Deep Learning

How Data Scientists become magicians

Convolutions

$$\mathbf{Y} = \mathbf{K} * \mathbf{X}.$$

Calculating the dot product over patches of **X** of similar size to **K**.

Convolution Hyperparameters

- The filter dimensions.
- The filter values.
- The stride between patches.
- The padding of the input.

Demo (see 2.3)

Convolution Output Dimensions

$$\mathbf{Y}_w = \frac{\mathbf{X}_w - \mathbf{K}_w + 2p}{s} + 1,$$
$$\mathbf{Y}_h = \frac{\mathbf{X}_h - \mathbf{K}_h + 2p}{s} + 1,$$

Example Filters - Identity

$$\mathbf{K}_{\text{Identity}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

returns the original image.

Example Filters - Gaussian Blur

$$\mathbf{K}_{\text{Gaussian Blur}} = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}$$

adds a blur to the image following the Gaussian distribution.

Example Filters - Box Blur

$$\mathbf{K}_{\text{Box Blur}} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

adds a blur to the image equally across the point's neighbors.

Example Filters - Sharpen

$$\mathbf{K}_{\text{Sharpen}} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

sharpens the image by adding some contrast between the pixel and its neighbors.

Example Filters - Line Detection

$$\mathbf{K}_{\text{Horizontal Lines}} = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}, \mathbf{K}_{\text{Vertical Lines}} = \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix},$$

$$\mathbf{K}_{45^\circ \text{ lines}} = \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}, \mathbf{K}_{\text{-}45^\circ \text{ lines}} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

detect lines at a particular angle in the image.

Example Filters - Outline Detection

$$\mathbf{K}_{\text{Outline}} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

detect key outlines within an image.

Example Filters - Gradient

$$\mathbf{K}_{x\ \text{gradient}} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \mathbf{K}_{y\ \text{gradient}} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix},$$

Example Filters - Sobel

$$\mathbf{K}_{x \text{ gradient}} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \mathbf{K}_{y \text{ gradient}} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix},$$

$$\mathbf{K}_{\text{Horizontal Sobel}} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \mathbf{K}_{x \text{ gradient}}, \mathbf{K}_{\text{Vertical Sobel}} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \times \mathbf{K}_{y \text{ gradier}}$$

Example Filters - Sobel

$$\mathbf{K}_{\text{Horizontal Sobel}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \mathbf{K}_{\text{Vertical Sobel}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

determine the image gradients.

Demo

Pooling

- Average pooling
- Max pooling

Pooling Example

2×2 filter with a stride of 2.

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 5 & 7 \\ 8 & 9 & 6 & 2 \\ 1 & 1 & 4 & 7 \\ 2 & 0 & 3 & 2 \end{bmatrix},$$

$$\mathbf{P}_{\text{average}} = \begin{bmatrix} 5 & 5 \\ 1 & 4 \end{bmatrix}, \mathbf{P}_{\text{max}} = \begin{bmatrix} 9 & 7 \\ 2 & 7 \end{bmatrix}.$$

Pooling Example - Dimension Check

2×2 filter with a stride of 2.

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 5 & 7 \\ 8 & 9 & 6 & 2 \\ 1 & 1 & 4 & 7 \\ 2 & 0 & 3 & 2 \end{bmatrix},$$

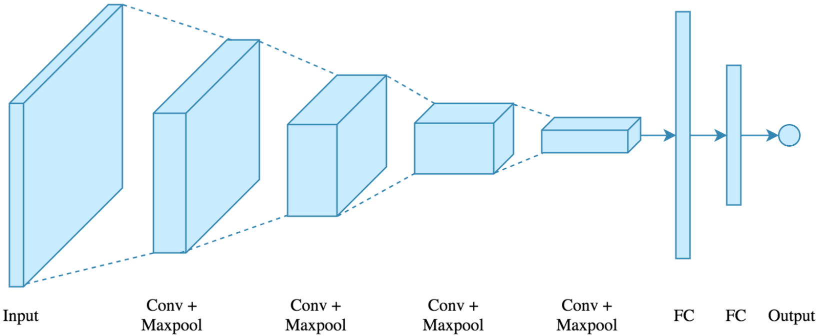
$$\begin{aligned} w &= \frac{\mathbf{X}_w - \mathbf{K}_w + 2p}{s} + 1, \\ &= \frac{4 - 2 + 2 * (0)}{2} + 1, \\ &= 2 \end{aligned}$$

Pooling Issue

Pooling is effective in practice despite being a very simple approach.

Convolutional Neural Networks (CNNs)

CNN


Input
Conv + Maxpool
Conv + Maxpool
Conv + Maxpool
Conv + Maxpool
FC
FC
Output

Convolutional Layer

- Layer has a number of filters, f (hyperparameter)
- Filter values are learned during model training
- Each filter applies to all of the image (shared weights)
- Output is similar to input shape with additional dimension f

Pooling Layer

- Applies pooling to input
- Lowers the dimensionality of the features

Training

The training process is similar to other feed forward neural networks.

- Apply a forward pass to determine a prediction
- Calculate error / loss function
- Update weights based on gradient
- Backpropogate the error and update previous weights

Overfitting

- Dropout
- Start with low number of filters then increase them

Questions