Pre-trained models for Image classification

https://www.geeksforgeeks.org/top-pre-trained-models-for-image-classification/

Pre-trained models for Object detection

1. YOLOv8 (Ultralytics)

- Type: Anchor-free, real-time detector
- Backbone: CSPDarknet
- Pros: Very fast, high accuracy, easy to train
- Use Case: Real-time detection (drones, surveillance, etc.)
- Model Zoo: YOLOv8 GitHub

2. Faster R-CNN (ResNet50/101-FPN)

- Type: Two-stage detector (Region Proposal + Classification)
- Pros: High accuracy, widely used in research
- Use Case: High-precision tasks (medical imaging, satellite)
- Available in: TorchVision

3. SSD (Single Shot MultiBox Detector) MobileNetV2

- Type: Single-shot detector
- Pros: Fast, lightweight, good for edge devices
- Use Case: Mobile/embedded systems
- Available in: TF Model Zoo

4. EfficientDet (D0-D7)

- Type: Scalable, efficient architecture
- Pros: Balances speed & accuracy, good for scaling
- Use Case: General-purpose detection
- Available in: TensorFlow Hub

5. RetinaNet (ResNet50-FPN)

- Type: Single-shot, FPN-based
- Pros: Handles class imbalance well (uses Focal Loss)
- Use Case: Dense object detection
- Available in: TorchVision

6. Mask R-CNN (ResNet50/101-FPN)

- Type: Extends Faster R-CNN for instance segmentation
- Pros: Detects objects + segmentation masks
- Use Case: Autonomous driving, medical imaging
- Available in: Detectron2

7. DETR (Detection Transformer)

- Type: Transformer-based (end-to-end)
- Pros: No anchor boxes, simpler pipeline
- Use Case: Research, novel architectures
- Available in: HuggingFace

8. YOLO-NAS (by Deci AI)

- Type: Neural Architecture Search-optimized YOLO
- Pros: Better accuracy than YOLOv8, optimized inference
- Use Case: Production deployment
- Available in: SuperGradients

9. CenterNet (Objects as Points)

- Type: Keypoint-based detection
- Pros: Simple, efficient, good for tracking
- Use Case: Pedestrian detection, sports analytics
- Available in: TF Model Zoo

10. Swin Transformer (Swin-T/Swin-B)

- Type: Transformer-based, hierarchical feature maps
- Pros: State-of-the-art accuracy
- Use Case: High-accuracy applications
- Available in: MMDetection

Model	Speed	Accuracy	Best For
YOLOv8	Fast	High	Real-time apps
Faster R-CNN	Slow	High	High-precision tasks
SSD MobileNet	Fast	Medium	Edge devices
EfficientDet	Medium	High	Scalable deployment
RetinaNet	Medium	High	Handling class imbalance
Mask R-CNN	Slow	High	Segmentation + detection
DETR	Slow	High	Research, transformer-based
YOLO-NAS	Fast	High	Optimized production
CenterNet	Fast	Medium	Keypoint-based detection
Swin Transformer	Slow	Very High	Cutting-edge accuracy

Pretrained models for facial recognition

1. FaceNet (Google)

- Architecture: Inception-ResNet-v1 / MobileNet
- Key Feature: Uses triplet loss for embedding learning
- Pros: High accuracy, widely used in production
- Available in: TensorFlow Hub

2. DeepFace (Facebook)

- Backbone: VGG-Face, Facenet, OpenFace
- Key Feature: Supports multiple face recognition models
- Pros: Easy-to-use API, good for research
- Repo: DeepFace GitHub

3. ArcFace (InsightFace)

- Architecture: ResNet100 / MobileNet
- Key Feature: Angular margin loss for better discrimination
- Pros: SOTA accuracy (99.8% on LFW)
- Available in: InsightFace GitHub

4. VGGFace (Oxford)

- Backbone: VGG16, ResNet50, SENet50
- Key Feature: Trained on 2.6M faces (VGGFace2 dataset)
- Pros: Strong baseline model
- Repo: VGGFace2 GitHub

5. OpenFace (CMU)

- Backbone: Inception-v1 (GoogLeNet)
- Key Feature: Lightweight, real-time capable
- Pros: Works well on low-power devices
- Repo: OpenFace GitHub

6. Dlib (ResNet-based)

- Model: Dlib's ResNet-34 with 29M parameters
- Key Feature: Uses MMOD loss (max-margin object detection)
- Pros: Robust, works well with small datasets
- Repo: Dlib GitHub

7. SphereFace (CVPR'17)

- Architecture: Sphere20, Sphere64
- Key Feature: Angular softmax loss for hypersphere embedding
- Pros: High discriminative power
- Repo: SphereFace GitHub

8. SFace (CVPR'19)

- Backbone: Light CNN
- Key Feature: Selective loss for noisy datasets
- Pros: Robust to label noise
- Paper: SFace CVPR Paper

9. MagFace (CVPR'21)

- Architecture: ResNet100 + adaptive margin
- Key Feature: Magnitude-aware angular margin loss
- Pros: Balances hard/easy samples well
- Repo: MagFace GitHub

10. CurricularFace (CVPR'20)

- Backbone: ResNet100
- Key Feature: Curriculum learning strategy in loss
- Pros: Improves training convergence
- Repo: CurricularFace GitHub

Performance Comparison (LFW Accuracy)

Model	Accuracy (LFW)	Speed (FPS)	Best For
ArcFace	99.83%	30	High-accuracy systems
FaceNet	99.65%	25	Production deployments
VGGFace2	99.40%	20	Research baselines
DeepFace	99.50%	15	Easy-to-use API
OpenFace	93.80%	60	Real-time (edge devices)
Dlib	99.38%	10	Small datasets
MagFace	99.80%	25	Noisy datasets

Common CNN Architectures

1. LeNet-5 (1998)

- Creators: Yann LeCun et al.
- Use Case: Digit recognition (e.g., MNIST)
- Key Features:
    - Early CNN architecture with 2 conv layers and 2 subsampling (pooling) layers
    - Fully connected layers at the end

2. AlexNet (2012)

- Creators: Alex Krizhevsky et al.
- Use Case: ImageNet classification (won 2012)

- Key Features:
    - 8 layers deep (5 conv + 3 FC)
    - ReLU activation
    - Dropout to reduce overfitting
    - GPU acceleration

3. VGGNet (2014)

- Creators: Visual Geometry Group, Oxford
- Key Features:
    - Very deep (16 or 19 layers)
    - 3x3 convolution filters
    - Simplicity & uniform architecture
    - Heavy on parameters → computationally expensive

4. GoogLeNet / Inception (2014)

- Creators: Google
- Key Features:
    - Inception modules (multi-scale convolutions in parallel)
    - 22 layers deep
    - Reduced parameters with 1x1 convolutions

5. ResNet (2015)

- Creators: Microsoft Research
- Key Features:
    - Introduced Residual Connections (skip connections)
    - Enabled training of very deep networks (up to 152 layers)

- Solved vanishing gradient problem

6. DenseNet (2016)

- Key Features:
  - Dense connections: each layer receives inputs from all previous layers
  - Better feature reuse
  - Fewer parameters than traditional deep CNNs

7. MobileNet (2017)

- Use Case: Mobile & embedded vision
- Key Features:
  - Depthwise separable convolutions
  - Lightweight & efficient
  - Ideal for edge devices

8. EfficientNet (2019)

- Creators: Google Brain
- Key Features:
  - Balances depth, width, and resolution (compound scaling)
  - Very high performance with fewer parameters
  - State-of-the-art accuracy/efficiency tradeoff

9. Vision Transformers (ViT) – Though not a CNN

- CNNs started being challenged by Transformers in vision tasks.
- ViT uses self-attention mechanisms instead of convolutions.

Architecture	Year	Depth	Params (M)	Key Features	Best Use Case
LeNet-5	1998	7	~0.06	First CNN, simple design	Digit recognition (e.g., MNIST)
AlexNet	2012	8	~60	ReLU, Dropout, GPU training	Image classification (ImageNet)
VGG-16	2014	16	~138	3x3 conv, deep and uniform	Transfer learning, feature extraction
GoogLeNet	2014	22	~7	Inception module, efficient	Real-time image classification
ResNet-50	2015	50	~25	Residual blocks, very deep	Object detection, segmentation
DenseNet	2016	121	~8	Dense connections, compact	Medical imaging, anomaly detection
MobileNet	2017	~28	~4.2	Lightweight, mobile-ready	Mobile apps, edge devices
EfficientNet-B0	2019	18	~5.3	Compound scaling, efficient	High-accuracy, low-resource tasks