Neural net applications for images

Victor Kitov

v.v.kitov@yandex.ru

Convolutional neural networks

- Convolutional neural network:
  - Used for image analysis
  - Consists of a set of convolutional layer / sub-sampling (pooling) layer pairs and several fully connected layers at the end.

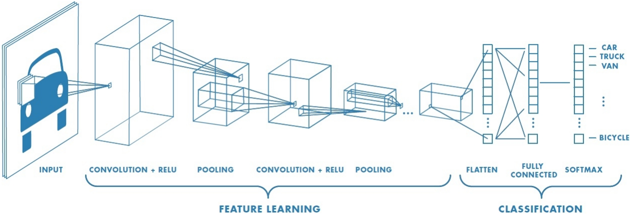
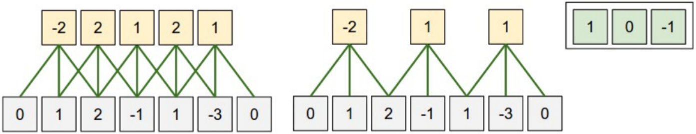


Table of Contents

1-D Convolution operation

1-D convolution [$W = 5$, $K = 3$, zero-padded with $\mathrm{P}{=}1$, $S = 1$ (left) and $S = 2$ (right)]



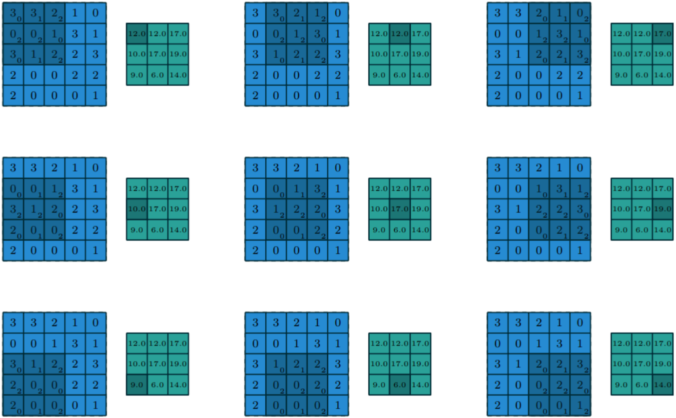
Parameters[1]:

- W - length of input
- K - kernel size
- P - amount of padding
- Type of padding (zero, extension, mirror)
- S - stride (offset of kernel)
- D - dilation (offset inside kernel)

[1] Depending on these parameters, what would be the size of output layer?

Convolution[2]

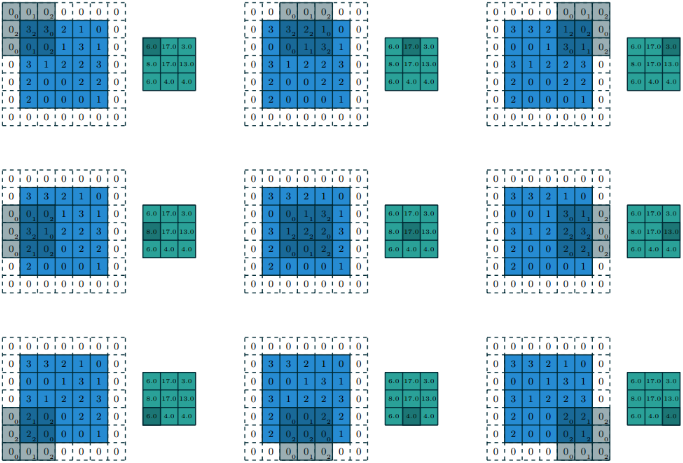
Single layer convolution:



[2]Illustrations from Dumoulin et al. 2018.

Convolution

Convolution with stride and zero-padding:

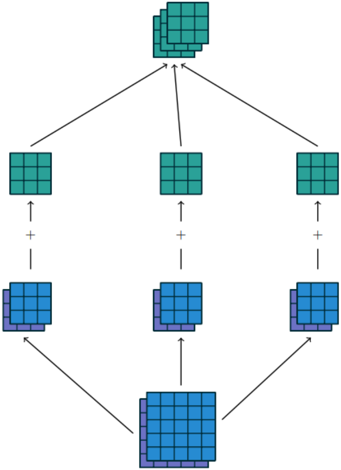


Padding

- Stride: to decrease dimensionality.
- Padding: to increase dimensionality.
- Padding types:
  - zero padding
  - same padding
  - mirror padding

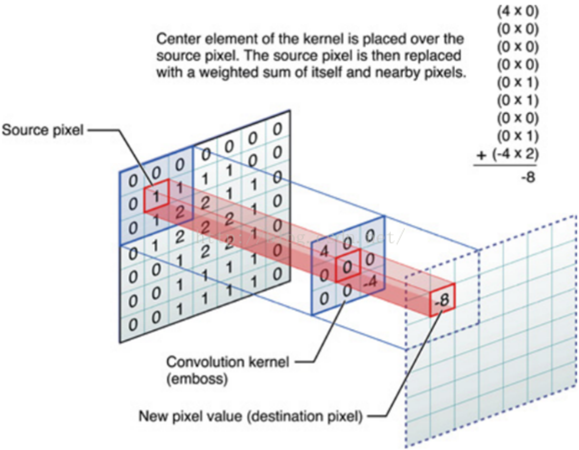
Convolution

2 layer input, 3 layer output convolution:



Convolution operation

2-D convolution

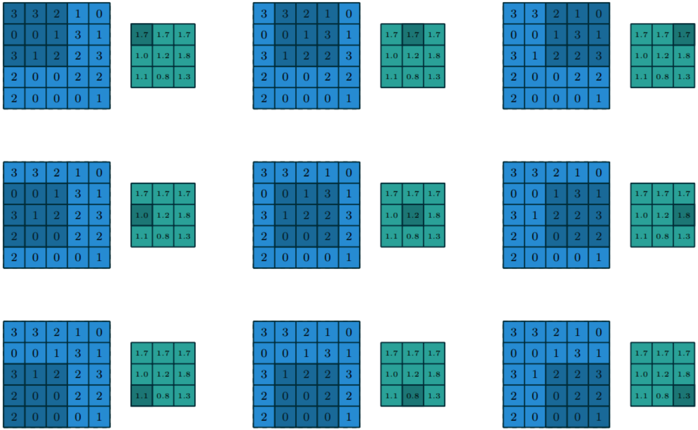


Comments

- Comments on convolution:
  - Locality: each neuron in the feature map takes output from small neighborhood of input layer neurons
  - Equivalence: the same transformation is applied by each neuron in the feature map
    - obtained by constraining sets of weights to each feature map layer neuron to be equal
  - This is feature extraction from a patch

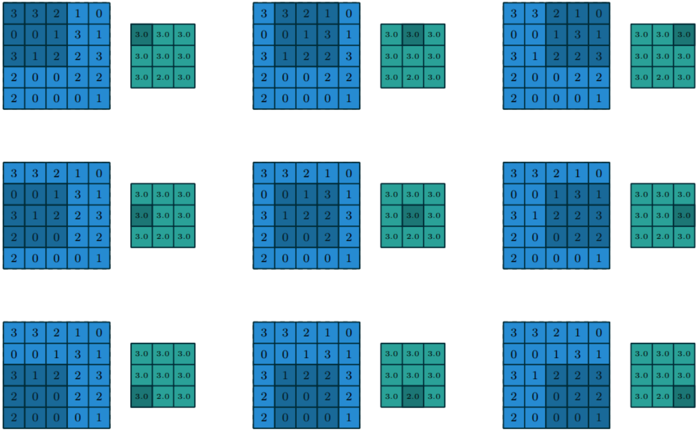
Average pooling

Average 3x3 pooling:



Max pooling

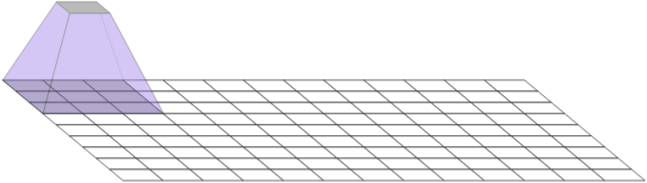
Max 3x3 pooling:



Transposed convolution

- Also known as fractinally strided convolution or deconvolution.
- "Stamps" kernel multiplied by feature intensity

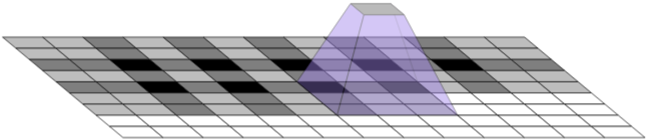
Transpose convolution:



Transposed convolution

- Leads to checkerboard artifacts due to kernel overlaps:

Checkerboard artifacts of transposed convolution:

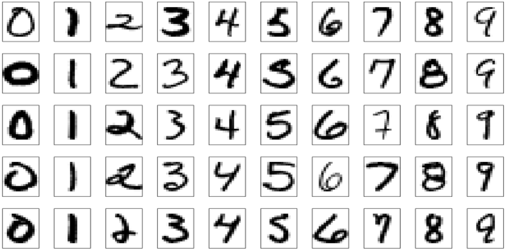


- Ways to avoid:
  - use non-overlapping stride
  - use nearest neighbours upsampling and convolution.

Table of Contents

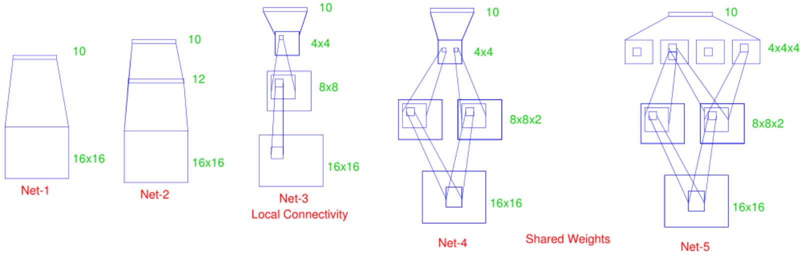
Case study (due to Hastie et al. The Elements of Statistical Learning)

ZIP code recognition task



Neural network structures

Net1: no hidden layer
Net2: 1 hidden layer, 12 hidden units fully connected
Net3: 2 hidden layers, locally connected
Net4: 2 hidden layers, locally connected with weight sharing
Net5: 2 hidden layers, locally connected, 2 levels of weight sharing



Results

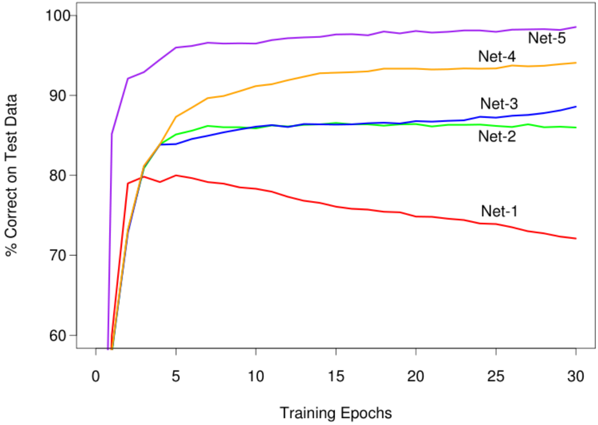
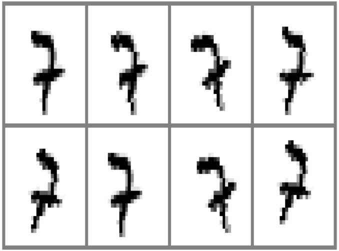

100
90
80
70
60
% Correct on Test Data
0
5
10
15
20
25
30
Training Epochs
Net-5
Net-4
Net-3
Net-2
Net-1

Table of Contents

Invariances

- When prediction should not depend on certain transformations - want to take this into account.
- Example: character recognition task
    - translation invariance
    - scale invariance
    - invariance to small rotations
    - invariance to small uniform noise
    - invariance to small smooth transformations



Invariances

- Approaches to build an invariant model:
  - **augment training set with invariantly transformed objects**
    - amount of possible transformations grows exponentially of #[invariances types].
  - **add regularization, penalizing changes in output after invariant transformations**
    - see *tangent propagation*
  - **extract features that are invariant to transformations**
    - e.g. color histogram, color gradient histogram
  - **build the invariance properties into the structure of neural network**
    - e.g. convolutional neural networks

Augmentation of training samples

1. generate a random set of invariant transformations
2. apply these transformations to training objects
3. obtain new training objects

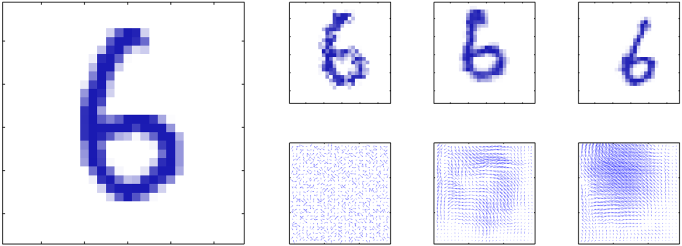


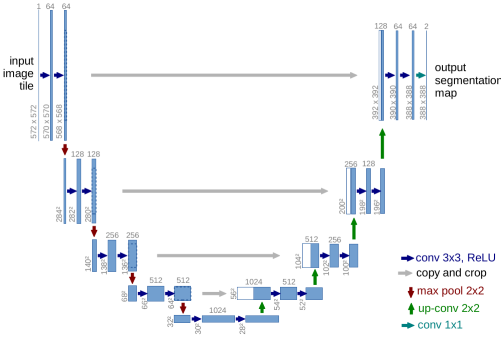
Table of Contents

Image segmentation[3]

- Segmentation - classification of every pixel of the image.
- Applications:
    - surveillance systems, autonomous driving, image classification, activity recognition on videos, etc.
- Model needs:
    - high level features to reconstruct object type
    - low level features to reconstruct boundaries

[3] Picture source.

U-net architecture[4]



Horizontal numbers = #[channels]; vertical numbers = spatial size.
White blocks - copied output of earlier layers; up-conv - rescaling & convolution.

[4] Ronneberger et al [2015].

Discussion

Key ideas of U-net:

- preserve spatial info at each layer
  - use only convolution, pooling, scaling.
  - don't use vectorization & fully connected layers
- 1st half - encoder; 2nd half - decoder.
- Encoder aggregates wider and wider local information
  - creating more abstract features
- Decoder reconstructs local information from
  - more abstract features (green input on figure)
  - lower level features (gray input on figure)