EE488 Special Topics in EE <Deep Learning and AlphaGo>

Sae-Young Chung

Lecture 9

October 25, 2017

Chap. 9 Convolutional Networks

- Convolution
- Examples (image classification & AlphaGo)
- Pooling

Convolution

- 1-dimensional continuous-time convolution

$$s(t) = \int_{-\infty}^{\infty} x(a)w(t-a)da = (x*w)(t) = x(t)*w(t)$$

- Convolution is commutative, i.e.,

$$x(t)*w(t) = \int_{-\infty}^{\infty} x(a)w(t-a)da = \int_{-\infty}^{\infty} w(a)x(t-a)da = w(t)*x(t)$$

- Convolution is linear, i.e., if $s_1(t) = (x_1 * w)(t)$ and $s_2(t) = (x_2 * w)(t)$, then

$$as_1(t) + bs_2(t) = ((ax_1 + bx_2) * w)(t)$$

- Convolution is translation invariant, i.e., if $x(t)$ is changed to $x(t-\tau)$, then

$$\int w(a)x(t-a-\tau)da = s(t-\tau)$$

- Linear time invariant (LTI) system can be specified as convolution, i.e., $y(t) = h(t) * x(t)$

Convolution

- 1-dimensional discrete-time convolution

$$s(i) = \sum_{m-\infty}^{\infty} x(m)w(i-m) = \sum_{m-\infty}^{\infty} w(m)x(i-m)$$

- 2-dimensional discrete-time convolution

$$S(i,j) = (I*K)(i,j) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} I(m,n)K(i-m,j-n) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} K(m,n)I(i-m,j-n)$$

- In many deep learning libraries, $K(m,n)$ is replaced by $K(-m,-n)$, i.e.,

$$S(i,j) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} K(-m,-n)I(i-m,j-n) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} K(m,n)I(i+m,j+n)$$

- This is actually cross-correlation, but it is still called convolution in many deep learning libraries.

Convolution

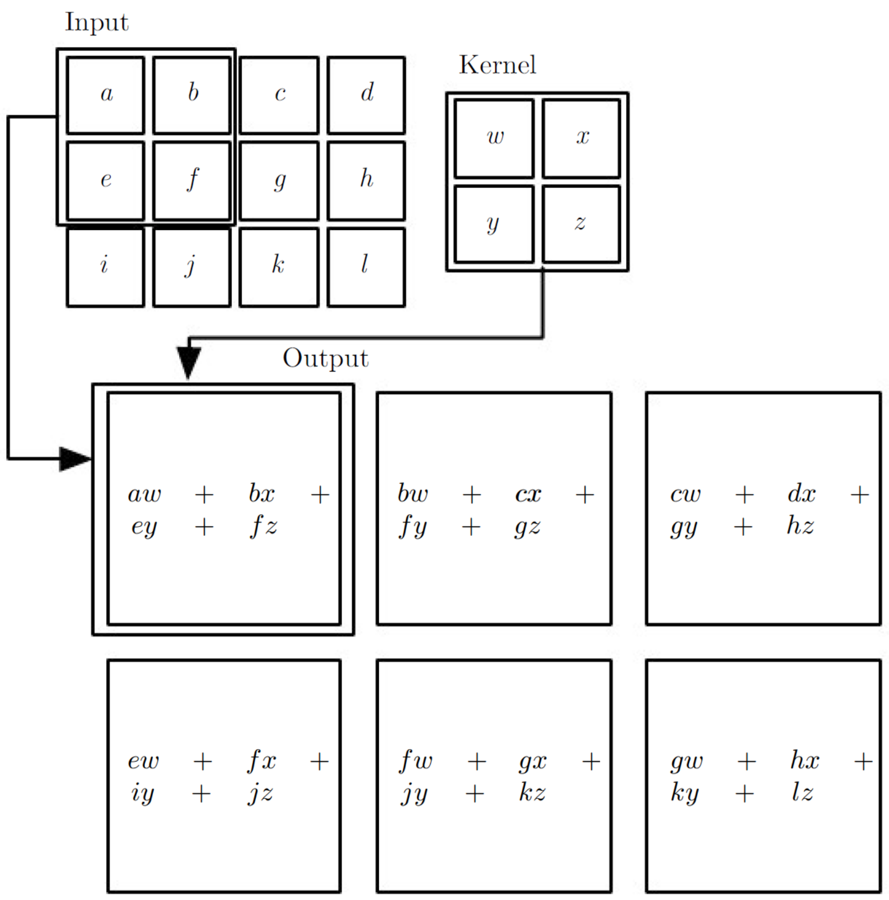
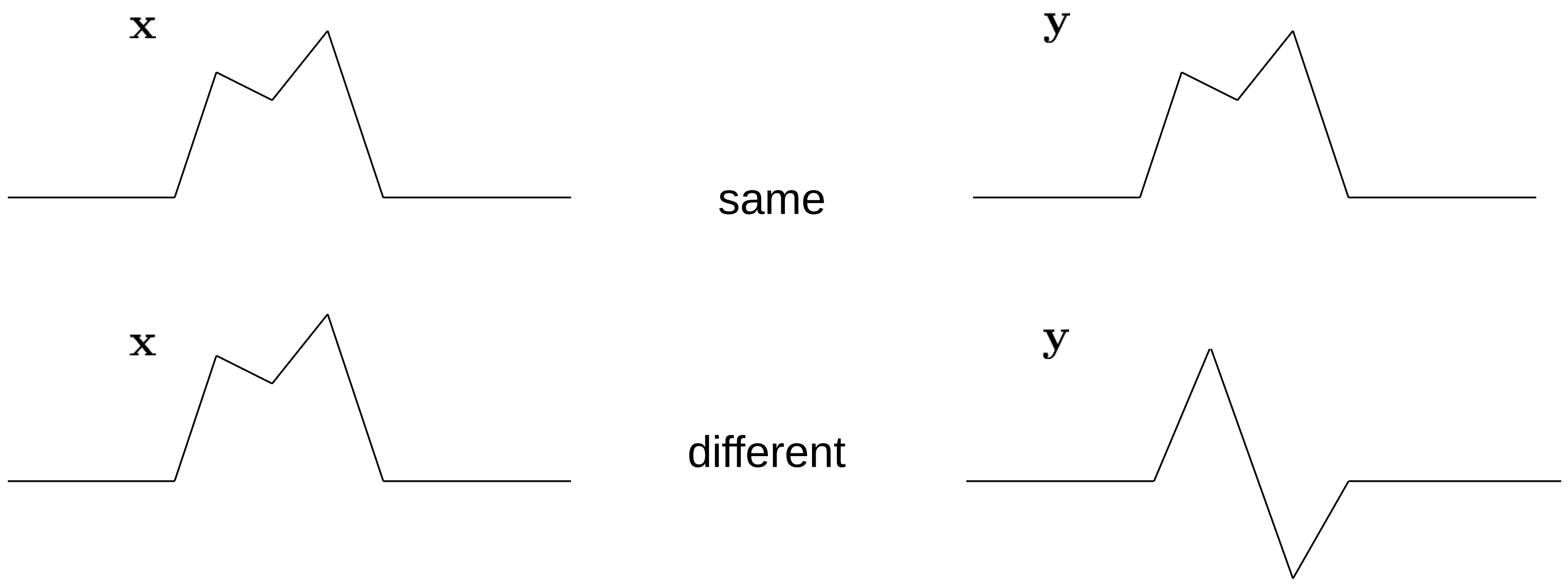


Fig. 9.1

Distance between Two Signals

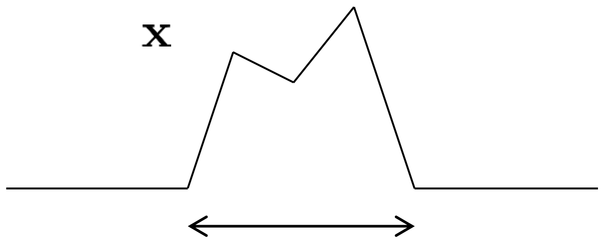
- How to measure how close two signals are to each other?
- Example) Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$$

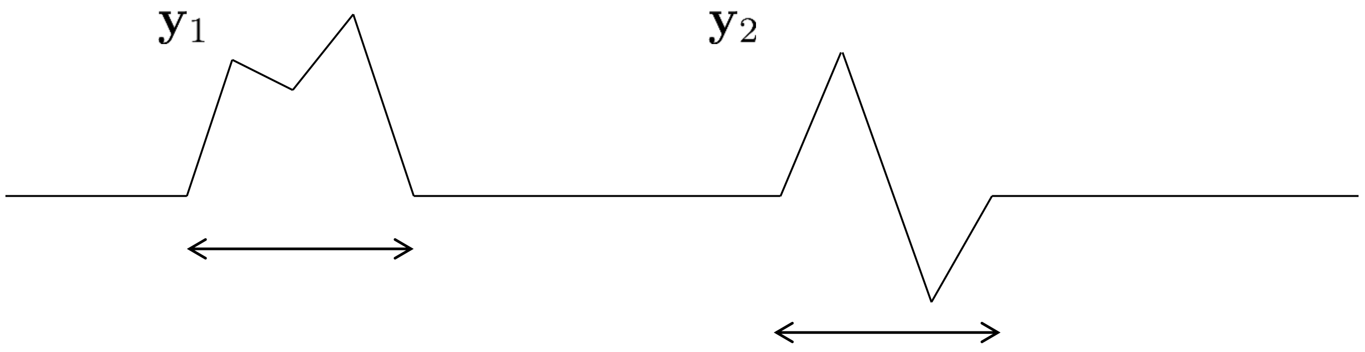


Finding a Pattern

How can you find $\mathbf{x}$ (template)



in the following signal?



Finding a Pattern

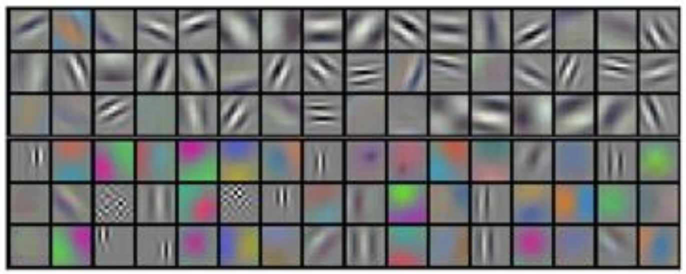
- $d(\mathbf{x},\mathbf{y})^2 = \|\mathbf{x}-\mathbf{y}\|^2 = (\mathbf{x}-\mathbf{y})\cdot(\mathbf{x}-\mathbf{y}) = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\mathbf{x}\cdot\mathbf{y}$
  - Inner product between $\mathbf{x}$ and $\mathbf{y}$: $\mathbf{x}\cdot\mathbf{y} = \mathbf{x}^T\mathbf{y}$
- Let's normalize signals so that the distance is invariant under scaling (e.g., audio volume or image brightness), then

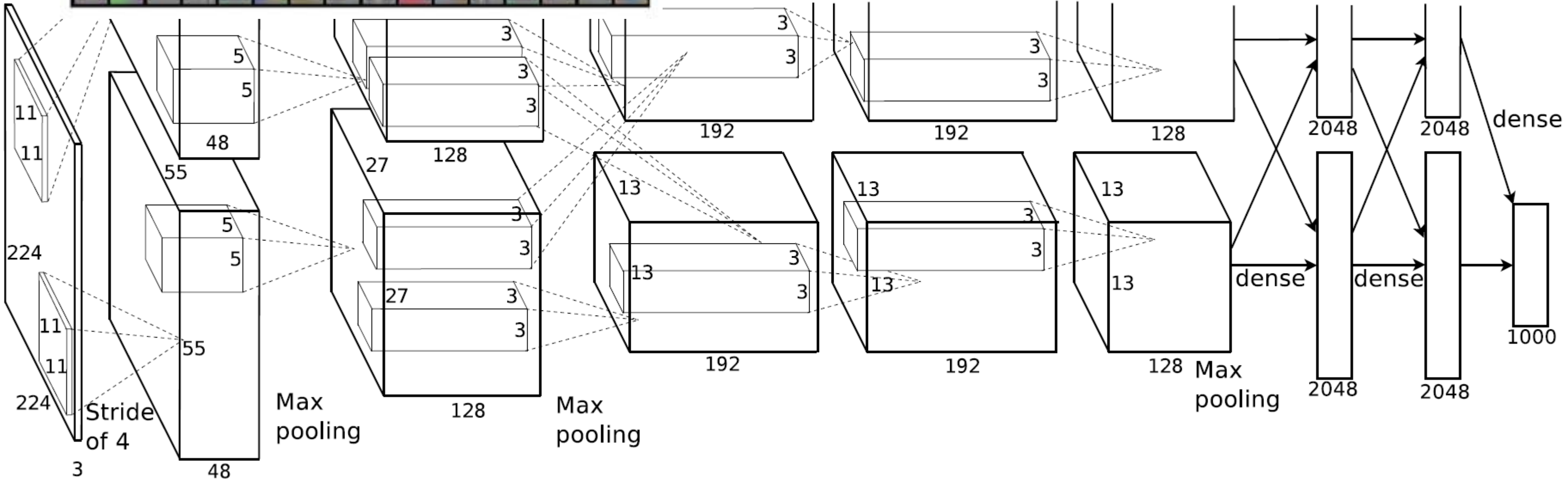
$$d\left(\frac{\mathbf{x}}{\|\mathbf{x}\|},\frac{\mathbf{y}}{\|\mathbf{y}\|}\right)^2 = \left\|\frac{\mathbf{x}}{\|\mathbf{x}\|} - \frac{\mathbf{y}}{\|\mathbf{y}\|}\right\|^2 = 1 + 1 - 2\frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\cdot\|\mathbf{y}\|}$$

- Therefore, instead of comparing $d\left(\frac{\mathbf{x}}{\|\mathbf{x}\|},\frac{\mathbf{y}_1}{\|\mathbf{y}_1\|}\right)$ and $d\left(\frac{\mathbf{x}}{\|\mathbf{x}\|},\frac{\mathbf{y}_2}{\|\mathbf{y}_2\|}\right)$, we can compare $\frac{\mathbf{x}\cdot\mathbf{y}_1}{\|\mathbf{y}_1\|}$ and $\frac{\mathbf{x}\cdot\mathbf{y}_2}{\|\mathbf{y}_2\|}$.
- Or, even simply we can compare $\mathbf{x}\cdot\mathbf{y}_1$ and $\mathbf{x}\cdot\mathbf{y}_2$ if $\|\mathbf{y}_1\| = \|\mathbf{y}_2\|$.
- Using convolution, we can compute multiple correlation (or inner product) values between a template (kernel) and the input by sliding the kernel over the input.

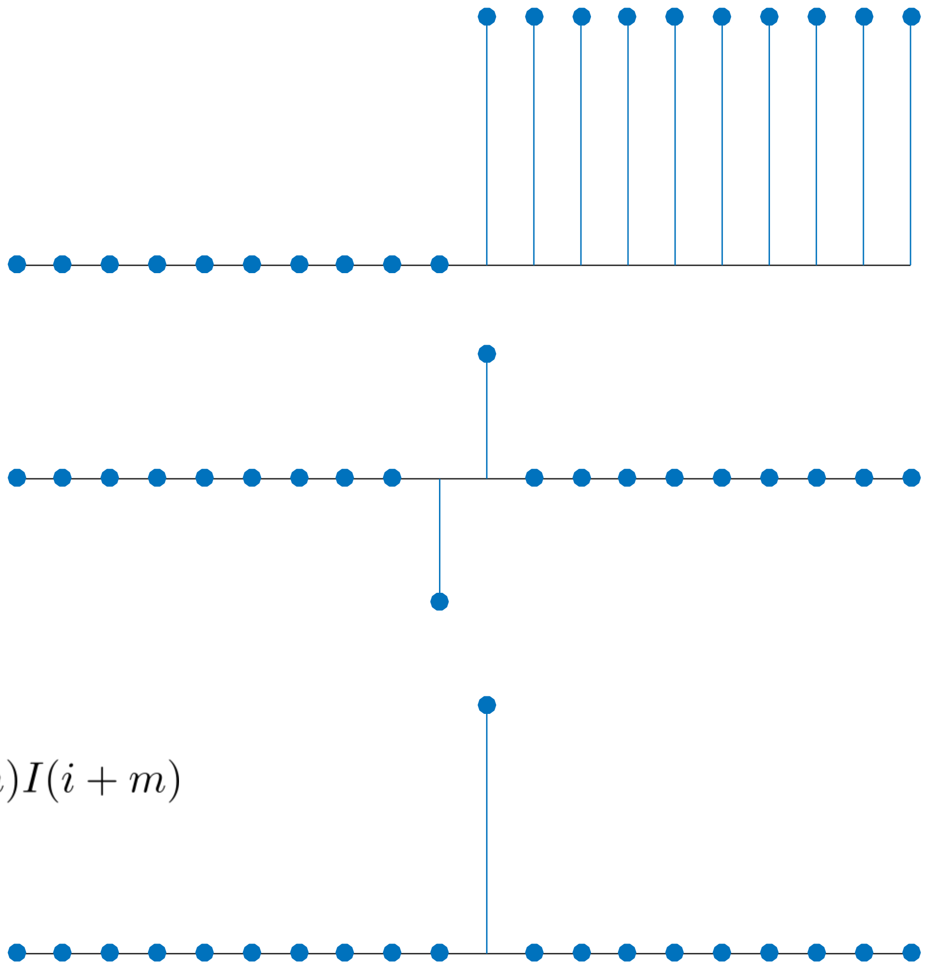
CNN for Image Classification

- Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton, “ImageNet classification with deep convolutional neural networks”, NIPS 2012
- 60 million parameters and 650,000 neurons



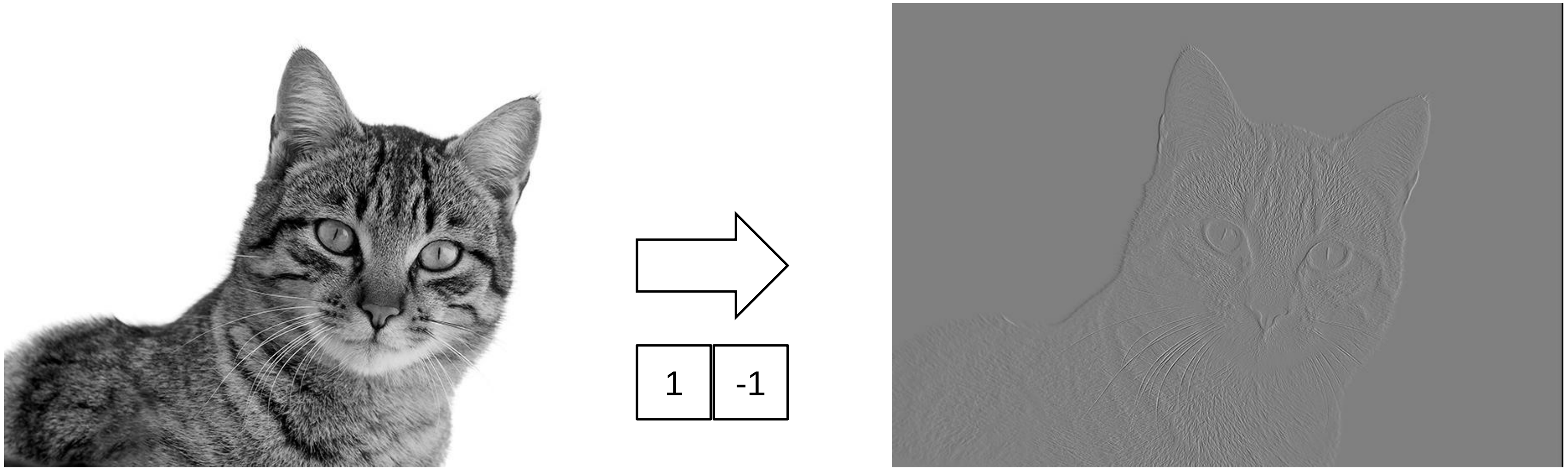


Edge Detection

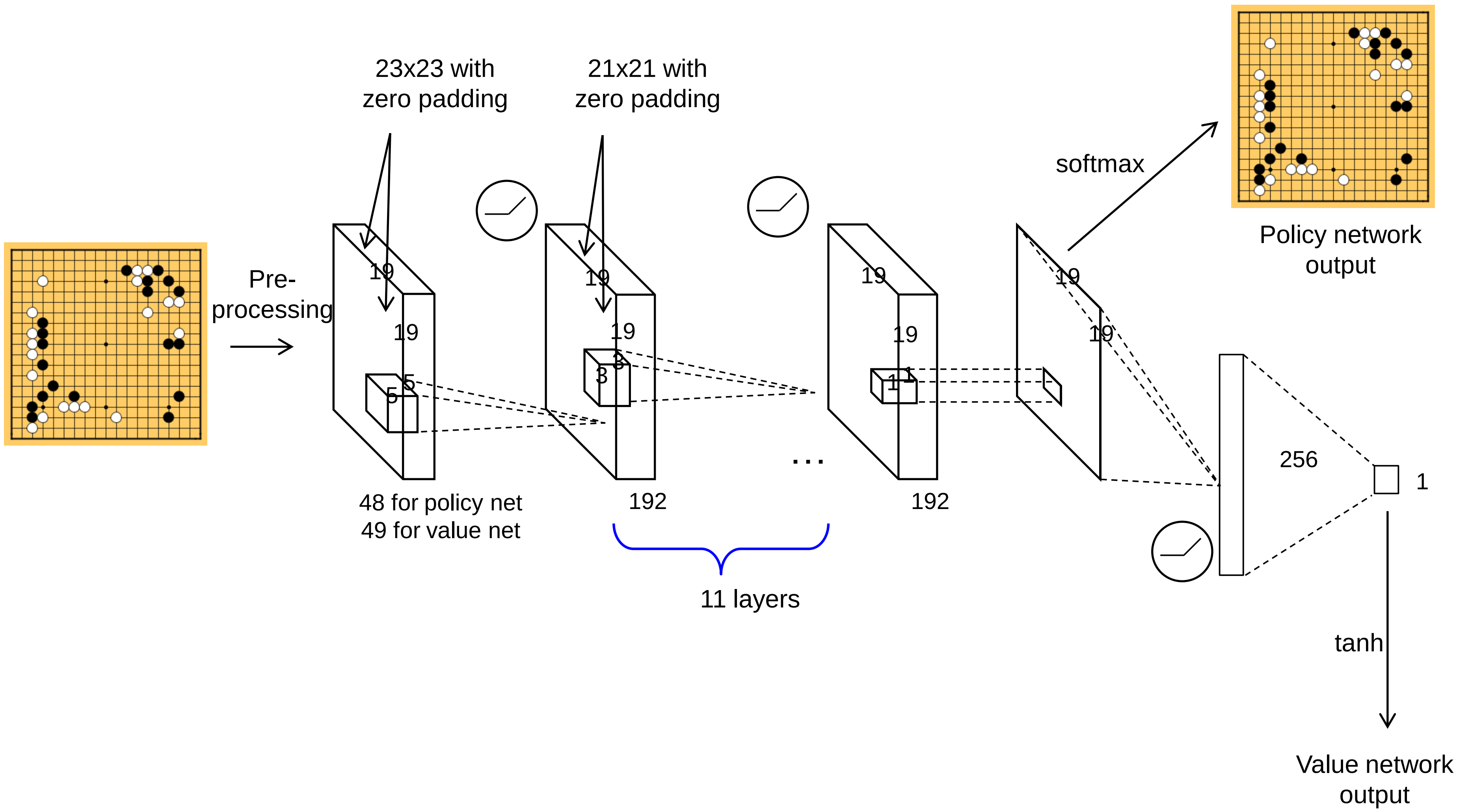


$(I * K)(i) = \sum_m K(m)I(i+m)$

Edge Detection



CNN in AlphaGo



Convolution

- First convolutional layer in AlexNet
  - $S(i,j,k) = \sum_{m,n,c} I(i+m, j+n, c) K(m,n,c,k)$, $1 \leq i+m, j+n \leq 227$, $1 \leq c \leq 3$, $1 \leq k \leq 96$, $1 \leq i,j \leq 217$, $0 \leq m,n \leq 10$ (ignoring stride)
  - From $I(*,*,)$ & $K(*,*,,)$ to $S(*,*,)$: 2-dim convolution
  - From $I(,,*)$ & $K(,,*,*)$ to $S(,,*)$: matrix multiplication
  - Number of parameters (not including bias): $(11 \times 11 \times 3) \times 96 = 34,848$
  - If fully connected: $(227 \times 227 \times 3) \times (55 \times 55 \times 96) \sim 4.5 \times 10^{10}$
- Due to parameter sharing (same filter coefficients are used across the whole input image) and sparse connection (only 11×11 instead of covering the whole image), the number of parameters is drastically reduced. This is not only computationally more efficient, but can also prevent overfitting.
- Although it has far fewer parameters than a fully connected layer, it can still perform many useful tasks, e.g., edge detection.
- Since translation invariance is an important aspect in image recognition, translational invariance in convolution is helpful. For example, edge detection needs to work the same way no matter where the edge is in the image.
- Due to locality in images, small filters can detect many useful features, e.g., an edge in a small region of an image.
- In later stages of CNN, information from many different regions of the input image can be combined and processed together. Therefore, using a small-sized filter in each layer does not necessarily limit the spatial span for processing.

Pooling

- It is OK to lower spatial resolution as we go through layers since the input image has high resolution whereas feature detection at later layers does not require such high resolution.
- E.g., in AlexNet, $227 \times 227 \to 55 \times 55 \to 27 \times 27 \to 13 \times 13$
- Max pooling can be used to lower resolution without degrading performance much, e.g., we can take the maximum in 4×4 samples as we go from 227×227 to 55×55. Then, we move by 4 samples and take another maximum and so on (stride is 4). (In AlexNet, max pooling is done differently. See next page.)

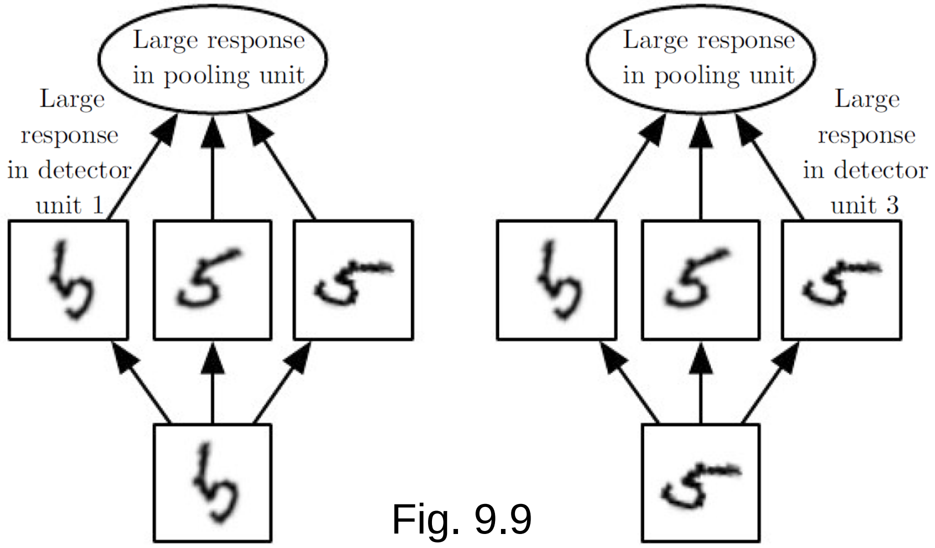

Large response in pooling unit
Large response in detector unit 1
Large response in pooling unit
Large response in detector unit 3

Fig. 9.9

Pooling

- TensorFlow example

```
x_image = tf.reshape(x, [-1,227,227,3])
W_conv = tf.Variable(tf.truncated_normal([11, 11, 3, 48], stddev=0.1))
b_conv = tf.Variable(tf.constant(0.1, shape=[48]))
h_conv = tf.nn.conv2d(x_image, W_conv, strides=[1, 4, 4, 1], padding='VALID')
h_relu = tf.nn.relu(h_conv + b_conv)
h_norm = tf.nn.local_response_normalization(h_relu, depth_radius=2, alpha=2e-5, beta=0.75, bias=1.0)
h_pool = tf.nn.max_pool(h_norm, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1], padding='VALID')
```

- `padding='VALID'` in `conv2d` means each output of the convolution is from a valid region of the input. Since the input is 227×227 and the filter size is 11×11, the output of the convolution has size $(227 - 11 + 1) \times (227 - 11 + 1) = 217 \times 217$.
  - If `padding='SAME'` is used in `conv2d`, then the output of the convolution is 227×227, i.e., the same as the input using zero padding.
- `strides=[1, 4, 4, 1]` in `conv2d` means taking one out of 4 samples in each dimension (decimation). Thus, with stride, the size of the output of `conv2d` becomes $\lceil \frac{217}{4} \rceil \times \lceil \frac{217}{4} \rceil = 55 \times 55$.
- `ksize=[1,3,3,1]` in `max_pool` means taking the maximum in 3×3 samples
- `strides=[1,2,2,1]` in `max_pool` means shifting by 2 samples after taking a maximum
- Therefore, the output of max pooling is $\lfloor \frac{55}{2} \rfloor \times \lfloor \frac{55}{2} \rfloor = 27 \times 27$ since `padding='VALID'` is used.
  - The output of max pooling will be $\lceil \frac{55}{2} \rceil \times \lceil \frac{55}{2} \rceil = 28 \times 28$ if `padding='SAME'` is used instead.

Examples of Signals

- Audio signal
  - $x(t)$: mono audio, scalar, 1 dimensional (i.e., time domain)
  - $(L(t), R(t))$: stereo audio, vector signal of size 2, still 1-dimensional (i.e., time domain)
- Image signal
  - $I(x, y)$: monochrome (intensity only) image, 2-dimensional (i.e., 2 spatial axes - x and y axes)
  - $(R(x, y), G(x, y), B(x, y))$: full color image (red, green, blue), vector signal of size 3, still 2-dimensional (i.e., 2 spatial axes)
- Video signal
  - $(R(x, y, t), G(x, y, t), B(x, y, t))$: full color video (red, green, blue), vector signal of size 3, 3-dimensional (i.e., 2 spatial axes + 1 temporal axis)

Dimension & Channel

- 1-dim convolution: audio
  - Single channel: mono audio
  - Multiple channels: stereo audio, multi-track audio
- 2-dim convolution: image or audio spectogram (time-frequency representation)
  - Single channel: monochrome image
  - Multiple channels: color image (R,G,B or C,M,Y,K)
- 3-dim convolution: 3-dim image such as CT or MRI image or video (2 spatial dimensions + 1 temporal dimension)
  - Single channel: monochrome video
  - Multiple channels: color video (R,G,B or C,M,Y,K)

Assignments, etc.

- Reading assignment: Chapters 1 & 2 of RL book
  - “Reinforcement learning: An introduction” by R. Sutton and A. Barto
    - Draft available online at http://incompleteideas.net/sutton/book/the-book-2nd.html
    - Most recent version is dated June 19, 2017
- Problem set #3
  - Training deep learning using TensorFlow
  - Due: November 8
- No class on November 29 due to KAIST entrance exam