データサイエンス
第11回

~深層学習・画像認識~

情報理工学系研究科
創造情報学専攻
中山 英樹

レポート課題 1

- お疲れさまでした（提出132名）
  - 提出者一覧を確認してください
- ネタばらし
  - Facebook comment volume prediction
    https://archive.ics.uci.edu/ml/datasets/Facebook+Comment+Volume+Dataset

    K.Singh, R. Kaur, and D. Kumar, ” Comment Volume Prediction Using Neural Networks and Decision Trees”, In Proc. 17th UKSIM-AMSS International Conference on Modelling and Simulation, 2015.
- 最終結果

制限あり

Rank	Name	Mean ABS Error
1	z	3.7507134880708
2	zukkyun	3.7785930350644
3	amaotone	3.8785407265772
4	scarlet	3.8968
5	Hugh	3.921

制限なし

Rank	Name	Mean ABS Error
1	zukkyun	3.5861741273348
2	amaotone	3.6572861451587
3	YI	3.7196
4	shinkyo	3.7471082293018
5	nagano	3.7521

前処理・特徴抽出（次元圧縮）

- 分散正規化（標準化）
  - 各説明変数ごとに平均0、分散1に

$$x' = \frac{x - \mu_x}{\sqrt{\sigma_x^2 + \varepsilon}}$$

- ZCA白色化 (zero component analysis)
  - PCAを利用して無相関化も行う

$$C_X \mathbf{a} = \lambda \mathbf{a}$$

$$\mathbf{x}'' = A \Lambda^{-1/2} A^T \left(\mathbf{x} - \overline{\mathbf{x}} \right)$$

$$A = \left(\mathbf{a}_1 \, \mathbf{a}_2 \cdots \mathbf{a}_n \right) \quad \Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

- GBDT (Gradient boosting decision tree)
  - 決定木を用いたブースティングの一種（後述）
  - 各決定木で”どの葉ノードに落ちたか”を特徴量とする

Gradient boosting （勾配ブースティング）

- 損失関数を最も下げるように、弱識別器を追加

Algorithm 1: Gradient_Boost

1. $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$
2. For $m = 1$ to M do:
3. $\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \; i = 1, N$
4. $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$
5. $\rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
6. $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
7. endFor

end Algorithm

それまでの識別器の予測値に関する損失関数の勾配

訓練データに関する負勾配

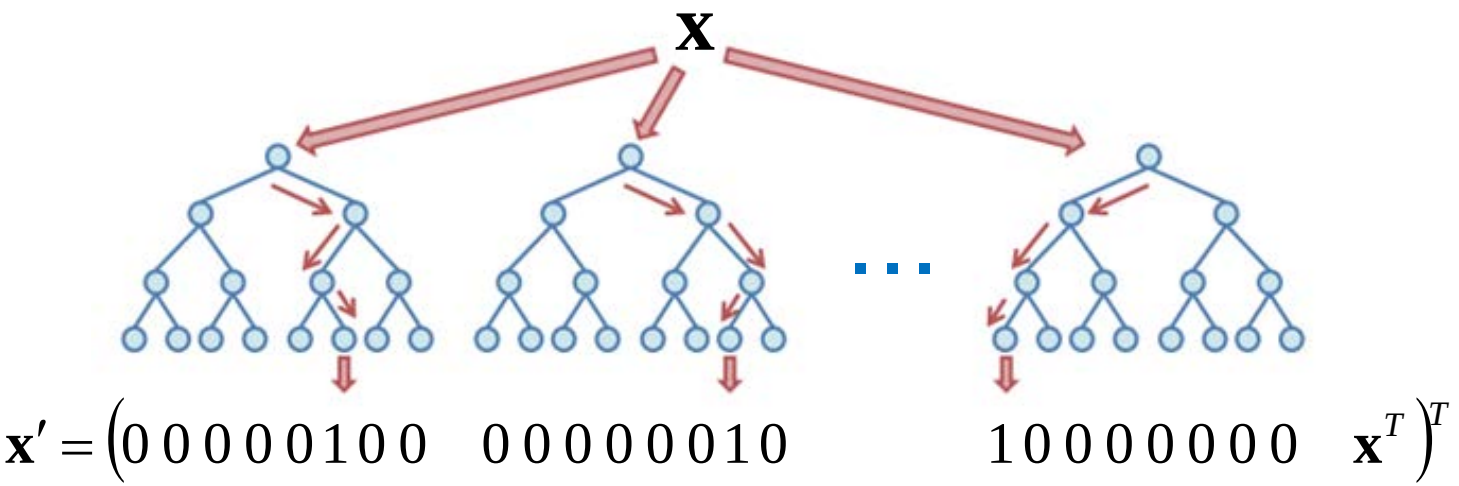
負勾配を回帰するように次の識別器を学習

識別器の重みを別途探索

Jerome H. Friedman, “Greedy function approximation: A gradient boosting machine”, Ann. Statist., Volume 29, Number 5, pp.1189-1232, 2001.

Gradient boosting decision tree (GBDT)

- 勾配ブースティング法によって決定木をアンサンブル
  - ランダムフォレストはバギング
  - Xgboost、LightGBM というソフトウェアが有名
- 各決定木でどの葉ノードに落ちたかをメタ特徴にとる
  - 特徴選択
  - Kaggle等で流行の前処理の一つ



多かった要望

- 部屋が狭い（人が多すぎる）
- 声が聞こえない
- 演習時間が少ない
- 初心者にはついていけない
- 内容が多すぎる（話題を絞ったほうがよい）
- レポートを共有して欲しい

第三世代：Deep learning (深層学習)

- Deep learning (深層学習) (2006～)
  [Hinton and Salakhutdinov, Science, 2006]

- 従来扱われてきたよりもさらに多層のニューラルネット
  - 2010年頃で7～8層程度。現在は150層以上のものも！

- 音声認識・画像認識・自然言語処理などさまざまな分野で圧倒的な性能を達成

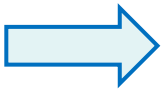
- 生データから目的変数に至るend-to-endの構造を学習
  - 従来の特徴量（に相当する構造）も自動的に獲得
  - パターン認識の文脈では表現学習（representation learning)とほぼ同義で扱われることも

なぜ deep がよいのか？

ある一定の表現能力を得ようとした場合に…

- 深いモデルの方が必要なパラメータ数が少なくて済むと考えられている [Larochelle et al., 2007] [Bengio, 2009] [Delalleau and Bengio, 2011]
  (※単純なケース以外では完全に証明されていない)

（ちゃんと学習できれば）



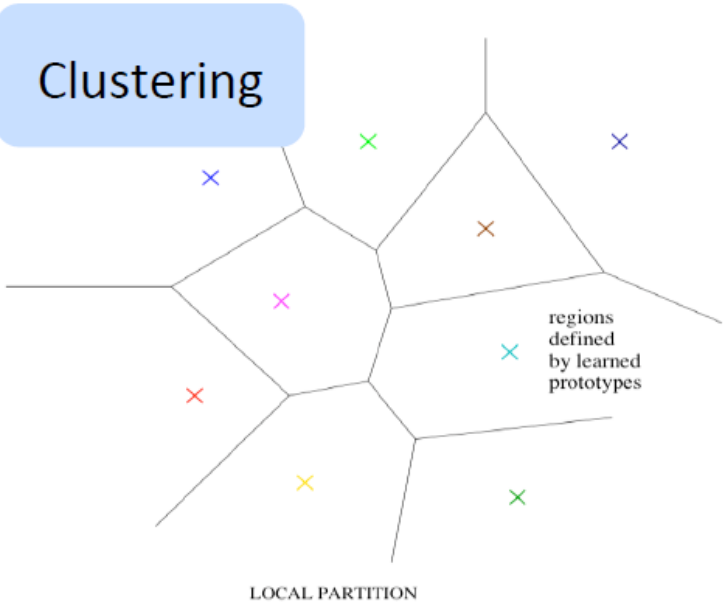
- 汎化性能
- 計算効率
- スケーラビリティ

反論もある:
“Do Deep Nets Really Need to be Deep?”
[Ba & Caruana, 2014]

入力空間の領域分割の観点からの分析 [Bengio]

- 細かく分割できるほどより表現能力が高い

- 浅いモデルでは、パラメーター数に対し線形に領域数が増加
  - クラスタリング、最近傍法、カーネル密度推定、SVM、etc.

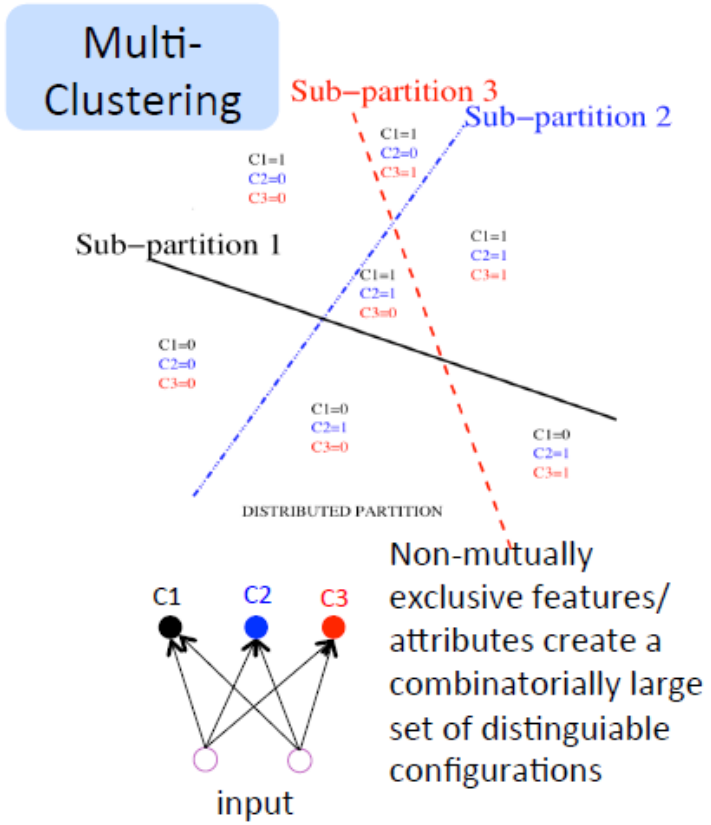


Bengio, “Deep Learning for AI”
(http://www.iro.umontreal.ca/~bengioy/talks/AGI-2aug2014.pdf)
より一部引用

入力空間の領域分割の観点からの分析 [Bengio]

- 深いモデルでは、パラメーター数に対し指数的に領域数が増加
- 複数の潜在ノードの活性の組み合わせで領域を表現可能
- 一つの入力の情報が複数のノード(特徴)に分散して表現される

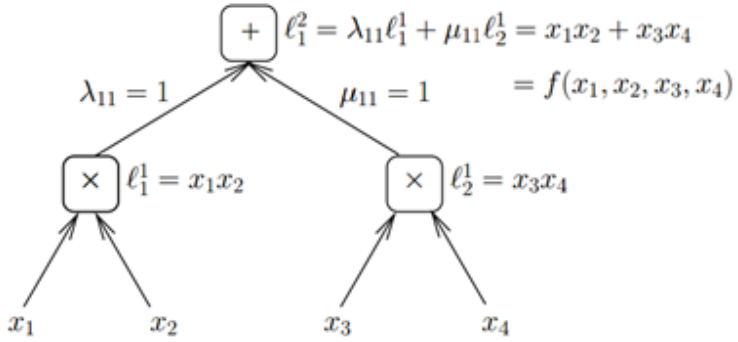
= 分散表現 (distributed representatior

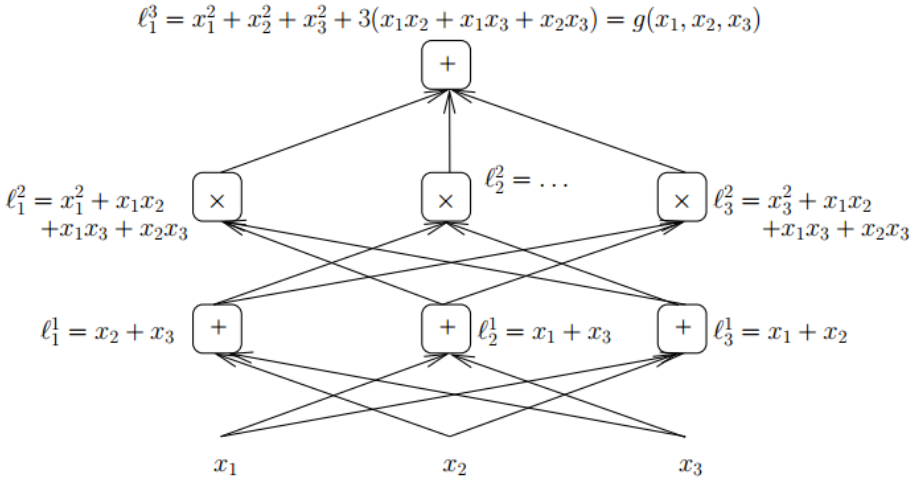


Bengio, “Deep Learning for AI”
(http://www.iro.umontreal.ca/~bengioy/talks/AGI-2aug2014.pdf)
より一部引用

証明されているネットワークの例

- Sum-product network [Poon and Domingos, UAI'11]
  - 各ノード(ニューロン)が入力の和か積を出力するネットワーク



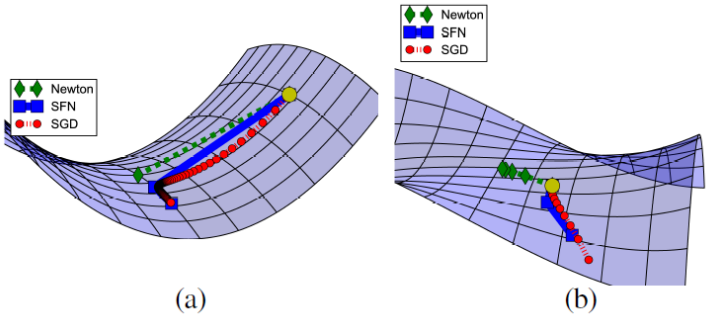


- 同じ多項式関数を表現するために必要なノード数の増加
  - 浅いネットワークでは指数的
  - 深いネットワークでは線形

[Delalleau & Bengio, NIPS'11]

局所最適解？

- [Dauphin+, NIPS'14]
  - 高次元の問題では局所解より saddle-point が支配的になる (という仮説と実験的観察)



Y. N. Dauphin et al., "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization", In Proc. NIPS, 2014.

- [Choromanska+, AISTATS'15]
  - 大規模な問題では、多くの局所最適解が大域的最適解に十分近くなる？
  - 物理系(Spin-glassモデル)のハミルトニアンによる多層フィードフォワードニューラルネットの損失関数の分析
  - モデルの非現実的な前提が多いが、少しずつこれを取り除いて一般的な証明に近づこうという方向性を示す

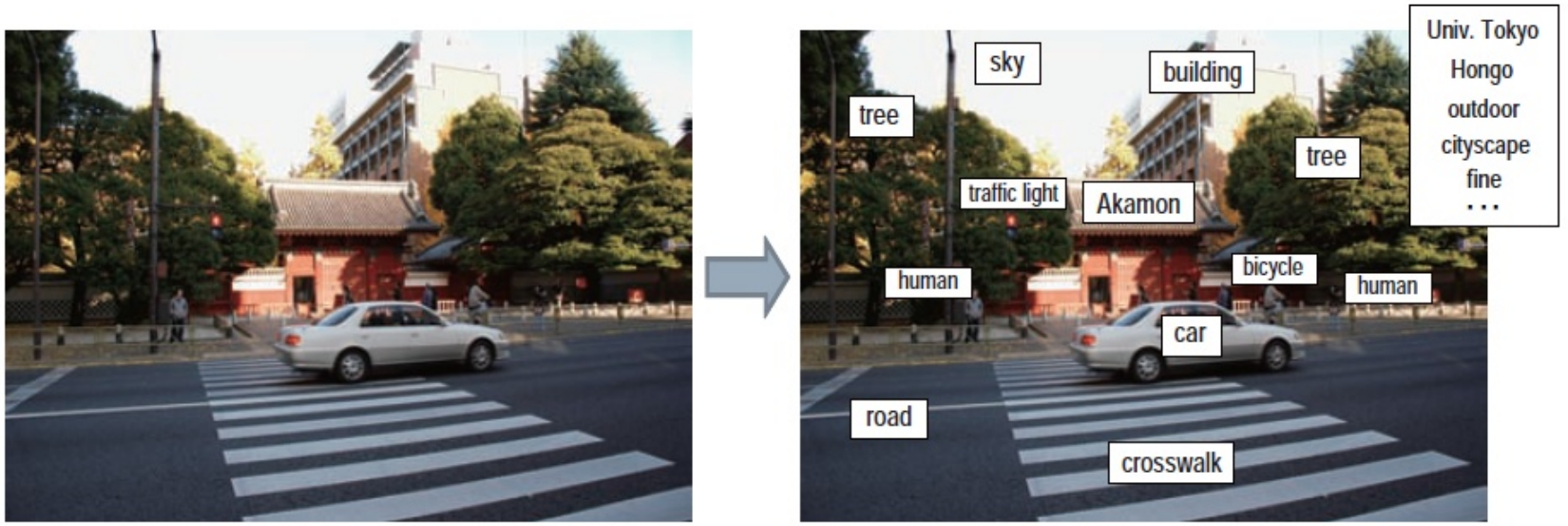
A. Choromanska et al., "The Loss Surfaces of Multilayer Networks", In Proc. AISTATS, 2015.

- [Kawaguchi, NIPS'16]
  - モデルの非現実的な前提を大幅に取り除く

Kawaguchi., "Deep Learning without Poor Local Minima", In Proc. NIPS, 2016.

一般物体認識（一般画像認識）

- 制約をおかない実世界環境の画像を言語で記述
  - **一般的な物体やシーン**、形容詞、印象語
  - 2000年代以降急速に発展（コンピュータビジョンの人気分野）
  - 幅広い応用先



一般物体認識の主要なタスク

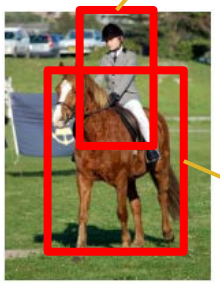
易

難

- Categorization (カテゴリ識別)
  - 映ってる物体の名称を答える
  - 物体の位置を答える必要はない

horse
human

- Detection (物体検出)
  - 矩形で物体の位置を切り出す



- Semantic Segmentation (Scene Parsing)
  - ピクセルレベルで物体領域を認識

Figure from
[Ramanan et al, ICCV'09]

Large-scale recognition

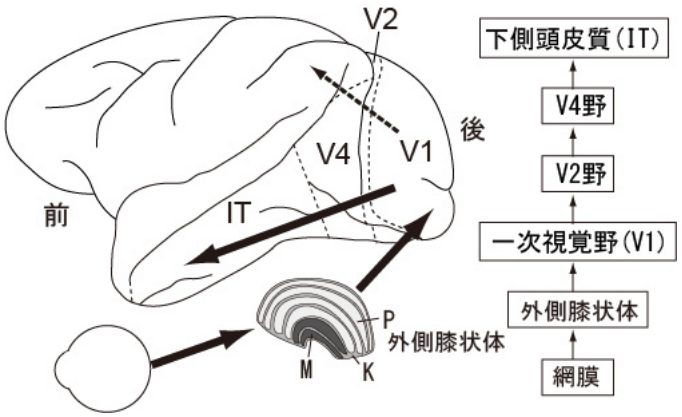
2010

カテゴリ数：10^3~10^4

サンプル数： 10^6~10^7

Figure from Russakovsky et al., ILSVRC'14 slides.

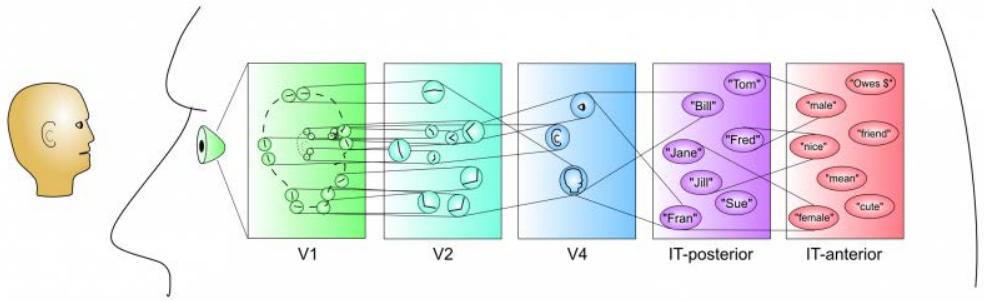
脳(視覚野)の生理学的知見



マカクザルの視覚経路

https://bsd.neuroinf.jp/wiki/色選択性細胞

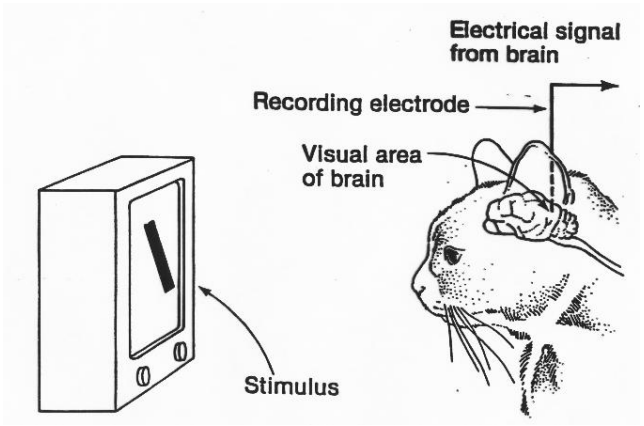
- 階層構造を有する
- 段階的に抽象的な特徴に反応するニューロンが現れる
- 網膜からV1までは, 網膜像の位相幾何学的構造が保たれるように投射される
- V1ニューロンは受容野を持つ

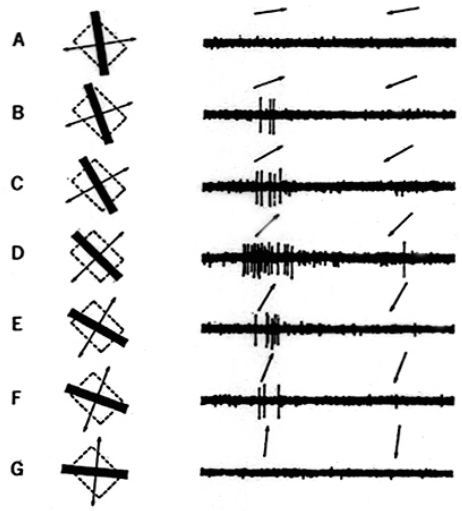


https://grey.colorado.edu/CompCogNeuro/index.php/CCNBook/Perception

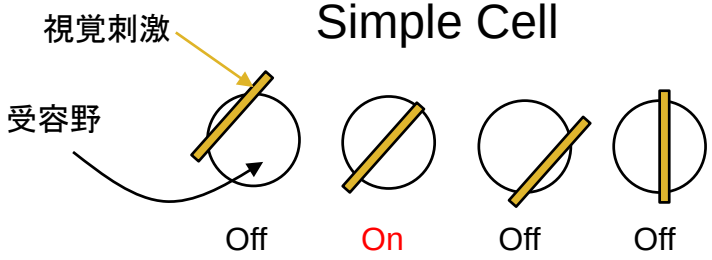
一次視覚野(V1)

- Hubel & Wiesel, 1959
  - 猫の一次視覚野の各ニューロンが、入力パターン(エッジ)の向きに対して選択的に反応することを発見（1981年**ノーベル**賞）



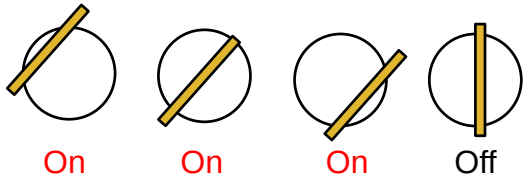


[Hubel and Wiesel, 1968]



Simple Cell

位置・方向に対する選択性

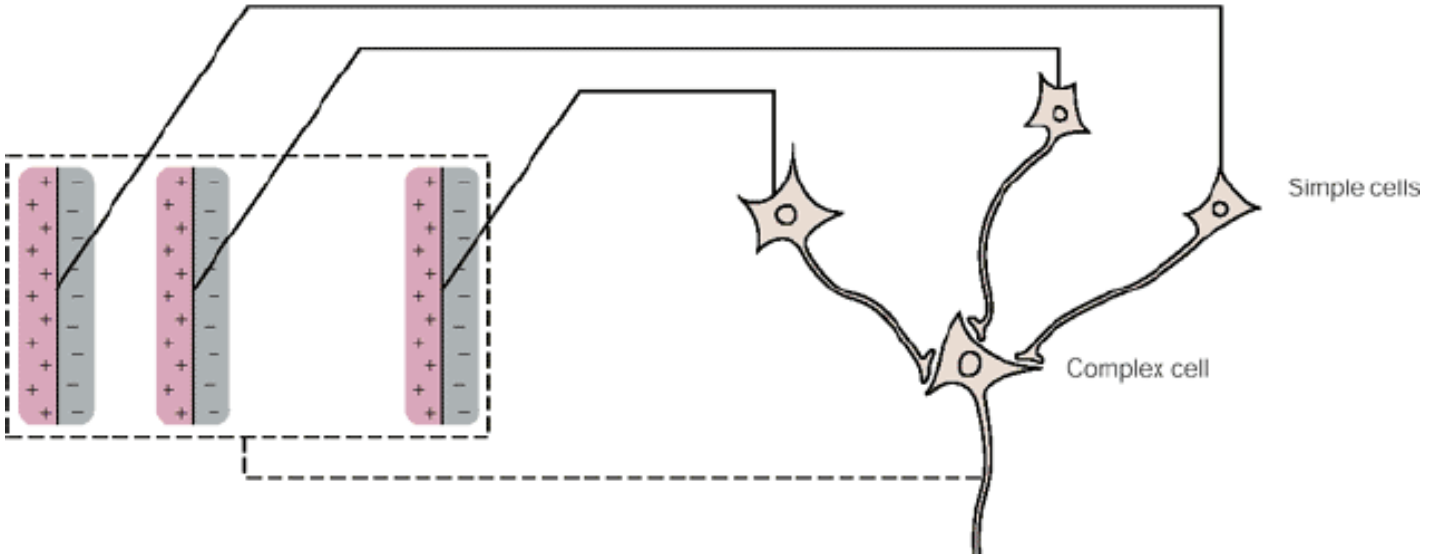


Complex Cell

方向のみに対する選択性
（位置に対する不変性）

Hubel-Wiesel 階層仮説

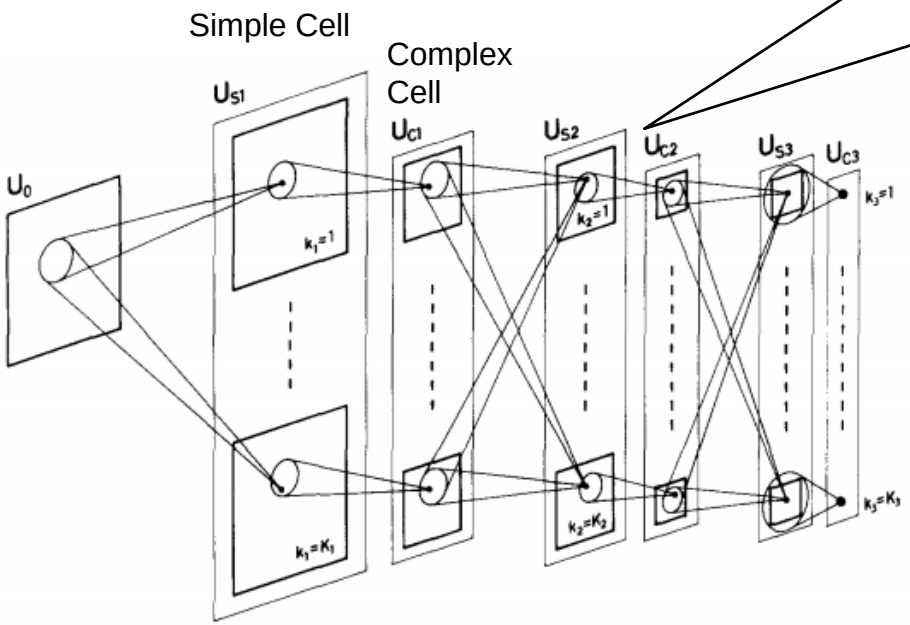
- Simple cell の出力を合成すればComplex cell の挙動は説明できる (という仮説)

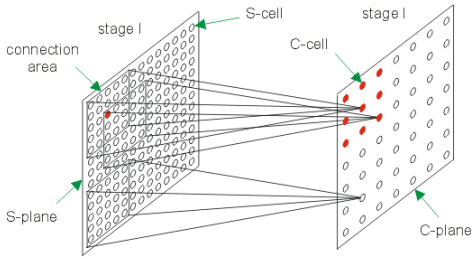


https://kin450-neurophysiology.wikispaces.com/Visual+Cortical+Neurons

Neocognitron

- 福島邦彦先生、1980年代前後
  - 畳み込みニューラルネットワークの原型
  - Simple Cell → Complex Cell の結合はpoolingに対応
  - 段階的に解像度を落としながら、局所的な相関パターンを抽出





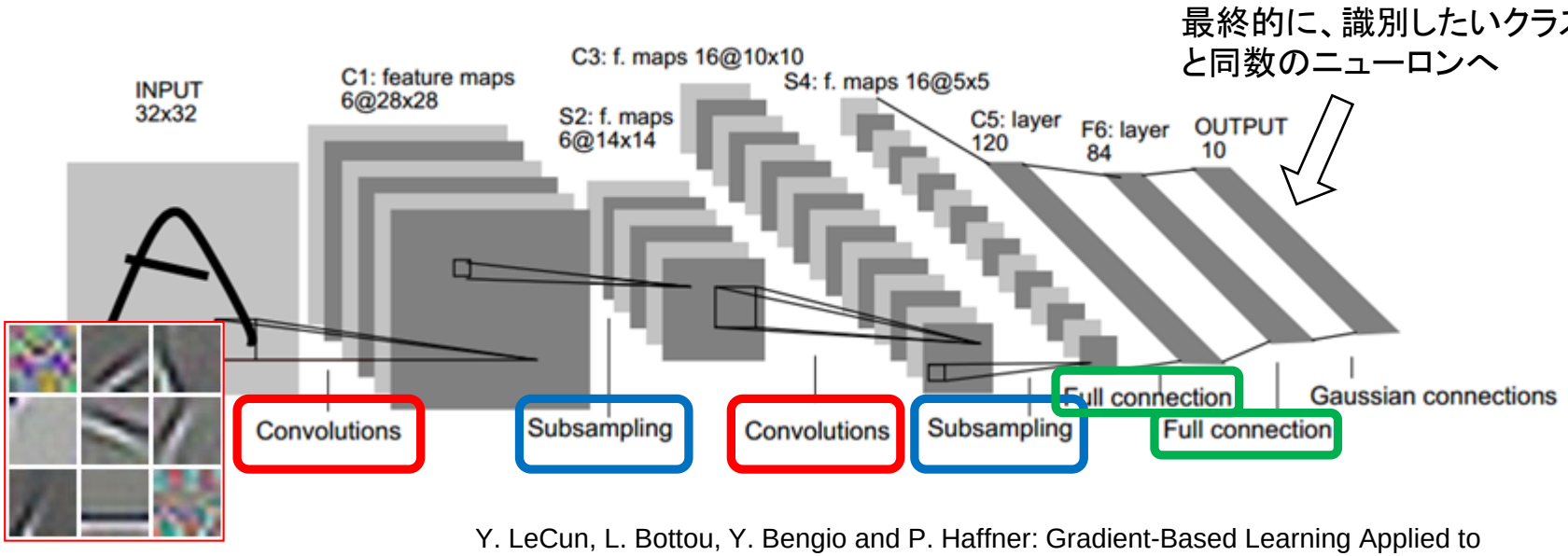
結合重み(フィルタ)の共有
=convolution

http://www.kiv.zcu.cz/studies/predmety/uir/NS/Neocognitron/en/func-C-cell.html

Kunihiko Fukushima, “Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position“, Biological Cybernetics, 36(4): 93-202, 1980.

Convolutional neural network (CNN)

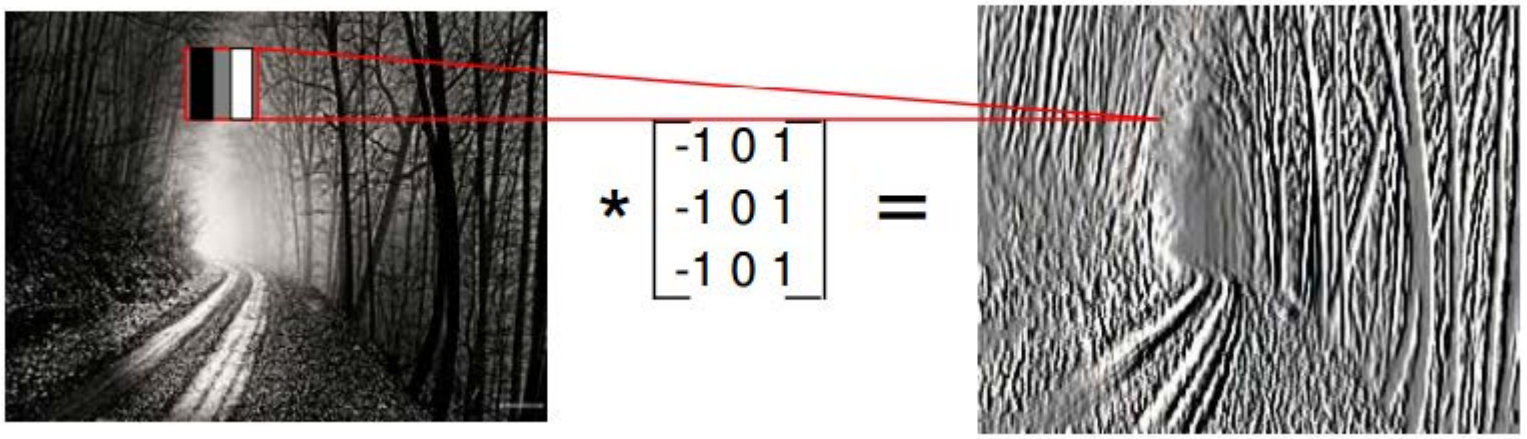
- 局所領域(受容野)の畳み込みとプーリングを繰り返す多層パーセプトロン
  - 単純な全結合ネットワークと比べて大幅にパラメータ数を削減
  - 入力の位置に関する不変性



Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, 1998.

Convolution

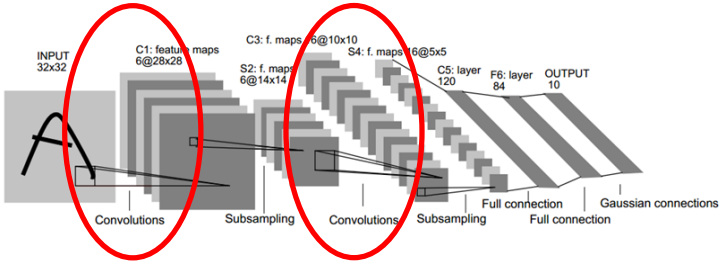
- 一般的なフィルタだと…
  - 例）エッジ抽出



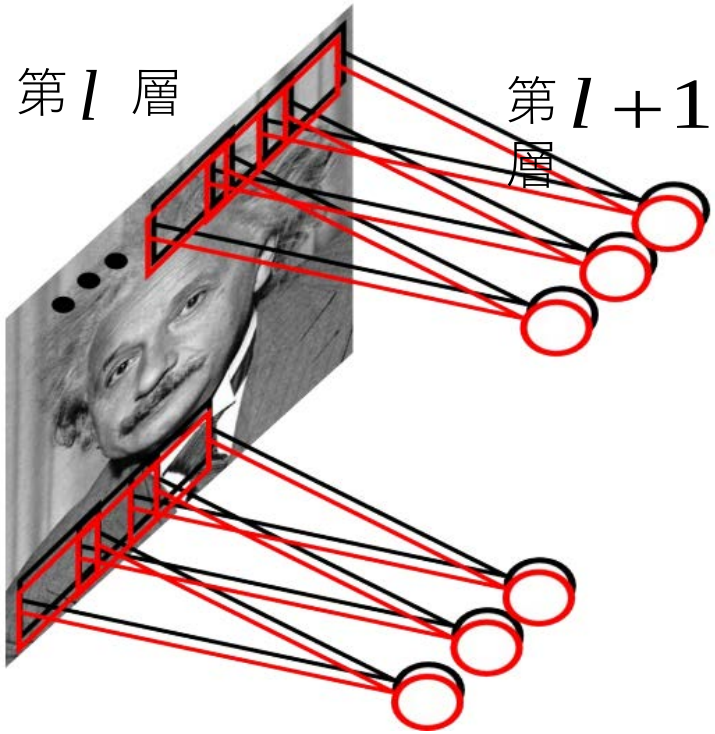
- CNNでは識別に有効なフィルタ係数をデータから学習する

Source: M. Ranzato, CVPR'14 tutorial slides

畳み込み層



- 各フィルタのパラメータは全ての場所で共有
  - 色の違いは異なる畳み込みフィルタを示す



※もちろん入力は生画像のみとは限らない（中間層など）

非線形活性化関数（とても重要）

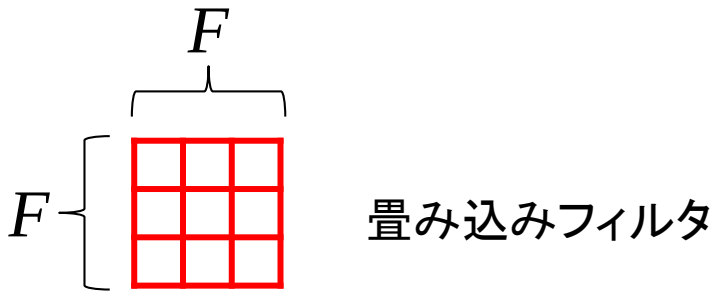
$$\mathbf{z}^{l+1} = h\left(\mathbf{W}^{l+1} * \mathbf{z}^{l} + \mathbf{b}^{l+1}\right)$$

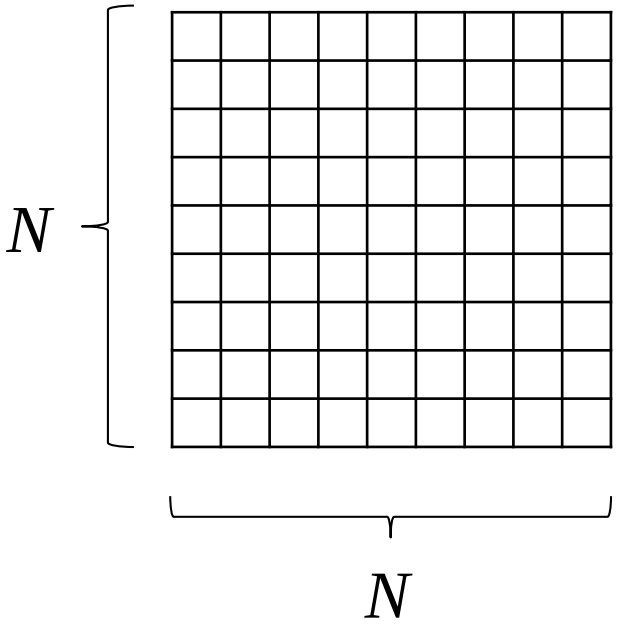
フィルタの係数　入力　バイアス

Source: M. Ranzato, CVPR'14 tutorial slides

畳み込み層

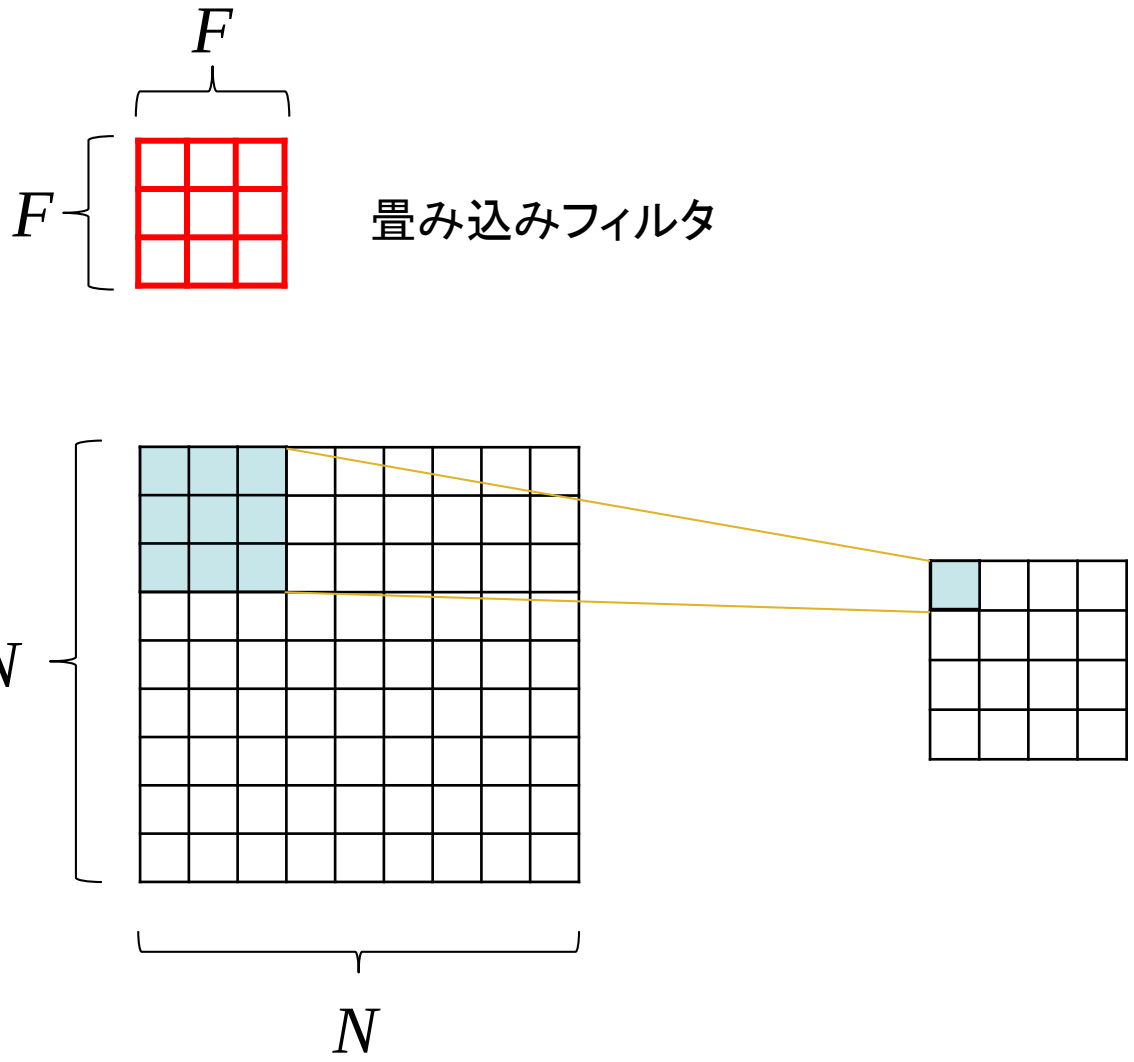
- （まず簡単のため）入力一層、フィルター一つの場合





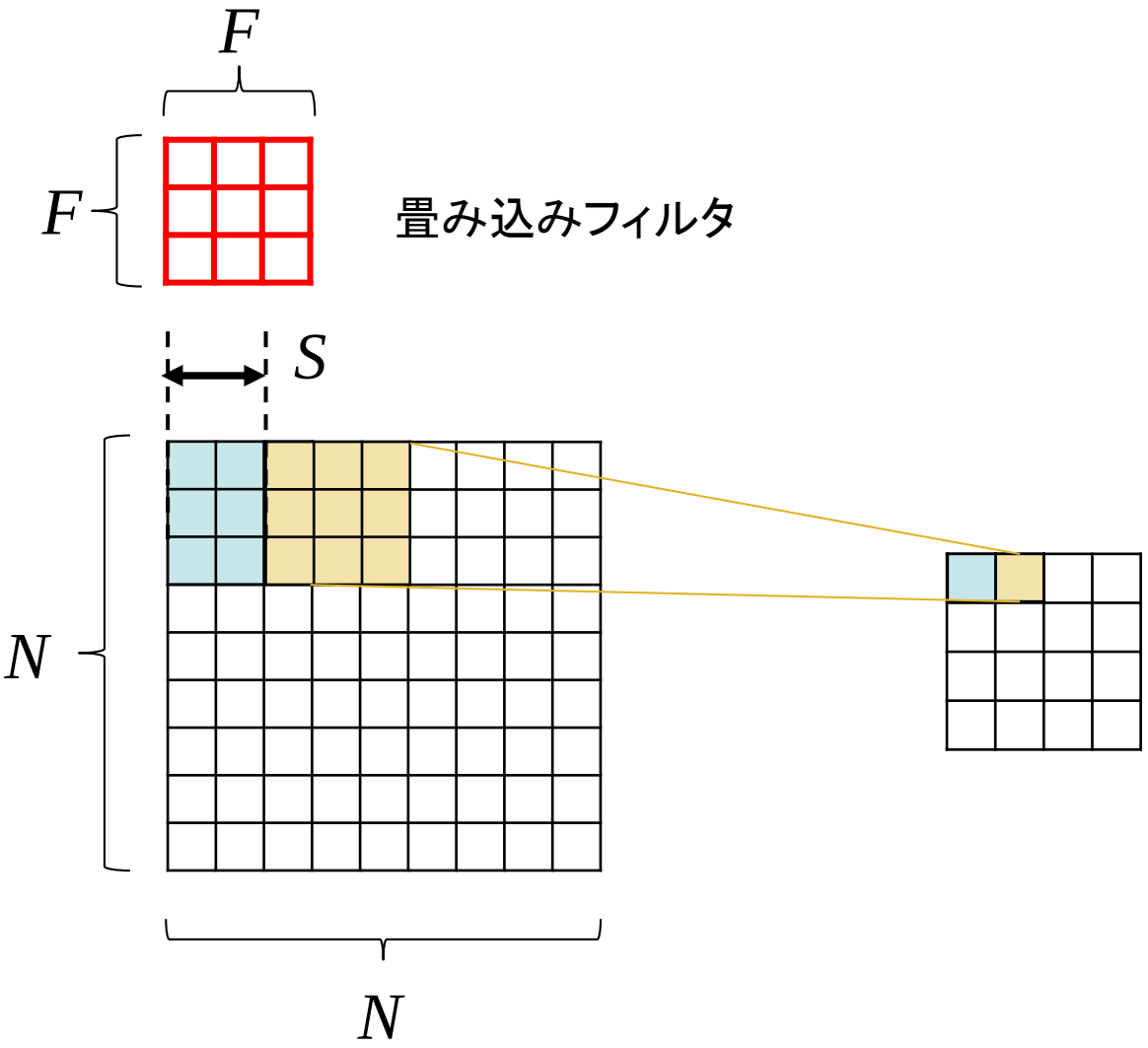
畳み込み層

- （まず簡単のため）入力一層、フィルター一つの場合



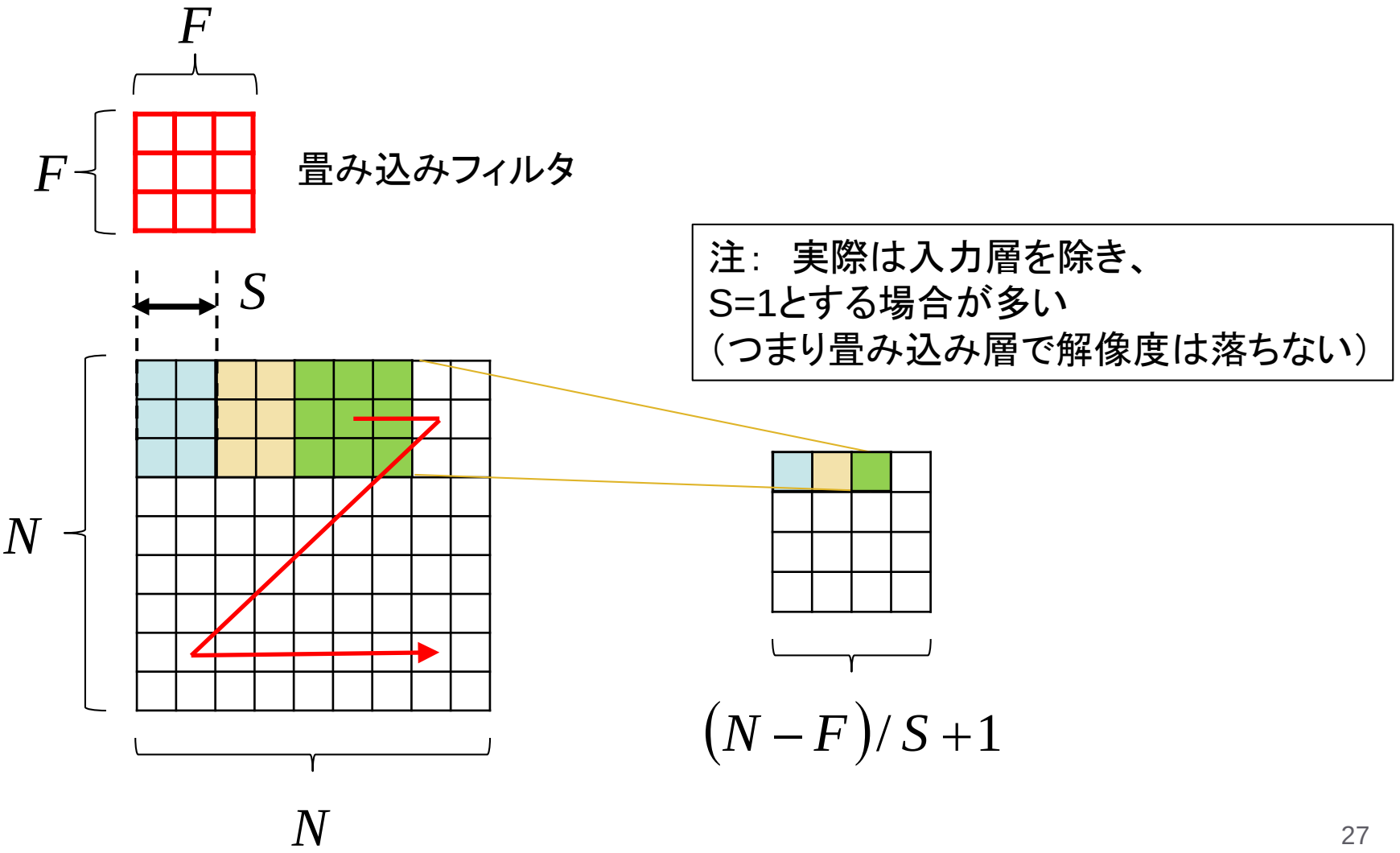
畳み込み層

- （まず簡単のため）入力一層、フィルター一つの場合



畳み込み層

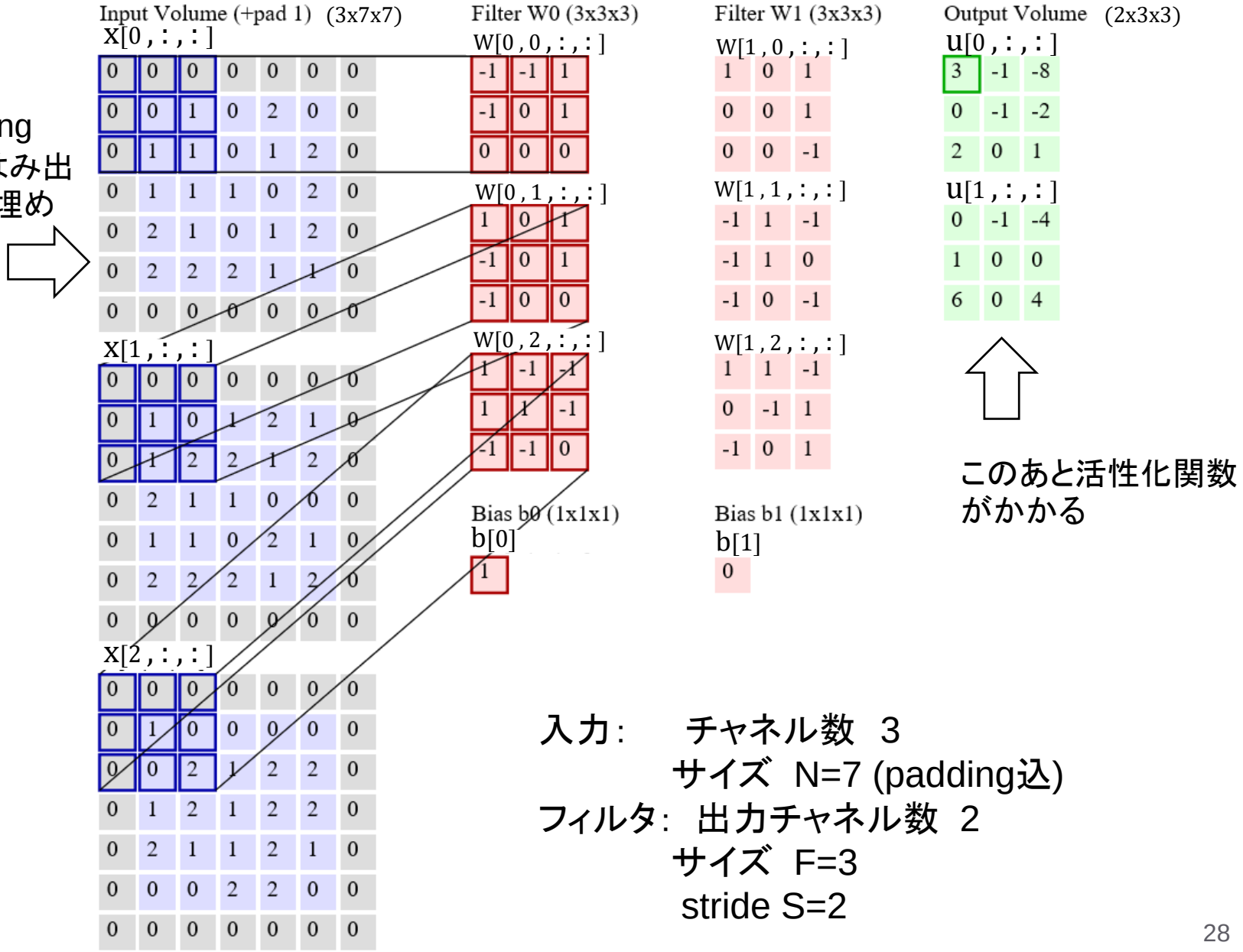
- （まず簡単のため）入力一層、フィルター一つの場合



もう少し詳しく

http://cs231n.github.io/convolutional-networks/ を改変

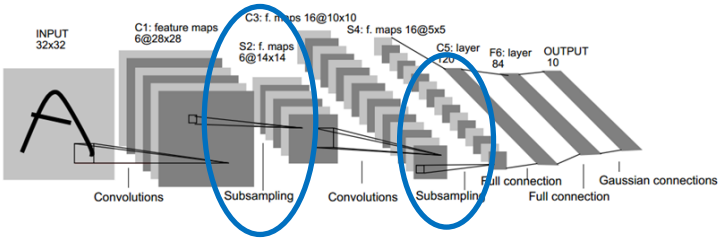
Zero-padding
フィルタがはみ出す分をゼロ埋め
$(F-1)/2$



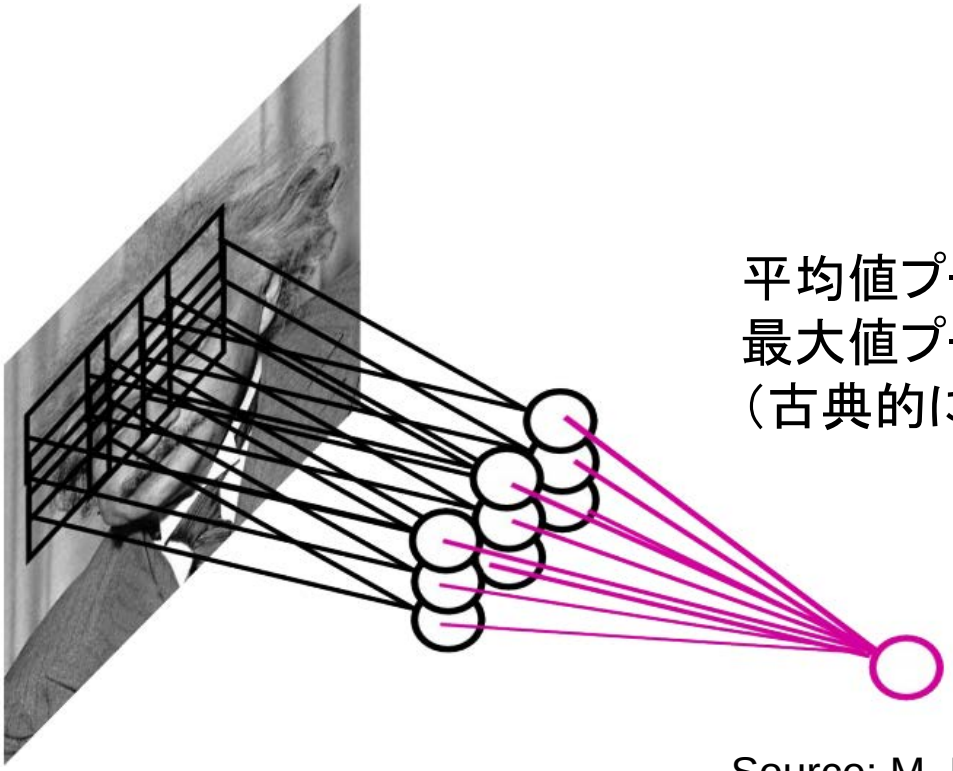
このあと活性化関数がかかる

入力: チャネル数 3
サイズ N=7 (padding込)
フィルタ: 出力チャネル数 2
サイズ F=3
stride S=2

プーリング層



- 一定領域内の畳み込みフィルタの反応をまとめる
  - 領域内での平行移動不変性を獲得

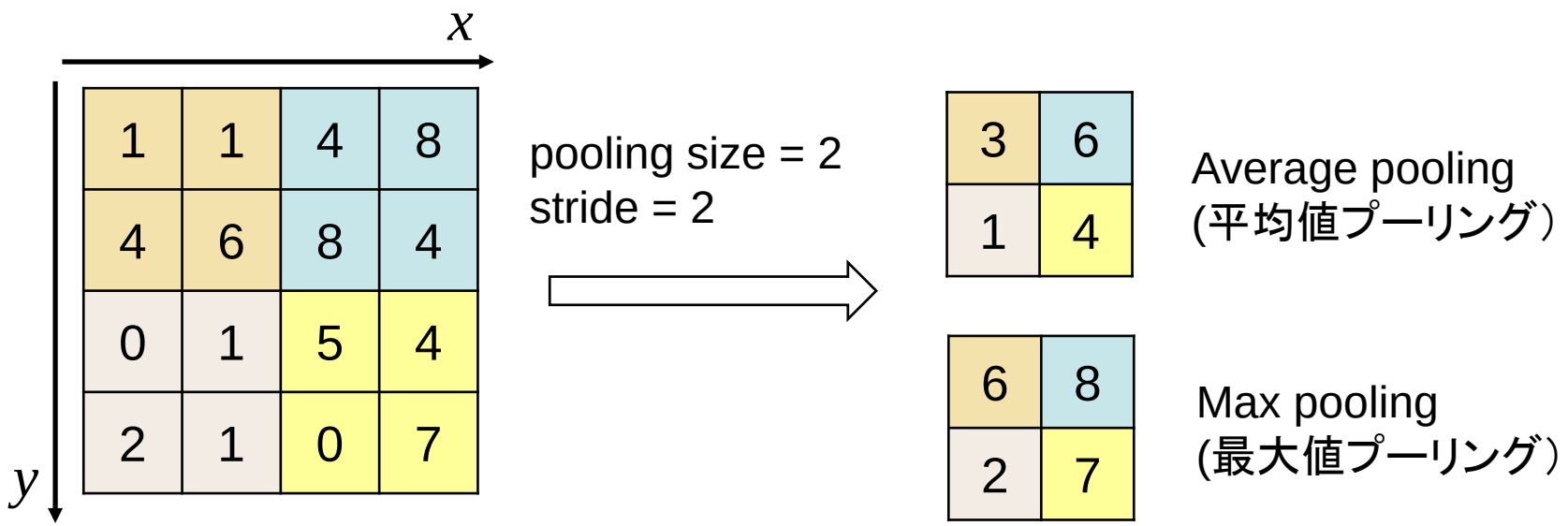


平均値プーリング、
最大値プーリングなど
（古典的には単純なサンプリング）

Source: M. Ranzato, CVPR'14 tutorial slides

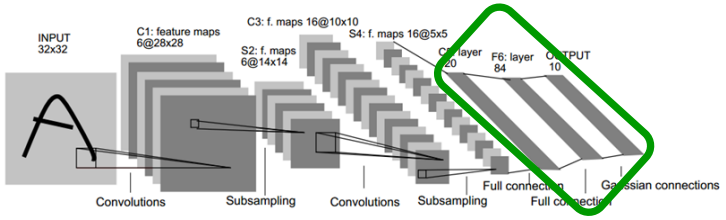
プーリング層

- Average pooling: 局所領域の平均値をとる
- Max pooling: 局所領域の最大値をとる

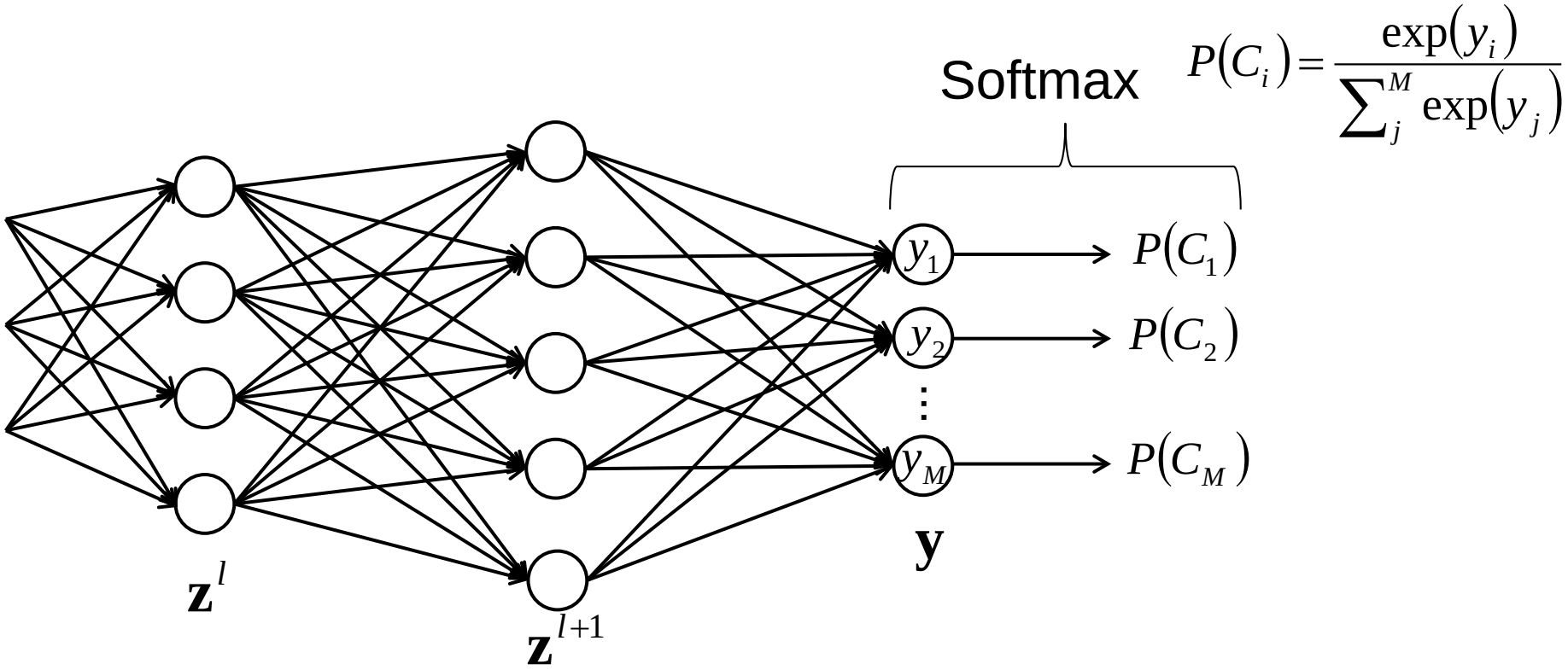


- 他にも Lp pooling, stochastic pooling などいろいろ

全結合層・出力層



- 要するにただの多層パーセプトロンです
  (前回資料参照)



Backprop: 畳み込み層

- 全受容野での誤差を束ねて更新

$z_{k,i}^{l} = h\left(u_{k,i}^{l}\right)$: k番目の需要野のi番目の出力値 (第l層)

$\delta_{k,i}^{l}$: k番目の需要野のi番目の誤差値 (第l層)

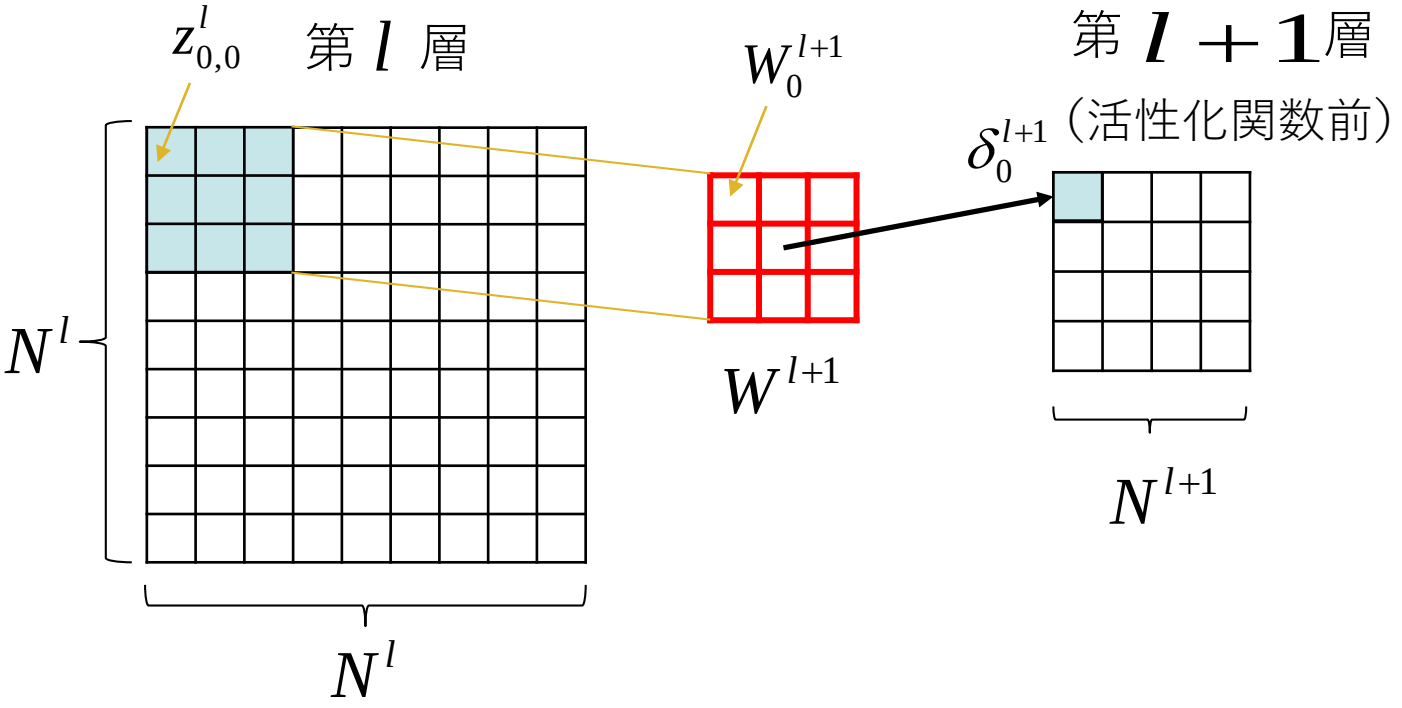
δ_{k}^{l+1} : 第$l+1$層の対応する場所の誤差

W_{i}^{l+1} : フィルタのi番目の係数

$$\frac{\partial J}{\partial W_{i}^{l+1}} = \sum_{k=0}^{\left(N^{l+1}\right)^2-1} \delta_{k}^{l+1} z_{k,i}^{l}$$

$$\delta_{k,i}^{l} = h'\left(u_{k,i}^{l}\right) \sum_{j}^{N^{l+1}} \delta_{j}^{l+1} W_{i'}^{l+1}$$

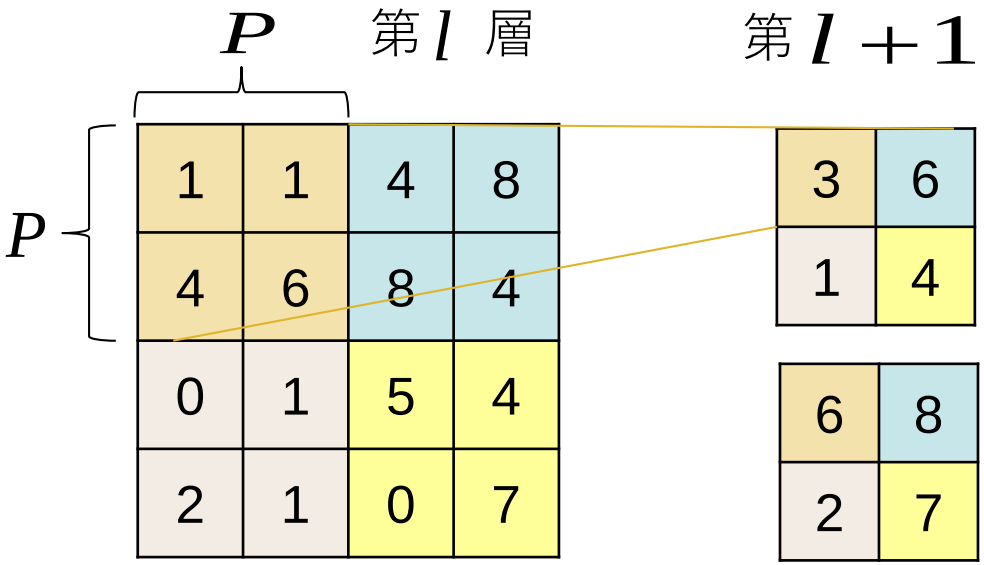
結合がある部分のフィルタ係数



Backprop: プーリング層

- Average pooling の場合は簡単
- Max pooling の場合、feedforward時に選ばれたユニットにのみ誤差が伝播する（覚えておく必要がある）

$$\delta_{k,i}^{l} = \sum_{i}^{P^2} \delta_{k}^{l+1} W_{i}^{l+1}$$



Average pooling
(平均値プーリング)

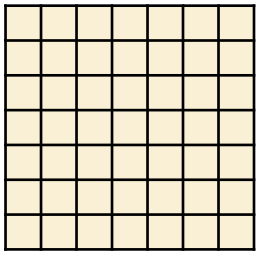
$$\delta_{k,i}^{l} = \delta_{k}^{l+1} / P^2$$

Max pooling
(最大値プーリング)

$$\delta_{k,i}^{l} = \begin{cases} \delta_{k}^{k+1} & \text{if } i = \arg\max_{j}\left(z_{k,j}^{l}\right) \\ 0 & \text{otherwise.} \end{cases}$$

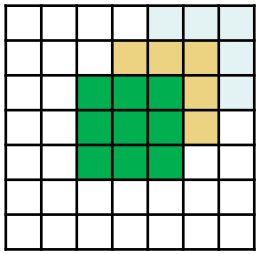
例）畳み込み層

- 7 x 7 の畳み込みは、3 x 3の畳み込み層を3つ積めば意味的に等価



$7 \times 7 = 49$

≒



$3 \times (3 \times 3) = 27$

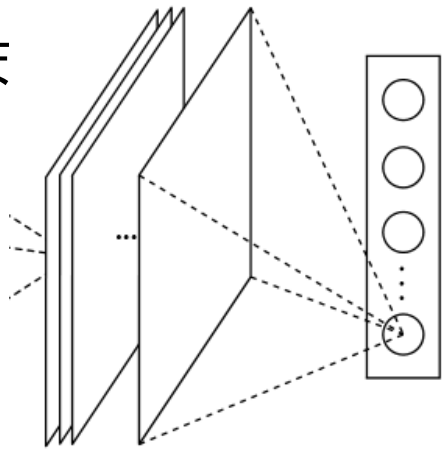
より少ないパラメータで、
より深い非線形性！

- 現在は、3 x 3や1 x 1の小さな畳み込み層をたくさん積むのが基本
  - 更に、3x3を3x1と1x3にばらす(factorization)することも…

全結合層はいらない？

- CNNのパラメータの大半は全結合層に集中
  - あくまで一層内の線形結合。非線形性は増えない。
  - ないよりはあった方がよいが、割に合わない

- 最近のCNNの多くは全結合層を持たない
  - Global average pooling：最終層の平均値プーリングをとり、
    そのまま



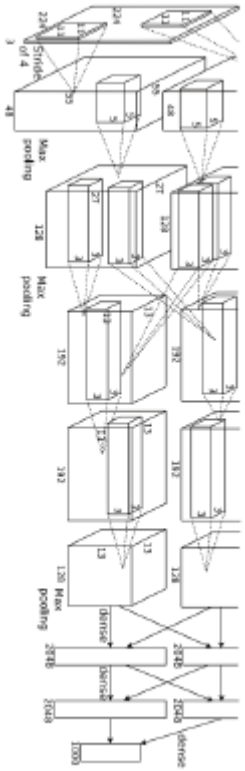
Min Lin et al., “Network In Network”, In Proc. ICLR, 2014.

更に深く、広く…

2015 MSRA
(152層)

- 2012年以降劇的な向上が続いてきた

2012 AlexNet
(8層)



2014 VGG
(19層)

2014 GoogLeNet
(22層)

深いNNの学習を支える技術

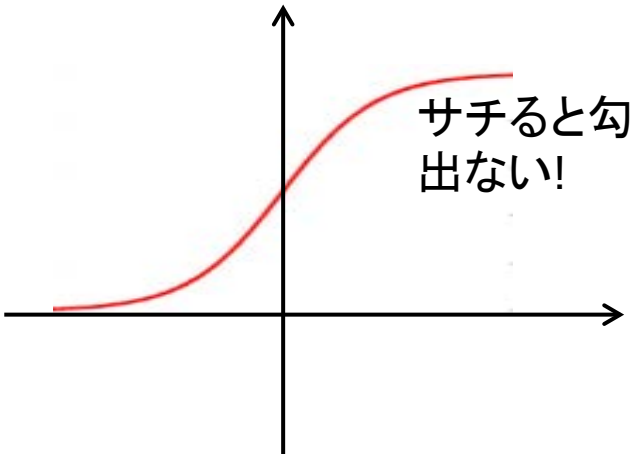
- ネットワークの構成要素
  - 活性化関数：ReLU
  - 過学習抑制手法：Dropout
  - バッチ正規化
  - ネットワーク初期化法
  - Residual learning
- 最適化手法
  - SGD、Adadelta、Adam
- 画像認識特有の工夫
  - 画像の前処理
  - データ拡張

~10層

20~30層

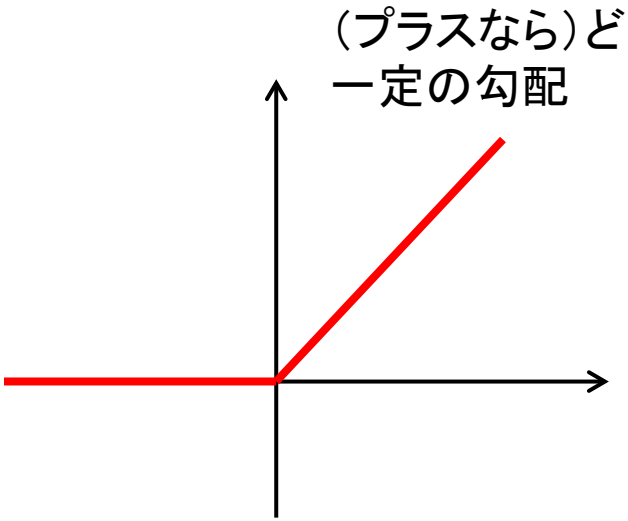
→ 100~1000層

Rectified Linear Units (ReLU)



$$\frac{1}{1+\exp(-x)}$$

シグモイド関数



$$\max(0, x)$$

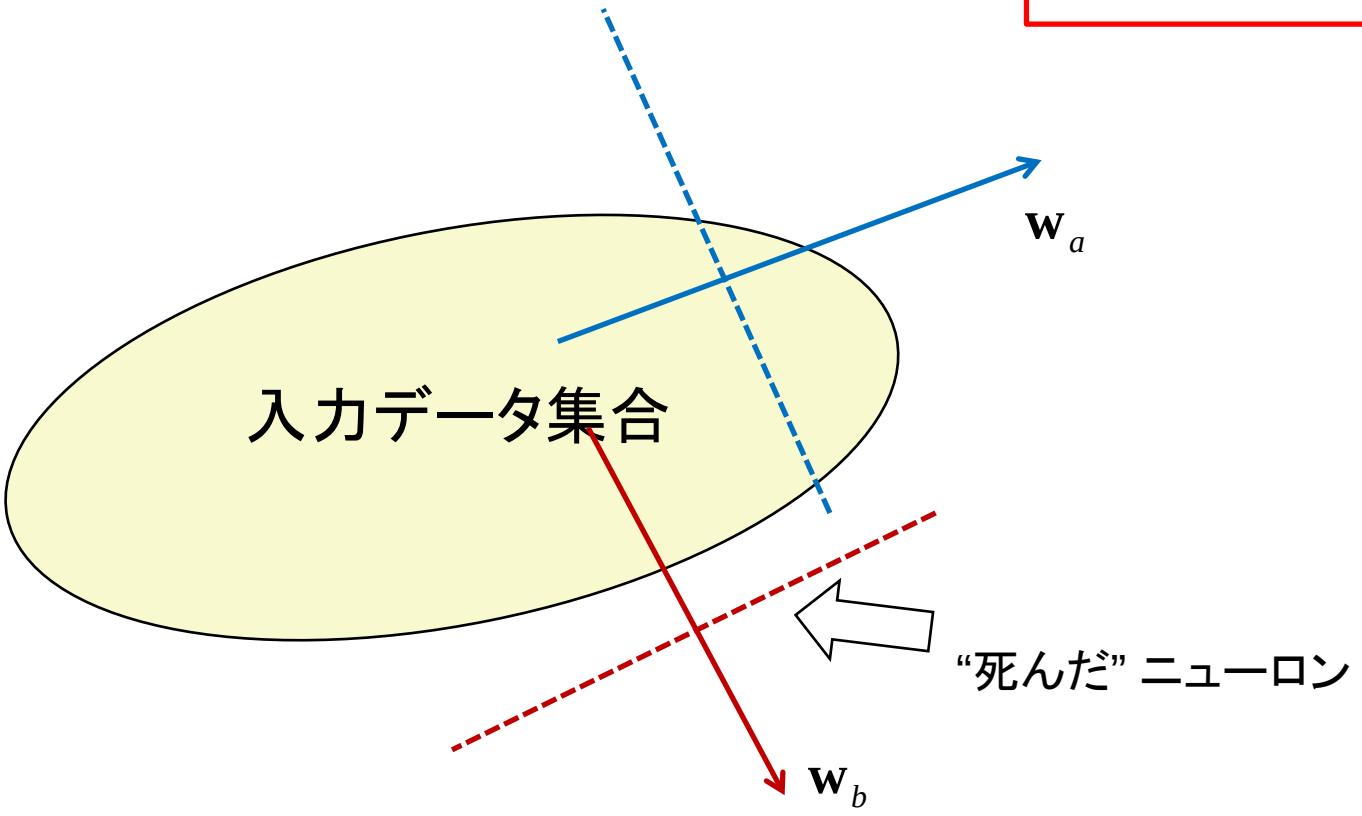
Rectified linear units (ReLU)
[Nair & Hinton, 2010]

例)単純パーセプトロン の更新式 $\mathbf{w}_{new} = \mathbf{w}_{old} + \eta(y_i - \hat{y}_i) f'(\mathbf{w}_{old}^T \mathbf{x}_i - \theta_{old}) \mathbf{x}_i$

ReLUの弱点

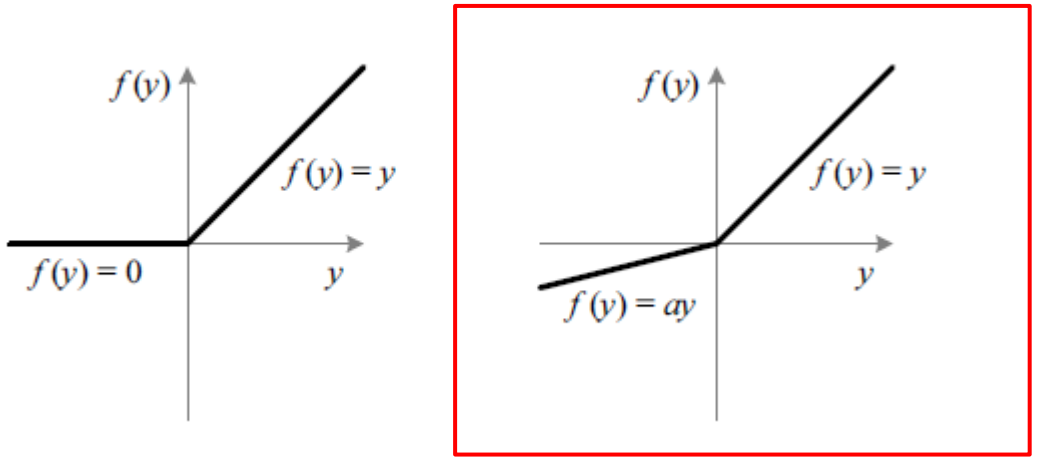
- すべての入力データが負になると、常に微分がゼロとなる
  - → パラメータが二度と更新されなくなる

$$\mathbf{w}_{new} = \mathbf{w}_{old} + \eta\left(y_i - \hat{y}_i\right) f'\left(\mathbf{w}_{old}^T \mathbf{x}_i - \theta_{old}\right)\mathbf{x}_i$$



Leaky ReLU

- 負の側にも少し勾配を与えたReLU



- MSR (2015)
  - PReLU: 負側の勾配係数もパラメータの一つとしてチューニング
  - ILSVRC'2014 のデータセットで 4.94% error

He et al., "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification", arXiv preprint, 2015.

Xu et al., "Empirical Evaluation of Rectified Activations in Convolution", arXiv preprint, 2015.

ReLUの発展形

Table 3: Non-linearities tested.

Name	Formula	Year
none	$y = x$	-
sigmoid	$y = \frac{1}{1+e^{-x}}$	1986
tanh	$y = \frac{e^{2x}-1}{e^{2x}+1}$	1986
ReLU	$y = \max(x, 0)$	2010
(centered) SoftPlus	$y = \ln(e^x + 1) - \ln 2$	2011
LReLU	$y = \max(x, \alpha x), \alpha \approx 0.01$	2011
maxout	$y = \max(W_1 x + b_1, W_2 x + b_2)$	2013
APL	$y = \max(x,0) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s)$	2014
VLReLU	$y = \max(x, \alpha x), \alpha \in 0.1, 0.5$	2014
RReLU	$y = \max(x, \alpha x), \alpha = \text{random}(0.1, 0.5)$	2015
PReLU	$y = \max(x, \alpha x), \alpha$ is learnable	2015
ELU	$y = x$, if $x \geq 0$, else $\alpha(e^x - 1)$	2015

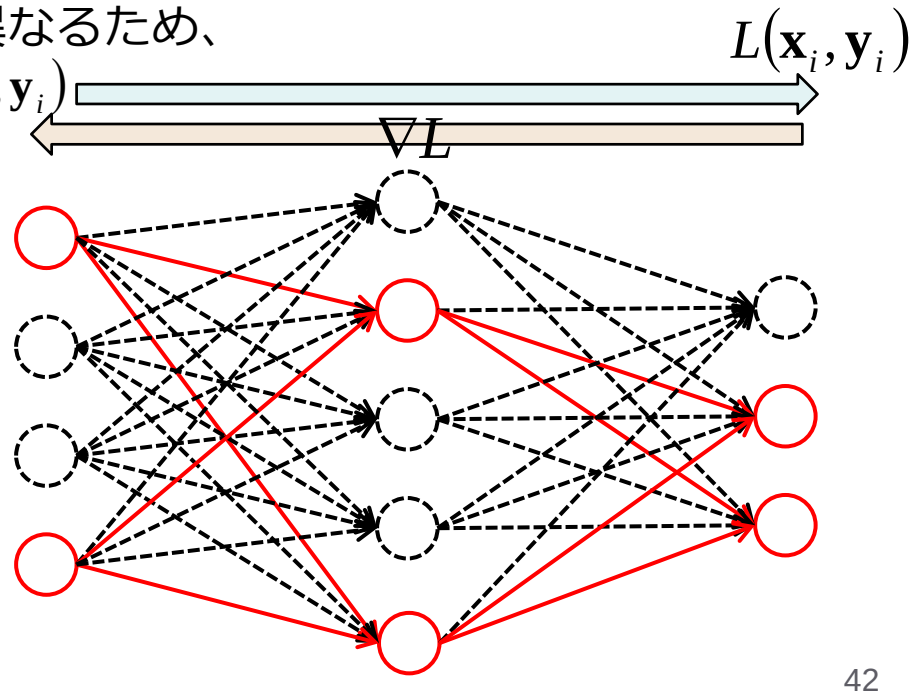
Mishkin et al., “Systematic evaluation of CNN advances on the ImageNet”, arXiv:1606.02228v1, 2016.

Dropout [Hinton, 2012]

- 各訓練データのフィードバックの際に、一定確率(0.5)で中間ニューロン（ユニット）を無視
- テスト時は全ニューロンを使うが、結合重みを半分にする

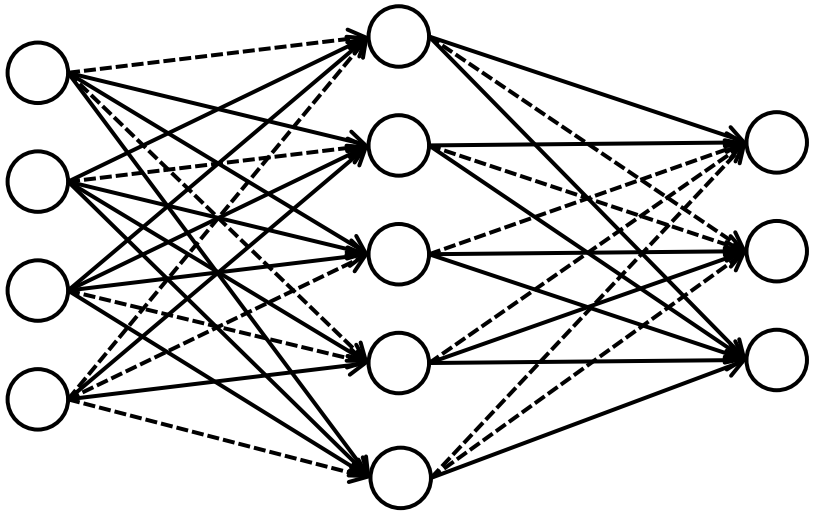
- 多数のネットワークを混ぜた構造
  - 訓練データが各ニューロンごとに異なるため、バギングと同様の効果（ただしパラメータは共有）

- L2正則化に近い効果
  - [Wager et al., NIPS'13]



亜種

- Dropconnect [Wan et al., ICML'13]
  - ニューロンではなく、結合をランダムに落とす
  - Dropoutよりよいらしい？



- Standout [Ba et al., NIPS'13]
  - Dropoutで落とすニューロンをランダムでなく適応的に選択する

Batch normalization

- 各層で、ミニバッチごとに入力を正規化
  - 低層の変化に伴う入力の共変量シフトに追従
  - 学習を約14倍高速化、精度向上 (特に20層以上の多層モデルで効果を発揮)

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

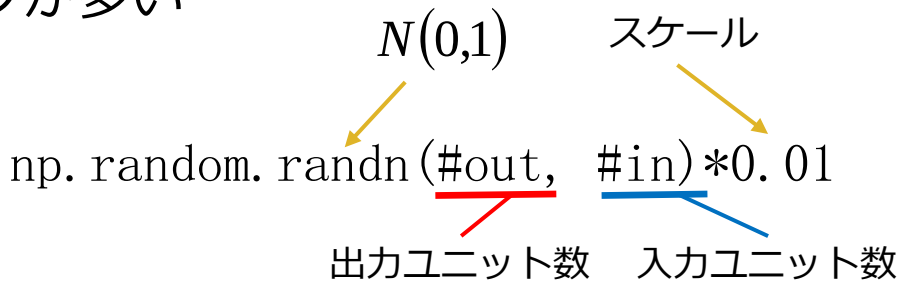
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

学習

Ioffe and Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, In Proc. ICML, 2015.

ネットワーク重みの初期化

- 標準正規分布からのサンプリングが多い
- 基本形
  - 小さいネットワークならOK
  - スケールパラメータの調整が難しい



- Xavier Glorot and Yoshua Bengio (2010)：*Xavier initialization*
  - tanh関数による活性が前提
  - ReLUのような非対称な関数では前提が成立しない

```
np.random.randn(#out, #in)/np.sqrt(#in)
```

- He et al., (2015)

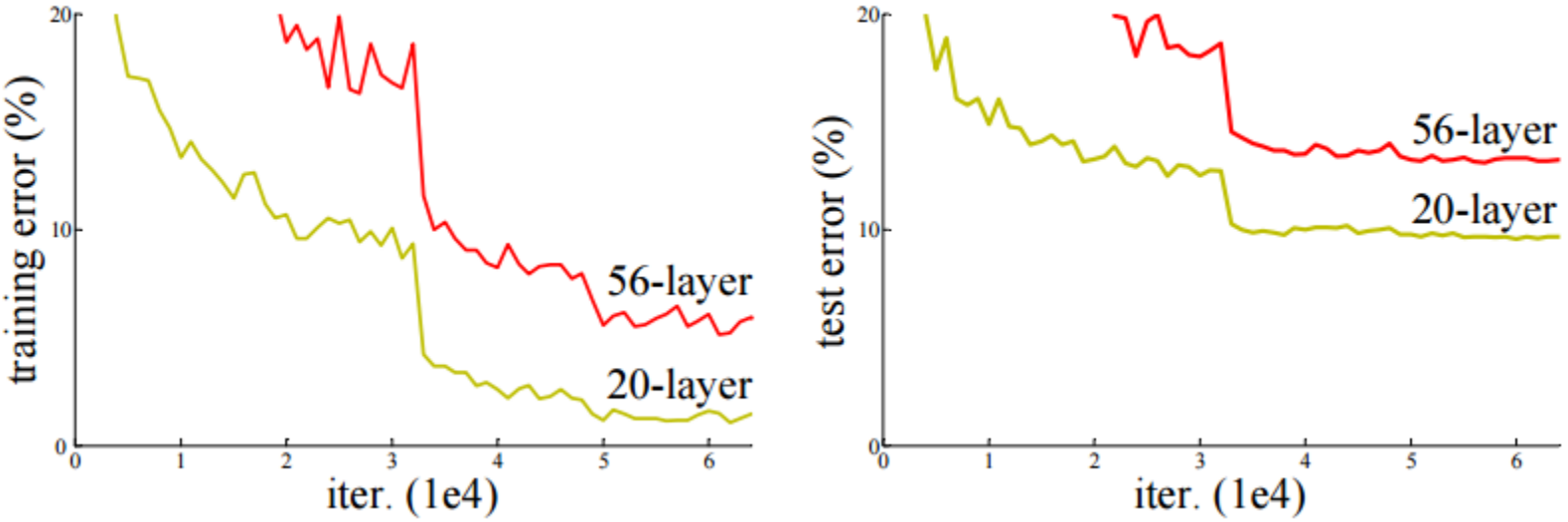
```
np.random.randn(#out, #in)/np.sqrt(#in/2)
```

  - 30層以上のネットワークではXavierと比較して顕著な優位性
- その他も研究多数（重要なトピックの一つ）
  - C.f. Mishkin and Matas, “All you need is a good init”, ICLR 2016.

クイズ

- ネットワークを大きくすればするほど…
  - 訓練誤差は、大きくなる？小さくなる？
  - テスト誤差は、大きくなる？小さくなる？

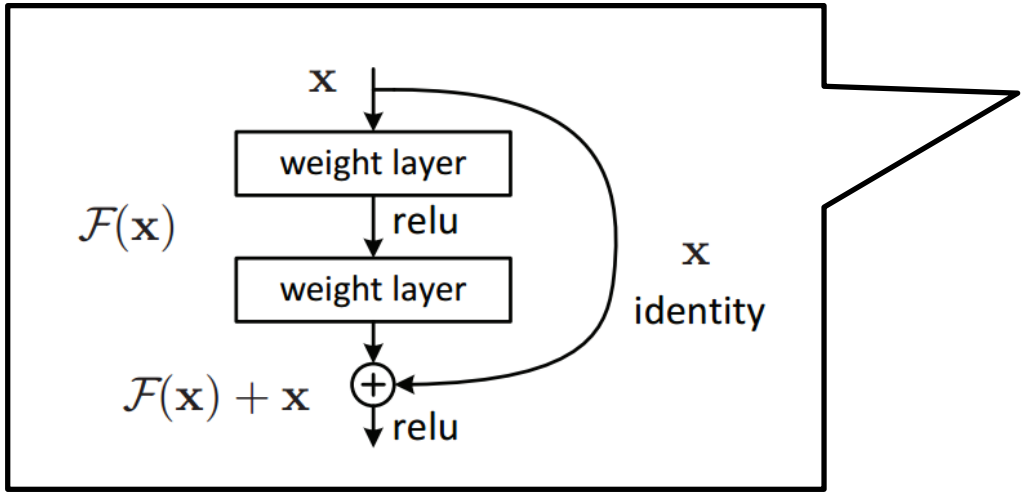
超多層ネットワークへ

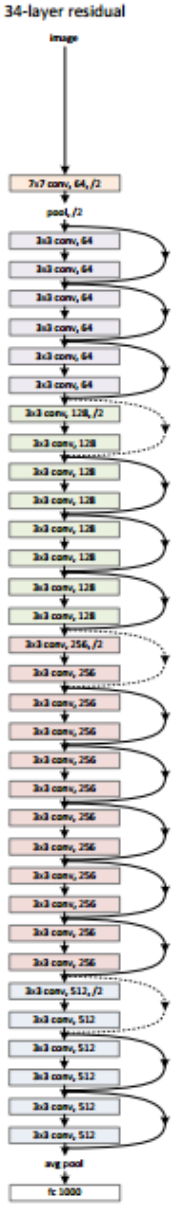


He et al., “Deep Residual Learning for Image Recognition ”, arXiv preprint, 2015.

- 超多層(50層以上)になると，訓練誤差もテスト誤差も大きくなる
  = アンダーフィッティング
- 低層のパラメータがほとんど更新されないので、結局学習が進まない

Deep Residual Learning (Hu et al., 2015)





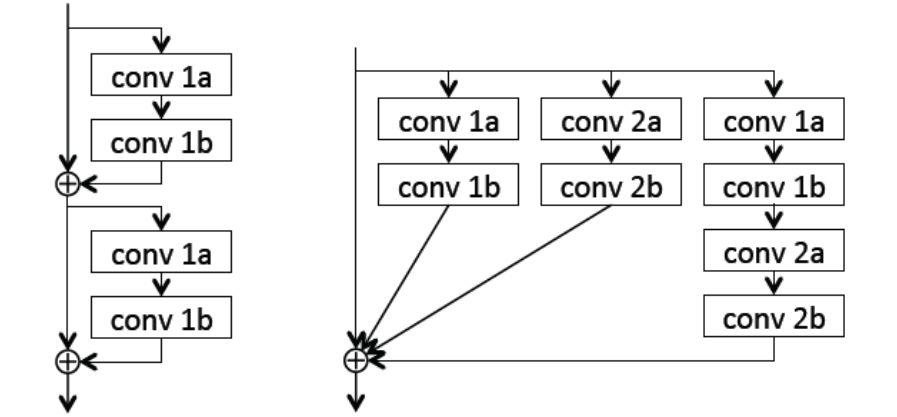
- 低層の入力をバイパスする構造を入れる
- 低層のパラメータの更新速度を速める
- 様々な深さのネットワークのアンサンブル

He et al., “Deep Residual Learning for Image Recognition ”, arXiv preprint, 2015.

Srivastava et al., “Highway Networks”, ICML 2015 deep learning workshop, 2015.

ResNetを展開すると…

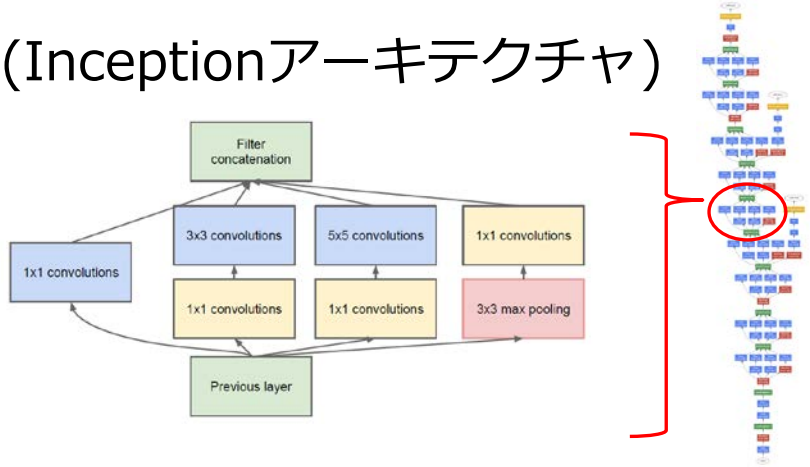
- ResNet構造は、さまざまな深さのサブネットワークのアンサンブルと等価（マルチパス）[Veit et al., 2016]



(a) 2 Residual Blocks (b) Unraveled Network of 2 Residual Blocks

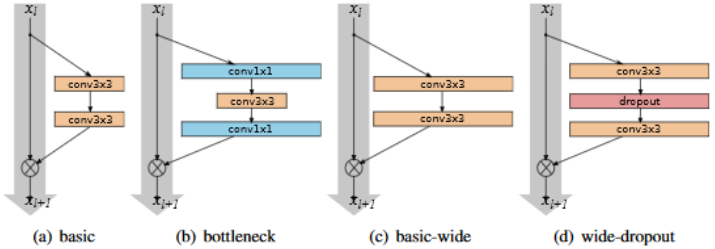
[武田, FIT'17より引用]

- 参考：GoogLeNet (Inceptionアーキテクチャ)

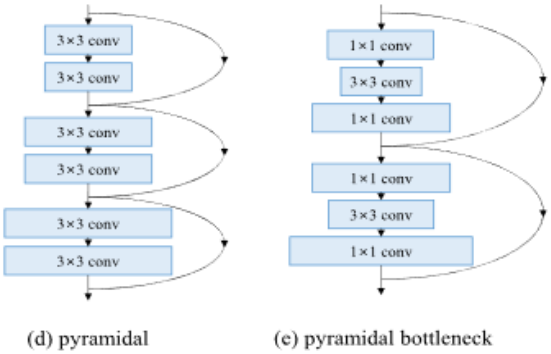


ResNet以降（１）

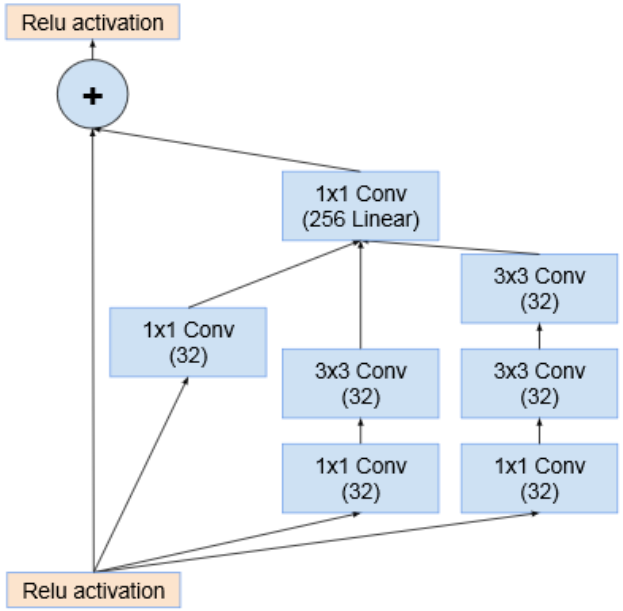
- 広さの拡張、マルチパス
  - 広さと深さのトレードオフ



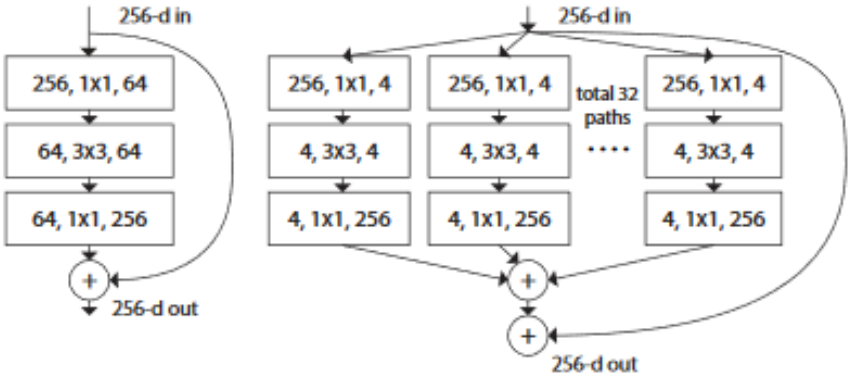
Wide ResNet [Zagoruyko+, 2016]



Pyramidal ResNet [Han+, 2016]



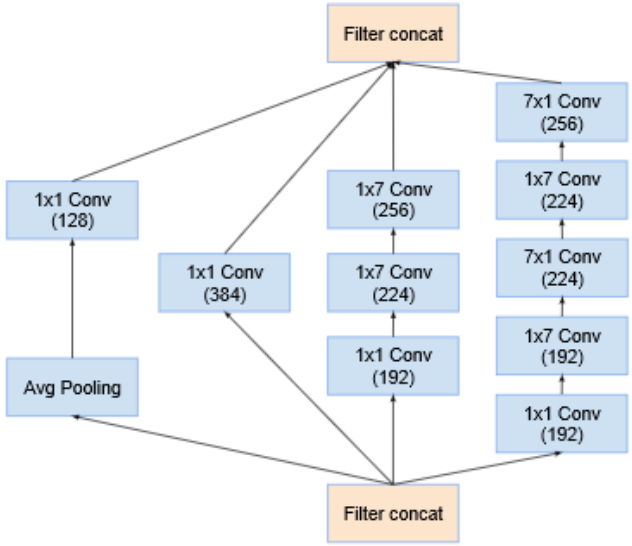
Inception ResNet
[Szegedy+, 2016]



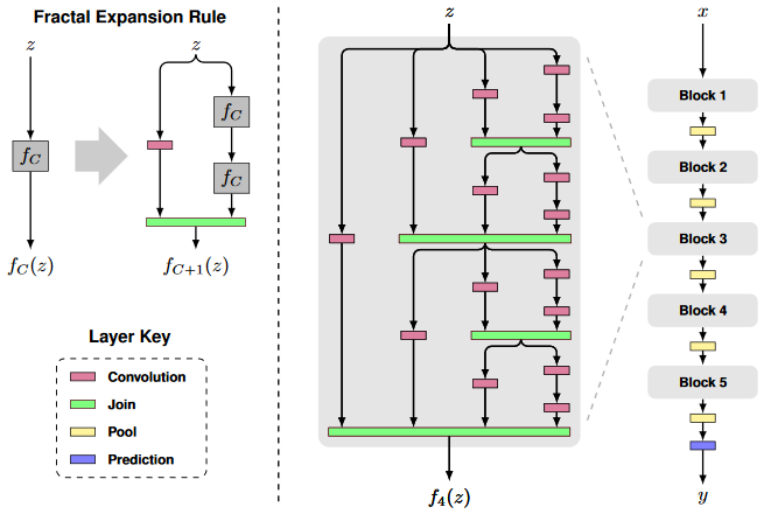
ResNeXt [Xie+, 2016]

ResNet以降（２）

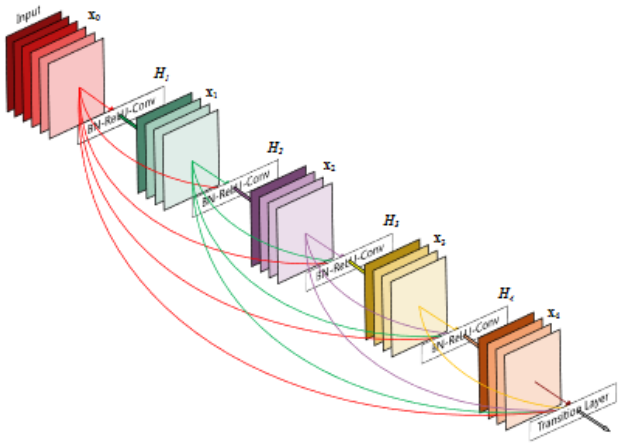
- ResNet亜種
  - Identity mappingでないskip connection
  - ネットワークつなぎ芸



Inception-v4 [Szegedy+, 2016]



FractalNet [Larsson+, 2017]



DenseNet [Huang+, 2016]

最適化手法の発達

- 確率的勾配降下法 (stochastic gradient descent)

  ※深層学習のために出てきたものではない

  - 1サンプルごとに目的関数の勾配を出し、重みを更新
  - 学習が圧倒的に高速化

  （注意）学習サンプルはシャッフルしておくこと

- 更新式
  - 最急降下法

$$\mathbf{w} \Leftarrow \mathbf{w} - \alpha \frac{\partial}{\partial \mathbf{w}} \left\{ \frac{1}{N} \sum_{i=1}^{N} \partial L\left(\mathbf{x}_i, y_i\right) \right\} \mathbf{x}_i$$

  - 確率的勾配降下法

$$\mathbf{w} \Leftarrow \mathbf{w} - \alpha \frac{\partial L\left(\mathbf{x}_i, y_i\right)}{\partial \mathbf{w}} \mathbf{x}_i$$

ミニバッチによるSGD

- ある程度サンプルを束ねて更新

$$\mathbf{w} \Leftarrow \mathbf{w} - \gamma \frac{\partial}{\partial \mathbf{w}} \left\{ \frac{1}{B} \sum_{i=1}^{B} L(\mathbf{x}_i, y_i) \right\}$$

- バッチ内のデータの評価は並列化可能
  - 一般にSGDの並列化は難しいが、GPUの実装法まで含めて研究が進められている
    Coates et al., “Deep learning with COTS HPC systems”, ICML’13

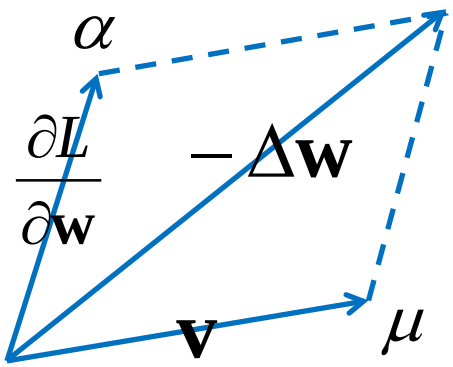
- 深層学習における損失関数は、しばしば鞍点(saddle point)やプラトー(plateau)が問題となる
  - どうやってそのような場所を抜けるか？

SGD + momentum

- 設定すべきハイパーパラメータ
  - 学習率、モメンタム、重み減衰率

$$\mathbf{v} \Leftarrow \mu\mathbf{v} + \alpha \frac{\partial L}{\partial \mathbf{w}}$$

$$\mathbf{w} \Leftarrow \mathbf{w} - \mathbf{v}$$



調整必須

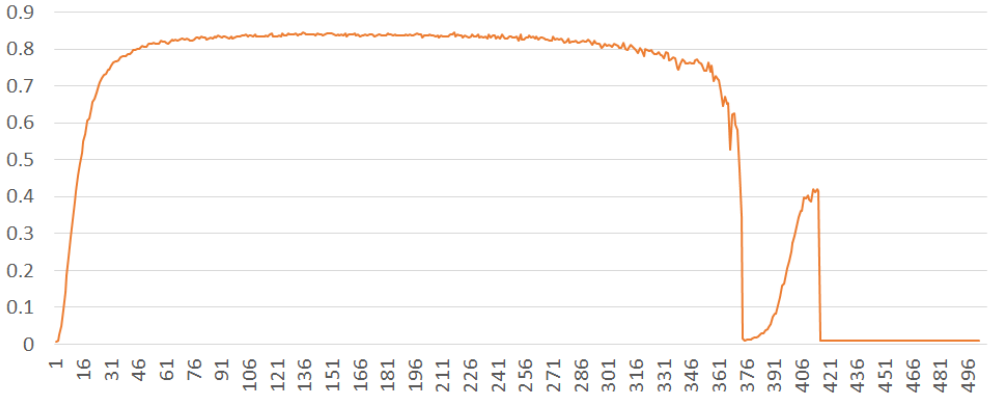
```
train_net: "lenet_t
base_lr: 0.01
momentum: 0.9
w          y: 0.000
m          0000
```

マジック
ナンバー？

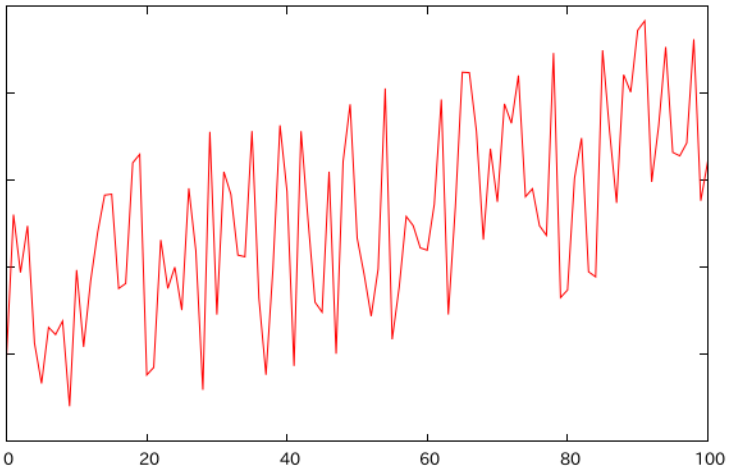
学習率の設定が一番重要

- しかし一筋縄にはいかない…

- 学習率が大きすぎる例
  - すぐ頭打ちになる
  - 途中で突然で破綻する
    ことも多い



- 学習率が小さすぎる例
  - おおむね線形に見える場合
  - 最終的にいいところまで行くが、
    時間がかかりすぎる



学習率のスケジューリング

- 時間の経過 (=学習の進行) に伴い、
  学習率を小さくしていく操作

- 例）cuda-convnet チュートリアル (Krizhevsky)
  - 0.001 (150エポック) → 0.0001 (10エポック) → 0.00001 (10エポック)
  - 精度向上が頭打ちになったら下げてみる？

- あるいは、単純に時間減衰させることもある
  - 1/t，exp(-t) など

他の勾配降下手法

- 学習率を自動的に調整する手法も有効
  （最適化の分野で盛んに研究されている）
  - AdaGrad [Duchi+, 2011]

$$r \Leftarrow r + \left|\frac{\partial L}{\partial \mathbf{w}}\right|^2 \qquad \mathbf{w} \Leftarrow \mathbf{w} - \frac{\alpha}{\sqrt{r} + \varepsilon}\frac{\partial L}{\partial \mathbf{w}}$$

  - Adam [Kingma+, 2015]

$$r \Leftarrow \gamma r + (1-\gamma)\left|\frac{\partial L}{\partial \mathbf{w}}\right|^2 \qquad \mathbf{w} \Leftarrow \mathbf{w} - \frac{\alpha}{\sqrt{\frac{r}{1-\gamma^t}} + \varepsilon}\frac{\mathbf{v}}{1-\beta^t}$$

$$\mathbf{v} \Leftarrow \beta \mathbf{v} + (1-\beta)\frac{\partial L}{\partial \mathbf{w}}$$

  - 他、AdaDelta、RMSProp等が有名
  - 多くの問題で実用的によい結果を得るが、
    しっかり学習率をスケジューリングされたSGDの方が優れているとされる

次回：1月10日（最終回）

- 今日の続き
- 自然言語処理応用
- 演習

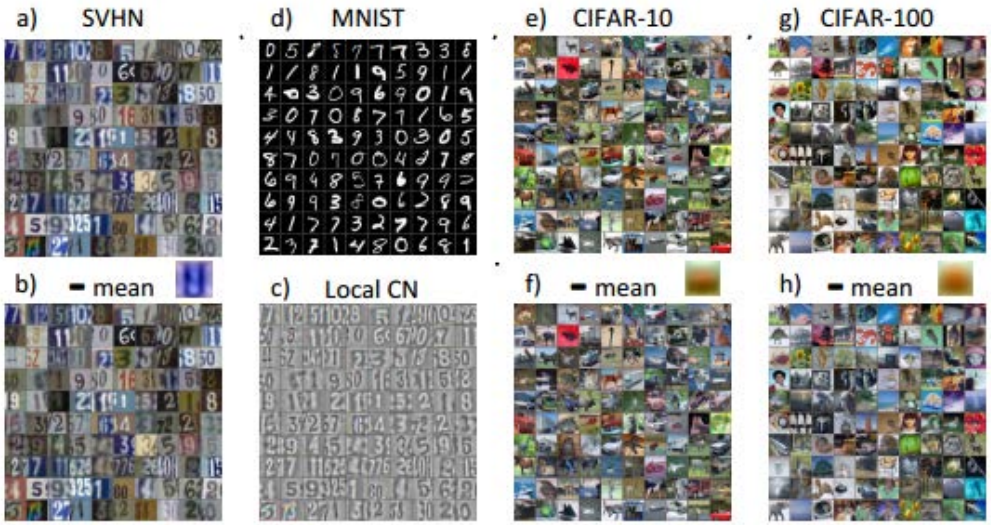
- 年末までに期末レポート課題を出します
  - 詳細は講義ページから

以下補足

入力画像の前処理

- 実用上かなり重要
  - ZCA whitening（白色化）
  - 平均、分散正規化　　…など

- 最終的な識別性能に大きく影響する
  - 低層の特徴抽出フィルタの感度を上げる



[Zeiler and Fergus, 2013]

ZCA白色化 (zero component analysis)

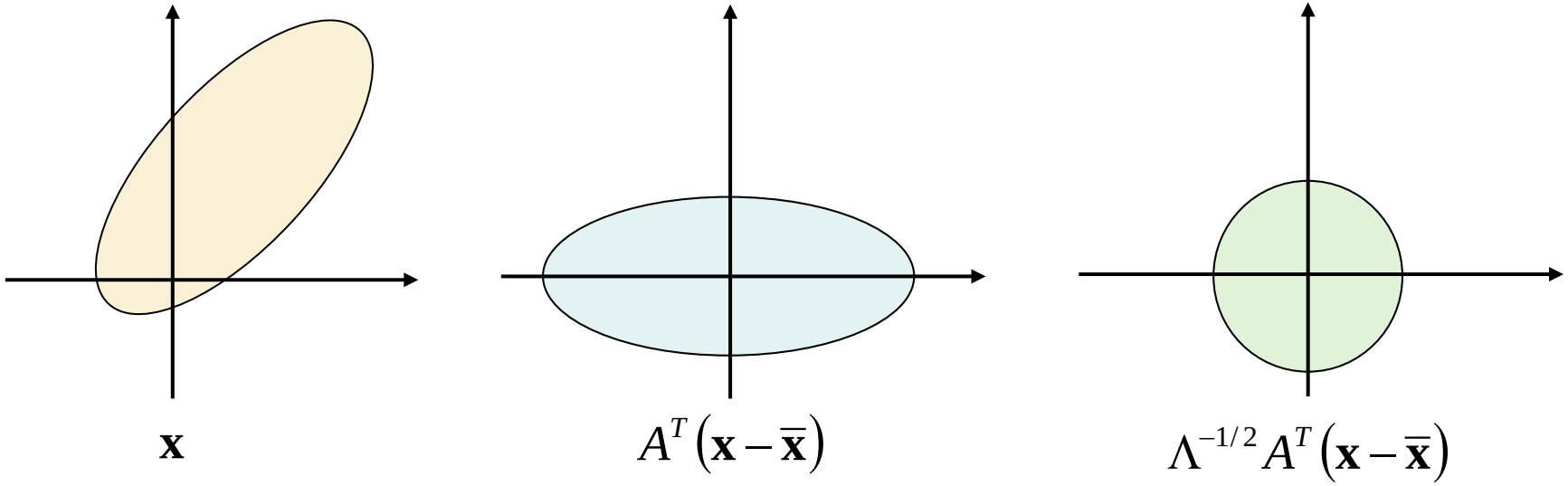
- PCAを利用して無相関化・分散正規化した後、元の空間に戻す

$$C_X \mathbf{a} = \lambda \mathbf{a}$$

$$\mathbf{x}^{ZCA} = A \Lambda^{-1/2} A^T \left(\mathbf{x} - \overline{\mathbf{x}} \right)$$

$$A = \left(\mathbf{a}_1 \, \mathbf{a}_2 \cdots \mathbf{a}_n \right) \quad \Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} + \varepsilon I$$

画素数次元の固有値問題＝大変。。



ZCA whitening

- 分散が小さい成分を拡大する効果
  ＝周波数成分が小さい要素（例えばエッジ）を強調

https://www.linkedin.com/pulse/machine-learning-diabetic-retinopathy-detection-alexander-rakhlin

グローバルコントラスト正規化

- Global Contrast Normalization (GCN)
  - 画像一枚ごとに行う前処理 （ZCAと違うので注意！）
  - チャネル(R,G,B) ごとに画素値を平均0、分散1にする
  - 画像の明るさのレベルを揃える効果

$$x^{Norm} = \frac{x - \bar{x}}{\sqrt{\sigma_x^2 + \varepsilon}}$$

- ZCAと合わせて使うのもよい (GCN→ZCA)

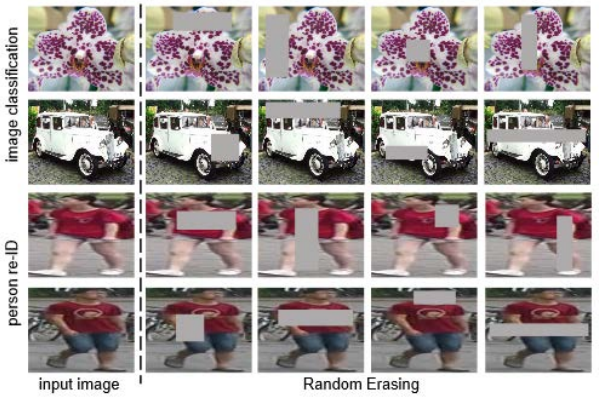
- ImageNetベースのものでは、データセット全体の平均ベクトルを引くだけの場合がほとんど
  - AlexNetなど

訓練データ拡張 (Data augmentation)

- アフィン変換、クロップ、コントラスト変化など、人工的にさまざまな変換を学習データに加える
  - 平行移動以外の不変性を学習させる
  - 何が効くかはタスク依存

[Dosovitskiy et al., 2014]

- Random erasing [Zhong et al., 2017]
  - ランダムに画像の一部を消去

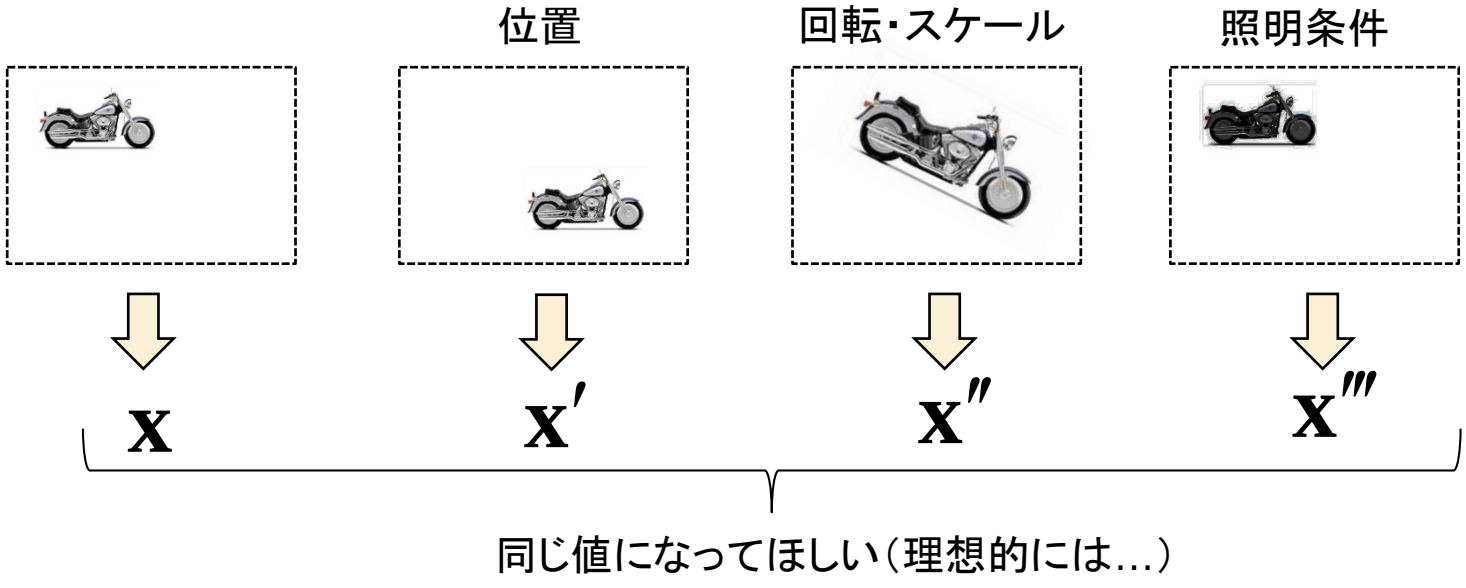


[Zhong et al., 2017]

参考：**Augmentor** [Marcus D. Bloice]
Image augmentation library in Python
https://github.com/mdbloice/Augmentor

不変構造　v.s. 訓練データ拡張

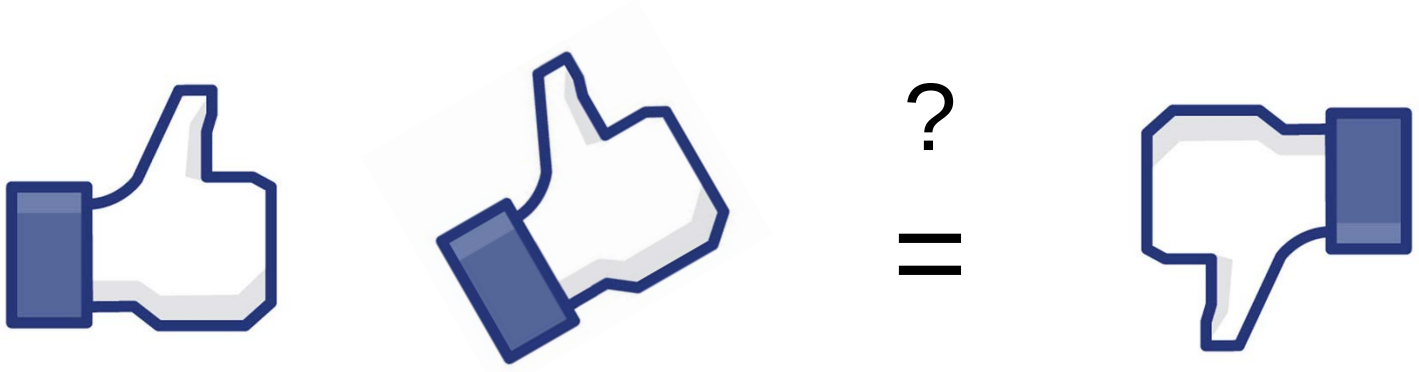
- よい特徴量（特徴表現）とは？
  - 多くの情報量を持つと同時に、入力の変動に対する不変性を実現する



- CNNはもともと平行移動不変性を持つ
  - 回転・スケールなど他の変動にも構造に不変性は入れられるだろうか？

    c.f. Dieleman et al., Rotation-invariant CNN, 2015.

不変性の程度はタスクしだい



- 今のところ、訓練データの方で不変性のさじ加減を調整することが有効 → データ拡張
- 普通のCNNも、いつもベストの構造とは限らない！
  - 画像中の位置が重要な場合など

訓練データ拡張の例

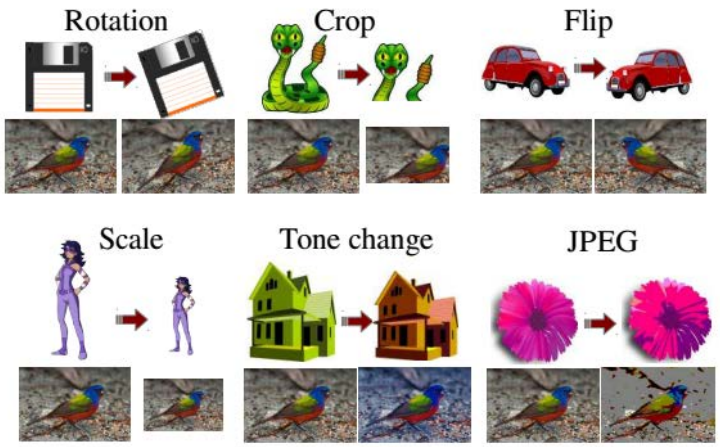


Figure 1: Illustrations of image transformations, on simple icons and on an image from the CUB dataset.

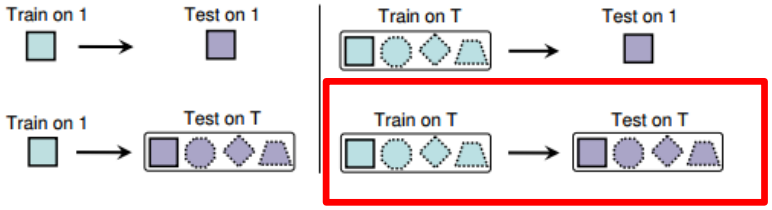


Figure 3: Illustration of the different training and test strategies. Both training and test can be done either on the original single image or on the set of all transformed images.

#transformations		CUB			ILSVRC-30		
#train	#test	SIFT	color	fusion	SIFT	color	fusion
1	1	16.5	23.5	28.2	34.1	28.9	40.4
1	T	17.8	24.9	28.8	36.2	30.6	42.5
T	1	20.0	26.4	31.9	38.7	31.5	44.4
T	T	**27.4**	**35.6**	**42.0**	**42.3**	**37.1**	**48.7**

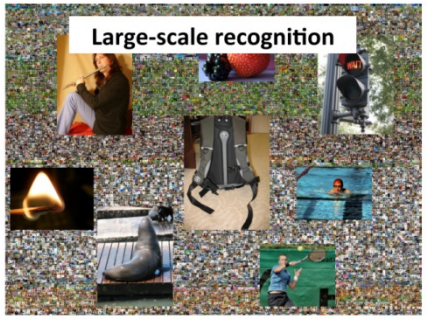
Table 1: Comparison of different training and test scenarios for $T = 6$, i.e. $T - 1$ corresponds to the number of transformations used for training and test. Score aggregation is performed by averaging.

- 訓練時だけでなく、テスト時にも同じ変動パターンを適用したデータを全て識別し、平均をとると精度向上
  - AlexNetは10パターンの平均

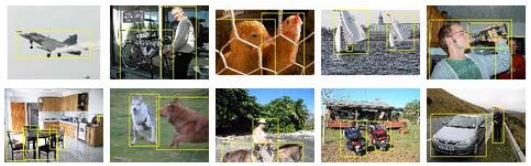
付録：外部大規模データを用いた転移学習

- 転移学習
  - 新規タスクの効果的な仮説を効率的に見つけ出すために，一つ以上の別のタスクで学習された知識を得て，それを適用する問題 [神嶌, 2010]

- 画像認識の場合
  - あるドメイン(データセット)で学習した識別器(特徴抽出器)を他ドメインでの識別器構築に役立てる



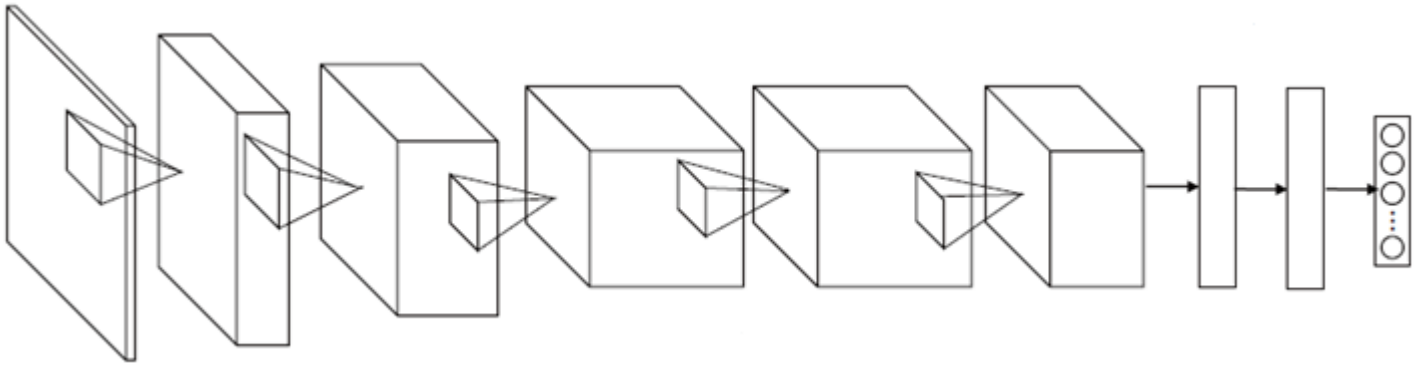
ImageNet ILSVRC'12
130万枚、1000クラス



PASCAL VOC 2007
5千枚、20クラス

CNNを用いた転移学習

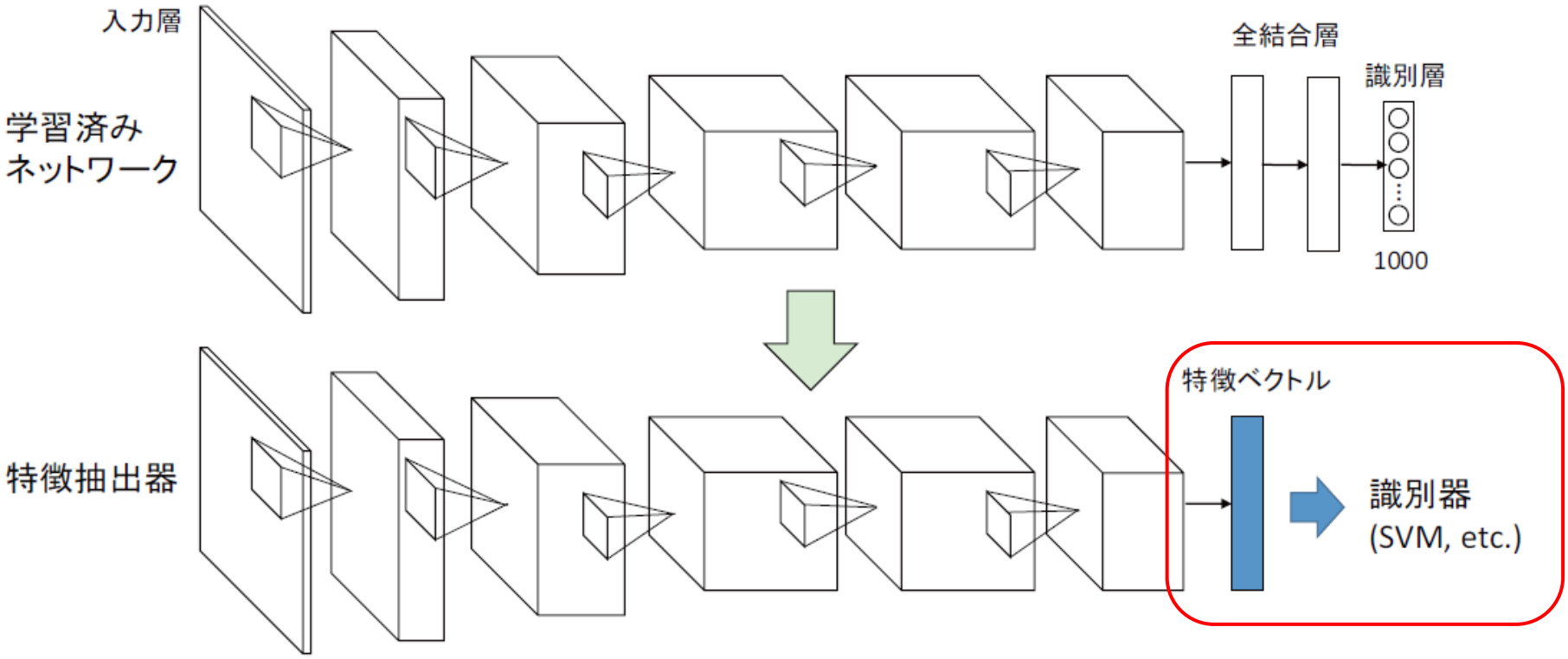
- 学習済ネットワークを転用
  - 転用先のタスクと何らかの関係がある(と期待できる)十分に大規模なデータセットで学習したネットワーク



- 大きく分けると二つのアプローチ
  - 特徴抽出器として利用 (Pre-trained feature)
  - Fine-tuning

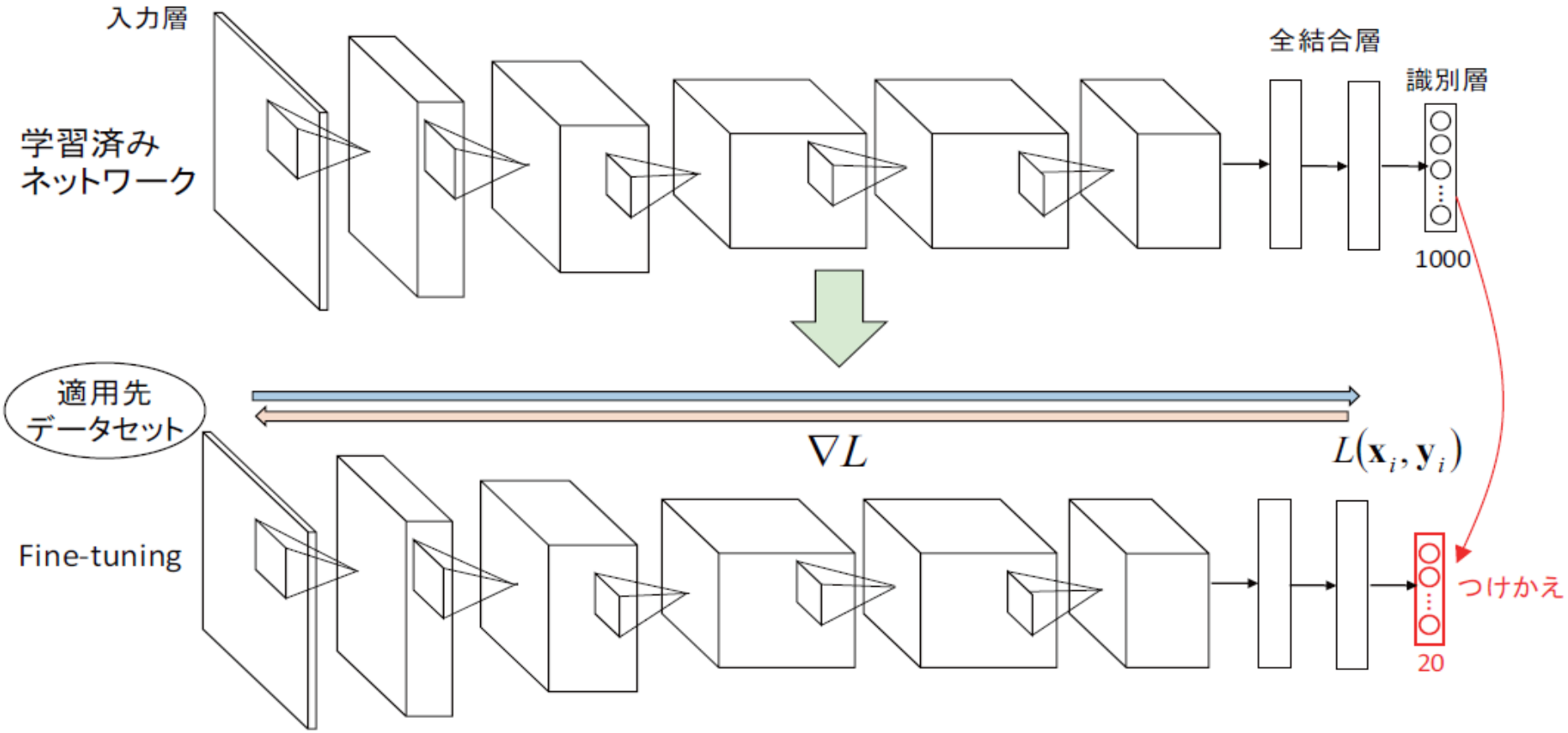
特徴抽出器として利用

- 学習済ネットワークを特徴抽出器として用いる
  - 中間層の出力を利用して識別器を構築
  - どの層を選ぶかは重要（タスク依存？）



Fine-tuning

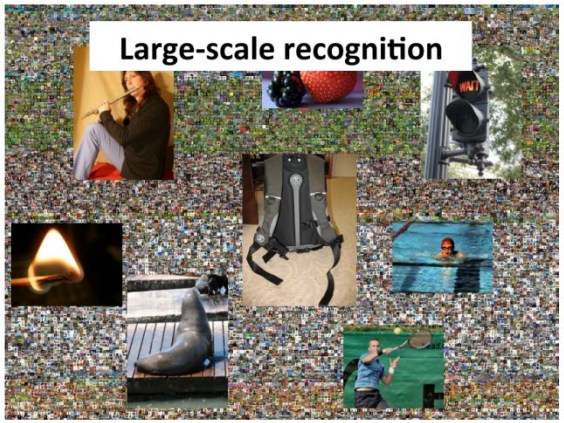
- 学習済ネットワークを初期値とし、適用先データセットで
  さらに学習を進める
- 教師なし事前学習とは異なる概念であることに注意



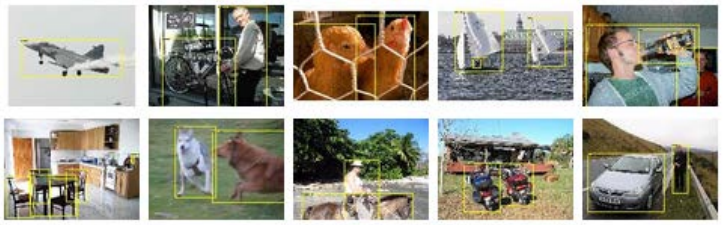
学習済みネットワークの効果

- ILSVRC 2012 → VOC 2007 の例 (検出成功率、mAP%)
  - フルスクラッチCNN: 40.7
  - Pre-trained feature: 45.5
  - Fine tuning: 54.1

Agrawal et al., “Analyzing the Performance of Multilayer Neural Networks for Object Recognition”, In Proc. ECCV, 2014.



ImageNet ILSVRC’12
130万枚、1000クラス



PASCAL VOC 2007
5千枚、20クラス

異なる学習済みネットワークの比較

- ILSVRC 2012 → VOC 2007 でfine-tuningをした場合の性能比較 (検出成功率、mAP%)
  - AlexNet : 58.5 ← ILSVRC'12 winner, エラー率16%
  - Small VGG: 60.2
  - VGG-16: 66.0 ← ILSVRC'14 二位, エラー率7.4%

Redmon et al., "You Only Look Once : Unied, Real-Time Object Detection", arXiv preprint, 2015.