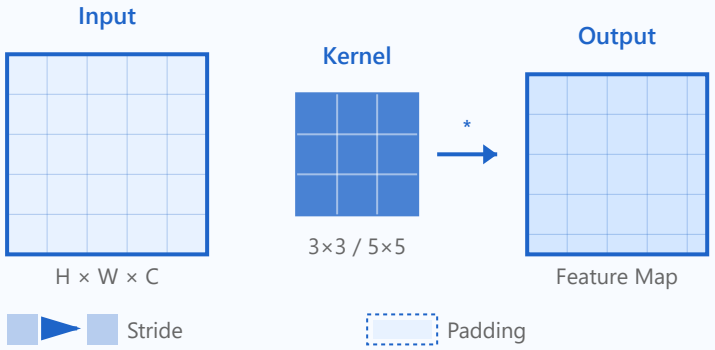
Convolution Operation

Convolution Process



Kernel/Filter Concepts

Small learnable matrices that slide across input to extract features. Common sizes: 3×3, 5×5, 7×7

Stride and Padding

Stride: step size of kernel movement. Padding: adding borders to preserve spatial dimensions

Feature Map Generation

Output of convolution operation. Each filter produces one feature map detecting specific patterns

Receptive Fields

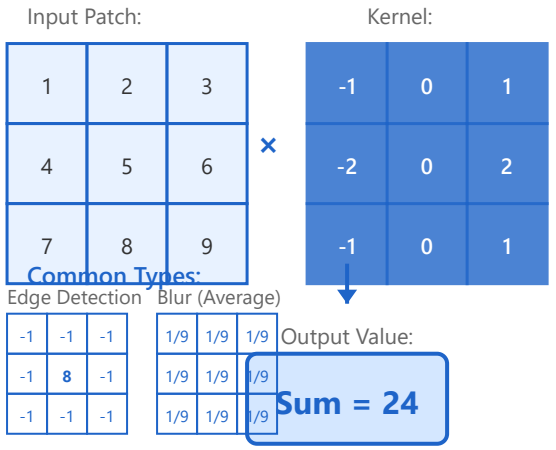
Region of input that affects a particular feature. Grows with network depth and kernel size

Special Case: **1×1 Convolutions** - Channel-wise operations for dimensionality reduction and cross-channel learning

Detailed Explanation of Convolution Concepts

1 Kernel/Filter Concepts

3×3 Kernel Example



Element-wise multiplication and summation

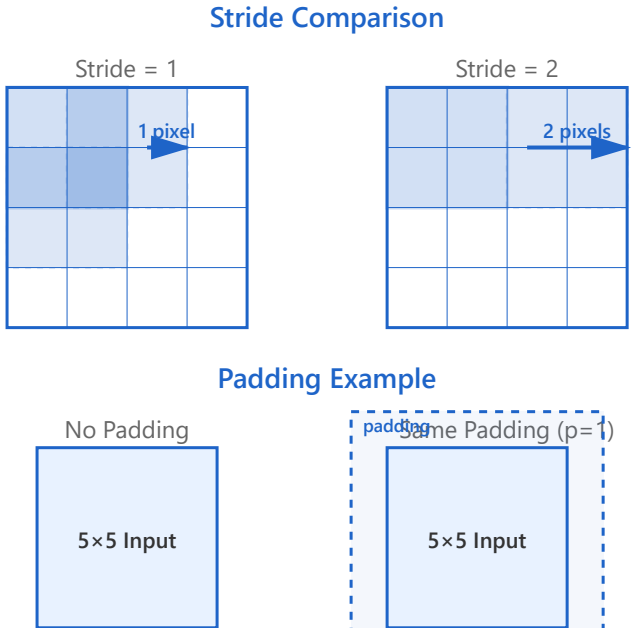
Kernels (also called filters) are small matrices of learnable weights that slide across the input image to extract features. The convolution operation performs element-wise multiplication between the kernel and the input patch, then sums all values to produce a single output value.

- **3×3 kernels:** Most commonly used in modern architectures (VGG, ResNet). Efficient and can capture local patterns while being computationally efficient
- **5×5 kernels:** Used in earlier networks (AlexNet). Capture wider spatial context but require more computation
- **7×7 kernels:** Typically used only in the first layer for large input images to quickly reduce spatial dimensions
- **Learnable weights:** Kernel values are learned through backpropagation during training to detect specific features like edges, textures, or complex patterns

- **Multiple channels:** For RGB images, kernels have depth equal to input channels (e.g., 3×3×3 for RGB)

```
Output = Σ(Input[i,j] × Kernel[i,j]) + bias
```

2 Stride and Padding



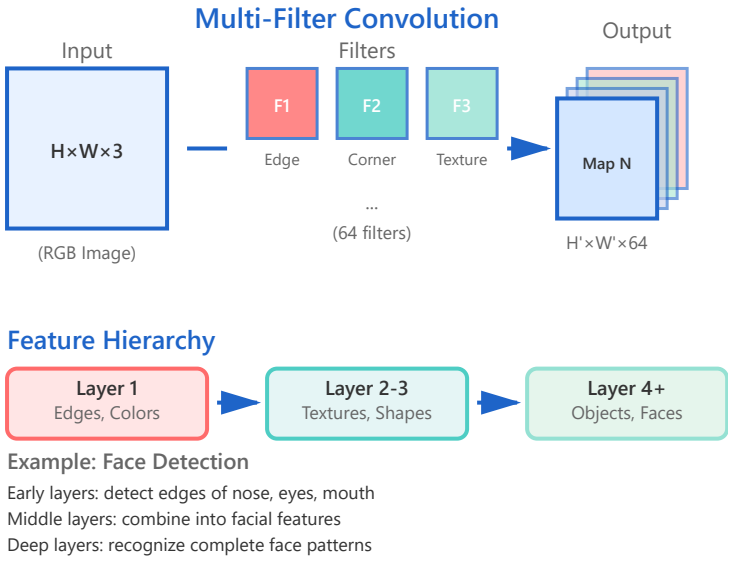
Impact of stride and padding on output size

Stride and padding are hyperparameters that control how the kernel moves across the input and how spatial dimensions are preserved or reduced.

- **Stride:** The number of pixels the kernel moves at each step. Stride=1 produces dense feature maps with maximum spatial resolution. Stride=2 reduces spatial dimensions by half, acting as downsampling
- **Valid padding (p=0):** No padding added. Output size decreases based on kernel size. Loses information at borders
- **Same padding:** Adds zeros around the border to maintain input spatial dimensions. Commonly used to preserve resolution throughout the network
- **Computational impact:** Larger stride reduces computation and memory requirements but may lose fine-grained spatial information
- **Trade-offs:** Smaller stride = more detail but higher cost; Larger stride = faster but coarser features

```
Output Size = ⌊(Input - Kernel + 2×Padding) / Stride⌋ + 1
```

3 Feature Map Generation



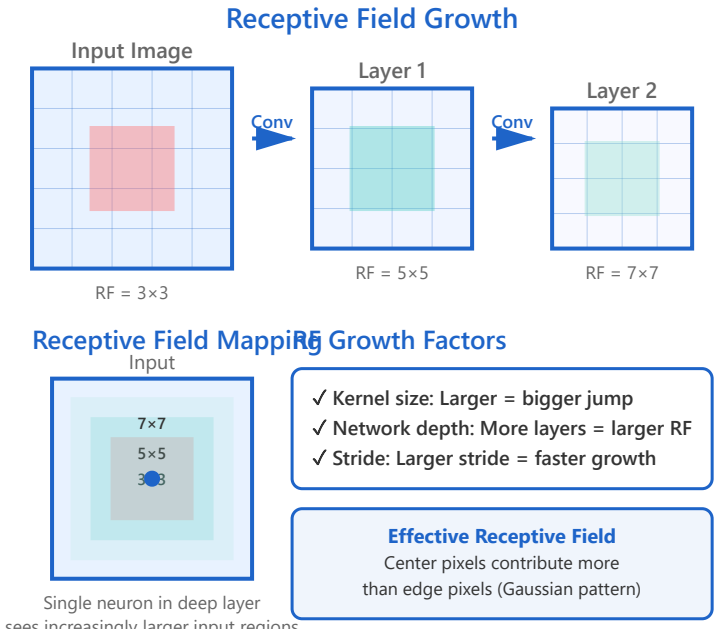
Multiple filters create feature map stack

Feature maps are the outputs produced by applying filters to the input. Each filter detects specific patterns and creates one feature map. Multiple filters run in parallel to extract diverse features.

- **One filter → One feature map:** Each convolutional filter produces a single 2D feature map (spatial dimensions H'×W')
- **Multiple filters:** Using N filters creates N feature maps, stacked to form a 3D output tensor (H'×W'×N)
- **Feature hierarchy:** Early layers detect simple features (edges, colors). Deeper layers combine these into complex patterns (textures, objects)
- **Activation patterns:** High values in a feature map indicate the presence of the pattern that filter is trained to detect
- **Depth increase:** Networks typically increase channel depth while reducing spatial dimensions (e.g., 224×224×3 → 112×112×64 → 56×56×128)
- **Spatial preservation:** Feature maps maintain spatial relationships from input, enabling localization tasks

```
Input (H×W×C_in) * Filters
(K×K×C_in×C_out) → Output (H'×W'×C_out)
```

4 Receptive Fields



Receptive field expands through network layers

The receptive field is the region in the input image that affects a particular neuron's output in a layer. It grows as we go deeper in the network, allowing neurons to aggregate information from increasingly larger spatial contexts.

- **Definition:** The area of the input image that influences the activation of a single neuron in a given layer
- **Growth mechanism:** Each convolutional layer increases the receptive field. A neuron in layer N sees a larger region than one in layer N-1
- **Calculation:** For a 3×3 kernel with stride 1: Layer 1 RF = 3×3, Layer 2 RF = 5×5, Layer 3 RF = 7×7, etc.
- **Effective receptive field:** Not all pixels in the theoretical RF contribute equally. Central pixels have stronger influence (Gaussian distribution)
- **Importance:** Larger receptive fields enable understanding of global context, crucial for tasks like object recognition and semantic segmentation
- **Design consideration:** Deep networks with small kernels (VGG) vs shallow networks with large kernels

(AlexNet) achieve similar receptive fields with different computational trade-offs

```
RF_out = RF_in + (Kernel_size - 1) × ∏(Strides_prev_layers)
```