Applied Machine Learning

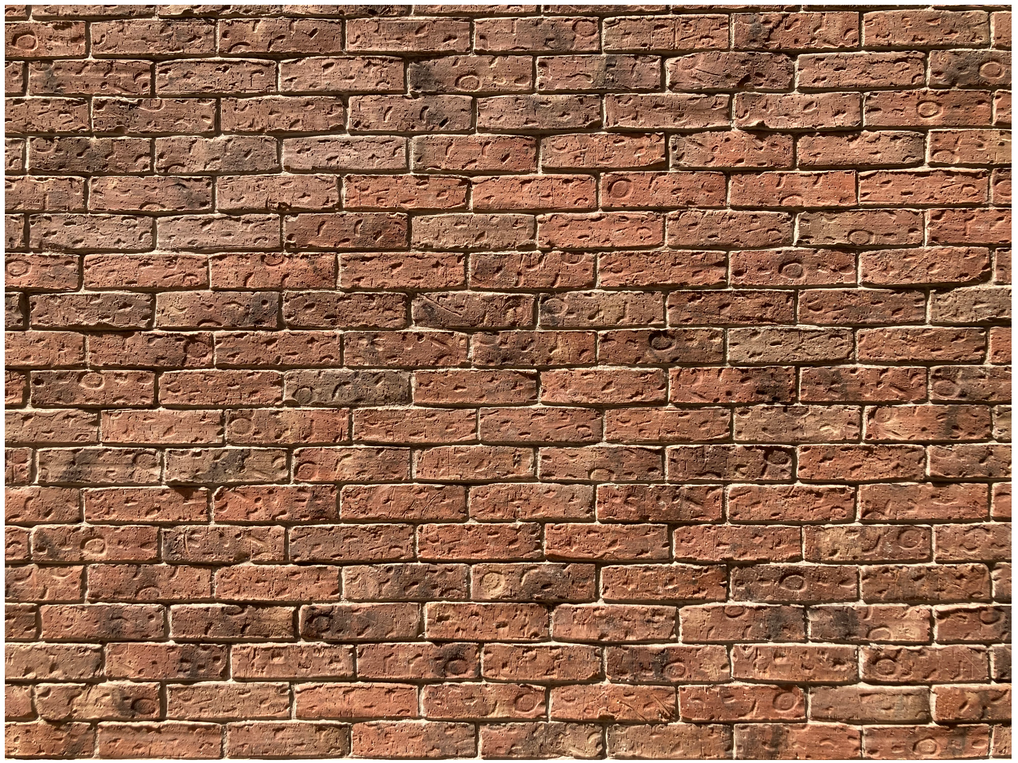
Convolutional Layer

Convolutional Layer

- Feature Maps
- From convolutional kernel to linear unit
- Convolutional Layer
- Pooling Layer

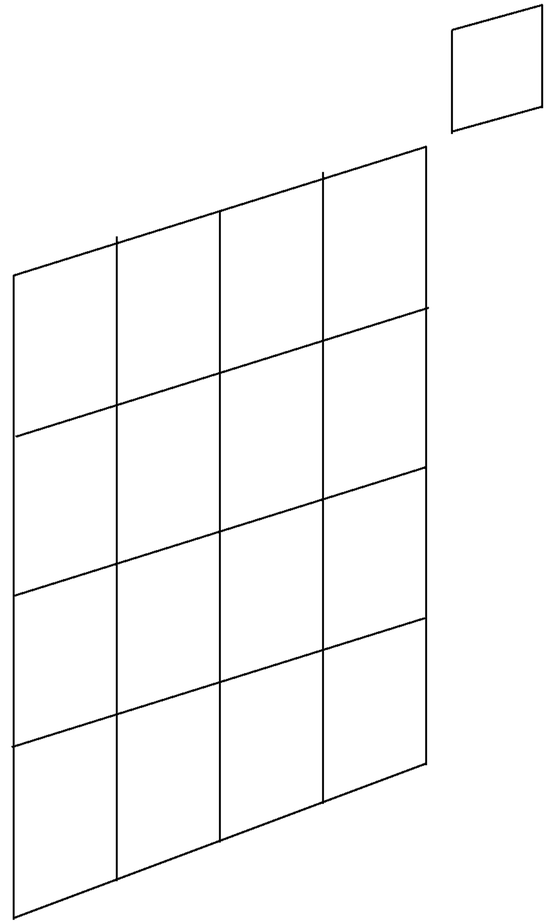
3-Dimensional Convolution

- Color image I
  - 3 planar channels: Red, Green, Blue (RGB): I_r, I_g, I_b
  - stacked in 3D array or Tensor, 1 slice per channel
- 3D convolution
  - Similar to 2D convolution
    - Kernel: 3D array or Tensor: 1 slice per channel
  - $N_{u,v}$: weighted sum within pixels defined by 3D Kernel around pixel (u, v)
    - 2-dimensional
- same interpretation as 2D convolution
- same properties
- similar padding and stride strategies



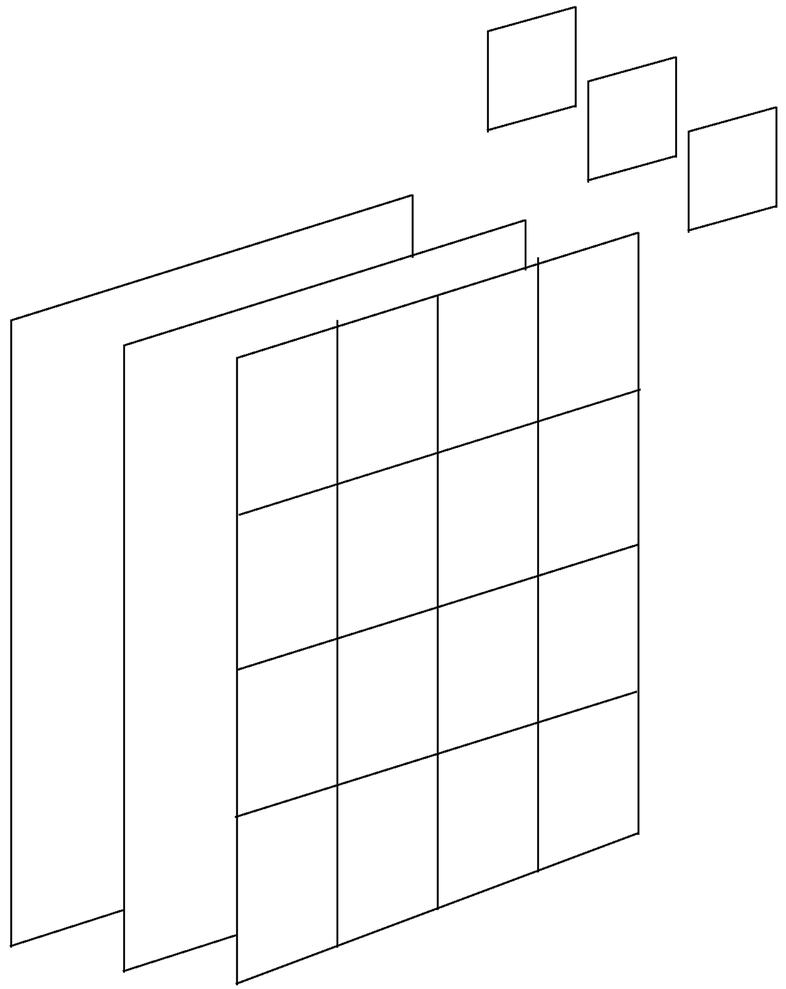
Feature Map

- Input: Image I of size $x \times y$ with D channels
  - 3-D: $x \times y \times D$
- Collection of kernels K: stack of L Kernels
  - K_l: Kernel for pattern $l \in \{1 \ldots L\}$
    - dimensions: same as input I: $k_i \times k_j \times D$
  - dimensions: $k_i \times k_j \times D \times L$
- Output: Stack of L responses: $N_l = I \otimes K_l$
  - 3-dimensional: $x_n \times y_n \times L$
    - x_n, y_n: depend on strides and padding



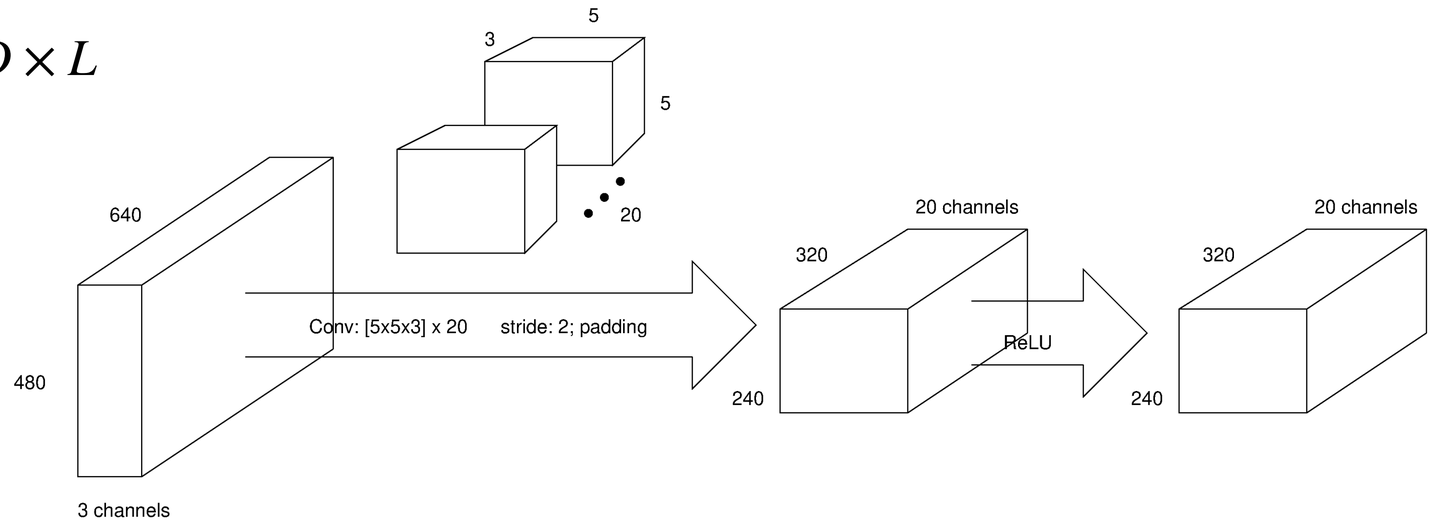
Convolutional Layer

- Linear Unit: $u = \sum_i w_i x_i + b$
- ReLU: $F(u) = \max(0, u)$
- Convolution: $N_{u,v} = \sum_{(i,j) \in K} I_{u+i,v+j} K_{i,j}$
- Linear unit implements convolution:
  - bias b: tune level of similarity for positive output
- Convolutional Layer
  - compute Feature Maps
  - learn kernel weights



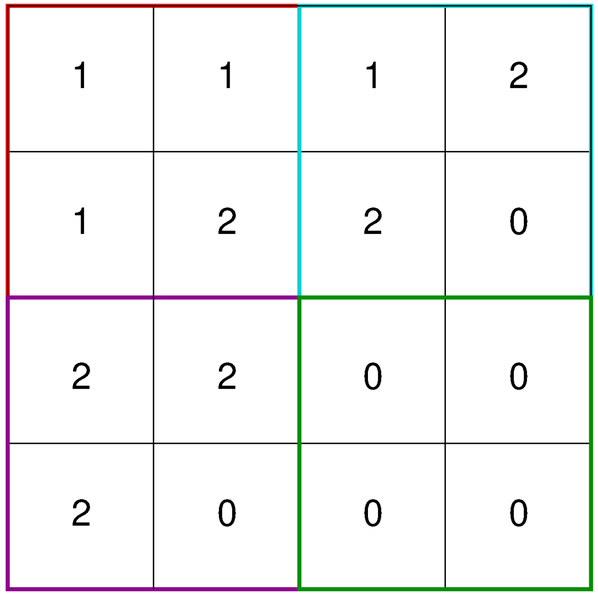
Convolutional Layer

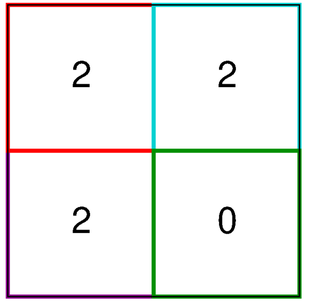
- Input: I of size $x \times y$ with D channels
  - 3-D: $x \times y \times D$
- Parameters: Shared among the $x \times y \times L$ units
  - Kernel K weights: dimensions: $k_i \times k_j \times D \times L$
  - Bias vector components: L
- Output: Stack of L convolutional responses
  - 3-dimensional: $x_n \times y_n \times L$
    - x_n, y_n: depend on strides and padding
- ReLU layer



Pooling Layer

- Input: Feature map: $x_n \times y_n \times L$
- regions of adjacent pixels
  - same features activate for most
  - some features do not activate at all
- Pooling: capture patterns in groups of pixels
  - Patch of size $\{p_x \times p_y\}$ Stride lenght: s
  - Output: Feature map: $\lfloor \frac{x_n}{s} \rfloor \times \lfloor \frac{y_n}{s} \rfloor \times L$
  - Pooling function for inputs I that overlap patch
    - Maximum Pooling: maximum value in patch
    - Average Pooling: average value in patch
- Typical sequence of layers: Convolutional => ReLU => Pooling
- Pooling is robust





Max

Avg

Convolutional Layer

- Feature Maps
- From convolutional kernel to linear unit
- Convolutional Layer
- Pooling Layer

Applied Machine Learning

Convolutional Layer