Optimization Problems: Convolutional Networks

Chih-Jen Lin
National Taiwan University

Last updated: April 23, 2021

Why CNN? I

- There are many types of neural networks
- They are suitable for different types of problems
- Note that neural networks may not be always better than other learning methods
- For example, fully-connected networks were evalueated for general classification data (e.g., data from UCI machine learning repository)
- They are not consistently better than random forests or SVM; see the comparisons (Meyer et al., 2003; Fernández-Delgado et al., 2014; Wang et al., 2018).

Why CNN? II

- We are interested in CNN because it's shown to be significantly better than others on image data

Convolutional Neural Networks I

- Consider a K-class classification problem with training data

$$(\boldsymbol{y}^i, Z^{1,i}), \quad i = 1, \ldots, l.$$

$\boldsymbol{y}^i$: label vector $\qquad Z^{1,i}$: input image

- If $Z^{1,i}$ is in class k, then

$$\boldsymbol{y}^i = [\underbrace{0, \ldots, 0}_{k-1}, 1, 0, \ldots, 0]^T \in R^K.$$

- CNN maps each image $Z^{1,i}$ to $\boldsymbol{y}^i$

Convolutional Neural Networks II

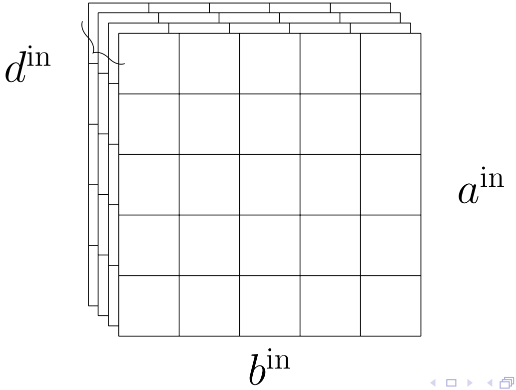
- Typically, CNN consists of multiple convolutional layers followed by fully-connected layers.
- Input and output of a convolutional layer are assumed to be images.

Convolutional Layers I

- For the current layer, let the input be an image

$$Z^{\text{in}} : a^{\text{in}} \times b^{\text{in}} \times d^{\text{in}}.$$

a^{in}: height, b^{in}: width, and d^{in}: #channels.



Convolutional Layers II

The goal is to generate an output image

$$Z^{\text{out},i}$$

of d^{out} channels of $a^{\text{out}} \times b^{\text{out}}$ images.

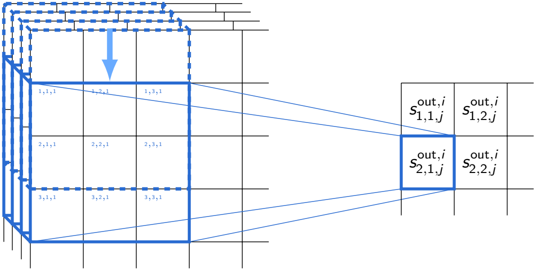
- Consider d^{out} filters.
- Filter $j \in \{1, \ldots, d^{\text{out}}\}$ has dimensions

$$h \times h \times d^{\text{in}}.$$

$$\begin{bmatrix} w^j_{1,1,1} & & w^j_{1,h,1} \\ & \ddots & \\ w^j_{h,1,1} & & w^j_{h,h,1} \end{bmatrix} \ldots \begin{bmatrix} w^j_{1,1,d^{\text{in}}} & & w^j_{1,h,d^{\text{in}}} \\ & \ddots & \\ w^j_{h,1,d^{\text{in}}} & & w^j_{h,h,d^{\text{in}}} \end{bmatrix}.$$

Convolutional Layers III

h: filter height/width (layer index omitted)



- To compute the jth channel of output, we scan the input from top-left to bottom-right to obtain the sub-images of size $h \times h \times d^{\text{in}}$

Convolutional Layers IV

- We then calculate the inner product between each sub-image and the jth filter
- The idea is that this inner product may extract local information of the sub-image
- For example, if we start from the upper left corner of the input image, the first sub-image of channel d is

$$\begin{bmatrix} z^i_{1,1,d} & \dots & z^i_{1,h,d} \\ & \ddots & \\ z^i_{h,1,d} & \dots & z^i_{h,h,d} \end{bmatrix}.$$

Convolutional Layers V

We then calculate

$$\sum_{d=1}^{d^{\text{in}}} \left\langle \begin{bmatrix} z^i_{1,1,d} & \dots & z^i_{1,h,d} \\ & \ddots & \\ z^i_{h,1,d} & \dots & z^i_{h,h,d} \end{bmatrix}, \begin{bmatrix} w^j_{1,1,d} & \dots & w^j_{1,h,d} \\ & \ddots & \\ w^j_{h,1,d} & \dots & w^j_{h,h,d} \end{bmatrix} \right\rangle + b_j, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ means the sum of component-wise products between two matrices.

- This value becomes the $(1, 1)$ position of the channel j of the output image.

Convolutional Layers VI

- Next, we use other sub-images to produce values in other positions of the output image.
- Let the stride s be the number of pixels vertically or horizontally to get sub-images.
- For the $(2, 1)$ position of the output image, we move down s pixels vertically to obtain the following sub-image:

$$\begin{bmatrix} z^i_{1+s,1,d} & \dots & z^i_{1+s,h,d} \\ & \ddots & \\ z^i_{h+s,1,d} & \dots & z^i_{h+s,h,d} \end{bmatrix}.$$

Convolutional Layers VII

- The $(2,1)$ position of the channel j of the output image is

$$\sum_{d=1}^{d^{\text{in}}} \left\langle \begin{bmatrix} z^i_{1+s,1,d} & \cdots & z^i_{1+s,h,d} \\ & \ddots & \\ z^i_{h+s,1,d} & \cdots & z^i_{h+s,h,d} \end{bmatrix}, \begin{bmatrix} w^j_{1,1,d} & \cdots & w^j_{1,h,d} \\ & \ddots & \\ w^j_{h,1,d} & \cdots & w^j_{h,h,d} \end{bmatrix} \right\rangle + b_j. \quad (2)$$

Convolutional Layers VIII

- The output image size a^{out} and b^{out} are respectively numbers that vertically and horizontally we can move the filter

$$a^{\text{out}} = \lfloor \frac{a^{\text{in}} - h}{s} \rfloor + 1, \quad b^{\text{out}} = \lfloor \frac{b^{\text{in}} - h}{s} \rfloor + 1 \quad (3)$$

- Rationale of (3): vertically last row of each sub-image is

$$h, h + s, \ldots, h + \Delta s \leq a^{\text{in}}$$

Convolutional Layers IX

Thus

$$\Delta = \lfloor \frac{a^{\text{in}} - h}{s} \rfloor$$

References I

M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15:3133–3181, 2014.

D. Meyer, F. Leisch, and K. Hornik. The support vector machine under test. *Neurocomputing*, 55:169–186, 2003.

C.-C. Wang, K.-L. Tan, C.-T. Chen, Y.-H. Lin, S. S. Keerthi, D. Mahajan, S. Sundararajan, and C.-J. Lin. Distributed Newton methods for deep learning. *Neural Computation*, 30(6): 1673–1724, 2018. URL http://www.csie.ntu.edu.tw/~cjlin/papers/dnn/dsh.pdf.