

NICO2AI #7

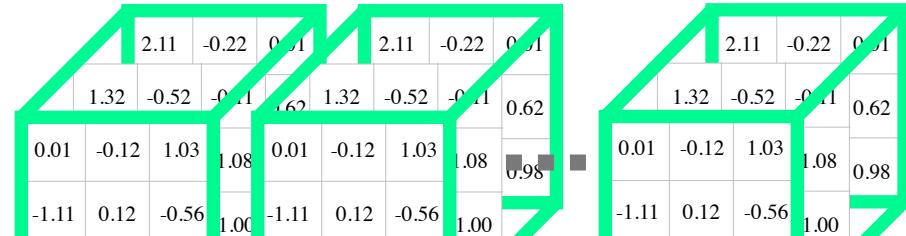
畳み込みニューラルネットワーク(2)

18/02/24
土屋祐一郎

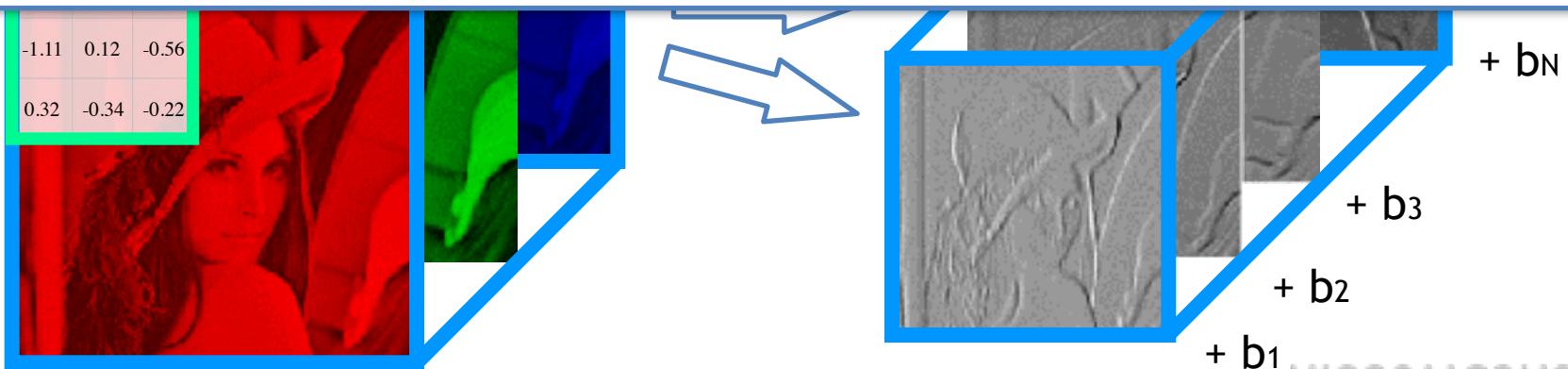
前回の復習

4D Convolution

適当に学習された3次元フィルタ×複数個



「3次元データに3次元フィルタを畳み込んで
2次元出力を得る処理」
を複数回行って、3次元出力を得る処理



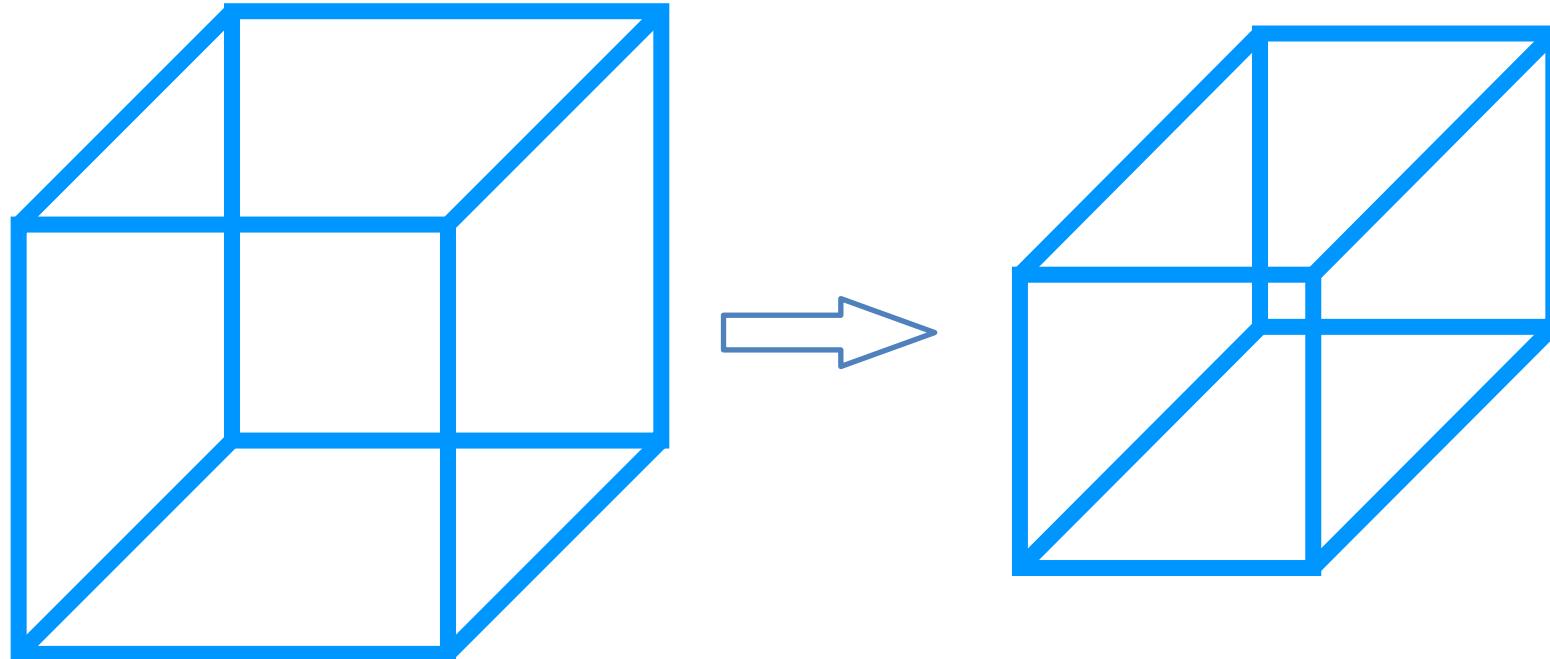
4D Convolution

簡略化して書くと...

「3次元データに4次元フィルタを畳み込んで
3次元出力を得る」
のがCNNの基本処理



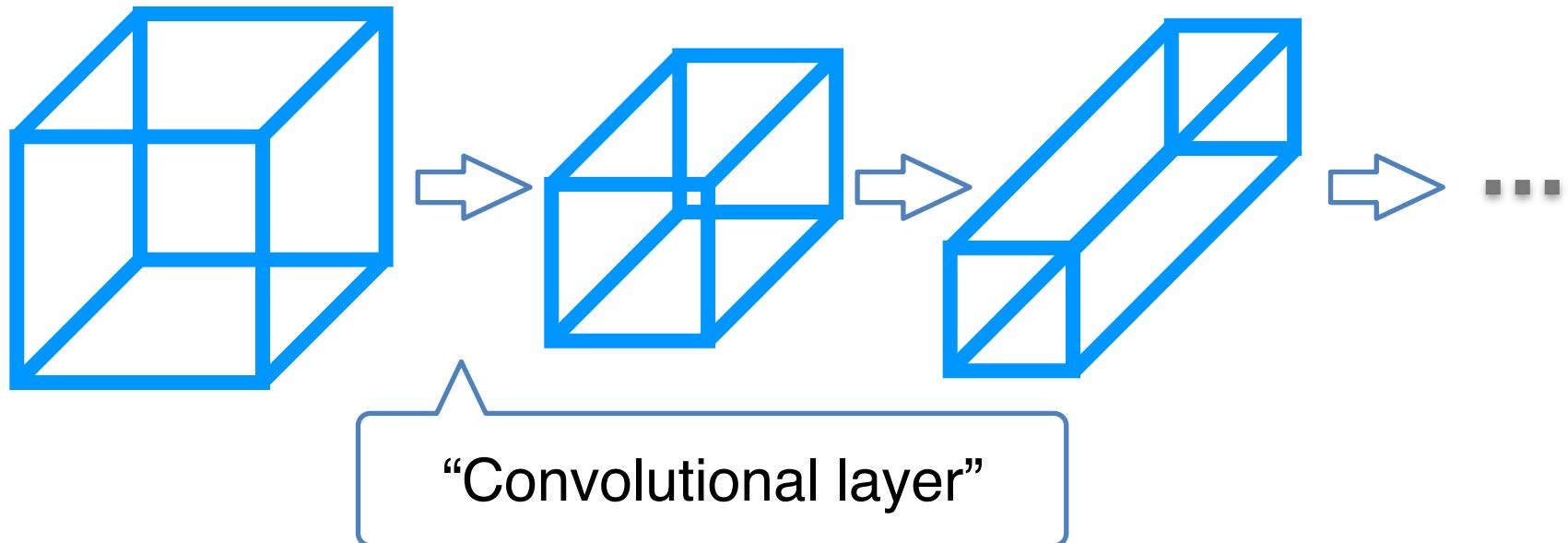
Convolutional Neural Networks



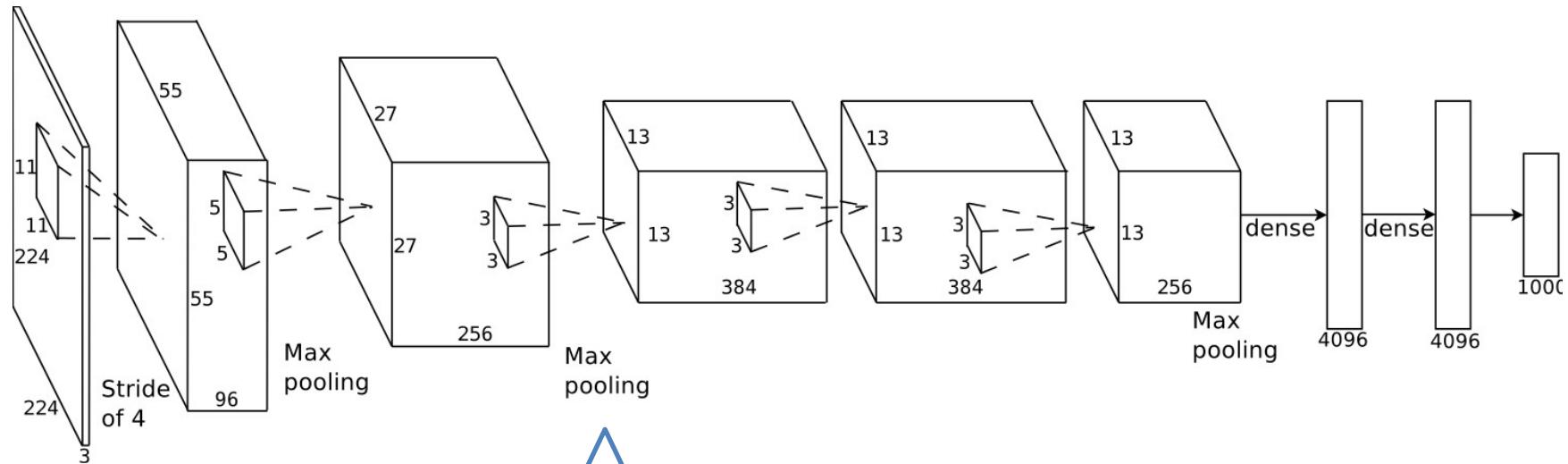
をたくさん繋げて...

Deep convolutional neural networks

6



Convolutional layer以外のlayer

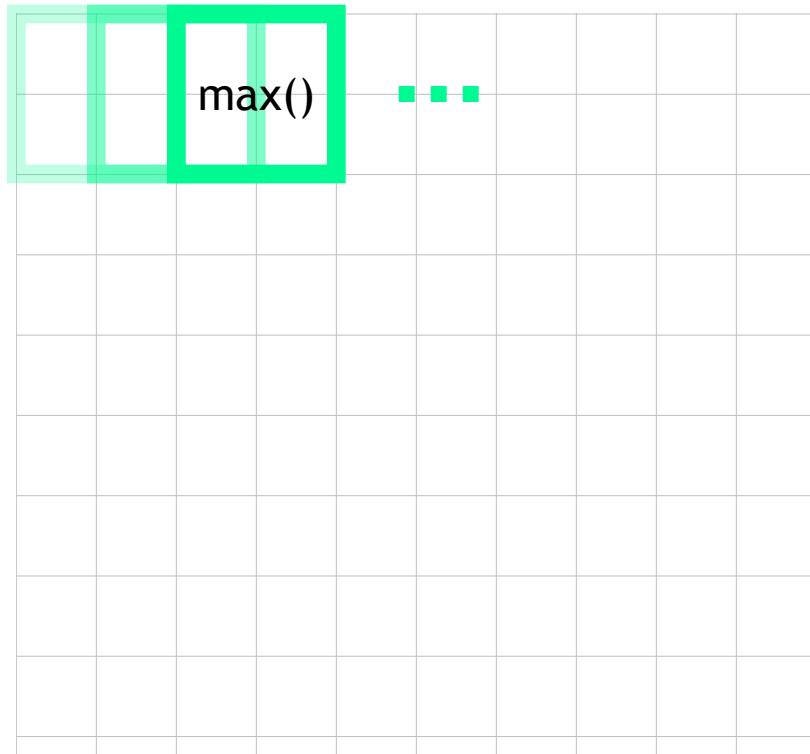


“Max pooling”

Max pooling

前の層にmax()フィルタをかける

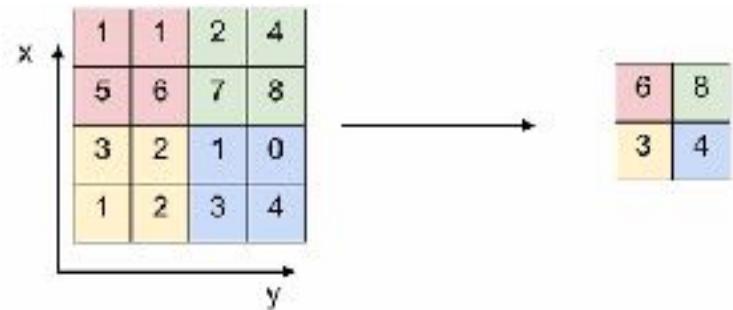
=適用範囲のうち、最大値のみをとり、残りを切り捨てる



ハイパーパラメータ：

- カーネルサイズk
- Stride s

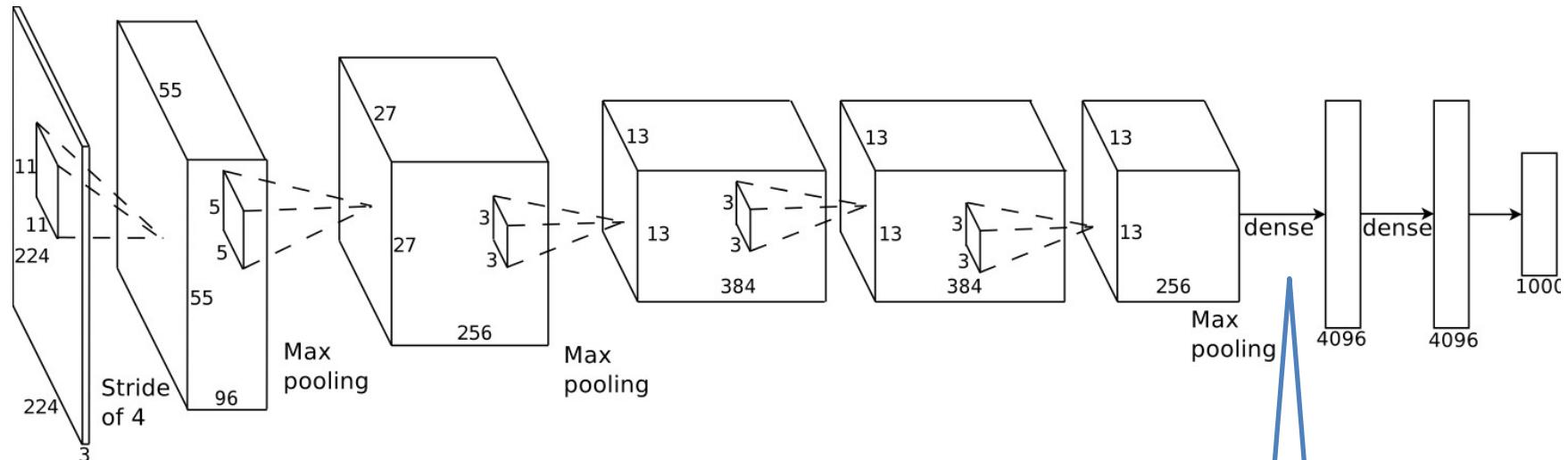
例: $k=2, s=2$



Pooling layer

- ▶ Max poolingの他にもいろいろ提案されている
 - ▶ Average pooling
- ▶ まとめてPooling layerと言ったりする

Convolutional layer以外のlayer



“Dense”

全結合層

- ▶ 普通の全結合NNをConvolutional Layerの後にくっつけることがよくある
- ▶ Dense connected layerと言ったりする
- ▶ Convolutional Layerの出力は(W, H, D)の3次元配列
→(W×H×D)の1次元配列にflattenして入力にする
(chainerだと気にしなくても良しなにやってくれる)

一般物体認識

一般物体認識

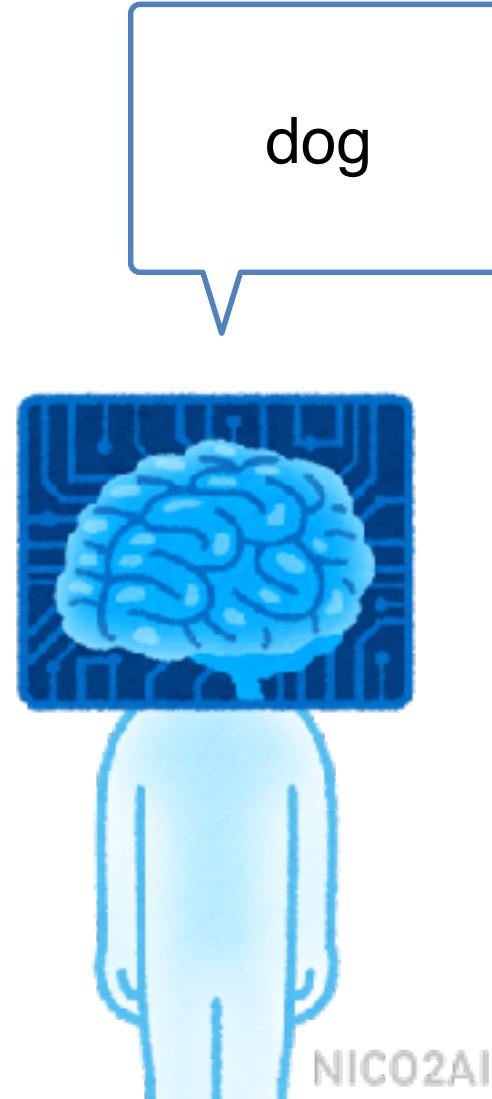
「一般物体認識」とは、制約のない一般的な実世界シーンの画像に対して、計算機がその中に含まれる物体もしくはシーンを一般的な名称で認識すること

[柳井, 2007]

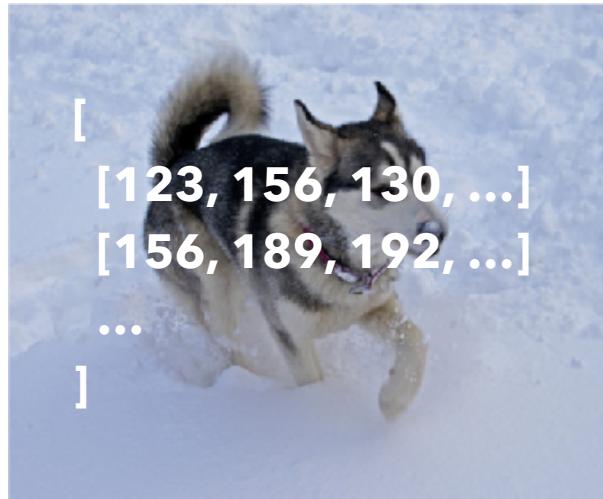
一般物体認識



dog

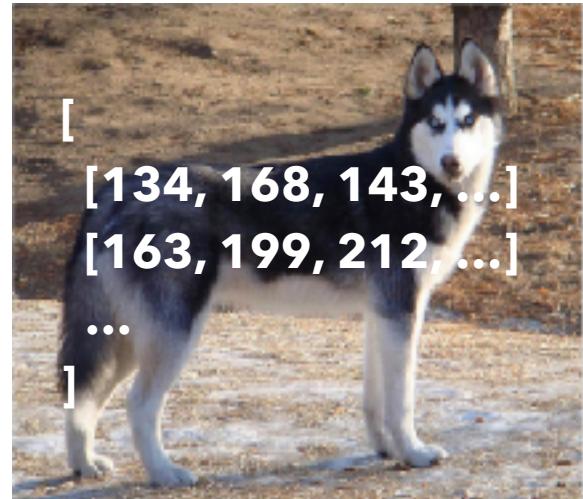


一般物体認識の難しさ



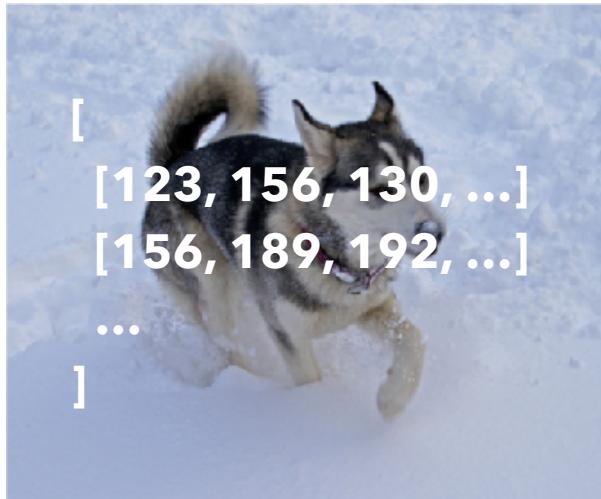
```
[  
[123, 156, 130, ...]  
[156, 189, 192, ...]  
...  
]
```

•≡•

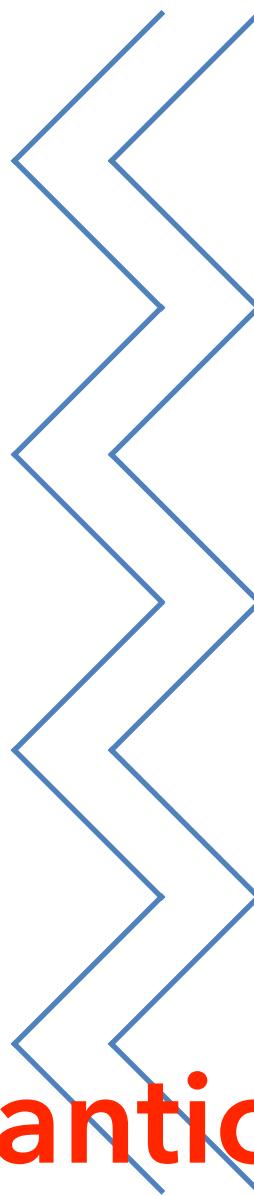


```
[  
[134, 168, 143, ...]  
[163, 199, 212, ...]  
...  
]
```

一般物体認識の難しさ



```
[  
  [123, 156, 130, ...]  
  [156, 189, 192, ...]  
  ...  
]
```



“dog”

Semantic gap

前Deep Learning時代の一般物体認識

17

局所特徴量

local feature descriptor

画像中の小領域を
特徴ベクトル化

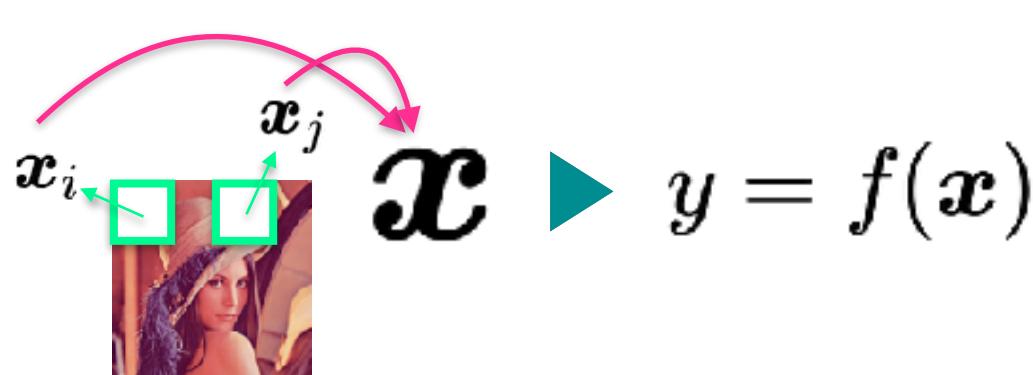


- SIFT
- SURF
- HOG
- etc...

大域特徴量

global feature descriptor

1枚の画像から得られた
局所特徴量をまとめて
1つのベクトルにする



- BoVW
- VLAD
- FisherVector
- etc...

識別器

classifier

大域特徴量を入力にし、
クラス分類

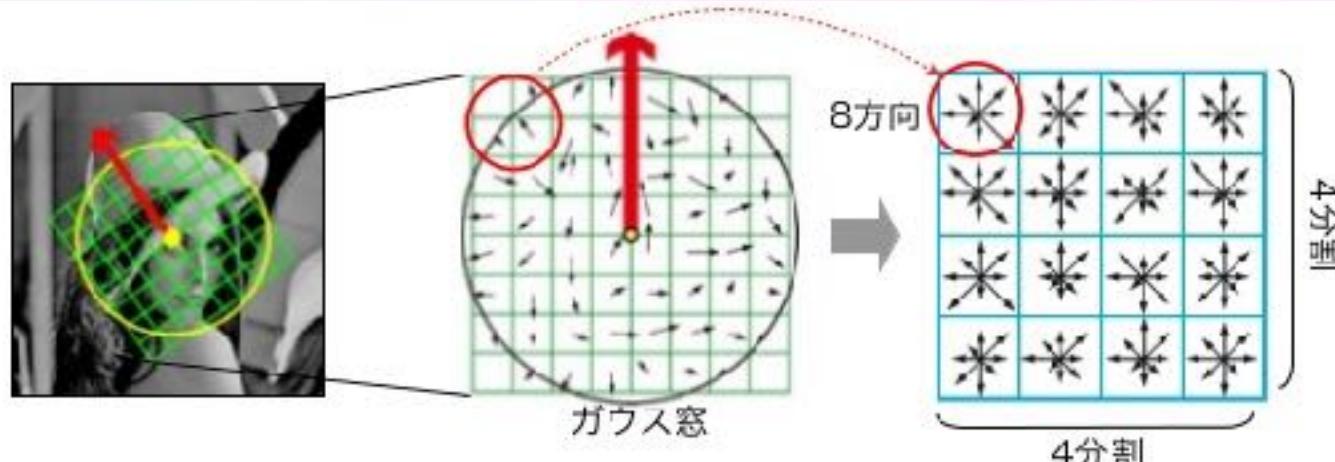
$$y = f(\mathbf{x})$$

- SVM
- kNN
- Decision tree
- etc...

物体認識のための局所特徴量

例：SIFT

特徴量の記述：特徴ベクトル算出



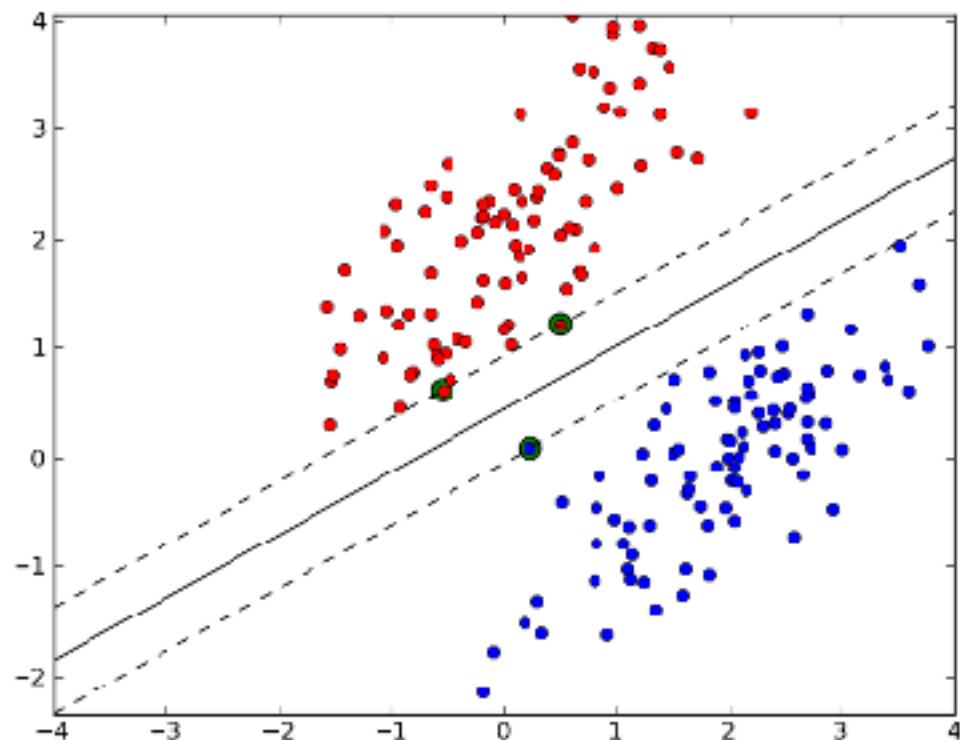
- 拡大縮小
 - 回転
 - 輝度
- の変化に強い

“Scale-Invariant Feature Transform”

識別器

特徴ベクトル x の空間をクラスに分ける
= 識別平面をつくる

例 : SVM



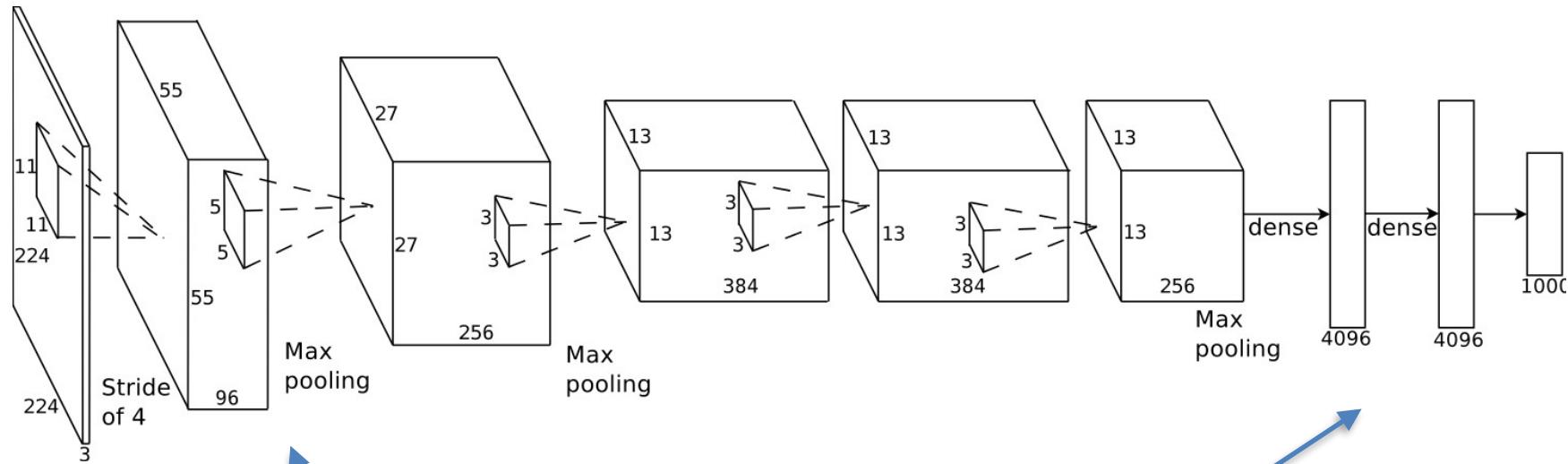
前Deep Learning時代の一般物体認識まとめ

20

- ▶ 局所特徴量ベースの画像認識がメインストリームだった
 - ▶ 局所特徴量抽出→大域特徴量コーディング→識別
- ▶ 特徴量は研究者が頑張って設計していた
 - ▶ 特定の特徴を捉える
 - ▶ 不変性を組み込む

CNNによる一般物体認識

End-to-Endの画像識別器 (Lecture7 再掲)²²



前半を特徴抽出器、

後半を識別器と見ることもできる

→SVMに置き換えたりもできる

従来 (Deep NN以前) 、研究者が手作業で頑張っていた
特徴フィルタの設計が自動化される

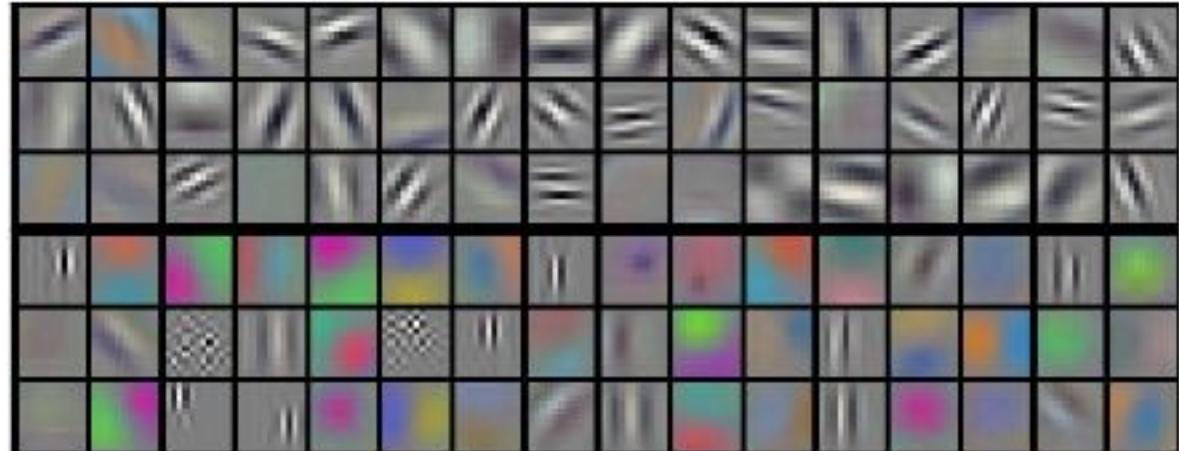
フィルタの自動獲得 (Lecture7 再掲)²³

(初期値は乱数で与えているのに)

CNNを学習させると、

識別に有効なフィルタが自動的に獲得される

学習済みAlexNetの1層目の可視化



Krizhevsky et.al., 2012

不变性

一般物体認識において望ましい不变性:

- スケール不变性
- 平行移動不变性
- 回転不变性
- アフィン変換不变性
- 輝度不变性

etc...

CNNはPooling layerによって、
(局所) 平行移動不变性を獲得している

Data augmentation (Lecture 7再掲)²⁵

- ▶ CNNはPoolingによって平行移動には強くなった
(平行移動不变性を獲得した)
- ▶ But, 回転不变性・鏡像不变性・affine変換不变性などの特性は持っていない
- ▶ →学習データを回転・反転・変形などして使うことで、これらの変換に対応する
- ▶ データ量が増え、過学習抑止効果も



CNNにおける不变性

- ▶ 一般的なCNNの不变性
 - ▶ (局所) 平行移動不变性
- ▶ Data augmentationによる不变性の付与
 - ▶ Horizontal flipping
 - ▶ Affine transformation
 - ▶ RGB shift
- ▶ 不变性をモデルに組み込むアプローチ
 - ▶ Spatial Transformer Networks
 - ▶ Oriented Response Networks
 - ▶ Deformable convolutional network
 - ▶ Scattering Networks

データの前処理

他にも、精度を高めるために
データに前処理を行う場合がある

- ▶ Mean subtraction
 - ▶ 平均→0
- ▶ Whitening
 - ▶ 平均→0
 - ▶ 分散→1、共分散→0

etc...

CNN特徴の可視化

可視化の重要性

よく言われる…

「Deep Learningはブラックボックス」



なぜうまくいくのかがわからず、
トライ＆エラーで改善するしかなくなってしまう

特徴マップの可視化

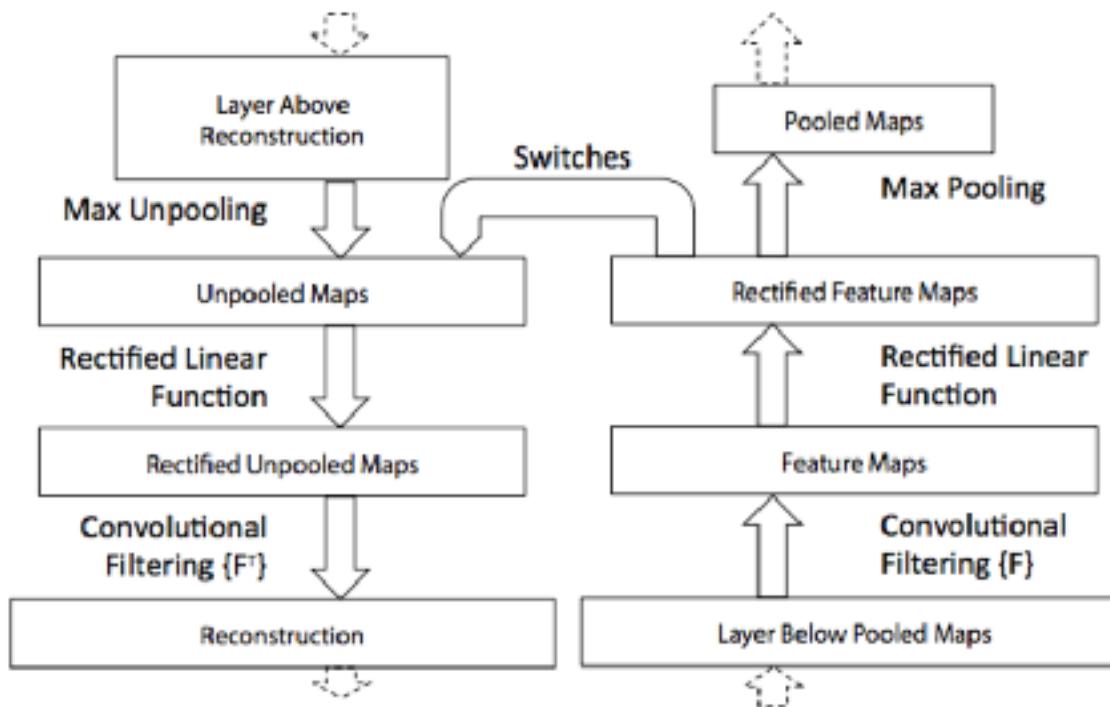
各特徴マップは、
入力画像のどのような特徴に反応しているのか？

- 特徴マップの出力を最大化する入力を探してみる
(w ではなく x を変数にしてSGD)
- 特徴マップの出力を逆伝播させてみる
etc...

Deconvolutionによる特徴マップの可視化 31

[Matthew D Zeiler et. al., 2013]

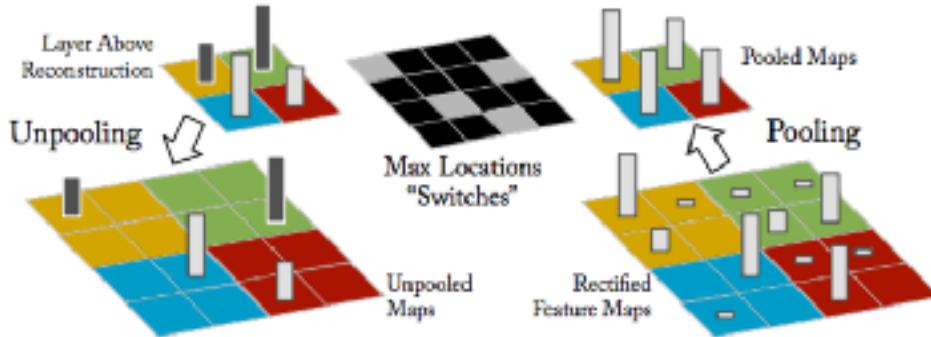
画像を入力して、
各特徴マップのactivationを入力空間に戻す



Deconvolution

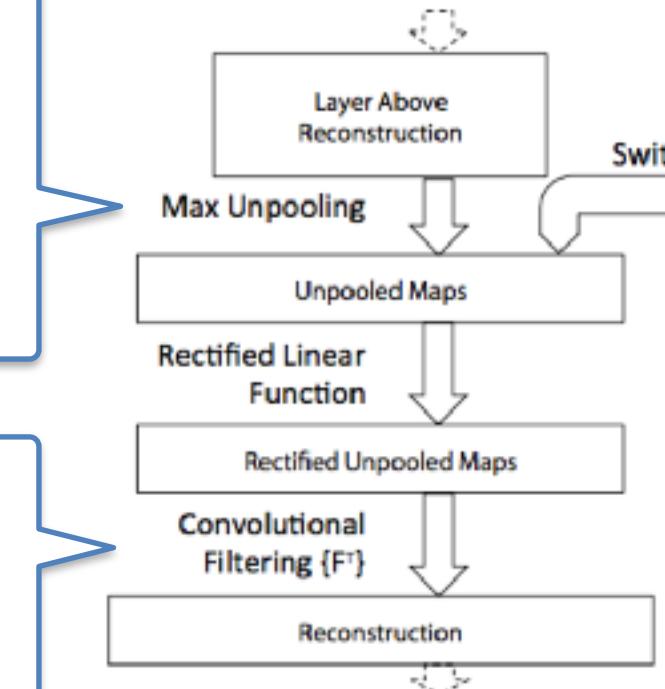
Unpooling

Feedforward時に伝播したピクセルを覚えておいて、そこに逆伝播させる

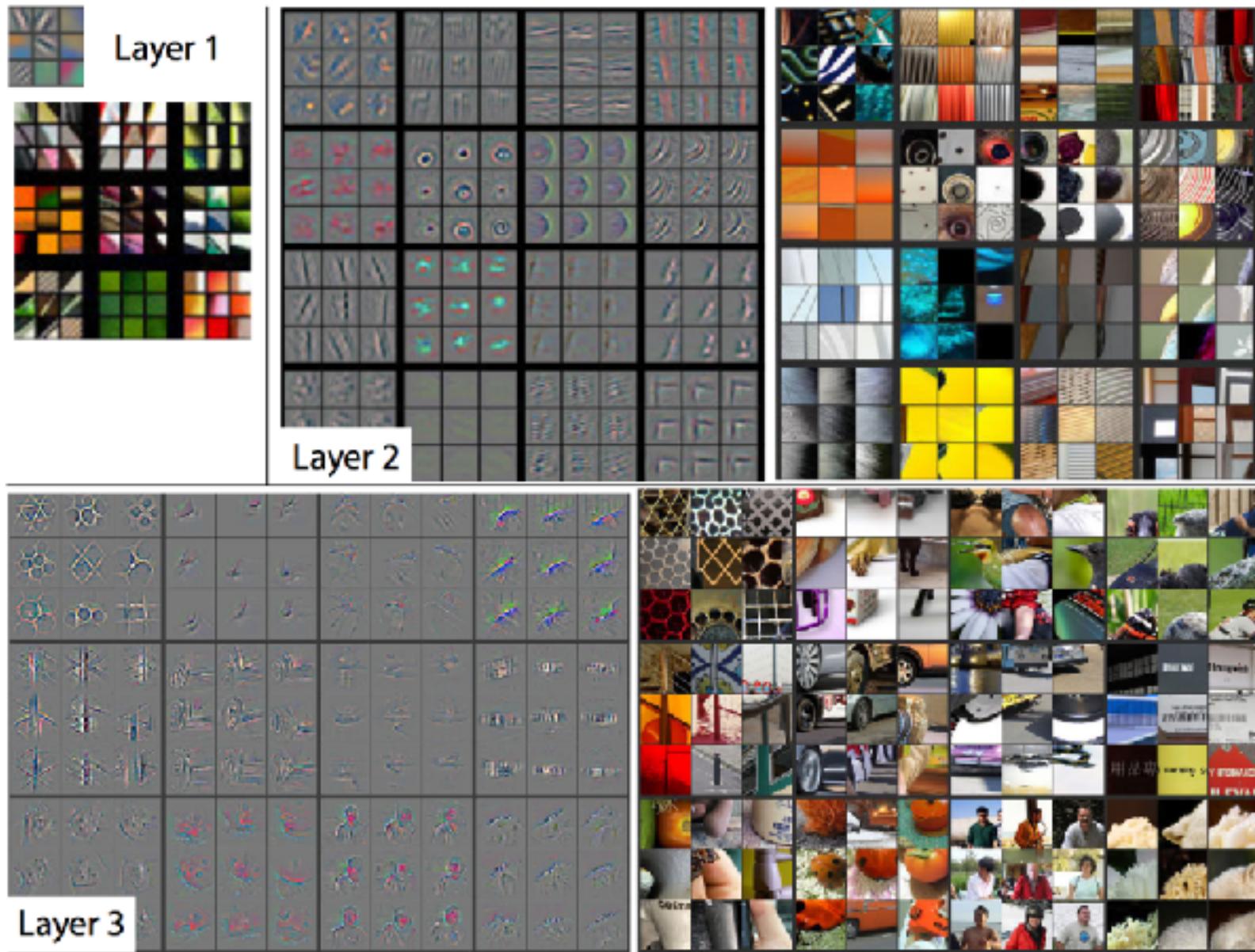


Deconvolution

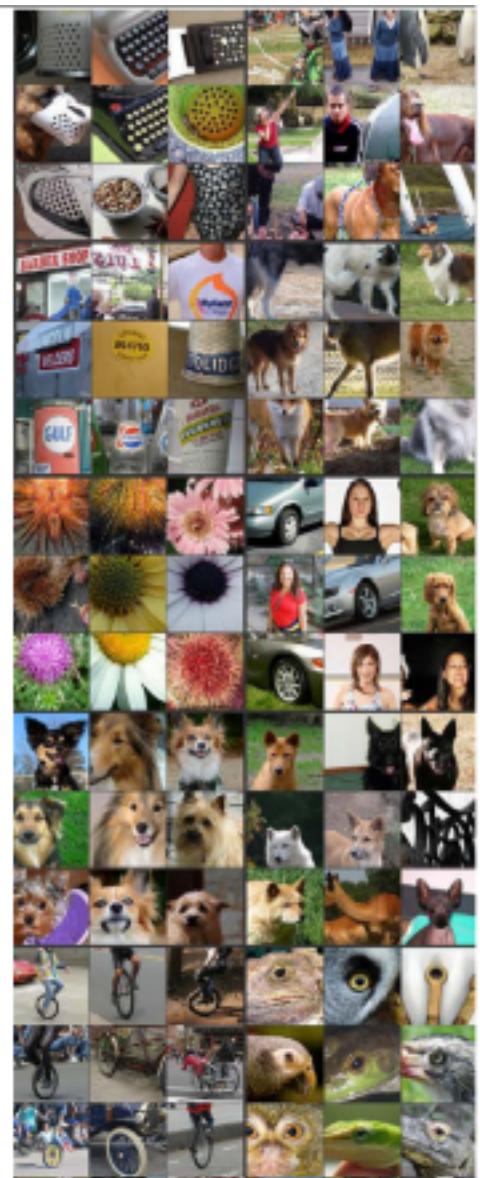
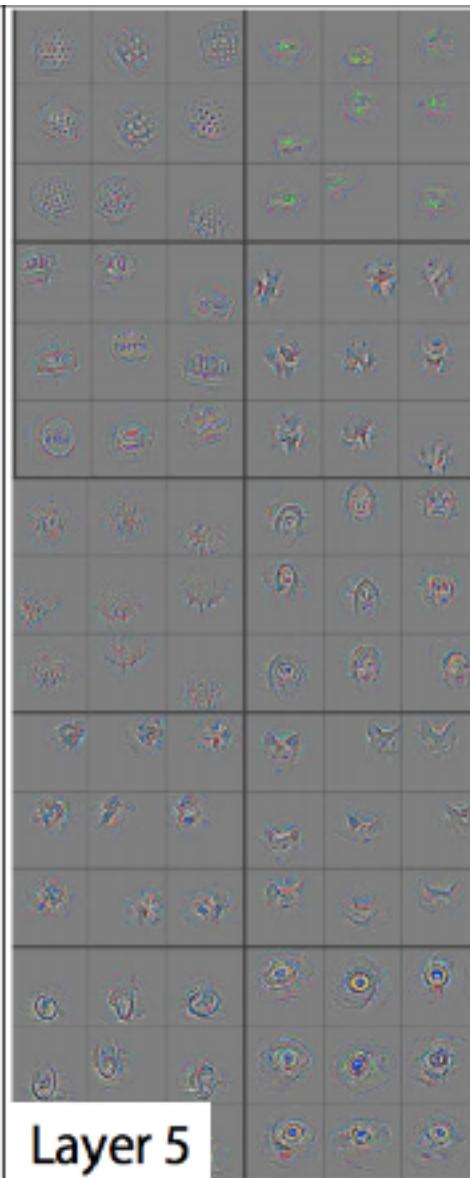
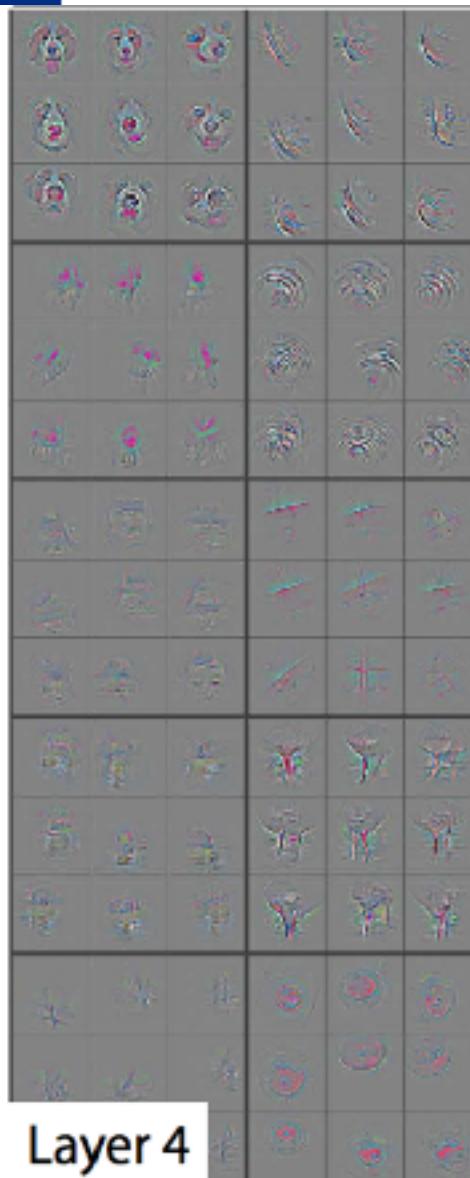
Feedforward時から上下左右反転させたfilterでConvolution



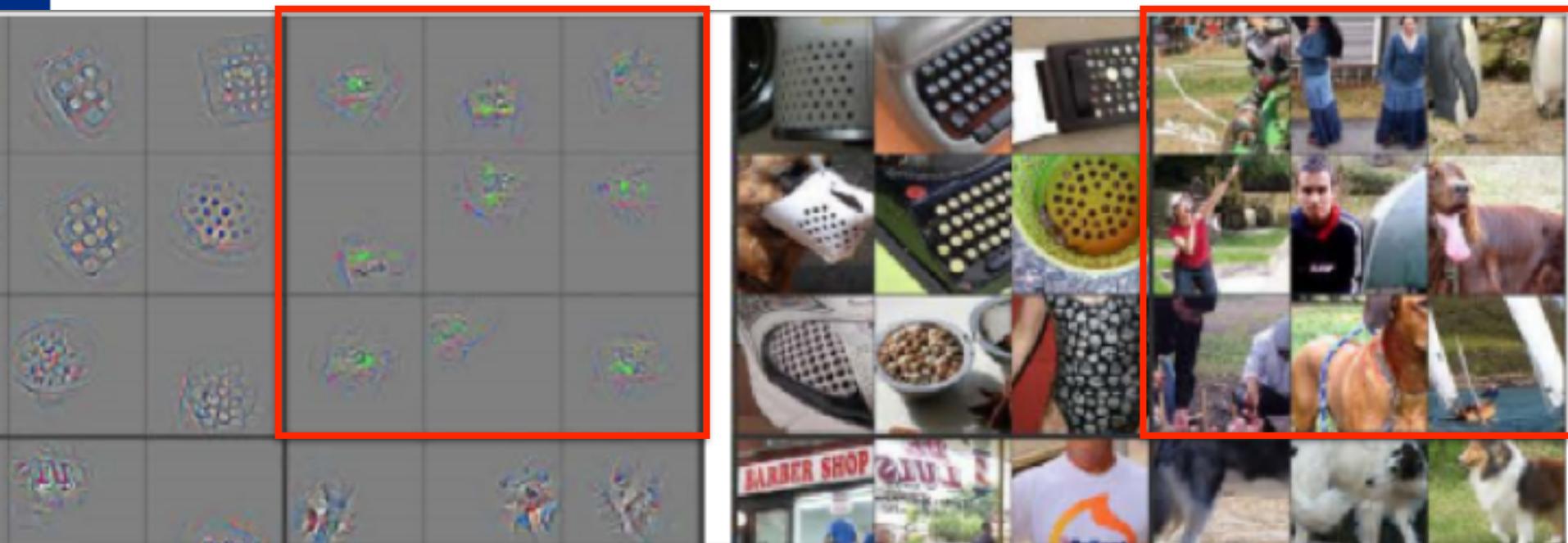
浅い層



深い層



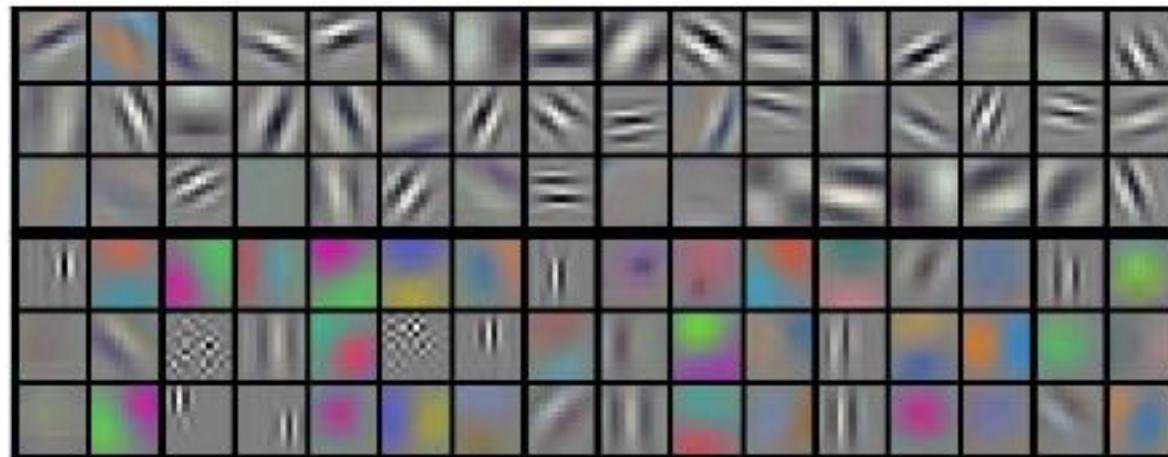
深い層



前景物体ではなく、
背景の草に強く反応していることがわかる
→このクラスの認識に何が寄与しているか？

フィルタの可視化

例：AlexNetの1層目の可視化



Krizhevsky et.al., 2012

フィルタの可視化によるアーキテクチャの改善

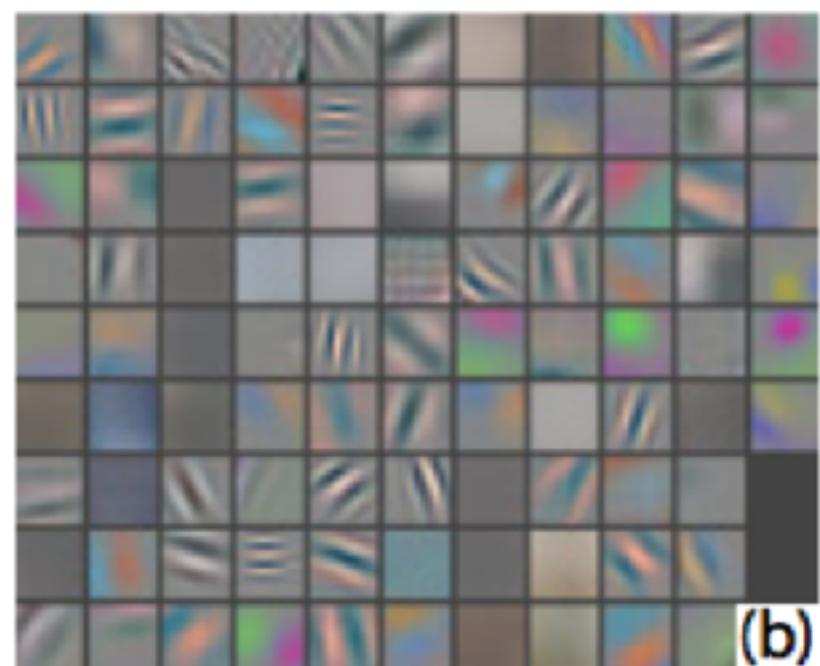
例：Alexnet [Krizhevsky et al., 2012]

1st layer



(a)

2nd layer



(b)

極端に高い・低い周波数成分ばかり

ぼやけてしまうフィルタが多い

フィルタの可視化によるアーキテクチャの改善

38

極端に高い・低い周波数成分ばかり

ぼやけてしまうフィルタが多い

1st layerのfilterが大きすぎ？

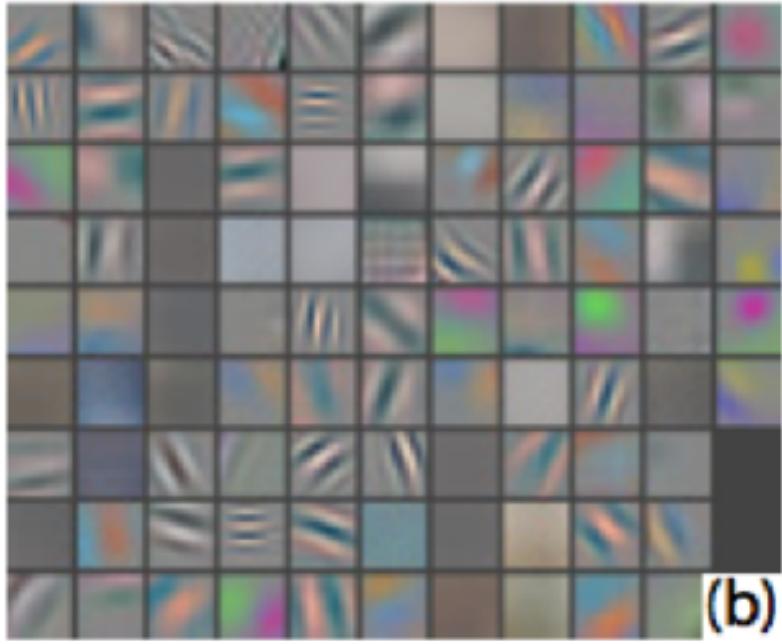
1st layerのstrideが長すぎ？

1st layerのfilterを 11×11 から 7×7 に

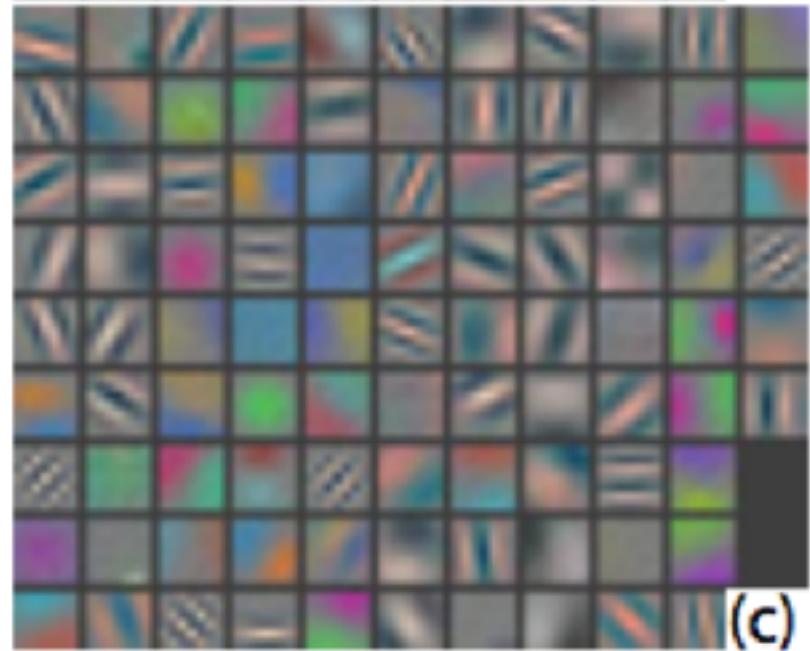
1st layerのstrideを4から2に

フィルタの可視化によるアーキテクチャの改善

39



改善前



改善後

- ▶ 問題の緩和、精度向上

(小課題)

- ▶ フィルタの可視化を体験してみる
- ▶ [https://cs.stanford.edu/people/karpathy/
convnetjs/demo/cifar10.html](https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html)

近年のCNN

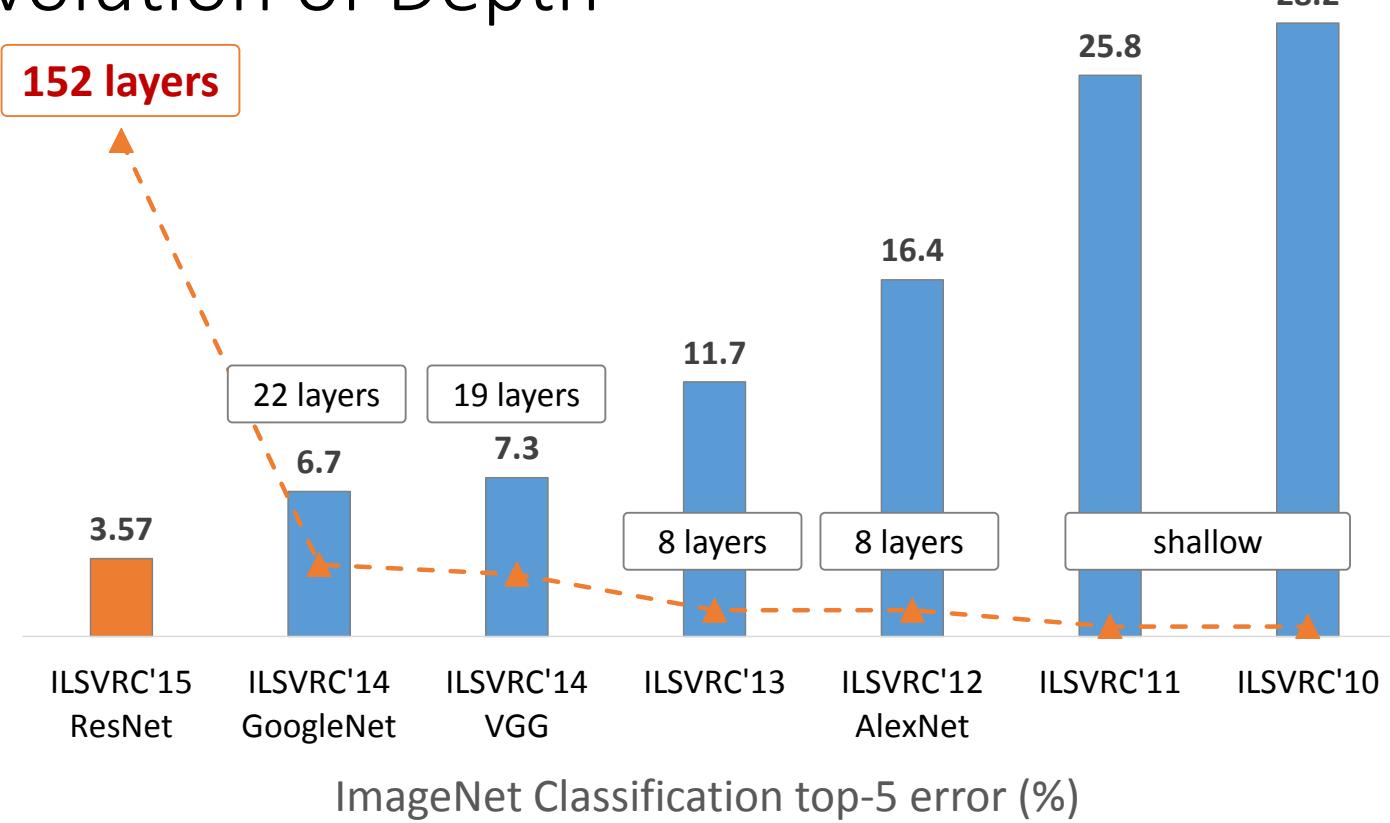
Imagenet Large Scale Visual Recognition Challenge:

- 大規模（120万枚・1000クラス）画像認識コンペ
- 一般物体認識の精度を競う

CNNの多層化

- ▶ 近年の一般物体認識におけるCNN：
いかにDeepにするかの戦い

Revolution of Depth



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

適切にDeepにすれば性能が上がる！

45

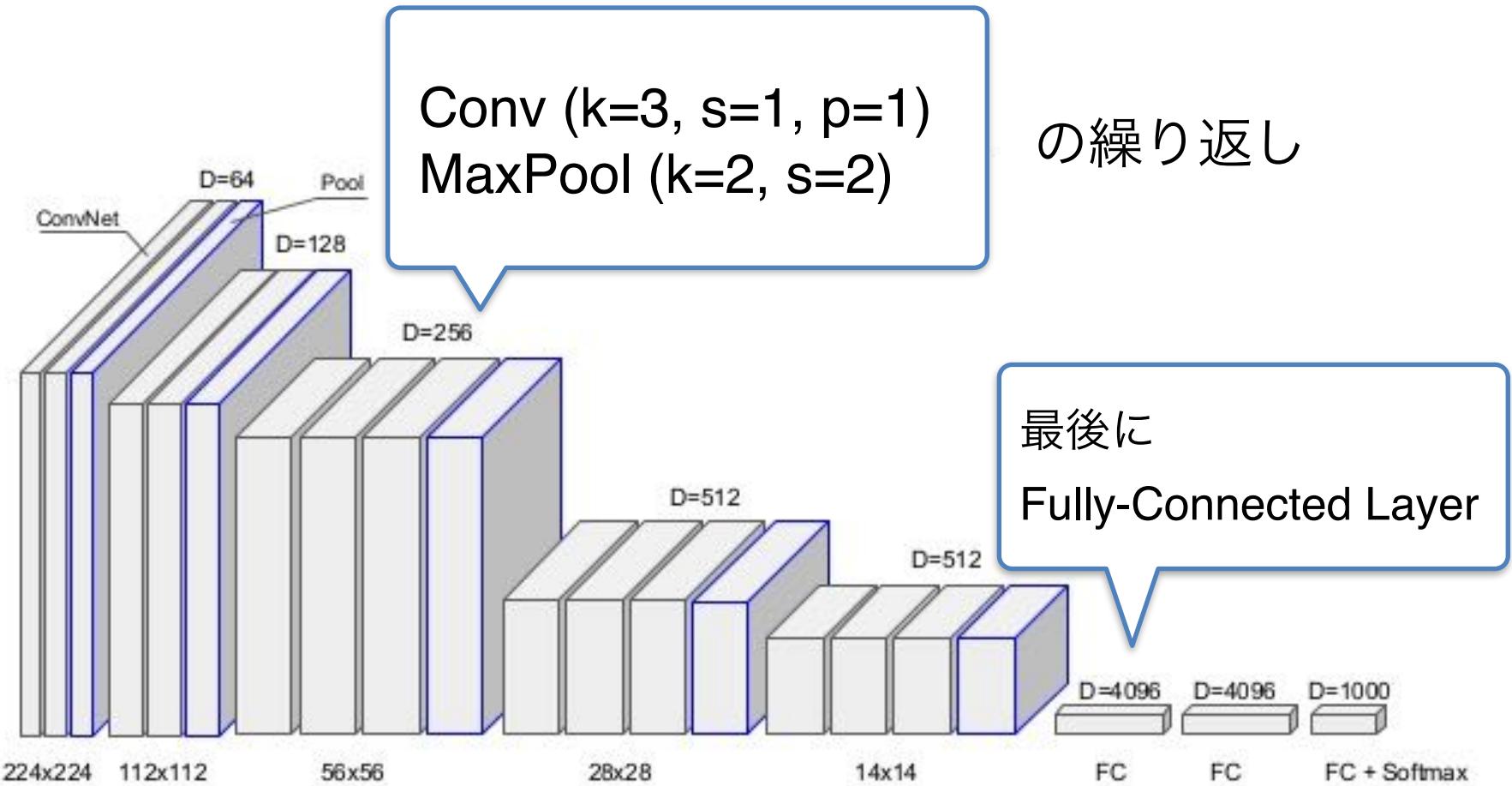
- ▶ 「適切に」？
- ▶ うまく学習させる：
Deepなネットワークは学習しづらい
 - ▶ 勾配消失・勾配爆発
 - ▶ 内部共変量シフト
 - ▶ 過学習
- etc...

本日紹介するCNN

- ▶ ILSVRC2014
 - ▶ GoogLeNet
 - ▶ VGGNet
- ▶ ILSVRC2015
 - ▶ ResNet

VGG Net

VGG-Net



Very Deep

16, 19層(weight layerの数)までDeepにできた

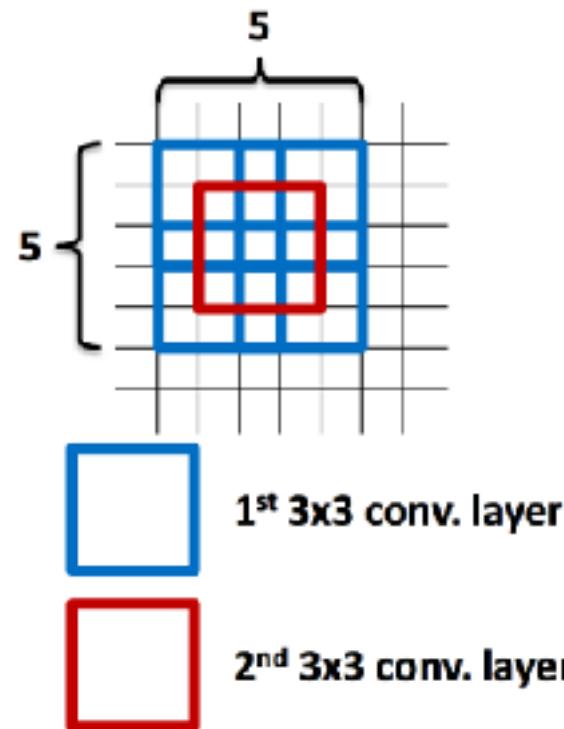
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	15 weight layers	16 weight layers	19 weight layers
Input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv1-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Conv層ではマップサイズは不变
Pooling層でダウンサンプリング

3x3 conv

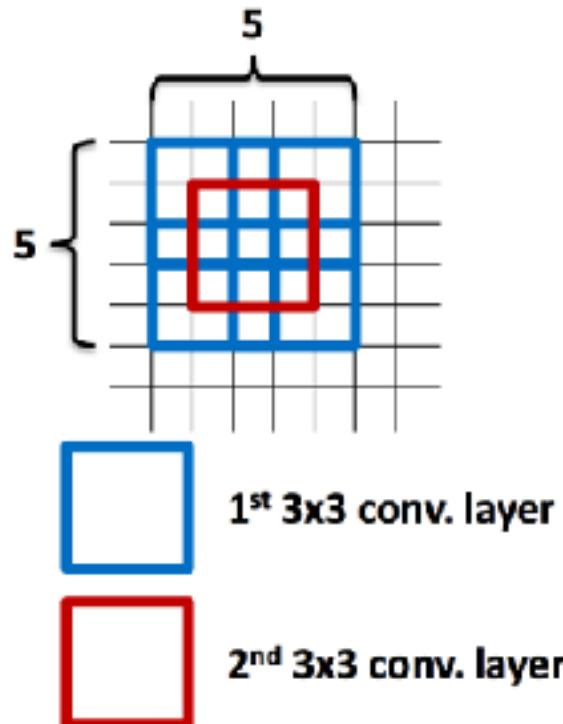
Net Configuration			
	C	D	E
weight layers	16 weight layers	16 weight layers	19 weight layers
(4 × 224 RGB image)			
-64	conv3-64	conv3-64	conv3-64
-64	conv3-64	conv3-64	conv3-64
maxpool			
-128	conv3-128	conv3-128	conv3-128
-128	conv3-128	conv3-128	conv3-128
maxpool			
-256	conv3-256	conv3-256	conv3-256
-256	conv3-256	conv3-256	conv3-256
conv1-256	conv3-256	conv3-256	conv3-256
maxpool			
-512	conv3-512	conv3-512	conv3-512
-512	conv3-512	conv3-512	conv3-512
conv1-512	conv3-512	conv3-512	conv3-512
maxpool			
-512	conv3-512	conv3-512	conv3-512
-512	conv3-512	conv3-512	conv3-512
-512	conv3-512	conv3-512	conv3-512

3x3 Conv × 2
 → 5x5 pixelの情報を
 コーディングできる

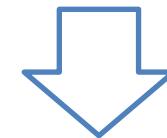


3x3 conv

3x3 Conv × 2
→ 5x5 pixelの情報を
コーディングできる



5x5 Conv × 1と何が違う？



- パラメータ数削減
 - 計算量削減
 - 過学習耐性
- 非線形変換(ReLU)増加
- 表現力向上

パラメータ数削減

例：

3x3conv × 3 vs. 7x7conv × 1

各層Cチャネルあるとして、

$$3x3\text{conv} \times 3: 3(3^2 C^2) = 27C^2$$

$$7x7\text{conv} \times 1: 7^2 C^2 = 49C^2$$

Table 2: Number of parameters (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

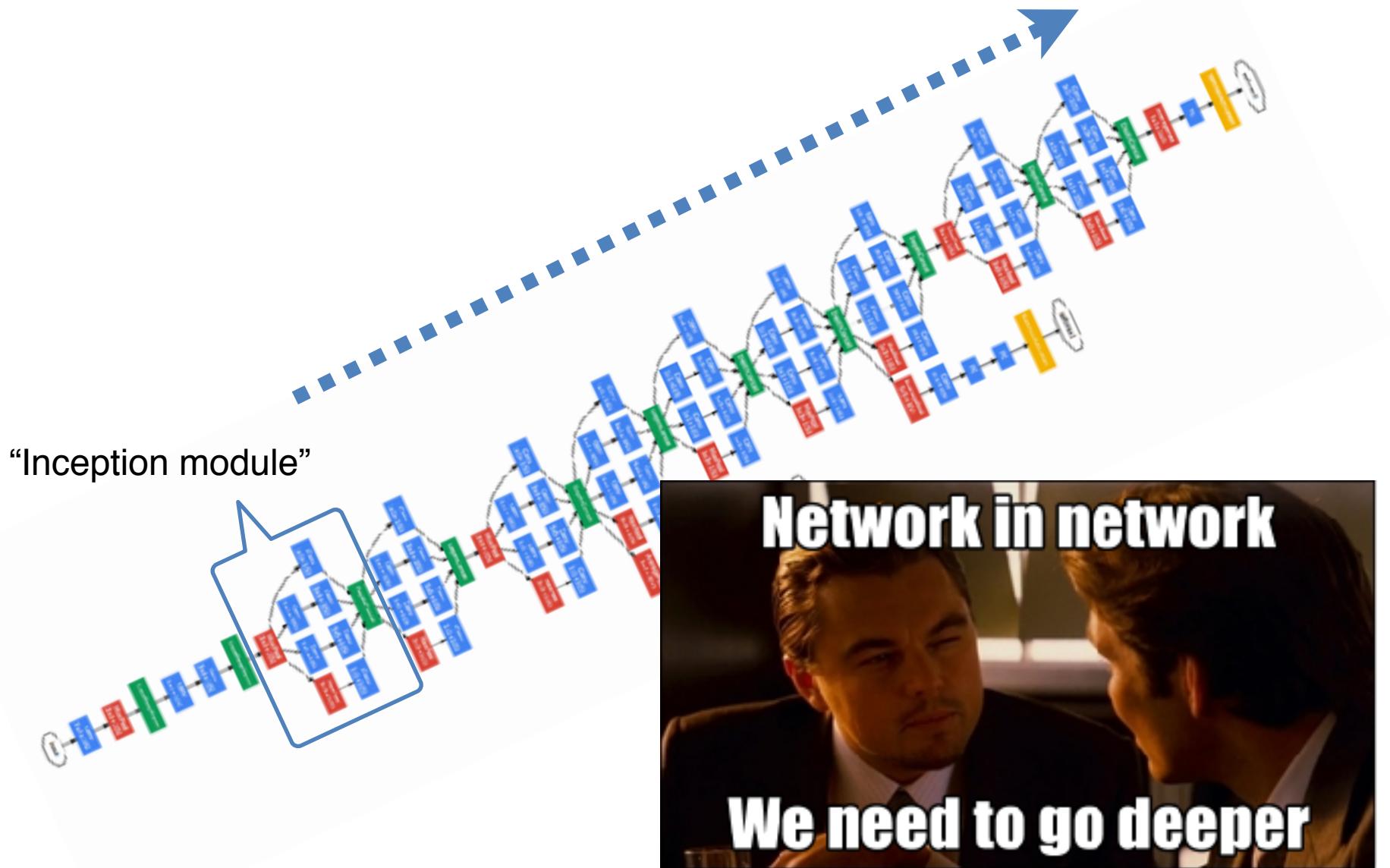
In Table 2 we report the number of parameters for each configuration. In spite of a large depth, the number of weights in our nets is not greater than the number of weights in a more shallow net with larger conv. layer widths and receptive fields (144M weights in (Sermanet et al., 2014)).

ILSVRC2013 5th

GoogLeNet

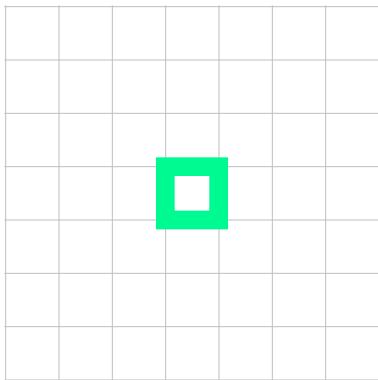
GoogLeNet

54

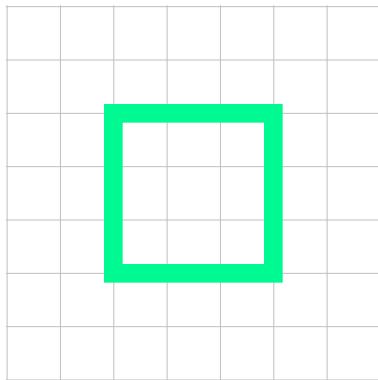


カーネルサイズはどれが良い？

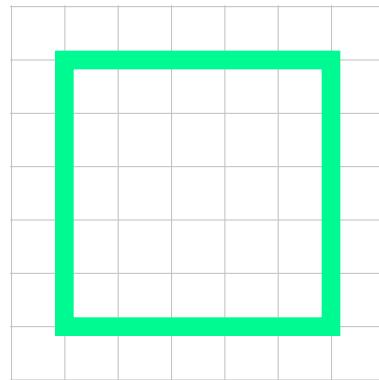
1x1



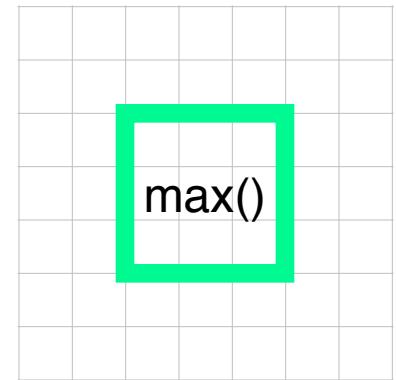
3x3



5x5



pooling

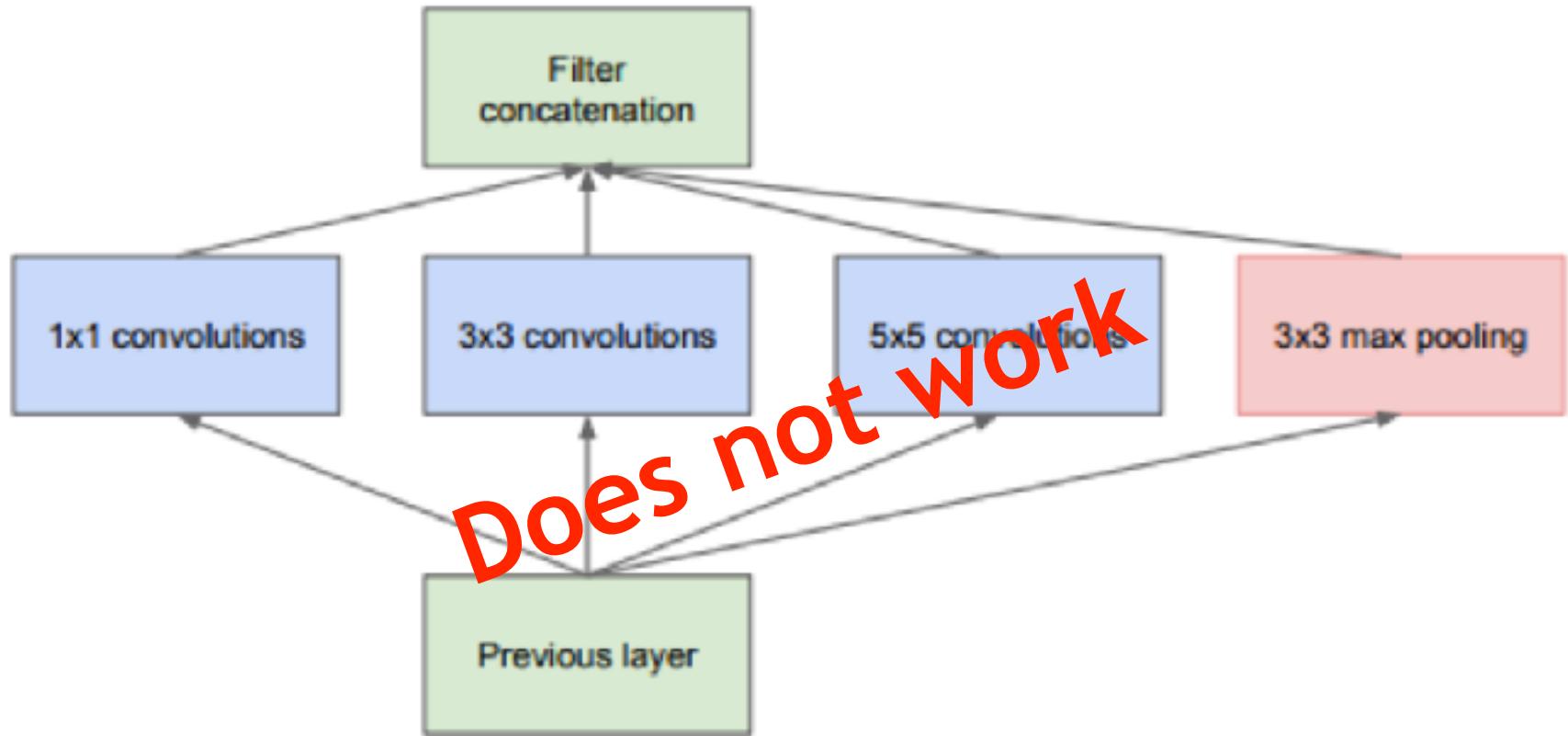


コーディングする範囲は狭いが、
非線形性を足しつつDeepにできる

より広い範囲の情報を
コーディングできるが
Deepにしづらい？

ConvNetには
Pooling Layerが
不可欠！

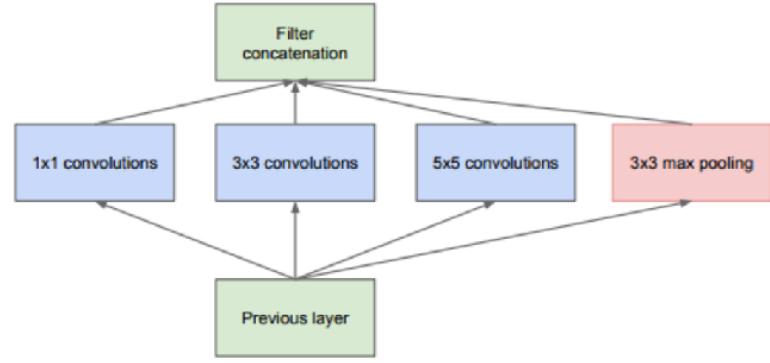
全部使う



(a) Inception module, naïve version

Naive inception module does not work

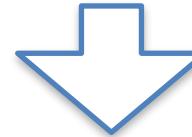
57



(a) Inception module, naïve version

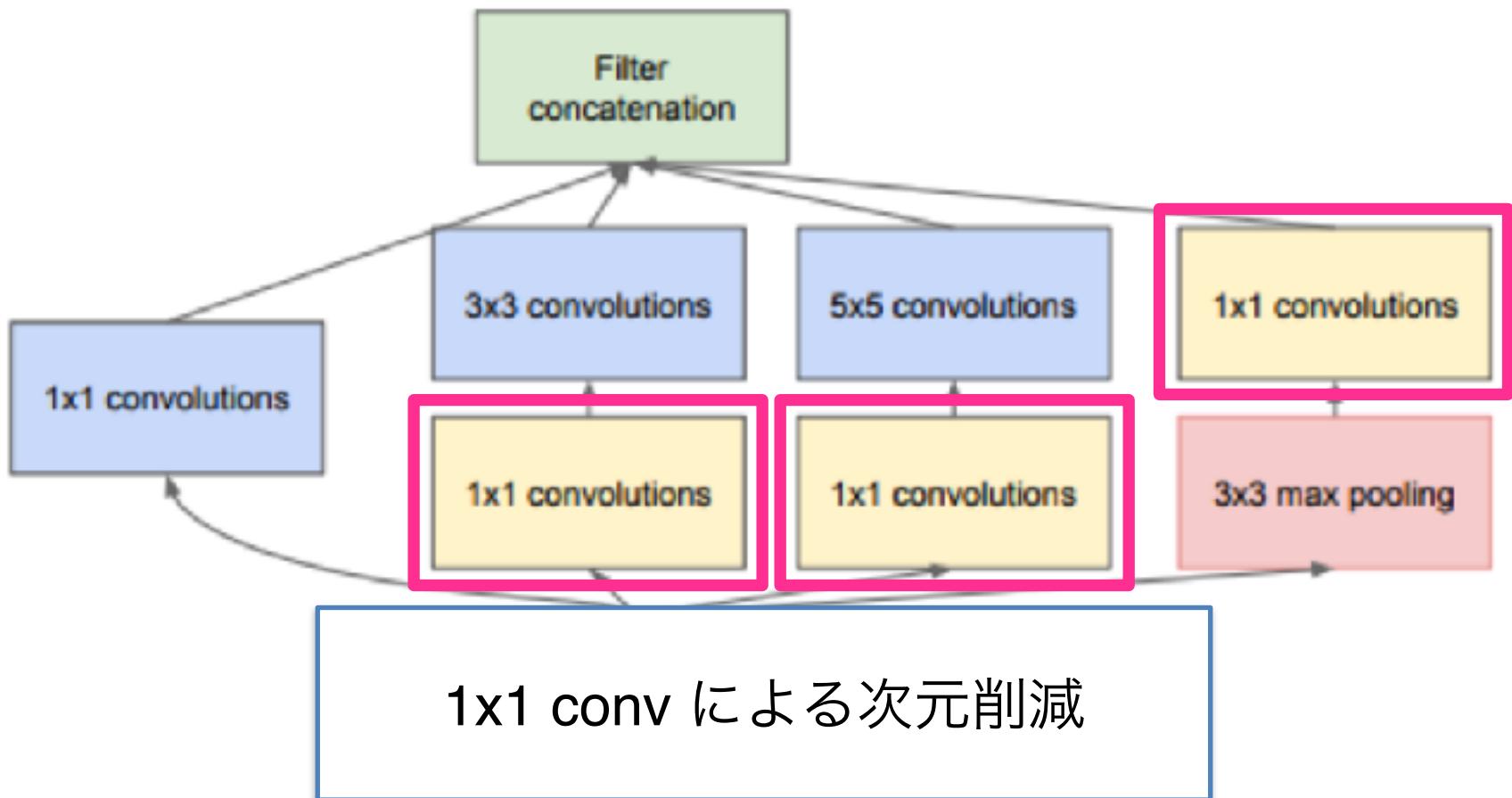


次元数が爆発的に増える
• 特に Pooling layer では
入力の次元数が保存される



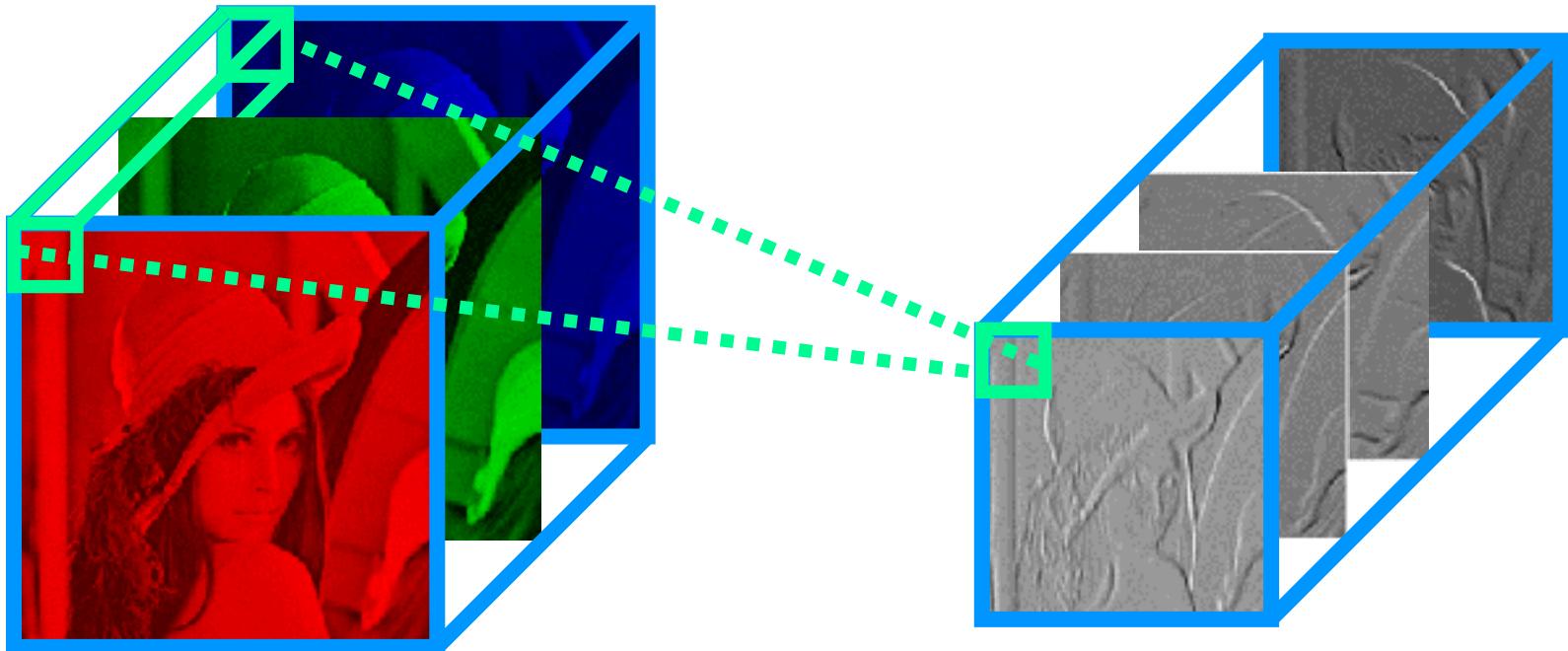
Inception moduleを
積んでいくとすぐ爆発

Inception module



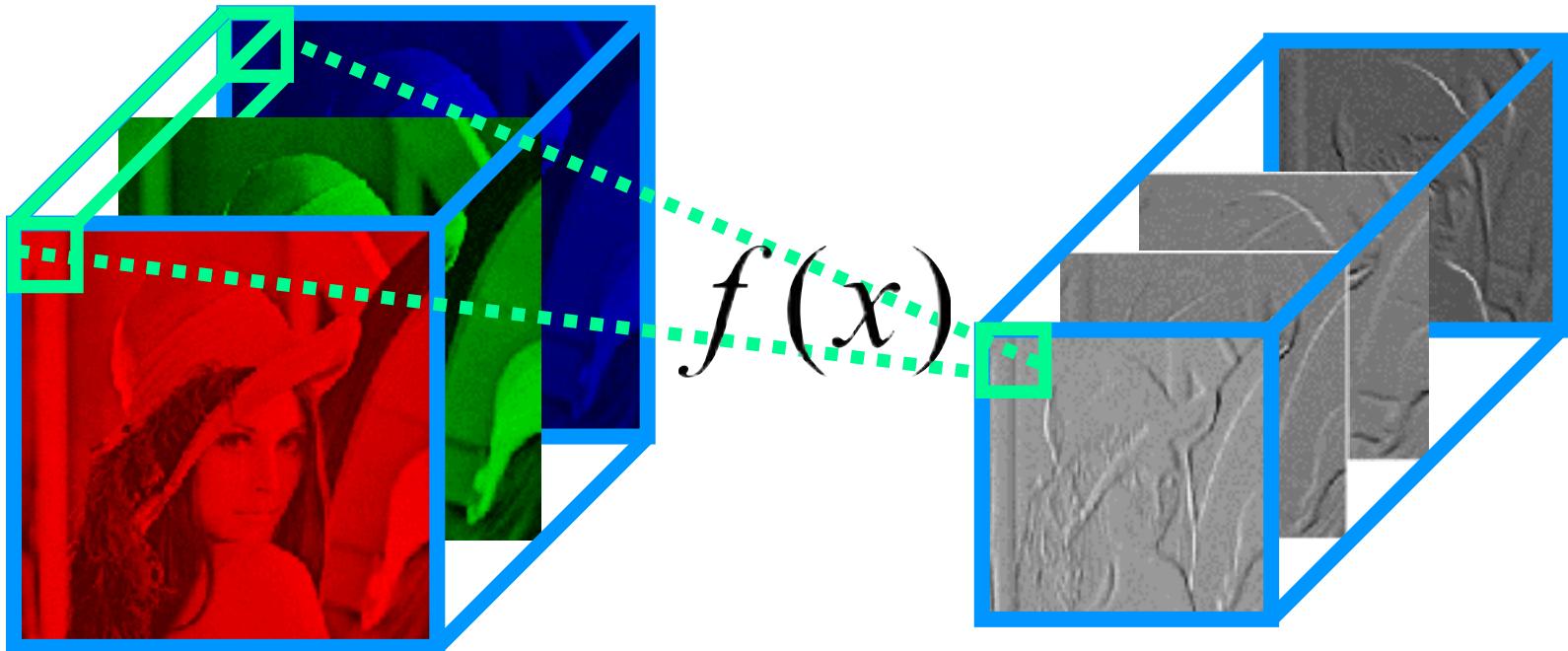
(b) Inception module with dimension reductions

1x1 conv



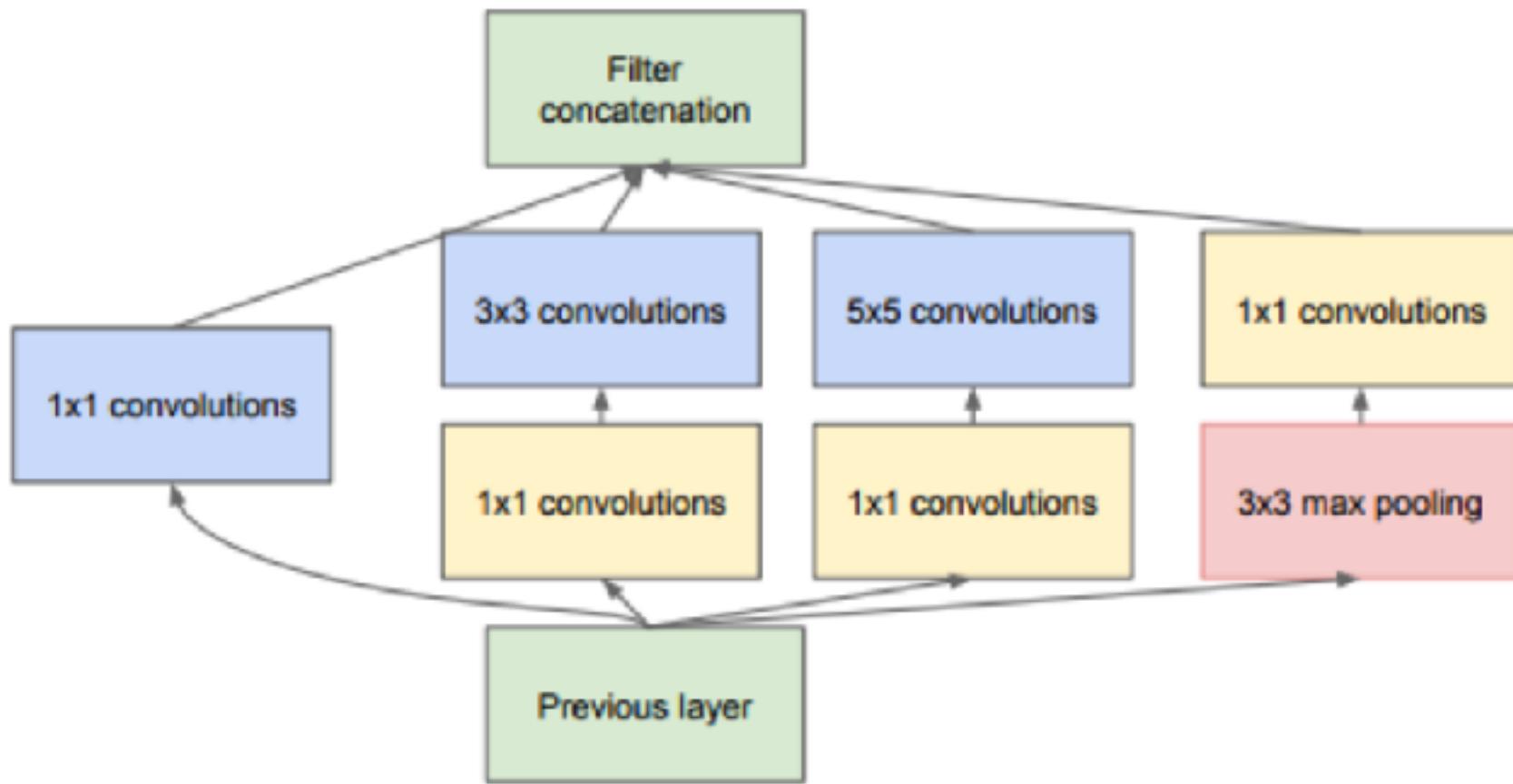
マップサイズ不变・線形変換

1x1 conv



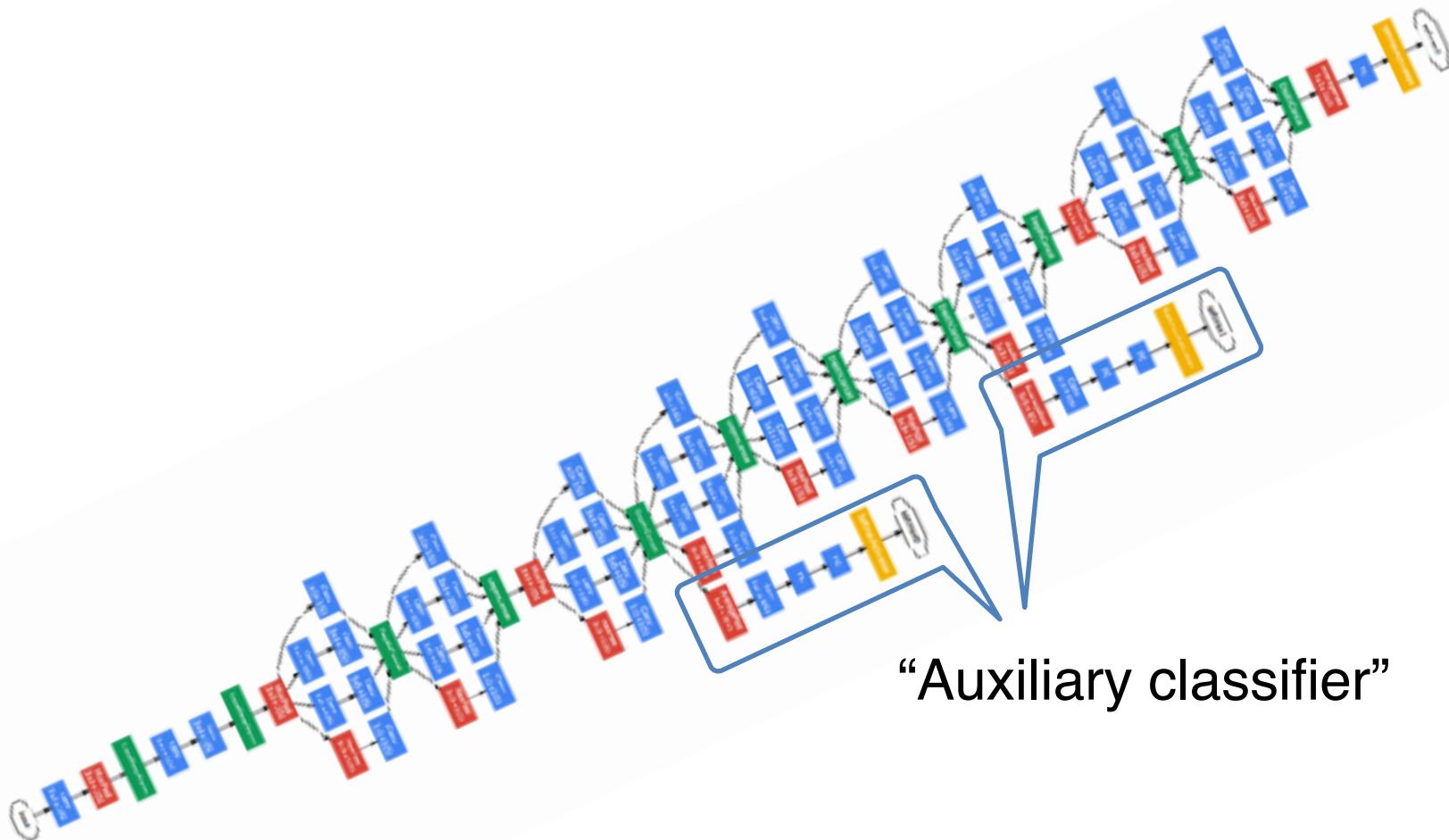
活性化関数による非線形性

Inception module



(b) Inception module with dimension reductions

Auxiliary classifiers



Auxiliary classifiers

- ▶ 勾配消失を防ぐために、
中間層からclassifierを生やし学習に使う
- ▶ 浅めのネットワークでもそれなりに精度が出る
→ 中間層の特徴もdiscriminativeにできるはず
というモチベーション
- ▶ 学習時、Auxiliary classifierのlossは、
重み(論文では0.3)を掛けてメインのlossに足される
- ▶ 識別時は無視される

Batch Normalization

内部共変量シフト

“入力データを正規化(Normalize)すると精度が上がる”
という知見

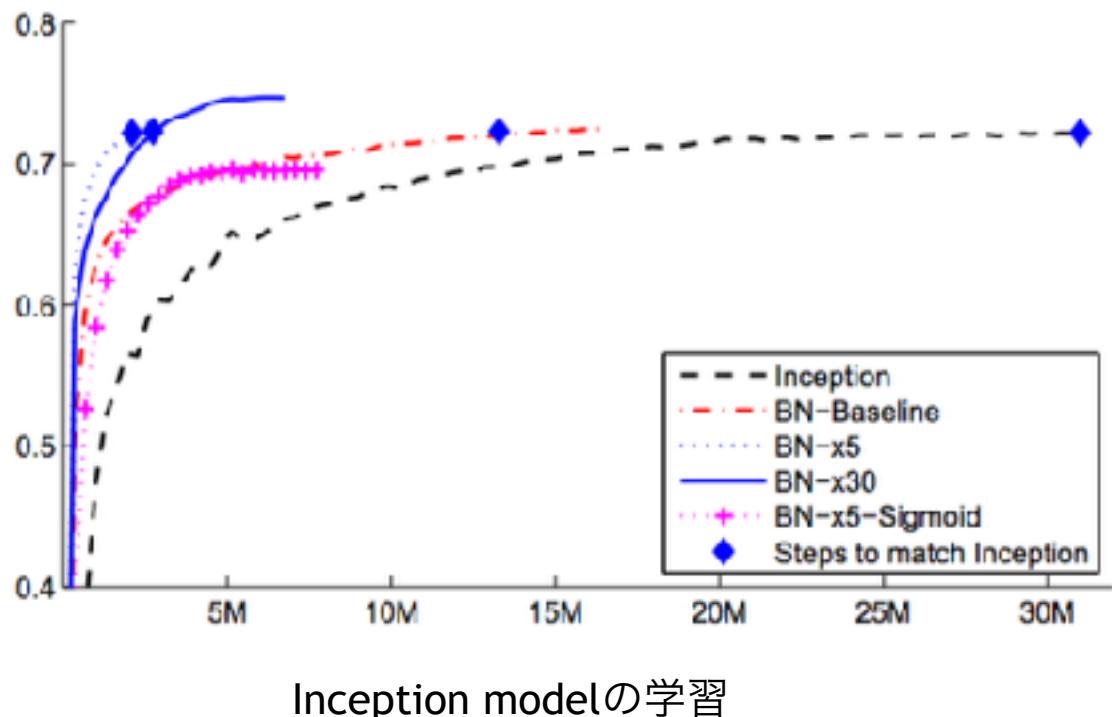
- 平均→0, 分散→1, 相関→0

but, せっかく入力を正規化しても、
学習が進んで浅い層のパラメータが変わるために、
深い層の入力はブレブレ
→ 内部共変量シフト Internal covariate shift

Batch normalization

層ごと、Mini-batchごとに平均、分散を正規化

- 学習の高速化
- 精度向上

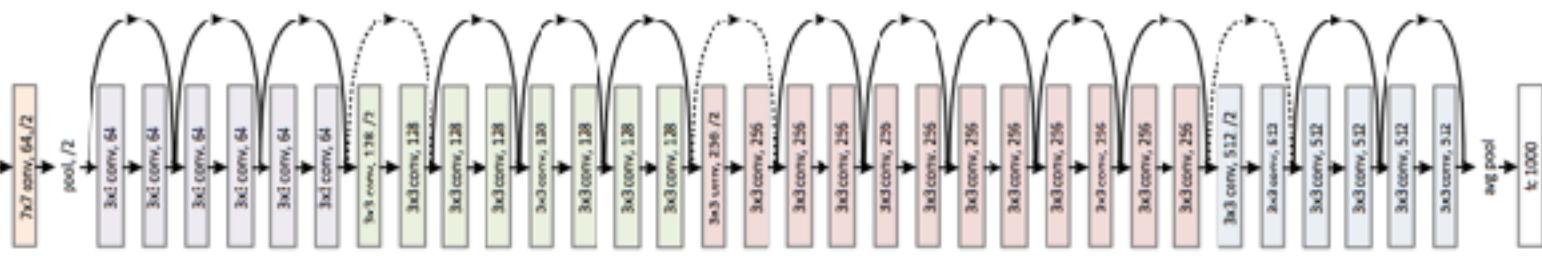


ResNet

ResNet

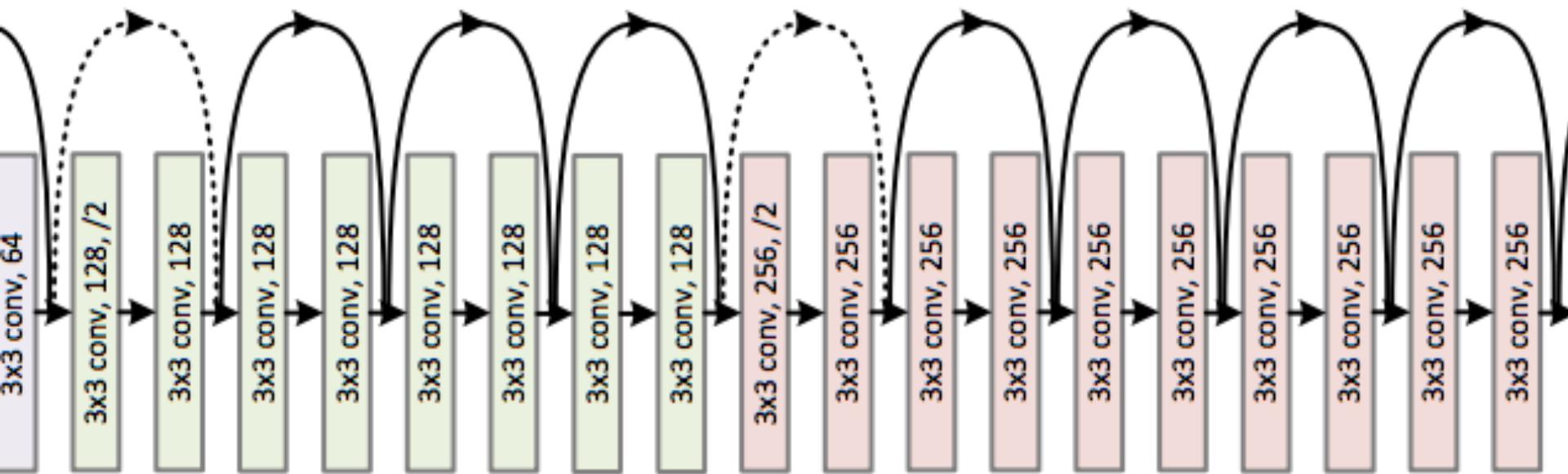
68

34-layer residual



ResNet

“Skip connection”



Degradation

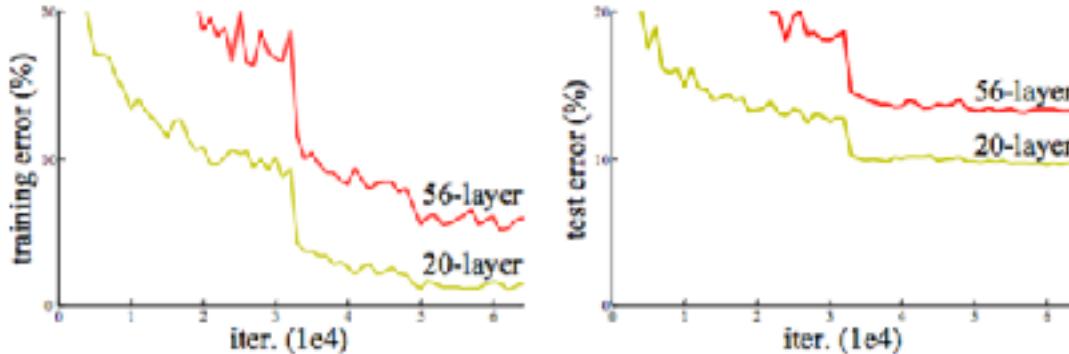


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

(Batch Normalizationなどの工夫の上であっても)
Deepにすると精度が下がる

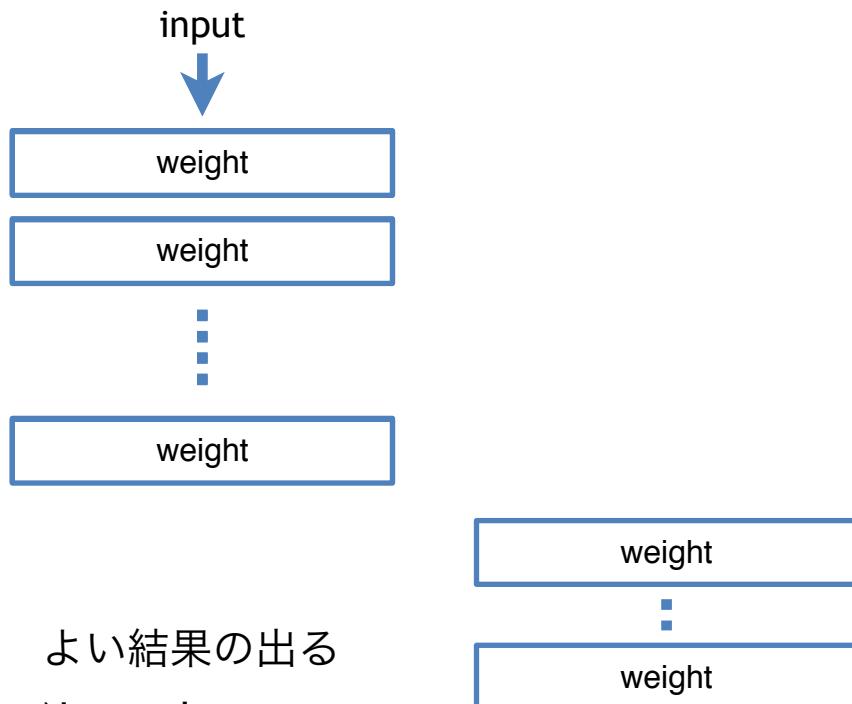
Degradation

71

(Batch Normalizationなどの工夫の上であっても)

Deepになると精度が下がる

→おかしくね？



問題意識：
恒等変換を学習しやすい
仕組みを作ってみよう

このsubnetworkに
恒等変換を学習させれば
少なくとも
悪くはならないはず

Residual learning

通常のnetwork:



weight

ReLU

weight

ReLU

\mathbf{Z}

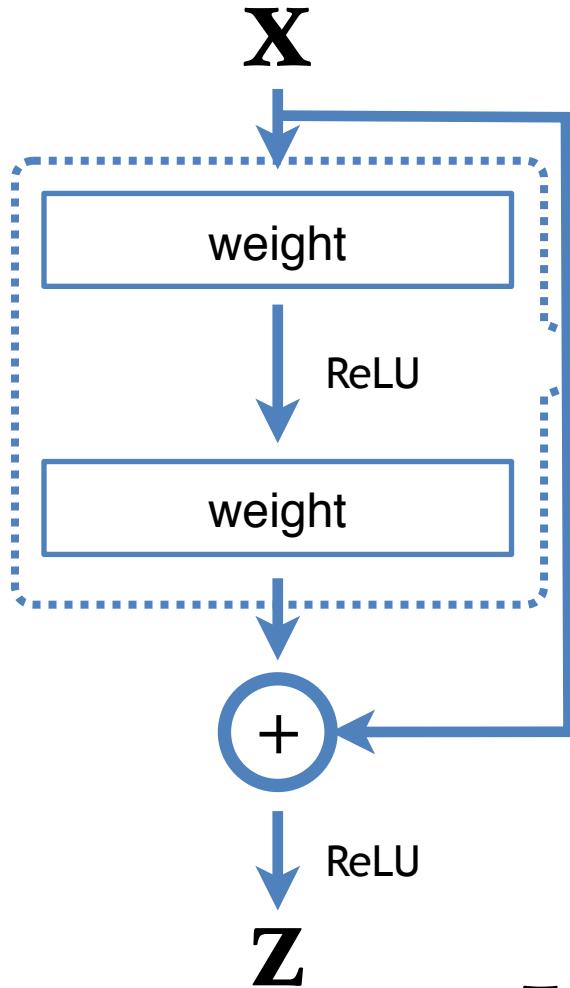
恒等変換を学習させたい時、

$$\mathbf{F}(\mathbf{x}) = \mathbf{x}$$

$$\mathbf{z} = \mathbf{F}(\mathbf{x})$$

Residual learning

Skip connection:



ある最適な F を学習する

$H(x)=x$ (恒等変換) のとき、
 $F(x)=0$

$$z = H(x) = F(x) + x$$

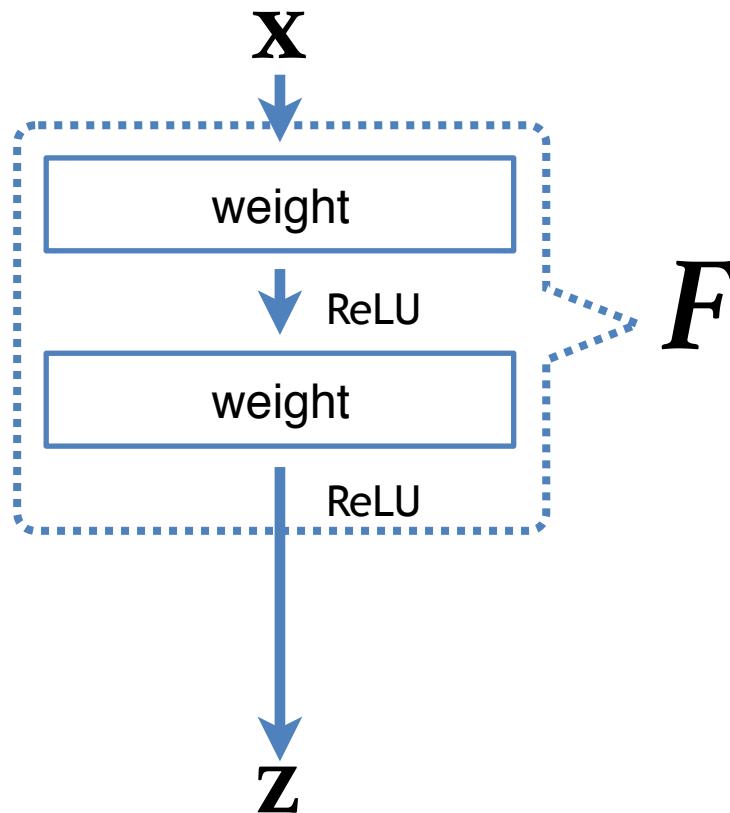
$F(x)=H(x)-x$
 "Residual(残差)"

Residual learning

恒等変換が欲しいときに、weight layerが学習すべき関数

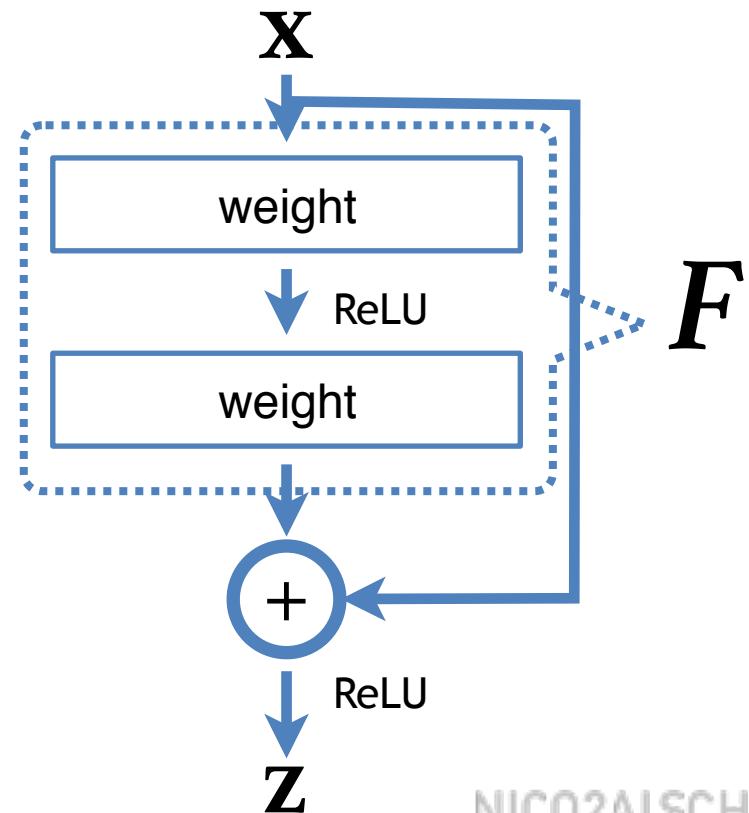
通常のnetwork

$$\mathbf{F}(\mathbf{x}) = \mathbf{x}$$



Residual learning

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}$$



Residual learning

恒等変換が欲しいときに、weight layerが学習すべき関数

通常のnetwork

$$\mathbf{F}(\mathbf{x}) = \mathbf{x}$$

Residual learning

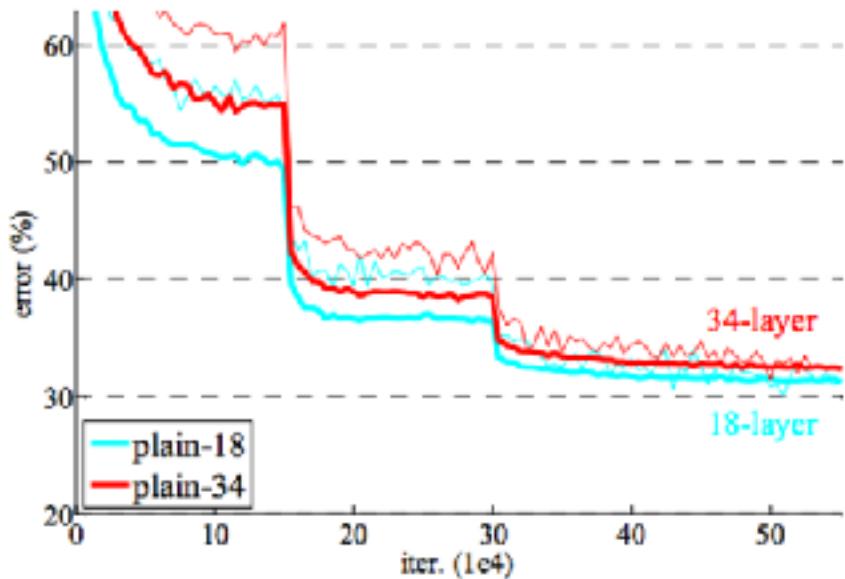
$$\mathbf{F}(\mathbf{x}) = \mathbf{0}$$

こっちのほうが
最適化しやすい！

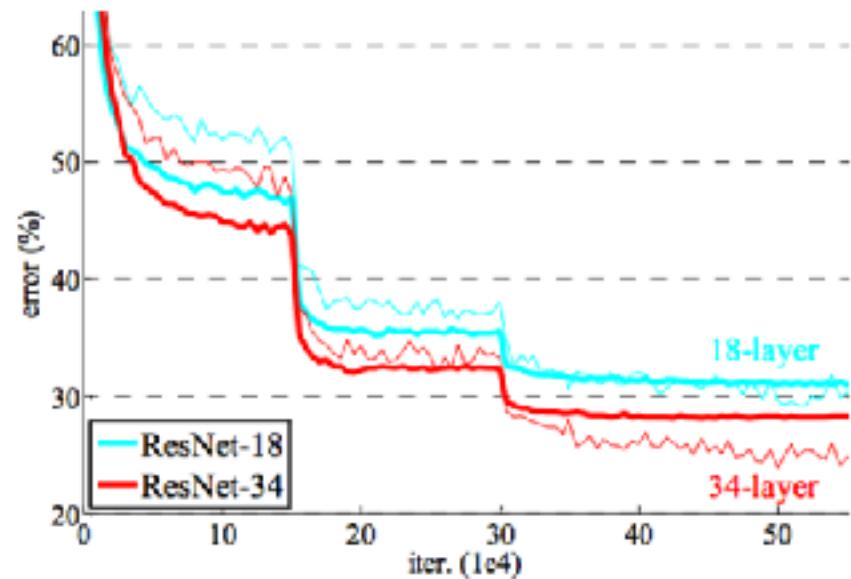
(...という仮説のもと、
Skip connectionを導入したら
うまくいった)

- ▶ なぜ通常のネットワークでは $\mathbf{F}(\mathbf{x}) = \mathbf{x}$ の獲得が難しいの？
 - ▶ 非線形変換も含むため、重みの調節が難しいのでは。
それに対して $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ の学習は
全ての重みをゼロに近づけるだけなので最適化しやすい。
- ▶ 本当に学習したい関数が恒等写像なんてことあるの？
 - ▶ 確かに現実のケースでは微妙。しかし、問題の緩和にはなる。
最適関数が恒等写像に近いなら、
どのみち恒等写像からの変動を見つけるのは難しくない。

Residual learningの効果

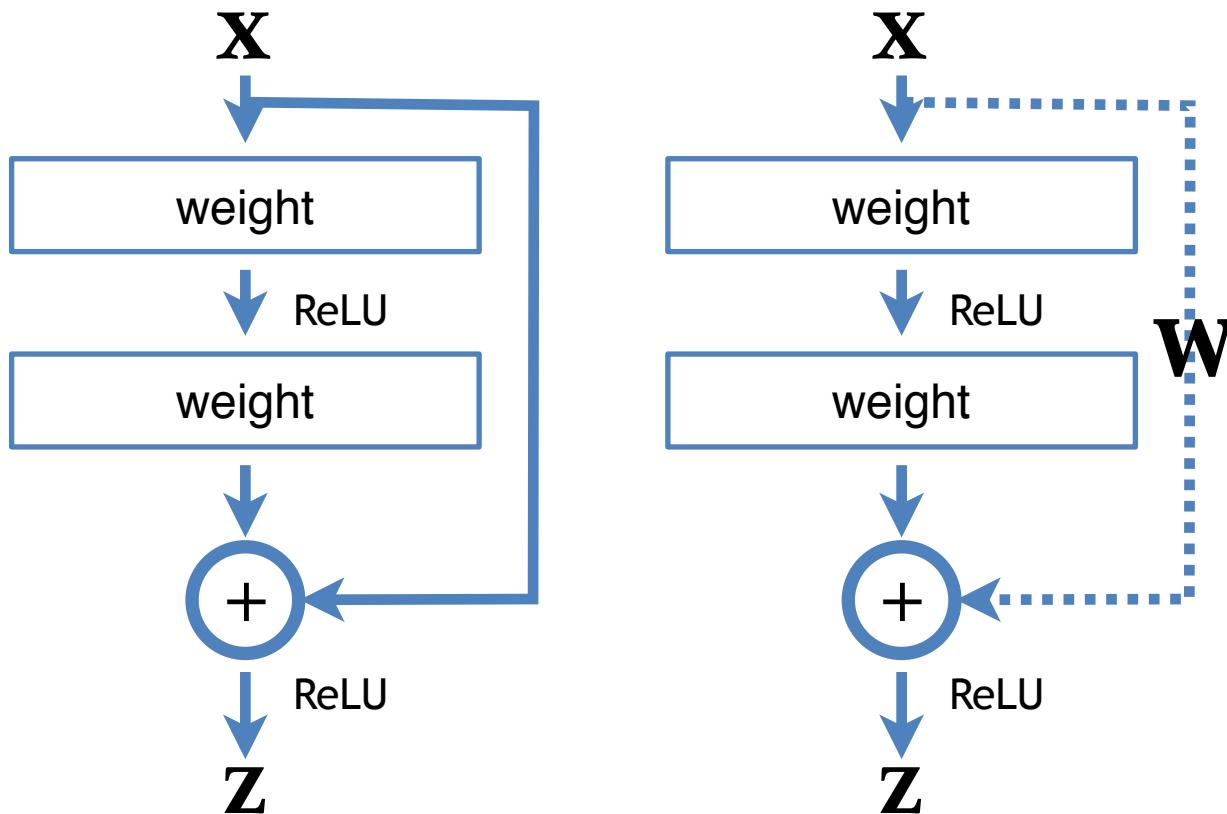


Skip connectionなし：
deepなネットワークは
逆に性能低下



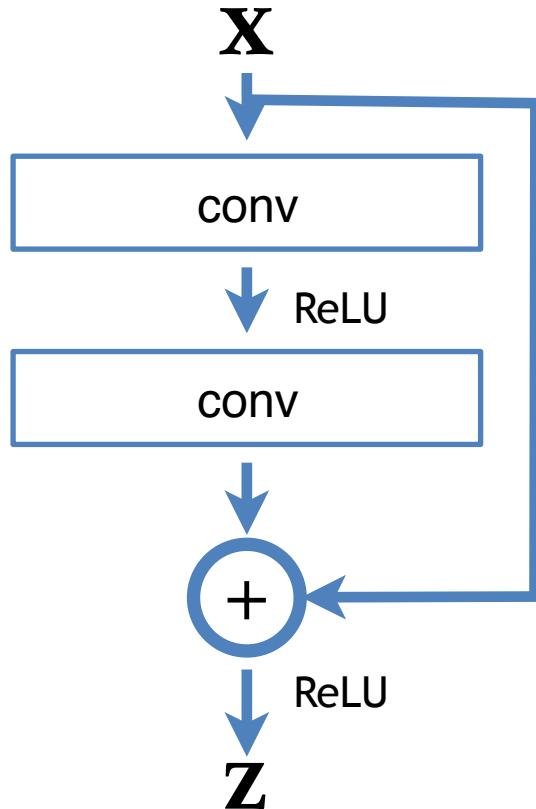
Skip connectionあり：
deepにすることで
性能向上

次元数が変わる場合



subnetworkを経て次元数が変わるとときは
恒等写像 \mathbf{x} の代わりに線形変換 \mathbf{Wx}

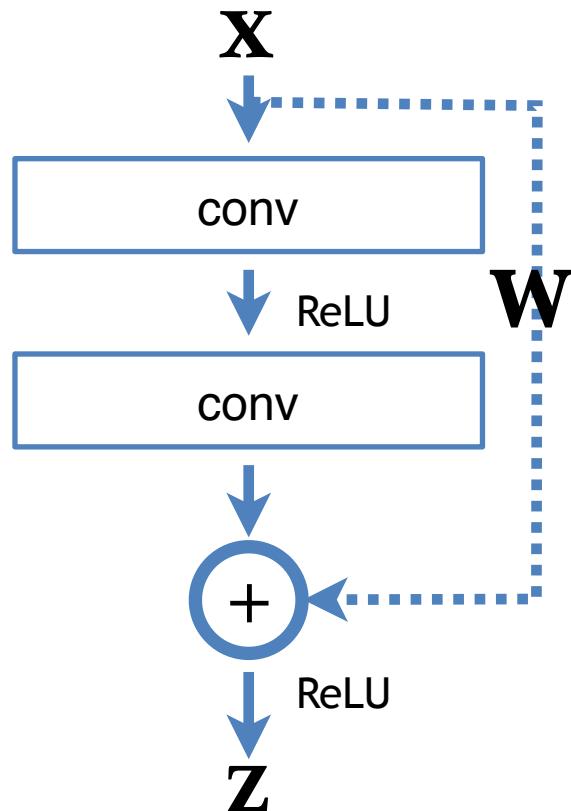
Convolution layerへの適用



ピクセルごとの足し合わせでOK

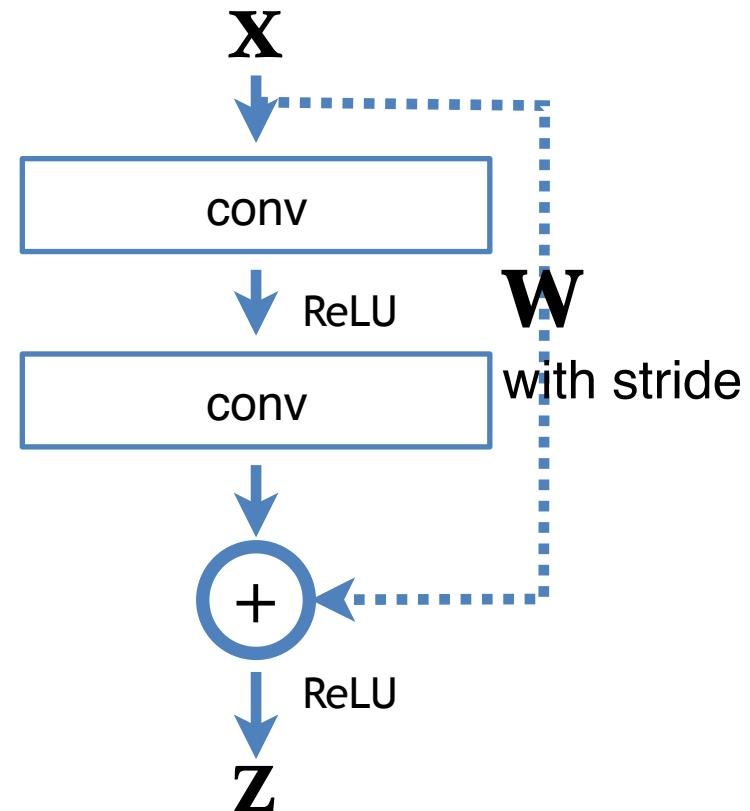
Convolution layerへの適用

convを経て
次元数が変わる場合



1x1 convで線形変換

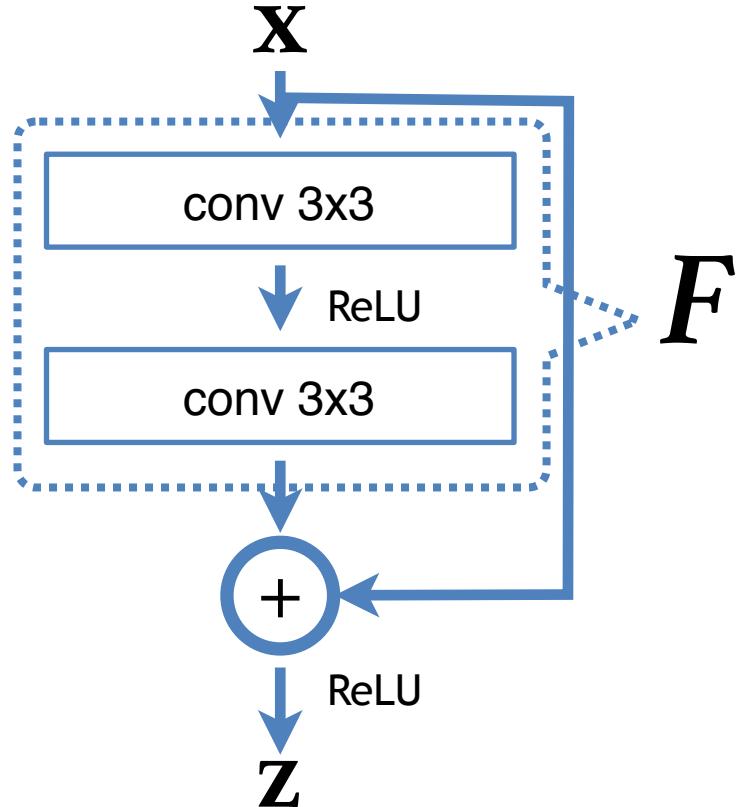
convを経て
マップサイズが変わる場合



適当なstrideで1x1 conv

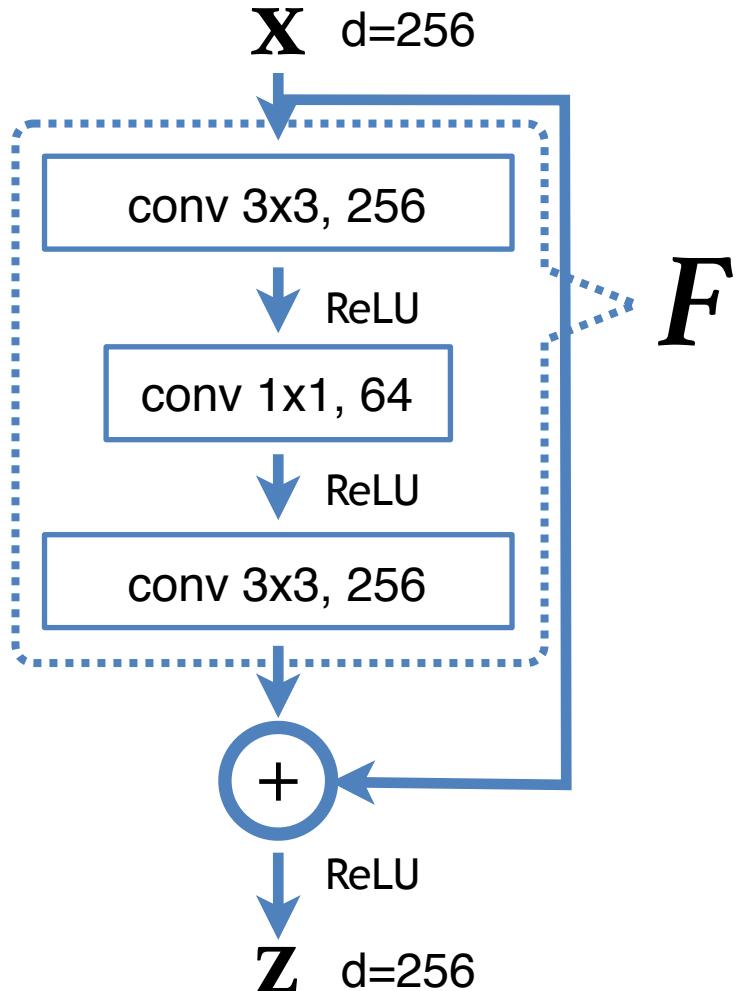
Going deeper with residual learning...

81



Going deeper with residual learning...

82



“bottleneck” building block

1x1 convで次元削減＆復元

- より Deep に
- パラメータ数はあまり変わらない

ResNet architectures

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1			average pool, 1000-d fc, softmax		
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)



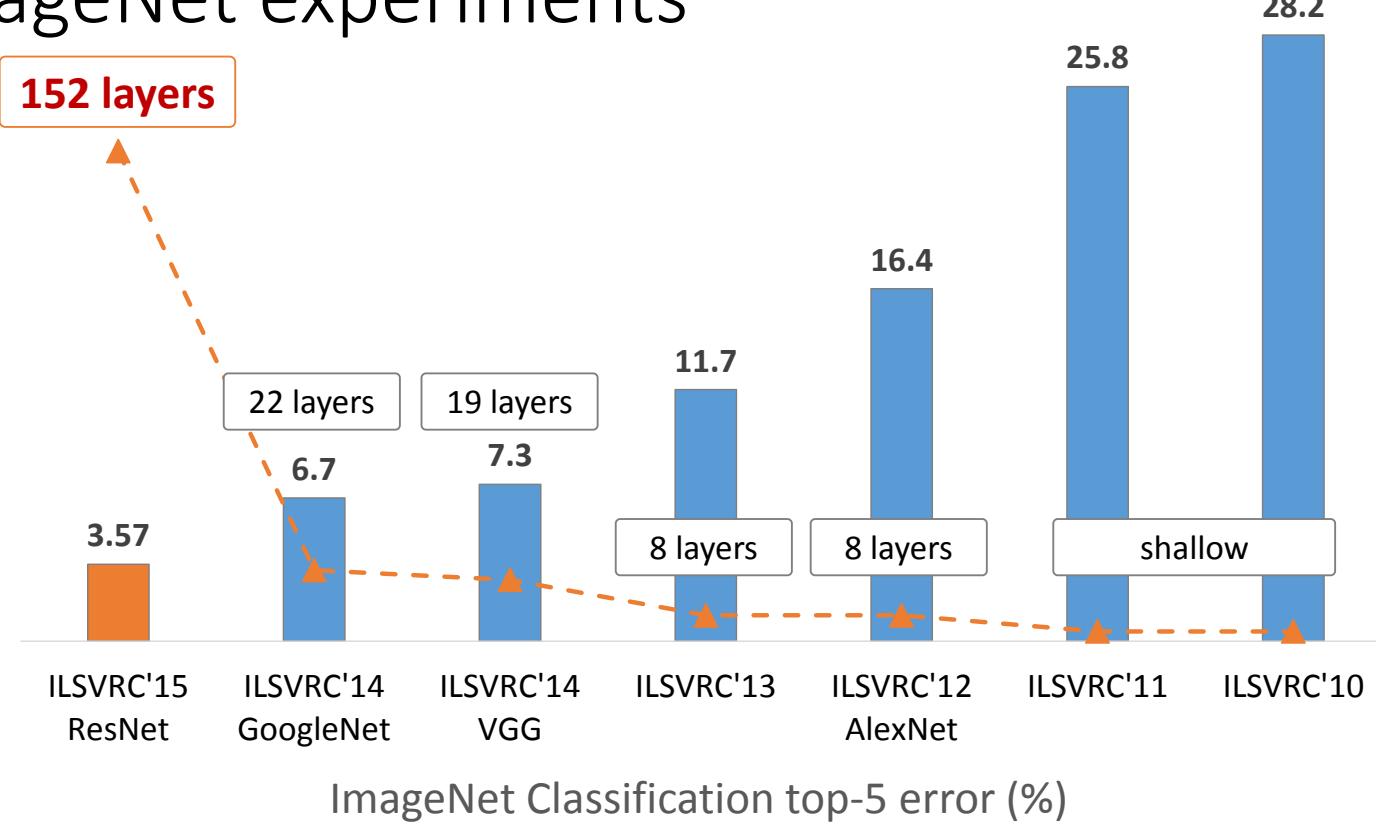
VGG, 19 layers
(ILSVRC 2014)



ResNet, **152 layers**
(ILSVRC 2015)



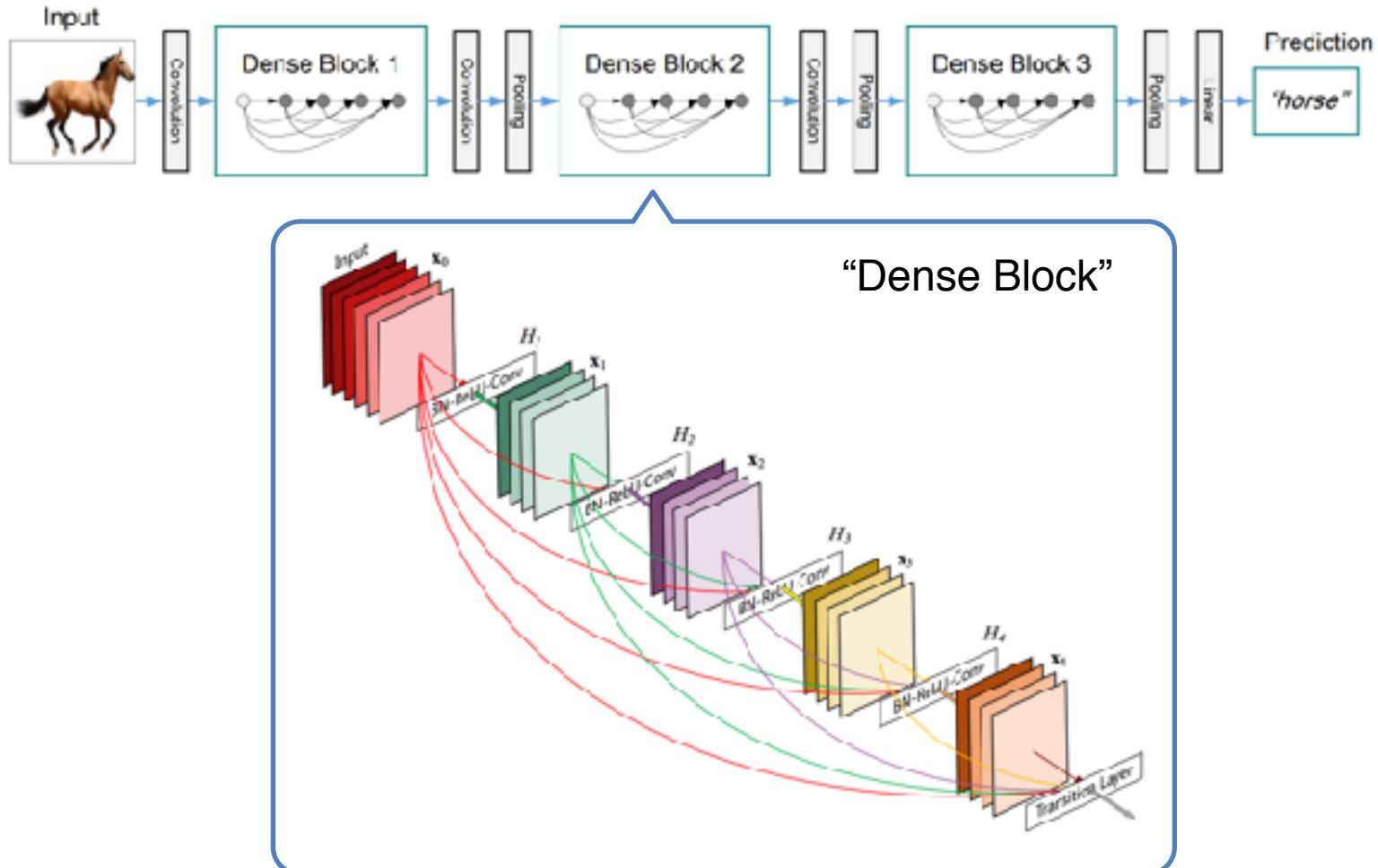
ImageNet experiments



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

その後

Dense Net [Gao Huang et al., 2016]



直前の層以外の情報も使える

→1層あたりのチャンネル数は少なくて済む

Fractal Net [Gustav Larsson et al., 2016]

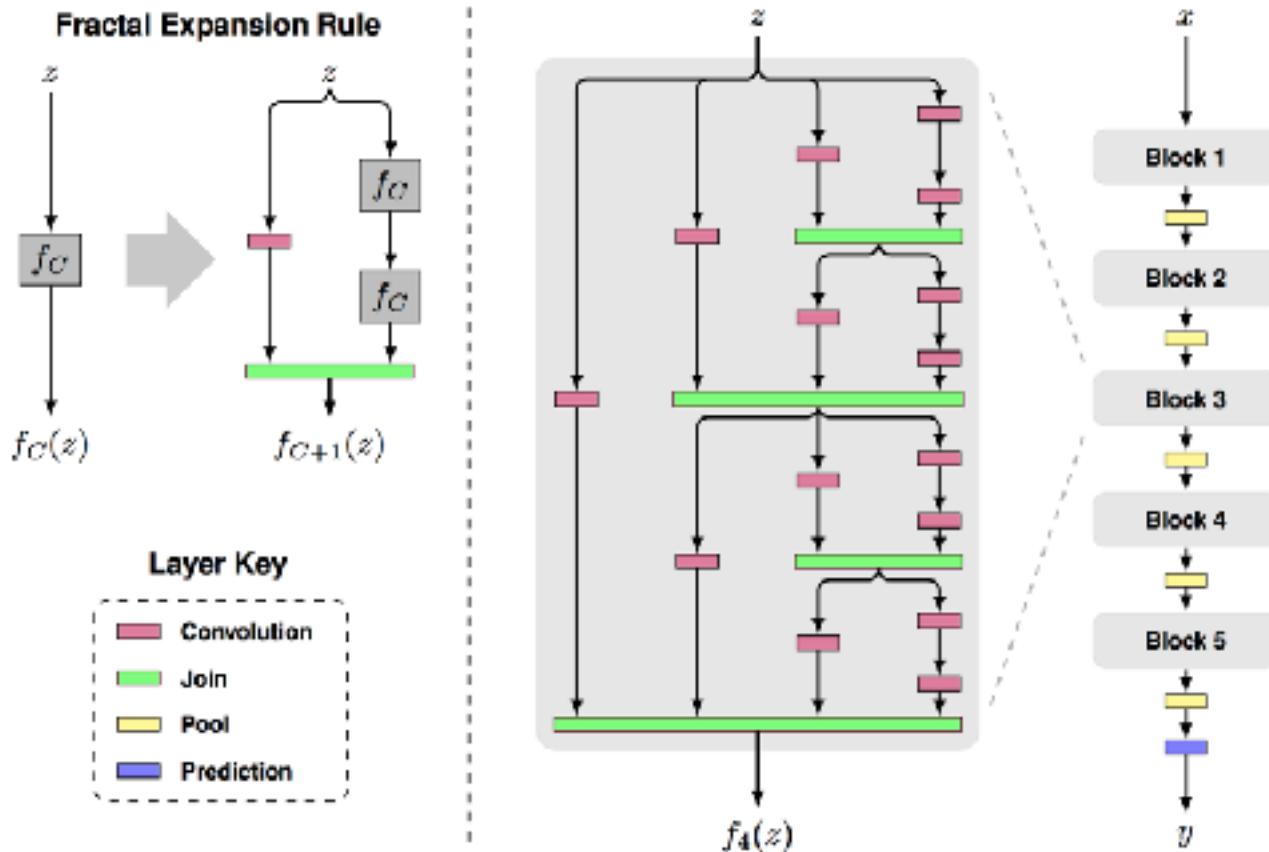
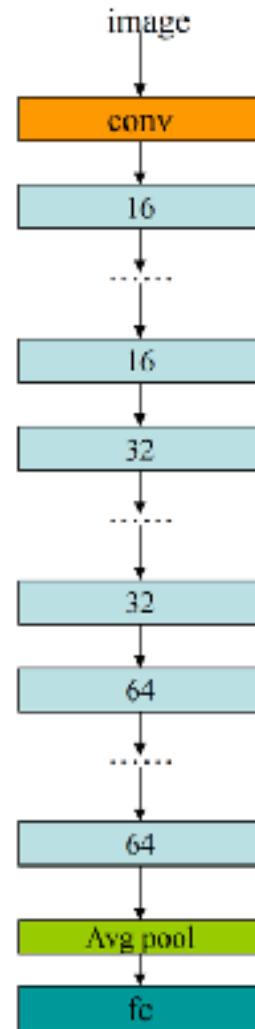


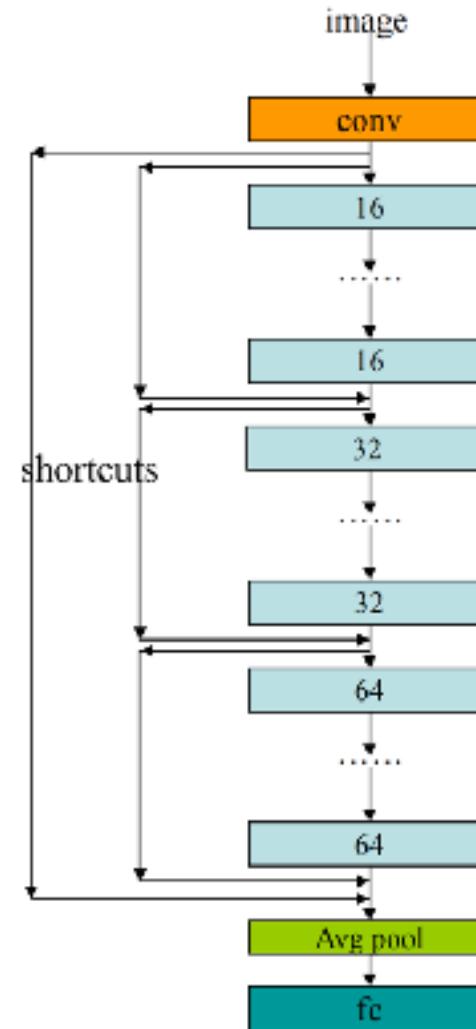
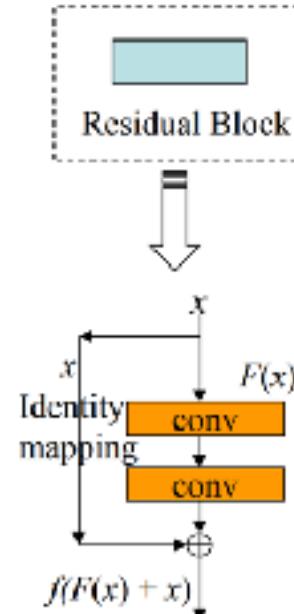
Figure 1: **Fractal architecture.** *Left:* A simple expansion rule generates a fractal architecture with C intertwined columns. The base case, $f_1(z)$, has a single layer of the chosen type (e.g. convolutional) between input and output. Join layers compute element-wise mean. *Right:* Deep convolutional networks periodically reduce spatial resolution via pooling. A fractal version uses f_C as a building block between pooling layers. Stacking B such blocks yields a network whose total depth, measured in terms of convolution layers, is $B \cdot 2^{C-1}$. This example has depth 40 ($B = 5, C = 4$).

Residual Networks of Residual Networks

[Ke Zhang et al., 2016]



Residual Networks



RoR

CNNと視覚情報処理の関係

単純型細胞と複雑型細胞

Hubel and Wieselによる知見:

V1は単純型細胞・複雑型細胞から構成される

- 単純型細胞
 - 方位選択性
 - 複雑型細胞
 - 単純型細胞の収斂
 - より広い受容野

生物の視覚モデルに基づいた物体認識手法

92

Neocognitron [Fukushima, 1980]

S細胞 (simple-cell)

≒ Convolutional Layer

特定のシンプルな刺激（向きなど）に反応

→ 特徴抽出

C細胞 (complex-cell)

≒ Pooling Layer

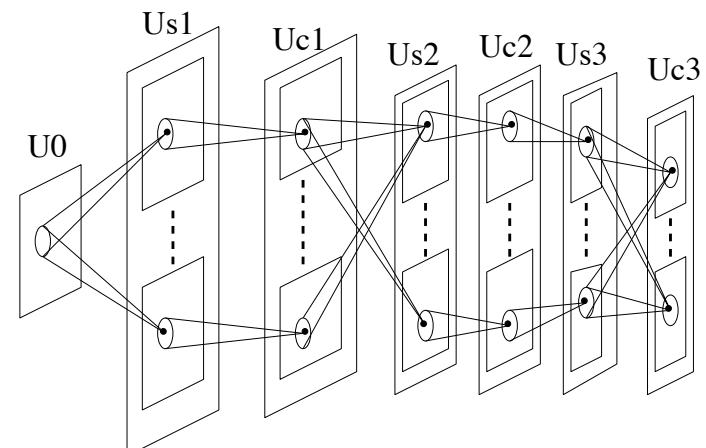
前の層の特定の範囲にあるS細胞に結合。

うち 1 つでも ONなら、ON

→ 不変性の獲得

Hubel and Wieselによる知見：

V1は単純型細胞・複雑型細胞から構成される



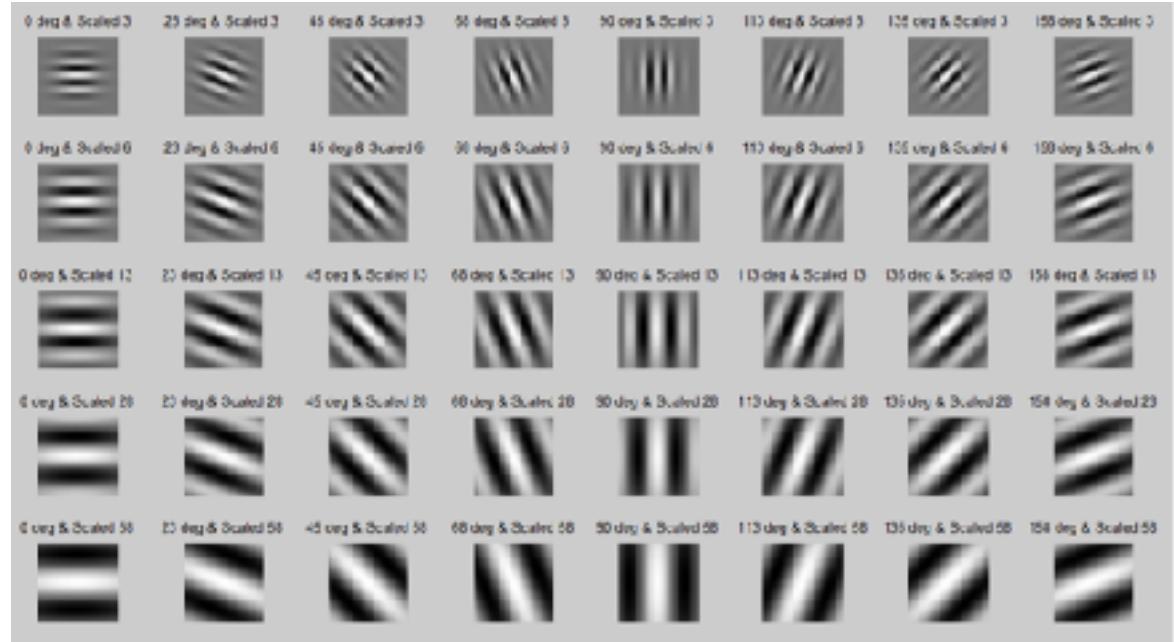
Gabor filter

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right)$$

$$x' = x \cos \theta + y \sin \theta \quad y' = -x \sin \theta + y \cos \theta$$

Gaussian × Sine wave

- 角度 θ
- スケール σ



単純型細胞 ≈ Gabor filter

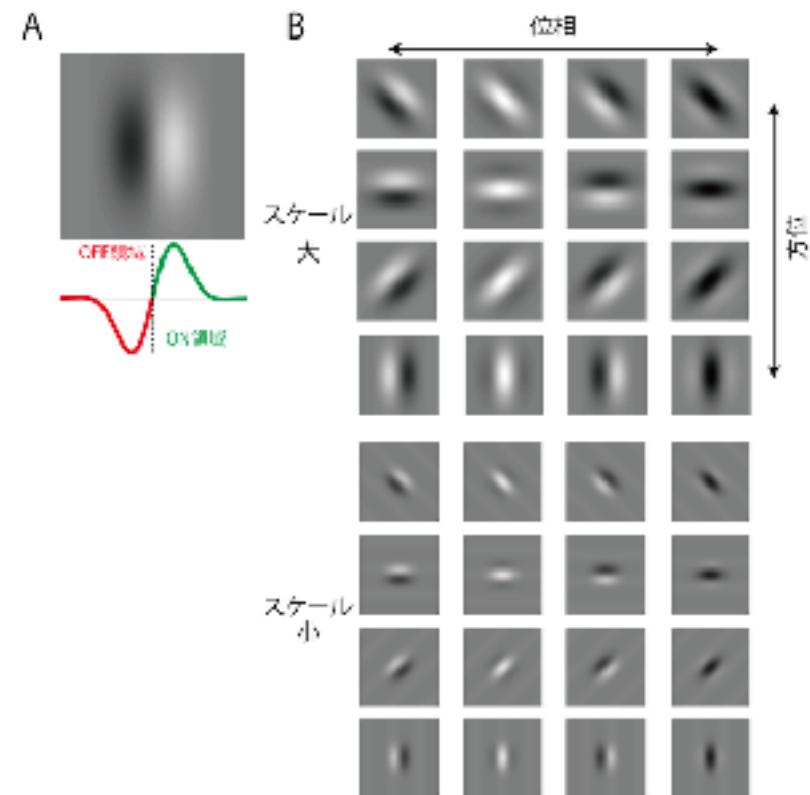
V1の単純型細胞はGabor filterで近似できる

- 角度 θ

- スケール σ

を調節することで、

様々な選択性の単純型細胞を近似

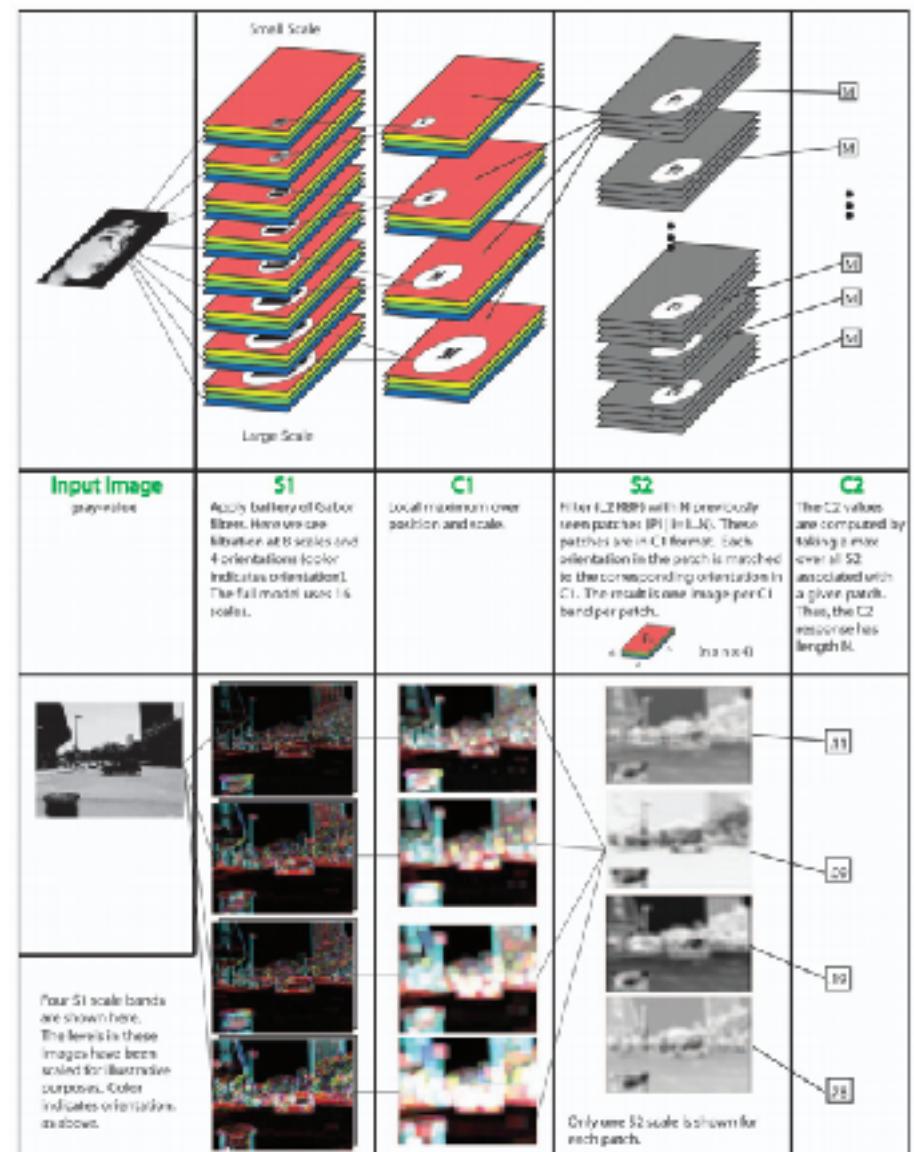


Gabor filterによる画像認識

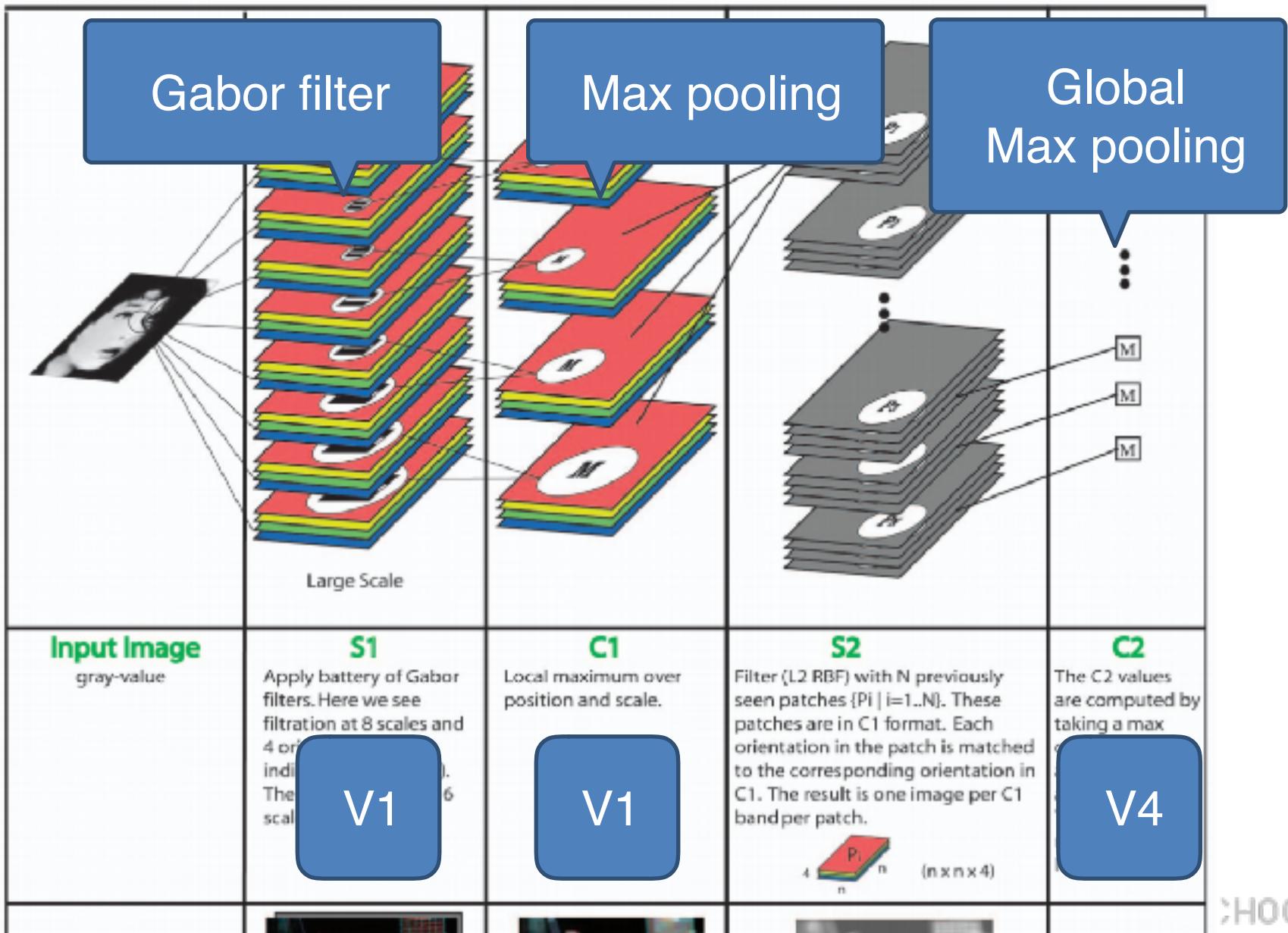
例:

腹側皮質視覚路に基づいたモデル

[Thomas Serre et. al., 2007]



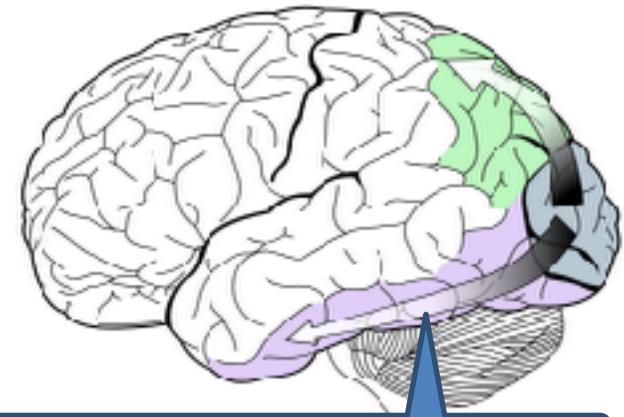
Gabor filterによる画像認識



腹側皮質視覚路

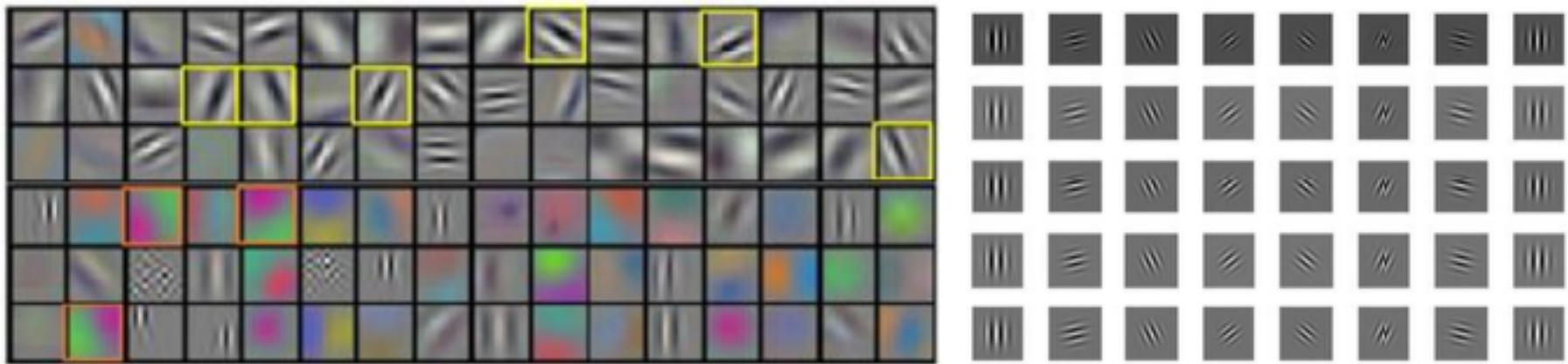
[Thomas Serre et. al., 2007] が模倣しようとした腹側皮質視覚路の性質

- ▶ 階層構造による不变性の獲得
- ▶ 階層が深くなるに従い、受容野は拡大し、より複雑な刺激に反応するようになる
- ▶ Immediate recognition taskでは、視覚情報はFeedforward
- ▶ 可塑性の獲得、学習はあらゆる階層で起こり、特に深い階層で顕著
 - ▶ Inferotemporal cortex (IT) / 下側頭葉
 - ▶ Prefrontal cortex (PFC) / 前頭前皮質



Ventral stream
腹側皮質視覚路

Gabor filter vs CNN特徴



CNNによって獲得される特徴の中には、

Gabor filterのようなものもある

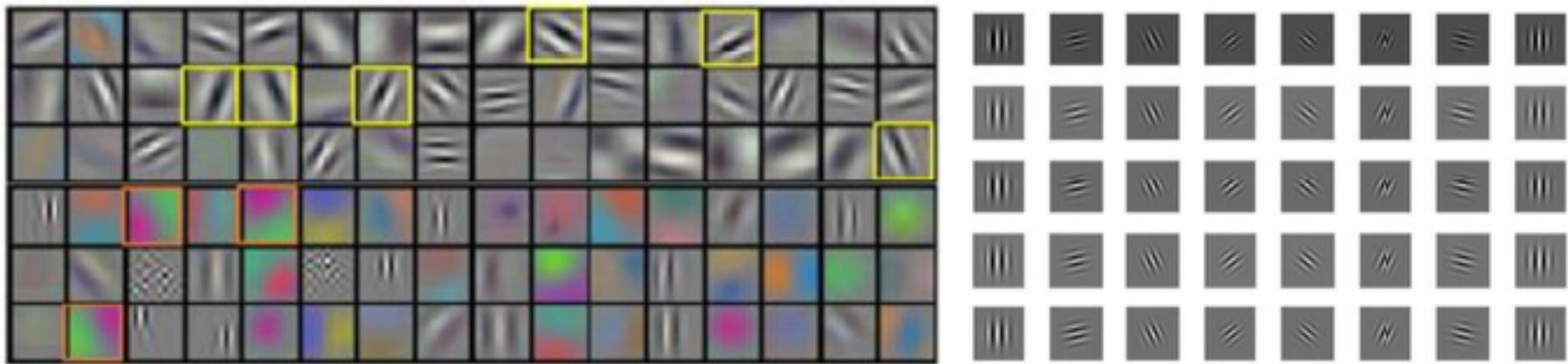
but, CNN特徴には、Gabor filterにはある

- 回転

- スケール

に対するパラメータが無い

Gabor filter vs CNN特徴



Gabor convolutional networks [Shangzhen Luan et. al., 2017]:
CNNのConvolutional layerをGabor filter化

脳はBackPropagationしているか?¹⁰⁰

No

... Inspired by the **biological implausibility** of back-propagation,
a few approaches have been proposed in the past
that could play a similar credit assignment role. ...
[Dong-Hyun Lee et. al., 2014]

“生物学的な妥当性はない”

▶ より生物学的な学習手法？

e.g. Difference Target Propagation [Dong-Hyun Lee et. al., 2014]

Adversarial Attack



x
“panda”
57.7% confidence

$+ .007 \times$



$\text{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y))$
“nematode”
8.2% confidence

=



$\mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y))$
“gibbon”
99.3 % confidence

[Goodfellow et al., 2014]

ニューラルネットワークを”騙す”

Adversarial Patch [Tom B. Brown et al., 2017]

<https://arxiv.org/abs/1712.09665>



<https://www.youtube.com/watch?v=i1sp4X57TL4>

畳み込みニューラルネットワークの研究動向

<https://www.slideshare.net/ren4yu/ss-84282514>

演習

CIFAR10 classification