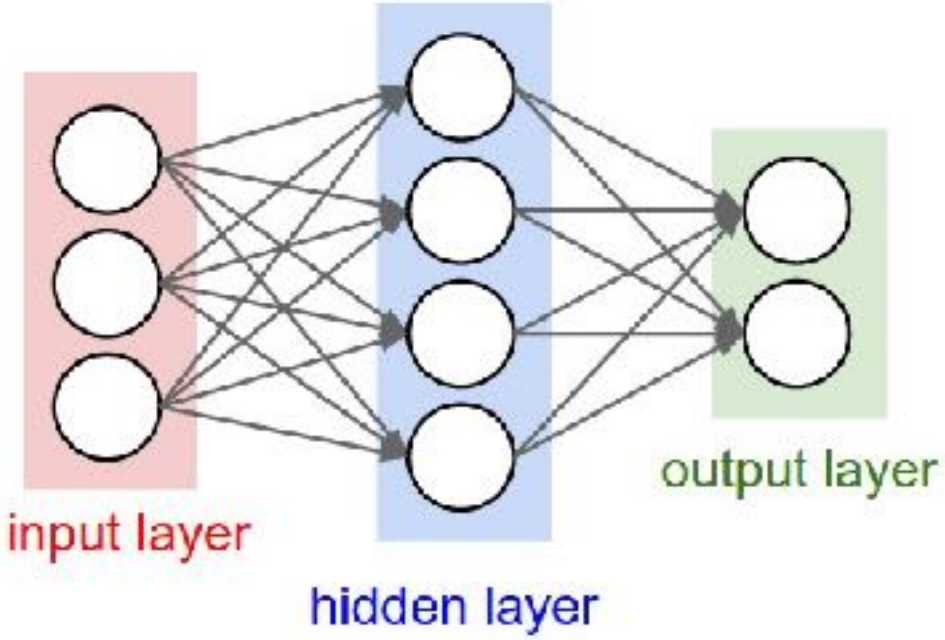
NICO2AI #6
畳み込みニューラルネットワーク(1)

18/02/17

土屋祐一郎

前回：全結合NN

例: 3 layers perceptron



http://cs231n.github.io/assets/nn1/neural_net.jpeg

3layers MLPの復習

- 演習

MLPをより高性能にしたい

- MLPは万能の関数近似器

  (universal function approximator)
  - 十分な数のパラメータ（＝ニューロン結合）があれば、任意の関数を任意の精度で近似できる
- ⇒層の数を増やそう！ (Going deeper!)
  - 浅いネットワークでニューロン数を増やすより効率が良い

  [Larochelle et al., 2007][Bengio, 2009][Delalleau and Bengio, 2011]
  - （単純なケース以外では証明されていない）

Going deeper, but…

- 層の数を増やせば性能は上がるはず
  - 「“適切な”パラメータを見つけられれば」…
- しかし、現実にはうまくいかない…
  - 学習がうまくいかない

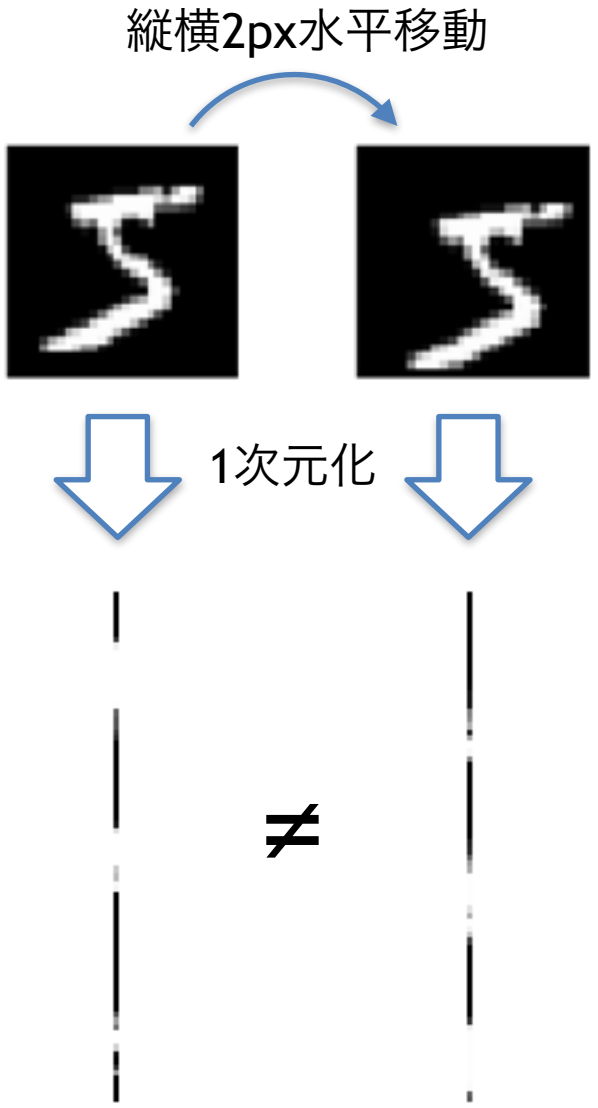
全結合NNをDeepにするときの問題

4, 5, …, 10, … layers MLPを作ると…

- 過学習
  - パラメータ多すぎ
- 学習が進まない
  - パラメータ多すぎ
- 計算が重い
  - パラメータ多すぎ

画像への適用時の問題

- 水平移動に弱い
- 回転に弱い
- affine transformに弱い



Convolutional Neural Networks

- 全結合NNの問題（の一部）を解決
  - パラメータ数削減
  - 平行移動への対応

    など
- 画像認識タスクへの適用が最初のブレイクスルー
- 近年は自然言語処理への適用も

Convolutional Neural Networks
概観

- いきなりですが、最初に、

  Convolutional Neural Networks (CNN)
  がどういうものかざっくり説明します

- その後、
  - CNNがどうして全結合NNの問題を解決しているのか
  - 生物の視神経との関連はどうなっているのか

  …などについて話します

Convolution = Filtering

例：ラプラシアンフィルタ

1	1	1
1	−8	1
1	1	1

畳み込み（Convolution）！

Filtering = ピクセルごとの乗算→加算

例： 0×0 + 0×0 + 0×0
+ (-0.5)×0 + 0×0 + 0.5 × 1
+ 0×0 + 0×0 + 0×1
= 0.5

0	0	0
-0.5	0	0.5
0	0	0

を適用

0	0	0	0	0	0	0	0
0.5	0.5	0	0	0	0	-0.5	-0.5
0.5	0	0	0	0	0	0	-0.5
0.5	0	0	0	0	0	0	-0.5
0.5	0	0	0	0	0	0	-0.5
0.5	0	0	0	0	0	0	-0.5
0.5	0.5	0	0	0	0	-0.5	-0.5
0	0	0	0	0	0	0	0

Filtering = ピクセルごとの乗算→加算

例：

0	0	0
-0.5	0	0.5
0	0	0

を適用

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0.5	0.5	0	0	0	0	-0.5	-0.5
0.5	0	0	0	0	0	0	-0.5
0.5	0	0	0	0	0	0	-0.5
0.5	0	0	0	0	0	0	-0.5
0.5	0	0	0	0	0	0	-0.5
0.5	0.5	0	0	0	0	-0.5	-0.5
0	0	0	0	0	0	0	0

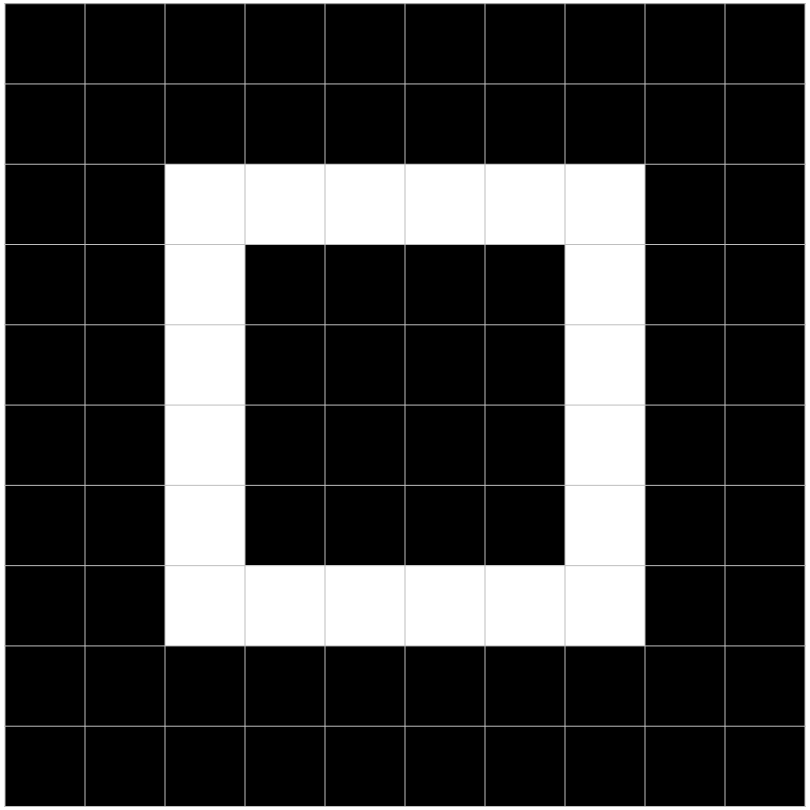
Filtering = ピクセルごとの乗算→加算

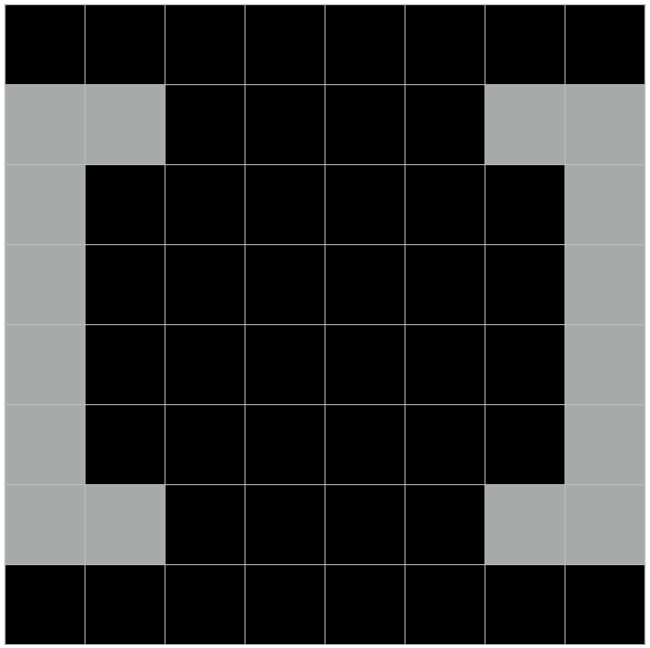
例：

0	0	0
-0.5	0	0.5
0	0	0

を適用

＝縦エッジ検出フィルタ

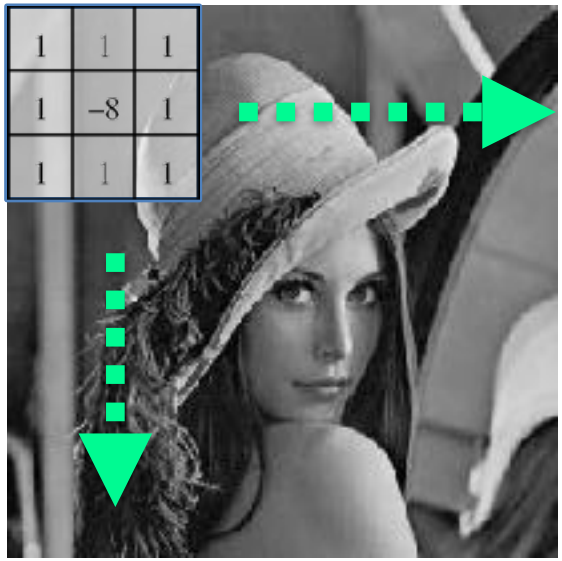


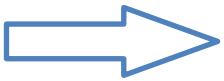


Convolution = Filtering

例：ラプラシアンフィルタ

1	1	1
1	−8	1
1	1	1

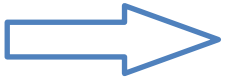




2D convolution

適当に学習されたフィルタ

0.01	-0.12	1.03
-1.11	0.12	-0.56
0.32	-0.34	-0.22

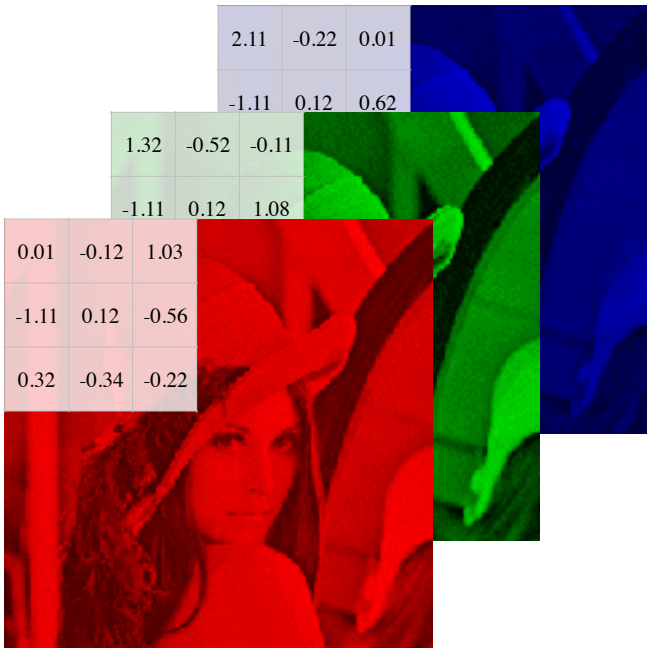


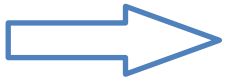
3D Convolution

適当に学習されたフィルタ for R,G,B

2.11	-0.22	0.01
1.32	-0.52	-0.11
0.01	-0.12	1.03
-1.11	0.12	-0.56
0.32	-0.34	-0.22

0.01	0.62	
1.08	0.98	
-1.00		

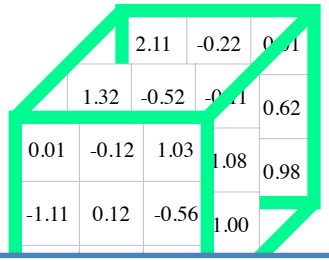




3D Convolution

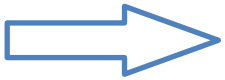
適当に学習されたフィルタ for R,G,B

＝適当に学習された3次元フィルタ



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

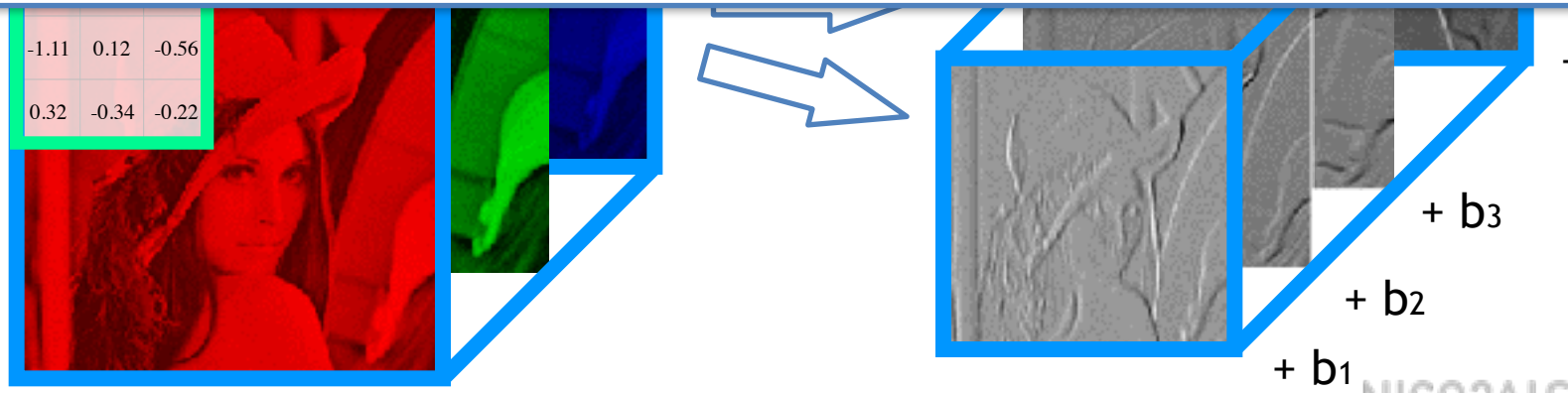


+ b

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

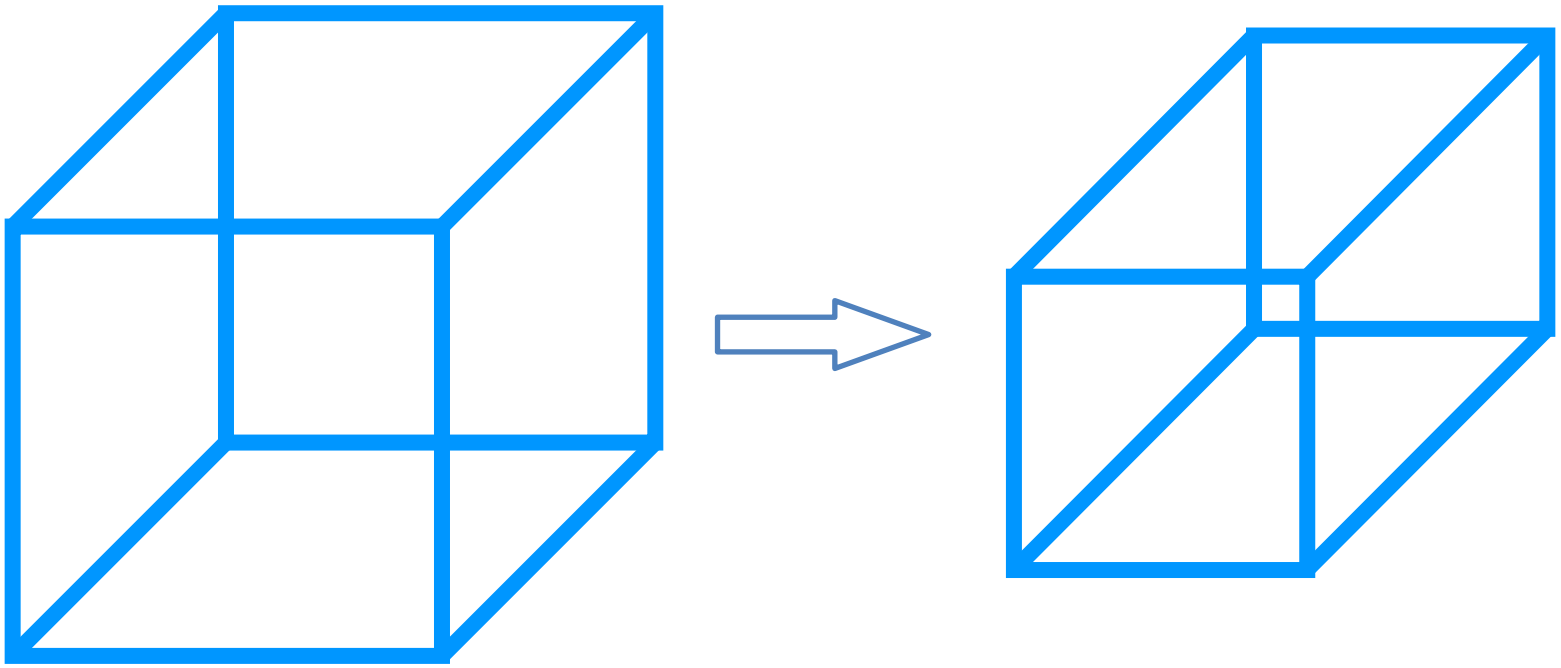


4D Convolution

簡略化して書くと...

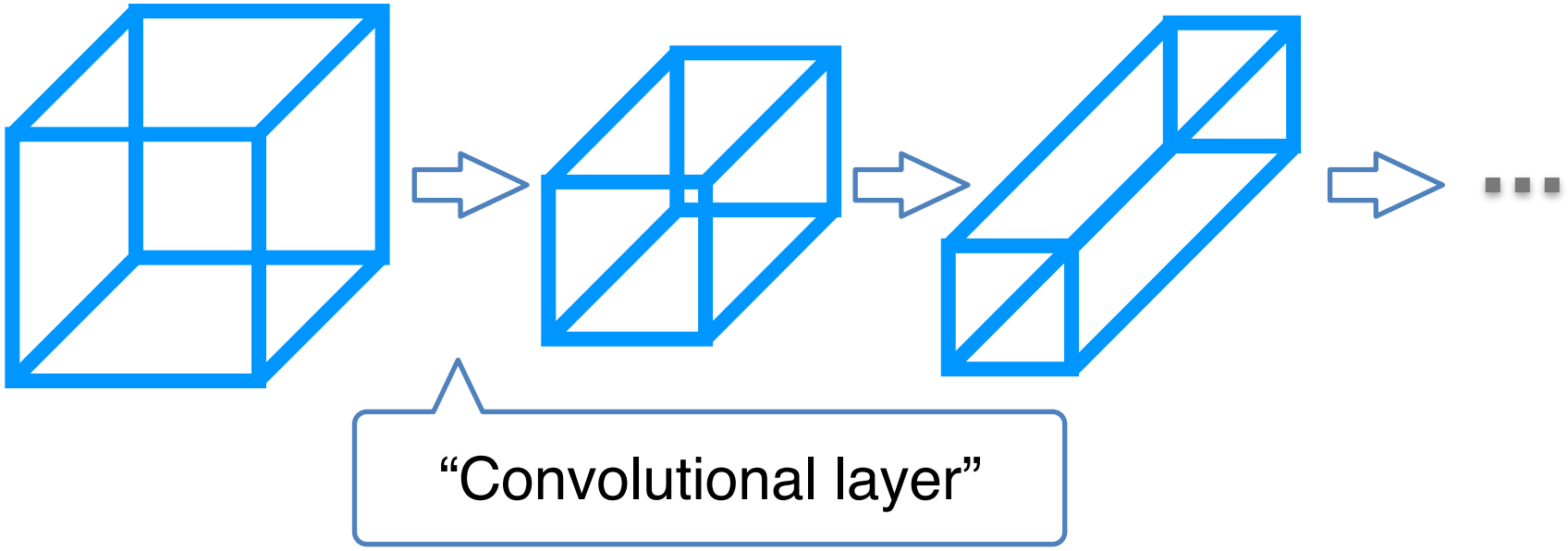
「3次元データに4次元フィルタを畳み込んで
3次元出力を得る」
のがCNNの基本処理

Convolutional Neural Networks

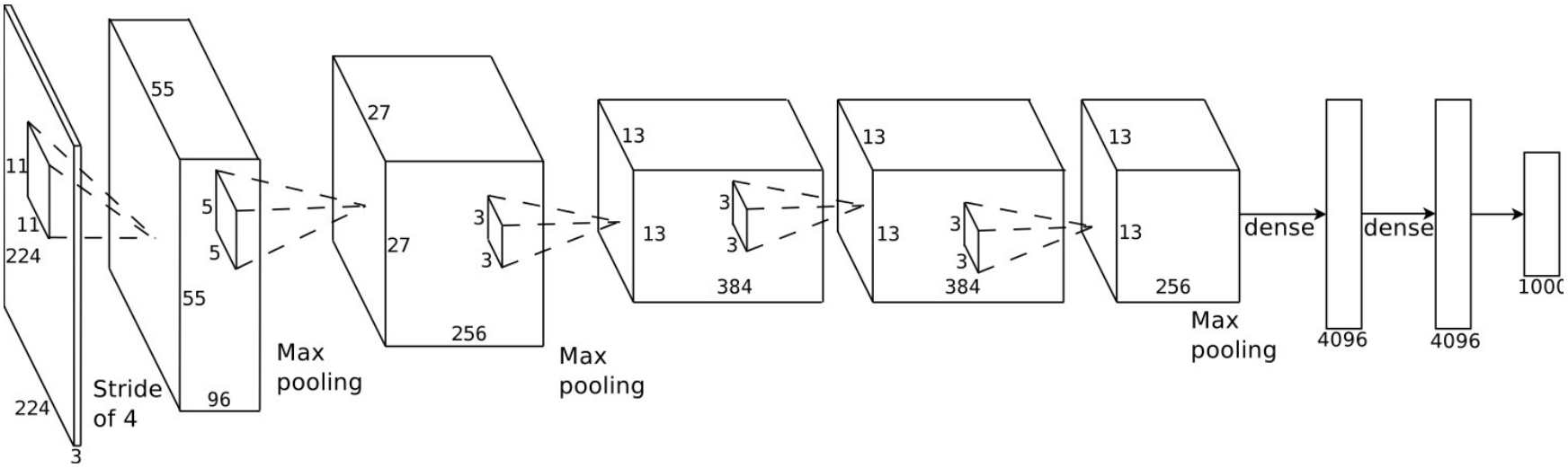


をたくさん繋げて...

Deep convolutional neural networks



AlexNet

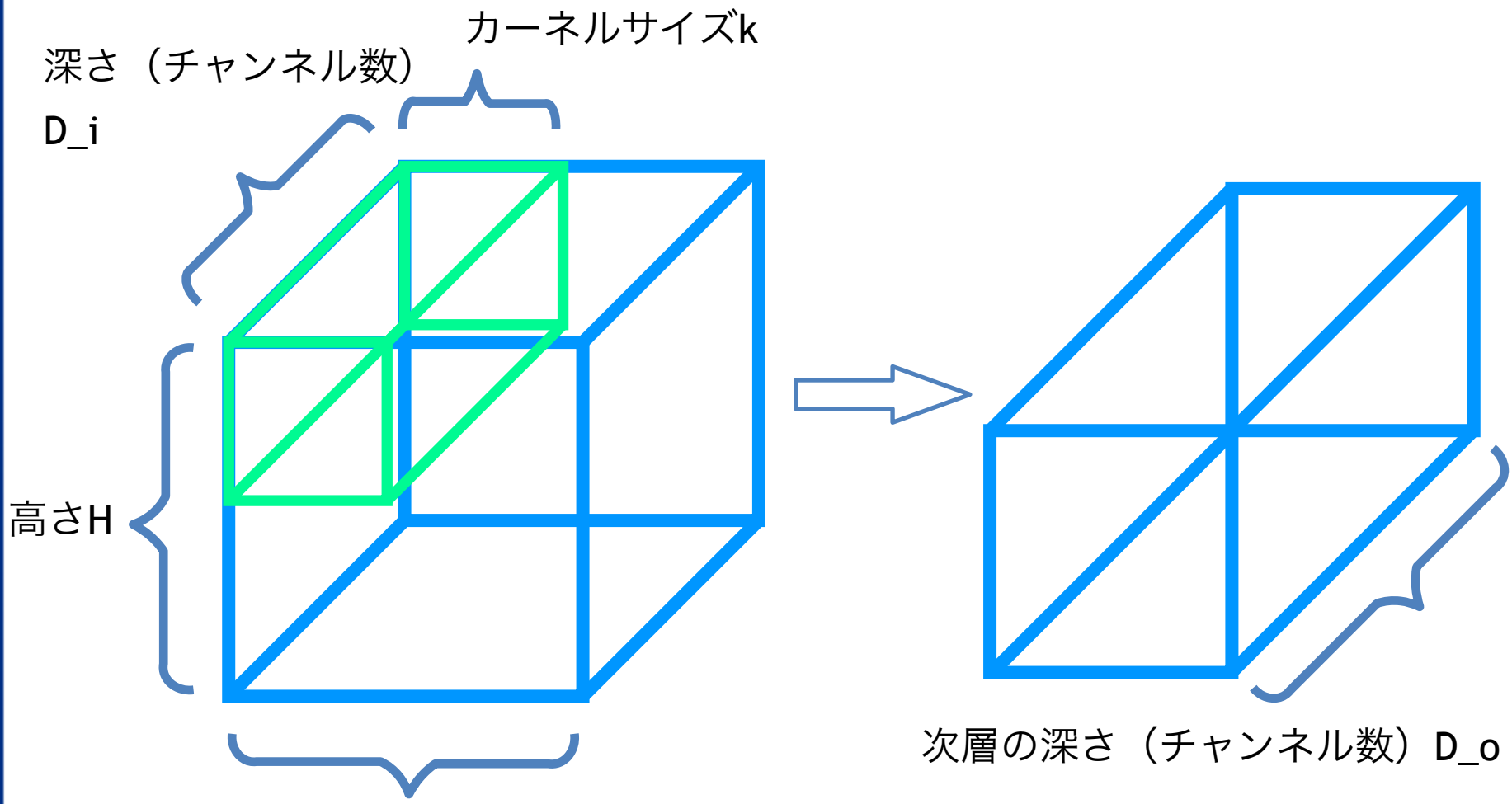


Krizhevsky et.al., 2012

大規模画像認識コンペティションILSVRC2012で優勝

Deep Learningブームの火付け役

DCNN各層のハイパーパラメータ



2層目以降のW, Hは、

その前の層のW, H, kから決まる

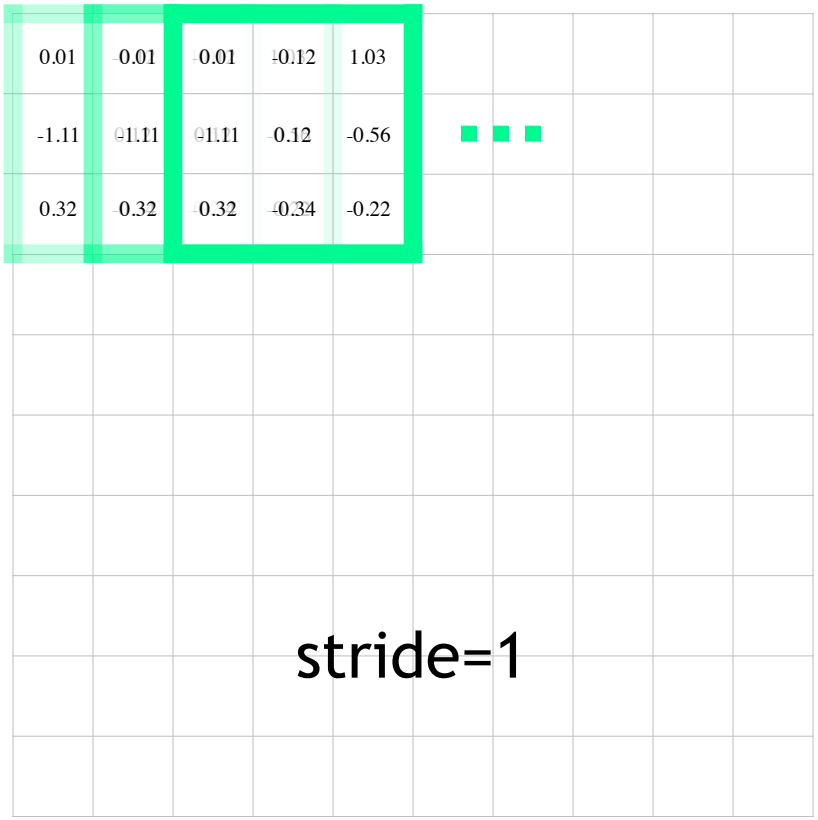
ここまでのまとめ

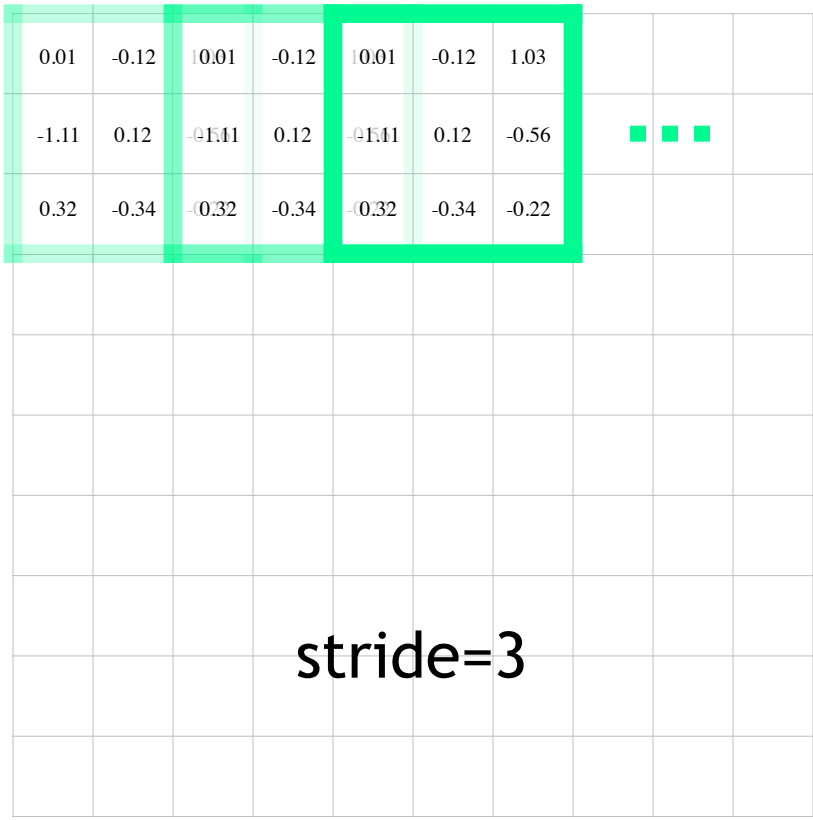
- Convolutional Neural Networks
  =Convolutional Layer (畳み込み層)を繋げたタイプのニューラルネットワーク
- Convolution=畳み込みは、フィルタ演算！
  - ただし、2次元画像へのフィルタ適用と違って、深さ方向も考える

Convolutional Neural Networks の細かい話

Stride

Stride = フィルタを何ピクセルずつずらしていくか





Padding

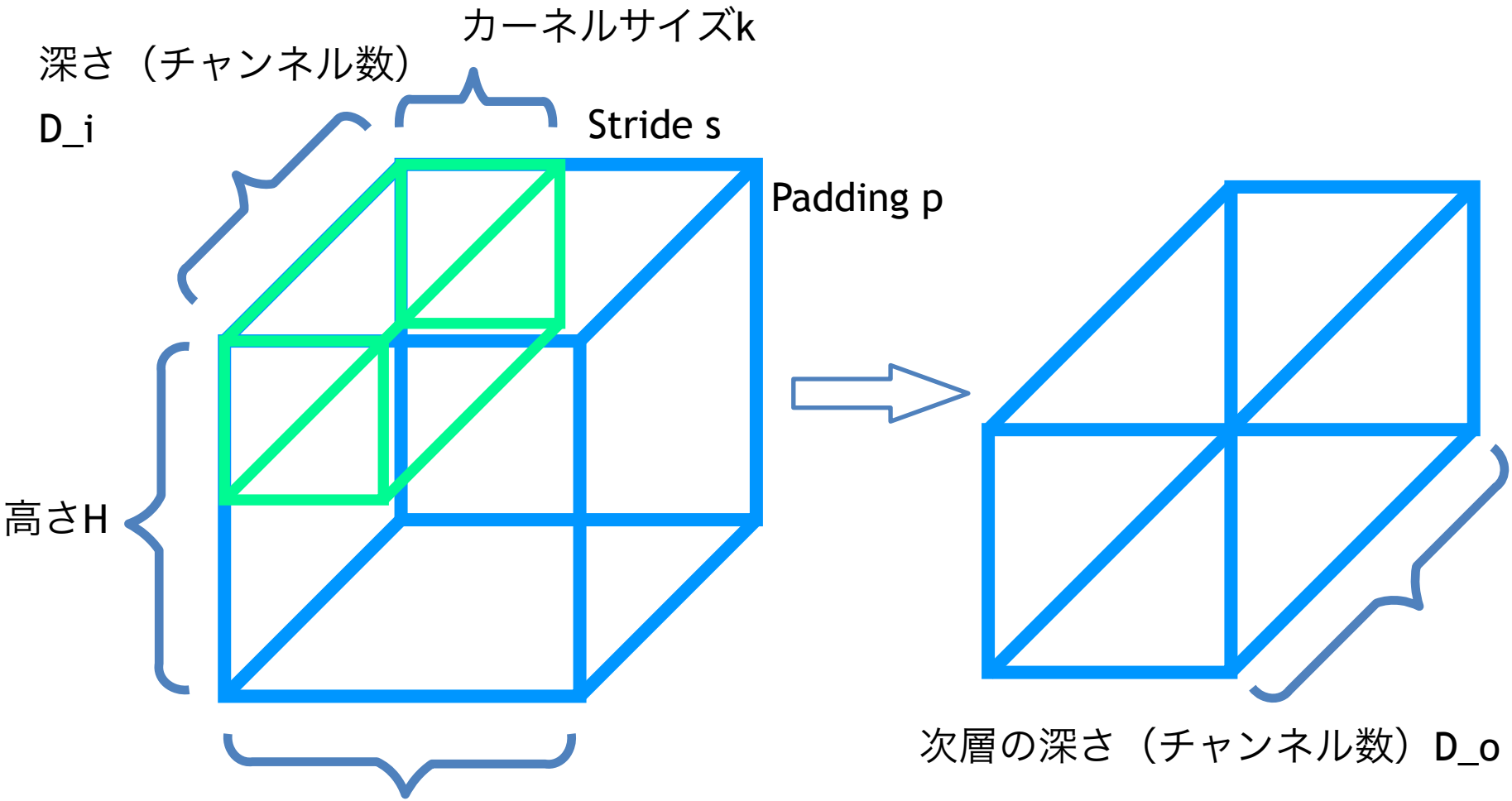
Padding = 入力の上下左右をゼロで埋めて広げる

→画像のエッジ部分の情報を適切に扱えるようになる

例：padding=2

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	101	148	156	129	179	168	201
0	0	130	149	176	198	139	189	112
0	0	146	176	129	182	194	127	182

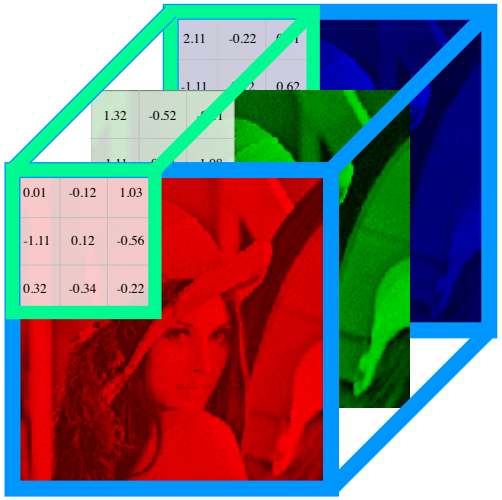
（再掲）DCNN各層のハイパーパラメータ



2層目以降のW, Hは、

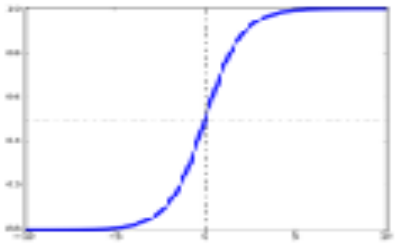
その前の層のW, H, kから決まる

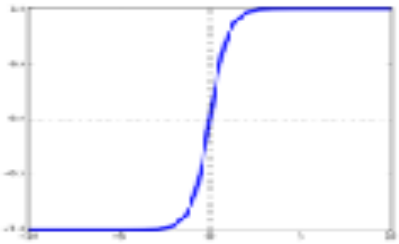
活性化関数

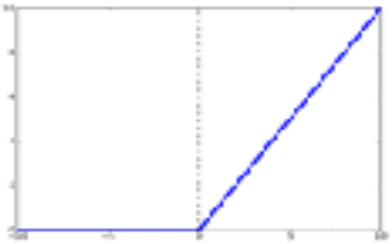


$$f(x)$$

- Sigmoid
- tanh
- ReLU

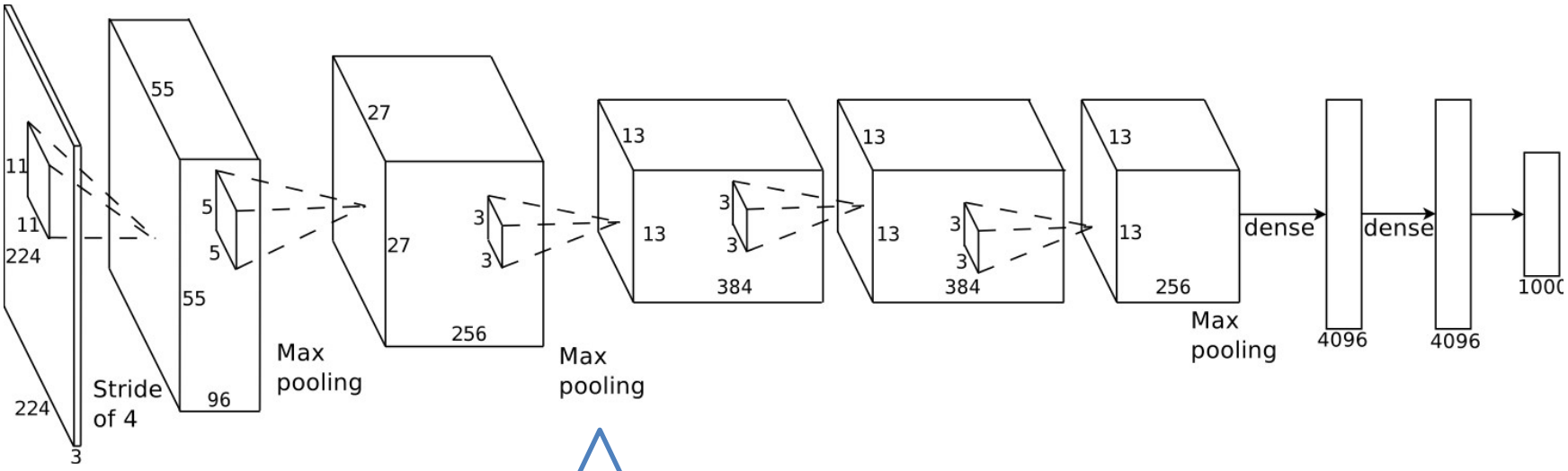






https://thinkit.co.jp/sites/default/files/639904.png

Convolutional layer以外のlayer

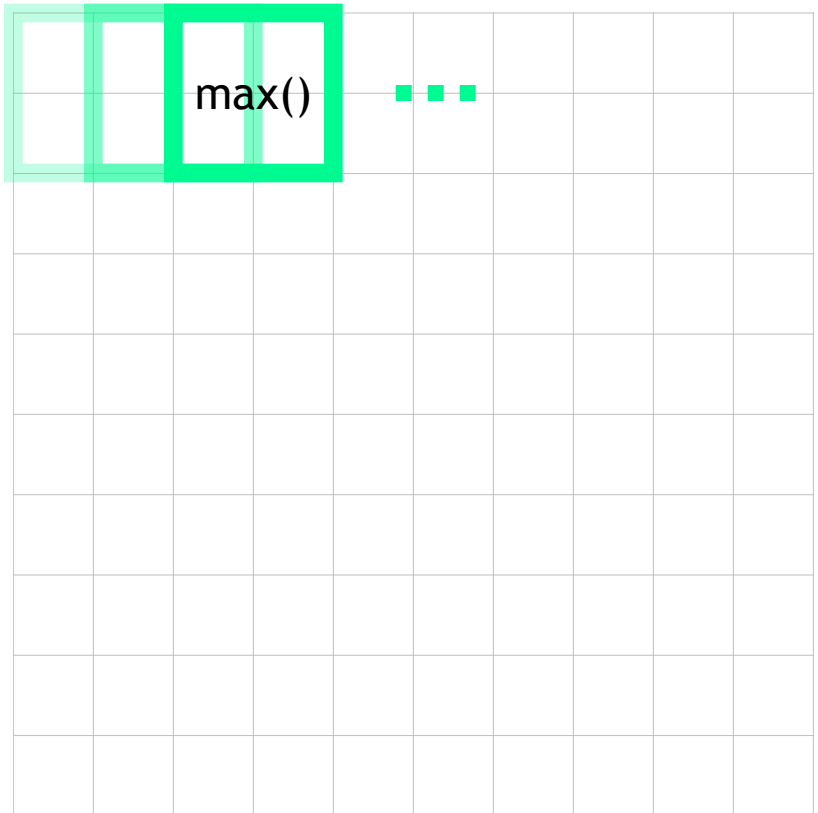


“Max pooling”

Max pooling

前の層にmax()フィルタをかける

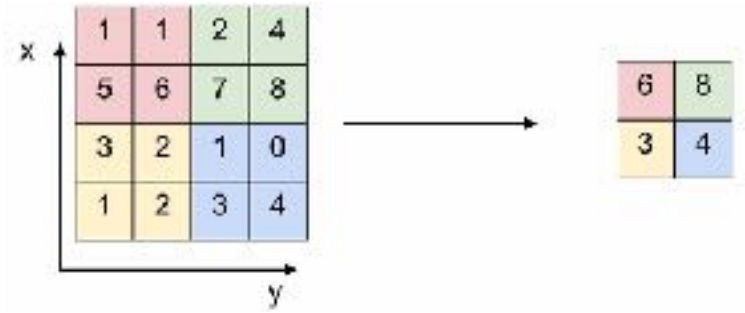
=適用範囲のうち、最大値のみをとり、残りを切り捨てる



ハイパーパラメータ：

- カーネルサイズk
- Stride s

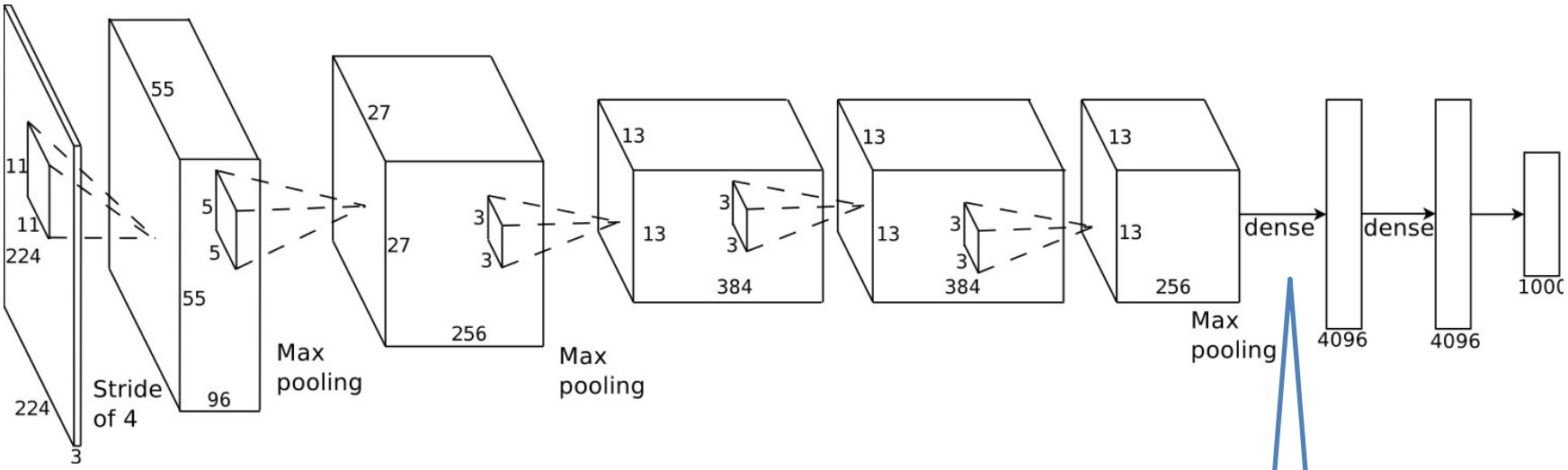
例: k=2, s=2



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)

CNNの学習

誤差逆伝播 Back propagation

- CNNの学習もBPによって行う
- Convolutional Layer
  - 前の層の影響するピクセルに
    誤差を蓄積させていく
    - 逆畳み込みのような感じ
- Pooling Layer
  - 前の層の影響するピクセルを覚えておいて、
    誤差を逆伝播させる
    - (chainerが良しなにやってくれます)

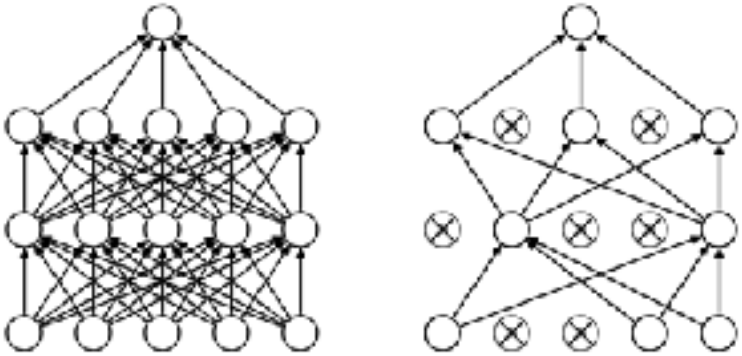
CNNの性能を上げるための様々なテクニック

Data augmentation

- CNNはPoolingによって平行移動には強くなった（平行移動不変性を獲得した）
- But, 回転不変性・鏡像不変性・affine変換不変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

Dropout [N. Srivastava et al., 2014]

- 学習時：ニューロンを一定の確率pで無視する
- 推論時：全ニューロンを使い、結合重みをp倍する



- 同時に複数のネットワークを学習し、
  平均を取ったのと似た効果が得られる
  （アンサンブル学習）
  →正則化と同様の効果＝過学習耐性

Mean normalization

- データセット全体のRGB毎のピクセル値の平均を計算しておき、入力画像から引く
  - 各(x,y)座標毎に平均をとったり、
    x,yは無視して画像全体で平均をとったり

CNNの定性的な特徴

CNNの定性的な特徴

- パラメータ数が少ない
  - Weight sharing
    - 同じ重み(フィルタ)を複数領域で共有
    - 対象ドメイン（画像）のlocalityが前提
  - 過学習しづらい
  - 計算が楽
- 平行移動不変性
  - pooling

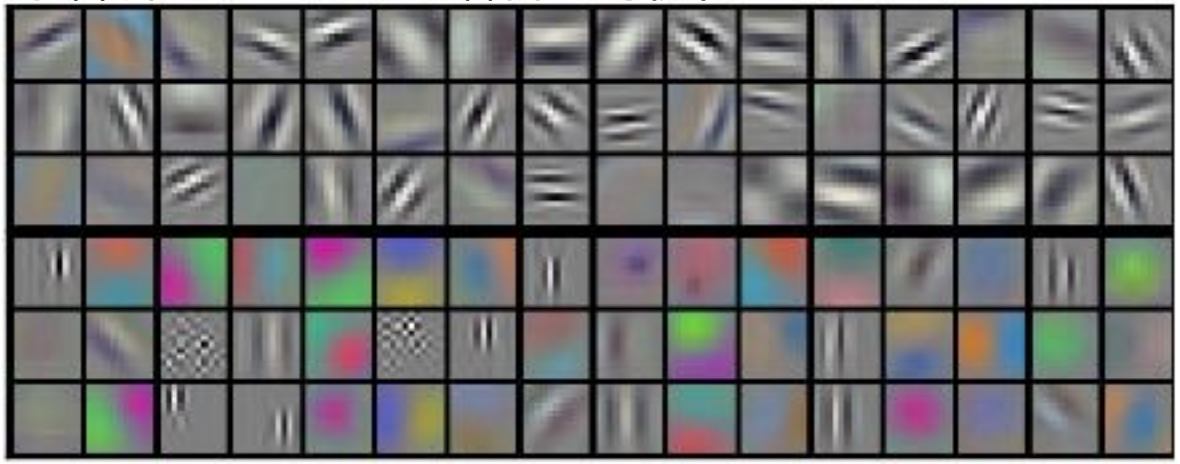
フィルタの自動獲得

（初期値は乱数で与えているのに）

CNNを学習させると、

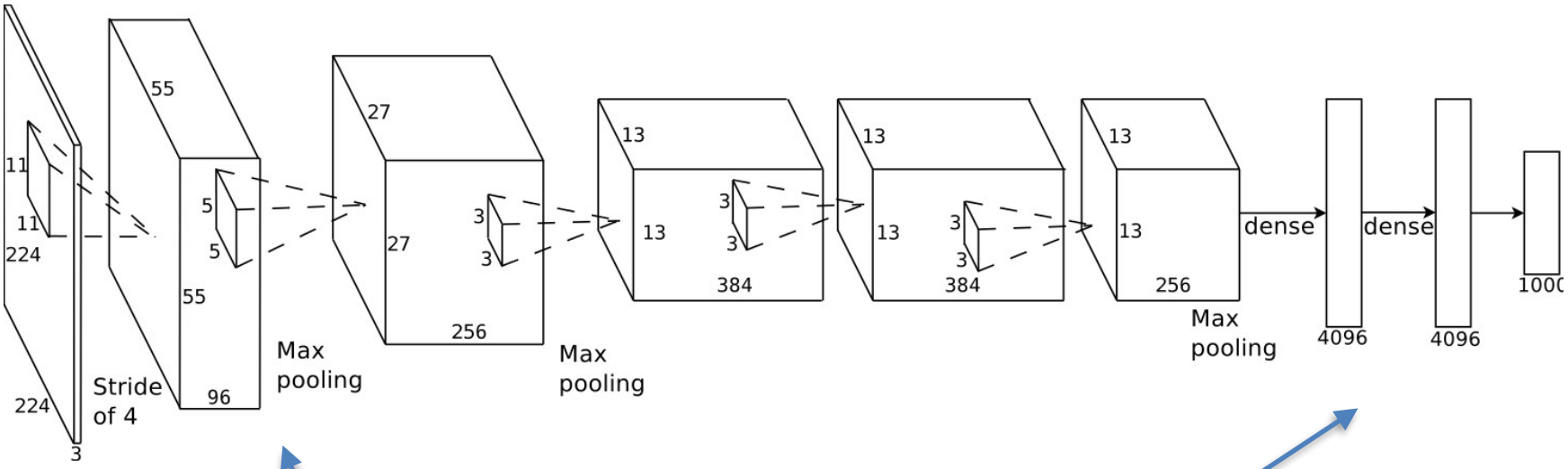
識別に有効なフィルタが自動的に獲得される

学習済みAlexNetの1層目の可視化



Krizhevsky et.al., 2012

End-to-Endの画像識別器



前半を特徴抽出器、

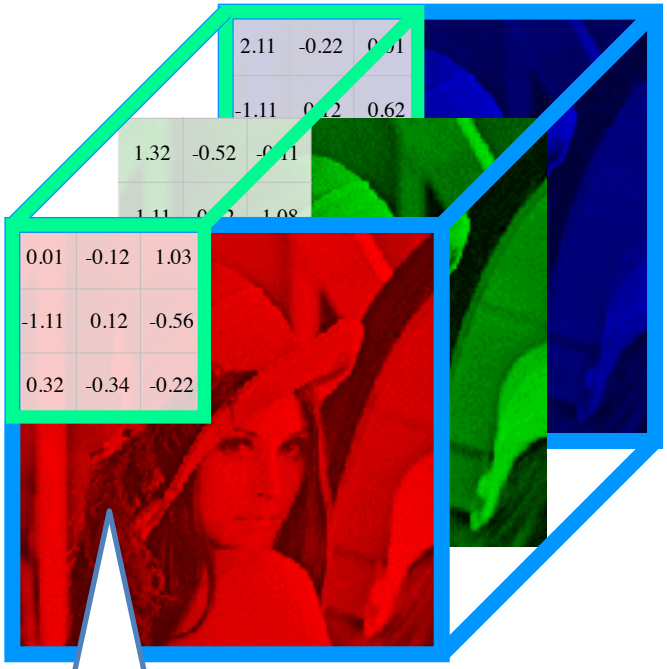
後半を識別器と見ることもできる

→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

特徴抽出器としてのCNN

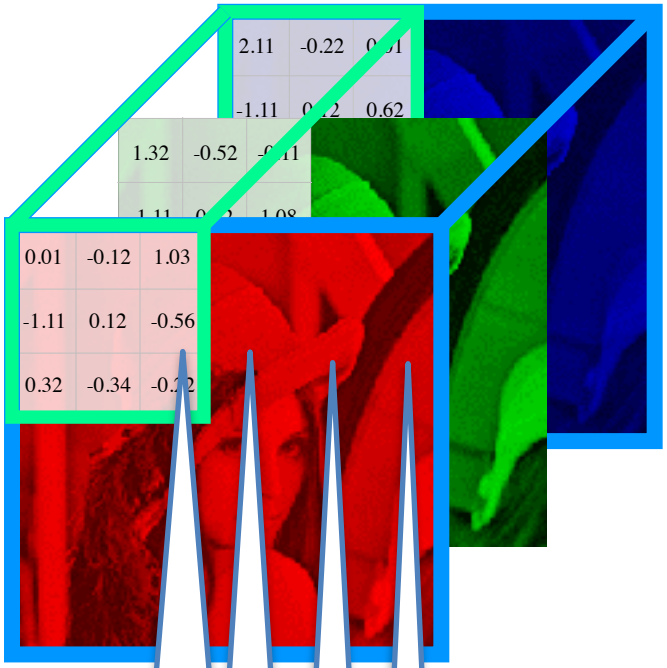
フィルタ＝特徴抽出

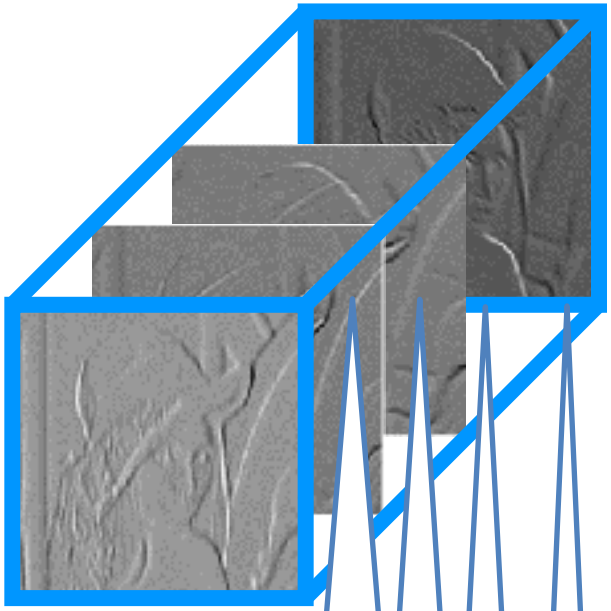


たとえば、
エッジに強く
反応するフィルタ

エッジが強調された
画像

各層の出力＝特徴マップ





エッジ（細）

エッジ（太）

Rチャネルのエッジに強く反応するフィルタ

～みたいな曲線に強く反応するフィルタ

“特徴マップ”

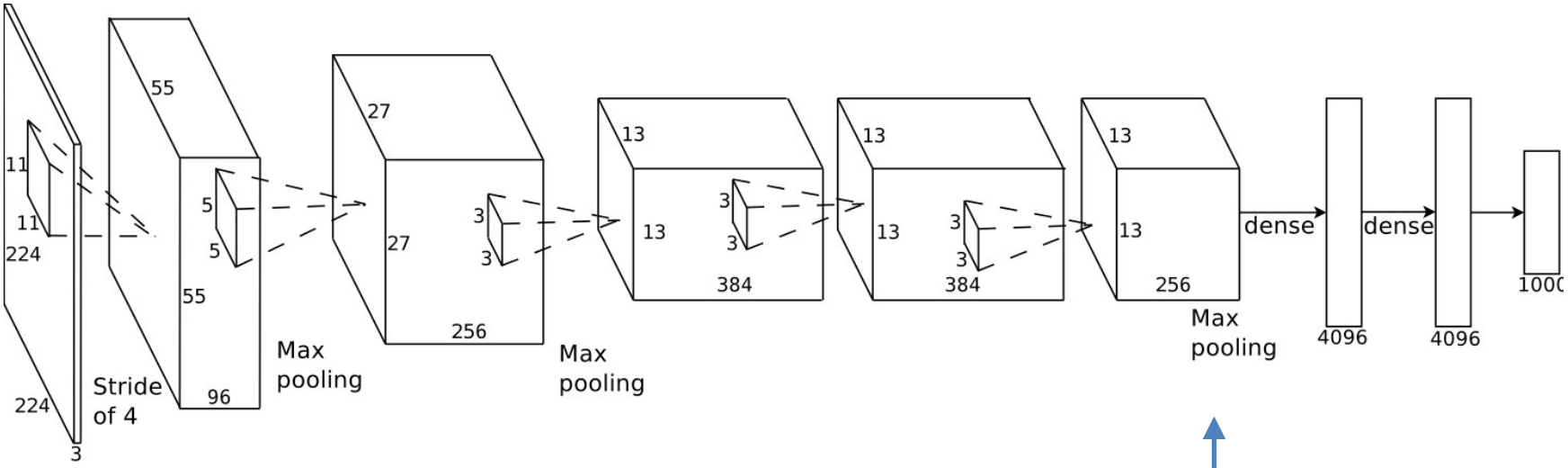
エッジ（細）が強調された画像

エッジ（太）が強調された画像

Rチャネルのエッジが強調された画像

～みたいな曲線が強調された画像

より高次の特徴へ



低次の特徴

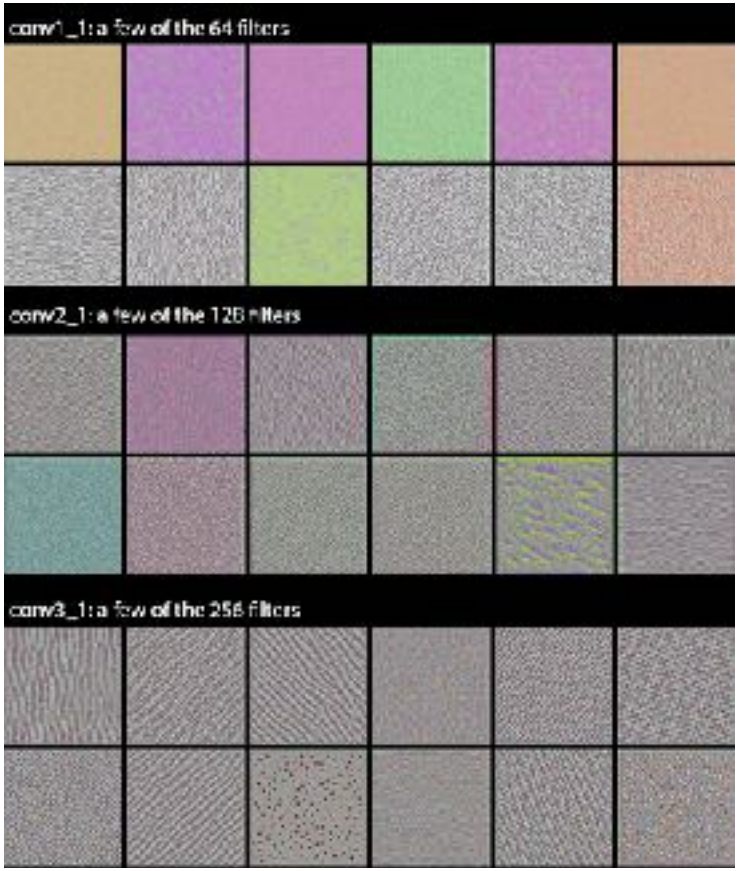
エッジ・形状・色など

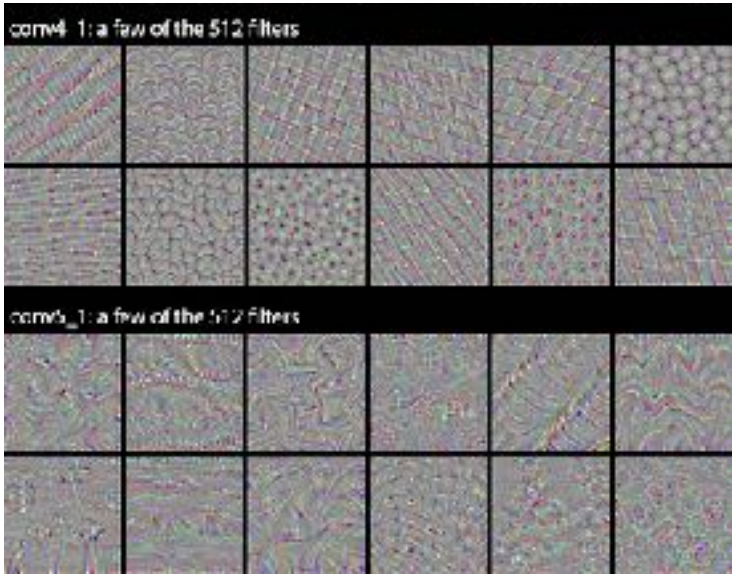
高次の特徴

「顔のパーツっぽさ」など

参考) VGG16の可視化

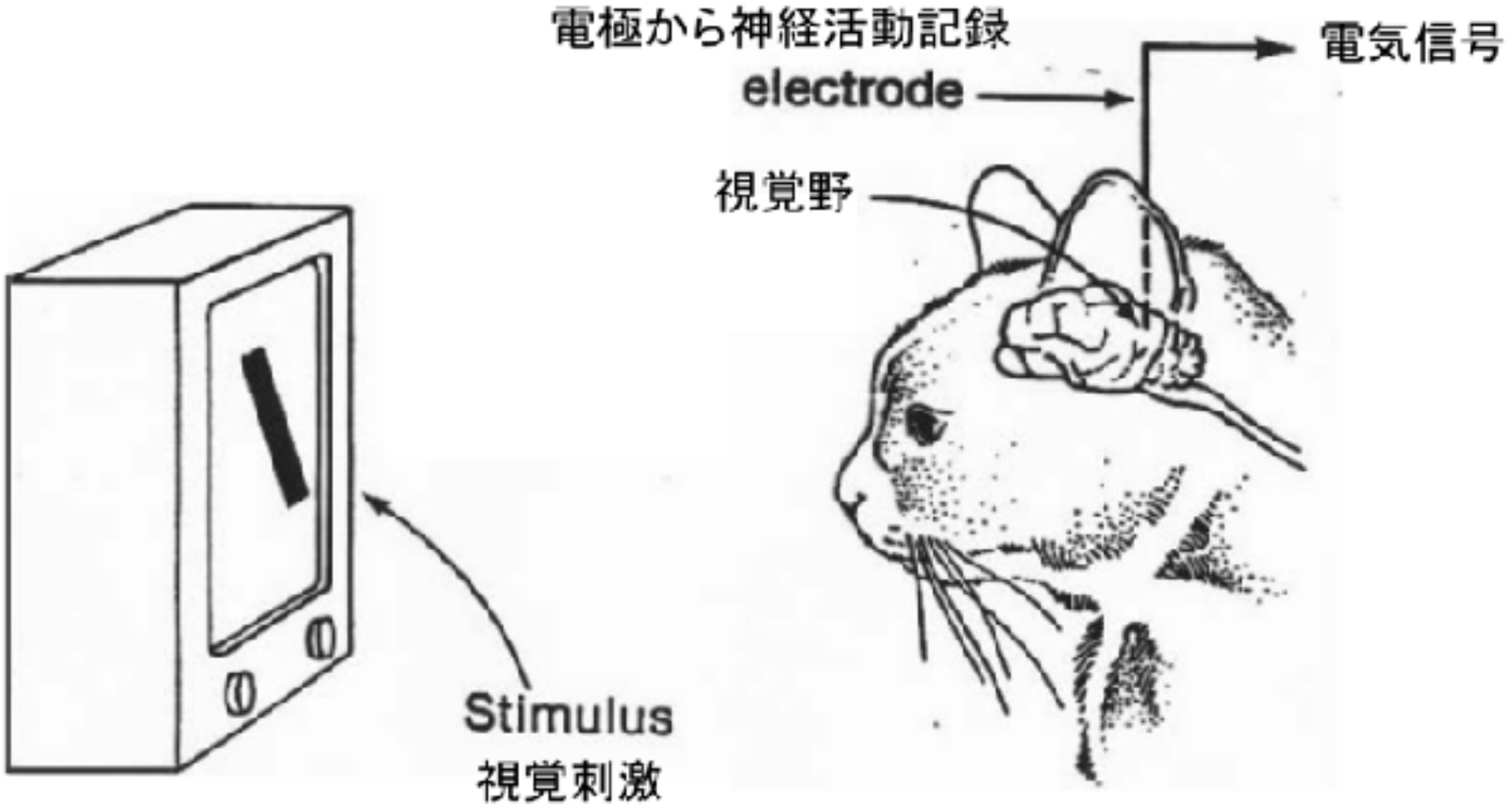
- https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html



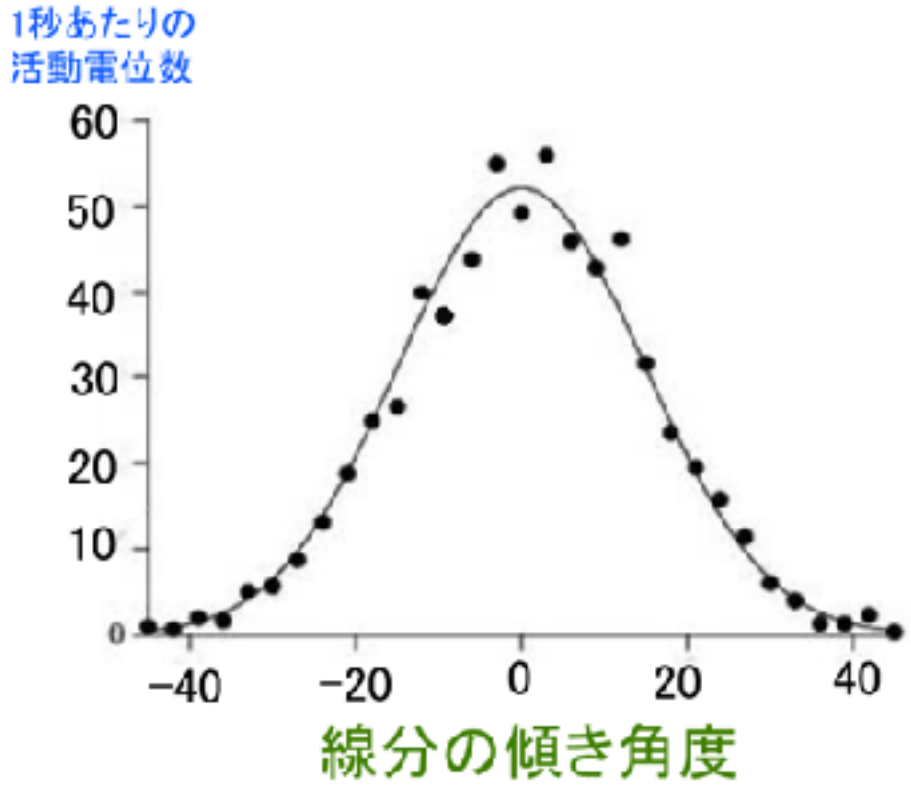


CNNと生物の視覚野の類似性

Hubel and Wieselの実験



傾き選択性のある細胞



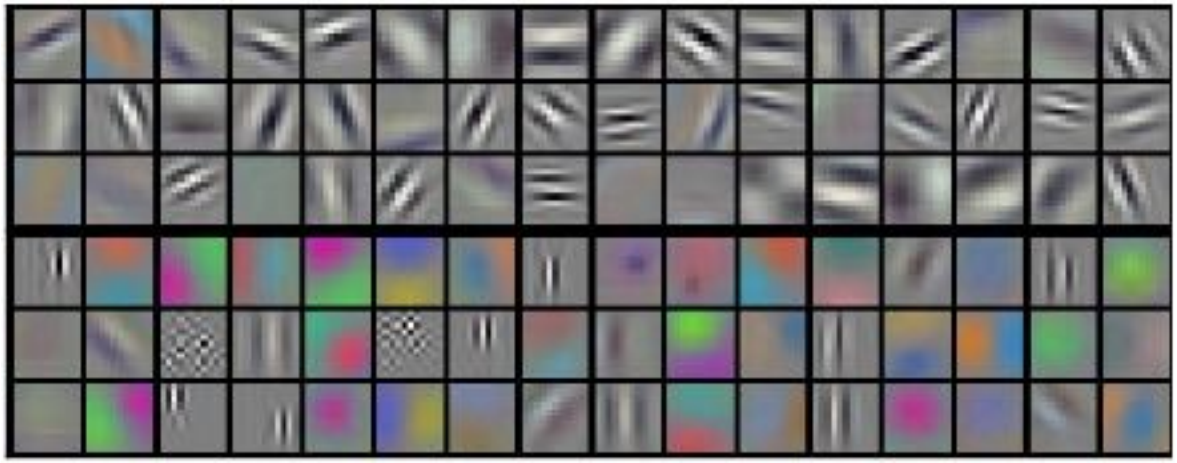
サルの第一次視覚野から記録した傾きに選択性を持つ細胞

Hubel and Wiesel, 1968

CNNとの類似性

AlexNetの1層目の可視化

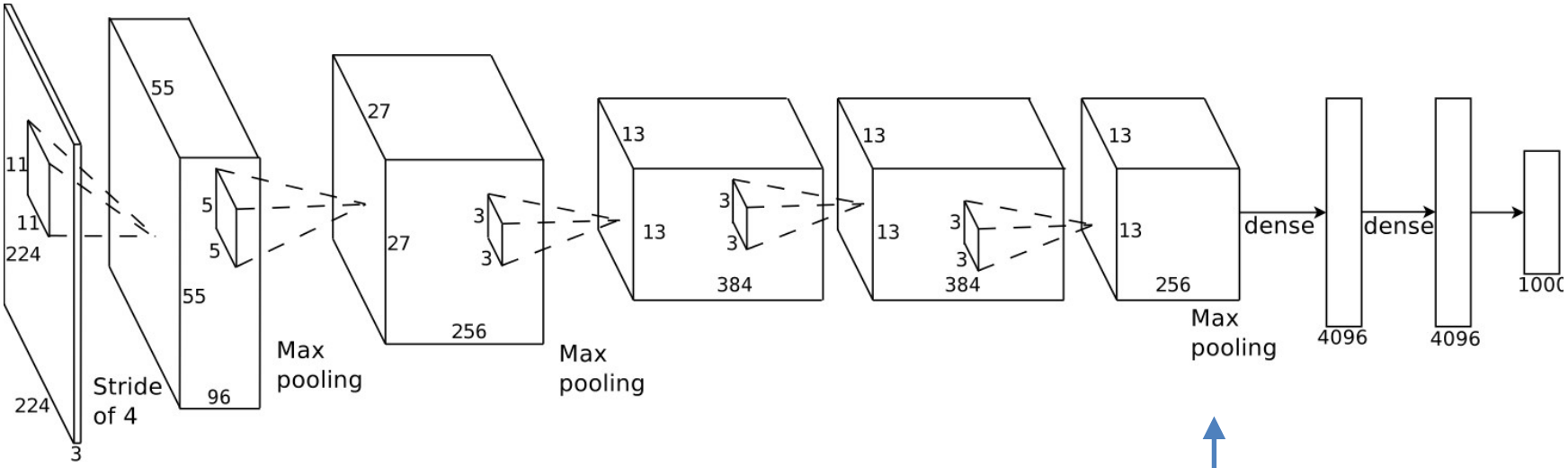
→単純なパターンに反応するフィルタが獲得されている



Krizhevsky et.al., 2012

→1次視覚野と類似！（？）

（再掲）より高次の特徴へ

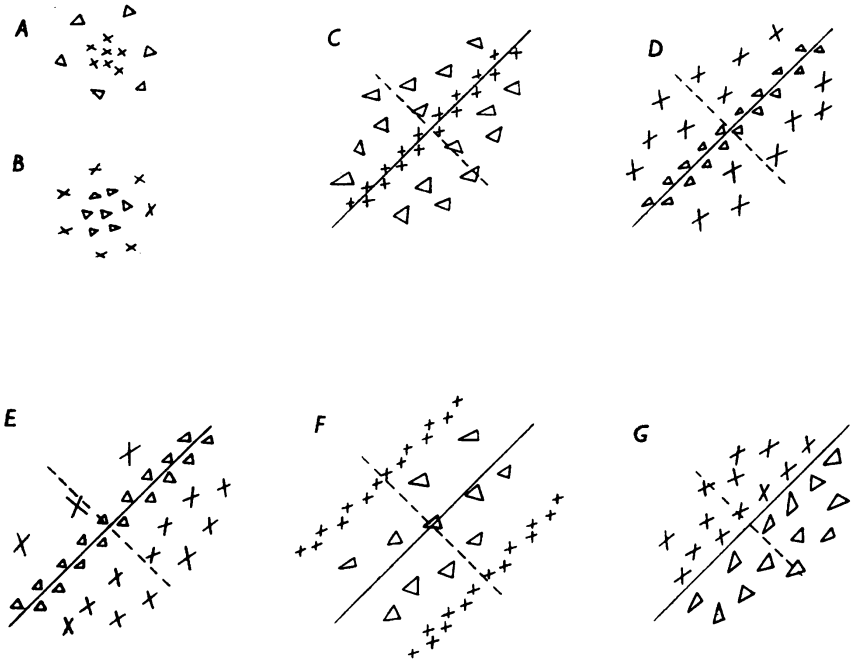


低次の特徴
エッジ・形状・色など

高次の特徴
「顔のパーツっぽさ」など

簡単なパターン抽出の組み合わせで高次の特徴を表現できる

単純型細胞・複雑型細胞

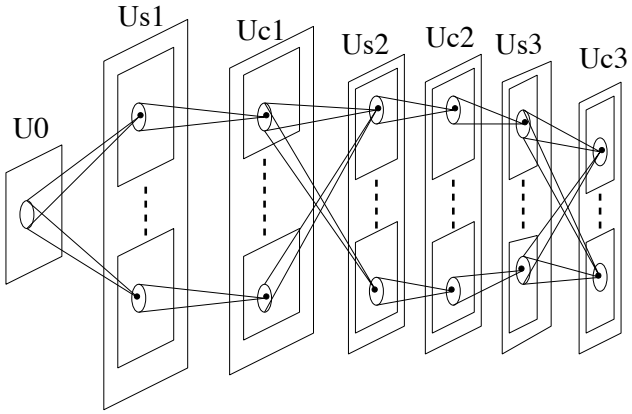


Text-fig. 2. Common arrangements of lateral geniculate and cortical receptive fields. *A*. 'On'-centre geniculate receptive field. *B*. 'Off'-centre geniculate receptive field. *C–G*. Various arrangements of simple cortical receptive fields. ×, areas giving excitatory responses ('on' responses); △, areas giving inhibitory responses ('off' responses). Receptive-field axes are shown by continuous lines through field centres; in the figure these are all oblique, but each arrangement occurs in all orientations.

単純細胞の組み合わせで複雑細胞が構成される

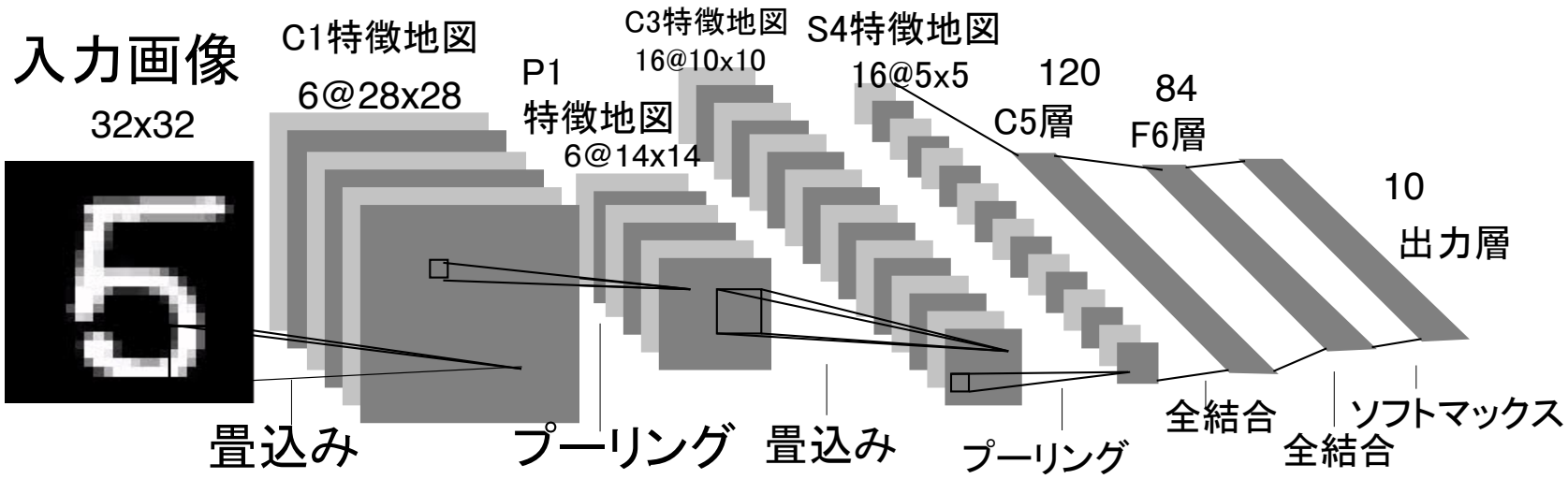
視覚神経モデル化の歴史

- ネオコグニトロン [Fukushima, 1980]
  - S 細胞と C 細胞との繰り返し。
    最初の多層（深層）化された物体認識モデルととらえられる
  - S 細胞：生理学の単純細胞 simple cells に対応。
    受容野 receptive fileds の概念を実現。
    特徴抽出、特徴検出を行う。
  - C 細胞：複雑細胞 complex cells に対応。
    広い受容野。位置、回転、拡大縮小の差異を吸収

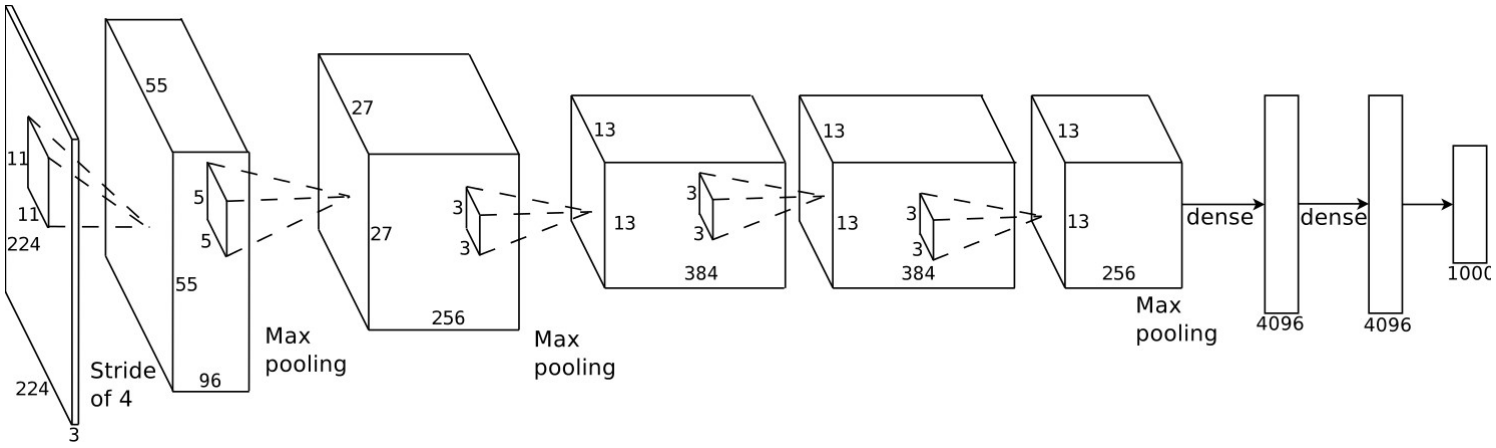


CNNの進化

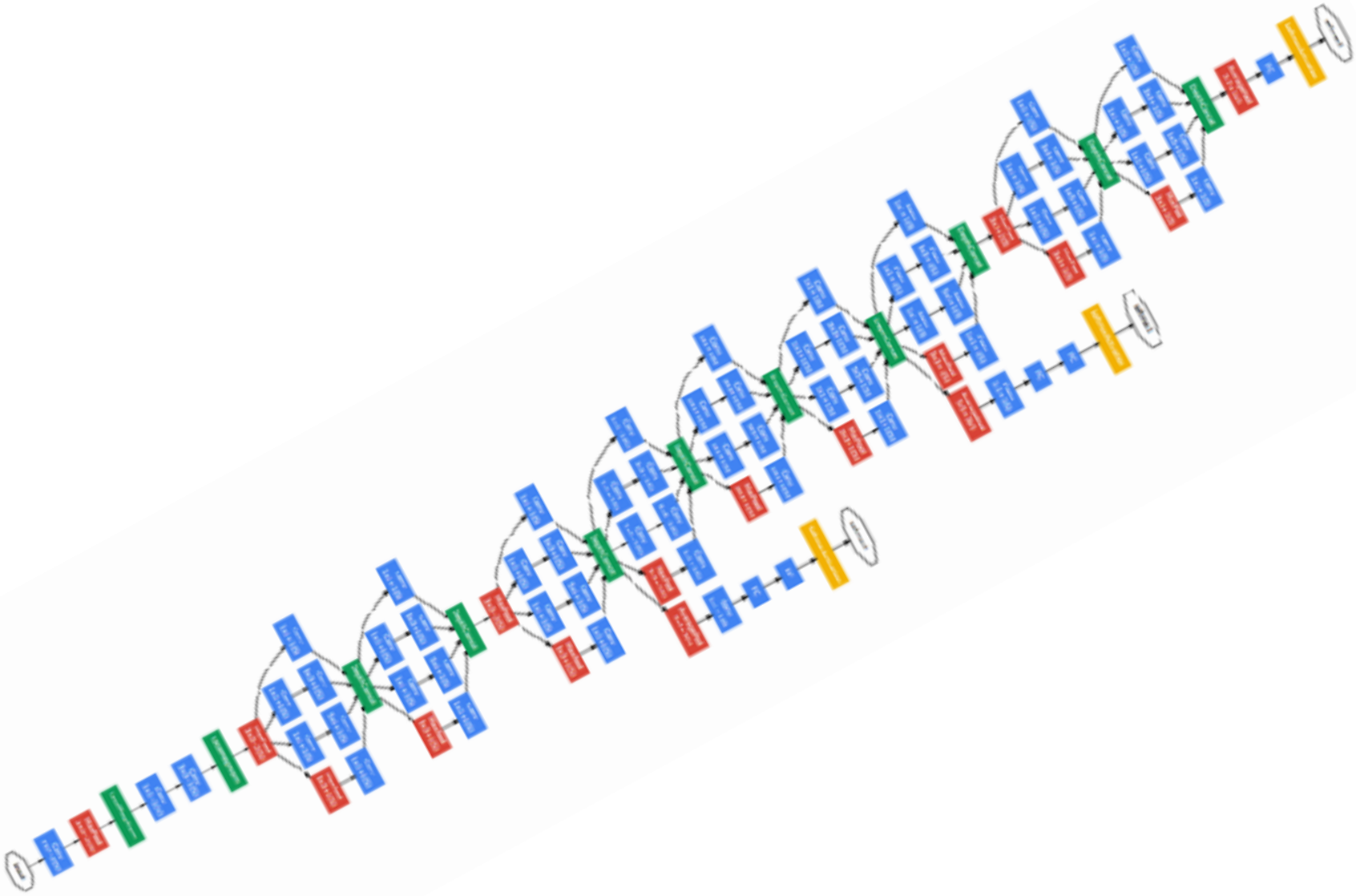
LeNet [LeCun et al., 1998]



AlexNet [Krizhevsky et.al., 2012]



GoogLeNet [Szegedy et al., 2014]



ResNet [He et al., 2015]

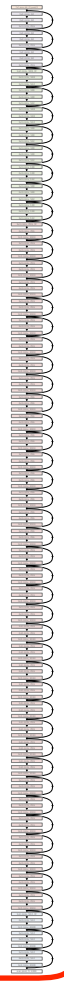
Microsoft Research

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)



ICCV15 International Conference on Computer Vision

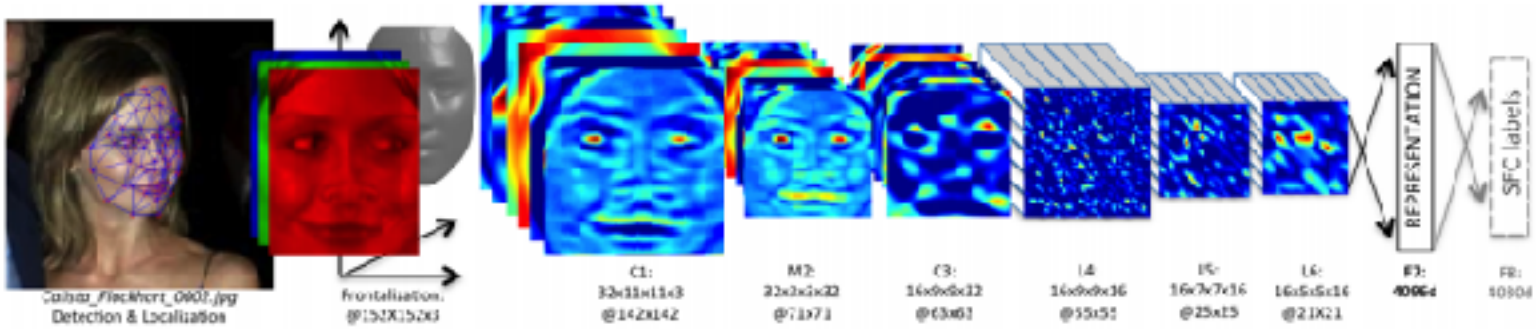
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

CNNの進化やばい

- 2012年のAlexNet以降、
  数え切れない数のモデルが提案されている
  （紹介しきれません）
  - VGG
  - GoogLeNet
  - ResNet
  - …
- 基本的に、よりDeepに、Deepに…
- 生物の視覚神経との類似性はどこへやら
- CNNの独自進化の結果、逆に生物を理解する成果が出るかも…？

局所結合ネットワーク Locally Connected Layers

- 結合の重みが位置によって違う
  ＝Weight sharing**しない**
- 画像のアラインメントが仮定されるタスクではCNNより高精度なものも
  - Deep Face [Taigman et al., 2014]



Chainerでやってみる

紹介した道具は全てchainerに用意されています

- Convolutional layer
  - Hyper parameters: k, s, p, (w, h)
- Pooling layer
  - Type: max, average, …
  - Hyper parameters: k, s
- Dropout
  - Hyper parameters: p
- Data augmentation
- Normalizations
- etc…

注意

- 先にMNISTのダウンロードを実行しておく