Computational Social Science with Images and Audio

Elliott Ash, **Philine Widmer**

20 October 2023

Let us briefly discuss computer vision and (mis)information.

(Inspired by recent sad events around the world.)

- (Social) Media is flooded with (visual) information on recent tragedies in world politics
  - Lying anywhere between fully verifiable to entirely fake
- Verification takes resources (in particular, time)
  - For one verified or debunked information piece, multiple new ones are posted
- Many actors, many motives
  - Are (social) media platforms interested in truth? Citizens? Politicians?
  - Other motivations than truth?

How can computer vision help/debunk false info?

- First question: is the image as such fake, or is the meta-information (who, where, what) incorrect?
  - For example: deep fake vs. wrong context
- Some helpful approaches:
  - Reverse image search
  - Temporal consistency (image metadata)
  - Image forensics
  - Face detection
  - Deepfake detection
  - Geolocation verification
- Some concerns
  - Sophisticated manipulations
  - Privacy concerns

Let's move back to today's main topic: CNN.

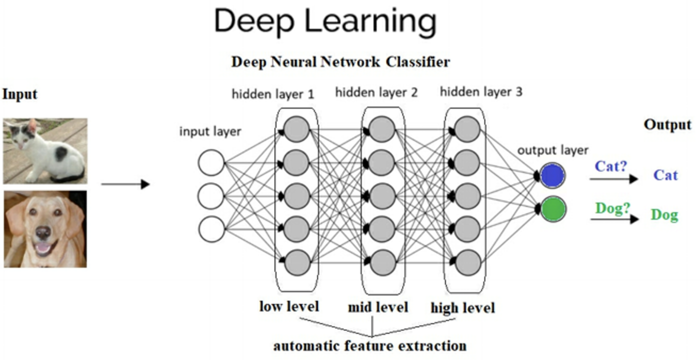

Figure: Dey (2018)[1]

[1] Dey, S. (2018). Hands-On Image Processing with Python: Expert Techniques for Advanced Image Analysis and Effective Interpretation of Image Data. Packt Publishing Ltd.

Convolutional layers use convolution to filter input data.

- With an input matrix I and a filter matrix K, the convolution is:

$$(I * K)(x, y) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} I(i, j) \times K(x - i, y - j)$$

- Consider this example image:

$$I = \begin{bmatrix} 0 & 2 & 50 \\ 55 & 97 & 230 \\ 234 & 235 & 0 \end{bmatrix}$$

- ... and a 3×3 filter:

$$K = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

What is the dimension of convolved output?

- What is the convolved output from the previous slide?
- What dimensions would the output have if the filter was of dimension 2×2? Why?
  - How many times can we put this filter on I above?

$$F = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

  - Hint1: It depends on whether we pad the image (e.g., add some zeros at the borders) and...
  - Hint2: ... on the stride, which defines the step size at which the filter evaluates the next position

Convolution is the basis of hierarchical feature learning.

- Each filter (like K from earlier) produces a unique feature map

$$\text{Image} \xrightarrow{\text{Filter } K} \text{Feature Map}$$

- A CNN layer typically contains multiple filters
- Accordingly, an input can produce multiple feature maps in one convolutional layer
  - In a given layer, if we put filter F_1 and F_2 of 2×2 each (with stride 1 and no padding) on our image I, there will be two feature maps of 2x2 each
  - Hence, the layer will pass two 2x2 feature maps to the next layer
  - Disclaimer: heavily simplified for intuition
- Over training, the network learns the optimal filter values that minimize the loss
- Pooling layers after convolution can down-sample the feature maps

The above intuition generalizes to multi-channel images.

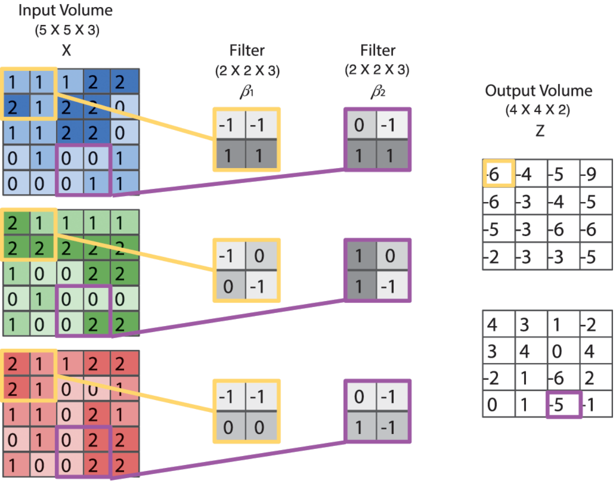

Figure: Convolutional Layer in a CNN for Image Processing (from Webb Williams et al. (2020; cf. first lecture)

CNN come with many hyperparameters.

- **Epochs/iterations:** Number of times the model trains over the full set of images *(e.g., 100 epochs)*
- **Learning rate:** How much weights/coefficients change in optimization: essentially scales the gradient *(e.g., 0.01)*
- **Dropout rate:** To avoid overfitting; share of weights set to 0 *(e.g., 0.5)*
- **Batch size:** Number of samples processed before updating *(e.g., 32 samples)*
- **Ratio of train-validation:** Split of images into sets *(e.g., 80-20 split)*
- **Loss function:** Evaluates model accuracy *(e.g., Mean Squared Error)*

CNN come with many hyperparameters (continued).

- **Activation functions:** Nonlinear transformations (cf. last lecture) *(e.g., ReLU)*
- **Optimizer:** Minimizes the loss function *(e.g., Adam)*
- **Momentum:** Considers previous gradients to avoid stagnation or oscillations *(e.g., 0.9)*
- **Step size:** Defines learning rate decay interval *(e.g., 10 epochs or 10 batches/iterations)*
- **Gamma:** Learning rate decay after every step-size *(e.g., 0.1 decay rate)*