutional Blocks 4 & 5:** Each block contains four convolutional layers (512 filters each) followed by max-pooling.
- **Fully Connected Classifier:** Three dense layers (FC1: 4096 neurons, FC2: 4096 neurons, FC3: 1000 neurons for ImageNet).
- **Output Layer:** Softmax activation function for classification.
- **Architecture Diagram**



1.4. Transfer Learning Protocol

- Initialize the model with weights pre-trained on the ImageNet dataset.
- Freeze parameters within all convolutional blocks to preserve learned feature representations.
- Replace the original final fully connected layer (FC3) with a new, task-specific dense layer containing a number of neurons equal to the target classes, followed by a softmax activation.

2. ResNet (Transfer Learning)

2.1. Original Paper Reference

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. Microsoft Research.

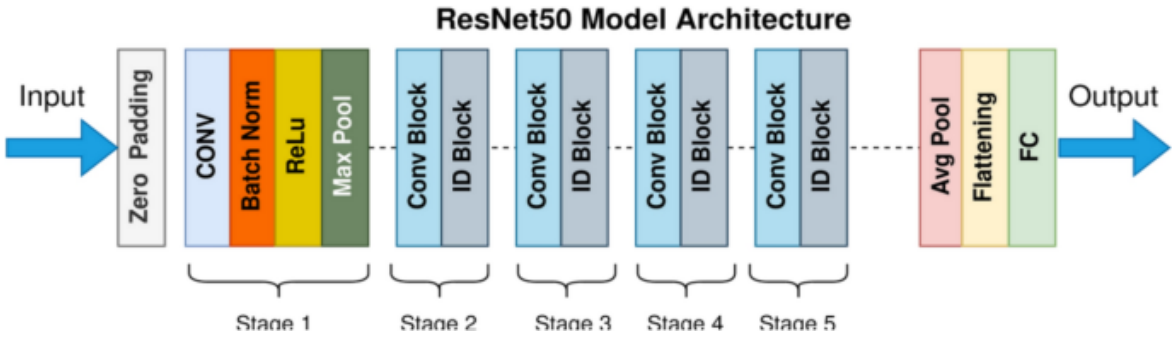
2.2. Core Architectural Principles

- Introduces residual learning frameworks to address the vanishing/exploding gradient problem in very deep networks.
- Implements skip connections (identity mappings) that bypass one or more layers, enabling the training of networks with hundreds or thousands of layers.

- The fundamental building block learns a residual function: *F(x) + x*, where *x* is the input to the block.

2.3. Residual Block and Architecture Overview (ResNet-50 Example)

- **Input:** 224 × 224 × 3 imagery.
- **Initial Processing:** A single 7×7 convolutional layer followed by 3×3 max-pooling.
- **Residual Stages:** Four sequential stages, each comprising multiple bottleneck residual blocks (e.g., [1×1, 64], [3×3, 64], [1×1, 256]) with increasing filter counts (64, 128, 256, 512).
- **Output Processing:** Global average pooling layer, followed by a fully connected layer and softmax activation.
- **Architecture Diagram**



2.4. Transfer Learning Protocol

- Utilize a model pre-trained on ImageNet (e.g., ResNet-50, ResNet-101).
- Common practice involves freezing the weights of the initial convolutional and early residual stages.
- The final fully connected classification layer is replaced and retrained for the specific downstream task.

3. Inception V1 (GoogLeNet) – Transfer Learning

3.1. Original Paper Reference

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2014). *Going Deeper with Convolutions*. Google Research.

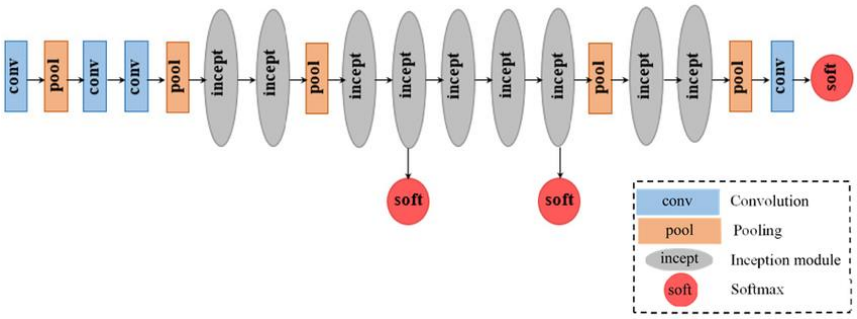
3.2. Core Architectural Principles

- Proposes the "Inception module," which performs multi-scale feature extraction by applying convolutional filters of different sizes (1×1, 3×3, 5×5) and a pooling operation in parallel.
- Employs 1×1 convolutions for dimensionality reduction and to control computational complexity.
- Designed to approximate a sparse network structure with dense, computationally efficient components.

3.3. Inception Module and Architecture Flow

- **Inception Module:** Concatenates the output feature maps from four parallel branches: 1×1 conv, 3×3 conv (preceded by 1×1), 5×5 conv (preceded by 1×1), and 3×3 max-pool (followed by 1×1).
- **Overall Network:** A stack of such Inception modules, interspersed with occasional max-pooling layers for spatial reduction.
- **Classifier:** Concludes with an average pooling layer, a fully connected layer, and a softmax output.

- **Architecture Diagram**



4. MobileNet (Transfer Learning)

4.1. Original Paper Reference

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. Google Research.

4.2. Core Architectural Principles

- Built upon *Depthwise Separable Convolutions*, which factorize a standard convolution into two distinct operations: a depthwise convolution (applying a single filter per input channel) followed by a pointwise convolution (1×1 convolution to combine channel outputs).
- Introduces two hyperparameters—Width Multiplier and Resolution Multiplier—to explicitly trade off between latency/accuracy and model size/accuracy.

4.3. Architecture Flow

- **Input:** Standard RGB image input.
- **Core Building Block:** A sequence of layers comprising: Depthwise Convolution → Batch Normalization → ReLU → Pointwise Convolution → Batch Normalization → ReLU.
- **Overall Structure:** A linear stack of these separable convolution blocks with gradually increasing filter numbers and periodic strides for spatial downsampling.
- **Classifier:** Final global average pooling, a fully connected layer, and a softmax activation.

- **Architecture Diagram**