Here's a comparison table for various computer vision models based on characteristics such as kernel type (depthwise, atrous, deformable, regular), kernel size, stride, connections, parameters, and other architectural details. This table highlights how each model’s structure supports different computational or functional requirements.

Model	Kernel Type	Kernel Size	Stride	Connections	Parameters	Pooling	Special Characteristics
LeNet	Regular	(5 \times 5)	1	Sequential fully connected	Low (~60k)	(2 \times 2) average	Early CNN for digit classification (MNIST); simple and efficient.
AlexNet	Regular	(11 \times 11), (5 \times 5), (3 \times 3)	4 (first layer), 1	Sequential fully connected	High (~62M)	(3 \times 3) max	Uses ReLU and dropout to prevent overfitting; good for large-scale images.
VGG	Regular	(3 \times 3)	1	Sequential	Very high (~138M)	(2 \times 2) max	Uniform architecture with deep, consistent convolution layers.
GoogLeNet (Inception)	Regular, (1 \times 1) for reduction	(1 \times 1), (3 \times 3), (5 \times 5)	Varied per inception module	Inception modules	Moderate (~6.8M)	Max and average	Multi-scale features with parallel convolution paths.
ResNet	Regular, (1 \times 1) for bottleneck layers	(3 \times 3), (1 \times 1)	2 (initially), 1 in blocks	Residual connections (skip)	High (~25M for ResNet-50)	Global average in final layer	Skip connections allow very deep networks, solving vanishing gradient issues.
DenseNet	Regular, (1 \times 1) for bottleneck layers	(3 \times 3), (1 \times 1)	1 in dense blocks	Dense connections (layer-to-layer)	Lower (~7.2M for DenseNet-121)	Global average	Dense connections reduce redundancy, improve gradient flow.
MobileNet	Depthwise separable	(3 \times 3) depthwise, (1 \times 1) pointwise	Varied per layer	Depthwise separable convolutions	Low (~4.2M for MobileNetV1)	Global average	Optimized for mobile devices with lightweight convolutions.
EfficientNet	Depthwise separable	(3 \times 3), (5 \times 5)	Varies per stage	Compound scaling (width, depth, resolution)	Varies (5M - 66M)	Global average	Efficient scaling from B0 to B7 allows for balanced resource use.
YOLO	Regular	(1 \times 1), (3 \times 3)	Varies	Fully convolutional (detection)	Moderate (~65M)	None	Real-time object detection; single forward pass for fast inference.

Model	Kernel Type	Kernel Size	Stride	Connections	Parameters	Pooling	Special Characteristics
DeepLab	Atrous (dilated)	(3 \times 3)	1	Dilated convolutions	High (varies)	None, uses atrous convolutions	Dilated convolutions capture large context in segmentation.
Deformable ConvNets	Deformable	(3 \times 3), flexible offsets	1	Adaptive offsets	High (varies)	None	Flexible receptive fields improve handling of object deformations.

Summary of Key Differences

- **Kernel Type**:
  - **Regular**: Basic convolutional layers (e.g., LeNet, AlexNet).
  - **Depthwise**: Reduces computation by separating spatial and channel-wise convolutions (e.g., MobileNet, EfficientNet).
  - **Atrous**: Dilated convolutions capture larger context without downsampling (e.g., DeepLab).
  - **Deformable**: Allows flexible receptive fields to adjust for object shape and scale (e.g., Deformable ConvNets).
- **Kernel Size**:
  - **VGG** and **ResNet** primarily use (3 \times 3) kernels for deeper architectures, while **AlexNet** and **Inception** use a variety of kernel sizes for multi-scale feature capture.
- **Connections**:
  - **Skip Connections** in **ResNet** and **Dense Connections** in **DenseNet** mitigate gradient vanishing and allow deep layers.
  - **Inception Modules** in **GoogLeNet** allow multi-scale processing within a single layer.
- **Parameters**:
  - **EfficientNet** uses compound scaling for efficient resource use across different model sizes, while **MobileNet** optimizes for mobile use.
  - **YOLO** maintains a moderate parameter count, focusing on speed for object detection.

This table captures the design principles of each model, which enable them to excel in different computer vision tasks.