Popular CNN Architectures

What is a Pre-trained CNN?

A pre-trained CNN is a convolutional neural network model that has already been trained on a large dataset, and whose learned weights you can reuse for new tasks like classification, detection, or feature extraction. These models save huge amounts of time and compute and are central to transfer learning in computer vision.

A normal CNN (like a simple custom model) typically has:

- A few stacks of conv + pool layers
- Followed by a couple of dense (fully connected) layers
- Trained from scratch on a smaller dataset

In contrast, **pre-trained architectures:**

- Are trained on huge datasets (e.g., ImageNet)
- Often **much deeper** with innovative blocks (Inception modules, residual blocks, separable convs)
- Use architectural tricks to reduce computation or enable depth (skip connections, separable convs)
- Can be *used as backbones* and fine-tuned for new tasks

Popular CNNs

1. **LeNet-5** (1948–1998)
    - **What it is:** One of the earliest CNNs, designed for digit recognition.
    - **How it works:** Stack of convolution + subsampling (pooling) + tanh activations + fully-connected layers.
    - **Key difference vs. traditional CNN:** It established the basic CNN building blocks (conv + pool + FC) but was much **shallower and simpler**.
    - **Read more about it here:** https://en.wikipedia.org/wiki/LeNet

2. **AlexNet** (2012)
    - **What it is:** First large CNN to win the ImageNet challenge, showing deep learning's power.
    - **How it works:** 5 convolutional layers + 3 fully connected layers, with ReLU, dropout, data augmentation.
    - **Difference:** Much deeper and larger than classic CNNs. Introduced **ReLU activation** and **GPU training** as practical innovations.
    - **Read more about it here:** https://en.wikipedia.org/wiki/AlexNet

3. **VGGNet (VGG16 / VGG19)** (2014)
    - **What it is:** Deep network with uniform 3×3 filters, 16 or 19 weight layers.
    - **How it works:** Repeated small convolutions improve the receptive field with fewer parameters per layer.
    - **Difference:** Much **deeper** than AlexNet; smaller filters than earlier models.
    - **Read more about it here:** https://en.wikipedia.org/wiki/VGGNet

4. **GoogLeNet / Inception** (2014)
    - **What it is:** Network that uses *Inception modules* i.e. parallel convs with different filter sizes.
    - **How it works:** Inception blocks apply 1×1, 3×3, 5×5 convs in parallel + pooling, then concatenate.
    - **Difference: Parallel paths** in a layer instead of simple stacking, reducing parameters and increasing feature diversity.
    - **Read more about it here:** https://en.wikipedia.org/wiki/Inception_(deep_learning_architecture)

5. **ResNet (Residual Networks)** (2015)
    - **What it is:** Introduces *residual blocks* to allow very deep CNNs (50+ layers).
    - **How it works:** Skip (identity) connections let gradients flow easily through deep layers.
    - **Difference:** Solves *vanishing gradient* problems to train extremely deep networks.
    - **Common usage:** ResNet-50, ResNet-101, etc.
    - **Read more about it here:** https://en.wikipedia.org/wiki/Residual_neural_network

6. **MobileNet** (2017)
    - **What it is:** CNN optimized for mobile/embedded devices.
    - **How it works:** Uses *depthwise separable convolutions* (splits standard conv into depthwise + pointwise), drastically reducing computation.
    - **Difference vs. normal CNN:** Lightweight architecture that trades off some accuracy for speed/efficiency.
    - **Read more about it here:** https://en.wikipedia.org/wiki/MobileNet

7. **Xception** (2016)
    - **What it is:** Interprets Inception modules as depthwise separable convolutions.
    - **How it works:** It replaces Inception inside with full depthwise separable convolution blocks.
    - **Difference:** More efficient parameter use than standard Inception.
    - **Read more about it here:** https://arxiv.org/abs/1610.02357

8. **SqueezeNet** (2016)
    - **What it is:** CNN designed to achieve AlexNet-level accuracy with far fewer parameters (~1.2M).
    - **How it works:** “Fire modules” use 1×1 convs to squeeze features then expand them.
    - **Difference:** Ultra-small model for resource-limited environments.
    - **Read more about it here:** https://en.wikipedia.org/wiki/SqueezeNet

9. **DenseNet** (2017 to current)
    - **What it is:** CNN where *every layer connects to all subsequent layers*.
    - **How it works:** Each layer receives all previous feature maps as input → *feature reuse*.
    - **Difference:** Reduces vanishing gradients and boosts efficiency compared to vanilla stacking.
    - **Read more about it here:** https://pytorch.org/hub/pytorch_vision_densenet/

10. **EfficientNet** (2019 to current)
    - **What it is:** Efficient network that systematically scales depth, width, and resolution.
    - **How it works:** Uses a compound coefficient to scale all three dimensions instead of ad-hoc scaling.
    - **Difference:** Very good accuracy per parameter/compute.
    - **Read more about it here:** https://en.wikipedia.org/wiki/EfficientNet