

Convolutional (畳み 込み) Networks

Chapter 9

楊 森 (Sen Yang) 2019/3/23

The Convolution Operation①畳み込みは何?

レーザーセンサー: $x(t)$ t時刻の宇宙船の位置をアウトプットしている

①ノイズがあるので、移動平均を求めよう。②最新の測定が関連性が高いので、加重平均を求めよう。重み関数:

加重平均の宇宙船位置
(畳み込み) :

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

入力
(input)

カーネル
(kernel)

$\omega(a)$

aは測定の年齢

実際は離散畳み込み: $s(t) = (x * \omega)(t) = \sum_{a=-\infty}^{\infty} x(a)\omega(t - a)$

The Convolution Operation②画

像に関する畳み込み

2次元畳み込み: $S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n)$

可換性: $S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n)$

相互相關関数 (畳み込みじゃないが、よく畳み込みと呼ばれる):
 $S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n)$

離散畳み込みは、マトリクス掛け算と等価

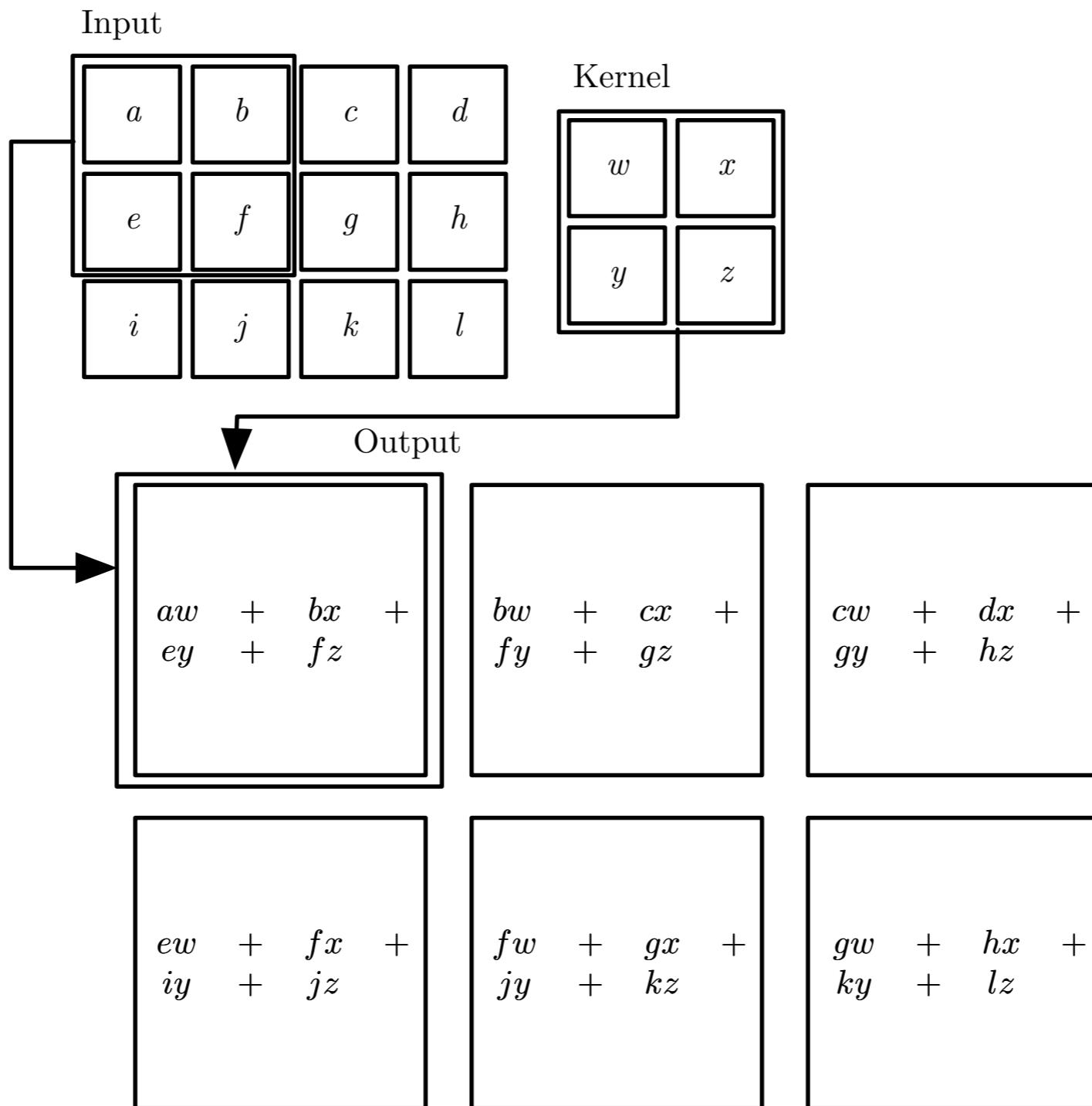
doubly block circulant matrix (巡回行列):
二次元畳み込みはdoubly block巡回行列との掛け算と等価

$$\begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & \cdots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & \ddots & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \dots & \dots & a_2 & a_1 & a_0 \end{bmatrix}$$

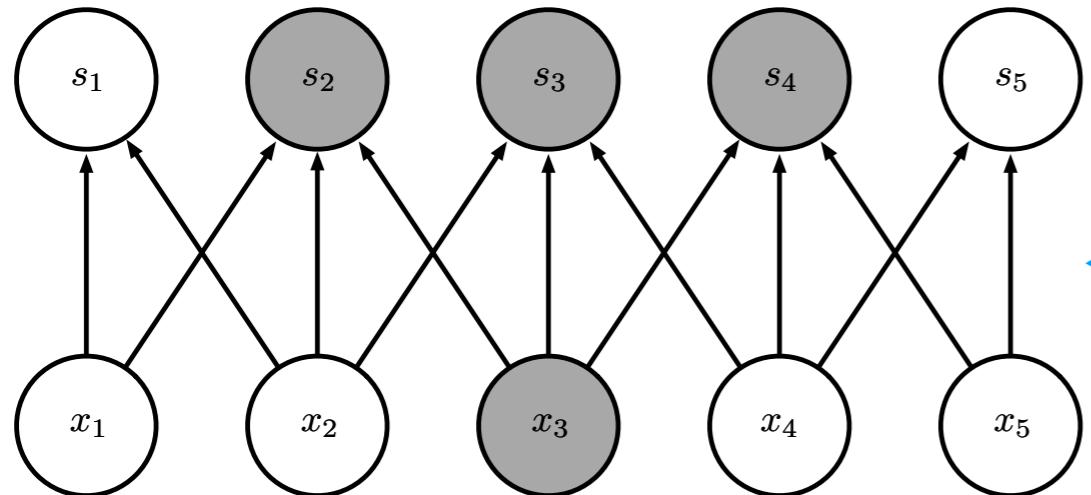
Toeplitz matrix:
单変量畳み込みは
Toeplitz matrixと
の掛け算と等価

$$\begin{bmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{bmatrix}$$

The Convolution Operation③マトリクス掛け算の二次元畳み込み実現

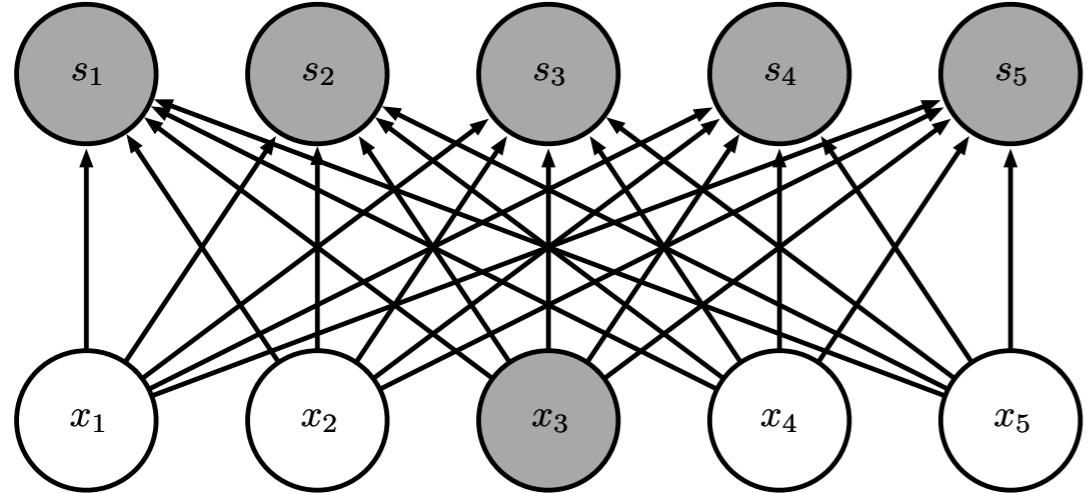


モチベーション①Sparse Interaction (Sparse Connectivity, Sparse Weights)



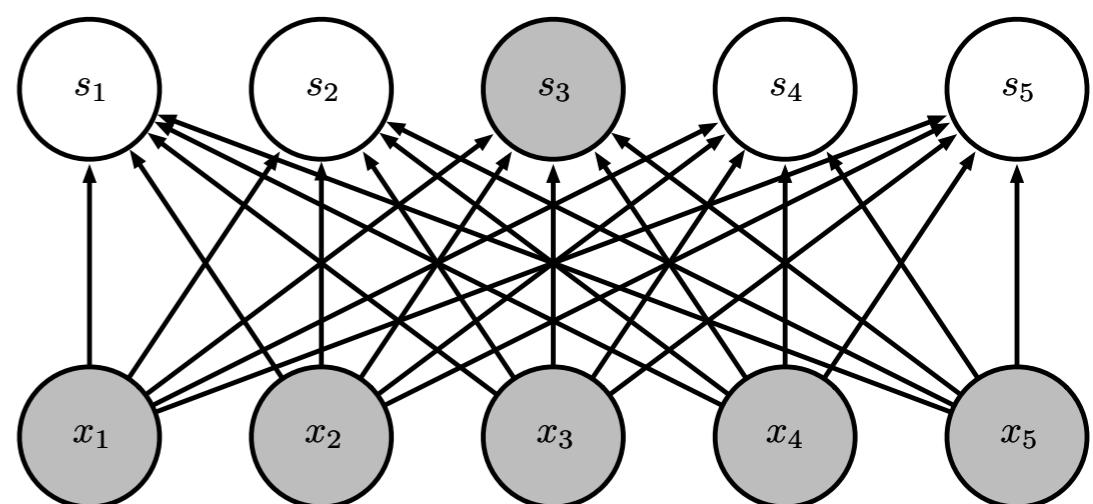
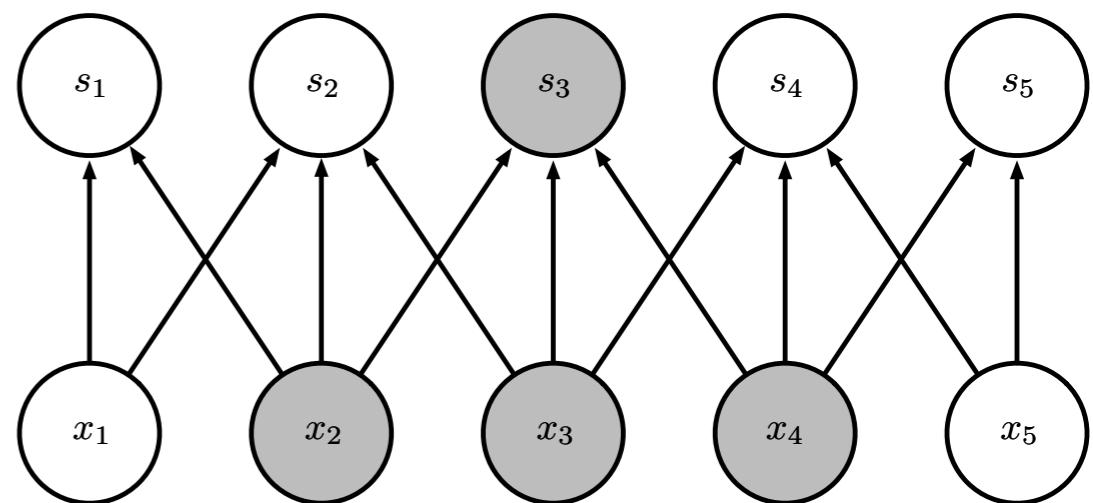
パラメータ数: $k \times n$ (3*5)
計算複雑さ: $O(k \times n)$
 k は、kernel行列の元素数

右側: 上
から下を見る

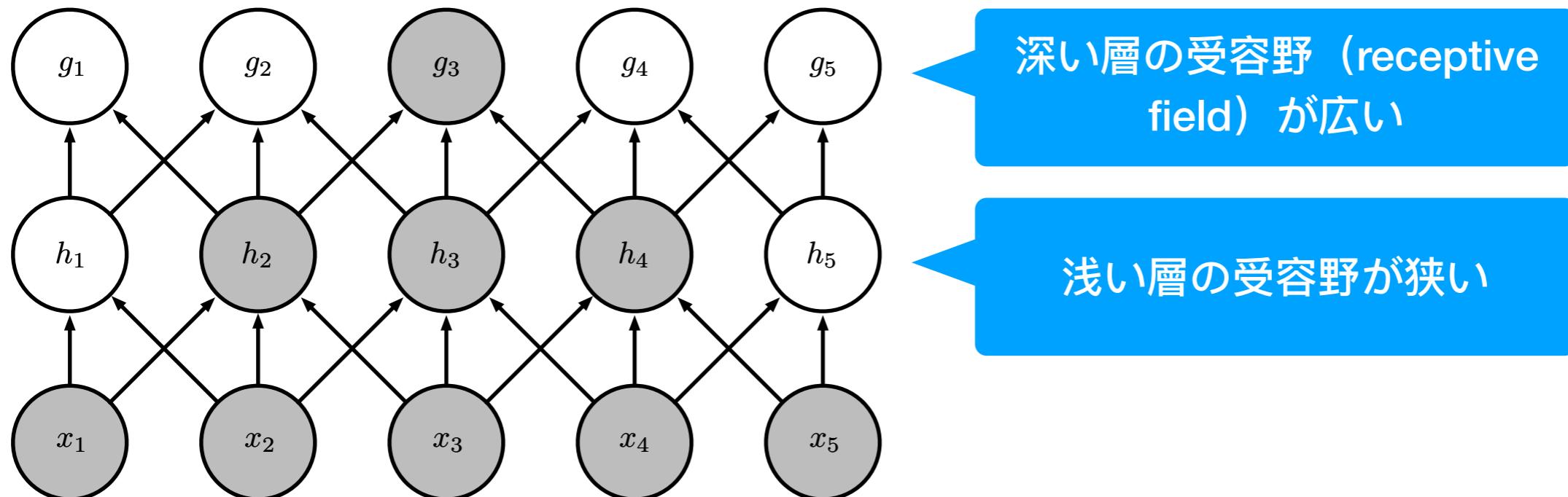


左側: 下
から上を見る

パラメータ数: $m \times n$ (5*5)
計算複雑さ: $O(m \times n)$

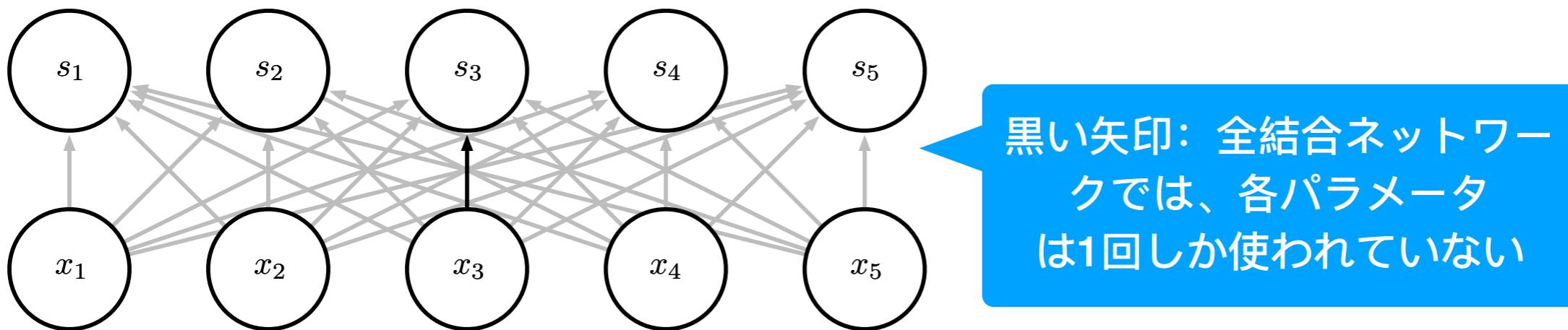
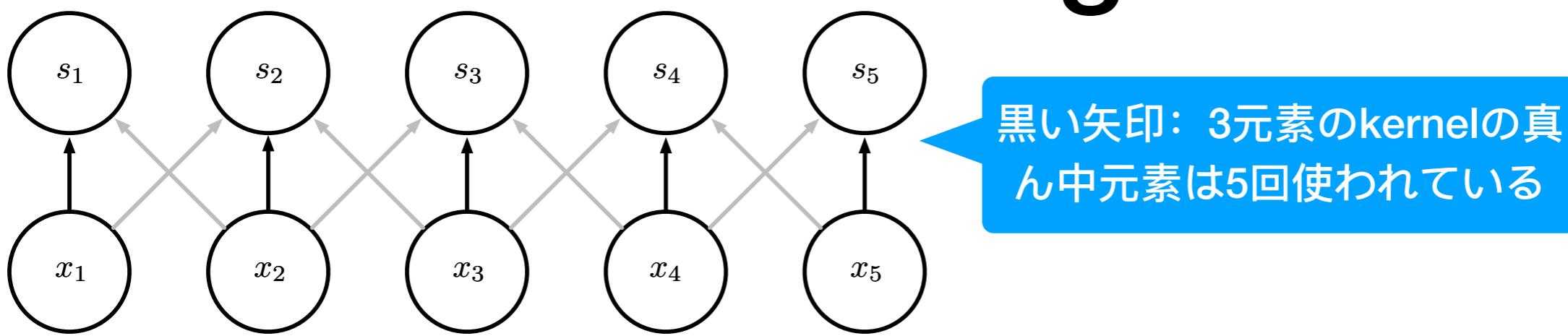


モチベーション①Sparse Connectivityでの間接Connection



畳み込みネットワークに直接ConnectionがSparseだが、深い層は全ての入力画像データと間接的に繋がっている。

モチベーション②Parameter Sharing



モチベーション③Equivariant (同変) Representation。

同変: inputが変わると、outputも同じく変わる。

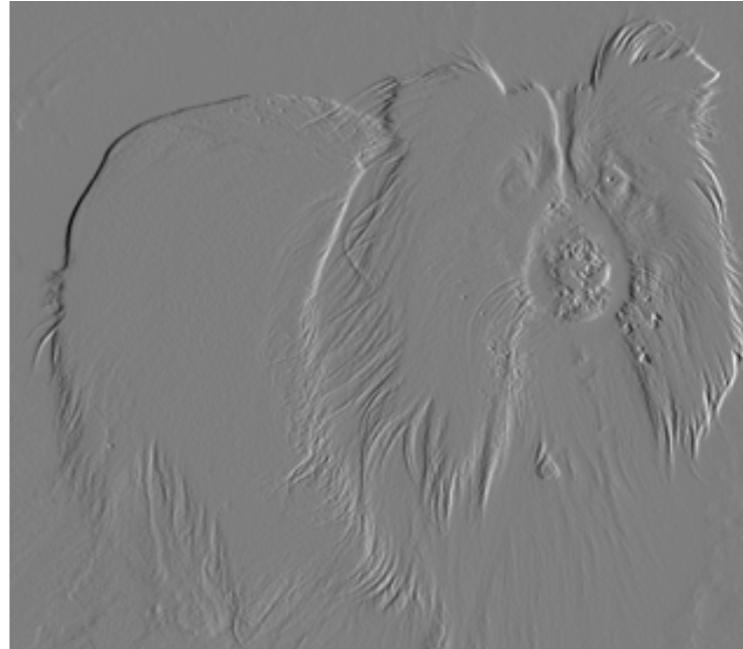
畳み込みは、移動に同変 (scaleや回転には同変じゃない)。

つまり、先に移動して、また畳み込むことと、先に畳み込んで、また移動すること
は同様です。

モチベーション②Sparse Connectivity とParameter Sharingの効率上の効果

Edge Detection (エッジ検出) :
(式: ピクセル-右隣りピクセル)

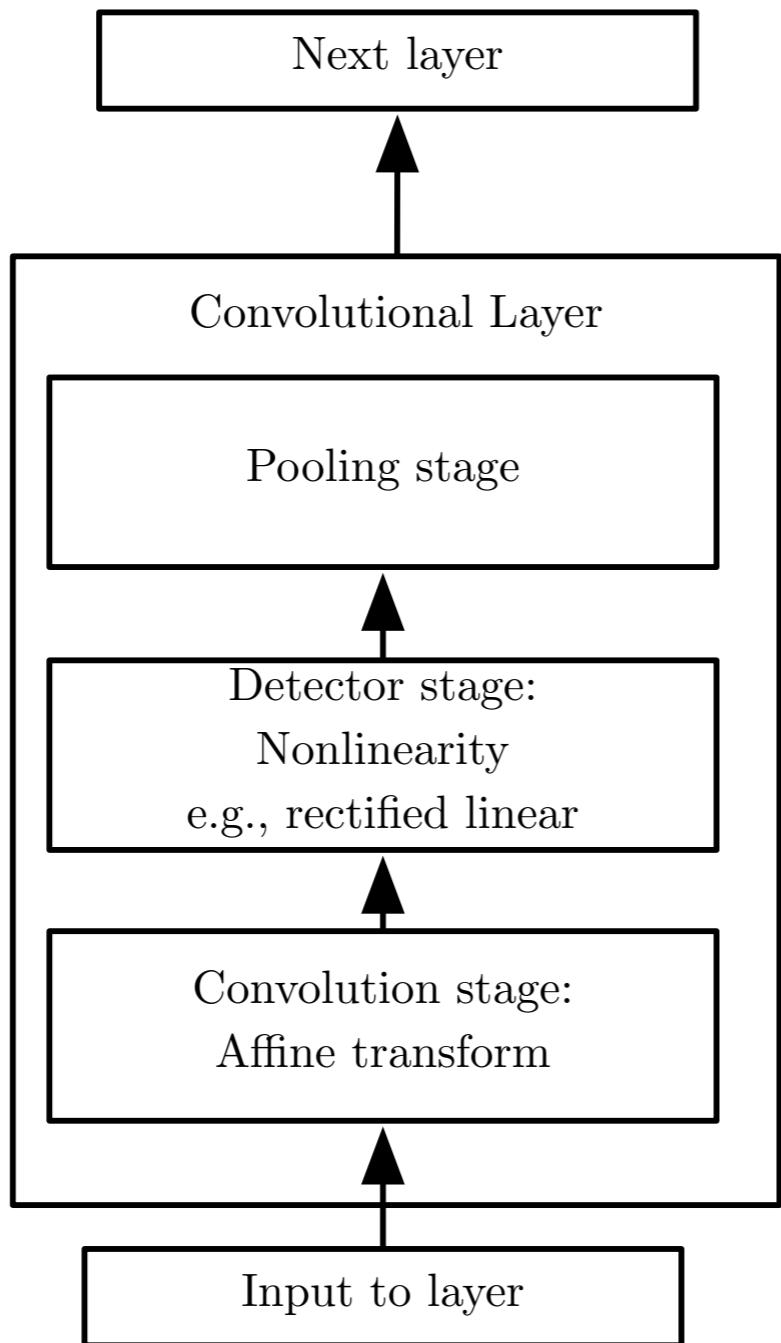
画像ピクセル: 320*280
(width*height)



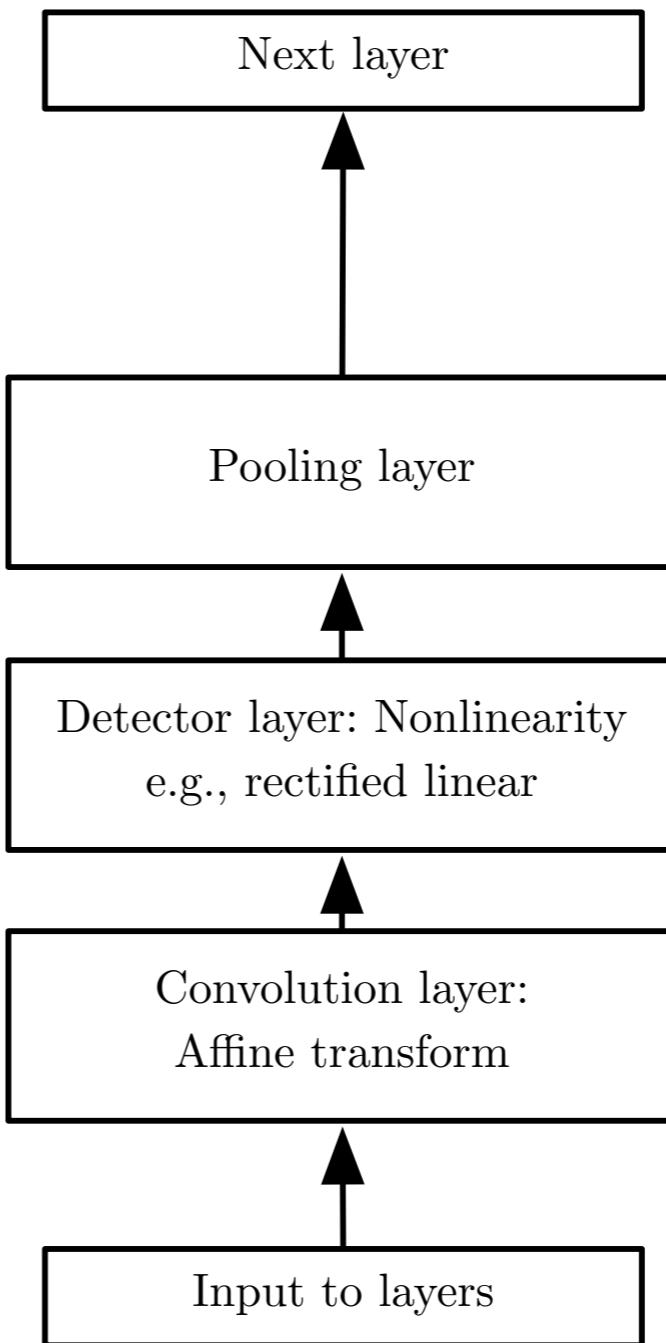
畳み込みだと: $320*280*3=267960$ (2つ掛け算 + 1つ足し算)
全結合だと: $320*280*319*280$ (80億以上)

畳み込み層の構造や用語（本書は左の用語を使う）

Complex layer terminology

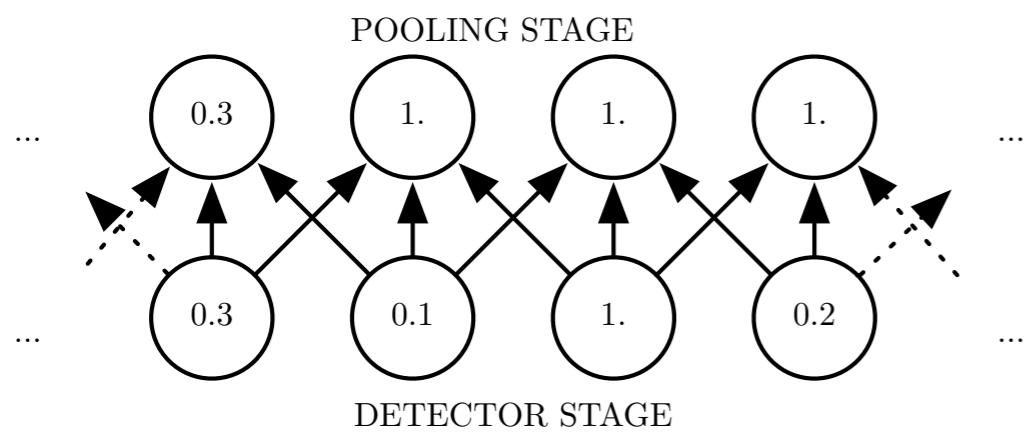
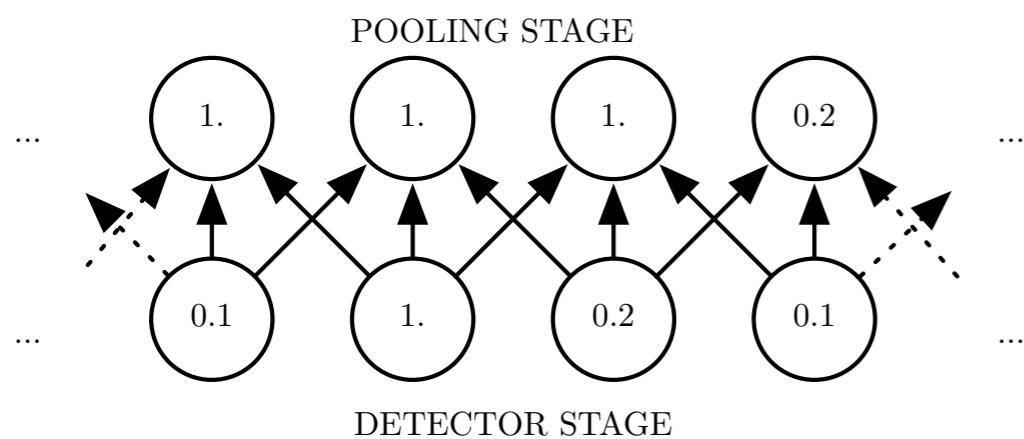


Simple layer terminology

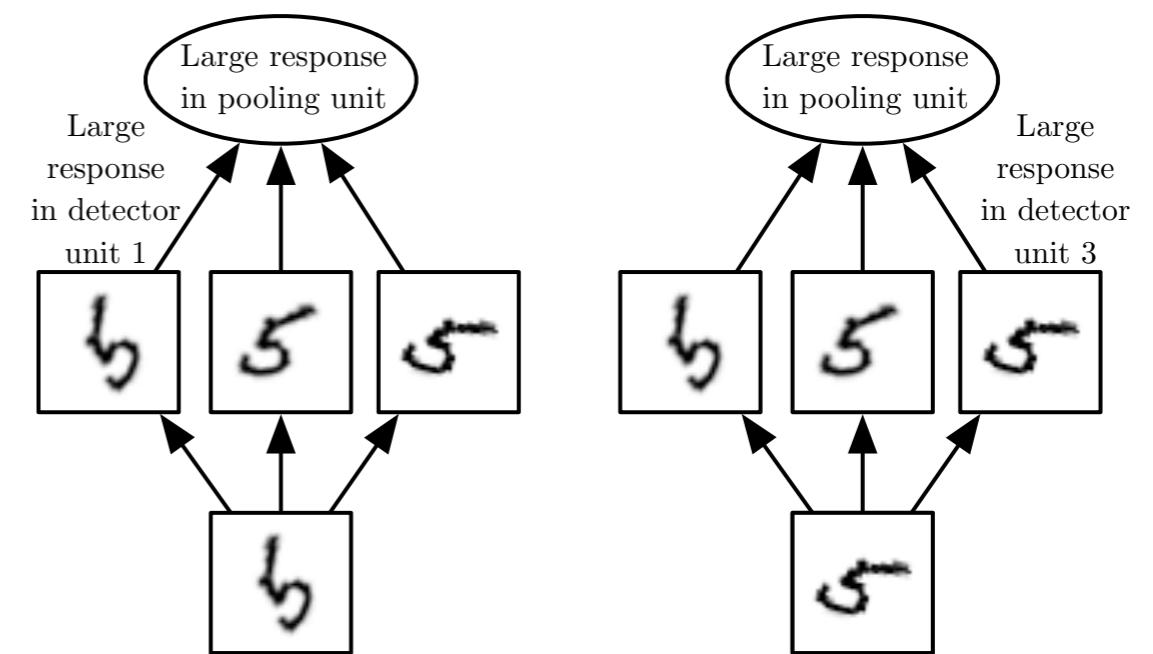


Pooling①できること

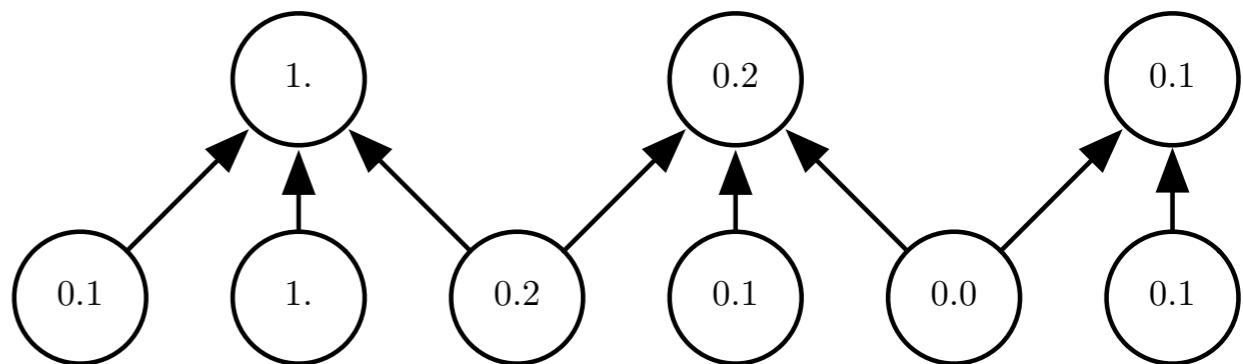
不变性 (Invariance) (ローカル移動に不变性、生まれつき)



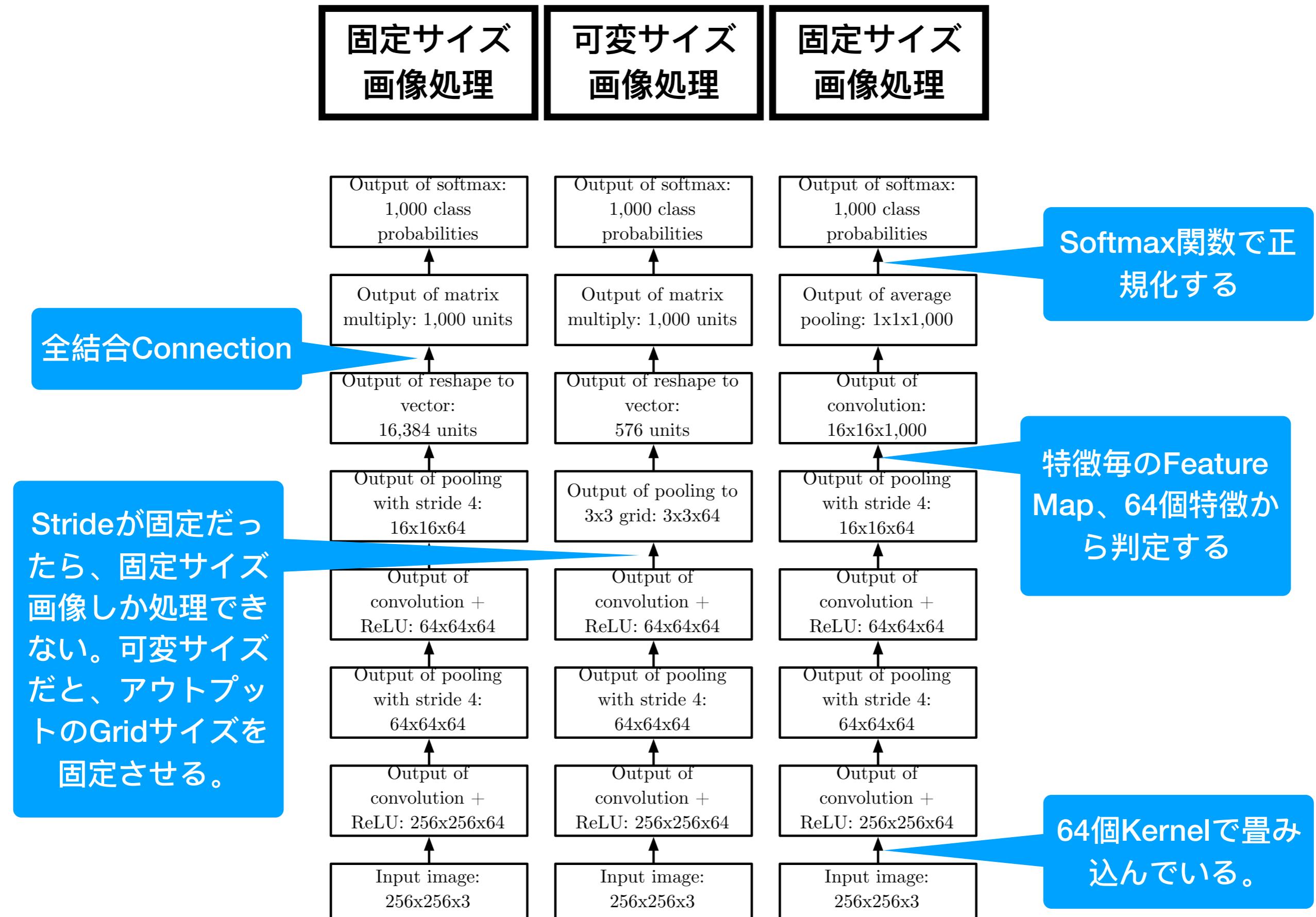
不变性習得できる (回転など)



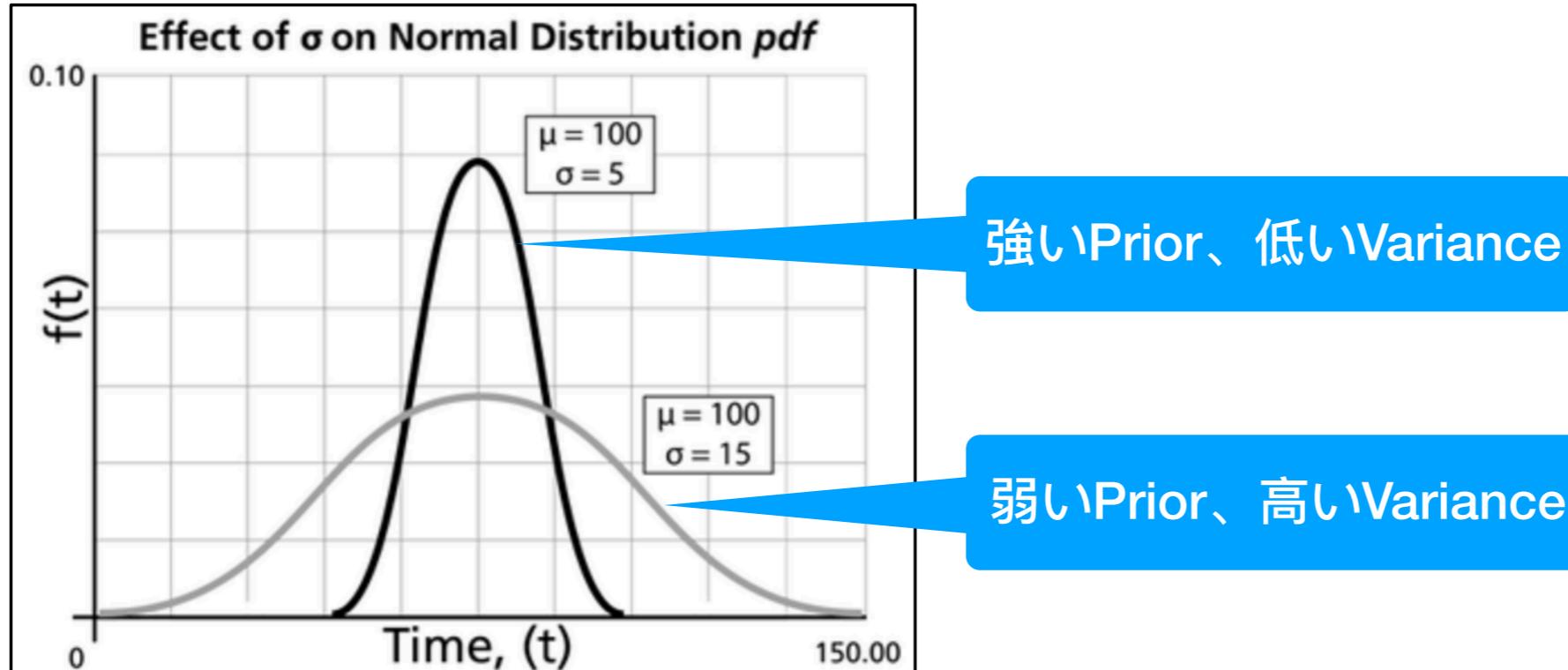
Downsampling



Pooling②分類用畳み込みネットワークの構造例



Convolution and Pooling as an Infinitely Strong Prior



ConvolutionのPrior: 学習した関数は移動に同変性を持つべき。また、Local Interactionしか含まないべき（グローバルの画像情報は含まない）。

デメリット: Priorと空間プロパティが一致しないとUnderfittingが発生する。例えば、もし離れている位置の情報を組み込めば、ConvolutionのPriorはふさわしくなくなる。

PoolingのPrior: Unitは小さい移動に不变性を持つべき。

基本畳み込み関数のバリアント①4D

Kernel畳み込み（チャネルあり）

カラー画像の場合、3チャネルがあるので、4D Kernelが必要

$$K_{i,j,k,l}$$

i: output channel
j: input channel
k, l: k rows and l columns offset

$$V_{i,j,k}$$

$$Z_{i,j,k}$$

V: input
Z: output
i: channel
j: row
k: column

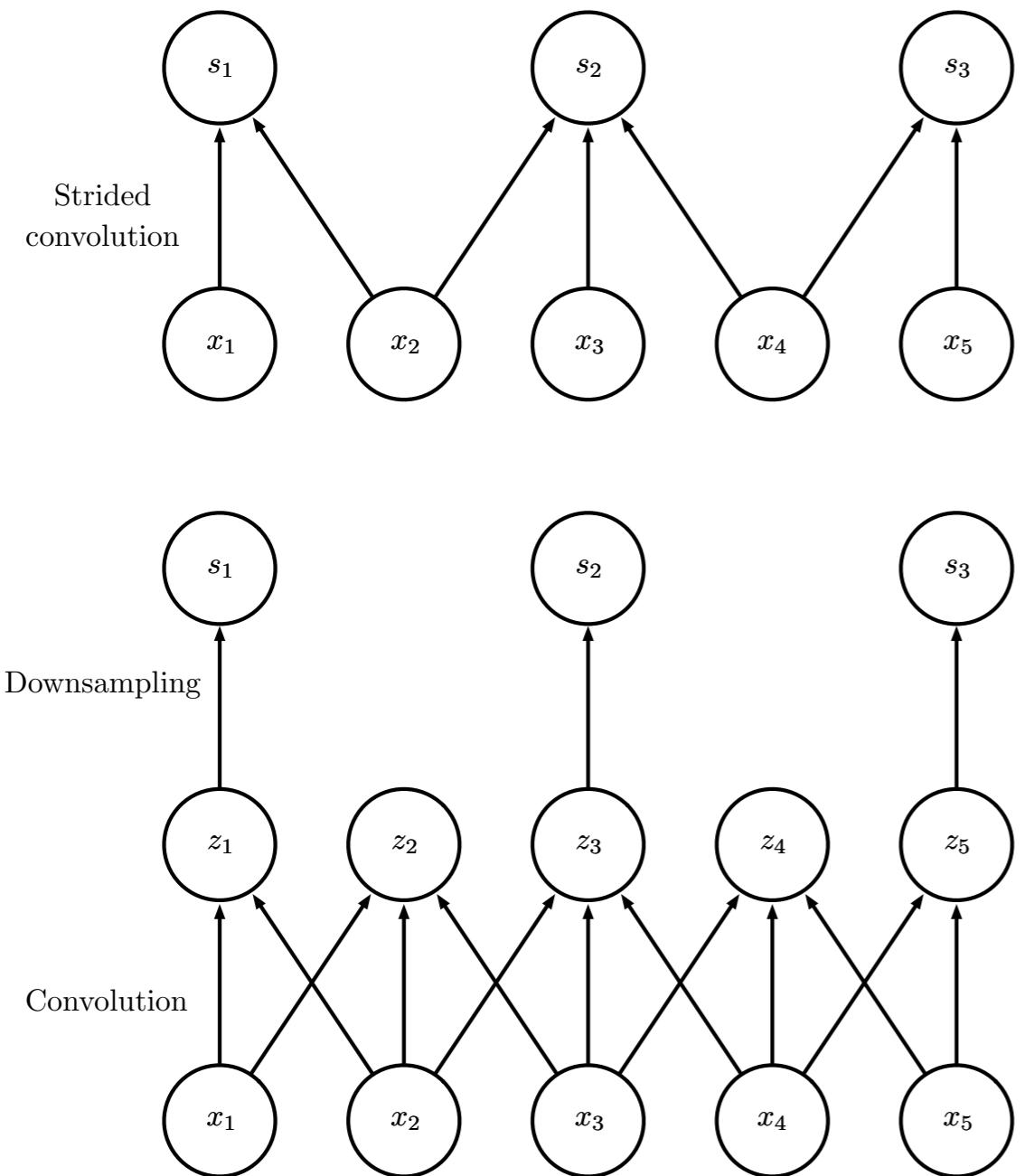
$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n}$$

Strideあり（計算コスト低減のため）畳み込み：

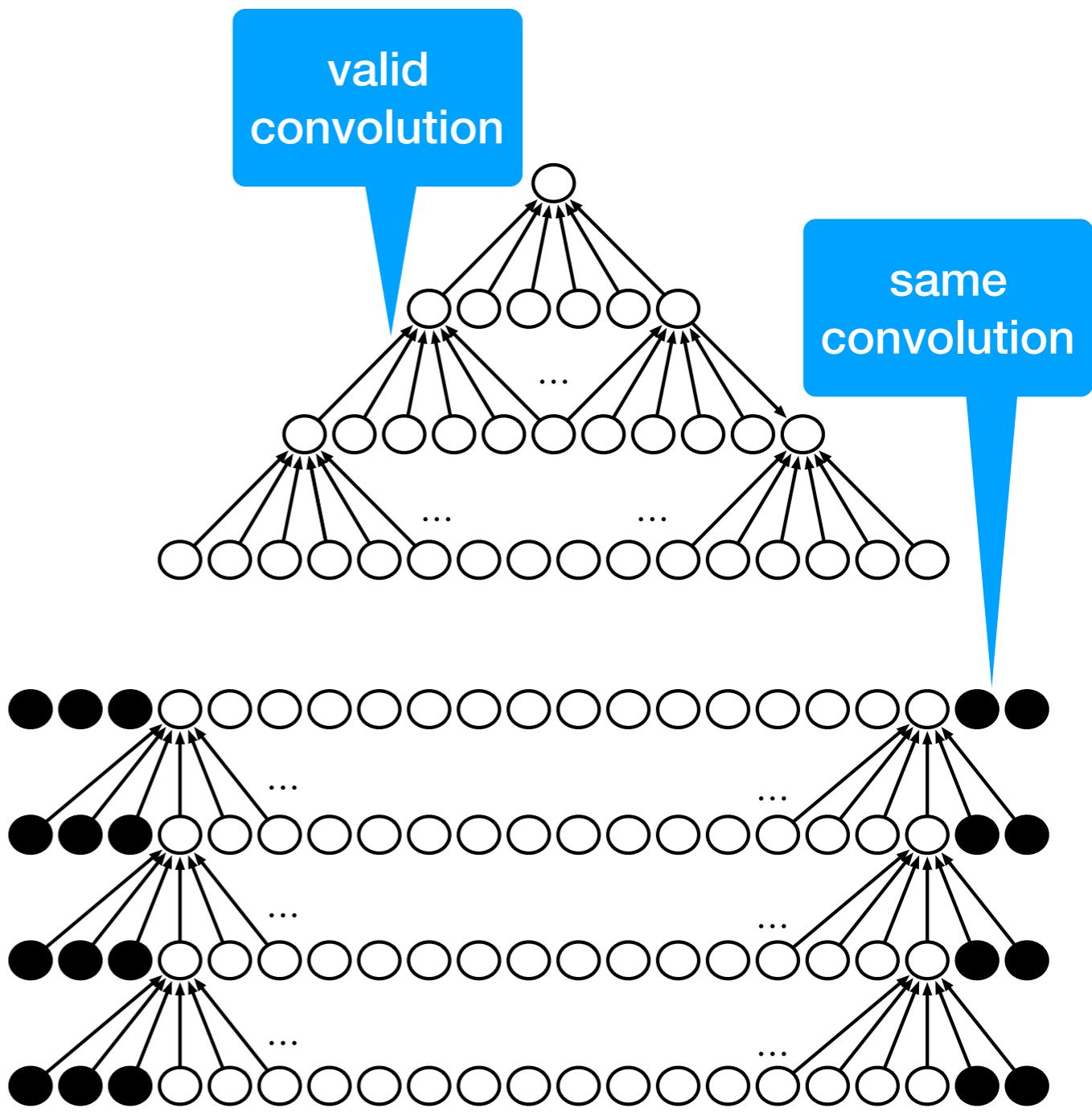
$$Z_{i,j,k} = c(K, V, s)_{i,j,k} = \sum_{l,m,n} \left[V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n} \right]$$

s: Stride

左側: Strideありの畳み込みは、畳み込み + Downsampling と等価だが、畳み込み + Downsamplingの方がたくさん の無駄な計算がある

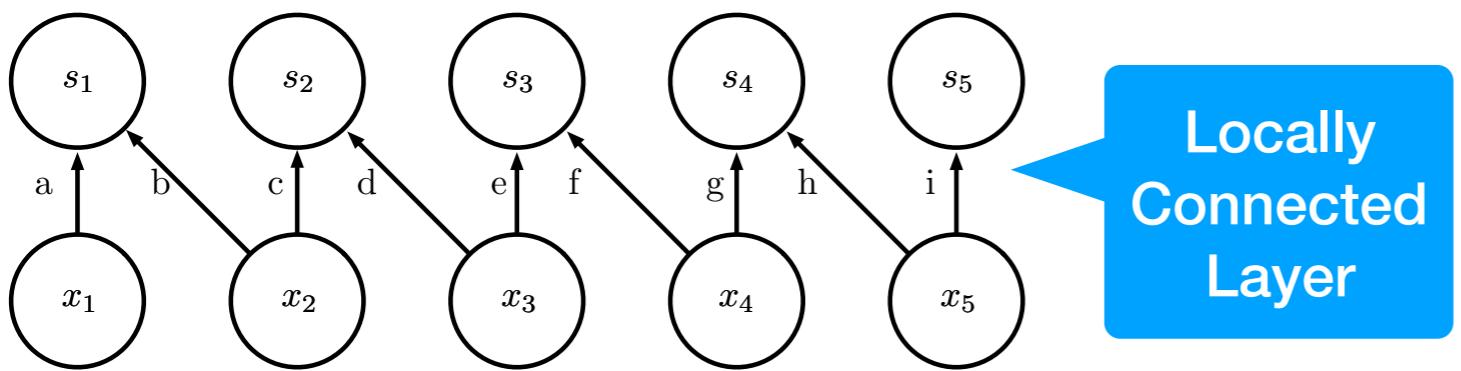


右側: zero padding。zero paddingをすると、任意層数のネットワークを構築できる



基本畳み込み関数のバリエント②Locally Connected Layer (Unshared Convolution)

$$Z_{i,j,k} = \sum_{l,m,n} [V_{l,j+m-1,k+n-1} \omega_{i,j,k,l,m,n}]$$



$\omega_{i,j,k,l,m,n}$

i: output channel

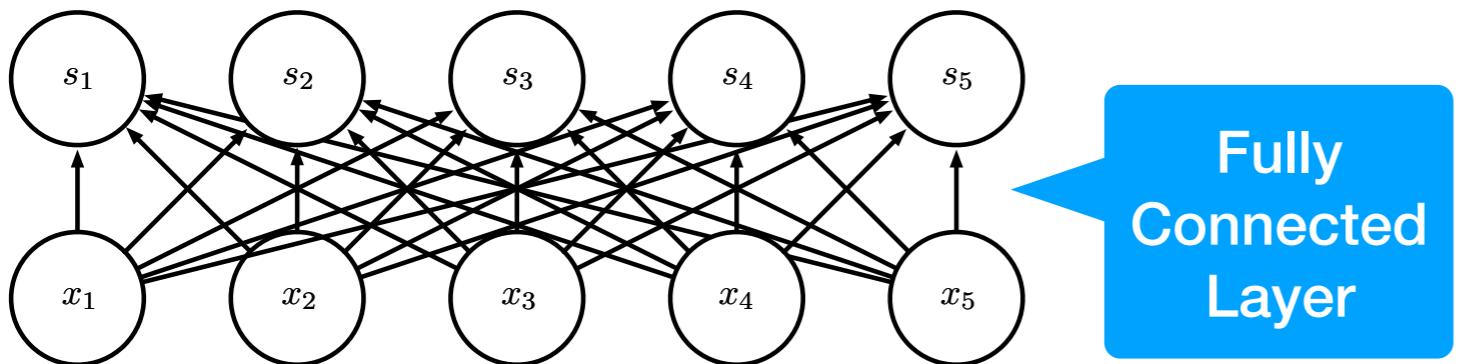
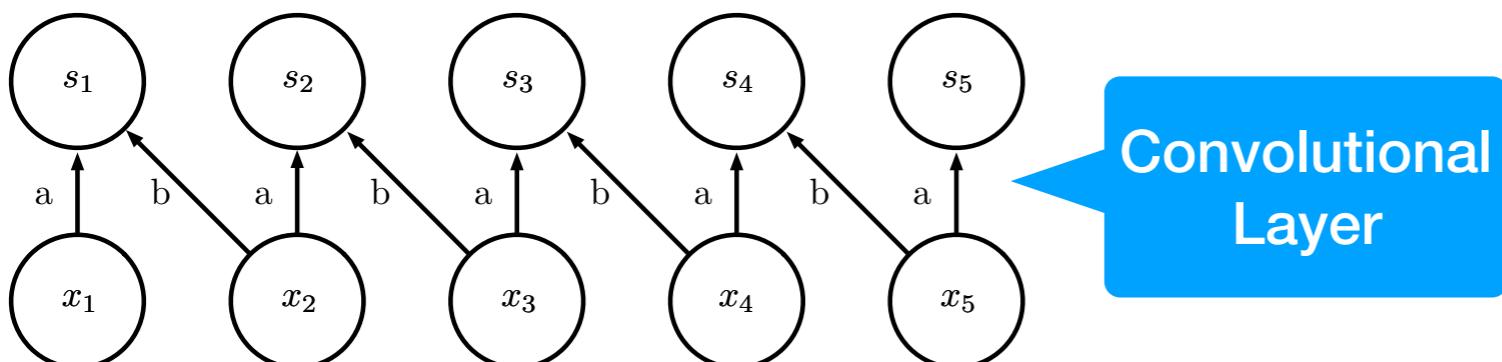
j: output row

k: output column

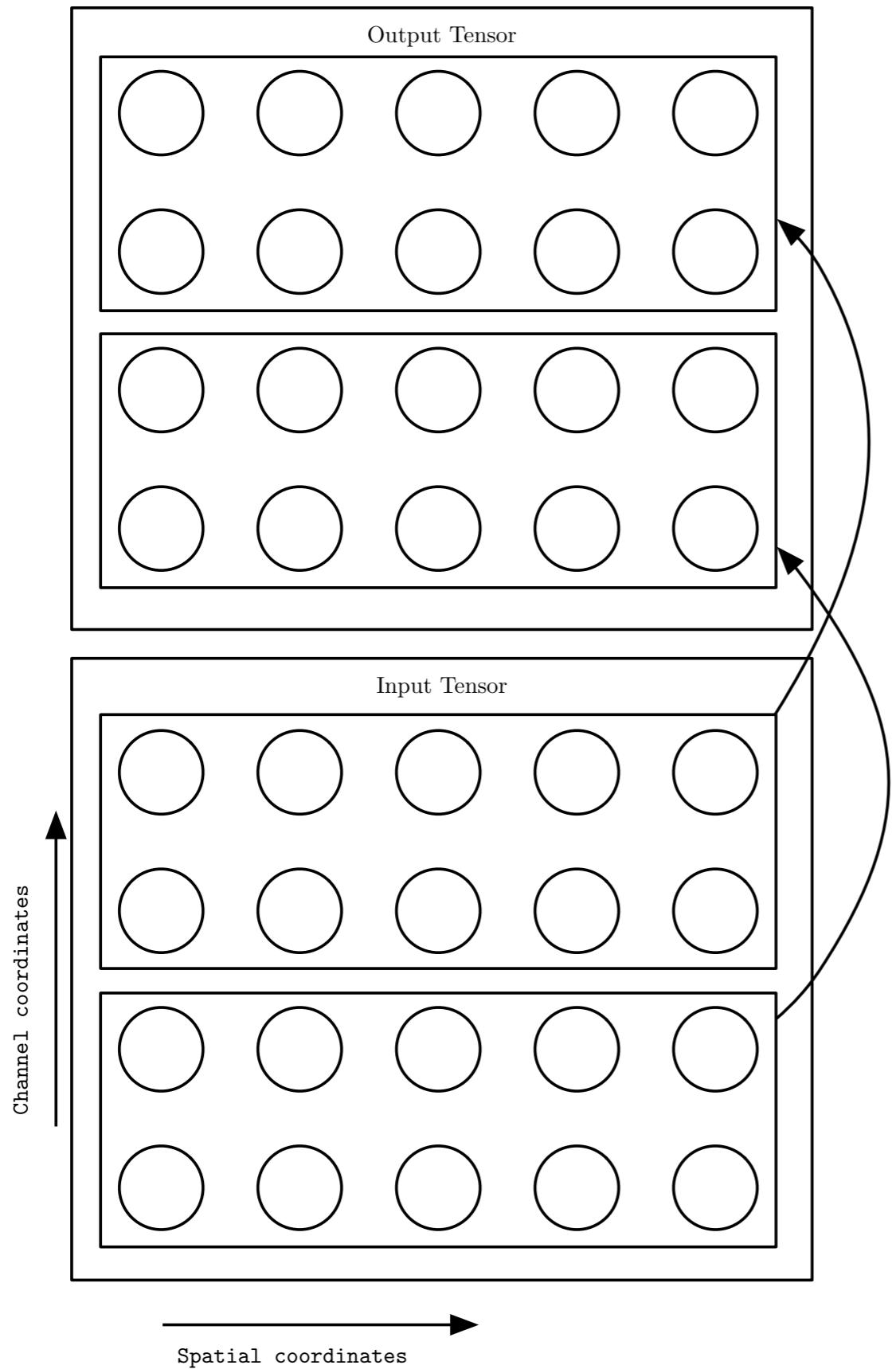
l: input channel

m: row offset within input

n: column offset within input



特徴が小さい空間の関数、しかも重複しないことがわかったら、Locally Connected Layerが役に立つ。例えば、画像が顔かを判定する時、画像の下半分に口を探せばOK。

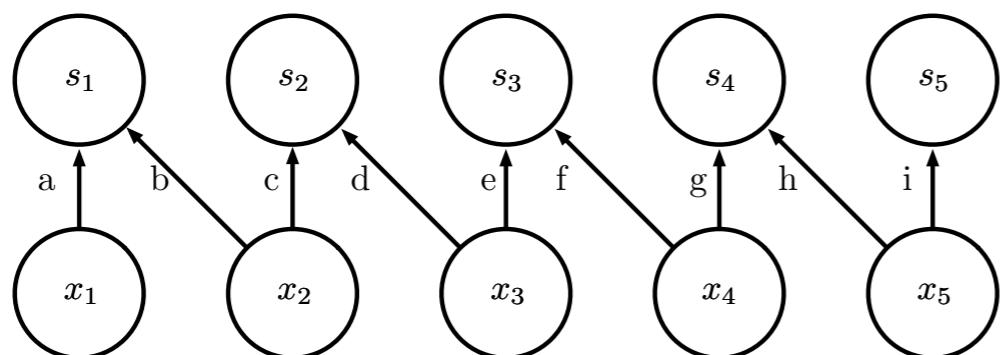


計算コストを更に低減するため、**Connectivity**に更に制限できる。例えば、左の図のように、**Input**の1-2番目チャネルと**Output**の1-2番目チャネルを繋げて、**Input**の3-4番目チャネルと**Output**の3-4番目チャネルを繋げる。(A common way is to make the first m output channels connect to only the first n input channels, the second m output channels connect to only the second n input channels)

基本畳み込み関数のバリアント

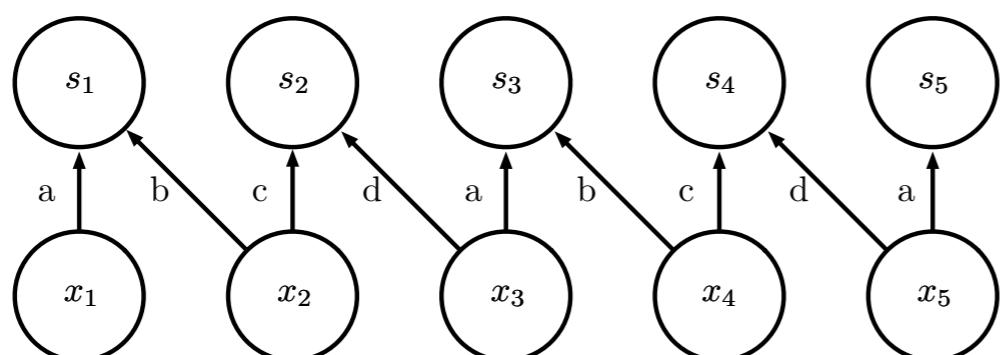
③Tiled (タイル) Convolution

Locally Connected LayerとConvolutional Layerの折衷:
 $Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n,j \% t+1, k \% t+1}$

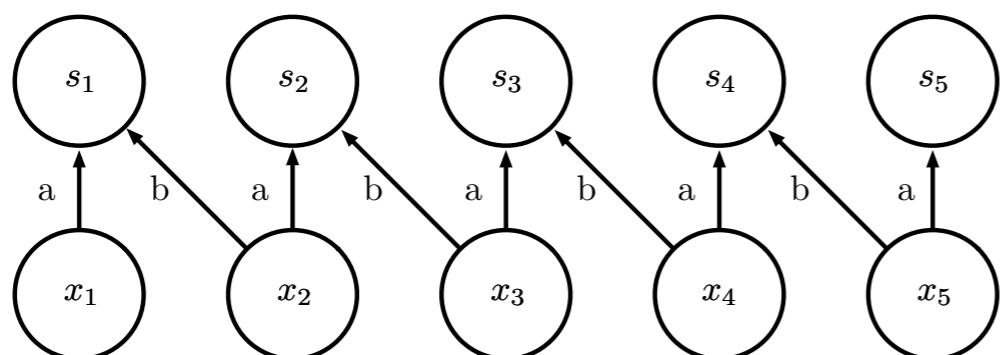


Locally
Connected
Layer

Kernelの選択。 $t \times t$ 個Kernelがある。
rowとcolumnに違う t もあり得る



Tiled
Convolution



Convolutional
Layer

基本畳み込み関数のバリアント

④ Backpropagationの計算

Strideあり
畳み込み: $c_{K,V,s}$ 畳み込み
output: Z Loss
Function: $J(V, K)$

Z に対する微分:

$$G_{i,j,k} = \frac{\partial}{\partial Z_{i,j,k}} J(V, K)$$

Kernelの重みに対する微分:

$$g(G, V, s)_{i,j,k,l} = \frac{\partial}{\partial K_{i,j,k,l}} J(V, K) = \sum_{m,n} G_{i,m,n} V_{j,(m-1)\times s+k, (n-1)\times s+l}$$

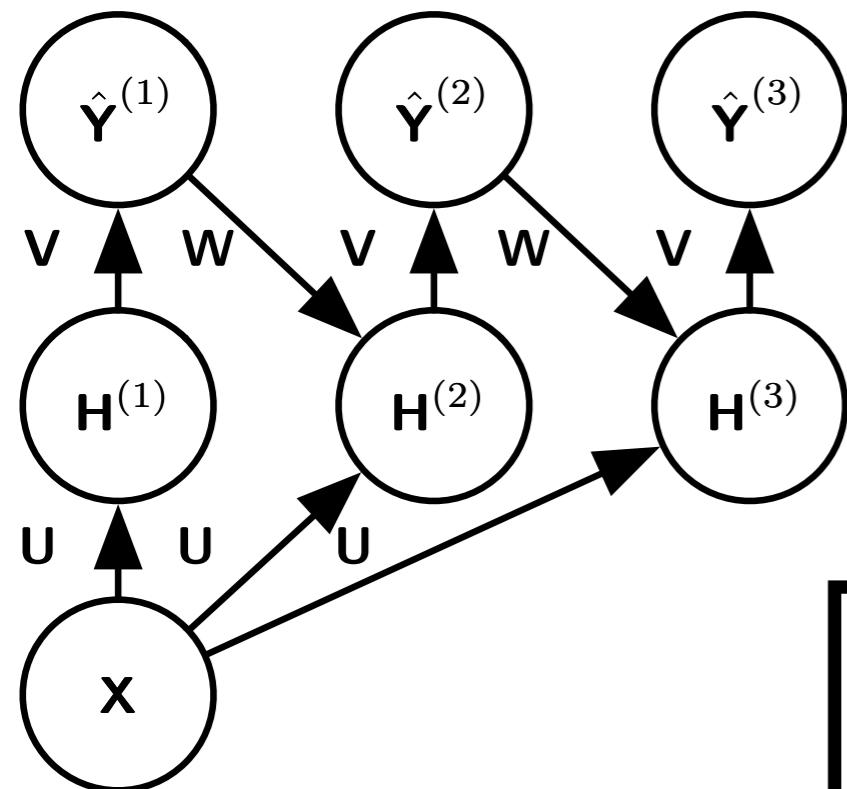
V に対する微分
(最低層以外) :

$$h(K, G, s)_{i,j,k} = \frac{\partial}{\partial V_{i,j,k}} J(V, K)$$

$$= \sum_{\substack{l, m \text{ s.t.} \\ (l-1) \times s + m = j}} \sum_{\substack{n, p \text{ s.t.} \\ (n-1) \times s + p = k}} \sum_q K_{q,i,m,p} G_{q,l,n}$$

Structured Outputs (構造化アウトプット)

畳み込みネットワークは高次元構造化されたオブジェクトを出力できる。例えば:



$$S_{i,j,k}$$

ピクセル (j,k) がクラス*i*に属する確率。
つまり、ピクセル毎のラベリングができる。

回帰型畳み込みネットワークのピクセルラベリングの例。リファインしている。
 U, V, W は全部Kernelです。

Efficient Convolution Algorithms (効率的畳み込みアルゴ) : もし d 次元Kernelが d 個ベクトルの外積になれば、 d 個1D畳み込みの方が効率的です。

d 個1次元畳み込みは、 $O(w^*d)$ の時間やメモリが必要。
1個 d 次元畳み込みは、 $O(w^d)$ の時間やメモリが必要。

	Single channel	Multichannel
1-D	<p>Audio 波形。時間軸に対して畳み込みする。</p> <p>Audio waveform: The axis we convolve over corresponds to time. We discretize time and measure the amplitude of the waveform once per time step.</p>	<p>Skeleton animation data: Animations of 3-D computer-rendered characters are generated by altering the pose of a “skeleton” over time. At each point in time, the pose of the character is described by a specification of the angles of each of the joints in the character’s skeleton. Each channel in the data we feed to the convolutional model represents the angle about one axis of one joint.</p>
2-D	<p>フーリエ変換した Audioデータ。周波数軸や時間軸がある。</p> <p>Audio data that has been preprocessed with a Fourier transform: We can transform the audio waveform into a 2-D tensor with different rows corresponding to different frequencies and different columns corresponding to different points in time. Using convolution in the time makes the model equivariant to shifts in time. Using convolution across the frequency axis makes the model equivariant to frequency, so that the same melody played in a different octave produces the same representation but at a different height in the network’s output.</p>	<p>Color image data: One channel contains the red pixels, one the green pixels, and one the blue pixels. The convolution kernel moves over both the horizontal and the vertical axes of the image, conferring translation equivariance in both directions.</p>
3-D	<p>体積データ。例えば、コンピュータ断層撮影 (CT)。</p> <p>Volumetric data: A common source of this kind of data is medical imaging technology, such as CT scans.</p>	<p>Color video data: One axis corresponds to time, one to the height of the video frame, and one to the width of the video frame.</p>

Skeletonアニメーション。Poseは3つチャネル。

カラー画像。

畳み込みネットワークの入力データタイプ

カラービデオ。

The Neuroscientific (神経科学) Basis for Convolutional Networks

① 置み込みはV1 (Primary Visual Cortex、一次視覚野) のプロパティを取っている

① V1は2Dの構造です。網膜の画像をミラーリングする。CNNも2D Mapで特徴を取っている。CNNのConvolution State。V1 is arranged in a spatial map.

② V1には簡単細胞 (simple cells) がある。小さい空間内の特徴を検出する。CNNのDetector Stage。

③ V1には複雑細胞 (complex cells) がある。小さい空間移動に不变。CNNのPooling Stage。

④ Medical Temporal Lobe (側頭葉、そくとうよう) におばあさん細胞 (grandmother cells) がある。おばあさんの写真、もしくは「おばあさん」という文字を見ると反応する。他のいろんな対象の専用細胞がある。

retina → LGN (外側膝状体、がいそくしつじょうたい、lateral geniculate nucleus)
→ V1 → V2 → V4 → IT (下側頭葉、inferotemporal cortex)

ITはCNNの最後層と一番似ている。上記の流れは100ms以内で終わる。オブジェクトを続けて見ると情報の逆流が始まる。つまり、低層の活性化を更新する。

The Neuroscientific (神経科学) Basis for Convolutional Networks②畳み込みと哺乳類視覚システムの大きい違い

- ①人の目は大分非常に低解像度。foveaだけは高解像度。foveaは腕の長さで親指の爪の大きさをはっきり観れる。人の目はSaccadeして関連性が高い情報を取る。しかし、CNNの入力は高解像度の大きい写真。
- ②視覚システムは他の感覚と繋がっている。例えば聴覚、考え、気持ち。CNNは視覚だけ。
- ③視覚システムは、物体検出だけじゃなく、シーンの理解、物体間の関係、体動き用の3D情報が全部できる。
- ④V1などは高いレベルからのフィードバックに非常に影響されている。CNNにはフィードバックもあるが、切実な向上はまだない。
- ⑤CNNが使っている活性化関数やPooling関数があるが、視覚システムは全く違う関数を使っているかもしれない。簡単細胞や複雑細胞もパラメータが違う同じ細胞かもしれない。

The Neuroscientific (神経科学) Basis for Convolutional Networks③V1の重み関数: Gabor

Gabor関数:

$$x' = (x - x_0)\cos(\tau) + (y - y_0)\sin(\tau)$$

$$y' = -(x - x_0)\sin(\tau) + (y - y_0)\cos(\tau)$$

$$w(x, y; \alpha, \beta_x, \beta_y, f, \phi, x_0, y_0, \tau) = \alpha \exp(-\beta_x x'^2 - \beta_y y'^2) \cos(f x' + \phi)$$

関数

Response関数:

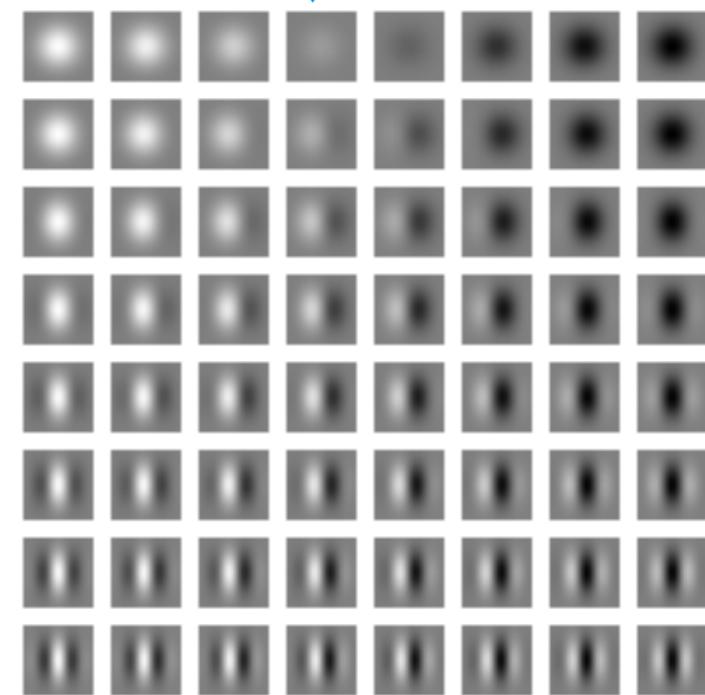
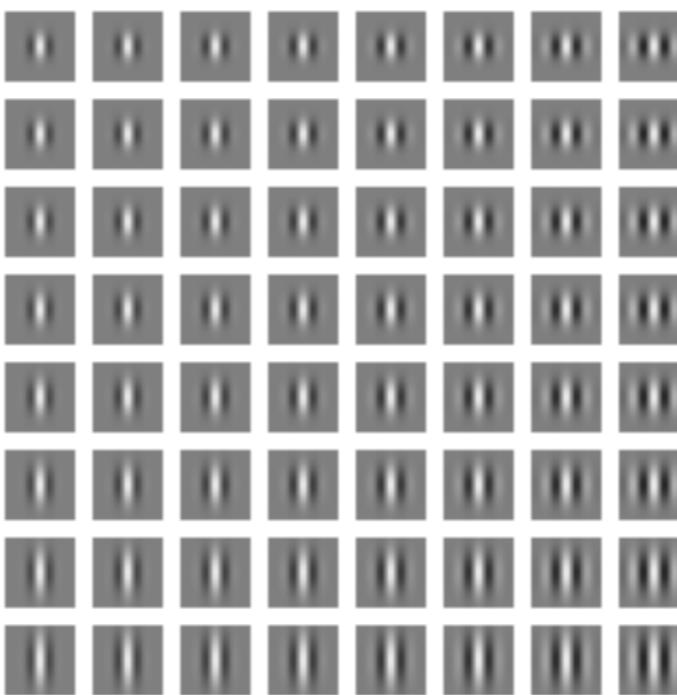
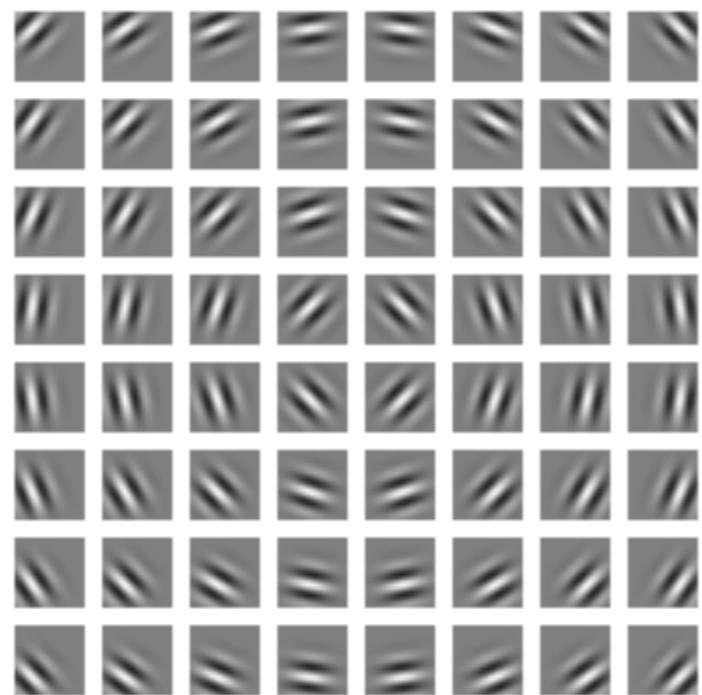
$$s(I) = \sum_{x \in \mathbb{X}} \sum_{y \in \mathbb{Y}} \omega(x, y) I(x, y)$$

x0,y0は座標系の原点
τは座標系の向き

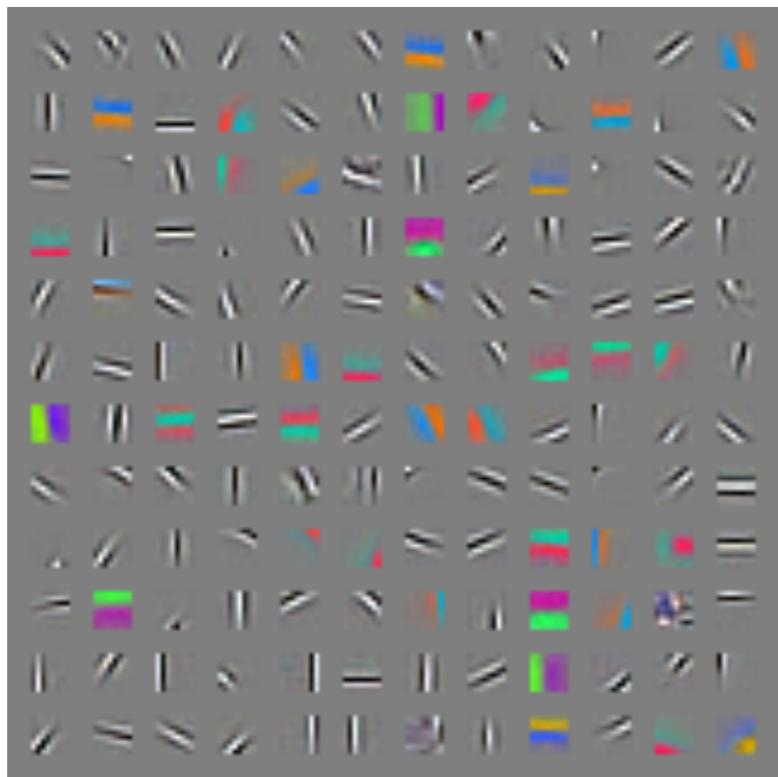
βx,βyはresponse域の大きさを決める

fは周期
φは位相

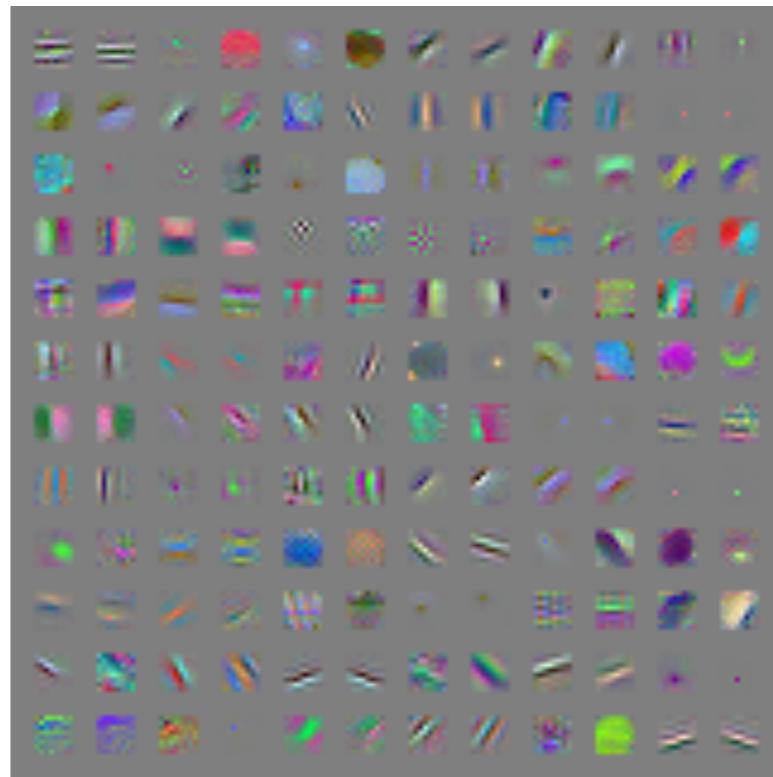
パラメータを変えると、Gabor関数(簡単細胞のResponseパターン)も変わる



The Neuroscientific (神経科学) Basis for Convolutional Networks④多数現代学習アルゴが学習した特徴はGabor関数と似ている



教師なし学習アルゴ（spike and slab sparse coding）が学習した重み



教師ありCNNが学習した第一層Kernel