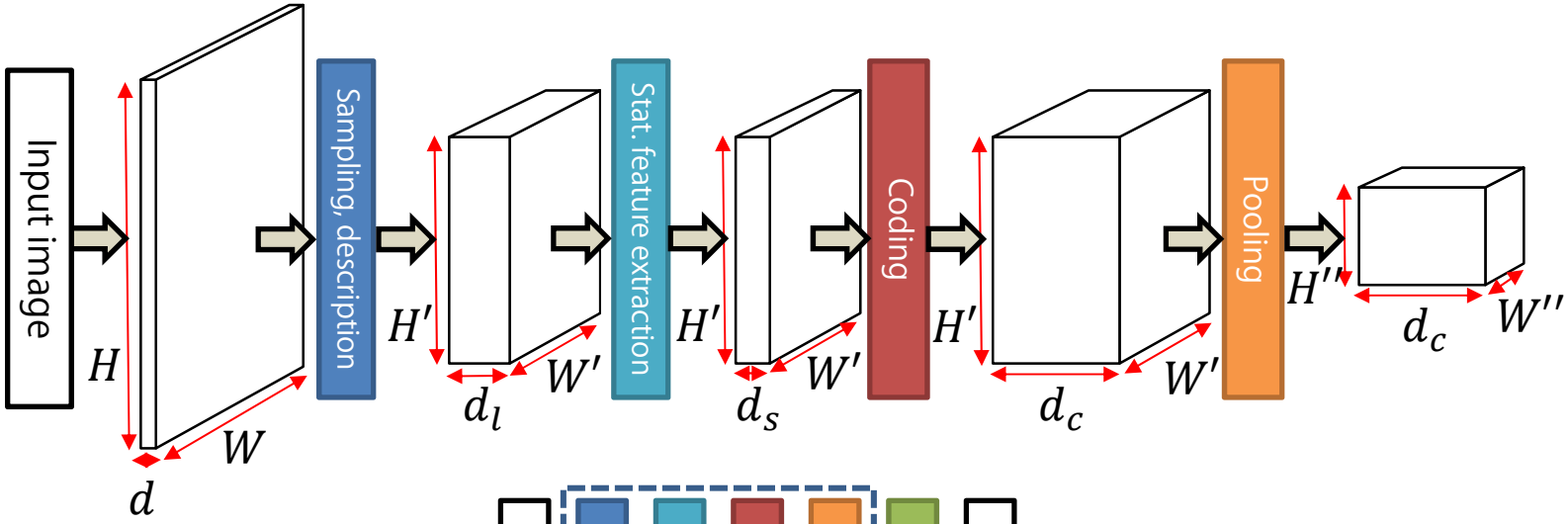
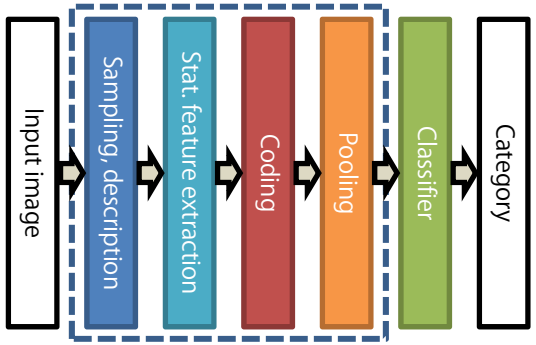
知能情報論
深層学習
畳み込みニューラルネットワーク

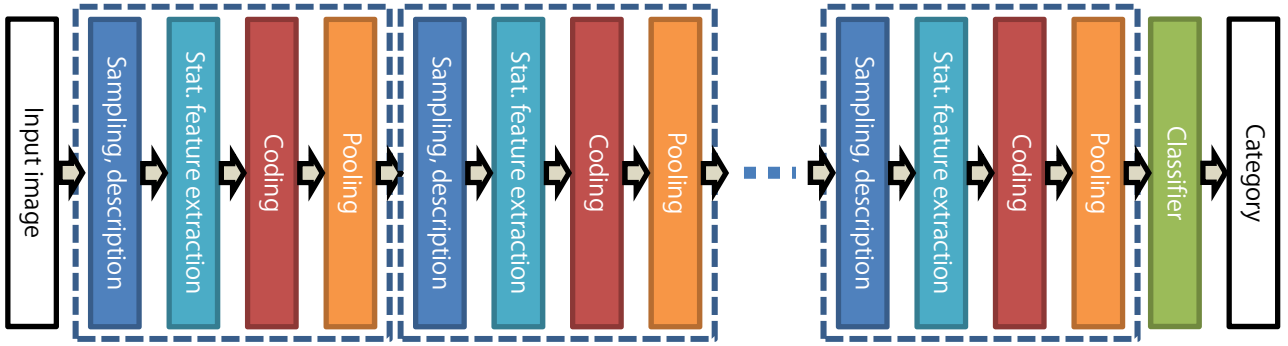
2016年6月15日
東京大学　大学院情報理工学系研究科
原田達也

Mapping Function for Visual Recognition



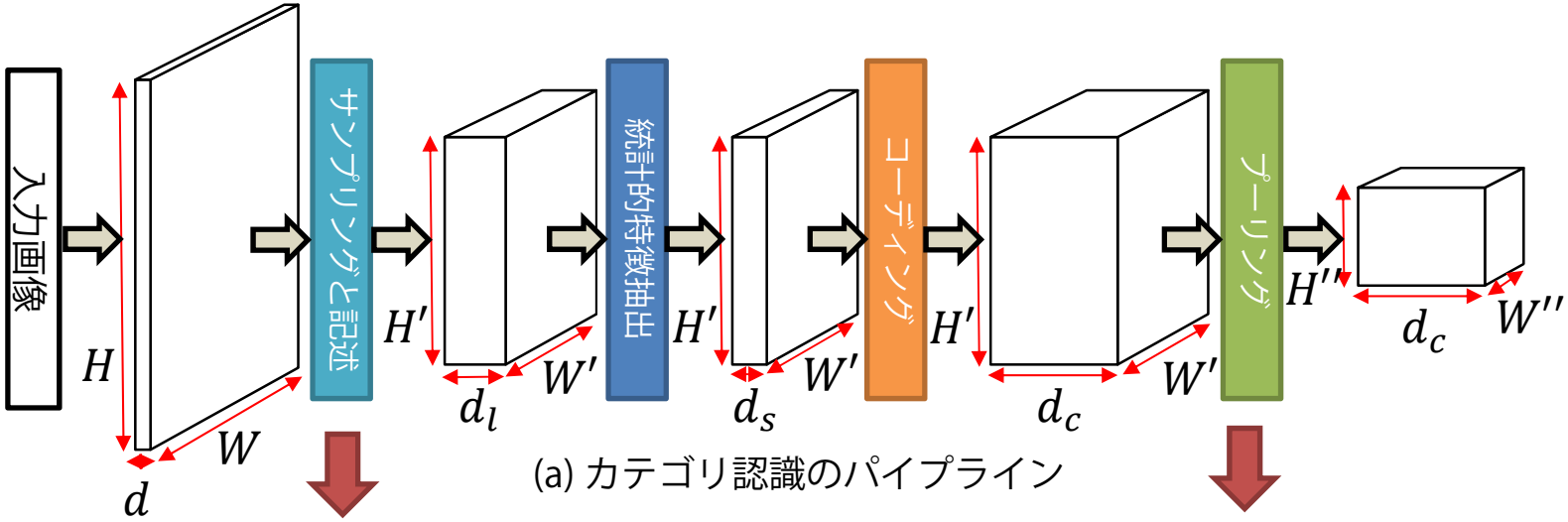


(a) Shallow structure

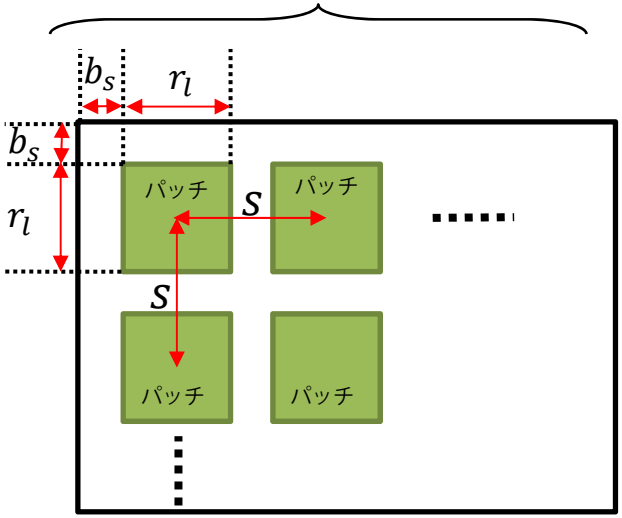


(b) Deep structure

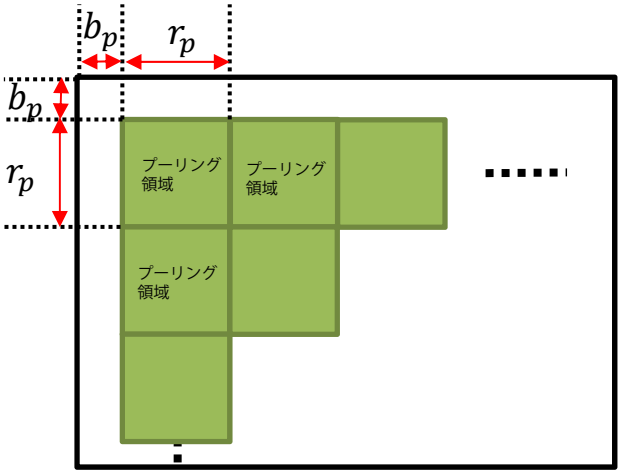
画像認識のパイプライン（写像関数）



(a) カテゴリ認識のパイプライン



(b) 局所特徴のサンプリング

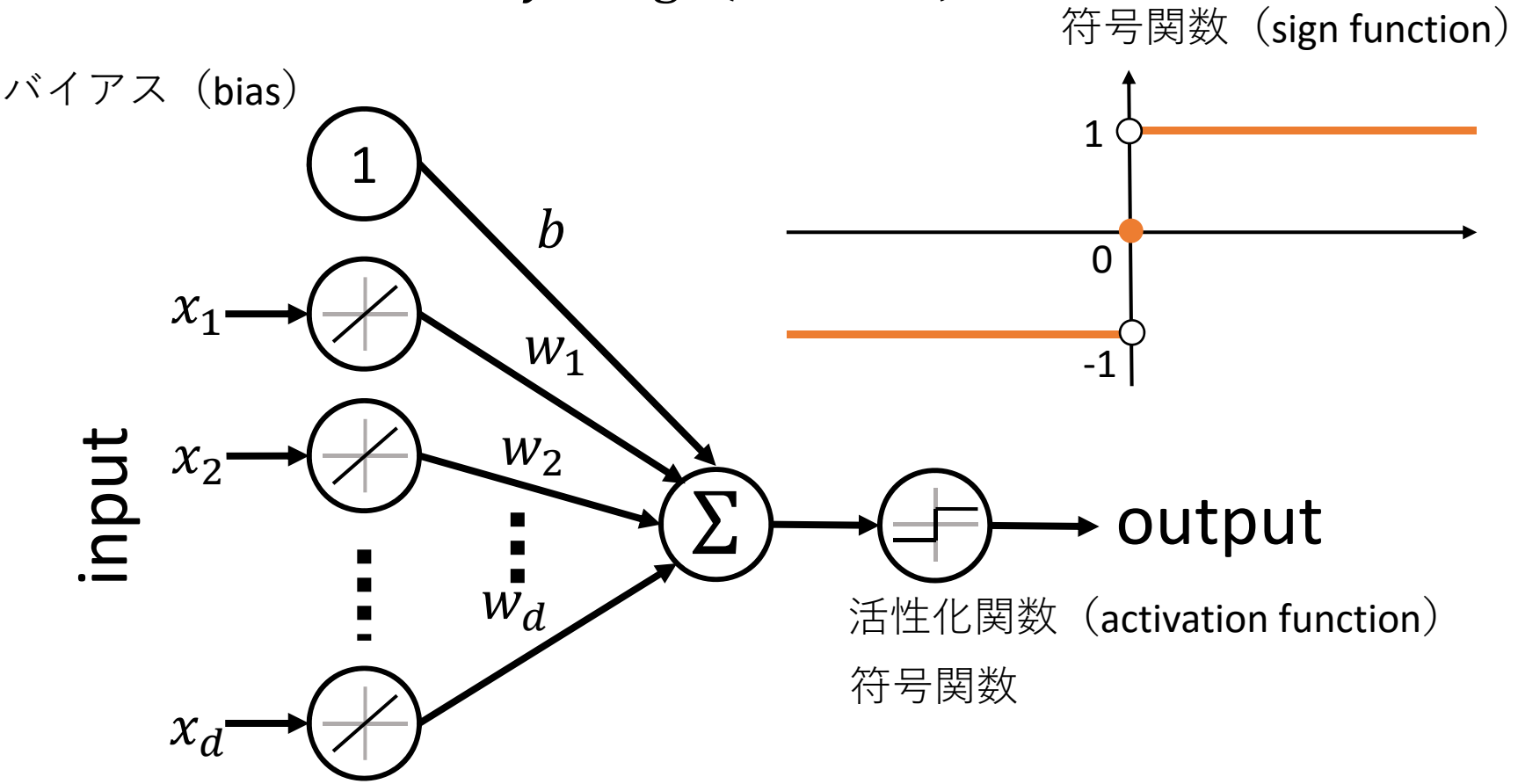


(c) プーリング

単一ニューロンモデル

- パーセプトロン（perceptron）
  - 線形識別器の基本
  - 活性化関数は符号関数（sign function）

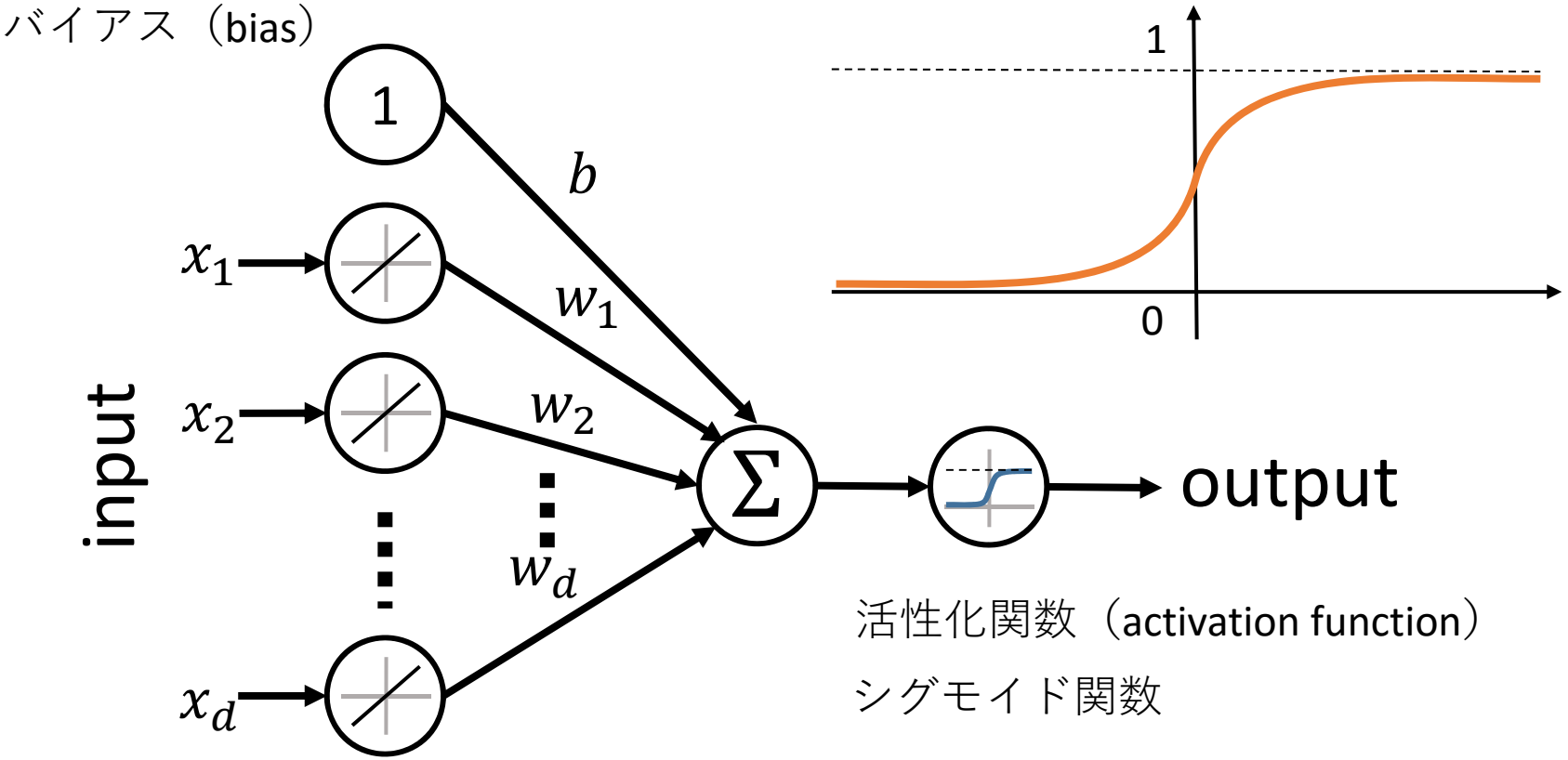
$$y = \mathrm{sgn}(\boldsymbol{w}^T \boldsymbol{x} + b)$$



単一ニューロンモデル

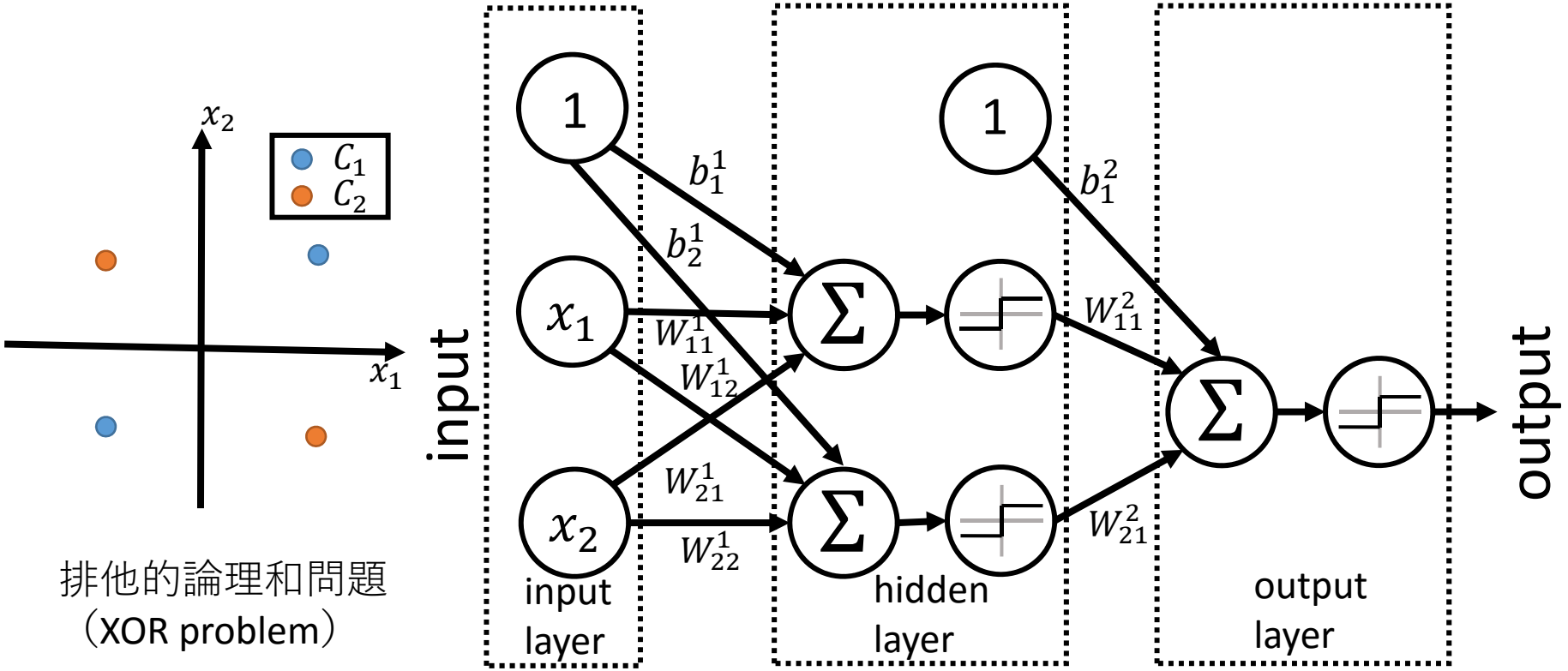
- ロジスティック回帰
  - パーセプトロンの活性化関数をシグモイド関数に置き換えた手法

$$y = \sigma(\boldsymbol{w}^T \boldsymbol{x} + b)$$



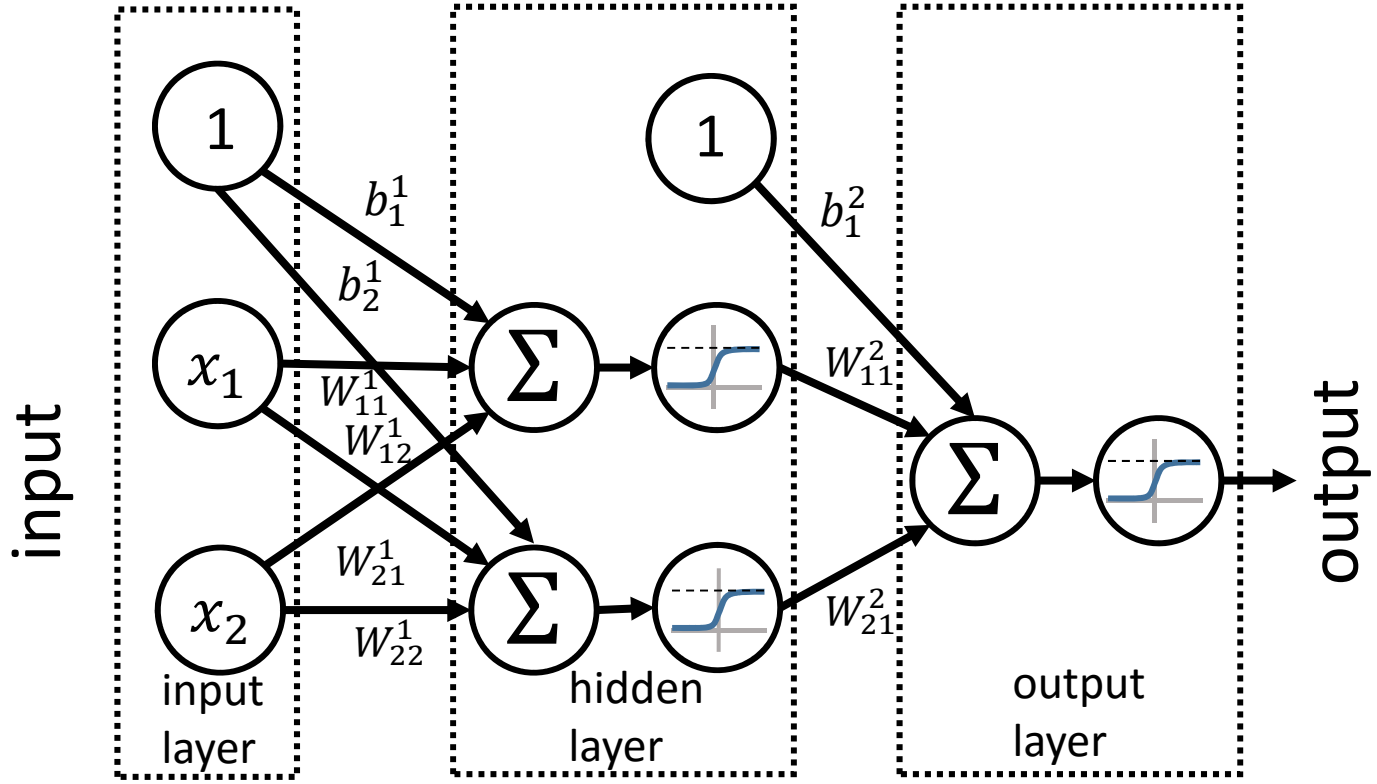
線形分離不可能なデータの識別

- 線形識別器は，例えば２次元の特徴空間では，直線１本で分離できるような問題しか対応できない．
  - 例）排他的論理和問題では一つの線形識別器では解けない．
- 線形識別器を２つ組み合わせた識別器を考えると，適切な重みを設定することで排他的論理和問題を解くことが可能



フィードフォワードニューラルネットワーク

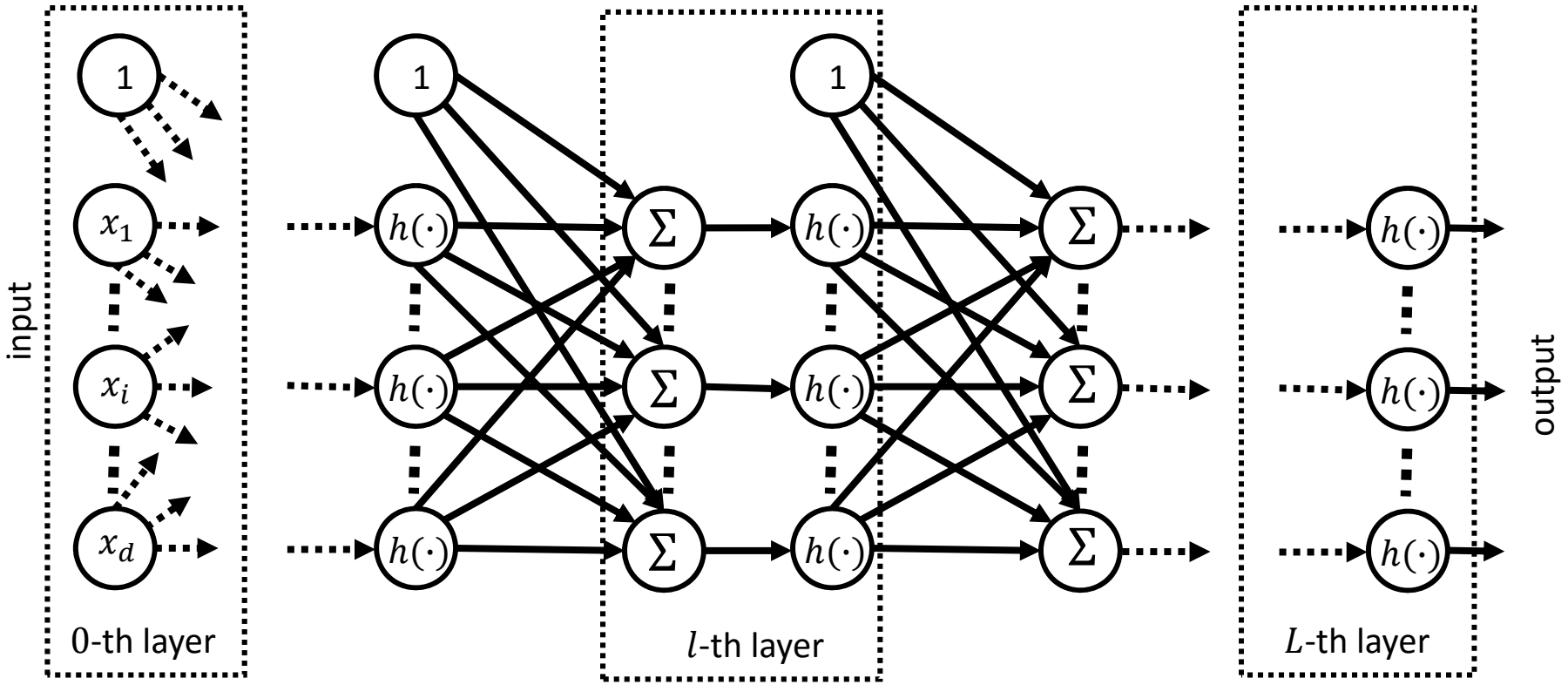
- フィードフォワードニューラルネットワーク（feedforward neural network）
  - 活性化関数としてシグモイド関数のような微分可能な関数が利用される.



入力層（input layer）隠れ層（hidden layer）出力層（output layer）

フィードフォワードニューラルネットワーク

- 隠れ層のユニット数や層の数を増やせば，より高い表現能力を持つ識別器を構成可能
- フィードフォワードニューラルネットワークは入力層，隠れ層，出力層の３種類の層で構成される．
- 隠れ層は複数の層を持つことが可能
- 本講義では，入力層は0番目の層，出力層をL番目の層とする．
- 活性化関数を$y = h(z)$としている．



ネットワークの計算と表記

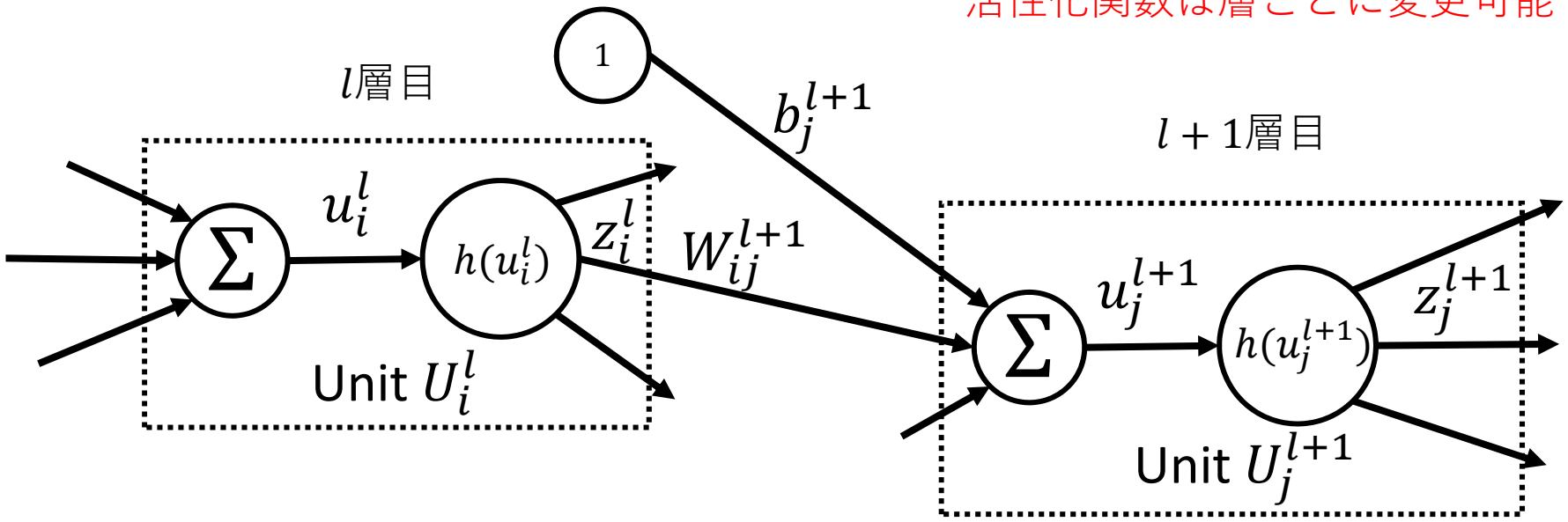
- l層目のユニット集合：$\{U_i^l\}_{i=1}^{|U^l|}$
- ユニットU_i^lとユニットU_j^{l+1}間の結合重み：W_{ij}^{l+1}
- l層目と$l+1$層目の結合重み：$W^{l+1} \in \mathbb{R}^{|U^l| \times |U^{l+1}|}$
- $l+1$層目へのバイアス：$\boldsymbol{b}^{l+1} \in \mathbb{R}^{|U^{l+1}|}$

$\boldsymbol{u}^{l+1} \in \mathbb{R}^{|U^{l+1}|}, \boldsymbol{z}^l \in \mathbb{R}^{|U^l|}$

ユニットへの入力　$u_j^{l+1} = \sum_{i=1}^{|U^l|} W_{ij}^{l+1} z_i^l + b_j^{l+1}$ ⇒ $\boldsymbol{u}^{l+1} = (W^{l+1})^T \boldsymbol{z}^l + \boldsymbol{b}^{l+1}$

ユニットの出力　$z_j^{l+1} = h(u_j^{l+1})$ ⇒ $\boldsymbol{z}^{l+1} = h^l(\boldsymbol{u}^{l+1})$

活性化関数は層ごとに変更可能



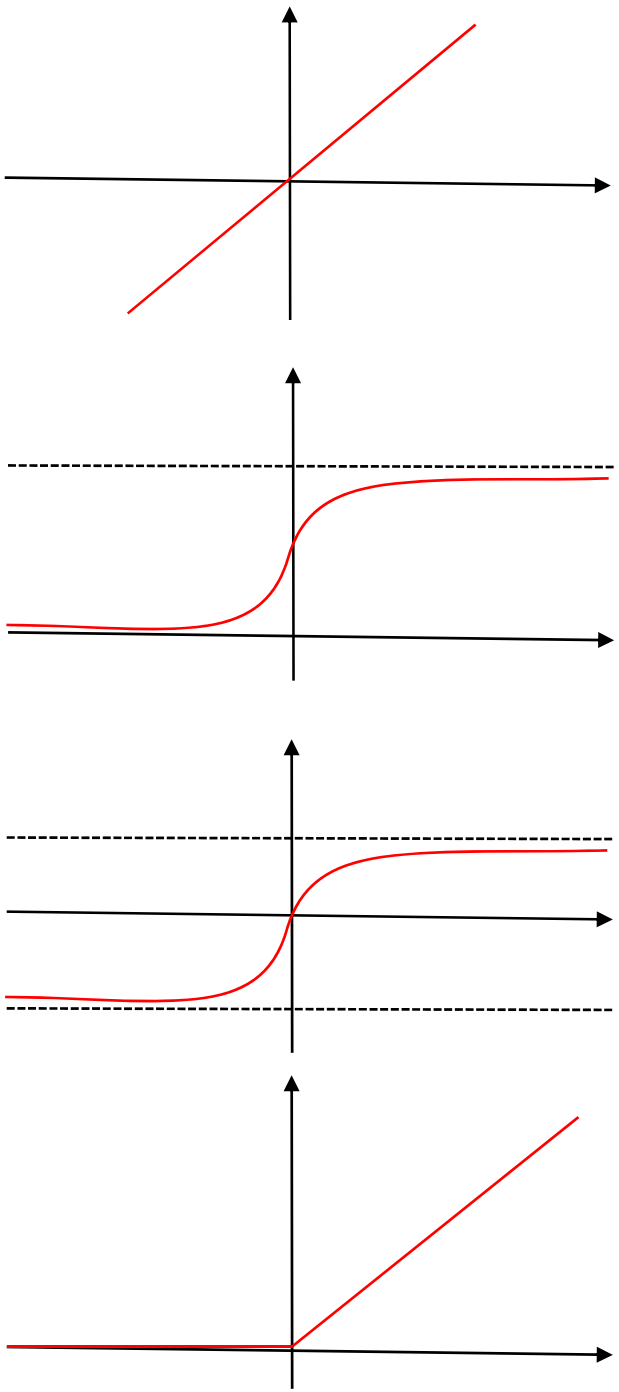
順向き伝播のアルゴリズム

--- Algorithm 1: forward propagation ---

input $\boldsymbol{x}, W = \{W^1 \cdots W^L\}, B = \{\boldsymbol{b}^1 \cdots \boldsymbol{b}^L\}$
output $\boldsymbol{y}$
set $\boldsymbol{z}^0 = \boldsymbol{x}$
for $l = 1, \ldots, L$ do
    % input of unit l
    $\boldsymbol{u}^l = (W^l)^T \boldsymbol{z}^{l-1} + \boldsymbol{b}^l$
    % output of unit l
    $\boldsymbol{z}^l = h^l(\boldsymbol{u}^l)$
end for
return $\boldsymbol{y} = \boldsymbol{z}^L$

活性化関数の例

- 線形関数
  - $h(z) = z$
  - $h'(z) = 1$
- シグモイド関数
  - $h(z) = \sigma(z) = \frac{1}{1+\exp(-z)}$
  - $h'(z) = \sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big)$
- ソフトマックス関数
  - $h_i(z) = \frac{\exp(z_i)}{\sum_k \exp(z_k)}$
  - $\frac{\partial h_i}{\partial h_j} = h_i(z)(\delta_{ij} - h_j(z))$
- tanh関数
  - $h(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
  - $h'(z) = 1 - h(z)^{\,2}$
- ReLU（Rectified Linear Unit）関数
  - $h(z) = \max(0, z)$
  - $h'(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0 \end{cases}$



損失関数と出力層の活性化関数の例

- 平均二乗誤差
  - 回帰モデルでよく利用される.
  - 活性化関数：$\boldsymbol{y}_i = h^L(\boldsymbol{u}_i^L)$
  - $J = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\|\boldsymbol{y}_i - \boldsymbol{t}_i\|^2\right) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\left\|h^L(\boldsymbol{u}_i^L) - \boldsymbol{t}_i\right\|^2\right)$
  - 出力層の微分：$\frac{\partial J}{\partial \boldsymbol{u}^L} = \frac{1}{N}\sum_{i=1}^{N} h^{L\prime}(\boldsymbol{u}_i^L)\odot\left(h^L(\boldsymbol{u}_i^L) - \boldsymbol{t}_i\right)$
  - 活性化関数が線形$\boldsymbol{y}_i = \boldsymbol{u}_i^L$の場合
    - $J = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\left\|\boldsymbol{u}_i^L - \boldsymbol{t}_i\right\|^2\right)$
    - $\frac{\partial J}{\partial \boldsymbol{u}^L} = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{u}_i^L - \boldsymbol{t}_i\right)$

アダマール積（Hadamard product）
要素毎の積

- 多クラス交差エントロピー誤差
  - 多クラス分類でよく利用される.
  - 活性化関数：$y_{ij} = \frac{\exp(\boldsymbol{u}_{ij})}{\sum_k \exp(\boldsymbol{u}_{kj})}$
  - $J = -\sum_{i=1}^{N}\sum_{k=1}^{K} I(t_i = k)\ln\frac{\exp\left(\boldsymbol{u}_{i,k}^L\right)}{\sum_{j=1}^{K}\exp\left(\boldsymbol{u}_{i,j}^L\right)}$
  - 出力層の微分：$\frac{\partial J}{\partial \boldsymbol{u}^L} = \frac{1}{N}\sum_{i=1}^{N}\left(h^L(\boldsymbol{u}_i^L) - \boldsymbol{t}_i\right)$

ネットワーク構造の例

多クラス識別の場合

線形関数

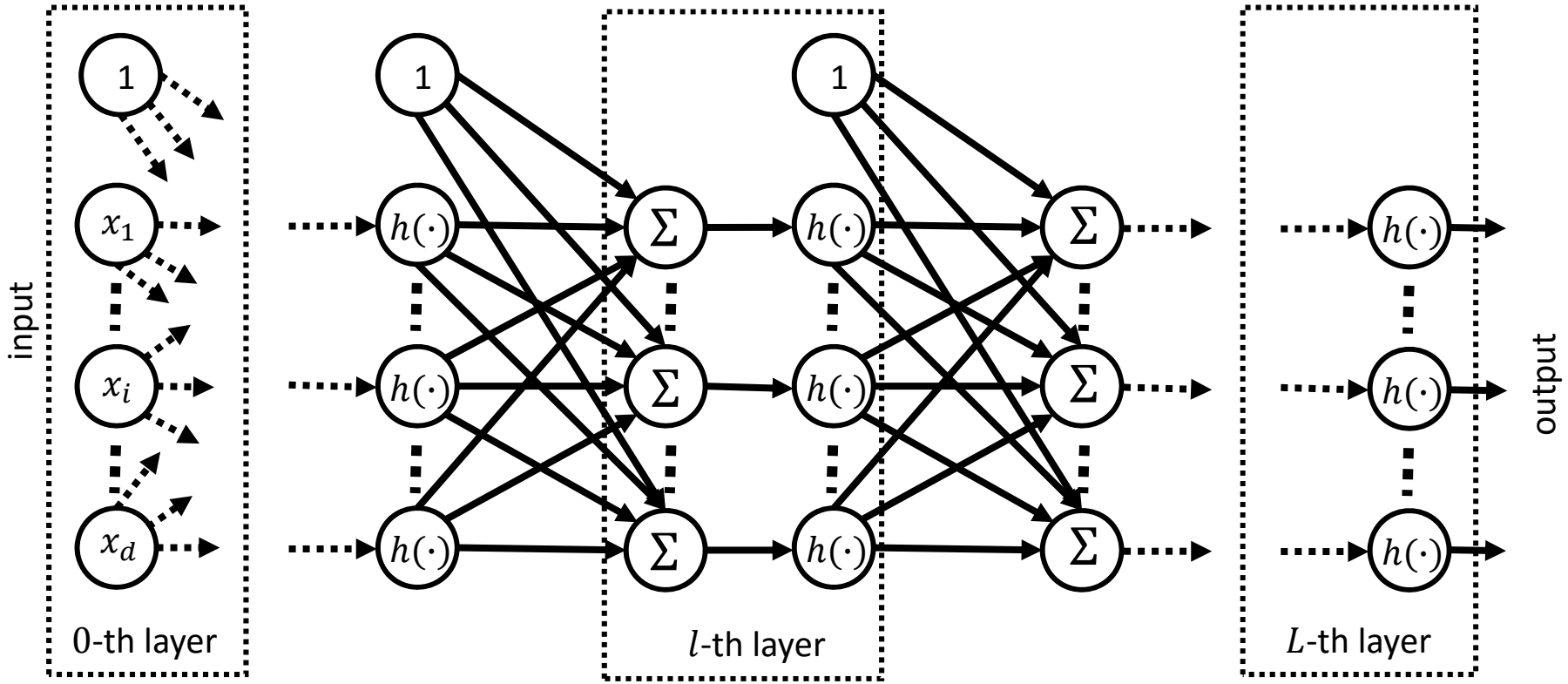
$$h^0(u) = u$$

シグモイド関数

$$h^l(u) = \frac{1}{1 + \exp(-u)}$$

ソフトマックス関数

$$h_i^L(\boldsymbol{u}) = \frac{\exp(u_i)}{\sum_k \exp(u_k)}$$



損失関数
交差エントロピー誤差

$$J = -\sum_{i=1}^{N}\sum_{k=1}^{K} I(t_i = k) \ln \frac{\exp\left(\boldsymbol{u}_{i,k}^L\right)}{\sum_{j=1}^{K} \exp\left(\boldsymbol{u}_{i,j}^L\right)}$$

フィードフォワードニューラルネットワークの学習

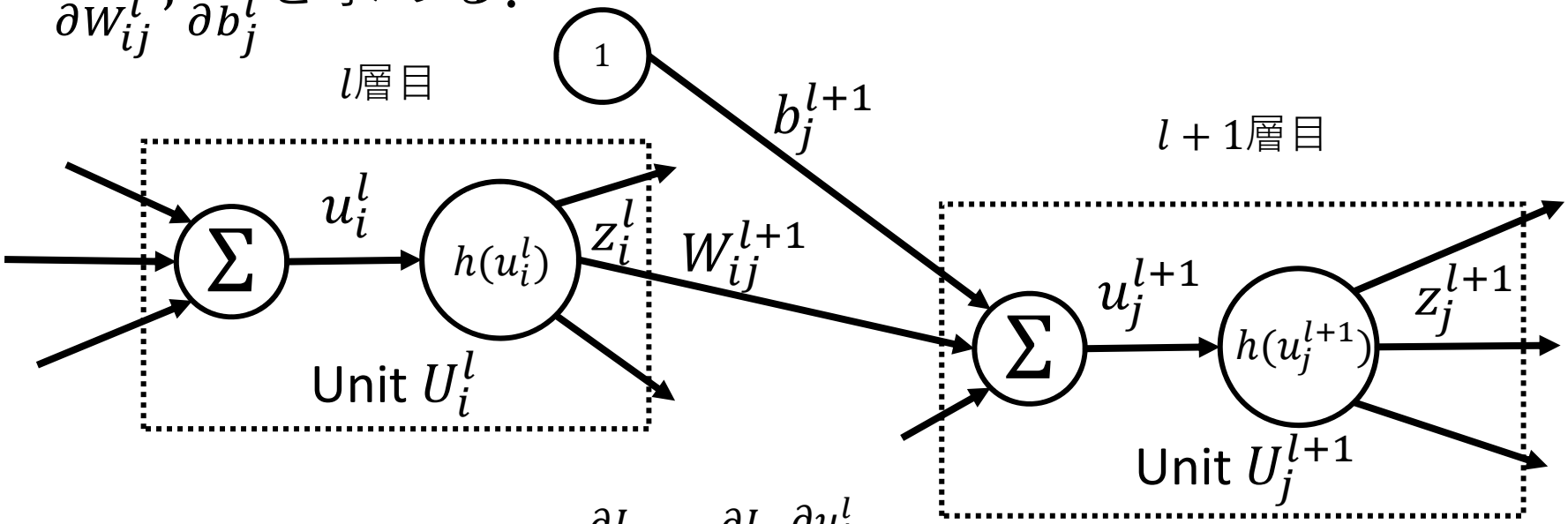
- 訓練データ集合$\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}_{i=1}^{N}$が与えられたときに，損失関数$J(W, B)$を（以後$J$と書く）最小化する重みパラメータ$W = \{W^1 \cdots W^L\}$とバイアスパラメータ$B = \{\boldsymbol{b}^1 \cdots \boldsymbol{b}^L\}$を求める．

- ここでは勾配降下法によりパラメータを求める．
  - 損失関数が非凸であるために局所解に陥る可能性があるが，実用上うまくいく場合が多い．
  - $W^l = W^l - \varepsilon \frac{\partial J}{\partial W^l}, \quad \boldsymbol{b}^l = \boldsymbol{b}^l - \varepsilon \frac{\partial J}{\partial \boldsymbol{b}^l}$

- パラメータの初期化
  - パラメータの初期値を0に近いランダムな値を設定．
  - $W_{ij}^l \sim \mathcal{N}(0, \epsilon^2), \quad b_j^l \sim \mathcal{N}(0, \epsilon^2)$

フィードフォワードニューラルネットワークの学習

- $\frac{\partial J}{\partial W_{ij}^l}, \frac{\partial J}{\partial b_j^l}$を求める.



連鎖法則（chain rule）

$$\frac{\partial J}{\partial W_{ij}^l} = \frac{\partial J}{\partial u_j^l}\frac{\partial u_j^l}{\partial W_{ij}^l}$$

$$u_j^{l+1} = \sum_{i=1}^{|U^l|} W_{ij}^{l+1} h(u_i^l) + b_j^{l+1}$$

誤差（error）

$$\frac{\partial J}{\partial u_j^l} = \delta_j^l$$

$$\frac{\partial u_j^l}{\partial W_{ij}^l} = z_i^{l-1}$$

$$\frac{\partial J}{\partial W_{ij}^l} = \delta_j^l z_i^{l-1}, \quad \frac{\partial J}{\partial b_j^l} = \delta_j^l$$

ベクトルでまとめて書くと

$$\frac{\partial J}{\partial W^l} = \boldsymbol{z}^{l-1}(\boldsymbol{\delta}^l)^T, \quad \frac{\partial J}{\partial \boldsymbol{b}^l} = \boldsymbol{\delta}^l$$

フィードフォワードニューラルネットワークの学習

- 誤差δ_i^lを求める.

$$\frac{\partial J}{\partial W_{ij}^l} = \delta_j^l z_i^{l-1}$$

順向き伝播で計算しておく

連鎖法則（chain rule）

誤差（error） $\delta_i^l = \frac{\partial J}{\partial u_i^l} = \sum_j^{|U^{l+1}|} \frac{\partial J}{\partial u_j^{l+1}} \frac{\partial u_j^{l+1}}{\partial u_i^l}$

$$u_j^{l+1} = \sum_{i=1}^{|U^{l+1}|} W_{ij}^{l+1} h(u_i^l) + b_j^{l+1}$$

上位層の誤差 δ_j^{l+1}

$$\frac{\partial u_j^{l+1}}{\partial u_i^l} = W_{ij}^{l+1} h'(u_i^l)$$

$$\delta_i^l = h'(u_i^l) \sum_{j=1}^{|U^{l+1}|} \delta_j^{l+1} W_{ij}^{l+1}$$

ベクトルでまとめて書くと

$$\boldsymbol{\delta}^l = h'(\boldsymbol{u}^l) \odot (W^{l+1} \boldsymbol{\delta}^{l+1})$$

現在の層の誤差は一つ上の層の誤差から計算可能

---- Algorithm 2: backward propagation (batch learning)---

input $X = (\boldsymbol{x}_1 \ \cdots \ \boldsymbol{x}_N)^T, W = \{W^1 \ \cdots \ W^L\}, B = \{\boldsymbol{b}^1 \ \cdots \ \boldsymbol{b}^L\}$

output $W = \{W^1 \ \cdots \ W^L\}, B = \{\boldsymbol{b}^1 \ \cdots \ \boldsymbol{b}^L\}$

initialize $W = \{W^1 \ \cdots \ W^L\}, B = \{\boldsymbol{b}^1 \ \cdots \ \boldsymbol{b}^L\}$

for $m = 1, \dots, M$ do % M: maximum iterations

for $l = 1, \dots, L$ do $\Delta W^l = \text{zeros}\left(\text{size}\left(W^l\right)\right), \Delta \boldsymbol{b}^l = \text{zeros}\left(\text{size}\left(\boldsymbol{b}^l\right)\right)$ end for

for $\boldsymbol{z}^0 = \boldsymbol{x}_i, i = 1, \dots, N$ do

% forward propagation

for $l = 1, \dots, L$ do $\boldsymbol{u}^l = (W^l)^T \boldsymbol{z}^{l-1} + \boldsymbol{b}^l, \ \boldsymbol{z}^l = h^l(\boldsymbol{u}^l)$ end for

% error of output layer

$\boldsymbol{\delta}^L = \frac{\partial J}{\partial \boldsymbol{u}^L}$

$\Delta W^L = \Delta W^L + \boldsymbol{z}^{L-1}(\boldsymbol{\delta}^L)^T, \ \Delta \boldsymbol{b}^L = \Delta \boldsymbol{b}^L + \boldsymbol{\delta}^L$

% back propagation

for $l = L - 1, \dots, 1$ do

$\boldsymbol{\delta}^l = h^{l\prime}(\boldsymbol{u}^l) \odot \left(W^{l+1} \boldsymbol{\delta}^{l+1}\right)$ % error

$\frac{\partial J}{\partial W^l} = \boldsymbol{z}^{l-1}(\boldsymbol{\delta}^l)^T, \ \frac{\partial J}{\partial \boldsymbol{b}^l} = \boldsymbol{\delta}^l$ % gradient

$\Delta W^l = \Delta W^l + \frac{\partial J}{\partial W^l}, \ \Delta \boldsymbol{b}^l = \Delta \boldsymbol{b}^l + \frac{\partial J}{\partial \boldsymbol{b}^l}$

end for

end for

% batch learning

for $l = 1, \dots, L$ do $W^l = W^l - \varepsilon \frac{1}{N} \Delta W^l, \ \boldsymbol{b}^l = \boldsymbol{b}^l - \varepsilon \frac{1}{N} \Delta \boldsymbol{b}^l$ end for

end for

---- Algorithm 3: backward propagation (online learning)---

input $X = (\boldsymbol{x}_1 \ \cdots \ \boldsymbol{x}_N)^T, W = \{W^1 \ \cdots \ W^L\}, B = \{\boldsymbol{b}^1 \ \cdots \ \boldsymbol{b}^L\}$

output $W = \{W^1 \ \cdots \ W^L\}, B = \{\boldsymbol{b}^1 \ \cdots \ \boldsymbol{b}^L\}$

initialize $W = \{W^1 \ \cdots \ W^L\}, B = \{\boldsymbol{b}^1 \ \cdots \ \boldsymbol{b}^L\}$

for $m = 1, \dots, M$ do % M: maximum iterations

for $\boldsymbol{z}^0 = \boldsymbol{x}_i, i = 1, \dots, N$ do

% forward propagation

for $l = 1, \dots, L$ do $\boldsymbol{u}^l = (W^l)^T \boldsymbol{z}^{l-1} + \boldsymbol{b}^l, \ \boldsymbol{z}^l = h^l(\boldsymbol{u}^l)$ end for

% error of output layer

$\boldsymbol{\delta}^L = \frac{\partial J}{\partial \boldsymbol{u}^L}$

$\frac{\partial J}{\partial W^L} = \boldsymbol{z}^{L-1}(\boldsymbol{\delta}^L)^T, \ \frac{\partial J}{\partial \boldsymbol{b}^L} = \boldsymbol{\delta}^L$

% back propagation

for $l = L - 1, \dots, 1$ do

$\boldsymbol{\delta}^l = h^{l\prime}(\boldsymbol{u}^l) \odot \left(W^{l+1} \boldsymbol{\delta}^{l+1}\right)$ % error

$\frac{\partial J}{\partial W^l} = \boldsymbol{z}^{l-1}(\boldsymbol{\delta}^l)^T, \ \frac{\partial J}{\partial \boldsymbol{b}^l} = \boldsymbol{\delta}^l$ % gradient

end for

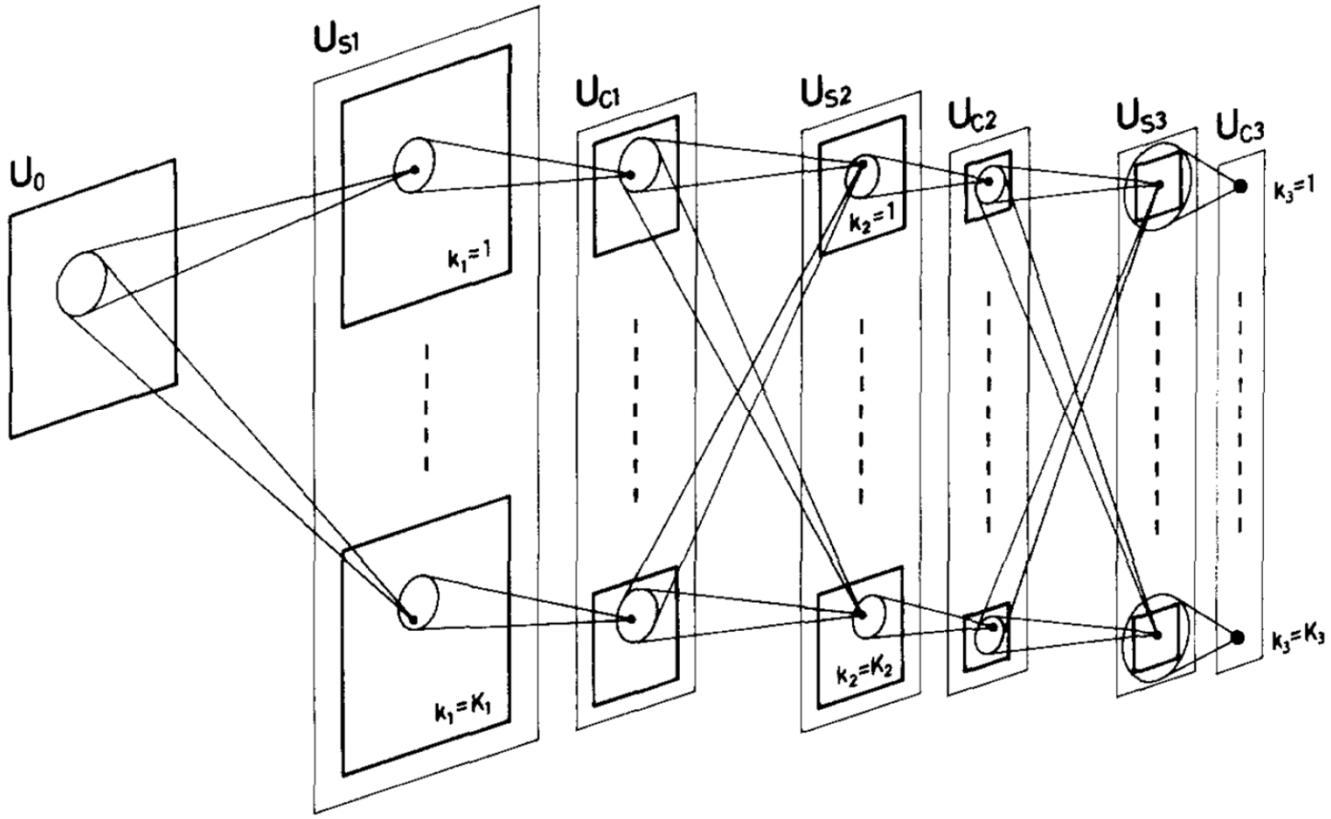
% online learning

for $l = 1, \dots, L$ do $W^l = W^l - \varepsilon \frac{\partial J}{\partial W^l}, \ \boldsymbol{b}^l = \boldsymbol{b}^l - \varepsilon \frac{\partial J}{\partial \boldsymbol{b}^l}$ end for

end for

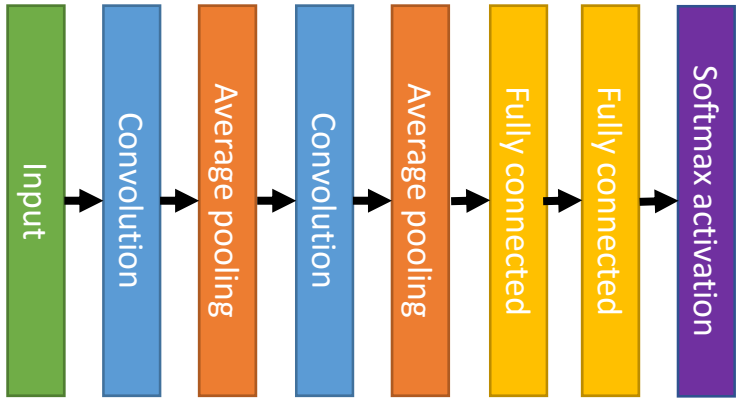
end for

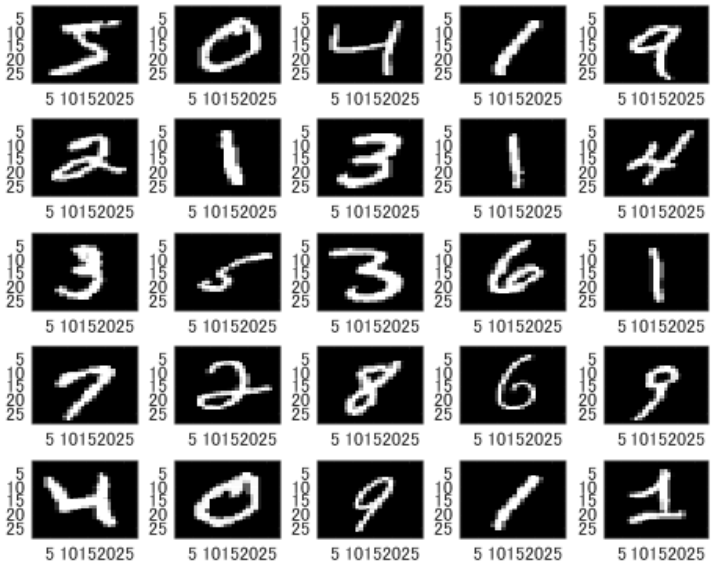
Neocognitron



Kunihiko Fukushima. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biol. Cybernetics 36, 193 202 (1980)

Convolutional Neural Networks





(a) LeNet-5, 1989

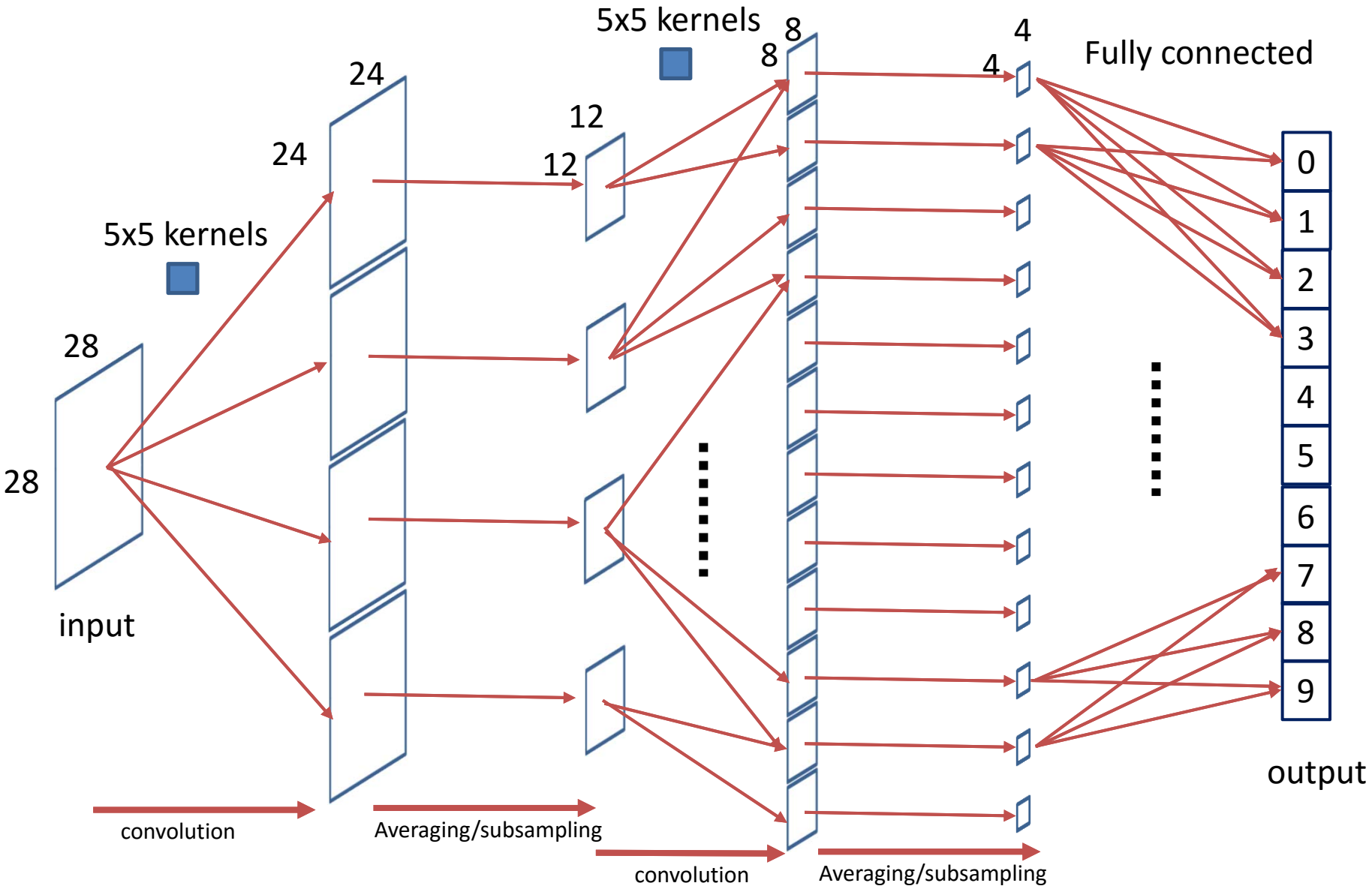
Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. NIPS 1989.

Modules of Convolutional Neural Networks

- Convolution layer
- fully-connected layer
- Pooling layer

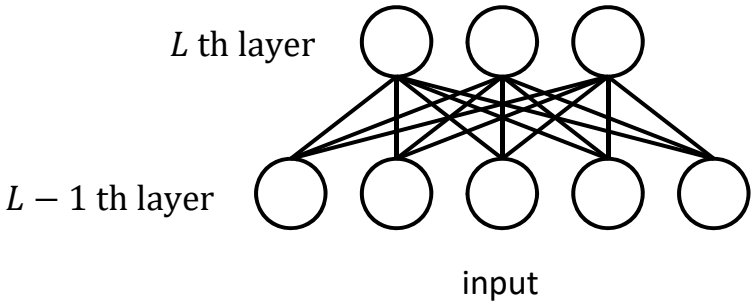
Example of CNNs Architecture

– Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In David Touretzky, editor, Advances in Neural Information Processing Systems 2 (NIPS*89), Denver, CO, 1990. Morgan Kaufman.

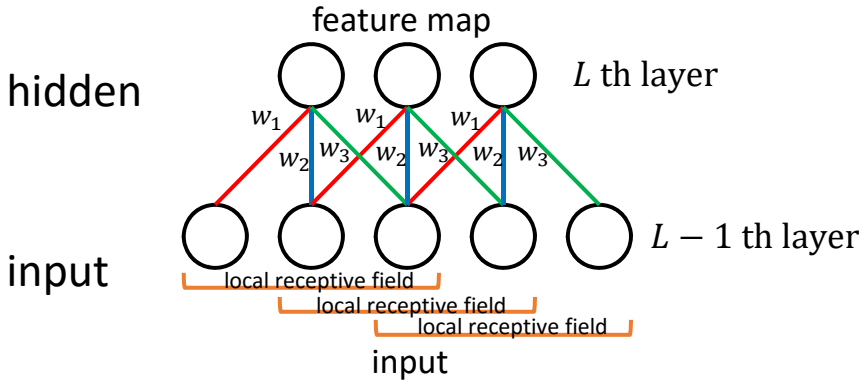


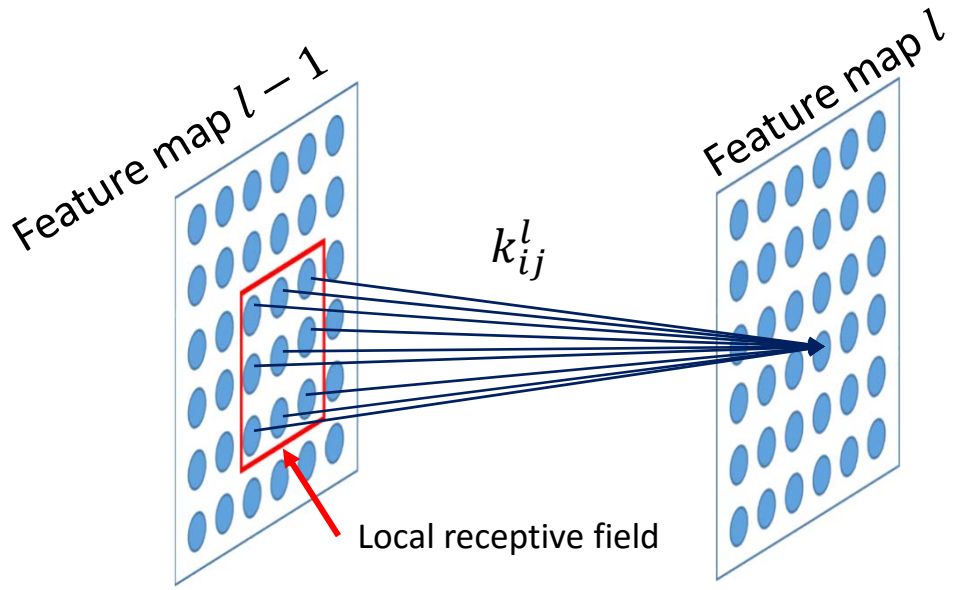
Convolutional and Fully-Connected Layer

Fully-Connected Layer



Convolutional Layer





$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j\right)$$

Logistic sigmoid function

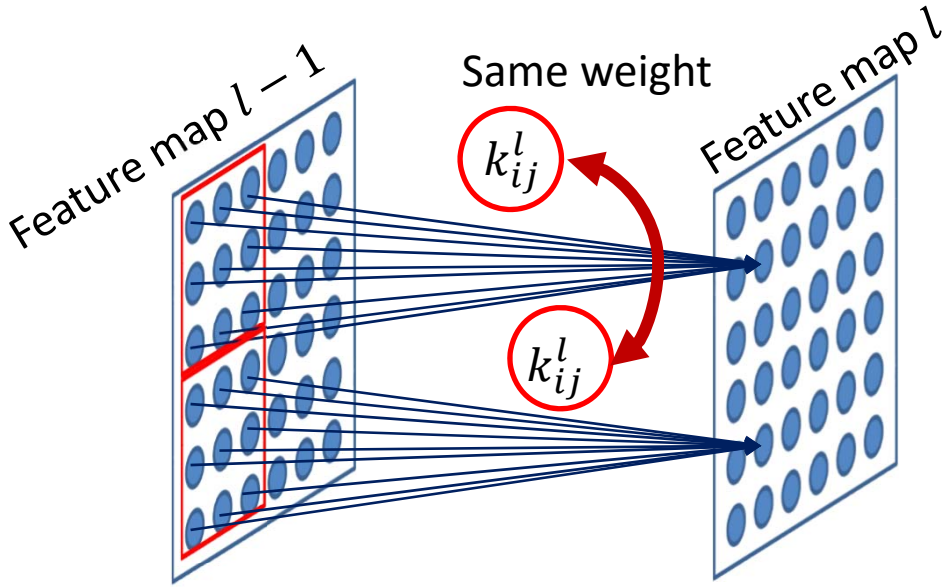
$$f(x) = (1 + e^{-\beta x})^{-1}$$

Hyperbolic tangent function

$$f(x) = a\tanh(bx)$$

Weight Sharing

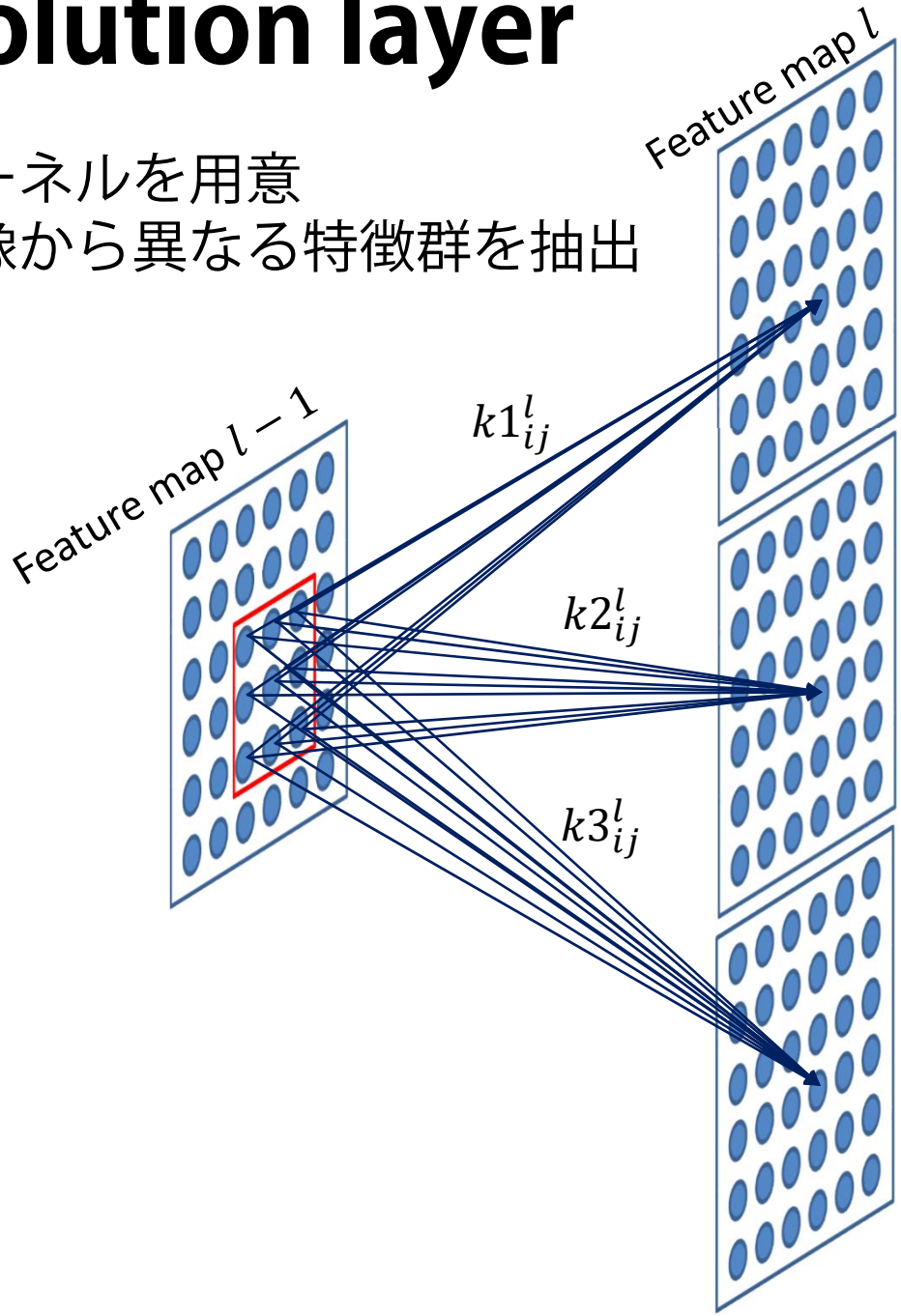
- 画像の一部で有効な特徴抽出であれば，画像の他の部分でも有効な特徴と考える．



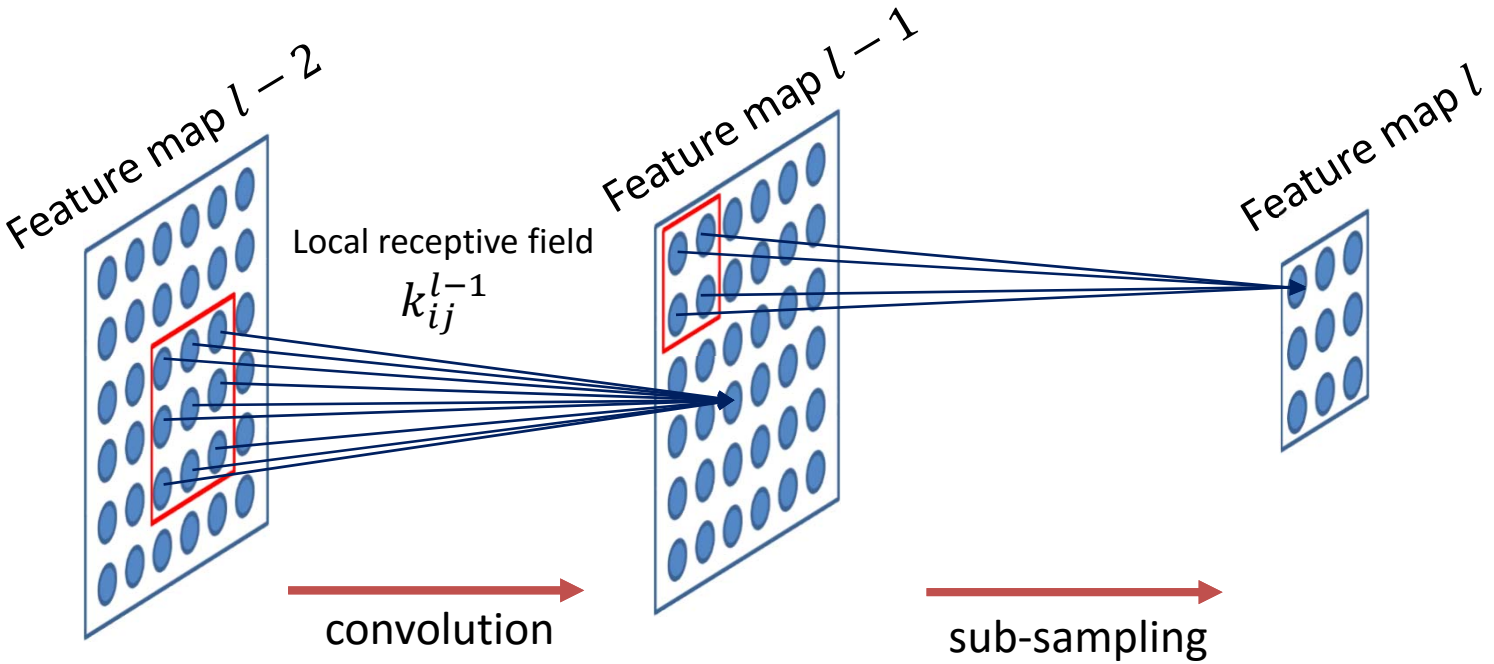
パラメータ数を大幅に減らすことができる．

Convolution layer

- 複数のカーネルを用意
- 同一の画像から異なる特徴群を抽出



Pooling Layer



subsampling function

$$x_j^l = f\left(\beta_j^l \mathrm{down}\left(x_i^{l-1}\right) + b_j^l\right)$$

Average pooling
Max pooling

畳み込み層の順向き伝播

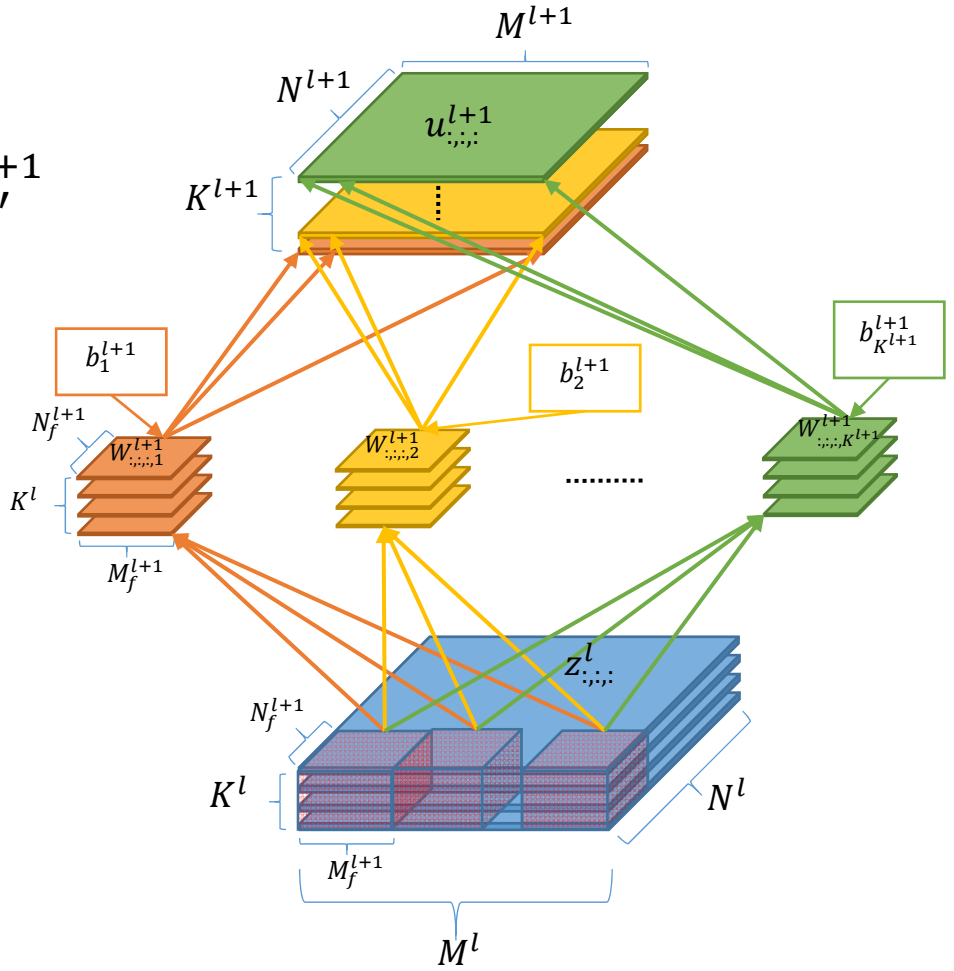
- 第l層の出力：$M^l \times N^l \times K^l$,
- フィルタ数：K^{l+1}，フィルタサイズ：$M_f^{l+1} \times N_f^{l+1} \times K^l$

ユニットへの入力

$$u_{i,j,k'}^{l+1} = \sum_{m,n,k} W_{m,n,k,k'}^{l+1} z_{i+m,j+n,k}^{l} + b_{k'}^{l+1}$$

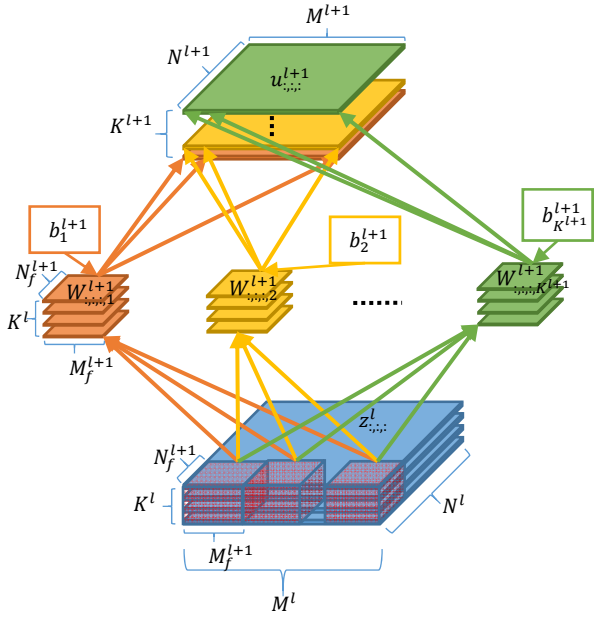
ユニットの出力

$$z_{i,j,k'}^{l+1} = h(u_{i,j,k'}^{l+1})$$



CNNの学習（畳み込み層）

- $\frac{\partial J}{\partial W^l_{m,n,k,k'}}, \frac{\partial J}{\partial b^l_{k'}}$を求める.



連鎖法則（chain rule）

$$\frac{\partial J}{\partial W^l_{m,n,k,k'}} = \sum_{i,j} \frac{\partial J}{\partial u^l_{i,j,k'}} \frac{\partial u^l_{i,j,k'}}{\partial W^l_{m,n,k,k'}}$$

$$u^l_{i,j,k'} = \sum_{m,n,k} W^l_{m,n,k,k'} z^{l-1}_{i+m,j+n,k} + b^l_{k'}$$

誤差（error）

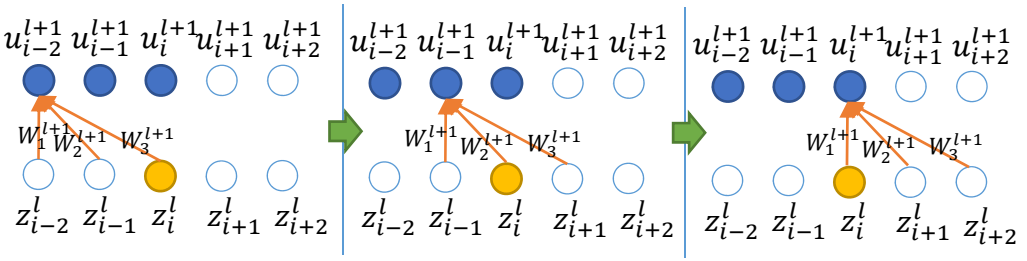
$$\frac{\partial J}{\partial u^l_{i,j,k'}} = \delta^l_{i,j,k'}$$

$$\frac{\partial u^l_{i,j,k'}}{\partial W^l_{m,n,k,k'}} = z^{l-1}_{i+m,j+n,k}$$

$$\frac{\partial J}{\partial W^l_{m,n,k,k'}} = \sum_{i,j} \delta^l_{i,j,k'} \cdot z^{l-1}_{i+m,j+n,k} \quad , \frac{\partial J}{\partial b^l_{k'}} = \sum_{i,j} \delta^l_{i,j,k'}$$

CNNの学習（畳み込み層）

- 誤差$\delta_{i,j,k}^{l}$を求める.



$$\frac{\partial J}{\partial W_{m,n,k,k'}^{l}} = \sum_{i,j} \delta_{i,j,k'}^{l} \cdot z_{i+m,j+n,k}^{l-1}$$

順向き伝播で計算しておく

連鎖法則（chain rule）

$$u_{i-m,j-n,k'}^{l+1} = \sum_{p,q,k} W_{p,q,k,k'}^{l+1} z_{i-m+p,j-n+q,k}^{l} + b_{k'}^{l+1}$$

誤差（error）$\delta_{i,j,k}^{l} = \frac{\partial J}{\partial u_{i,j,k}^{l}} = \sum_{m,n,k'} \frac{\partial J}{\partial u_{i-m,j-n,k'}^{l+1}} \frac{\partial u_{i-m,j-n,k'}^{l+1}}{\partial u_{i,j,k}^{l}}$

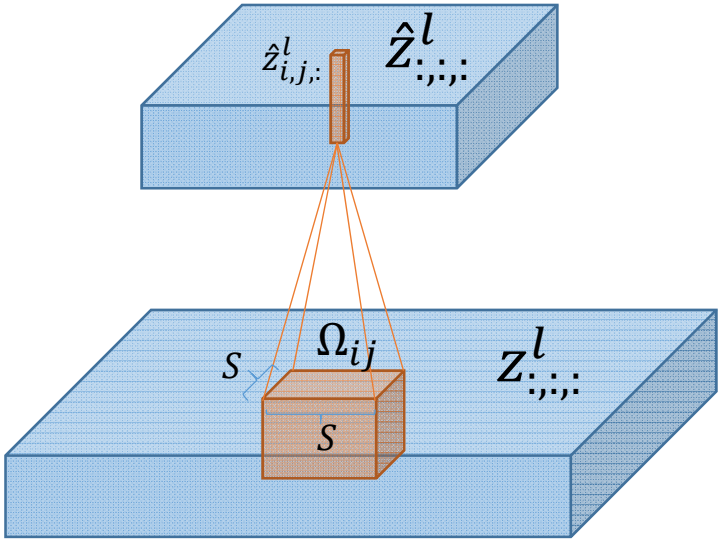
上位層の誤差 $\delta_{i-m,j-n,k'}^{l+1}$

$$\frac{\partial u_{i-m,j-n,k'}^{l+1}}{\partial u_{i,j,k}^{l}} = W_{m,n,k,k'}^{l+1} h'(u_{i,j,k}^{l})$$

$$\delta_{i,j,k}^{l} = h'(u_{i,j,k}^{l}) \sum_{m,n,k'} \delta_{i-m,j-n,k'}^{l+1} \cdot W_{m,n,k,k'}^{l+1}$$

現在の層の誤差は一つ上の層の誤差から計算可能

プーリング層の順向き伝播，逆向き伝播



順向き伝播（forward propagation）

平均値プーリング（average pooling）

$$\hat{z}^l_{i,j,k} = \frac{1}{S^2} \sum_{(m,n)\in\Omega_{ij}} z^l_{m,n,k}$$

最大値プーリング（max pooling）

$$\hat{z}^l_{i,j,k} = \max\{z^l_{m,n,k} : (m,n) \in \Omega_{ij}\}$$

逆向き伝播（backward propagation）

平均値プーリング（average pooling）

誤差をΩ_{ij}内のユニットに均等に配分

最大値プーリング（max pooling）

Ω_{ij}において最大値を観測したユニットに誤差を全て配分．その他のユニットは0

最適化

- 確率的勾配降下法

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \rho^{(t)} \nabla E_n(\boldsymbol{w}^{(t)})$$

- モーメンタム法
  - 例えば，目的関数の形が長くて狭い谷であり，緩やかに谷の長軸方向に下っている場合，最急降下法では谷の短軸方向に振動してしまい，目的関数を減少させる谷の長軸方向に下ることが難しくなる．
  - モーメンタム法であれば，振動成分を平滑化し，緩やかな谷の長軸方向に下るようにパラメータを更新させることが可能となる．

$$\begin{aligned} \Delta \boldsymbol{w}^{(t+1)} &= \mu \Delta \boldsymbol{w}^{(t)} - \rho^{(t)} \nabla E_n(\boldsymbol{w}^{(t)}) \\ \boldsymbol{w}^{(t+1)} &= \boldsymbol{w}^{(t)} + \Delta \boldsymbol{w}^{(t+1)} \end{aligned}$$

- Nesterov加速勾配法
  - モーメンタム法と似ているが，更新後のパラメータ近くの勾配を利用して，パラメータを更新している点が異なる．

$$\begin{aligned} \Delta \boldsymbol{w}^{(t+1)} &= \mu \Delta \boldsymbol{w}^{(t)} - \rho^{(t)} \nabla E_n(\boldsymbol{w}^{(t)} + \gamma \Delta \boldsymbol{w}^{(t)}) \\ \boldsymbol{w}^{(t+1)} &= \boldsymbol{w}^{(t)} + \Delta \boldsymbol{w}^{(t+1)} \end{aligned}$$

最適化

- AdaGrad
  - 適応的に学習係数を決定する手法
  - 更新回数に応じて学習係数が小さくなる．大きな勾配を用いたパラメータ更新が続けば学習係数が速く小さく，逆に小さな勾配が続けば現状の学習係数を維持してパラメータを更新する．

$$v^{(t+1)} = v^{(t)} + \nabla E_n(\boldsymbol{w}^{(t)})^2$$

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \frac{\rho^{(t)}}{\sqrt{v^{(t+1)}} + \epsilon} \nabla E_n(\boldsymbol{w}^{(t)})$$

- RMSProp
  - 勾配の小さな箇所にとどまる場合，学習係数が大きくなり勾配方向を増幅してパラメータを更新する．
  - この仕組みにより，プラトーや鞍点から抜け出しやすくなる．

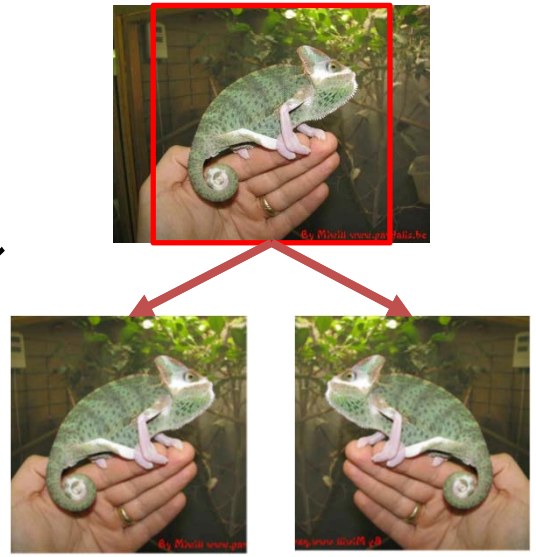
$$v^{(t+1)} = \beta v^{(t)} + (1-\beta)\nabla E_n(\boldsymbol{w}^{(t)})^2$$

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \frac{\rho^{(t)}}{\sqrt{v^{(t+1)}} + \epsilon} \nabla E_n(\boldsymbol{w}^{(t)})$$

Reducing Overfitting

- Dropout
  - G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.
  - テストセットのエラーを低減する良い手法として，多くの異なるネットワークの予測の平均を用いる
  - トレーニングデータが提示される毎に，隠れ層のニューロンの出力を1/2の確率で0とする．
    - トレーニングデータ毎に異なるネットワークアーキテクチャが選択される．ただしネットワークの重みは共有している．
  - N個の隠れ層ユニットを持つとすると，2^N個のネットワークによって予測されるラベルの確率の平均を用いてクラスラベルを予測していることになる．

Data Augmentation



- トレーニングデータを2048倍に水増し
  - データを平行移動して拡張
  - 水平の鏡像画像を用いて拡張
  - Deep CNNsのoverfittingを回避する

- RGBチャンネルの輝度を変化させデータを水増し
  - 照明変化に不変にする
  - Top1 errorで1%の性能向上
  - RGBのピクセル値にPCAをかける
  - 各画像のRGBピクセルに以下の値を加える.

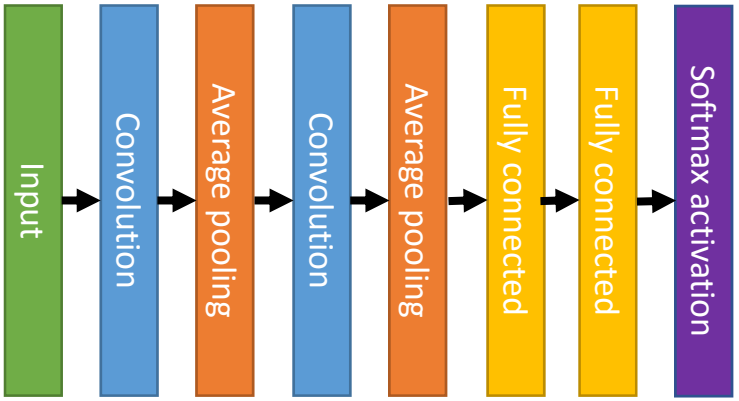
$$\begin{bmatrix} I_{xy}^R \\ I_{xy}^G \\ I_{xy}^B \end{bmatrix} = \begin{bmatrix} I_{xy}^R \\ I_{xy}^G \\ I_{xy}^B \end{bmatrix} + \underbrace{[\boldsymbol{p}_1 \quad \boldsymbol{p}_2 \quad \boldsymbol{p}_3]}_{\text{Eigenvectors}} \underbrace{\begin{bmatrix} \alpha_1\lambda_1 \\ \alpha_2\lambda_2 \\ \alpha_3\lambda_3 \end{bmatrix}}_{\text{Eigenvalues}}$$

Batch normalization

- データのスケールが変化することで学習が遅くなる.
- 白色化は毎回固有値問題を解く必要があり計算コストが高くなる欠点を持つ.
- ミニバッチで与えられたデータにおいて，入力データの全ての要素を白色化するのではなく，各要素毎に平均0，分散1に正規化をする.

input a mini-batch: $\mathcal{B} = \{\boldsymbol{x}_i\}_{i=1}^m$, parameters: (γ, β)
output $\{\boldsymbol{y}_i = \mathrm{BN}(\boldsymbol{x}_i)\}_{i=1}^m$
mini-batch mean $\mu_B^{(k)} \leftarrow \frac{1}{m}\sum_{i=1}^m x_i^{(k)}$
mini-batch variance $(\sigma_B^{(k)})^2 \leftarrow \frac{1}{m}\sum_{i=1}^m (x_i^{(k)} - \mu^{(k)})^2$
normalize $\hat{x}_i^{(k)} \leftarrow \frac{1}{\sqrt{(\sigma_B^{(k)})^2 + \epsilon}}(x_i^{(k)} - \mu_B^{(k)})$
scale and shift $y_i^{(k)} \leftarrow \gamma^{(k)}\hat{x}_i^{(k)} + \beta^{(k)}$

LeNet and Alex Net

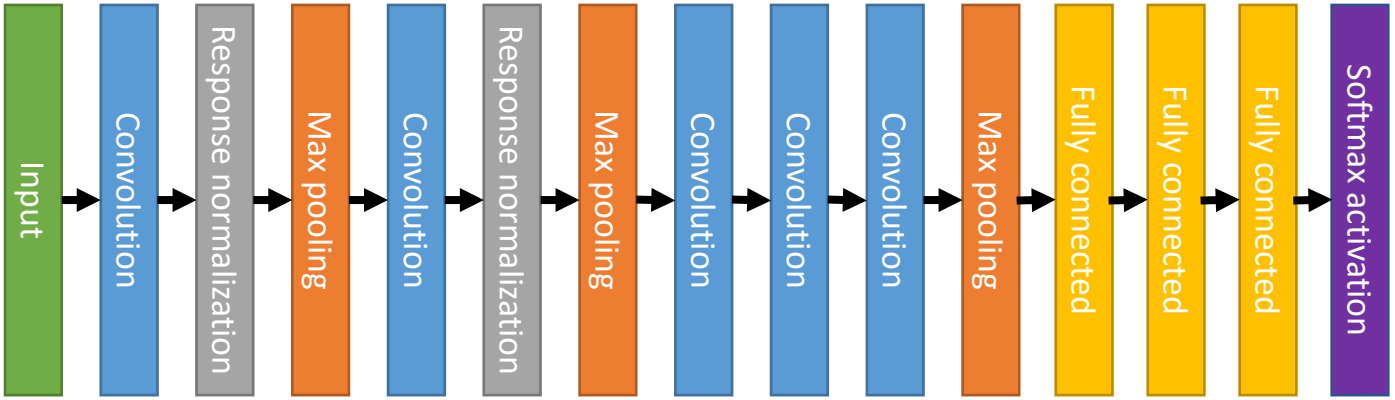


(a) LeNet-5, 1989

Features of Alex Net

- 5 conv. layers + 3 fully-connected layers
- ReLU nonlinearity
- Multiple GPGPUs
- Local response normalization
- Overlappling pooling
- Dropout
- Data augementation

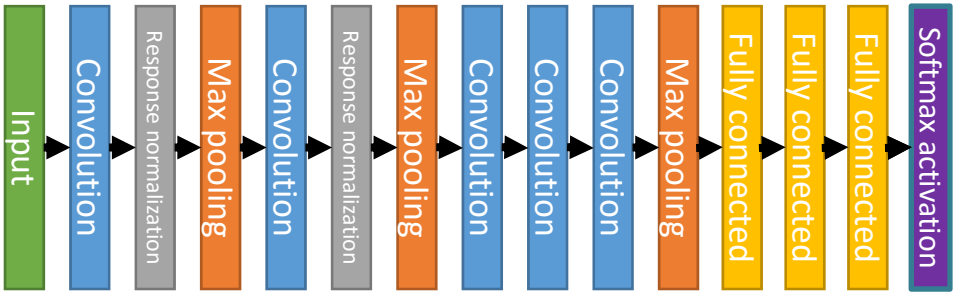
Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. NIPS 1989.



(b) AlexNet, 2012

Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. NIPS, 2012.

VGG Net

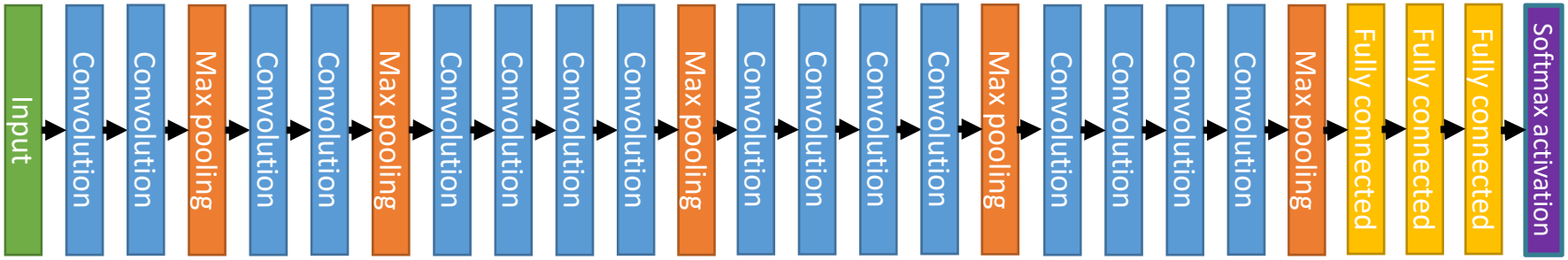


(b) AlexNet, 2012

Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. NIPS, 2012.

Features of VGG Net

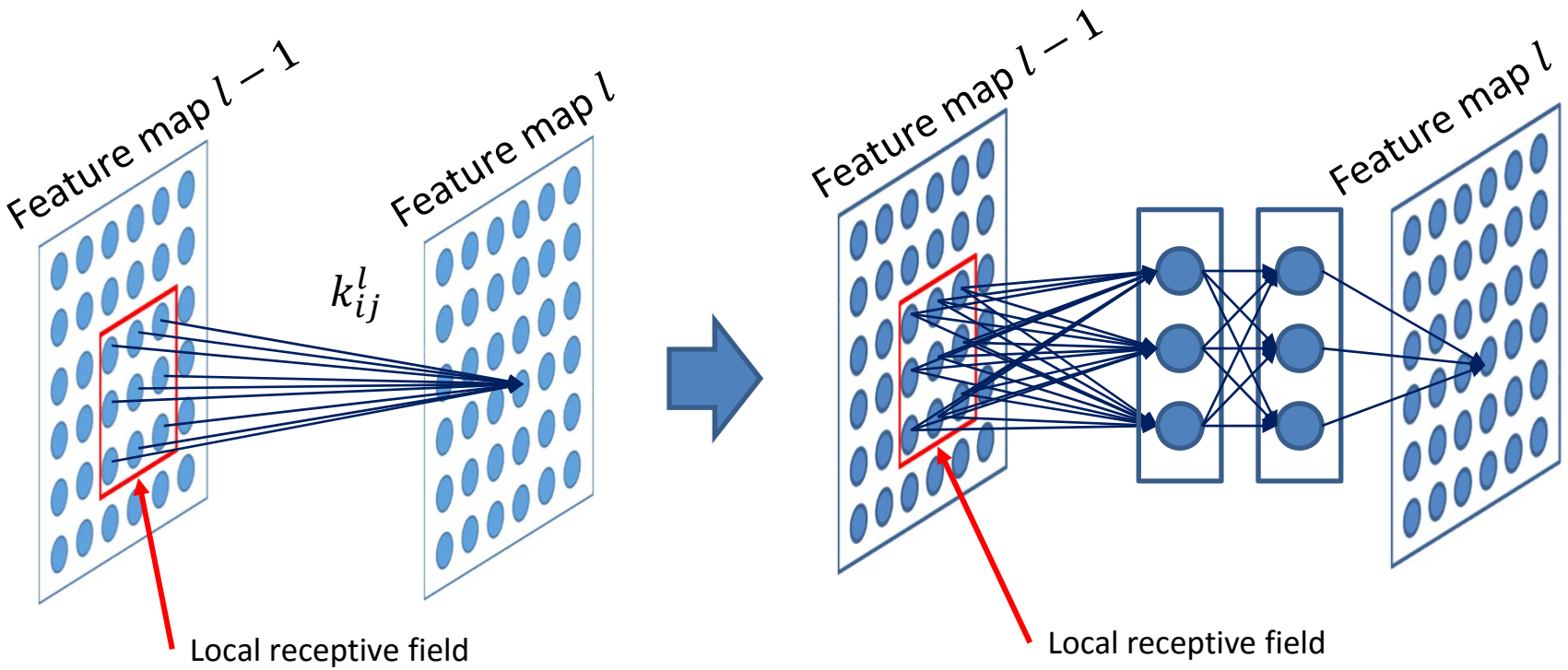
- 16 conv. layers + 3 fully-connected layers
- Very small receptive fields
- No Local response normalization
- Data parallelism



(c) VGG Net, 2014

Karen Simonyan, Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556, 2014.

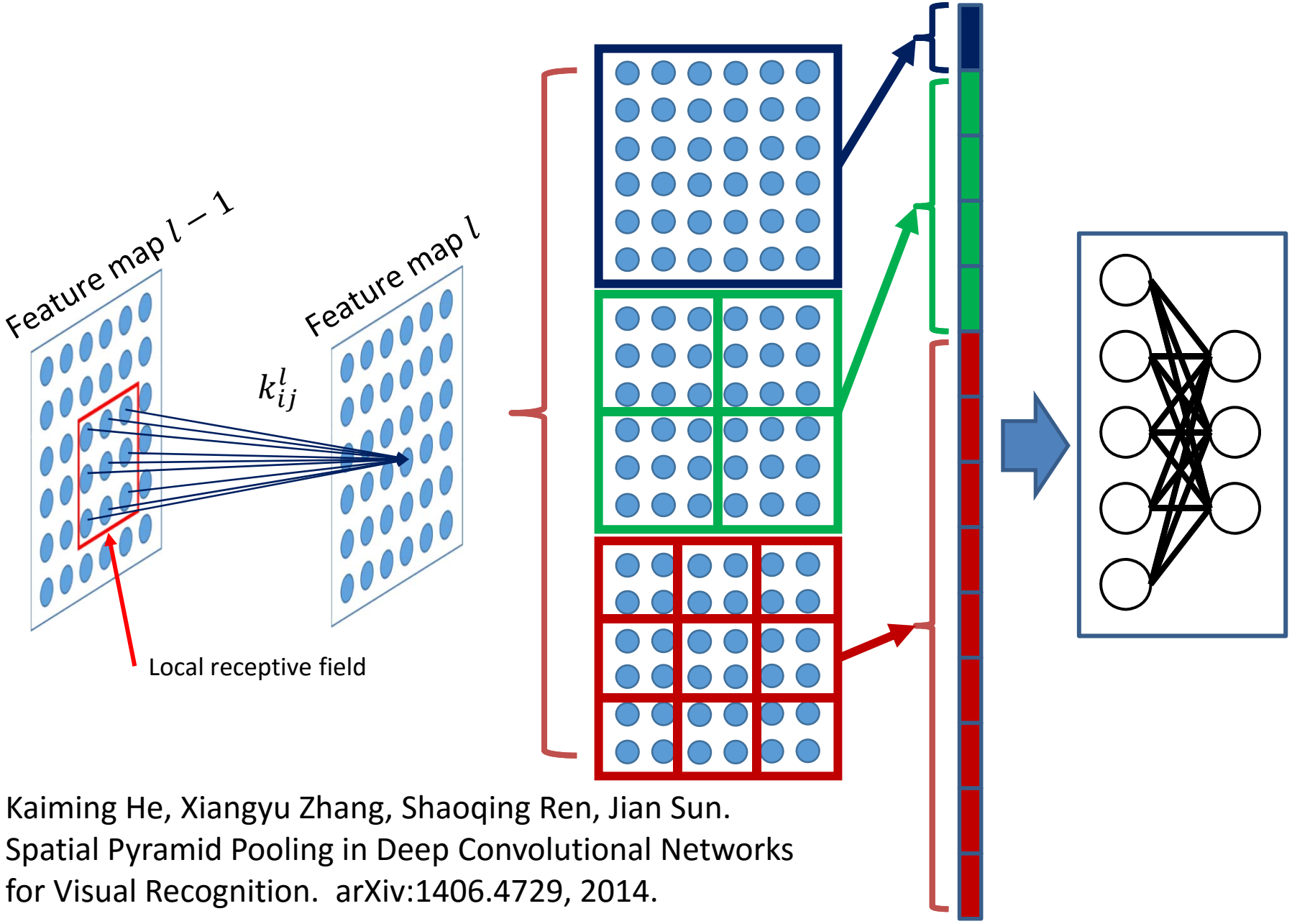
Network in Network



$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j\right)$$

Min Lin, Qiang Chen, Shuicheng Yan. Network In Network. arXiv:1312.4400, 2014.

Spatial Pyramid Pooling Layer



Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. arXiv:1406.4729, 2014.

SPP layer

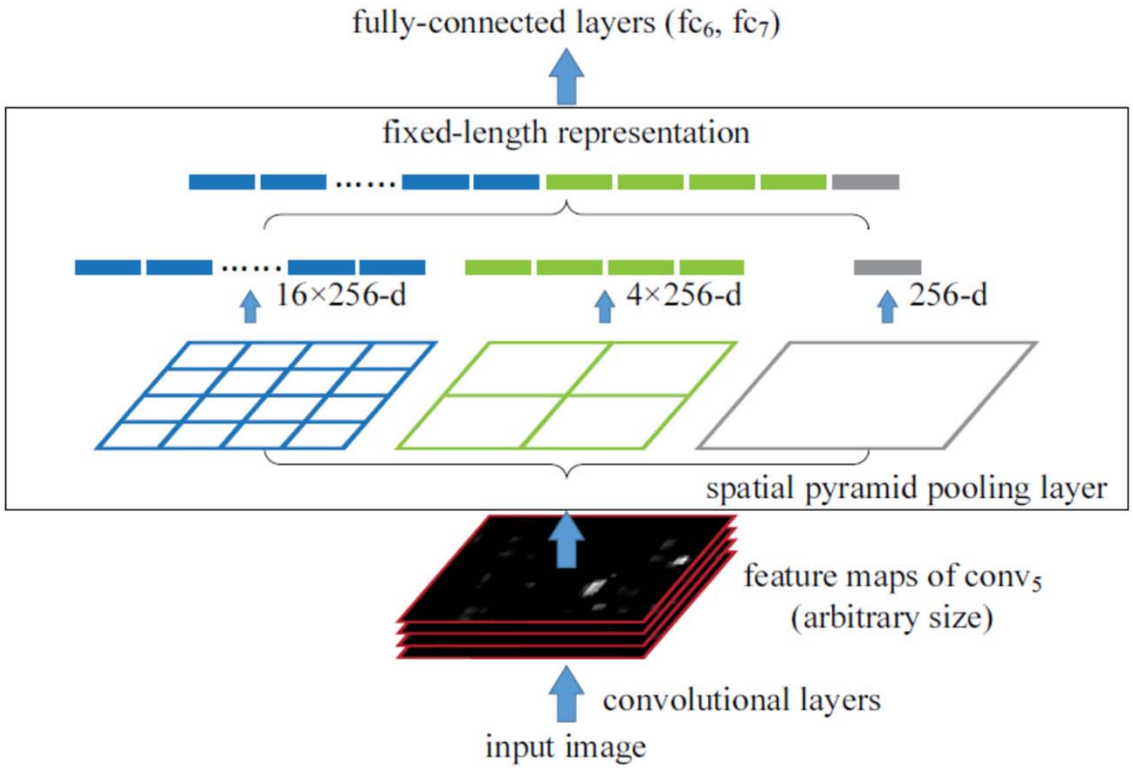
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. arXiv:1406.4729, 2014.

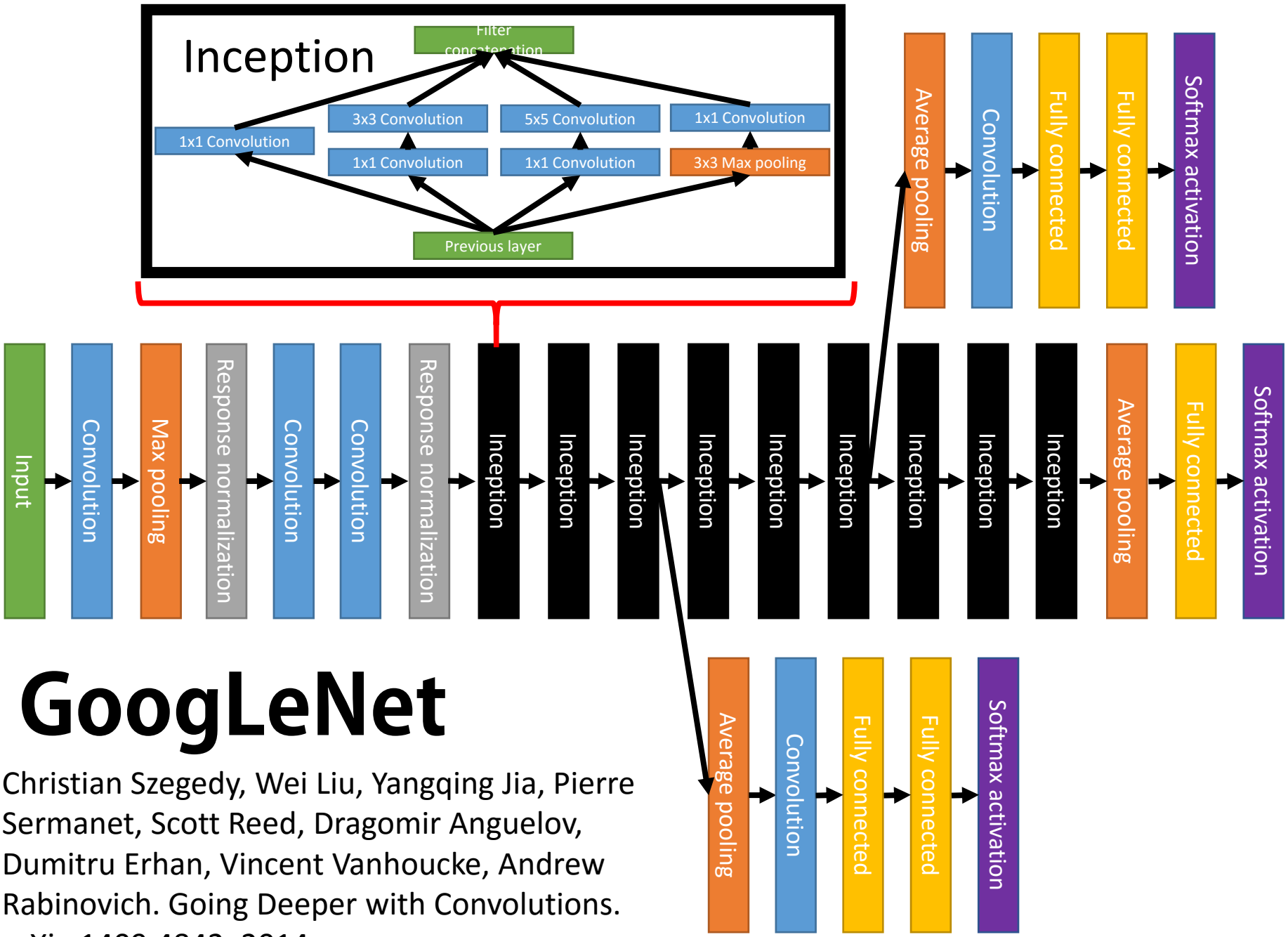
- 従来のCNNは入力画像をcropもしくはwarpして，一定の大きさの画像に変換してネットワークに入力する.
- Cropとwarpにより識別性能の低下が生じる可能性がある.
- 畳み込み層は任意のサイズの画像を扱うことができるが，全結合層は一定の次元数のみ扱うことができる.
- 畳み込み層と全結合層の間に，任意のサイズの入力を一定の次元の特徴に変換するSpatial Pyramid Pooling Layerを設ける.

SPP layer

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. arXiv:1406.4729, 2014.

- 畳み込み層と全結合層の間のプーリング層を Spatial Pyramid Pooling 層に置き換える.
- ビン数をM，最終畳み込み層のフィルタ数をkとするとSPP層によりkM次元の特徴量が得られる.



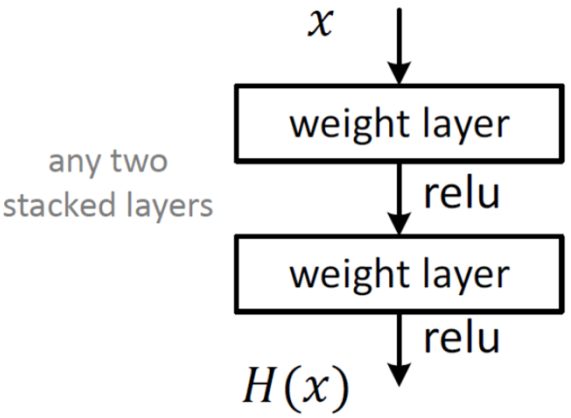


GoogLeNet

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. Going Deeper with Convolutions. arXiv:1409.4842, 2014.

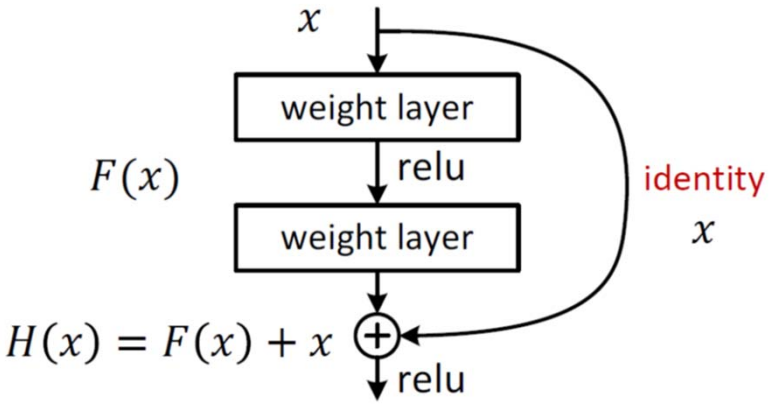
Residual Network

- Plaint net

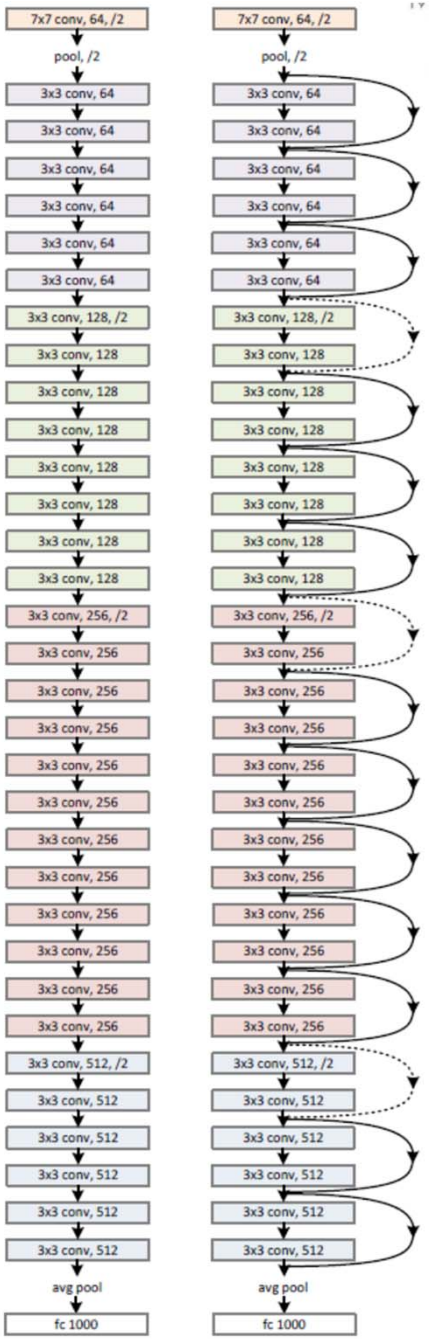


any two stacked layers

- Residual net



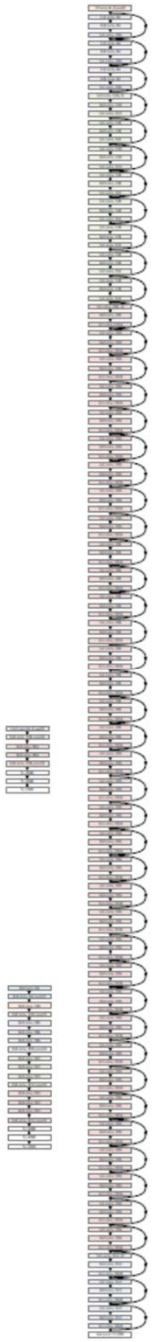
identity x



ResNet, 152 layers
(ILSVRC 2015)

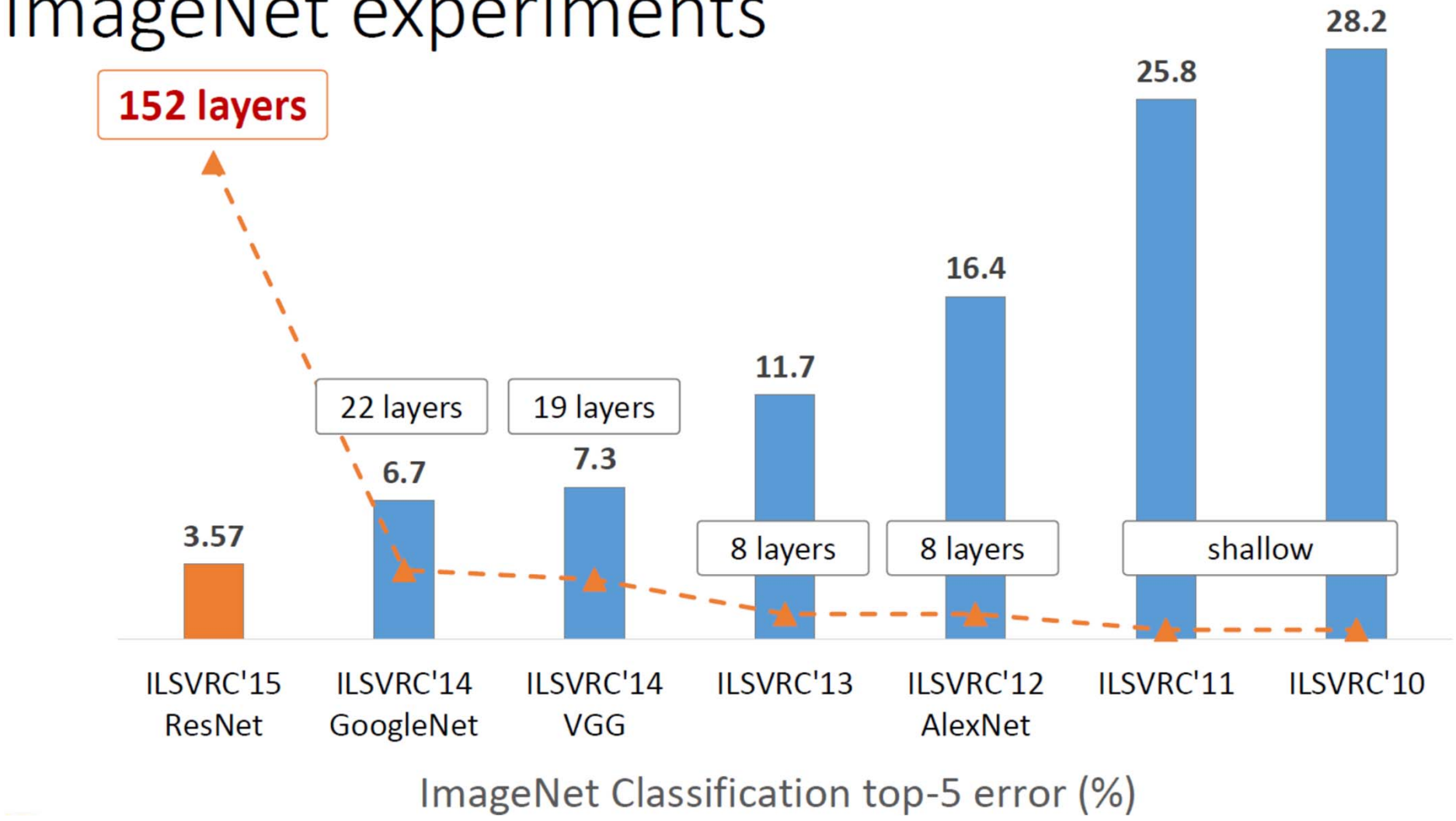
AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)



http://image-net.org/challenges/talks/ilsvrc2015_deep_residual_learning_kaiminghe.pdf

ImageNet experiments



http://image-net.org/challenges/talks/ilsvrc2015_deep_residual_learning_kaiminghe.pdf

Deep Leaning Frameworks

- TensorFlow
  - https://www.tensorflow.org/
- Theano
  - http://deeplearning.net/software/theano/
- Torch
  - http://torch.ch/
- Caffe
  - http://caffe.berkeleyvision.org/
- CNTK
  - https://www.cntk.ai/
- Chainer
  - http://chainer.org/

FireCaffe

- Forrest N. Iandola, Khalid Ashraf, Matthew W. Moskewicz, Kurt Keutzer. FireCaffe: near-linear acceleration of deep neural network training on compute clusters. arXiv:1511.00175v2

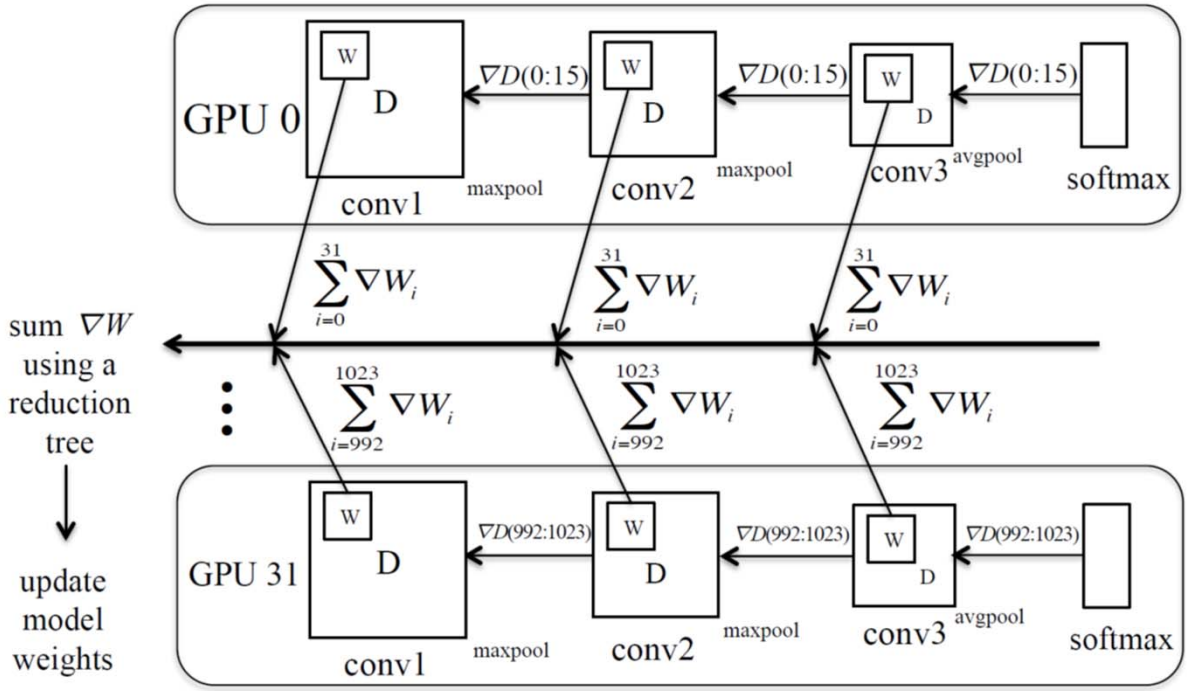


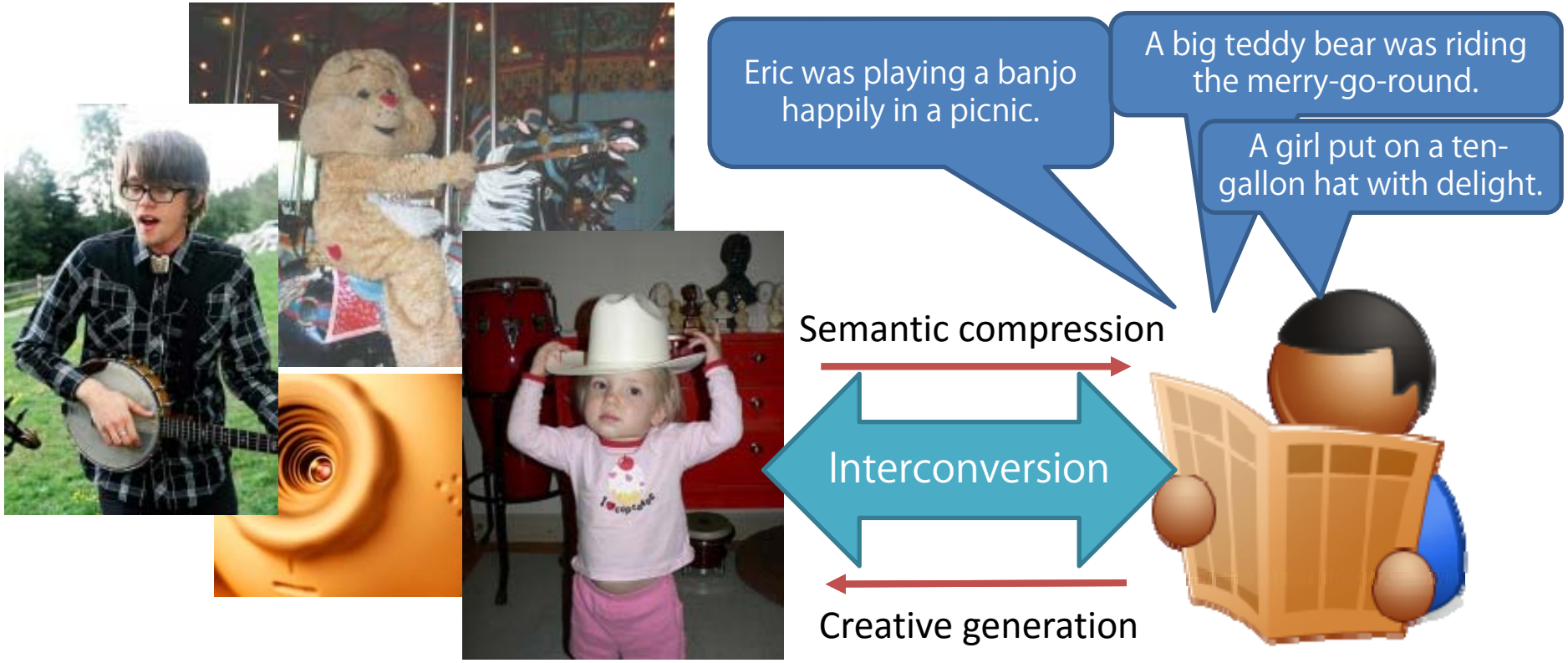
Figure 1. **Data parallel DNN training in FireCaffe:** Each worker (GPU) gets a subset of each batch.

Table 3. Accelerating the training of ultra-deep, computationally intensive models on ImageNet-1k.

	Hardware	Net	Epochs	Batch size	Initial Learning Rate	Train time	Speedup	Top-1 Accuracy	Top-5 Accuracy
Caffe	1 NVIDIA K20	GoogLeNet [41]	64	32	0.01	21 days	1x	68.3%	88.7%
FireCaffe (ours)	32 NVIDIA K20s (Titan supercomputer)	GoogLeNet	72	1024	0.08	23.4 hours	20x	68.3%	88.7%
FireCaffe (ours)	128 NVIDIA K20s (Titan supercomputer)	GoogLeNet	72	1024	0.08	10.5 hours	**47x**	68.3%	88.7%

Automatic Contents Generation

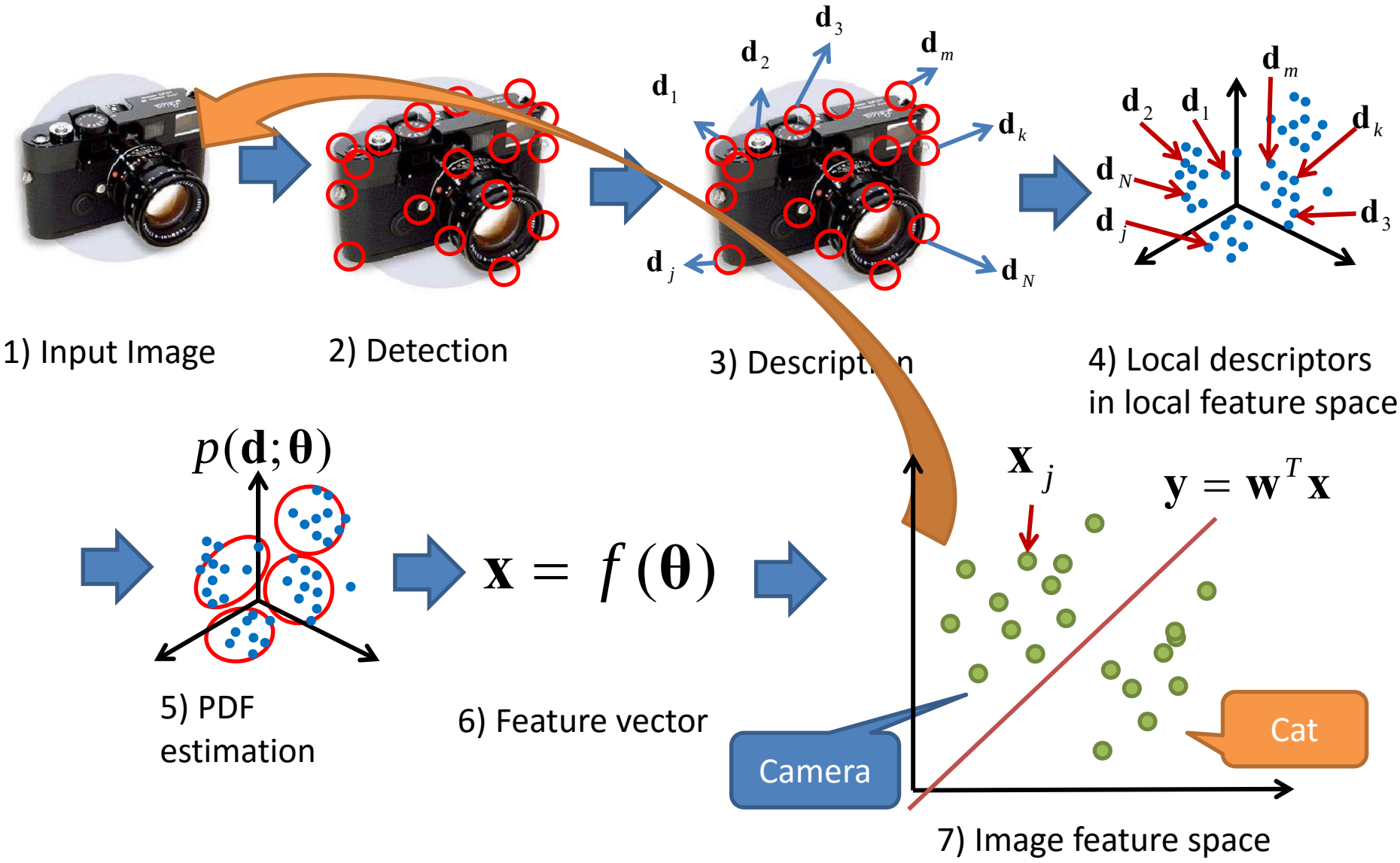
- Semantic compression
  - Conversion of images to labels and natural language
- Creative generation
  - Imagination of scenes and characters corresponds to stories



Building the creative intelligence in the machine is challenging!

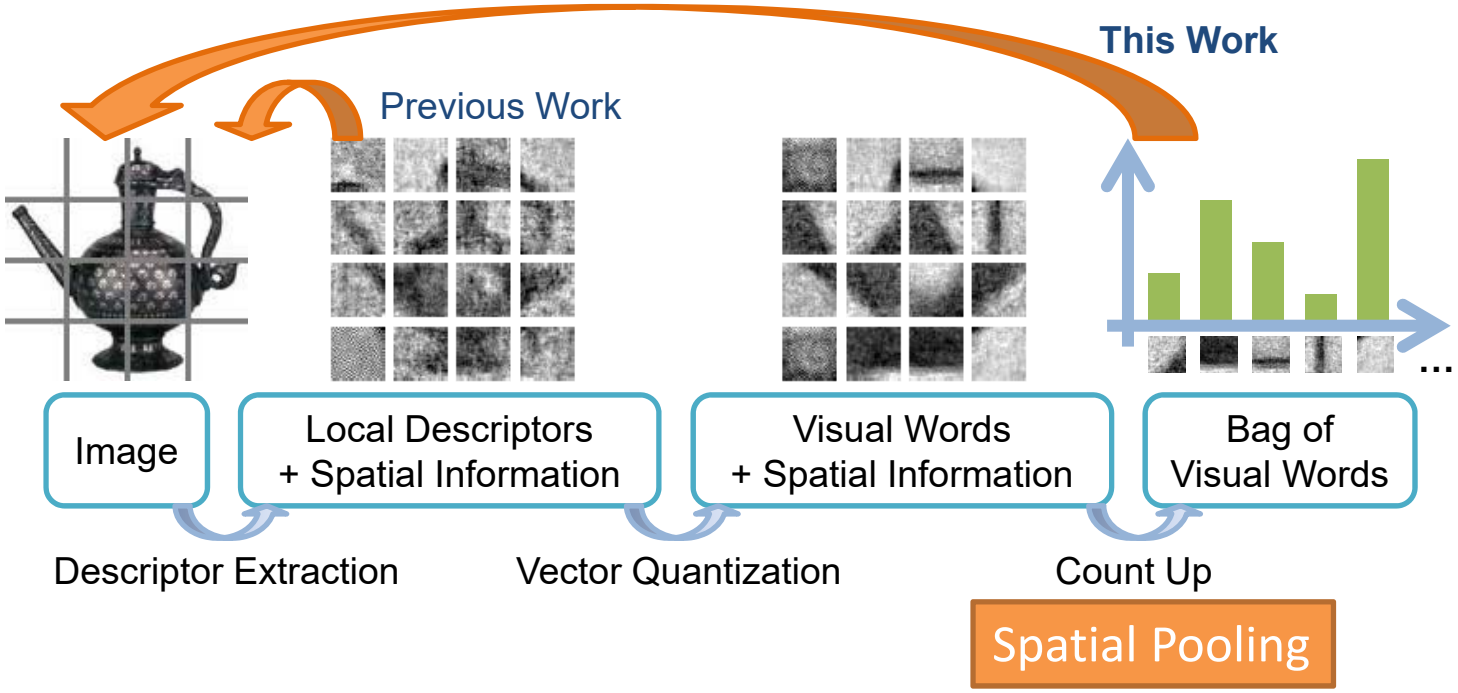
Image Feature and Recognition

How to generate a new image from the semantic information?



The image feature space is most closely related to the semantic information.

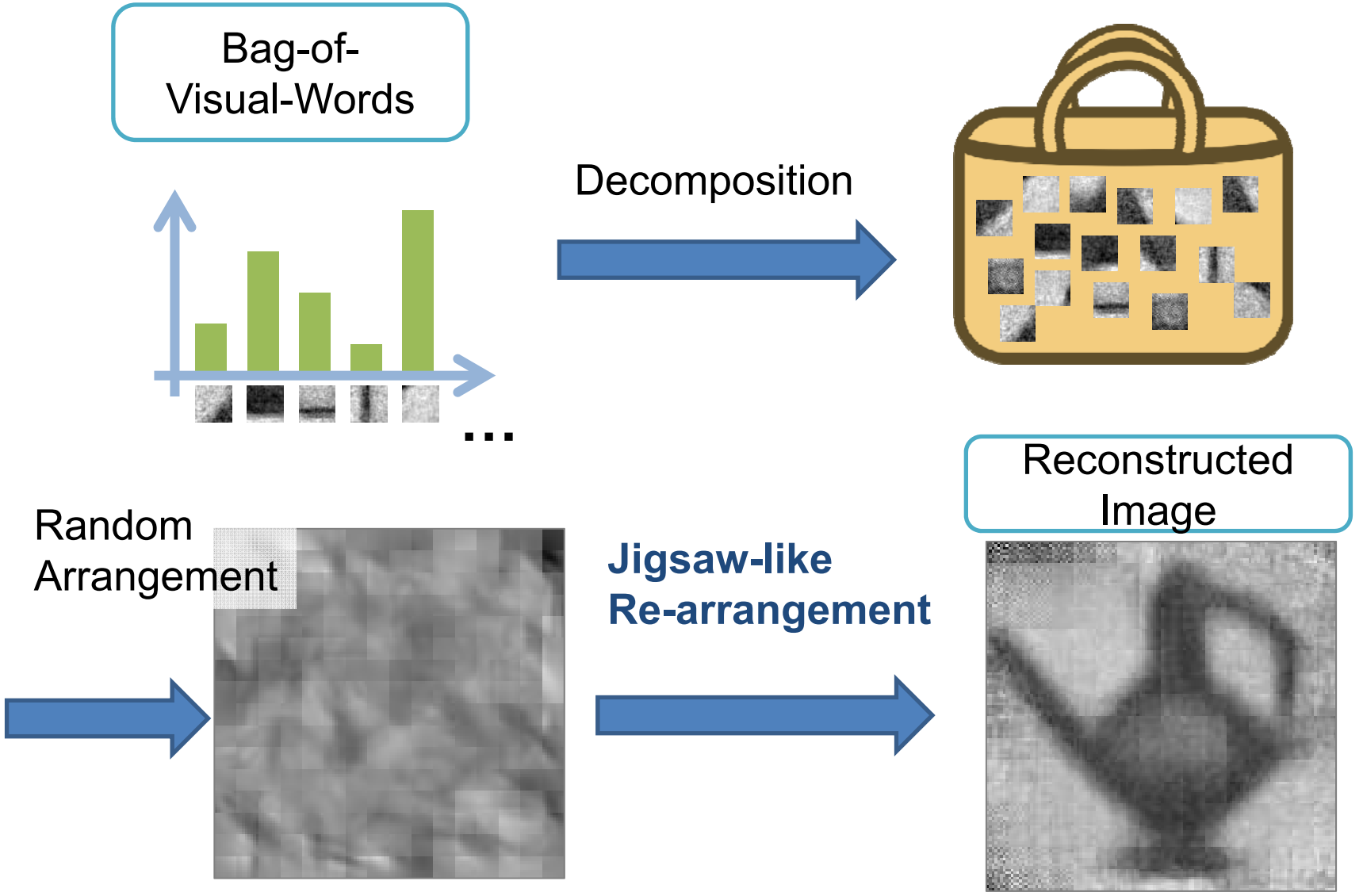
Bag-of-Visual-Words (BoVW)



- BoVW is the de-facto standard image features
- Problem
  - Estimation of spatial arrangement of visual words
  - Generation of an image patch from each visual word

Overview of Image Reconstruction from BoVW

H. Kato, T. Harada. Image Reconstruction from Bag-of-Visual-Words. CVPR, 2014.



Reconstruction Method

H. Kato, T. Harada. Image Reconstruction from Bag-of-Visual-Words. CVPR, 2014.

Estimation of spatial arrangement of visual words

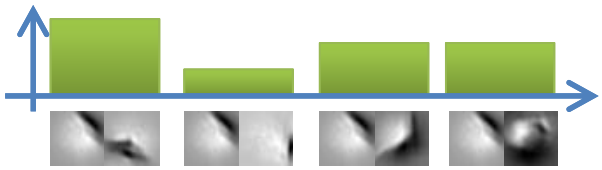
$$\min \quad \lambda \sum_{i,j,k,l=1}^{n} C^{a}_{ijkl} x_{ik} x_{jl} + (1-\lambda) \sum_{i,k=1}^{n} C^{l}_{ik} x_{ik}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ik} = 1, \quad \sum_{k=1}^{n} x_{ik} = 1, \quad x_{i,k} \in \{0,1\}$$

- $x_{ik} = 1$
  if the *i*-th visual word in BoVW is assigned to *k*-th grid point.
- **Adjacency Cost C^a**
  gives a reconstructed image consistent edges and shapes.

Large scale image database

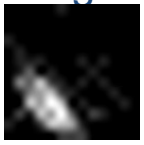
Co-occurrence histogram



- Global Location Cost C^l
  makes a reconstructed image globally feasible.

Similar images by BoVW

Spatial histogram of 

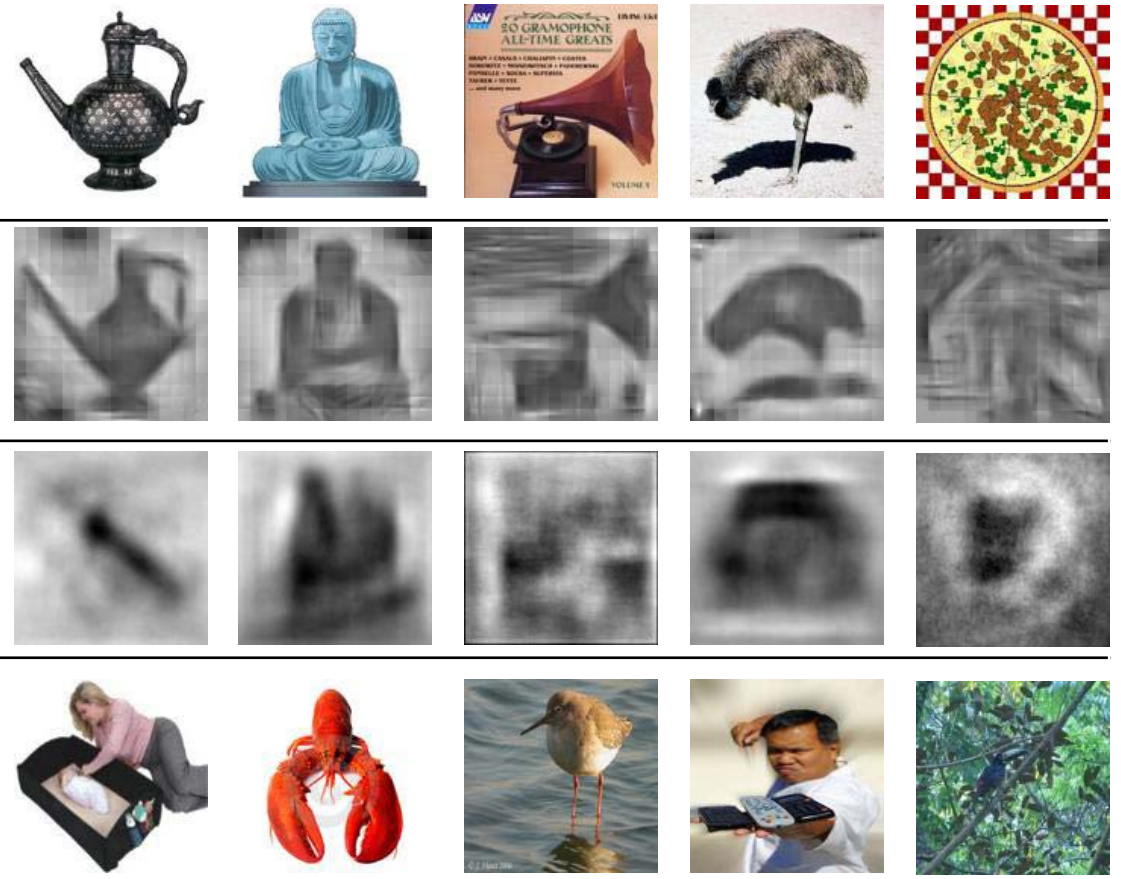


- Optimization
  - This problem can be result in the Quadratic Assignment Problem.
  - Solved by Hybrid algorithm of Genetic Algorithm and Hill Climbing.

Results

H. Kato, T. Harada. Image Reconstruction from Bag-of-Visual-Words. CVPR, 2014.

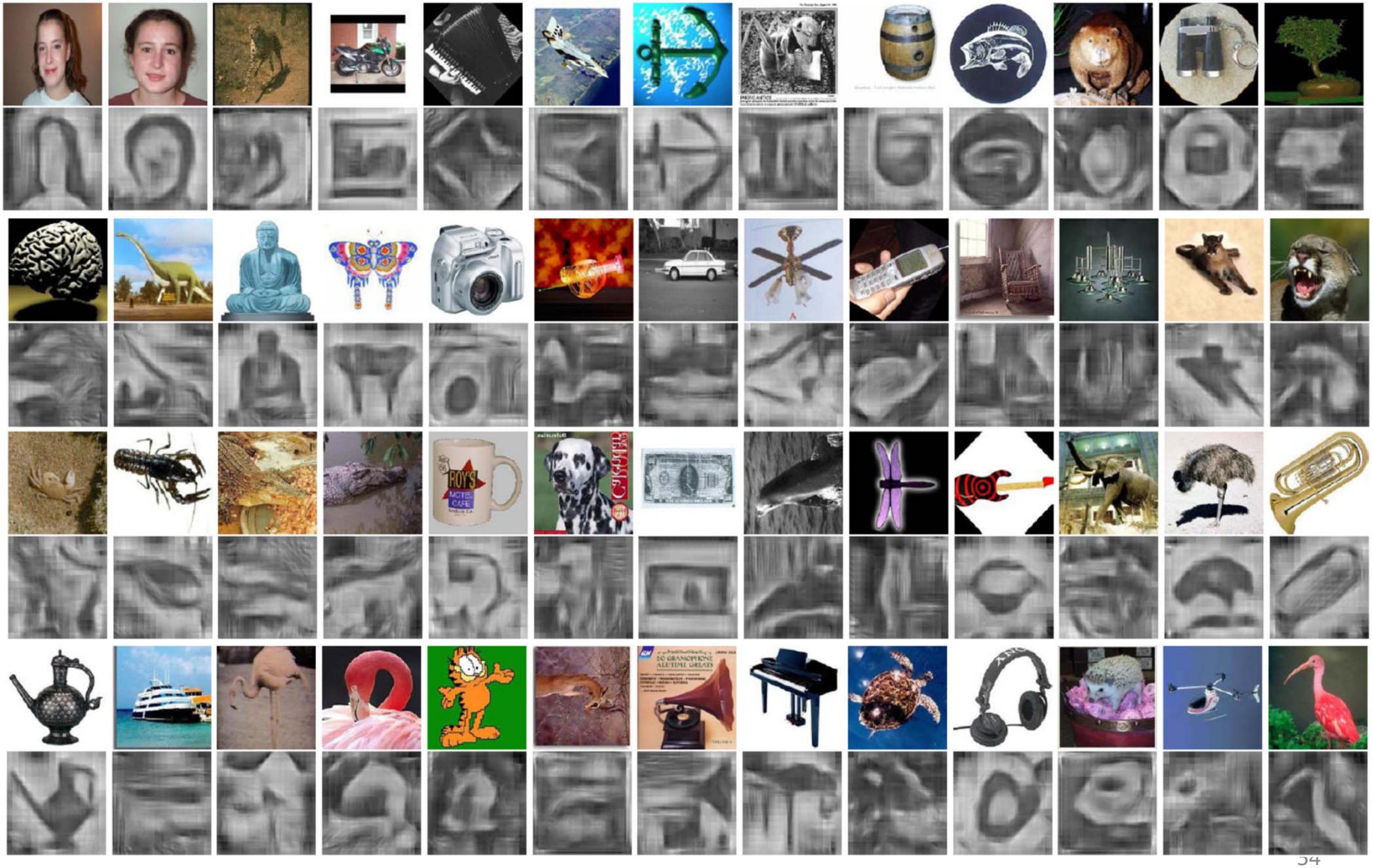
Good Results



*1 Vondrick et al., ICCV, 2013.

Results

H. Kato, T. Harada. Image Reconstruction from Bag-of-Visual-Words. CVPR, 2014.



単語からの画像生成

Hiroharu Kato and Tatsuya Harada. Image Reconstruction from Bag-of-Visual-Words. arXiv:1505.05190v1 [cs.CV] 19 May 2015.

- 識別機の可視化としての画像生成

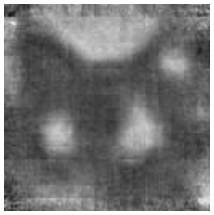
言語的意味：犬，猫，人

$$\mathbf{y} = \mathbf{w}^T \mathbf{x}$$

?

画像特徴

- 生成した画像の例


cat (bombay)

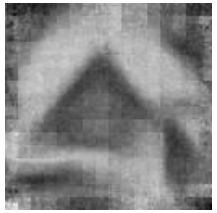

camera


grand piano


gramophone


headphone


joshua tree


pyramid


wheel chair

文章からの画像生成

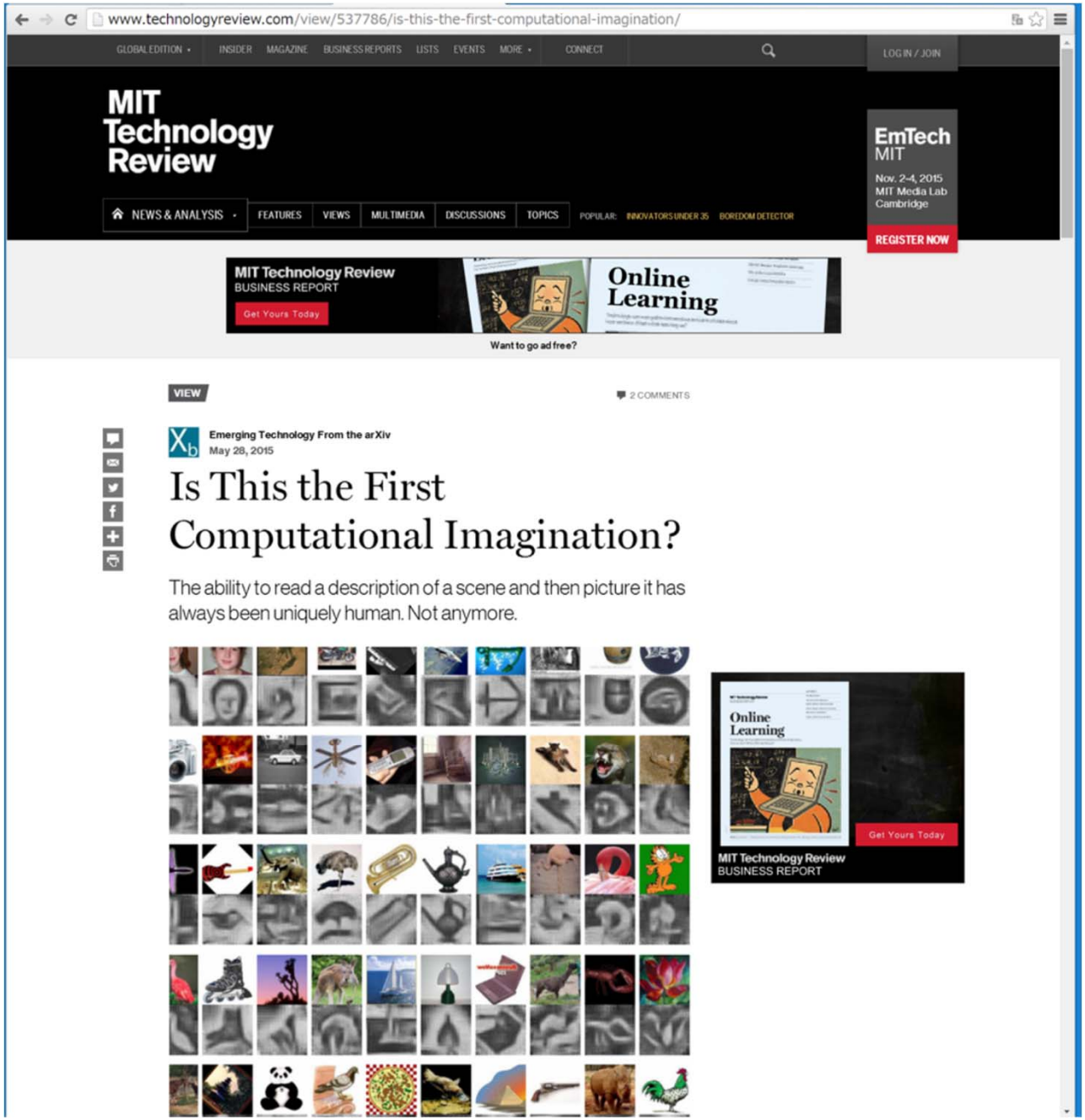
Hiroharu Kato and Tatsuya Harada. Image Reconstruction from Bag-of-Visual-Words. arXiv:1505.05190v1 [cs.CV] 19 May 2015.

- 生成した画像の例

(b) Bottle on a cliff.

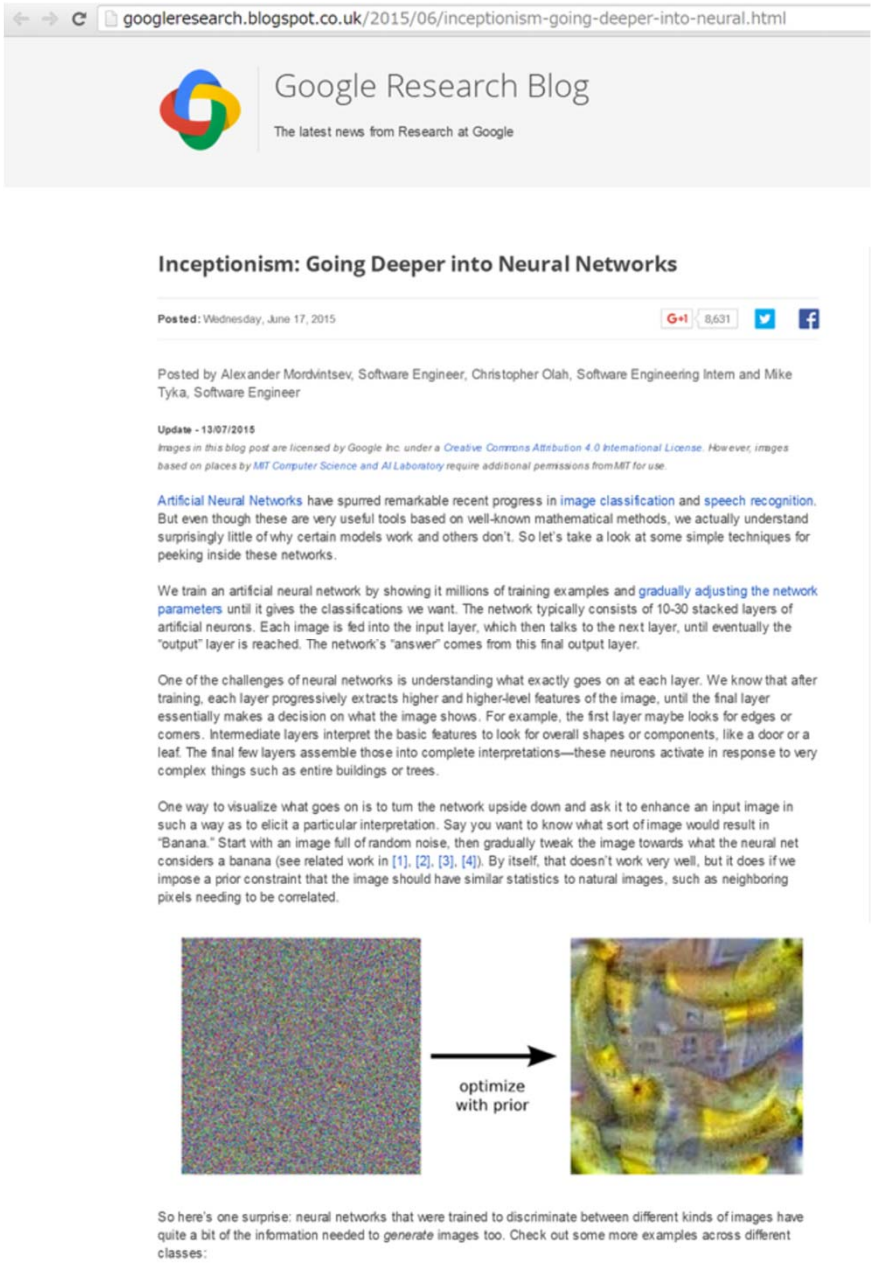
(c) Bus on a field.

(d) Monitor on a coast.



- MIT Technology Review
- Emerging Technology From the arXiv
- May 28, 2015
- http://www.technologyreview.com/view/537786/is-this-the-first-computational-imagination/

Inceptionism: Going Deeper into Neural Networks



googleresearch.blogspot.co.uk/2015/06/inceptionism-going-deeper-into-neural.html

Google Research Blog

The latest news from Research at Google

Inceptionism: Going Deeper into Neural Networks

Posted: Wednesday, June 17, 2015

Posted by Alexander Mordvintsev, Software Engineer, Christopher Olah, Software Engineering Intern and Mike Tyka, Software Engineer

Update - 13/07/2015

Images in this blog post are licensed by Google Inc. under a Creative Commons Attribution 4.0 International License. However, images based on places by MIT Computer Science and AI Laboratory require additional permissions from MIT for use.

Artificial Neural Networks have spurred remarkable recent progress in image classification and speech recognition. But even though these are very useful tools based on well-known mathematical methods, we actually understand surprisingly little of why certain models work and others don't. So let's take a look at some simple techniques for peeking inside these networks.

We train an artificial neural network by showing it millions of training examples and gradually adjusting the network parameters until it gives the classifications we want. The network typically consists of 10-30 stacked layers of artificial neurons. Each image is fed into the input layer, which then talks to the next layer, until eventually the "output" layer is reached. The network's "answer" comes from this final output layer.

One of the challenges of neural networks is understanding what exactly goes on at each layer. We know that after training, each layer progressively extracts higher and higher-level features of the image, until the final layer essentially makes a decision on what the image shows. For example, the first layer maybe looks for edges or corners. Intermediate layers interpret the basic features to look for overall shapes or components, like a door or a leaf. The final few layers assemble those into complete interpretations—these neurons activate in response to very complex things such as entire buildings or trees.

One way to visualize what goes on is to turn the network upside down and ask it to enhance an input image in such a way as to elicit a particular interpretation. Say you want to know what sort of image would result in "Banana." Start with an image full of random noise, then gradually tweak the image towards what the neural net considers a banana (see related work in [1], [2], [3], [4]). By itself, that doesn't work very well, but it does if we impose a prior constraint that the image should have similar statistics to natural images, such as neighboring pixels needing to be correlated.

So here's one surprise: neural networks that were trained to discriminate between different kinds of images have quite a bit of the information needed to *generate* images too. Check out some more examples across different classes:

- Posted: Wednesday, June 17, 2015
- Posted by Alexander Mordvintsev, Software Engineer, Christopher Olah, Software Engineering Intern and Mike Tyka, Software Engineer
- http://googleresearch.blogspot.co.uk/2015/06/inceptionism-going-deeper-into-neural.html

Generating Images from Captions with Attention

Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, Ruslan Salakhutdinov.
Generating Images from Captions with Attention. ICLR, 2016.

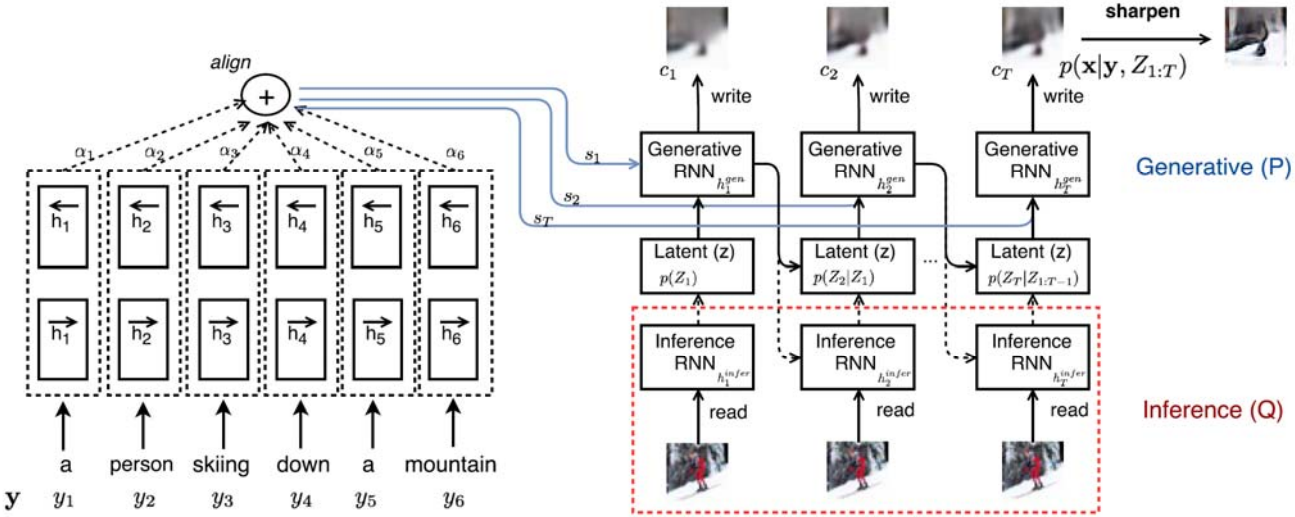
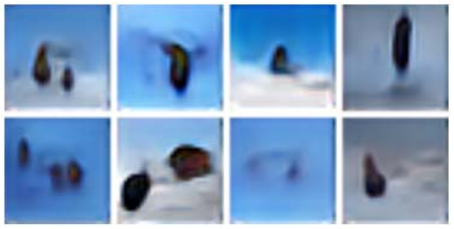


Figure 2: AlignDRAW model for generating images by learning an alignment between the input captions and generating canvas. The caption is encoded using the Bidirectional RNN (left). The generative RNN takes a latent sequence $z_{1:T}$ sampled from the prior along with the dynamic caption representation $s_{1:T}$ to generate the canvas matrix c_T, which is then used to generate the final image $\mathbf{x}$ (right). The inference RNN is used to compute approximate posterior Q over the latent sequence.

A red school bus parked in a parking lot.

A green school bus parked in a parking lot.

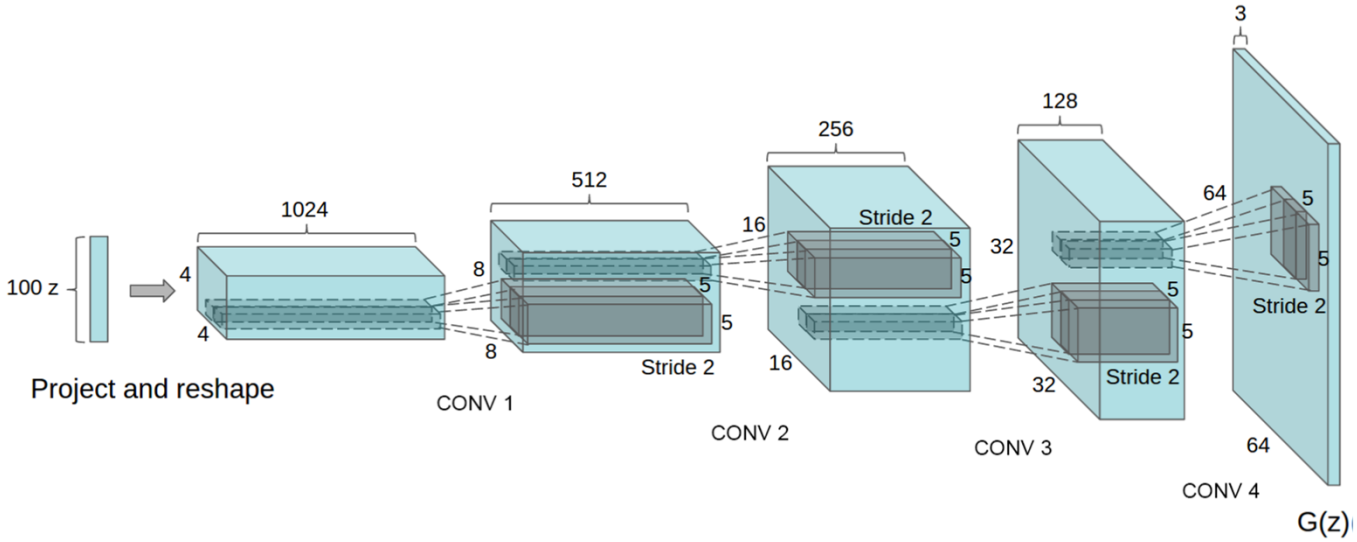
A stop sign is flying in blue skies.

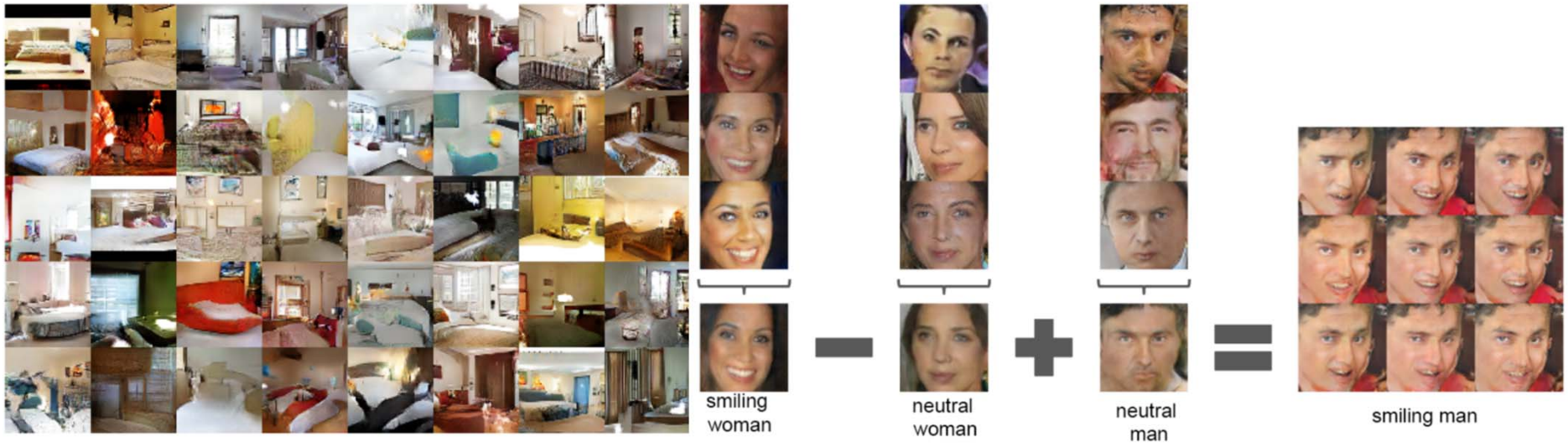


A herd of elephants flying in the blue skies.

Deep Convolutional Generative Adversarial Networks

Alec Radford, Luke Metz, Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv:1511.06434v2 [cs.LG] 7 Jan 2016





A Neural Algorithm of Artistic Style

- Leon A. Gatys, Alexander S. Ecker, Matthias Bethge. A Neural Algorithm of Artistic Style. arXiv:1508.06576, 2015.

Artistic style transfer for videos

Manuel Ruder, Alexey Dosovitskiy, Thomas Brox. Artistic style transfer for videos. arXiv:1604.08610v1 [cs.CV] 28 Apr 2016

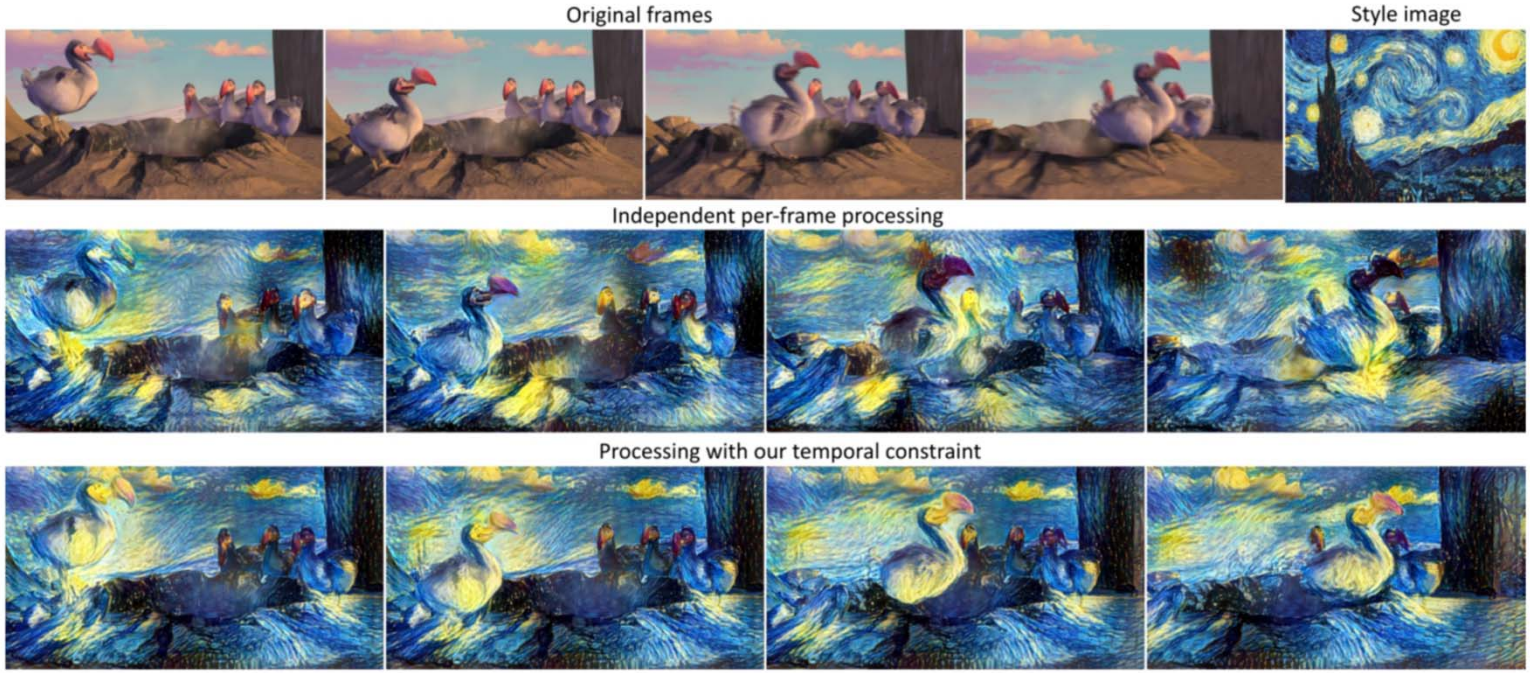


Fig. 1. Scene from *Ice Age* (2002) processed in the style of *The Starry Night*. Comparing independent per-frame processing to our time consistent approach, the latter is clearly preferable. Best observed in the supplemental video, see section 8.1.

Artistic style transfer for videos

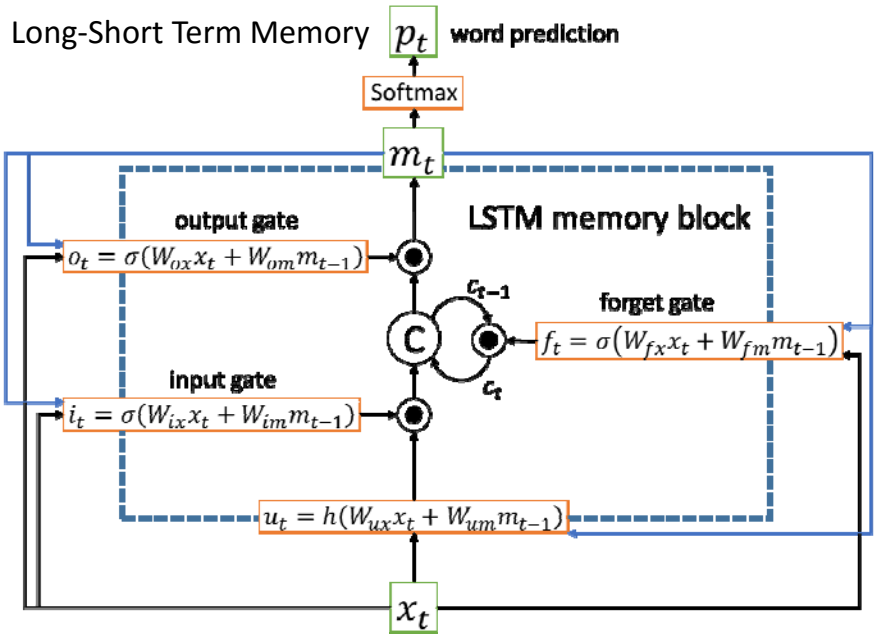
Manuel Ruder, Alexey Dosovitskiy, Thomas Brox. Artistic style transfer for videos. arXiv:1604.08610v1 [cs.CV] 28 Apr 2016

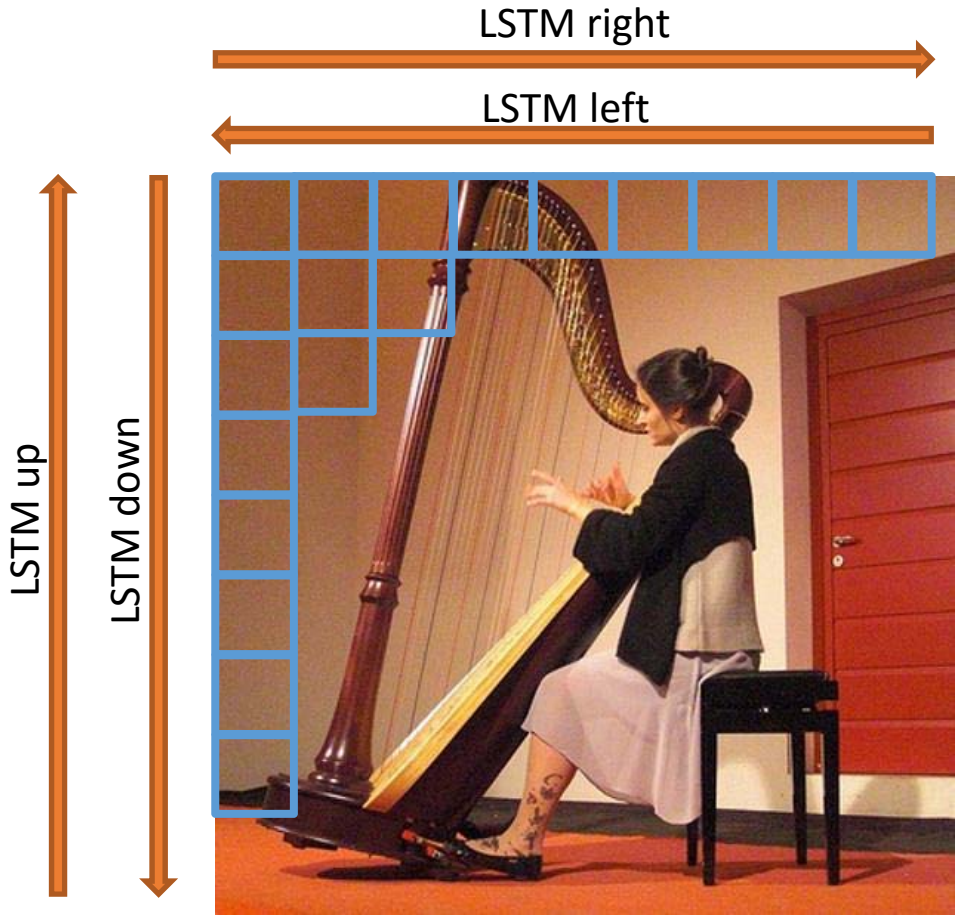
Naturalness of Images

H. Kato and T. Harada. arXiv:1511.02872, 2015.

- 2 dimensional LSTM
  - Hiroharu Kato, Tatsuya Harada. Visual Language Modeling on CNN Image Representations. arXiv:1511.02872, 2015.

$$
\begin{aligned}
u_{y,x,l} = & \frac{x+1}{W_l}||f_{y,x+1,l} - \mu^{\text{right}}_{y,x+1,l}||_2^2 + \\
& \frac{W_l-x+1}{W_l}||f_{y,x,l} - \mu^{\text{left}}_{y,x,l}||_2^2 + \\
& \frac{y+1}{H_l}||f_{y+1,x,l} - \mu^{\text{down}}_{y+1,x,l}||_2^2 + \\
& \frac{H_l-y+1}{H_l}||f_{y,x,l} - \mu^{\text{up}}_{y,x,l}||_2^2.
\end{aligned}
$$

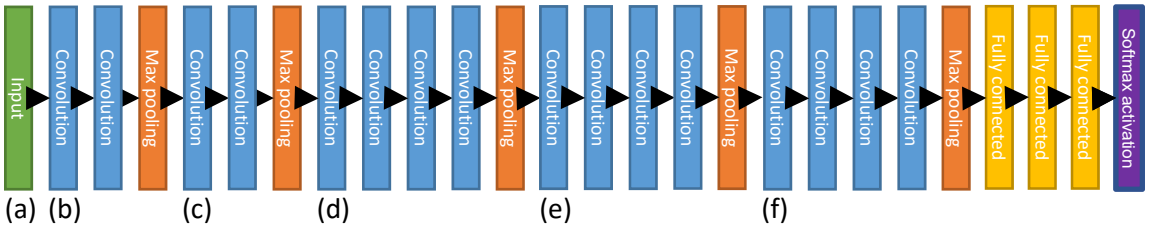


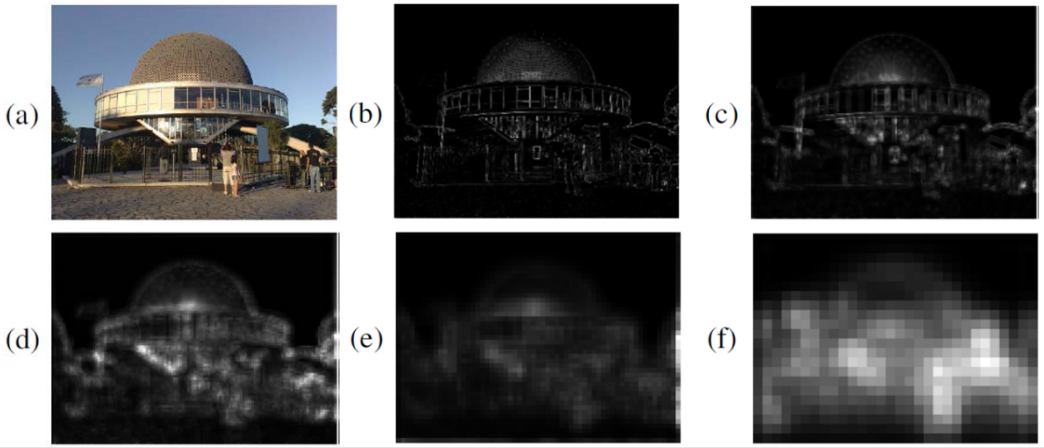


Saliency Detection

H. Kato and T. Harada. arXiv:1511.02872, 2015.

- Unnaturalness on each layer of VGGNet





Dataset	Mr-CNN	AWS	BMS	CA	eDN	HFT	ICL	IS	JUDD	LG	QDCT	Ours
MIT1003	.7190	.6945	.6939	.6718	.6273	.6526	.6667	.6686	.6631	.6823	.6686	**.7203**
Toronto	.7236	.7184	.7221	.6959	.6292	.6926	.6939	.7115	.6901	.6990	.7174	**.7323**

Table 3. Shuffled AUC score of each method and dataset. Scores aside from ours are cited from Liu *et al.* [36].

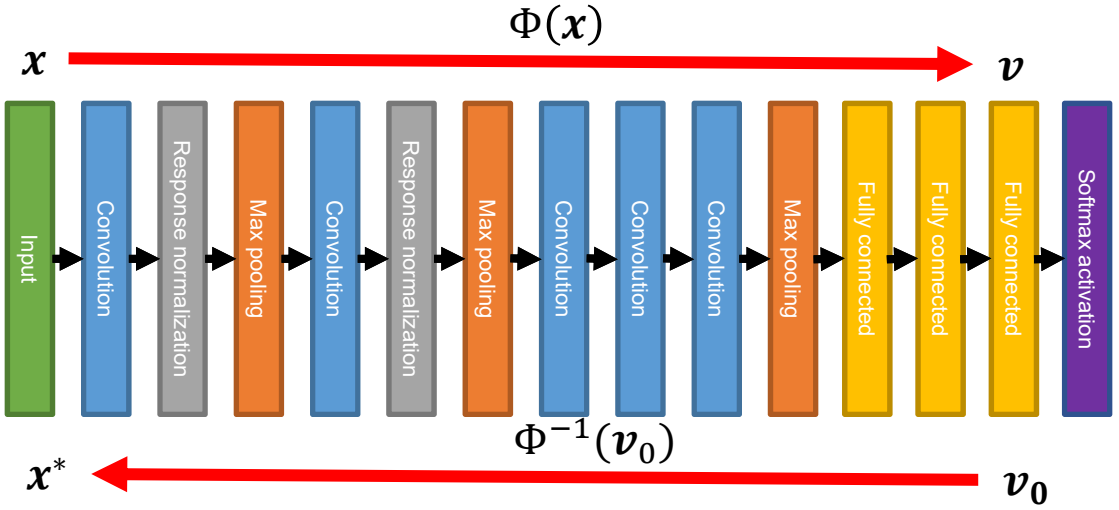
Dataset	SALICON	Deep Fix	Deep Gaze I^2	SalNet	Mr-CNN	AWS	CA	WMAP	IttiKoch2	Ours
MIT300	**.74**	.71	.71	.69	.69	.68	.65	.63	.63	.7096
CAT2000	-	.57	-	-	-	**.62**	.60	.60	.59	**.6221**

Table 4. Shuffled AUC score on MIT Saliency Benchmark. Scores are available online. These scores are retrieved on 30 October 2015.

Image Reconstruction

H. Kato and T. Harada. arXiv:1511.02872, 2015.

- Image reconstruction
  - Using unnaturalness as a regularizer



$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\mathbb{R}^{H\times W\times C}}{\operatorname{argmin}} \; l(\Phi(\boldsymbol{x}), \boldsymbol{v}_0) + \lambda\mathcal{R}(\boldsymbol{x})$$

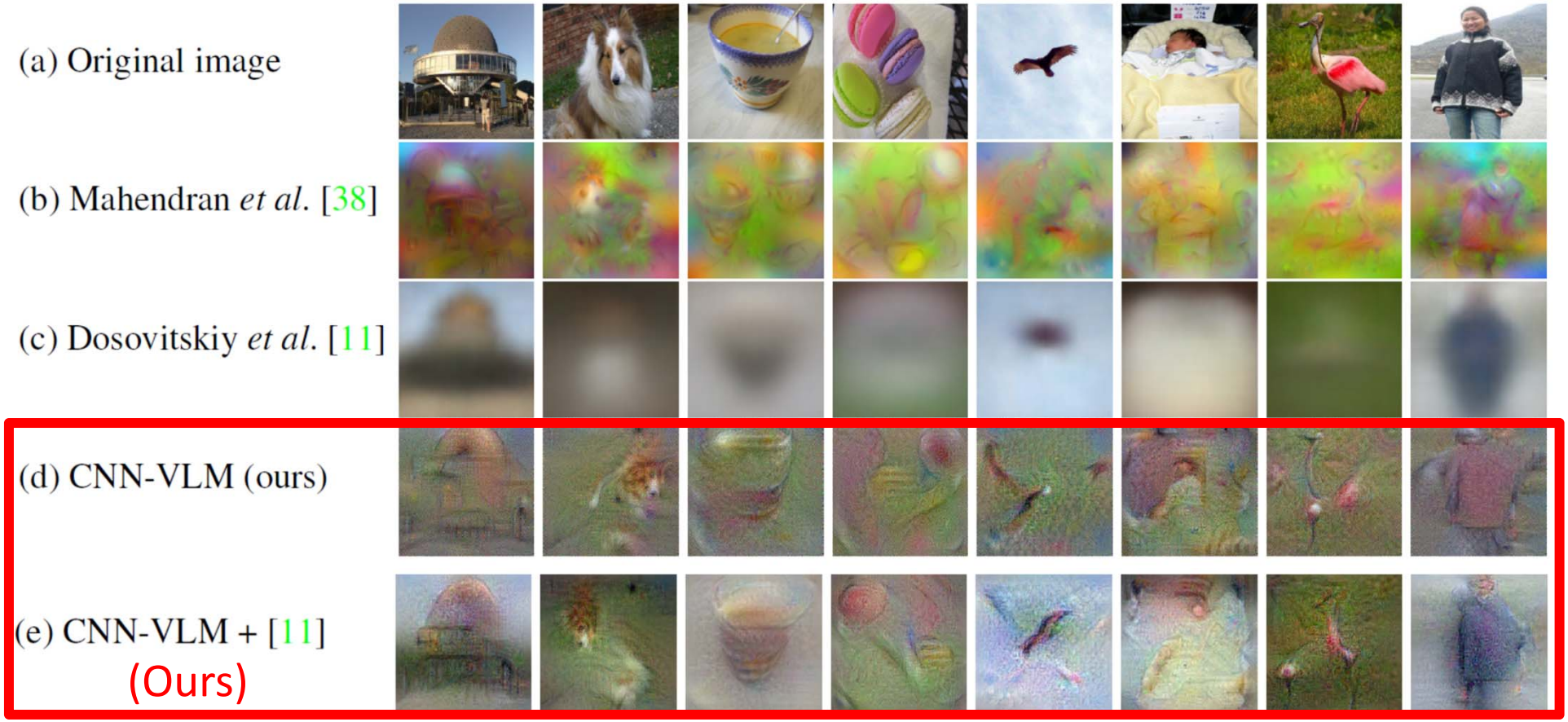
$$l(\Phi(\boldsymbol{x}), v_0) = \|\Phi(\boldsymbol{x}) - \boldsymbol{v}_0\|_2^2$$

$$\begin{aligned} u_{y,x,l} = \; & \tfrac{x+1}{W_l}\|f_{y,x+1,l} - \mu^{\text{right}}_{y,x+1,l}\|_2^2 + \\ & \tfrac{W_l-x+1}{W_l}\|f_{y,x,l} - \mu^{\text{left}}_{y,x,l}\|_2^2 + \\ & \tfrac{y+1}{H_l}\|f_{y+1,x,l} - \mu^{\text{down}}_{y+1,x,l}\|_2^2 + \\ & \tfrac{H_l-y+1}{H_l}\|f_{y,x,l} - \mu^{\text{up}}_{y,x,l}\|_2^2. \end{aligned}$$

Unnaturalness of image using 2D LSTM

Image Reconstruction Results

H. Kato and T. Harada. arXiv:1511.02872, 2015.



Robotic Grasping with Deep Learning

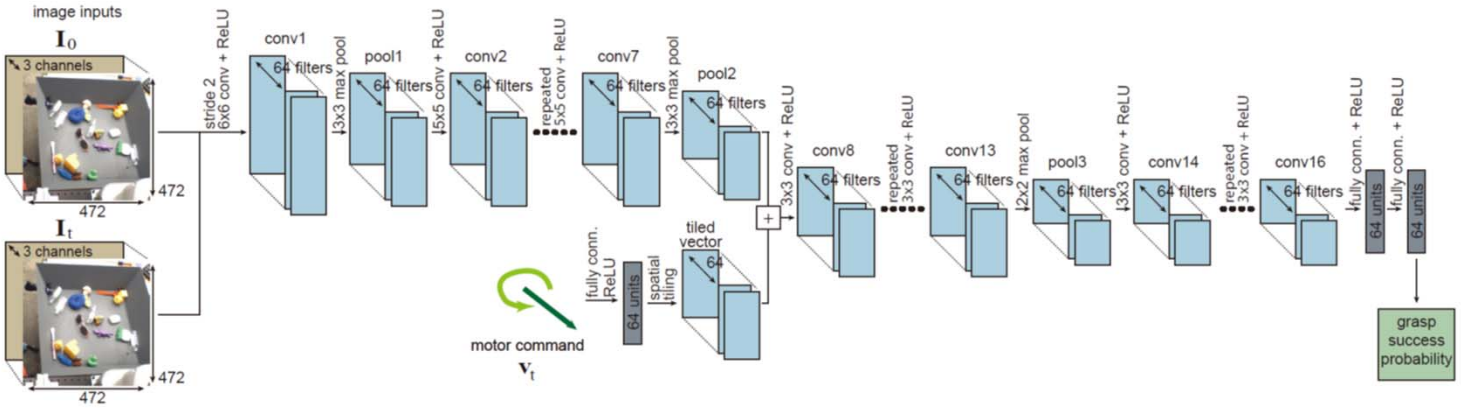


Figure 1. Our large-scale data collection setup, consisting of 14 robotic manipulators. We collected over 800,000 grasp attempts to train the CNN grasp prediction model.

Figure 10. Examples of difficult objects grasped by our algorithm, including objects that are translucent, awkardly shaped, and heavy.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Deirdre Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. arXiv:1603.02199v1 [cs.LG] 7 Mar 2016

Robotic Grasping with Deep Learning

Sergey Levine, Peter Pastor, Alex Krizhevsky, Deirdre Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. arXiv:1603.02199v1 [cs.LG] 7 Mar 2016