

現場で使えるディープラーニング基礎講座

DAY5

SkillUP AI

## 前回の復習

---

- 前回は何を学びましたか？

# 中間発表

# 中間発表

---

- 課題の成果を発表して頂きます。
- 1人当たりの時間は、**発表3分+質疑応答2分=5分**とします。
- 発表内容に入るものの
  - 1. 実装の進捗状況
  - 2. モデルの改良点と識別精度の変遷
  - 3. 学習用データでの識別精度とテスト用データでの識別精度の比較
  - 4. Arxiv.orgなどで見つけた論文とそこから得られた知見
  - 5. 今後の取り組み予定
- 事前に、課題をまとめたNotebookを講師にDMで渡してください。
- 発表時は、そのNotebookをプロジェクターに投影するようにします。時間が限られていますので、原則、個人PCでの発表は受け付けないようにしたいと思います。

# CNNの様々なモデル

# 目次

---

1. 著名なCNNモデル
2. 物体検出タスクとCNN
3. セマンティックセグメンテーションタスクとCNN
4. 様々な畳み込み
5. 様々なプーリング

## 著名なCNNモデル

---

# LeNet-5

- LeNet-5は、6層構造(CNN層が5つ)のCNN。
- 32×32ピクセルのパッチ画像を入力とする。
- 1998年ごろYann LeCunらによって提案された。
  - <http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf>
  - <http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>
- 置み込み層の考え方、1989年ごろから提案されていた。
  - <http://yann.lecun.com/exdb/publis/pdf/lecun-89e.pdf>

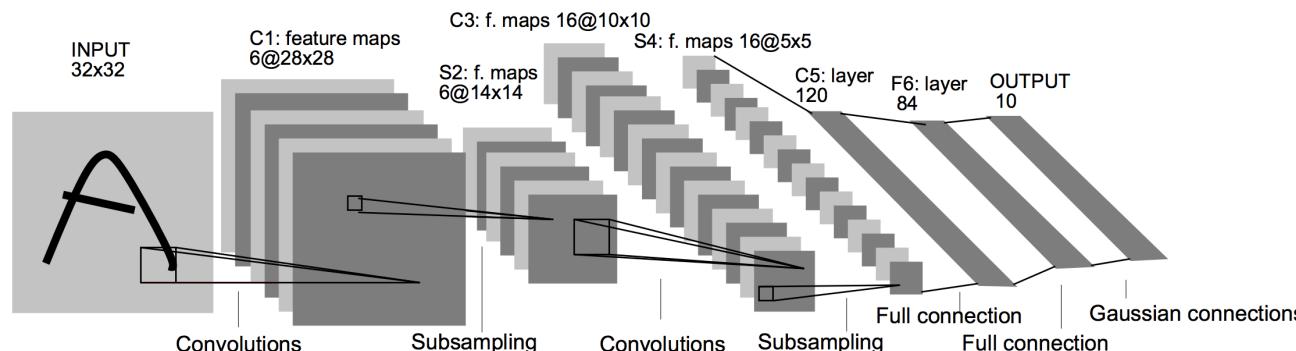
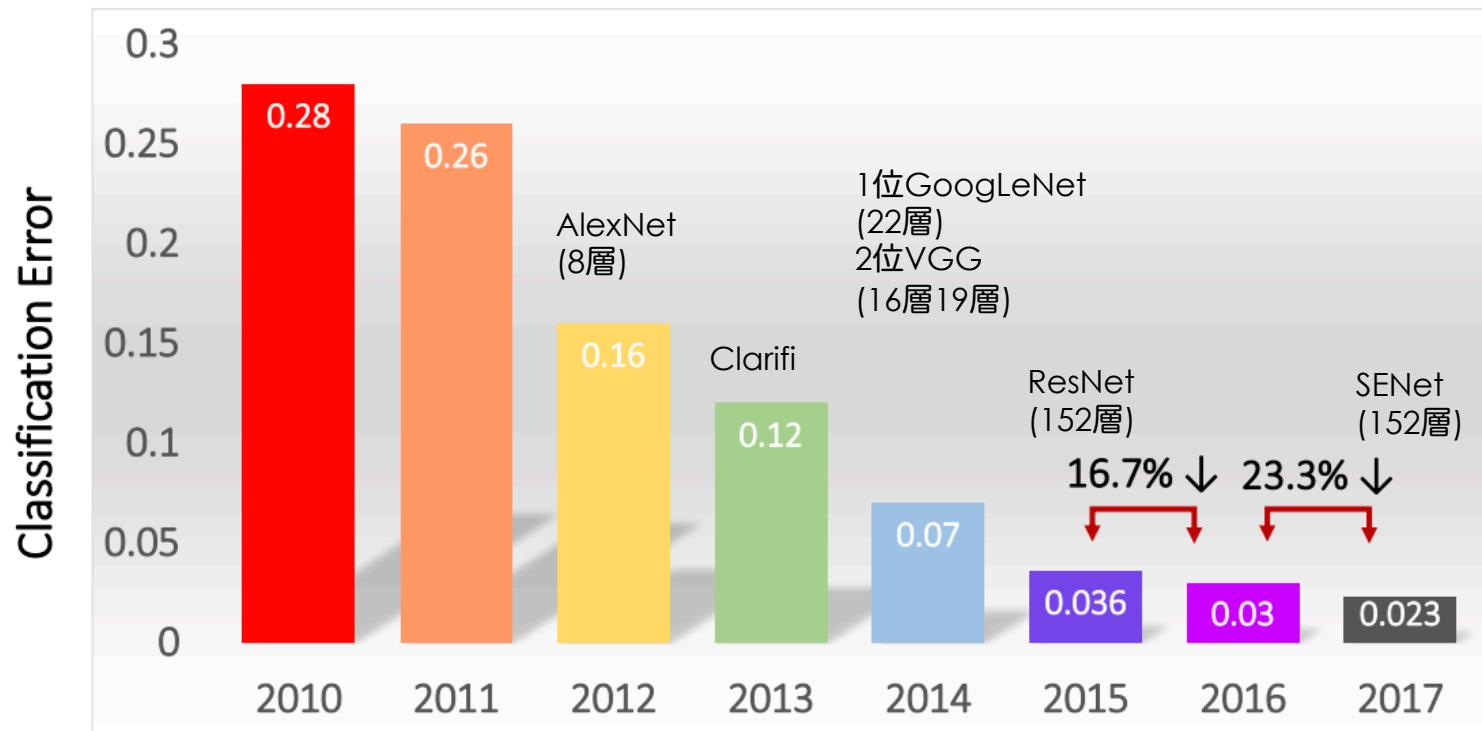


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

LeNetのネットワーク

# ILSVRC

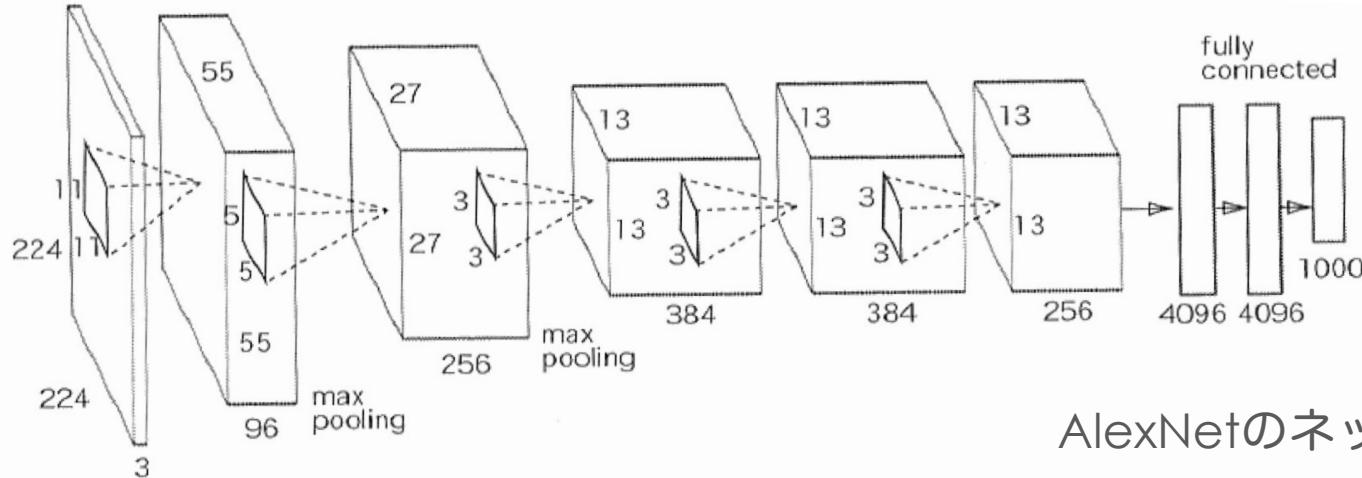
- ILSVRC (Image net Large Scale Visual Recognition Challenge)は、2010年から始まった大規模画像認識のコンテスト。



[http://image-net.org/challenges/talks\\_2017/ILSVRC2017\\_overview.pdf](http://image-net.org/challenges/talks_2017/ILSVRC2017_overview.pdf)

# AlexNet

- 2012年のILSVRCで1位になったネットワーク。
- 5層の畳み込み層と3層の全結合層で構成される。
- パラメータ数は、約6000万個。
- <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- [http://vision.stanford.edu/teaching/cs231b\\_spring1415/slides/alexnet\\_tugce\\_kyunghee.pdf](http://vision.stanford.edu/teaching/cs231b_spring1415/slides/alexnet_tugce_kyunghee.pdf)



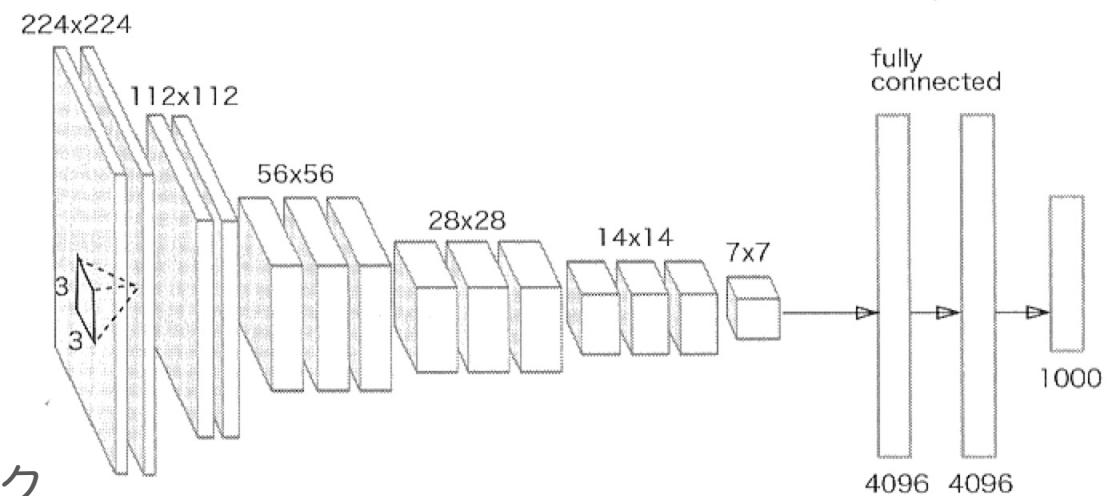
AlexNetのネットワーク

# VGG

---

- 2014年のILSVRCで2位になったネットワーク。
- 19層のネットワークはVGG-19、16層のネットワークはVGG-16と呼ばれる。
- AlexNetよりも深いネットワーク構造。
- VGGとは、Visual Geometry Group(Oxford大学の研究室)の略。
- パラメータ数は、約1億4000万個。
- 構造がシンプルなため、実務でよく使われる。
- <https://arxiv.org/pdf/1409.1556.pdf>

VGG-16のネットワーク



# GoogLeNet

- 2014年のILSVRCで1位になったネットワーク。
- <https://arxiv.org/pdf/1409.4842.pdf>

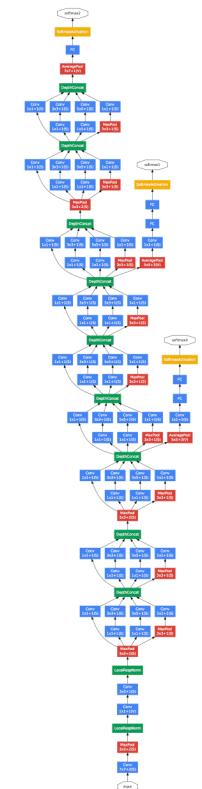
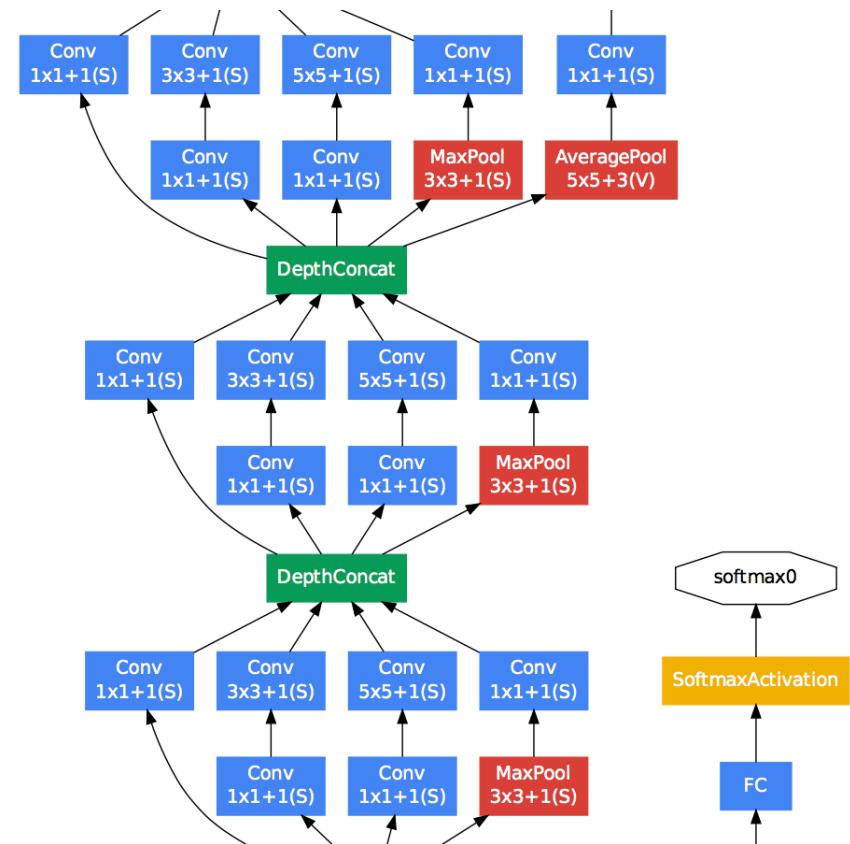
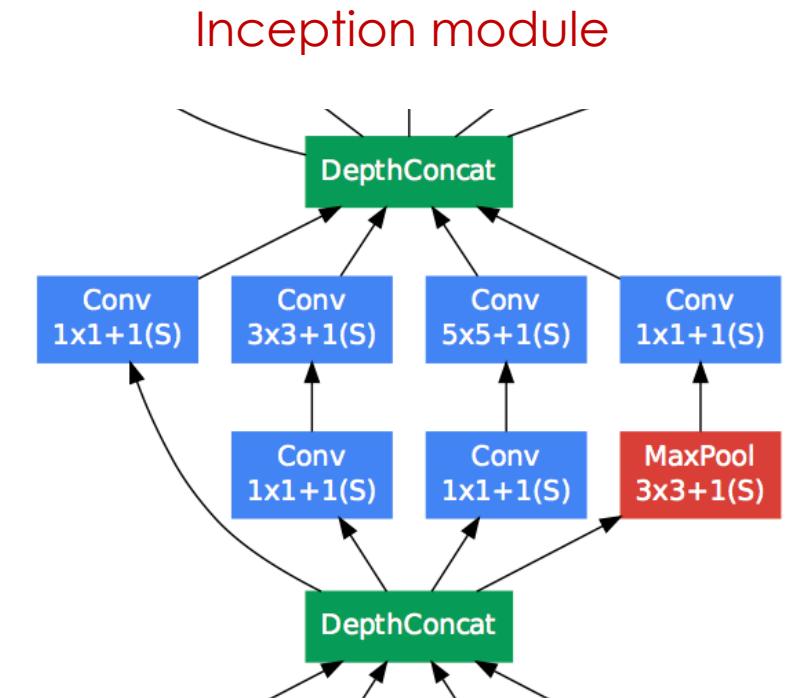


Figure 3: GoogLeNet network with all the bells and whistles



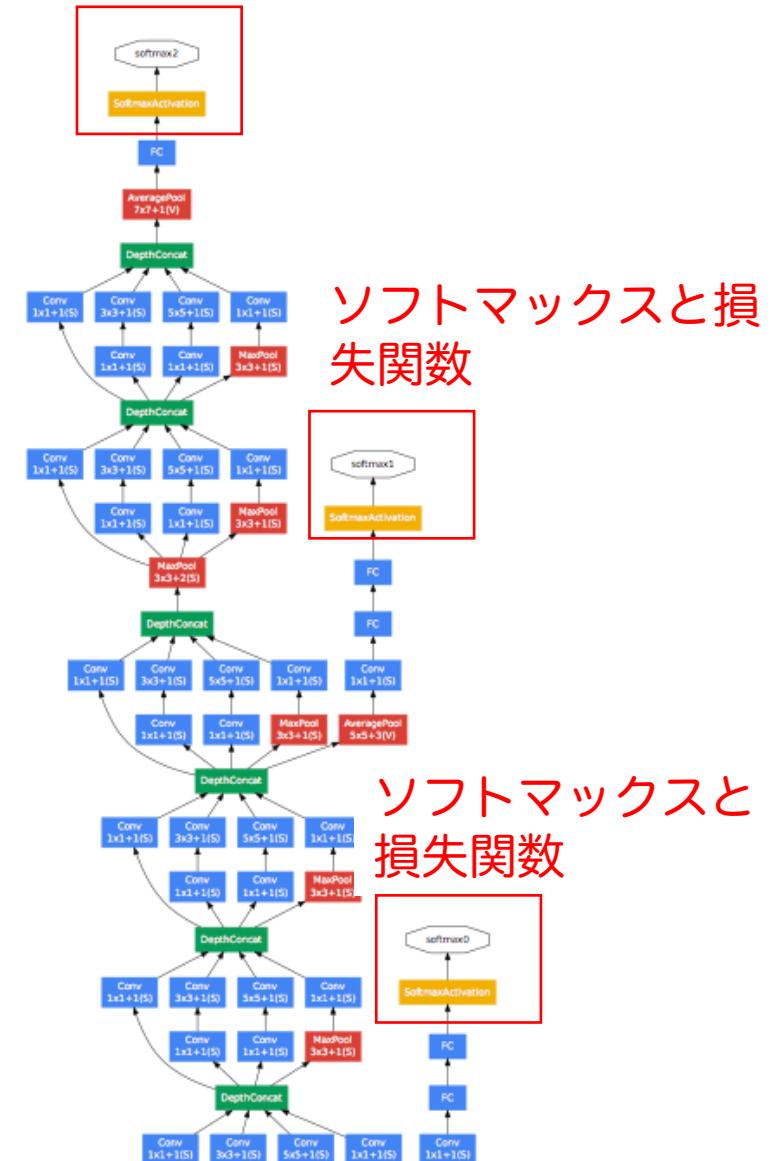
# GoogLeNetの仕組み

- Inception moduleと呼ばれる複数のフィルタ群により構成されたブロックが特徴。
- Inception moduleは、小さなネットワークから構成される。GoogleNetは、このモジュールを積み重ねた構成になっている。
- Inception moduleは、小さな畳み込みフィルタを並列に並べることで、より少ないパラメータで同等の表現力を実現することができている。
- Inception moduleには $1 \times 1$ の畳み込みフィルタが使われているが、このフィルターは次元削減と等価な効果がある。



# GoogLeNetの仕組み

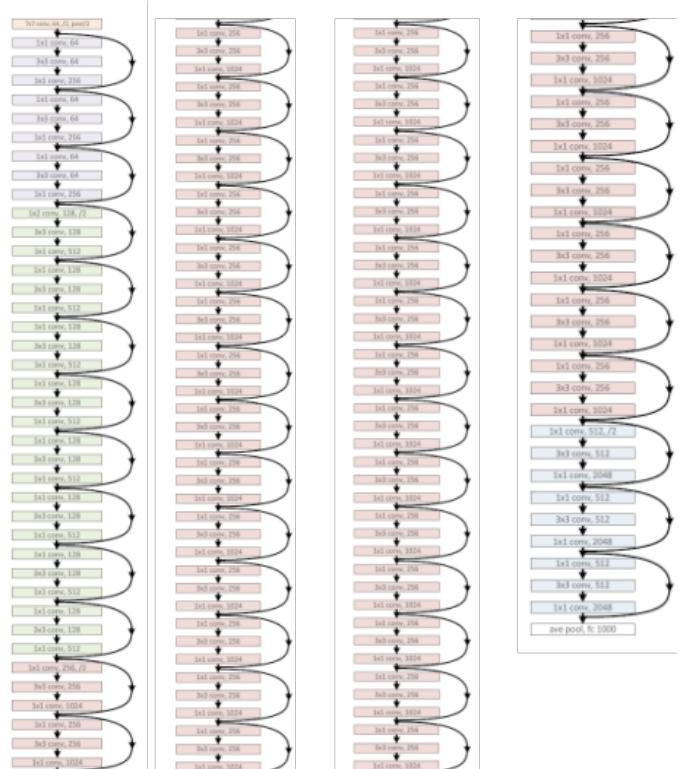
- GoogLeNetのもう一つの特徴として、補助的損失(auxiliary loss)がある。
- GoogLeNetの学習ではネットワークの途中から分岐させたサブネットワークにおいてもクラス分類を行っている。
- ネットワークの中間層に直接誤差を伝播させることで、勾配消失を抑えることができる。
- 複数の損失関数があることにより、アンサンブル学習と同様の効果が得られるため、汎化性能の向上が期待できる。
  - ただし、予測時は最終層の出力結果だけを用いる。



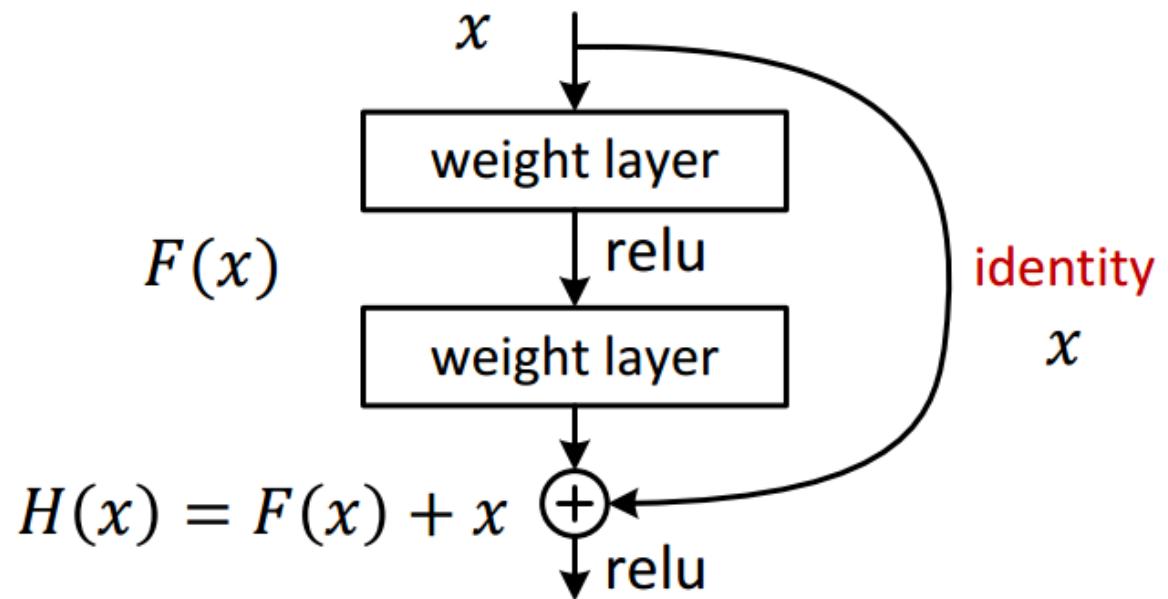
softmaxと損失関数

softmaxと損失関数

- 2015年のILSVRCで1位になったネットワーク。
- Microsoftのチームのよって開発された。
- <https://arxiv.org/pdf/1512.03385.pdf>



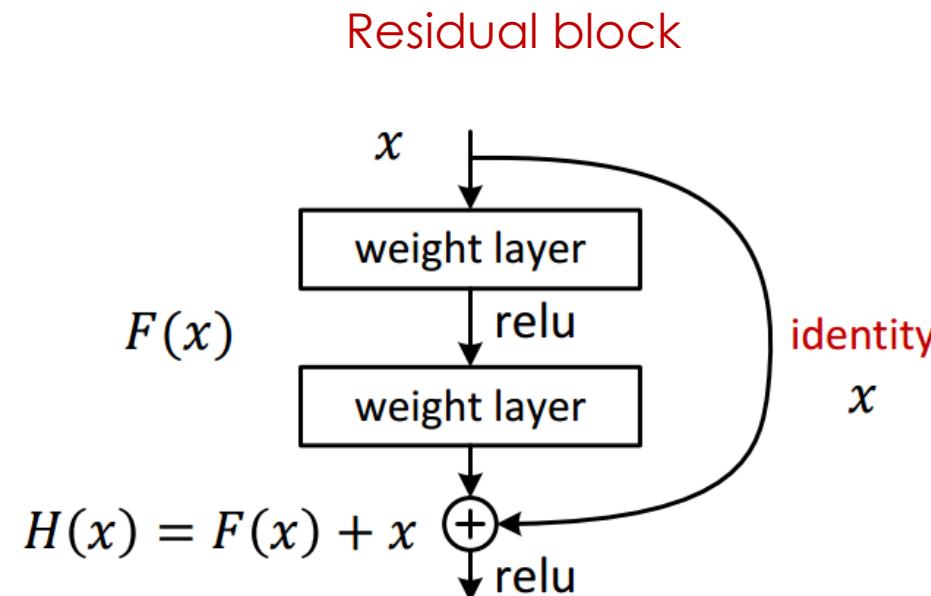
### • Residual net



# ResNetの仕組み

- ResNetは、Residual blockというユニットが連結された形になっている。
- Residual blockには、層をまたがる結合(Identity mapping)がある。このショートカット結合により、逆伝播計算時に勾配が減衰していくことを抑えることができる。
- この手法により、従来の「層を深くすると学習できない」という問題が解決された。

$F(x)$  では、入力  $x$  と出力  $H(x)$  の差(残差)である  $H(x) - x$  を学習しているため、Residual block(残差ブロック)と呼ばれる

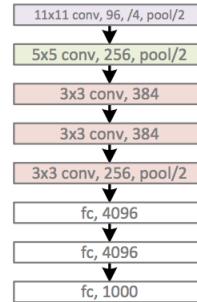


# 著名CNNの比較

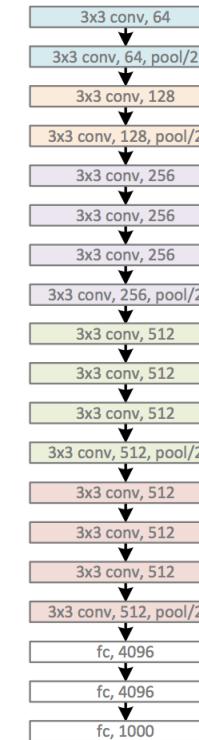
Microsoft  
Research

## Revolution of Depth

AlexNet, 8 layers  
(ILSVRC 2012)



VGG, 19 layers  
(ILSVRC 2014)



GoogleNet, 22 layers  
(ILSVRC 2014)



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

## Revolution of Depth

AlexNet, 8 layers  
(ILSVRC 2012)



VGG, 19 layers  
(ILSVRC 2014)



ResNet, 152 layers  
(ILSVRC 2015)



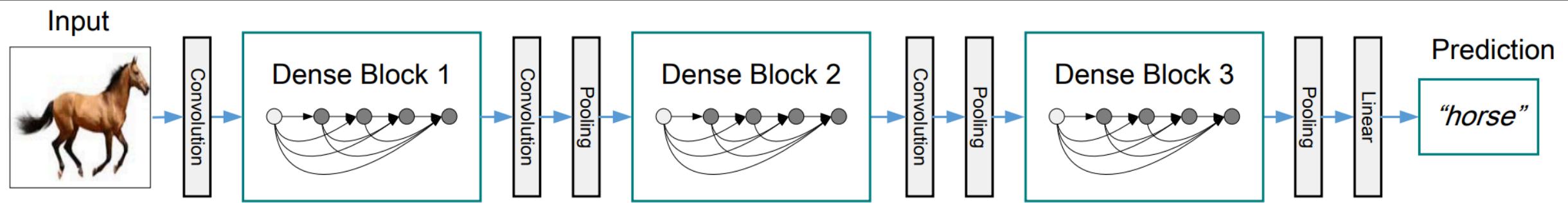
# DenseNet

---

- ResNetに似たショートカット結合をもつネットワーク。
- 特徴量を再利用する結合になっている。
- DenseNetの特徴
  - ショートカット結合を用いているため、**勾配消失が起きにくい**。
  - DenseNet以前のネットワークよりも、**パラメータ数を少なく**することができる。
- 原著論文
  - G.Huang, Z.Liu, L.van der Maaten, K.Q.Weinberger. Densely Connected Convolutional Networks. IEEE Conference on Pattern Recognition and Computer Vision (CVPR), 2016., <https://arxiv.org/pdf/1608.06993.pdf>

# DenseNet

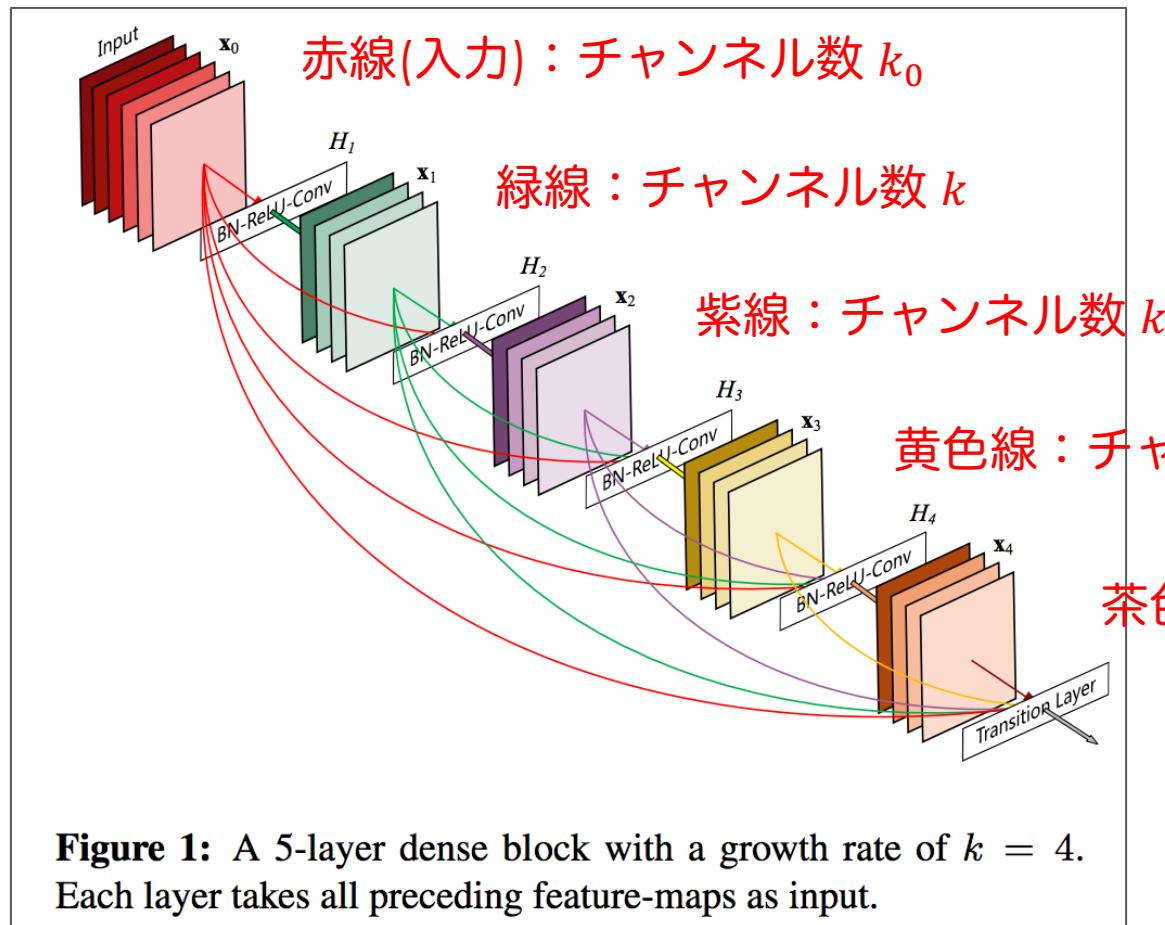
- DenseNetは、DenseBlockによって構成される。



**Figure 2:** A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

引用元: <https://arxiv.org/pdf/1608.06993.pdf>

- DenseBlockの結合方法を5層の場合で示す。



チャンネル数 $k$ は成長率(Growth rate)と呼ばれ、ハイパーパラメータになる。  
BN-ReLU-Convにおいて、異なる色の線は、チャンネル方向に結合される

黄色線：チャンネル数  $k$

茶色線：チャンネル数  $k$

白線(出力)：チャンネル数  $k_0+4k$

※モデルを小さくするために、transition layerにて、出力チャンネル数を  $k_0+4k$ よりも小さくさせる場合がある

引用元: <https://arxiv.org/pdf/1608.06993.pdf>

Any Questions?

## 物体検出タスクとCNN

---

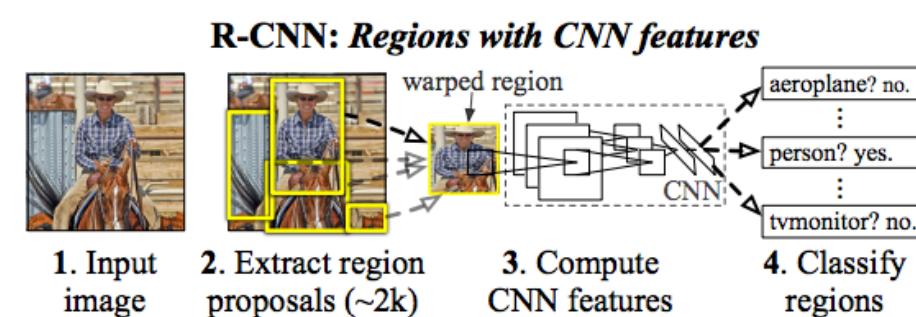
# 物体検出タスク

---

- ・ 物体検出タスクとは、ある画像中における物体の位置を検出し、それにラベルを付与するタスクのこと。
- ・ 物体検出で使われる主なアルゴリズムを以下に示す。
  - ・ R-CNN
  - ・ Fast R-CNN
  - ・ Faster R-CNN
  - ・ YOLO(you only look once)
    - ・ 2019年5月時点でv1~v3が発表されているが、ここではv1について紹介する。
  - ・ SSD(single shot multibox detector)
- ・ 物体検出タスクでは、物体を囲う矩形のことをバウンディングボックスと呼ぶ。

# R-CNN

- R-CNN(regions with CNN features)は、物体検出方法の1つ。
- 計算手順
  1. 入力画像に対して、物体が写っている領域の候補を選択的検索法(selective search method)で約2,000個抽出する。
  2. CNNのインプットの大きさに合うようにそれぞれの領域中の画像をリサイズする。
  3. それぞれの物体領域候補に対して、CNN (原著論文では AlexNet) で特徴マップを計算する。
  4. それぞれの領域に何が写っているかSVMで分類する。
  5. 物体の詳細位置を決めるために、特徴マップとバウンディングボックス座標を回帰させる問題を解く。
- 約2000画像の特徴を抽出するため、計算に時間がかかる。



引用元：Rich feature hierarchies for accurate object detection and semantic segmentation, Ross Girshick, et al.

<https://arxiv.org/pdf/1311.2524.pdf>

# 選択的検索法

- 選択的検索法(selective search method)とは、ピクセルレベルで類似する領域をグルーピングしていくことで候補領域を選出するアルゴリズム。
- 似たような特徴を持つ領域を結合していく、1つのオブジェクトとして抽出する。
- 計算手順
  - 画像を小さい領域群に分割する。
  - 隣り合う領域どうしの類似度を計算する。
  - 最も類似度の高い領域を統合する。
  - 領域が1つになるまで、2と3を繰り返す。

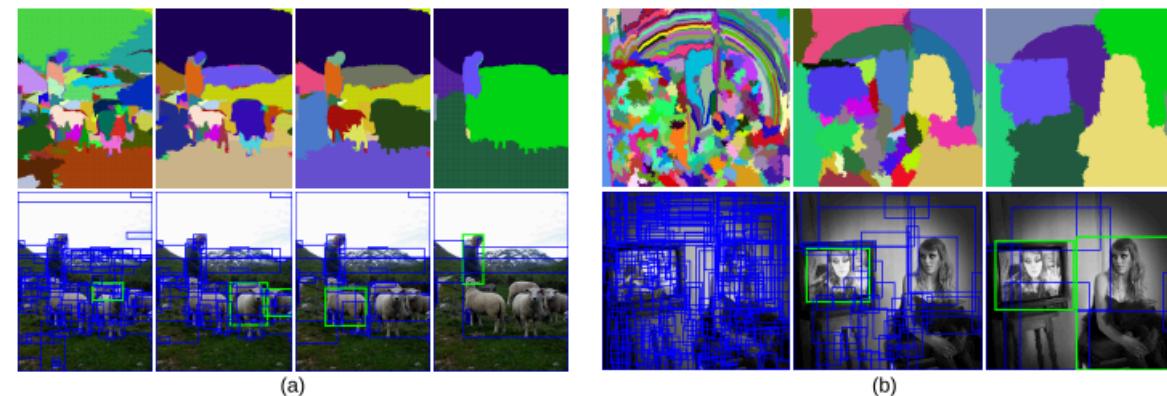


Figure 2: Two examples of our selective search showing the necessity of different scales. On the left we find many objects at different scales. On the right we necessarily find the objects at different scales as the girl is contained by the tv.

引用元： Selective Search for Object Recognition, J.R.R. Uijlings et al  
<https://koen.me/research/pub/uijlings-ijcv2013-draft.pdf>

# Fast R-CNN

- R-CNNは、提案された物体領域候補ごとに順伝播計算を行う必要があり、検出速度の遅いことが課題であった。
- 画像全体を入力して計算した特徴マップをすべての物体領域候補で再利用することで、R-CNNを高速化させたものがFast R-CNN。
- 計算手順
  1. 選択的検索法を利用して物体領域候補を算出する。
  2. 画像全体をCNNに入力して特徴マップを計算する。
  3. 物体領域候補ごとに以下の計算を行う。
    1. 物体領域候補に対応する特徴マップを抽出する。
    2. 特徴マップをRoI(Region of Interest)プーリング層に渡し、次元数の異なる特徴マップを一定の大きさのベクトルに変換する。
    3. 物体の詳細位置を決めるために、特徴マップとバウンディングボックス座標を回帰させる問題を解く。

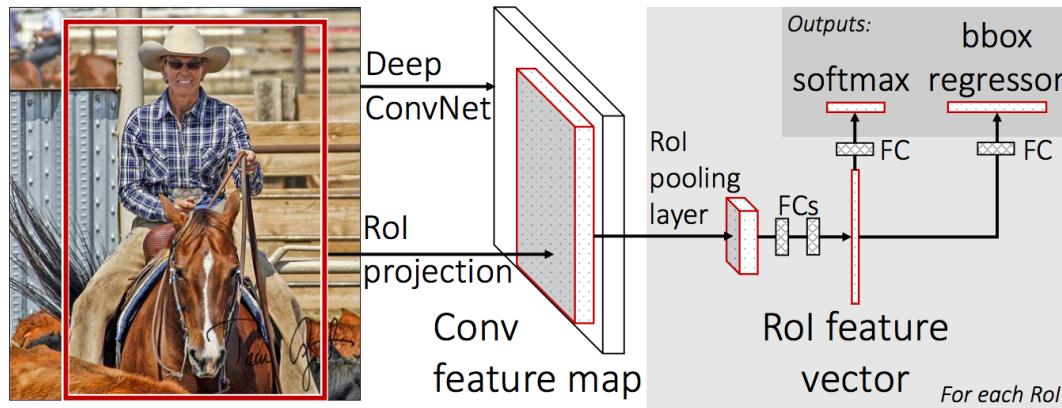


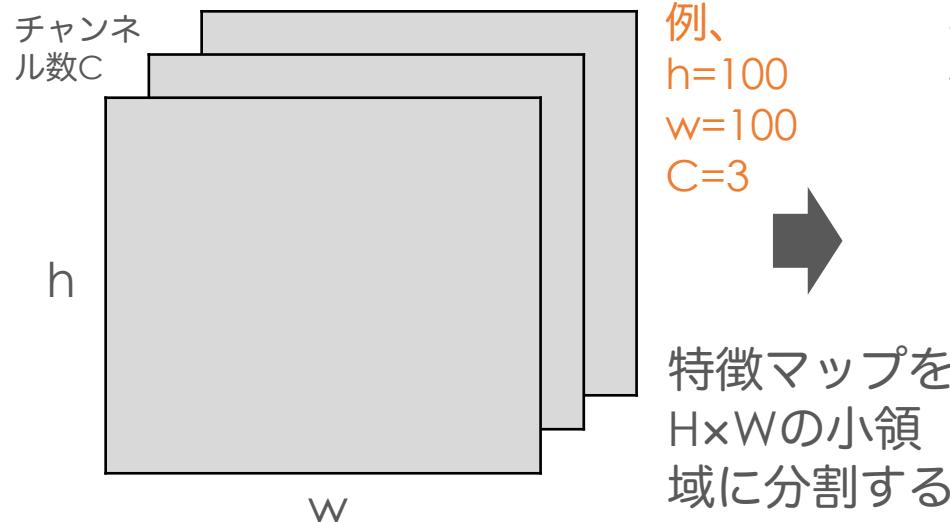
Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each ROI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per ROI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

引用元：Fast R-CNN, Ross Girshick,  
<https://arxiv.org/pdf/1504.08083.pdf>

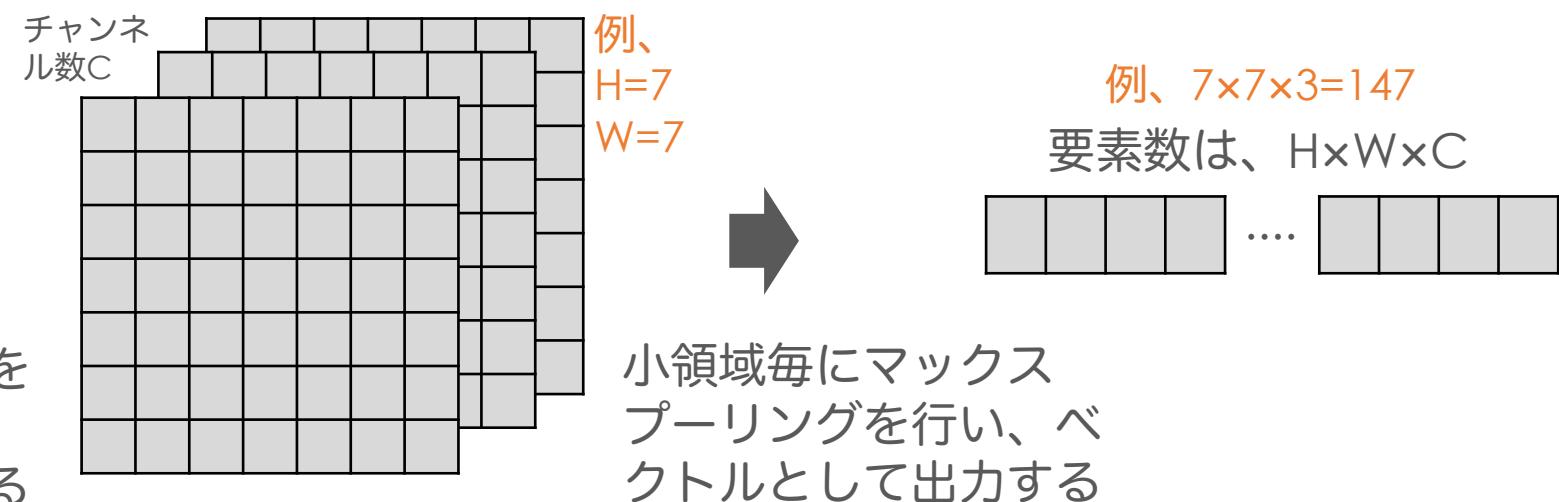
# Roiプーリング層

- Roi(Region of Interest)プーリング層では、物体領域候補の特徴マップが入力され、マックスプーリング演算を行い、要素数が $H \times W \times C$ のベクトルが出力される。HとWはあらかじめ決めておく分割数(例、 $H=7$   $W=7$ )であり、Cは特徴マップのチャンネル数である。
- $H \times W$ のそれぞれの領域について、 $1 \times 1$ の空間ピラミッドプーリングを行なっていることに相当する。

特徴マップ( $h \times w \times C$ )



Roiプーリング層の出力



# Faster R-CNN

- Fast R-CNNでは、ニューラルネットワークとは別のモジュールで物体領域候補を計算する必要があった。
- Faster R-CNNは、特徴マップから物体領域候補を推定する領域提案ネットワーク(RPN, region proposal network)とFast R-CNNの2つのネットワークで構成される。
- 2つのネットワークで特徴マップを共有しているため、計算が効率的。
- Fast R-CNN側のネットワークで、特徴マップとバウンディングボックス座標を回帰させる問題を解く。

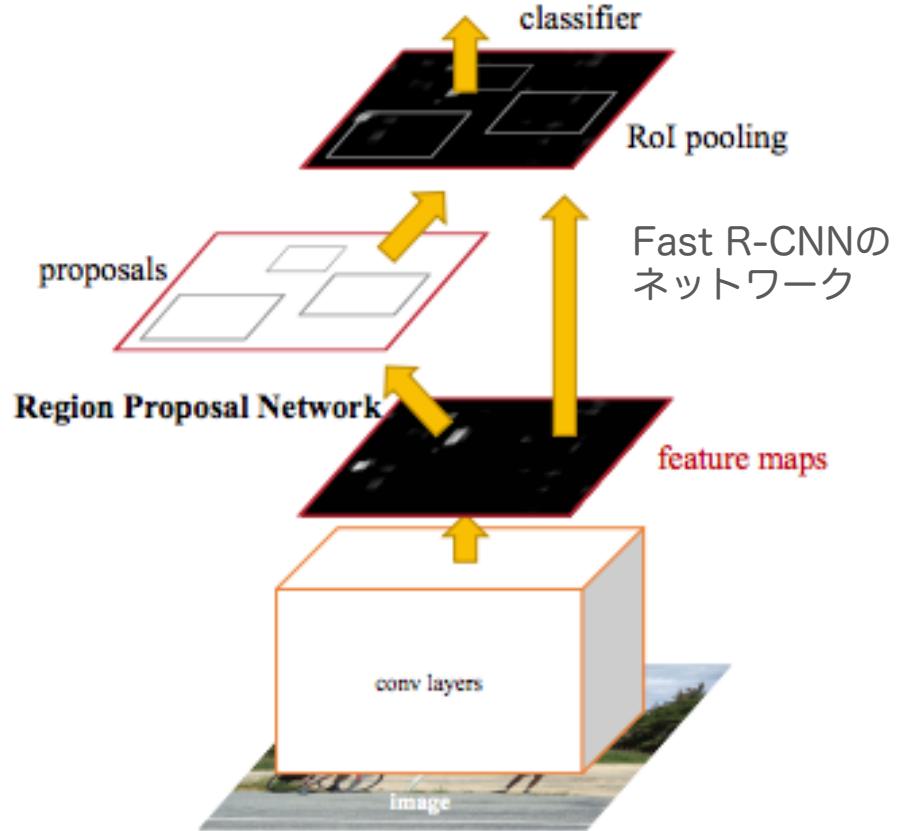


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

引用元：Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks ,  
Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun ,  
<https://arxiv.org/pdf/1506.01497.pdf>

# YOLO

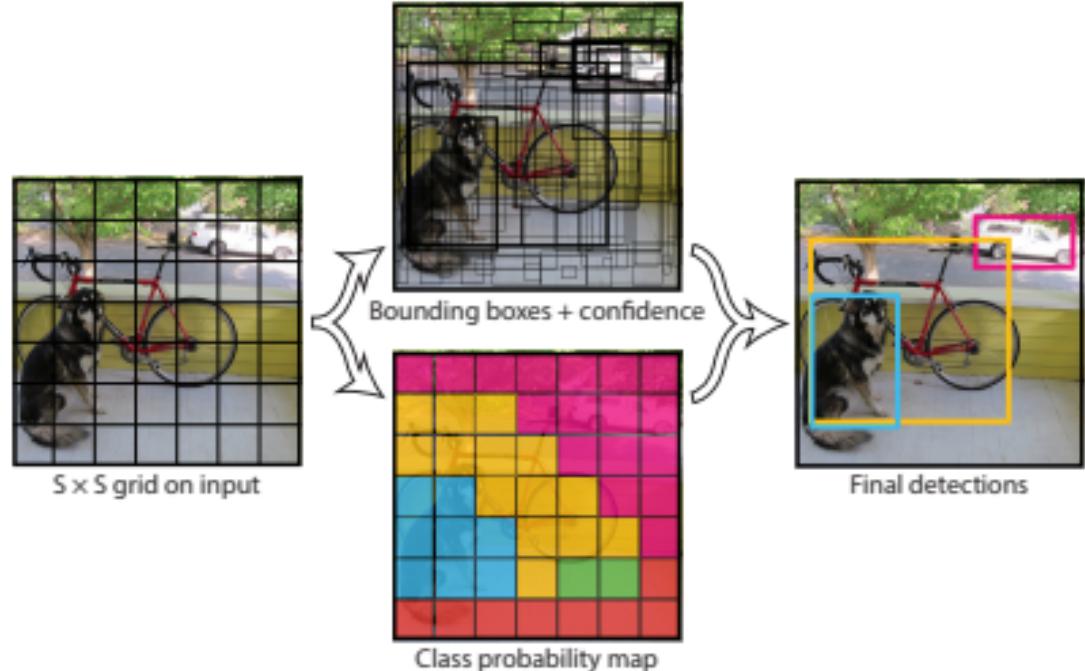
- YOLO (you only look once)は、**画像全体をグリッドに分割**(分割された領域をセルと呼ぶ)し、セル毎にクラスとバウンディングボックスを求める方法。
- 1つのニューラルネットワークで、**クラス分類問題とバウンディングボックス回帰問題**の両方を解く。
- Faster R-CNN に比べ、CNNのアーキテクチャがシンプルになり、検出速度が向上した。

S:分割数

B:各セルで検出できるバウンディングボックスの個数

5: バウンディングボックスのx座標、y座標、幅、高さ、信頼度の5つ

C:クラス数



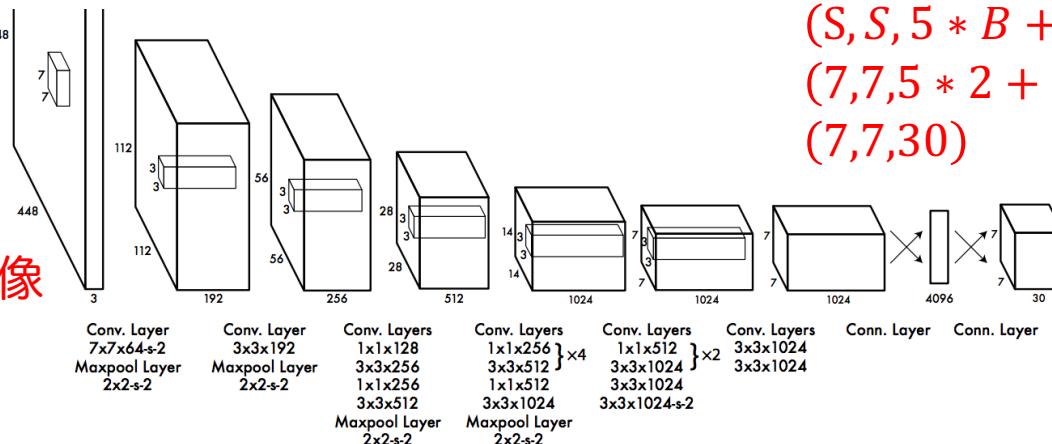
**Figure 2: The Model.** Our system models detection as a regression problem. It divides the image into an  $S \times S$  grid and for each grid cell predicts  $B$  bounding boxes, confidence for those boxes, and  $C$  class probabilities. These predictions are encoded as an  $S \times S \times (B * 5 + C)$  tensor.

引用元： You only look once: Unified, real-time object detection, edmon, Joseph, et al,

<https://arxiv.org/pdf/1506.02640.pdf>

- グリッドサイズ $S$ はユーザーが決める。(原論文では $7 \times 7$ で検証を実施)
- グリッド内で出力できるクラスは1つのみ。
- グリッド内で検出できる物体の数 $B$ はユーザーが決める。(原論文では2で検証を実施)
- 上記の理由により、グリッド内に大量のオブジェクトが写ってしまうような場合は精度が低くなる。
- YOLOのネットワーク構成例を以下に示す。

入力は $448 * 448$ 画像



引用元： You only look once: Unified, real-time object detection, edmon, Joseph, et al, <https://arxiv.org/pdf/1506.02640.pdf>

**Figure 3: The Architecture.** Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating  $1 \times 1$  convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution ( $224 \times 224$  input image) and then double the resolution for detection.

# YOLOの仕組み

- ・ 入力画像をCNNによって、特徴マップに変換する。
- ・ 特徴マップ全体を見渡して、各セル毎に、中心が存在しているような物体とそのクラスを同時に予測する。

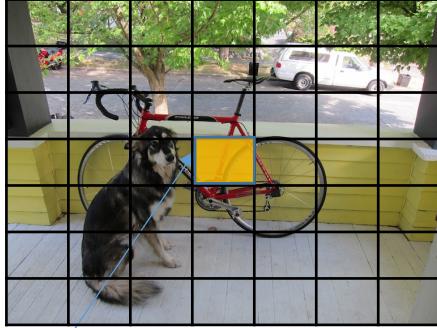


CNN

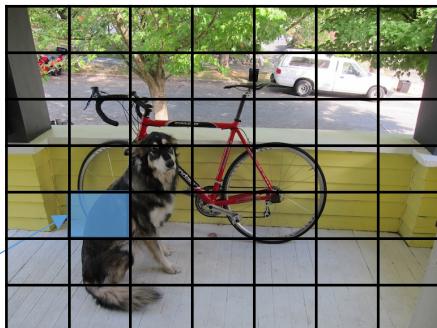


特徴マップ

全結合  
全結合  
全結合



「特徴マップ全体を見渡す限り、このセルには、**自転車**という物体の中心が存在しているそうだな」



「特徴マップ全体を見渡す限り、このセルには、**犬**という物体の中心が存在しているそうだな」



「特徴マップ全体を見渡す限り、このセルには、**自動車**という物体の中心が存在しているそうだな」

- SSD(single shot multibox detector)は、YOLOと似たモデル。
- 様々な階層の出力層からマルチスケールな検出枠を出力できるように工夫されている。
- CNNでは、出力層に近いほど特徴マップのサイズが小さくなるので、それぞれの層の特徴マップには様々なサイズの物体を検出できる情報があるはずである。つまり、出力層に近い特徴マップほど大きな物体を検知しやすくなり、入力層に近い特徴マップほど小さな物体を検知しやすくなると考えられる。
- YOLOより高精度。

複数のスケールの特徴マップを用いて、クラス分類問題とバウンディングボックス回帰問題を解いている。

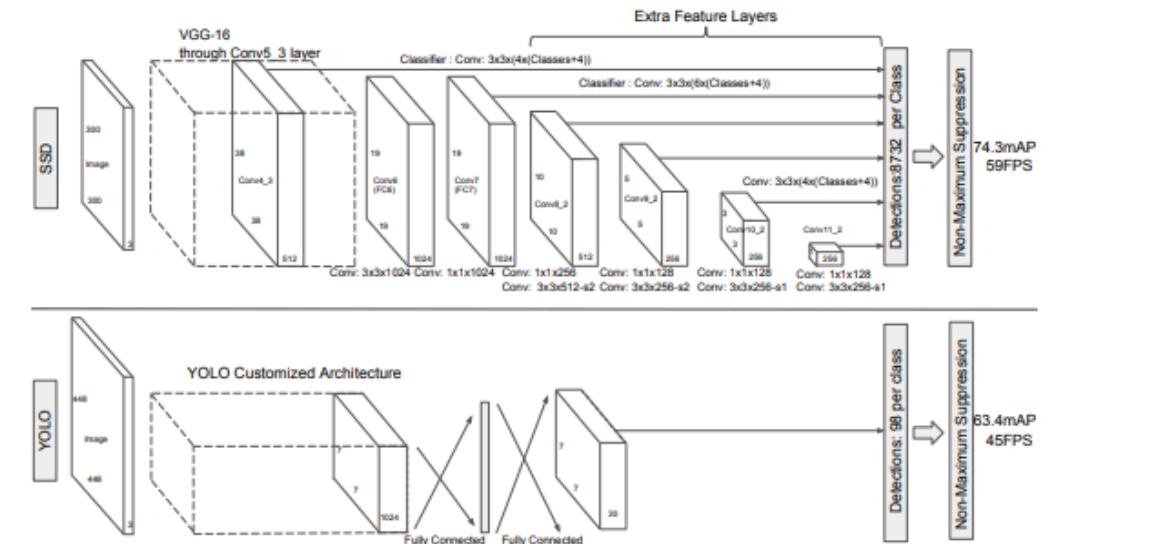
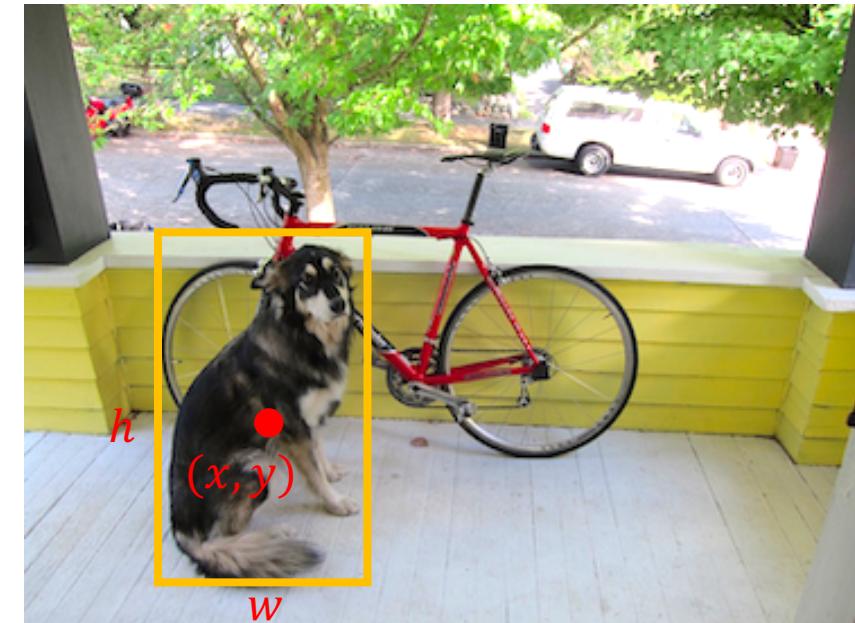
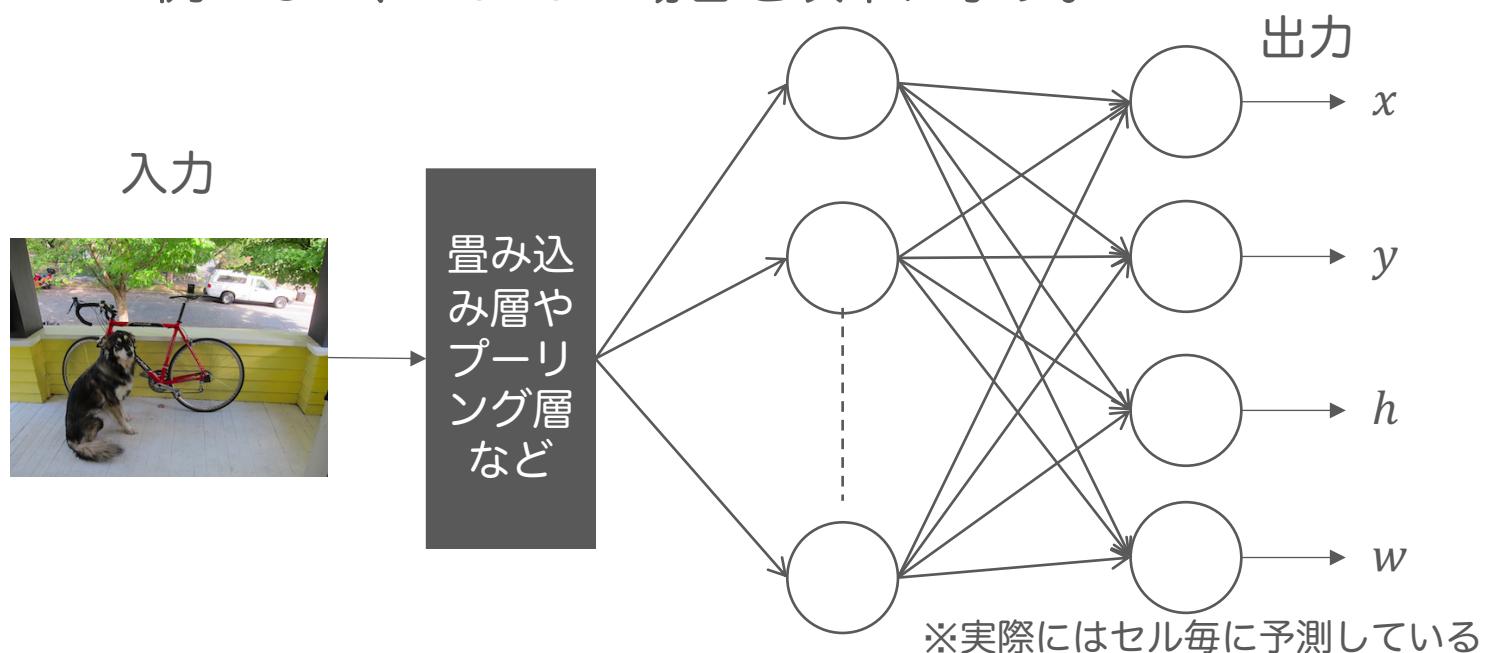


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a  $300 \times 300$  input size significantly outperforms its  $448 \times 448$  YOLO counterpart in accuracy on VOC2007 test while also improving the speed.

引用元： SSD: Single Shot MultiBox Detector, Liu, Wei, et al.  
<https://arxiv.org/pdf/1512.02325.pdf>

# 物体検出モデルにおけるバウンディングボックスの回帰

- 物体検出モデルでよく出てくる「バウンディングボックスを回帰する」とは、CNNによって求められた特徴量から、バウンディングボックスの位置と大きさを定めるための4つの数値を予測するという意味である。
- 例として、YOLOの場合を以下に示す。



$x$  : バウンディングボックスの中心の  $x$  座標<sup>※1</sup>  
 $y$  : バウンディングボックスの中心の  $y$  座標<sup>※1</sup>  
 $h$  : バウンディングボックスの高さ  
 $w$  : バウンディングボックスの幅

※1 学習データにおいては画像全体の相対座標 (0~1の値)。予測する際は、セル毎の相対座標 (0~1の値)を求めてから、画像全体の相対座標 (0~1の値)に変換している。

# 物体検出方法の比較

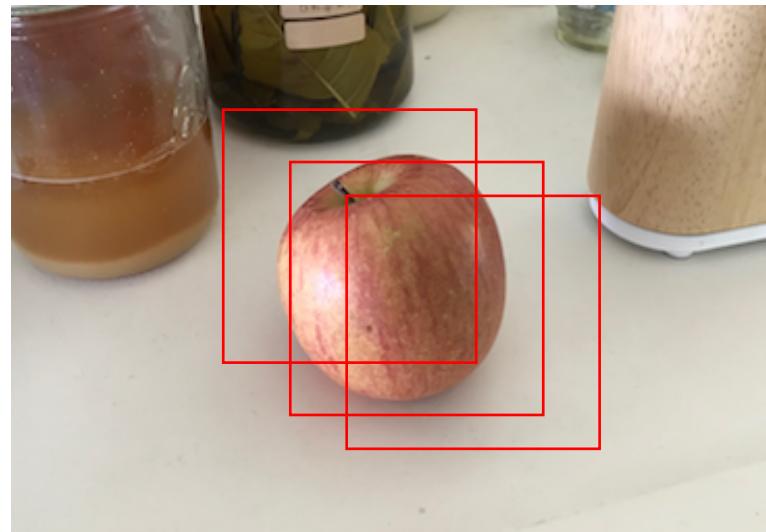
※BB : バウンディングボックス

※特徴マップとは、CNNで計算された特徴量のこと

	R-CNN	Fast R-CNN	Faster R-CNN	YOLO	SSD
特徴	約2000の物体領域候補のそれぞれについて、特徴量を算出する必要がある。	画像全体を入力して計算した特徴マップをすべての提案された領域で再利用することで、R-CNNを高速化させた。	特徴マップから物体領域候補を提案するRPNとFast R-CNNの2つのネットワークで構成される。	画像をグリッド状に分割して考えるのが特徴。分割された1つの領域をセルと呼ぶ。1つのNNで、クラス分類問題とBB回帰問題を解く。	複数のスケールの特徴マップを用いる。
BBの決定方法	選択的検索法で提案された物体領域候補ごとに、特徴マップとBBを回帰させる問題を解く。		RPNで提案された物体領域候補ごとに、特徴マップとBBを回帰させる問題を解く。	特徴マップとBBを回帰させる問題を解く。	
クラス分類の方法	特徴マップを線形SVMに入力しクラスを予測する。	特徴マップを用いて、ニューラルネットワークによりクラス分類問題を解く			
サイズを揃えるための工夫	異なる物体領域候補のサイズを一定のサイズに変換してから、CNNに入力する。	RoIPooling層を利用し、サイズの異なる特徴マップを一定の大きさのベクトルに変換する		特徴マップは一つであるため、サイズを揃える処理は不要。ただし、NNに入力される画像サイズを揃える処理は必要。	

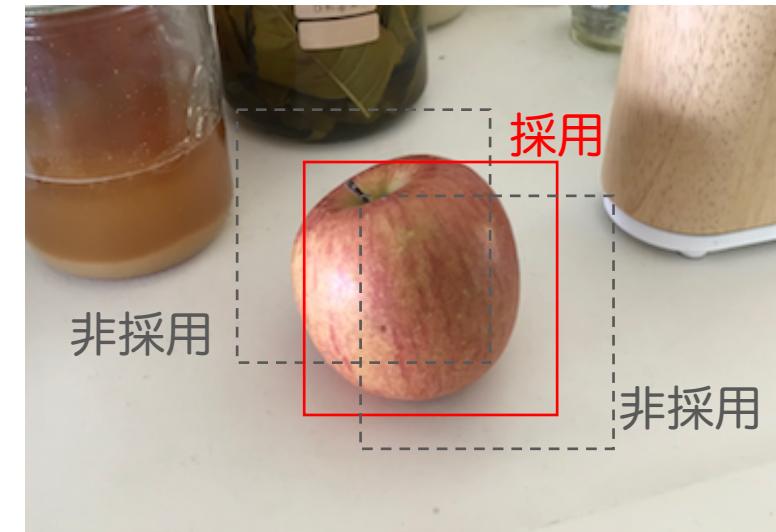
# 非最大値の抑制

- 物体検出の結果として、同一物体に対して複数のバウンディングボックスが検出されてしまうことがある。
- これを防ぐ方法として、非最大値抑制(non-maximam suppression, NMS)がある。



同一の物体に対して、複数のバウンディングボックスが検出されている

NMS  
→



NMSによって、バウンディングボックスを1つに絞ることができる

# 非最大値の抑制

---

- 非最大値抑制の手順

1. スコア(信頼度など)が最も高いバウンディングボックス(以下、BB)に「採用」とマークし、そのBBと一定の割合以上の重なりをもつ(IoUが閾値以上)領域に「非採用」とマークする。
2. 「採用」もしくは「非採用」のマークが付いていないBBの中で、スコアの最も高いBBに「採用」とマークし、これと一定の割合以上の重なりをもつ(IoUが閾値以上)BBに「非採用」とマークする。
3. 全てのBBについて、「採用」または「非採用」のいずれかがマークされるまで上記の手順を繰り返す。
4. 「採用」とマークされたBBの集合が最終的な検出結果となる。

# 物体検出の評価指標

- ・ バウンディングボックスの一致度を測る指標
  - ・ IoU
- ・ 物体検出結果の良さを総合的に測る方法
  - ・ AUC(area under curve)
    - ・ ROC曲線の下部面積
  - ・ AP(average precision)
    - ・ PR(precision recall)曲線の下部面積
  - ・ mAP(mean average precision)
    - ・ APの全クラス平均

<物体検出結果の良さを総合的に測る方法について>

- ・ 物体検出のコンテストでは、独自の定義がされる場合があるので、具体的な計算方法はそれを参照するのがよい。
- ・ 例えば、有名な物体検出コンテストの一つであるThe PASCAL Visual Object Classes (VOC) Challengeでは、年によって計算方法が変わっている。

2006年以前：ROC曲線に基づくAUC

2007年~2009年：AP(11点の近似計算)

2010年~2012年：AP(近似をせずに詳細に計算)

<http://host.robots.ox.ac.uk/pascal/VOC/>

# 物体検出の評価指標, IoU

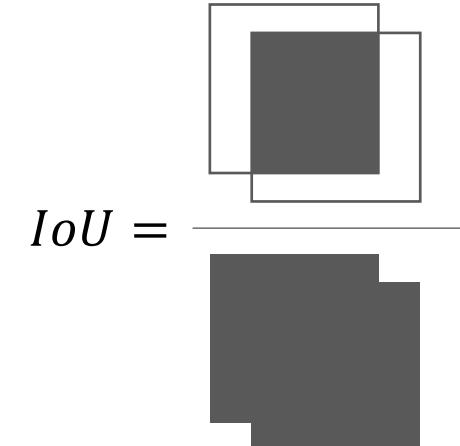
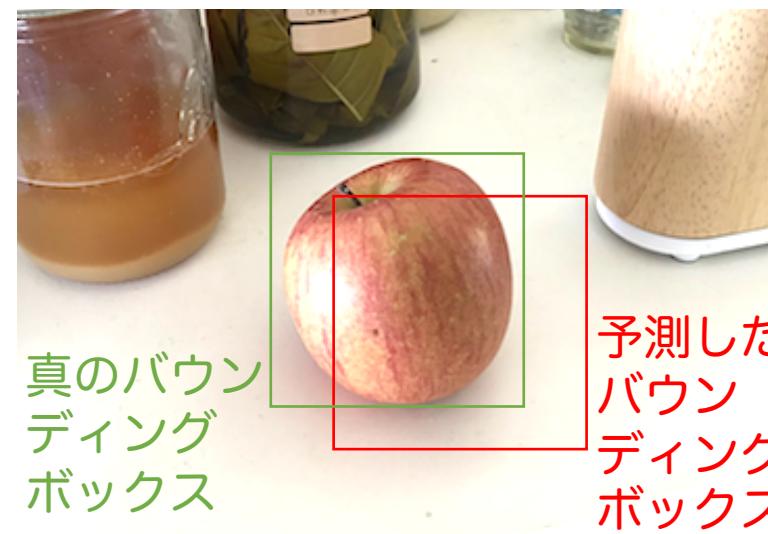
- 予測したバウンディングボックス $\mathcal{R}_p$ と真のバウンディングボックス $\mathcal{R}_g$ の一致度を定量化する指標として、IoU(intersection over union)がある。
- 一般に、IoUが0.5を超える場合に、良いバウンディングボックスの予測結果であると判断されることが多い。
- IoUの定義

$$IoU = \frac{area(\mathcal{R}_p \cap \mathcal{R}_g)}{area(\mathcal{R}_p \cup \mathcal{R}_g)}$$

*area()* : 指定した領域の面積

$\mathcal{R}_p$  : 予測したバウンディングボックス

$\mathcal{R}_g$  : 真のバウンディングボックス



# 物体検出の評価指標, APとmAP

---

- クラスの一致度の評価には、クラスごとの平均適合率(average precision, AP)が利用されることが多い。
- 平均適合率を全クラス平均したものは、mAP(mean average precision)と呼ばれる。

$$\text{mAP} = \frac{1}{N_c} \sum_k^{N_c} AP_k$$

$AP_k$  : クラス  $k$  の平均適合率

$N_c$  : クラスの数

# 物体検出の評価指標, APの計算方法

- APにはいくつかの算出方法があるが、ここでは、2007年~2009年のThe PASCAL Visual Object Classes (VOC) Challengeで用いられた方法を紹介する。
- 参照論文
  - The PASCAL Visual Object Classes (VOC) Challenge  
<http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham10.pdf>
- 参考ブログ
  - mAP (mean Average Precision) for Object Detection  
[https://medium.com/@jonathan\\_hui/map-mean-average-precision-for-object-detection-45c121a31173](https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173)
- 参考書籍
  - MLPシリーズ 画像認識,原田達也,講談社, 7.7.2節
- 参考コード
  - Chainerでの実装例  
<https://chainercv.readthedocs.io/en/stable/reference/evaluations.html#eval-detection-voc>  
[https://github.com/chainer/chainercv/blob/v0.13.1/chainercv/evaluations/eval\\_detection\\_voc.py](https://github.com/chainer/chainercv/blob/v0.13.1/chainercv/evaluations/eval_detection_voc.py)

# 物体検出の評価指標、APの計算方法

## • APの計算方法(VOC2007版)

1. ここでは、正解のバウンディングボックスを物体領域、検出されたバウンディングボックスを検出領域と呼ぶことにする。
2. 物体検出結果として、検出領域およびその領域の信頼度のスコアを算出する。
3. ある1つのクラスについて、以下の計算を行う。
  - A) データセット内の各画像について、以下の計算を行う。
    - a. 当該画像に含まれる各検出領域について、以下を計算する。
      - i. 当該検出領域と、当該画像に含まれる全ての物体領域とのIoUを算出する。
      - ii. どの物体領域ともIoUが50%を超えない検出領域は、偽陽性(false positive, FP)として扱う。
      - iii. IoUが50%を超える物体領域が存在する場合は、当該検出領域をIoUが最も高い物体領域に対応させる。
    - b. 物体領域に対応された検出領域のうち最も信頼度が高いものを真陽性(true positive, TP)、それ以外の検出領域を偽陽性(false positive, FP)として扱う。

# 物体検出の評価指標, APの計算方法

- B) 全ての画像に含まれる全ての検出領域について、最も信頼度が高いものから降順に1つずつ取り出していき、適合率(精度とも呼ばれる, Precision)と再現率(Recall)を計算していく。これは、検出領域を採用する閾値(信頼度のスコア)を変化させながら、適合率と再現率を算出していることに相当する。

$$Precision = \frac{TP}{TP + FP}$$

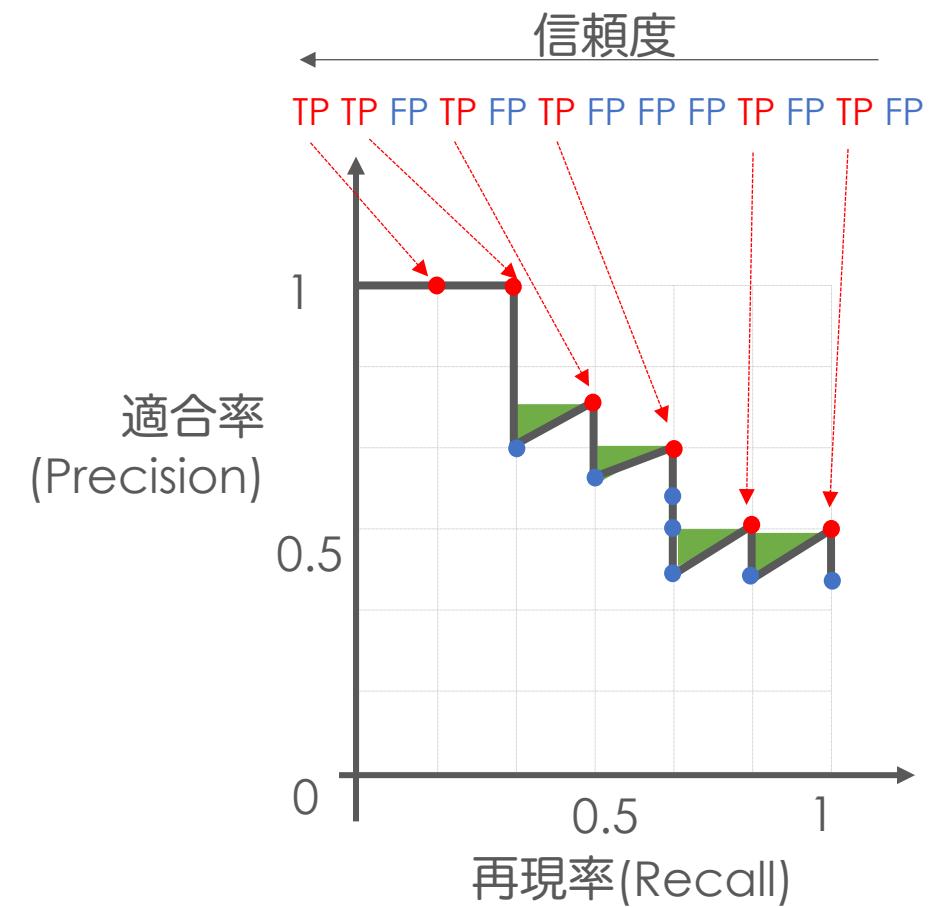
$$Recall = \frac{TP}{n_{pos}}$$

$n_{pos}$ : 該当クラスの物体領域の個数

- C) 全ての適合率と再現率を計算できたら、適合率と再現率のペアをプロットする。これをPR(precision recall)曲線という。
- D) 以下の方法でPR曲線の下部面積を近似的に算出する。この算出結果をAPと呼ぶ。ここで、各再現率 $r$ における適合率 $p(r)$ は、より高い $\tilde{r}$ で発生する $p(\tilde{r})$ で置き換えられる。これにより、右図における緑部面積のように補間されることになる。

$$p_{interp}(r) = \max_{\tilde{r}: \tilde{r} \geq r} p(\tilde{r}), \quad AP = \frac{1}{11} \sum_{r \in \{0, 0.1, 0.2, \dots, 1\}} p_{interp}(r)$$

PR曲線の例



緑部の面積は、近似的に面積を算出したことによって補間された部分

## 物体検出の評価指標, APの計算方法

---

- 具体的なAPの計算手順をAppendix\_AP.ipynbにまとめてある。
- Appendix\_AP.ipynbには、IoU算出の実装例、NMSの実装例、AP算出の実装例を掲載している。
- これら実装はE資格には出題されないので、E資格対策としては計算の大まかな流れだけ押さえておけば良い。

## 実際に物体検出タスクを計算してみたい方は

---

- YOLO
  - <https://pjreddie.com/darknet/yolo/>
  - <https://github.com/pjreddie/darknet>
- SSD
  - <https://github.com/weiliu89/caffe/tree/ssd>
- どちらも学習済みモデルが公開されているので、すぐにテストすることができる。

Any Questions?

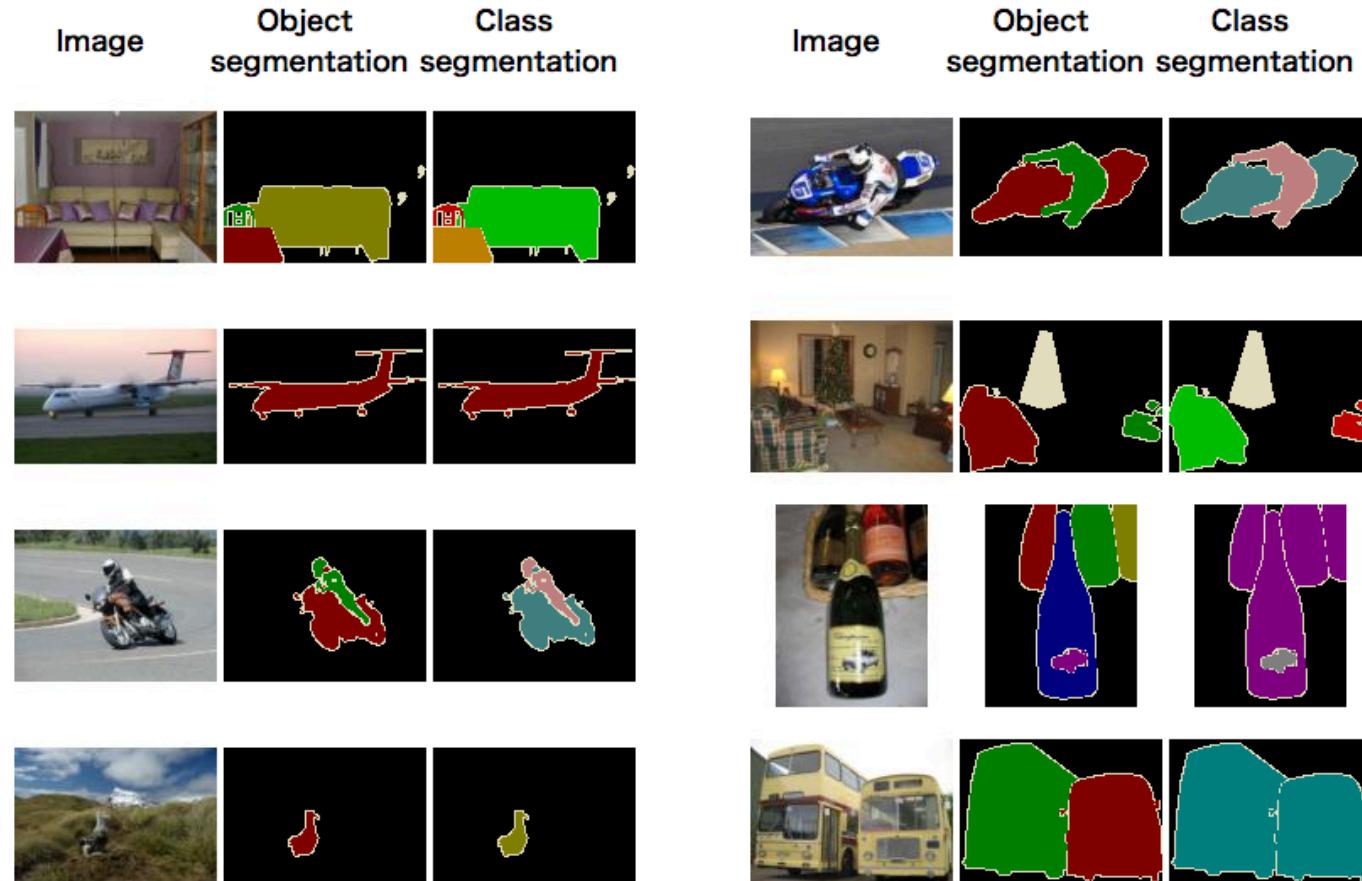
## セマンティックセグメンテーションタスクとCNN

# セマンティックセグメンテーションタスク

- セマンティックセグメンテーションタスクとは、画像を画素レベルで把握するために各画素に対してオブジェクトクラスを割り当てるタスク。簡単に言うと、ピクセル単位の分類問題。
- 自動運転や医療画像の分野において重要な技術。
- 画像分類同様、セマンティックセグメンテーションにおいても CNNは多くの成果を収めている。

## E資格対策

セマンティックセグメンテーションでは、ピクセルごとに分類問題を解くため、いくつかのピクセルが孤立して異なるクラスに割り当てられてしまう問題が発生する。これは、**条件付き確率場 (Conditional Random Field; CRF)** による後処理で改善される。



セマンティックセグメンテーションの例

引用元：<http://host.robots.ox.ac.uk/pascal/VOC/voc2012/segexamples/index.html>

# FCN

- FCN(Fully Convolutional Network, 全層畳み込みネットワーク)は、セマンティックセグメンテーションのためのネットワーク。全結合層を使わないCNN。
- CNNの全結合層を畳み込み層に置き換えることで、出力を分類クラスではなく二次元マップに変えることができる。
- FCN以前はネットワーク内に全結合層が存在することにより、固定サイズの画像しか扱うことができなかった。FCNでは、全結合層が存在しないため、あらゆるサイズの画像でセグメンテーションマップが生成できる。

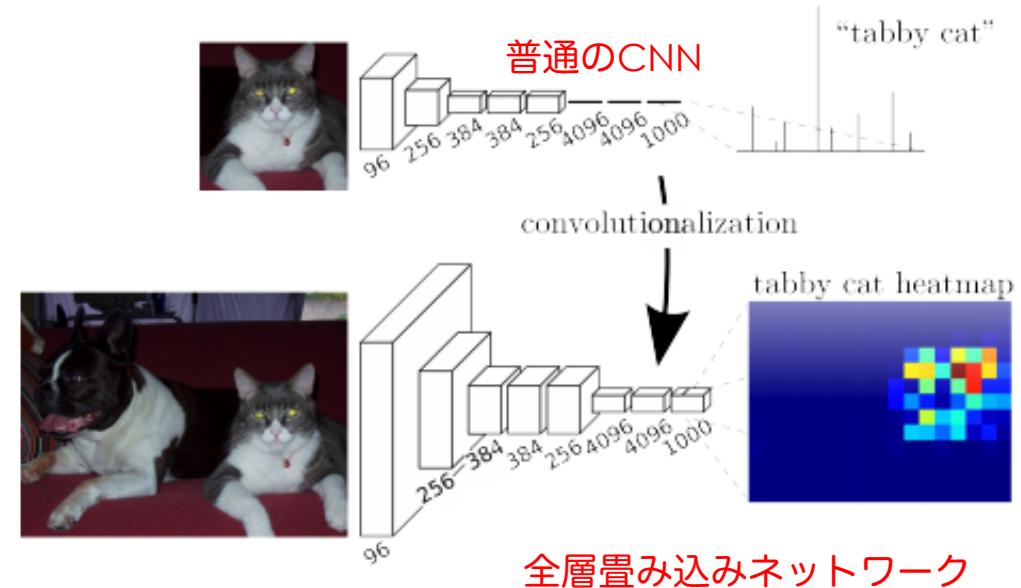


Fig. 2. Transforming fully connected layers into convolution layers enables a classification net to output a spatial map. Adding differentiable interpolation layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end pixelwise learning.

引用元：Fully Convolutional Networks for Semantic Segmentation,  
Evan Shelhamer, Jonathan Long, Trevor Darrell,  
<https://arxiv.org/pdf/1605.06211.pdf>

# FCN

---

- プーリング層では、特徴マップはダウンサンプリングされる。
- FCNでは、プーリング層の後、逆畳み込みを用いてアップサンプリングを施すことで、出力される画像サイズを大きくする。
- 損失は、ピクセル毎のクロスエントロピー誤差の合計。

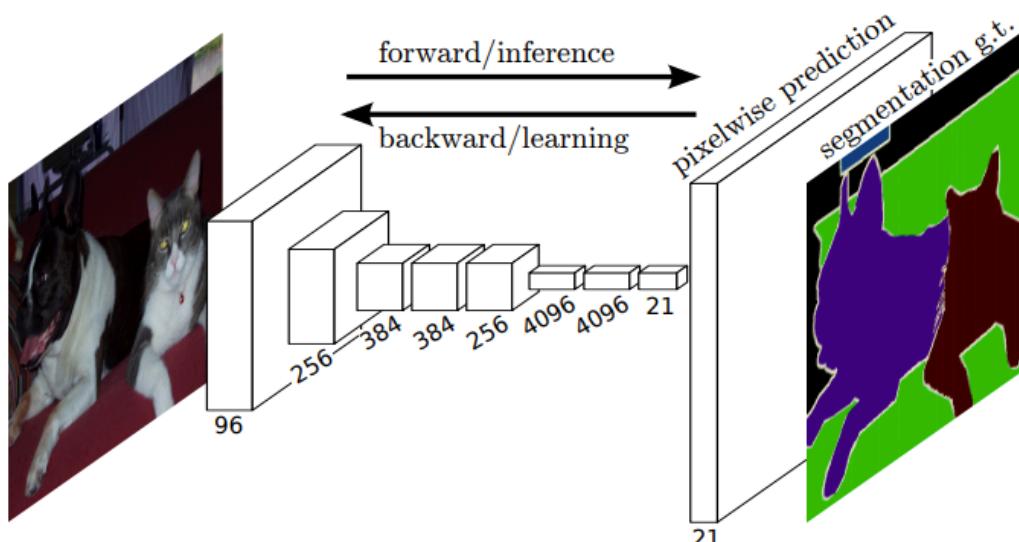
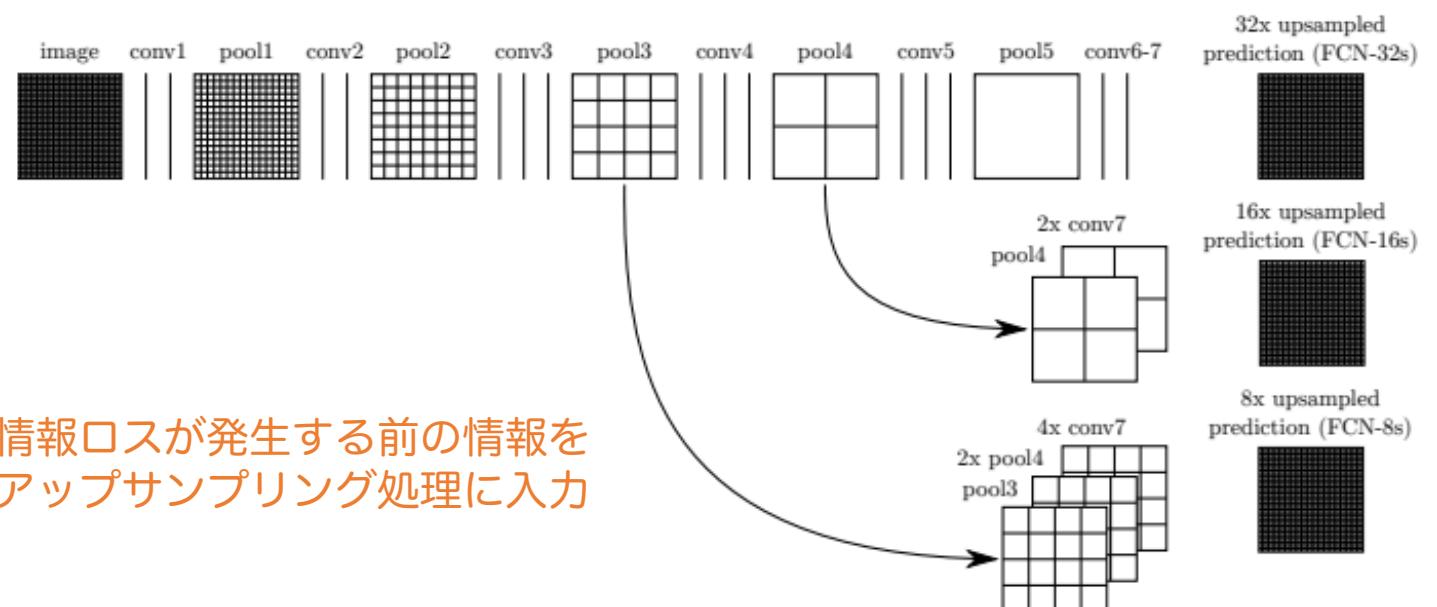


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

引用元：Fully Convolutional Networks for Semantic Segmentation,  
Evan Shelhamer, Jonathan Long, Trevor Darrell,  
<https://arxiv.org/pdf/1605.06211.pdf>

- ・ ダウンサンプリングされた特徴マップに対して単純に逆畳み込みを適用するだけでは、出力結果が粗くなってしまうため、**スキップ接続**を用いて**情報ロスが発生する前の情報をアップサンプリング処理**に入力する。



引用元：Fully Convolutional Networks for Semantic Segmentation, Evan Shelhamer, Jonathan Long, Trevor Darrell, <https://arxiv.org/pdf/1605.06211.pdf>

Fig. 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the pool4 layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from pool3, at stride 8, provide further precision.

# SegNet

- エンコーダ(encoder)部分とデコーダ(decoder)部分で構成されるセマンティックセグメンテーションのためのネットワーク。
- 損失は、ピクセル毎のクロスエントロピー誤差の合計。

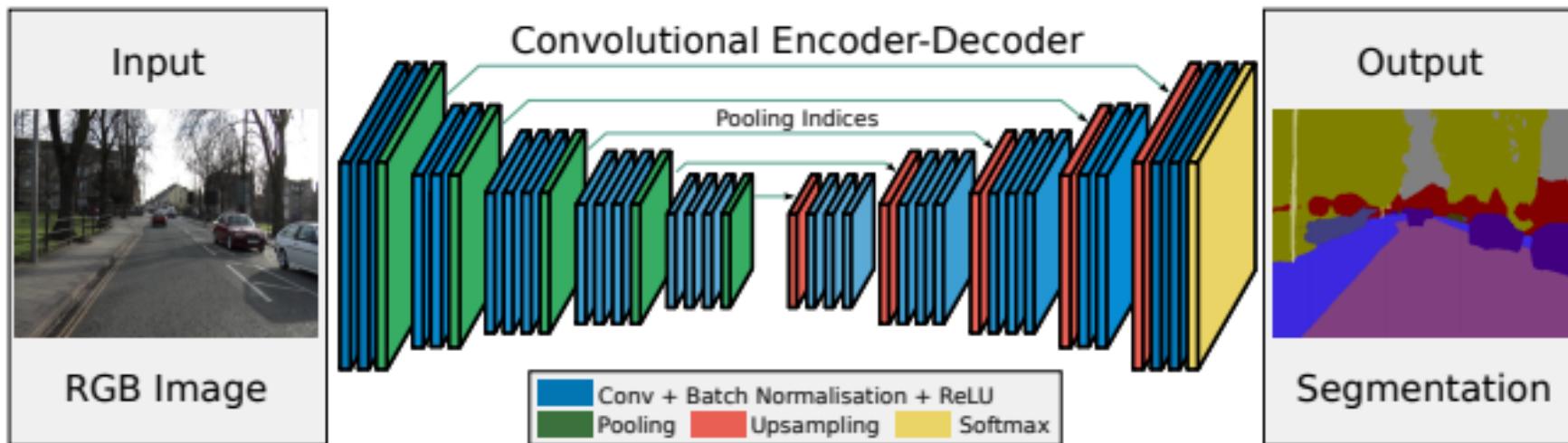


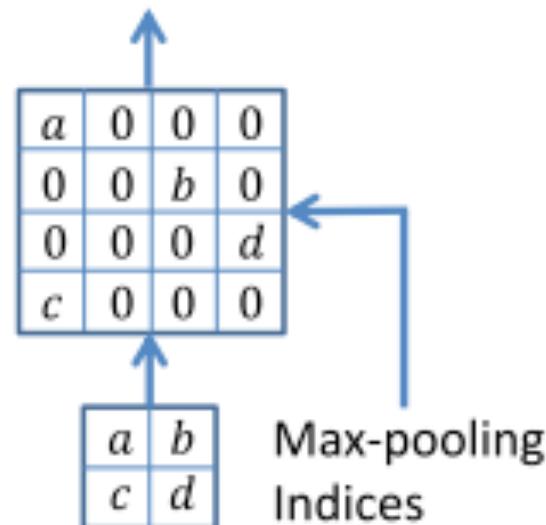
Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

引用元 : SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, Vijay Badrinarayanan, et al.  
<https://arxiv.org/pdf/1511.00561.pdf>

# SegNet

---

- FCNのようにプーリング前の特徴をアップサンプリング層にコピーするのではなく、エンコーダ部分のマックスプーリング層で採用した値の場所を記録しておき、デコーダ部分のアップサンプリング時にその場所を使って特徴マップを拡大する。
- これにより位置情報が保持される。また、FCNに比べてメモリの効率が良くなつた。



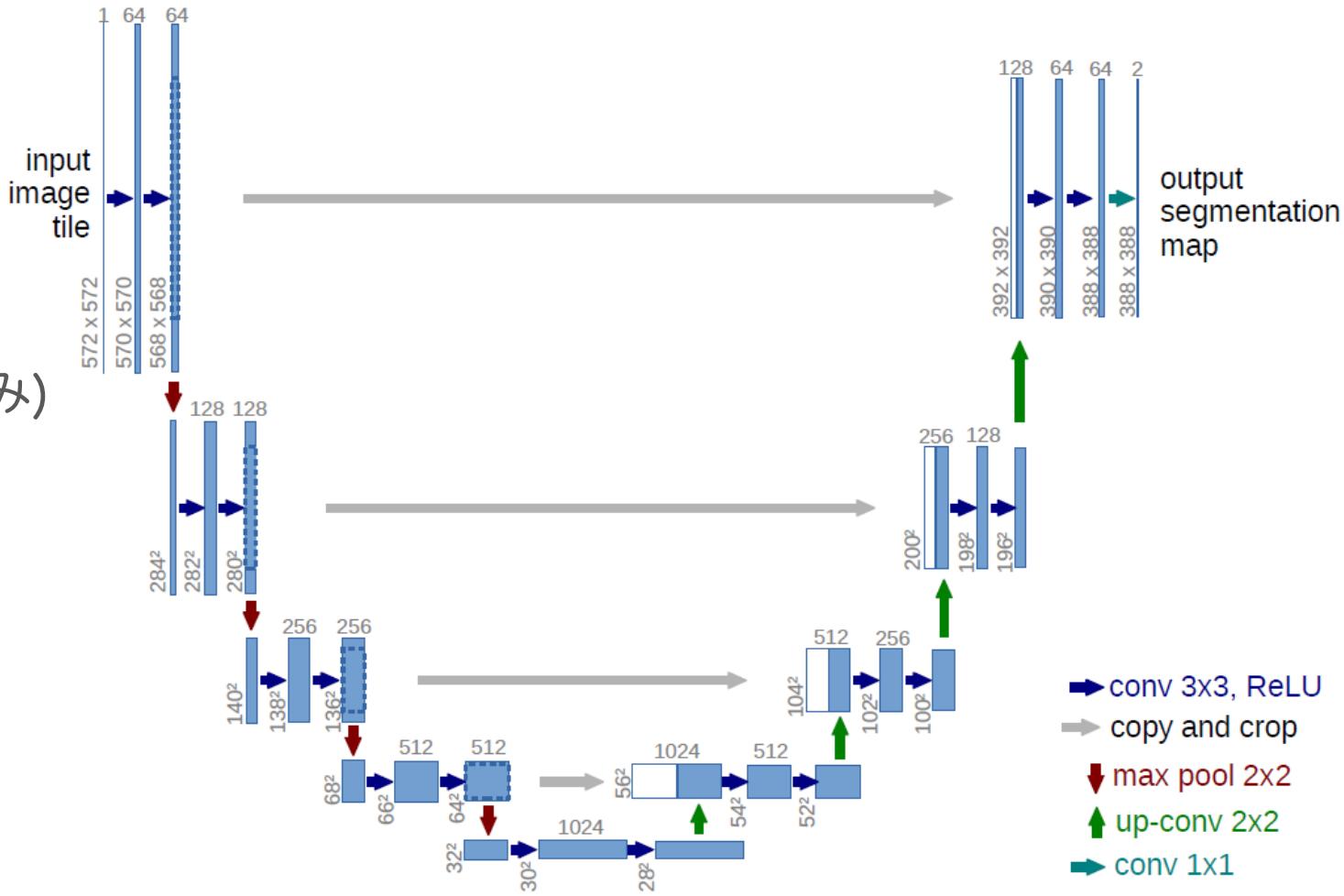
# U-Net

---

- セマンティックセグメンテーションタスク用のネットワーク
- 形がU字型をしているからU-Net
- Fully Convolutional Network(FCN), Segmentation Network(SegNet)を破って SotA(その時点で最高の性能)
- 多くの派生系があり、特に医用画像のセグメンテーションでは主流
- 原論文
  - U-Net: Convolutional Networks for Biomedical Image Segmentation, Olaf Ronneberger, Philipp Fischer, Thomas Brox,(MICCAI 2015)
  - 医用画像などデータ数が少ない場合でもうまく動くセグメンテーション用CNNを提案

# U-Net

- ・ 見た目通りのU-Net
- ・ 横向きのパスは畳み込み
- ・ 下向きのパスはmax-pooling
- ・ 上向きのパスはup-conv (逆畳み込み)
- ・ 最終的な特徴マップを $1 \times 1$  convでセグメンテーションマップに変換

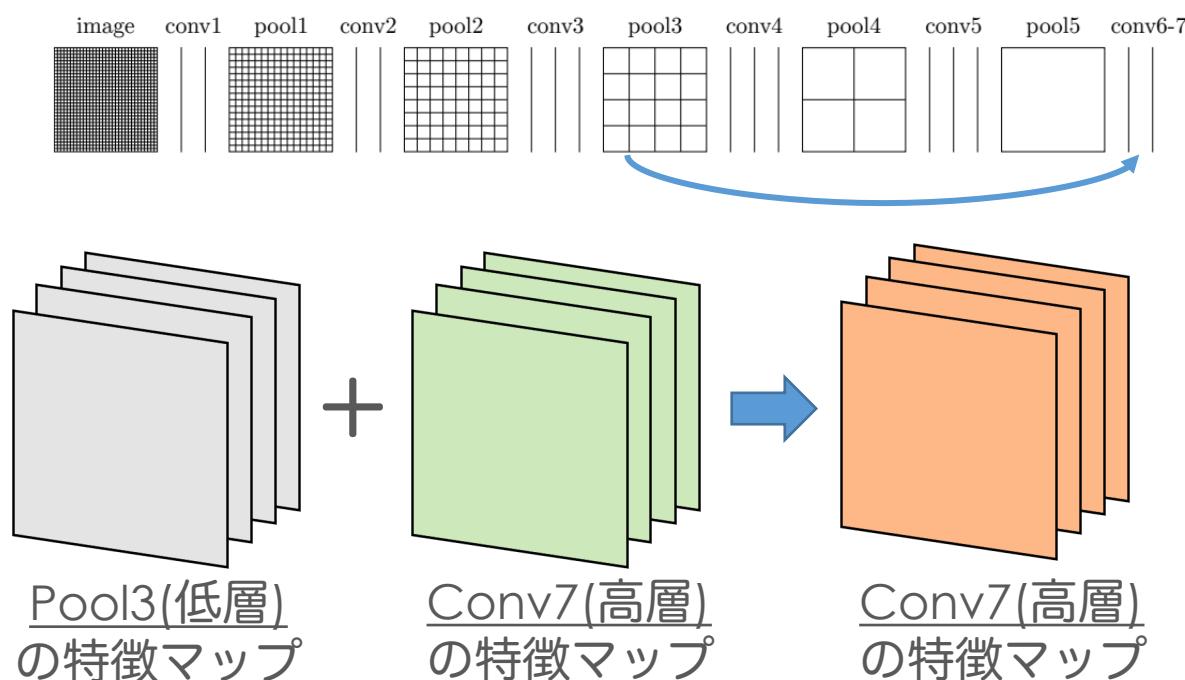


(原論文 Fig. 1から引用)

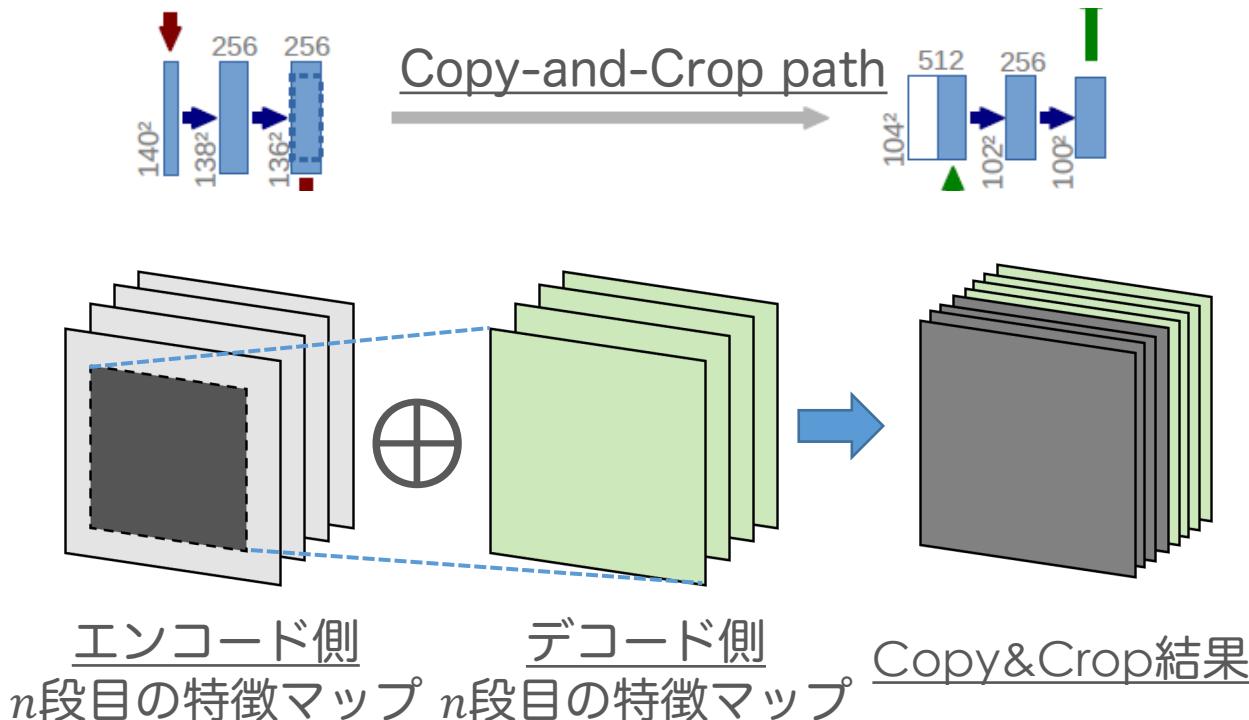
# U-Net

- FCNでは入力側の中間表現を出力側に反映する際、チャンネルごとの足し算を行っていた
- U-Netでは足し算ではなく”連結”することで入力を位置情報を保持している

FCNにおけるSkip-connection



U-NetにおけるCopy-and-Crop



Any Questions?

## 様々な畳み込み

---

# 様々な畳み込み

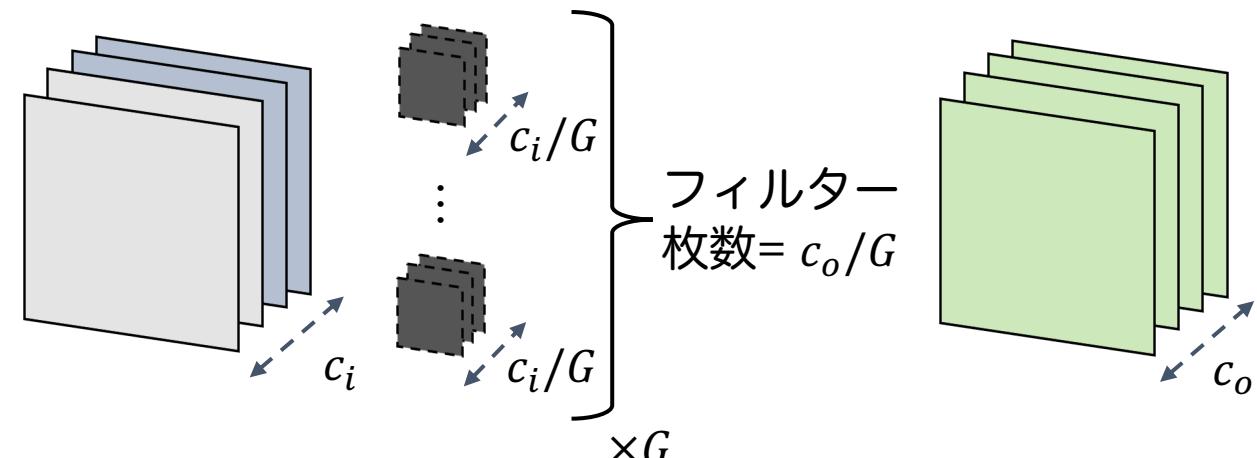
- 近年、様々な畳み込みの方法が提案されている。
- ここでは、特殊な畳み込み方法として以下の手法を紹介する。
  - Grouped Convolution
  - Dilated Convolution
  - Transposed Convolution
  - Depthwise Convolution
  - Pointwise Convolution
  - Depthwise Separable Convolution
- 参考サイト
  - [https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic)
  - <https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728>
  - <https://www.youtube.com/watch?v=T7o3xvJLuHk&feature=youtu.be>
  - <https://arxiv.org/pdf/1603.07285v1.pdf>
  - <https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d>
  - <https://www.youtube.com/watch?v=T7o3xvJLuHk&feature=youtu.be&app=desktop>



MobileNetを参照

# Grouped Convolution

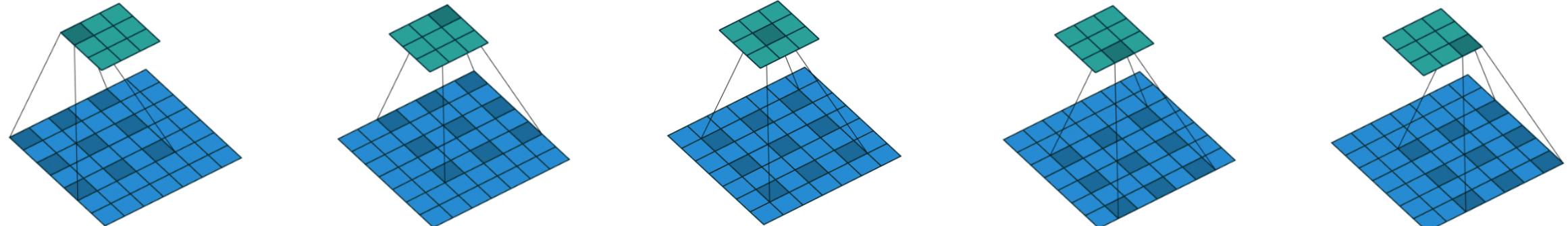
- Grouped Convolutionは、入力された特徴マップをチャンネル方向にグループ化し、グループ毎に畳み込む方法
  - グループ内のみでチャンネル間の相関をとる
  - グループ数 $G$ を多くすればより軽くなる
  - $N_p = c_i \cdot k^2 \cdot c_o // G$  ( $//$ は整除)
    - 入力のチャンネル数 $c_i$
    - 出力のチャンネル数 $c_o$
    - 畳み込みカーネルのサイズ $k^2$
    - 畳み込み層の学習パラメータ数  $N_p$



# Dilated Convolution

---

- 広げられた(dilated)範囲を畳み込む方法。
- フィルターサイズ $k$ が小さい畳み込みでは、大域的な特徴を一度に捉えることができない。
- かと言って、 $k$ を大きくするとその $2$ 乗でパラメータ数が多くなる。
  - $N_p = c_i \cdot k^2 \cdot c_o$
- 少ないパラメータ数で、大域的な特徴を捉えることがDilated Convolutionの目的。

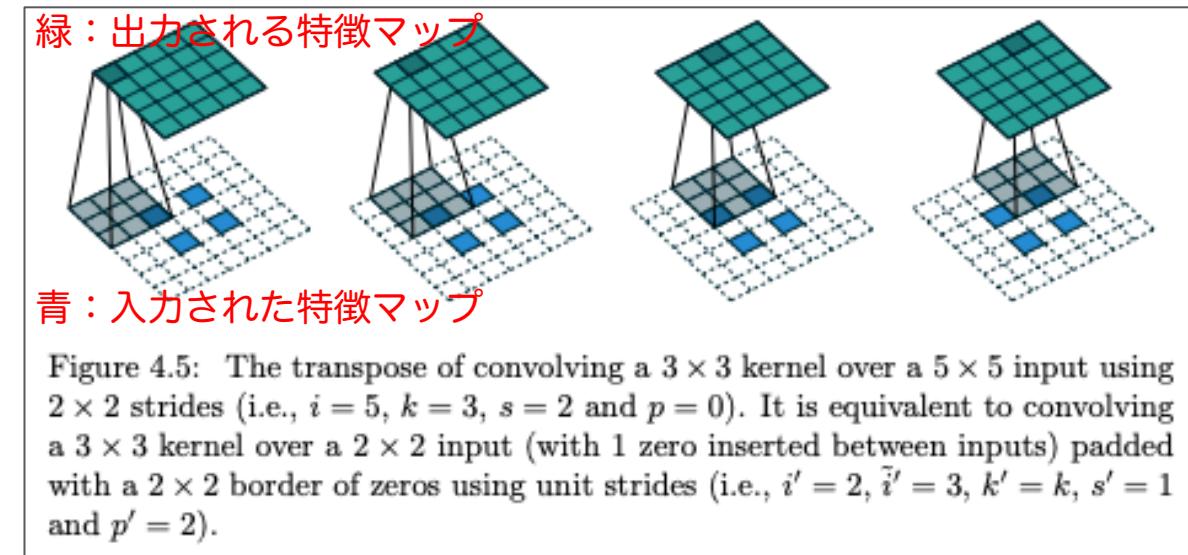
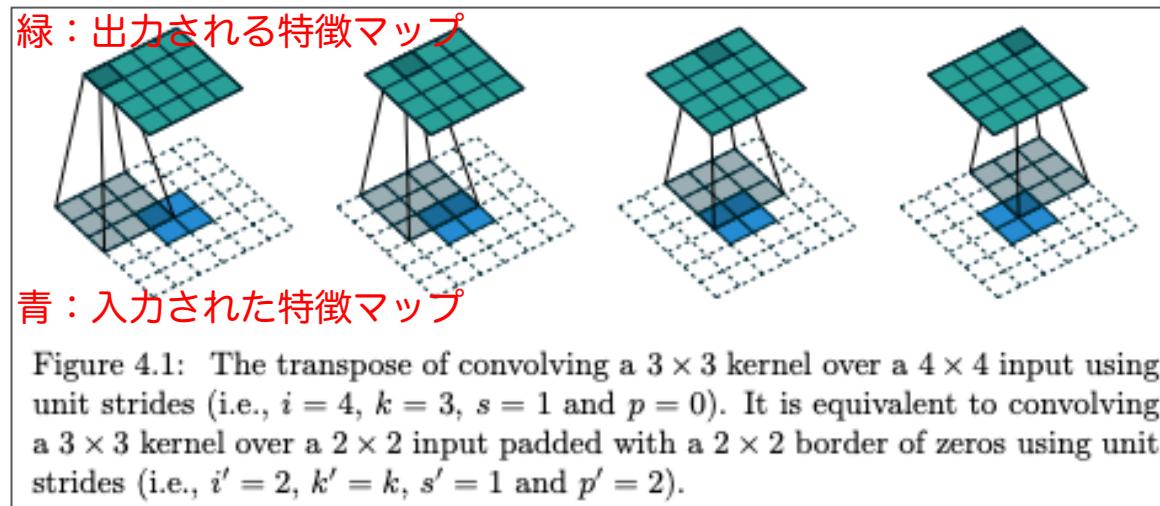


Dilated Convolutionの例

[https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic)

# Transposed Convolution

- Transposed Convolution(転置畳み込み)は、アップサンプリングする際に用いられる手法。
- Deconvolution(逆畳み込み)とも呼ばれる。



<https://arxiv.org/pdf/1603.07285v1.pdf>

## 参考：転置畳み込みという名前の由来

通常の畳み込みは、フィルター行列と特徴マップベクトルの積で計算できる。これに対し、逆畳み込みでは、フィルター行列の転置と特徴マップベクトルの積で計算できる。

参考記事：<https://medium.com/activating-robotic-minds/up-sampling-with-transposed-convolution-9ae4f2df52d0>

参考書籍：MLPシリーズ 画像認識, 原田達也, 講談社, 9.1.3節

# MobileNet

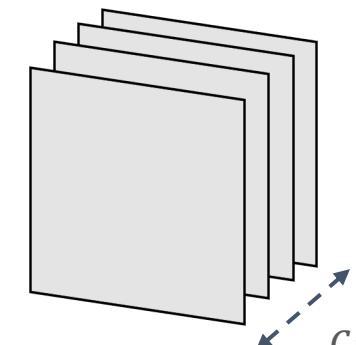
---

- 論文
  - MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, A. G. Howard et al. ,(Arxiv preprint, from Google)
- 主な提案
  - Depthwise Separable ConvolutionによるCNNの計算時間の削減手法
  - ハイパーパラメータで計算時間と認識性能をうまくハンドリングする手法
- 提案の背景
  - 従来のCNNモデルは高性能であるが計算が非常に重い。
  - CNNを軽量化し、スマホなどのモバイル機器でも使えるようにしたい。
  - リアルタイム認識の場合にレイテンシ(≒計算時間)をうまくコントロールしたい。

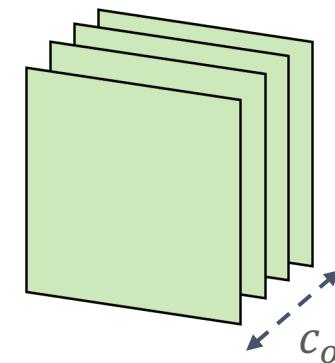
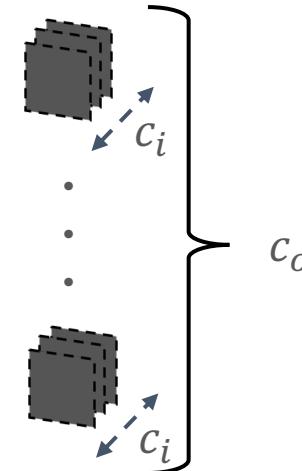
# 畳み込み処理のパラメータ数

- 従来のConvolution処理の何が重い？
- 入力のチャンネル数 $c_i$ , 出力のチャンネル数 $c_o$ , 畳み込みカーネルのサイズ  $k^2$  のときの畳み込み層の学習パラメータ数  $N_p = c_i \cdot k^2 \cdot c_o$ 
  - Convolutionの”計算量”は入力が $h \times w$ ならば  $O = h \cdot w \cdot N_p$
  - パラメータ数を減らすことは当然計算量を減らすことに直結  
※ (計算時間には直結しないことに注意は必要)
- 従来の畳み込みでは 入力マップの”チャンネル間”的相関を律儀に計算している部分がボトルネックになりうる ( $\times c_i$ )
  - ⇒ この部分を効率化すれば軽量なConvolutionになるのでは？

入力マップ

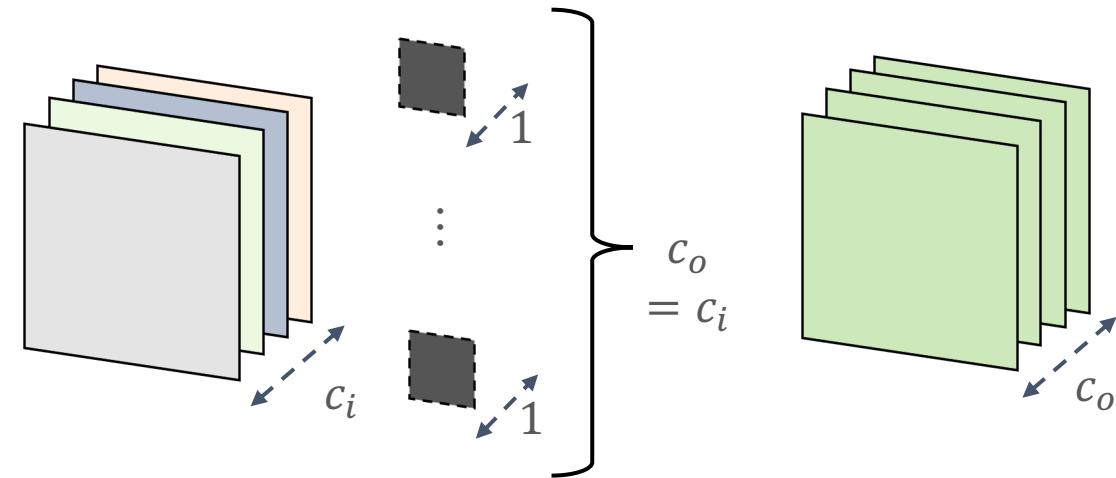


畳み込みフィルタ



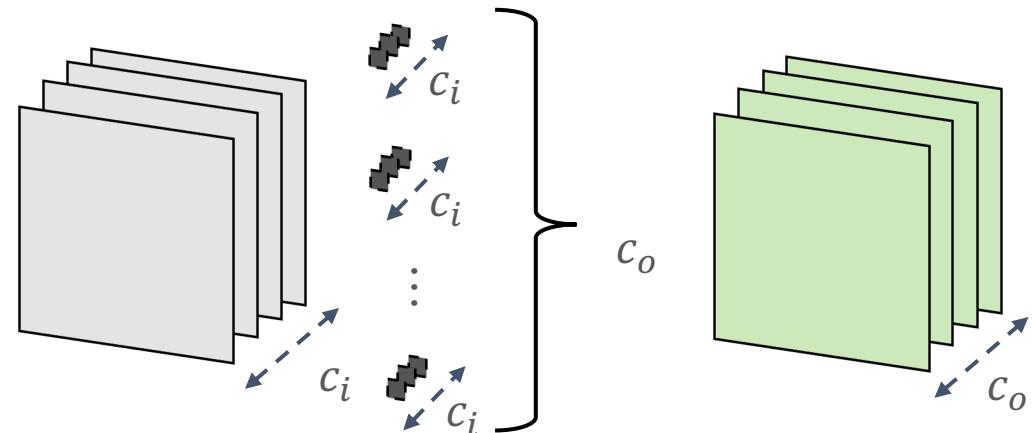
# Depthwise Convolution

- チャンネルごとに空間的畳み込みを行う。
- $c$ 間の相関は取らない。
  - $N_p = c_o \cdot k^2$



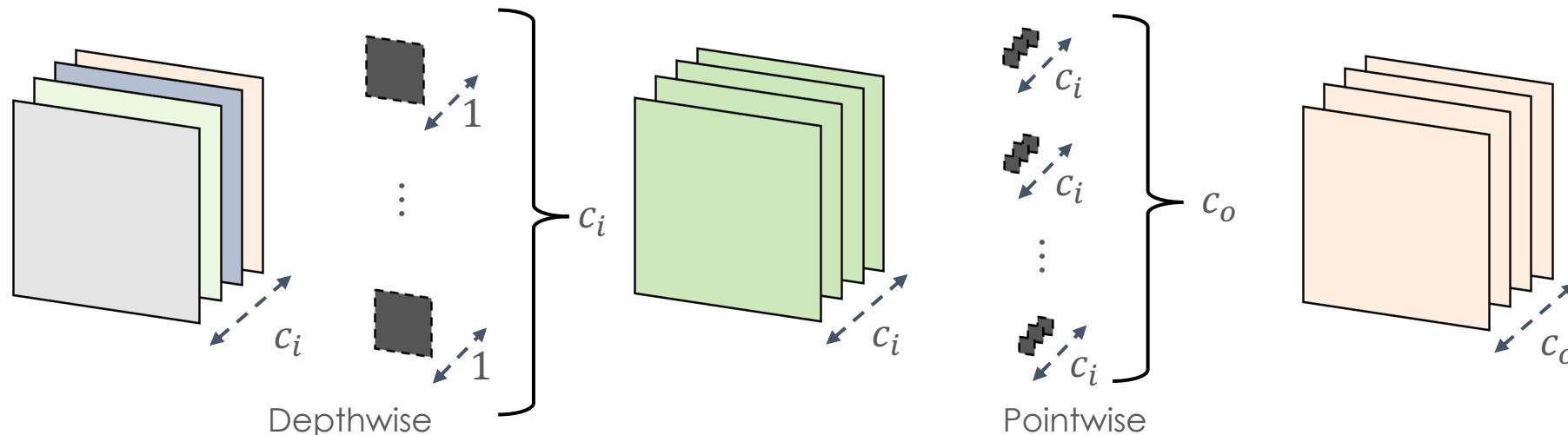
# Pointwise Convolution

- $N_p = c_i \cdot k^2 \cdot c_o$  で  $k^2$  を小さくする方策がある。
  - $k = k^2 = 1$  ならば  $N_p = c_i \cdot c_o$
- 空間的特徴(形状)を捉えずにチャンネル間の相関構造だけを変える特殊な畳み込み演算。
- Pointwise Convolution, 1x1 Convolution, Cascaded Cross Channel Pooling (CCCP) など色々な名前がある。
- 非常にわかりにくいが、モダンなCNNでは超重要技術
  - チャンネル数の辻褄を合わせることができる。



# Depthwise Separable Convolution

- Depthwise Convolutionは非常に軽量だが、チャンネル間の構造を捉えることができない。
  - 出力のチャンネル数 $c_o = c_i$ に固定されてしまうし、性能が出ない。
- Pointwise Convolutionと縦続接続することで「あとで」チャンネルの処理を行う。



- パラメータ数は $N_p = c_i(c_o + k^2)$ 
  - ありがちな $k = 3$ ;  $c_i, c_o > 64$ の場合ではパラメータ数はおよそ1/9に減少する。

Any Questions?

## 様々なプリング

# 様々なプーリング

---

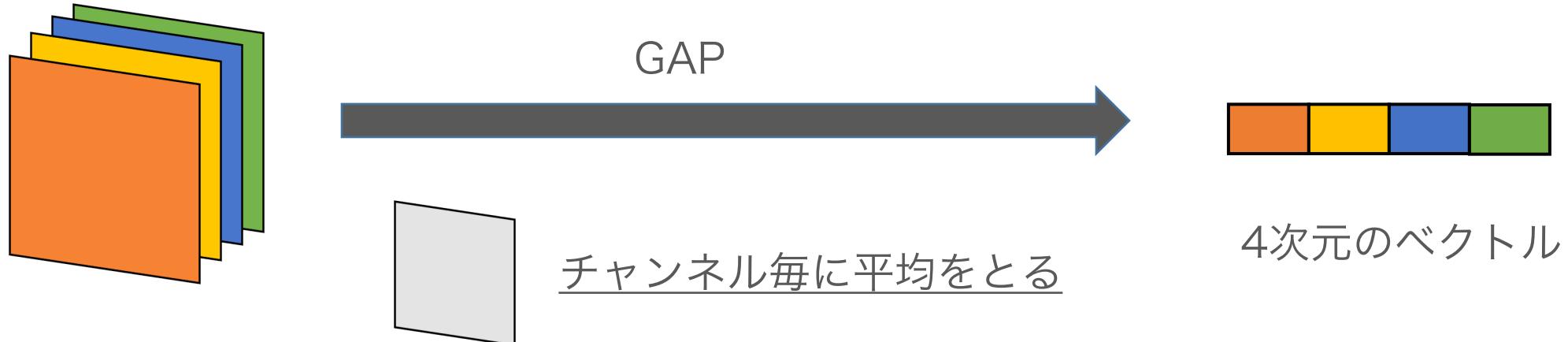
E資格対策

- ・近年、様々なプーリングの方法が提案されている。
- ・ここでは、特殊なプーリング方法として以下の手法を紹介する。
  - ・グローバル・アベレージ・プーリング
  - ・空間ピラミッドプーリング

# グローバル・アベレージ・ポーリング

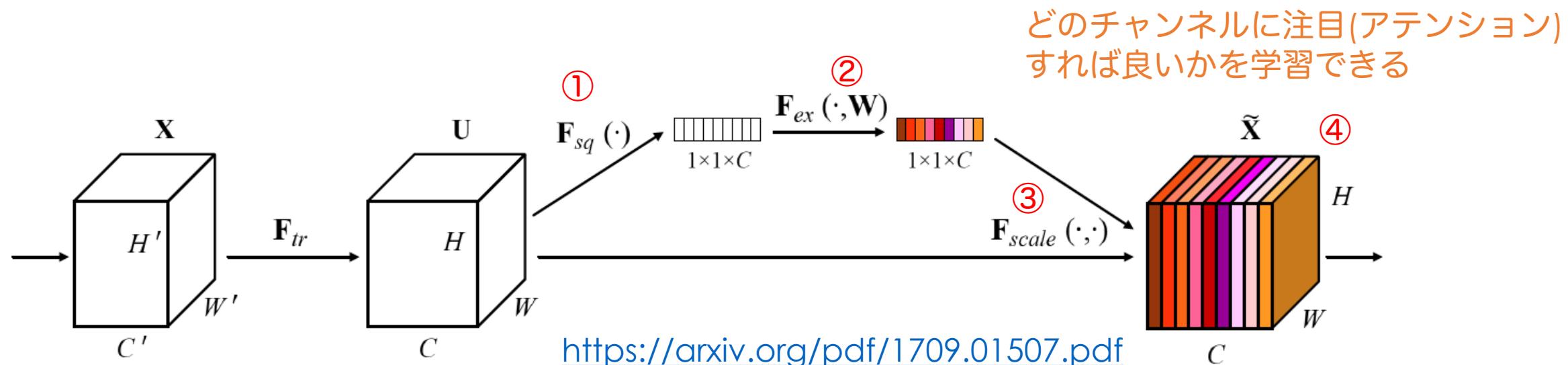
---

- グローバル・アベレージ・ポーリング(Global Average Pooling, GAP)は、入力された特徴マップについて、チャンネル毎に平均を取るポーリング演算。
- 計算結果は、特徴マップではなく、ベクトルになる。



# GAPの利用例

- GAPの利用例として、SE-NetのSE moduleを紹介する
- SE moduleの概要
  - ①GAPで $H \times W \times C$ の特徴マップを潰しC次元のベクトルに変換
  - ②チャンネル毎に重みを掛ける(重みは学習させる)
  - ③重みを掛けた後の値をシグモイド関数に通して(0, 1)に変換
  - ④それを元の特徴マップに掛けて出力する



# 空間ピラミッドプーリング

- 空間ピラミッドプーリング(SPP, spatial pyramid pooling)とは、Microsoft Research Asiaが2014年に開発した新しいプーリング層。
- 入力画像のサイズが異なっている場合でも、**出力のサイズがいつも同じになる**のが特徴。
- 仕組みは単純で、画像を格子状に、 $1 \times 1$ 、 $2 \times 2$ 、 $4 \times 4$ 、 $16 \times 16$ 、...と分割し、その中で、マックスプーリング(ほかのプーリングでもいい)を行う。その後、 $1 \times 1$ の出力+ $2 \times 2$ の出力+ $4 \times 4$ の出力+ $16 \times 16$ の出力...と、つなげたベクトルをSPPの出力とする。
- **出力のサイズはピラミッドのサイズとチャンネル数にのみによって決まる。**
- 参考：  
[http://imagenet.org/challenges/LSVRC/2014/slides/sppnet\\_ilsvrc2014.pdf](http://imagenet.org/challenges/LSVRC/2014/slides/sppnet_ilsvrc2014.pdf)

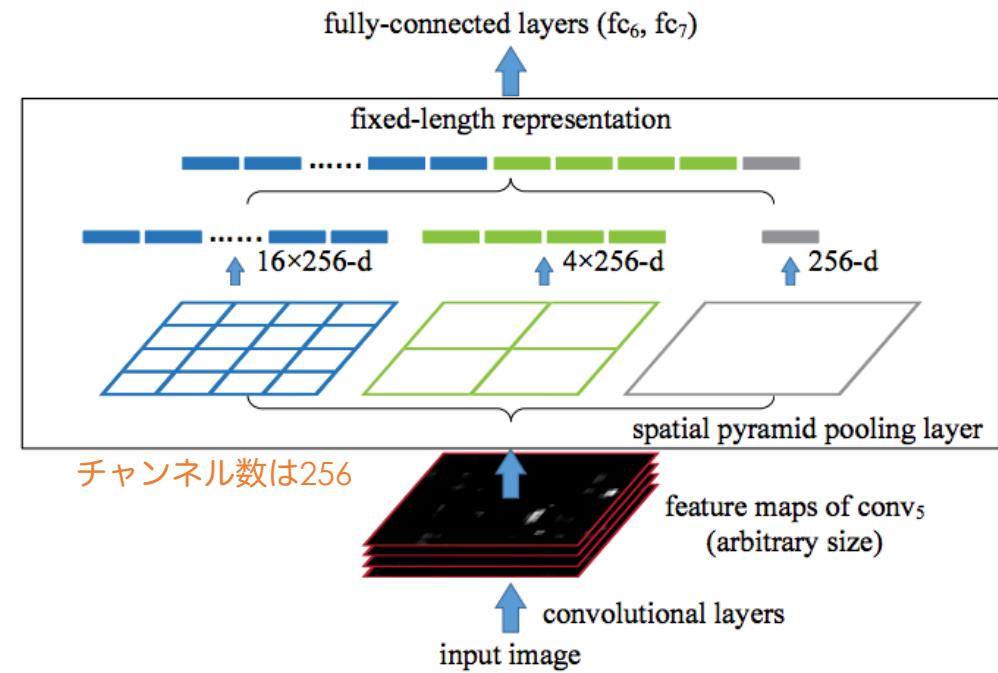


Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the conv<sub>5</sub> layer, and conv<sub>5</sub> is the last convolutional layer.

引用元：Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,,Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun  
<https://arxiv.org/pdf/1406.4729v4.pdf>

Any Questions?

# 自己符号化器

# 自己符号化器とは

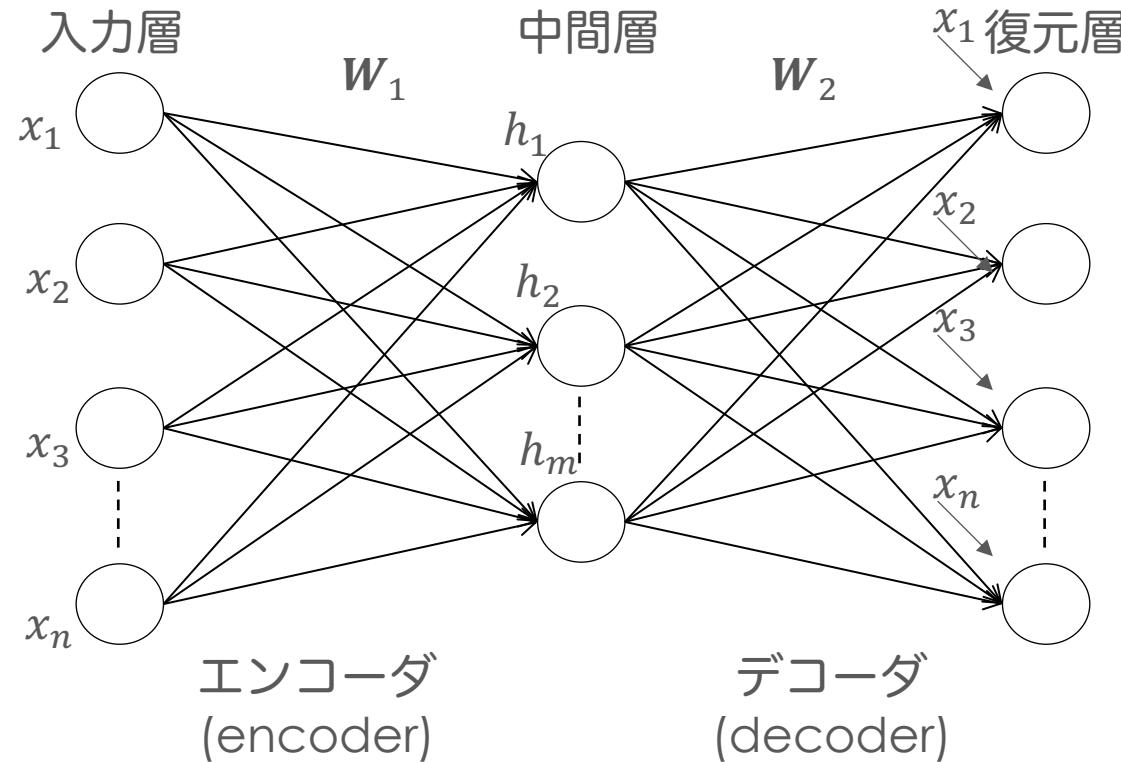
---

- ・自己符号化器(autoencoder)とは、エンコーダとデコーダを組み合わせたネットワークを組み、ノード間の重みを学習する方法。
- ・特徴抽出器(次元圧縮)、ノイズ除去、階層型NNの初期パラメータの取得、復元誤差を利用した異常検知などに利用される。
- ・スパース自己符号化器は、自己符号化器に正則化項を加えて学習させる方法。効率的にユニット数を少なくできる。
- ・雑音除去自己符号化器は、ランダムなノイズを付加したデータを入力層に入れる方法。復元層から出てくるデータがノイズを付加する前のデータに近くなるように学習させる。ノイズ除去の能力が備わることを期待できる。
- ・変分自己符号化器は、生成モデルの1つであり、本来の自己符号化器とは目的やネットワーク構成が大きく異なる。

# 自己符号化器型の計算グラフ

- 自己符号化器型のニューラルネットワークでは、入力層と復元層に同じデータを入れて学習を行う。
- 学習後、データ $x$ を入力したときの中間層の出力 $h$ が特徴量になる。

自己符号化器の例



## [演習] 自己符号化器を用いた次元削減

---

- 5\_1\_Autoencoder.ipynb
  - 自己符号化器を用いて、MNISTデータの次元を削減してみましょう。
  - TensorFlowを用います。

# 生成モデル

# 目次

---

1. 生成モデルとは
2. 変分自己符号化器
3. 敵対的生成ネットワーク

# 生成モデルとは

---

- ・ 生成モデルとは、「観測されたデータはある確率分布から生成されたサンプルである」と仮定するモデリングの方法。
- ・ 生成モデルでは、出力の分布だけでなく、入力の分布もモデル化する。
- ・ これに対し、出力の分布だけを求める方法は、識別モデルと呼ばれる。
- ・ 入力の分布がわかると、人工的なデータを生成することができるようになる。
  - ・ 例、顔写真の生成、室内写真の生成、文章のトピックの生成(トピックモデル)
- ・ 代表的な生成モデルとして、変分自己符号化器 (variational autoencoder, VAE) や敵対的生成ネットワーク (generative adversarial network, GAN) がある。

# クラス分類問題の3つのアプローチ

- クラス分類問題には3つのアプローチがある。参考：『パターン認識と機械学習, C.M.ビショップ, p.42』『イラストで学ぶ機械学習, 杉山将, p.8』
- クラス分類を行いたいだけであれば、識別モデルまたは識別関数で十分。生成モデルでは、クラス分類を行えることに加え、人工的なデータを生成することができる。



入力 $x$ はある確率分布から生成されたサンプルであると考え、その確率分布も同時に求める。

ある入力 $x$ とクラスの対応関係を確率的に考える。ロジスティック回帰, NNなど

確率という考え方は出てこない。SVMなど

## 变分自己符号化器

---

# 変分自己符号化器

---

- ・ 変分自己符号化器(variational autoencoder, VAE)は、生成モデルの1つ。
- ・ 通常の自己符号化器と同じく、入力を再現するように学習を行う。
- ・ 通常の自己符号化器と異なるのは、データが生成するメカニズムを仮定する点。
- ・ 潜在変数に確率分布(多次元正規分布)を仮定する。
- ・ 原著論文
  - ・ Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes.  
<https://arxiv.org/pdf/1312.6114.pdf>

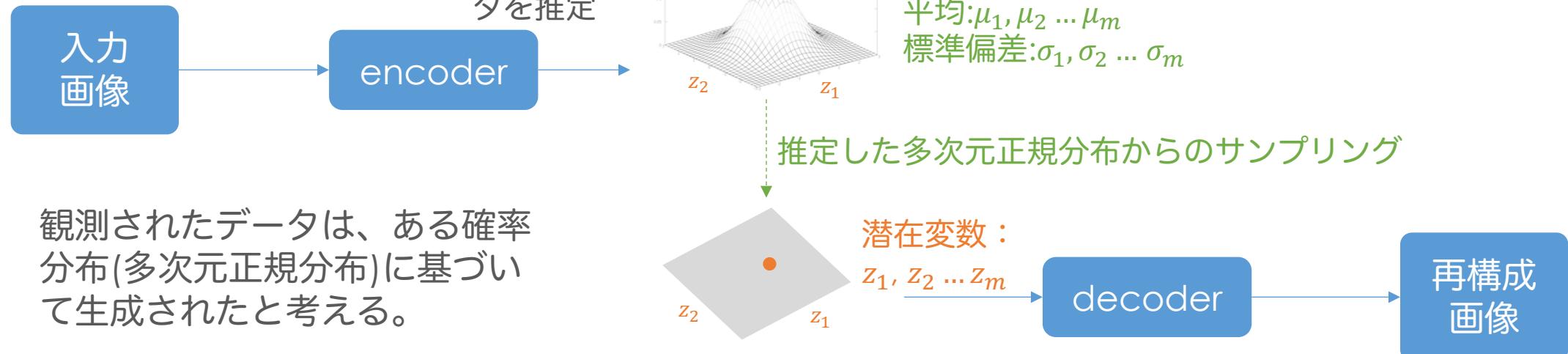
# 変分自己符号化器の概念図

- 通常の自己符号化器と変分自己符号化器の比較を以下に示す。

## 通常の自己符号化器

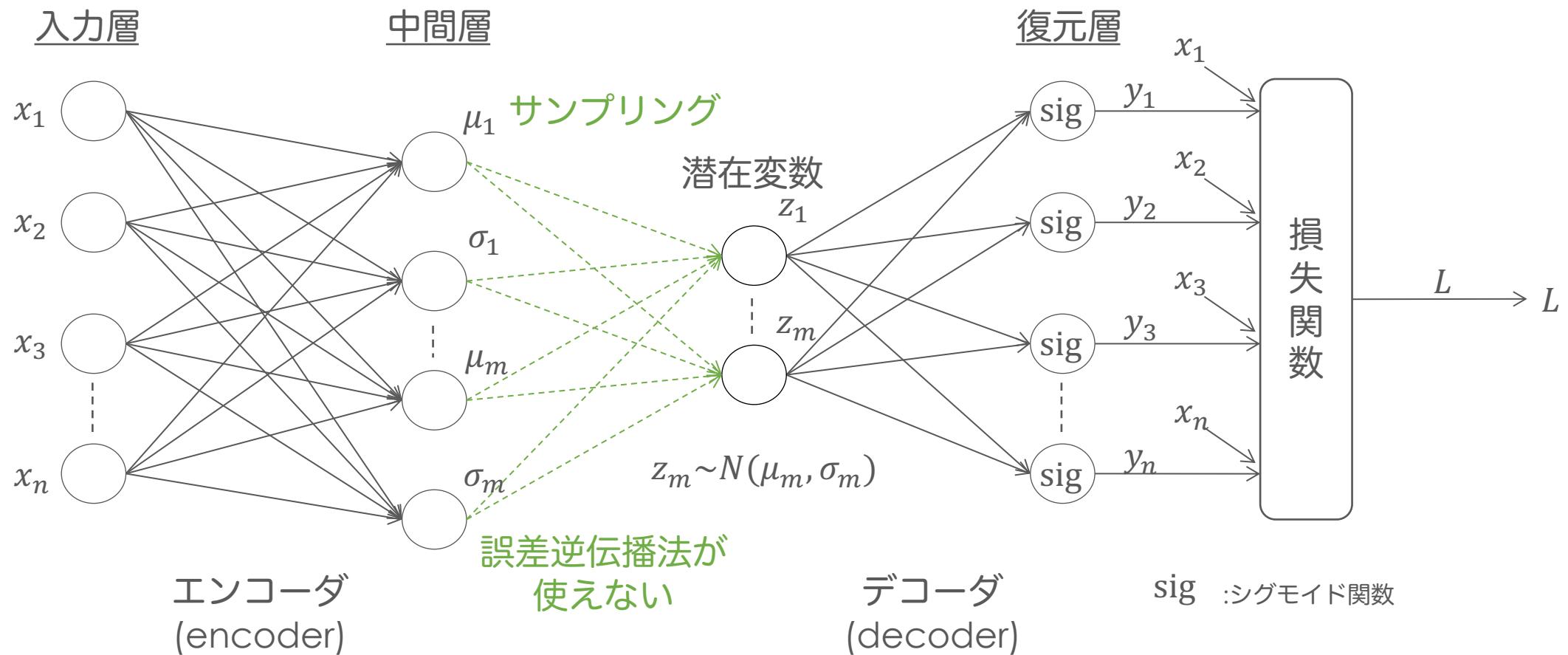


## 変分自己符号化器



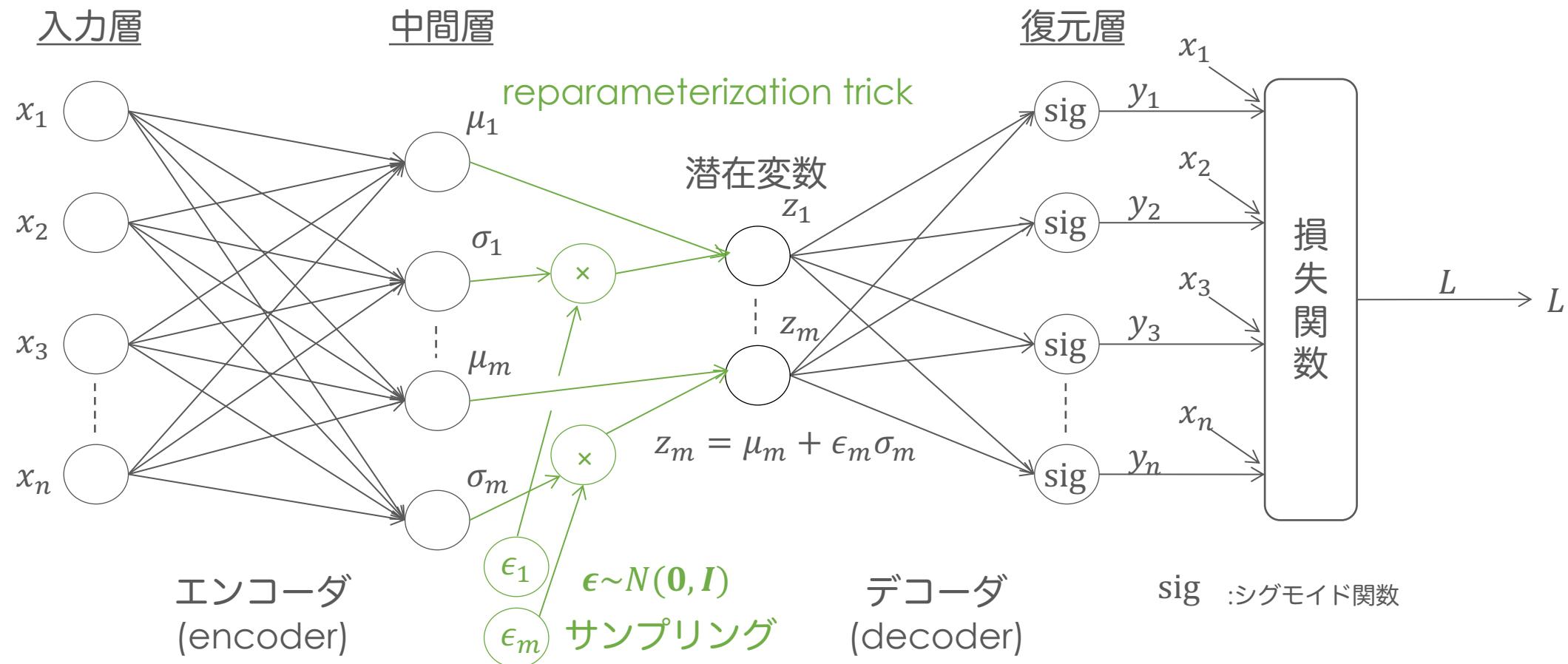
# 変分自己符号化器の計算グラフ

- VAEの計算グラフ例を以下に示す。



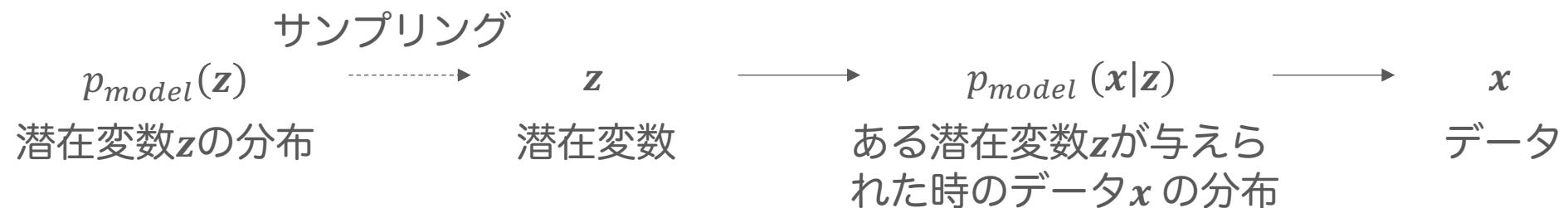
# 変分自己符号化器の計算グラフ

- このままだと、サンプリングのところで誤差逆伝播法が使えないのに、reparameterization trickという方法を用いる。



# 変分自己符号化器の確率モデルとしての解釈

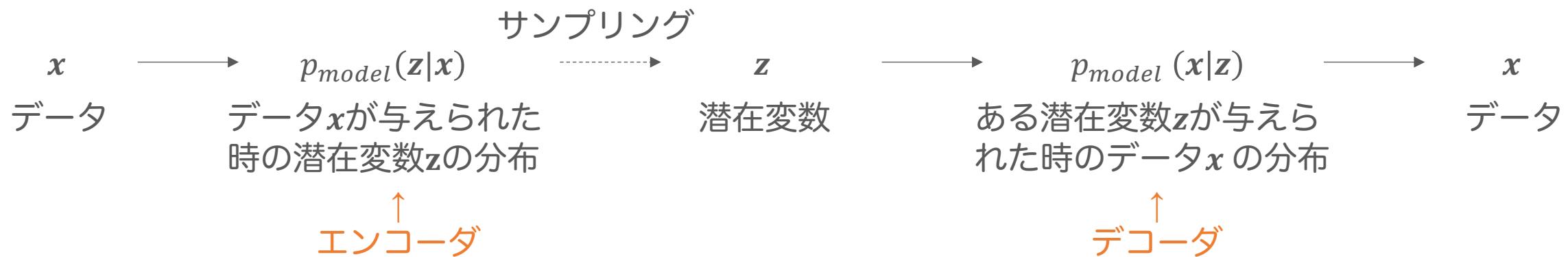
- 最終的に、以下のような生成モデルを作りたい。



- この生成モデルが完成すれば、 **zをサンプリングすることで、様々なデータxを生成する** ことができる。
- $z$ がサンプリングされる空間は、ある程度範囲を絞りたいので、  $p_{model}(z) = N(\mathbf{0}, I)$  とする。
- $p_{model} (x|z)$ が不明なので、学習によって求めるこにする。

# 変分自己符号化器の確率モデルとしての解釈

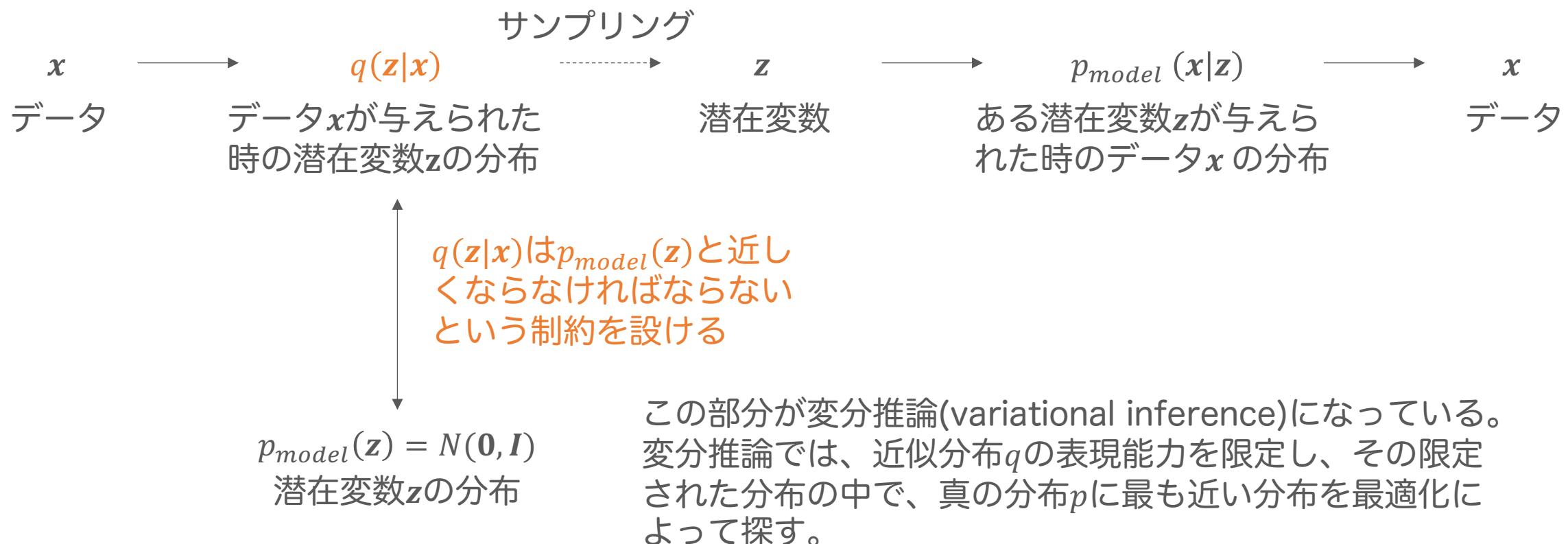
- 自己符号化器を用いれば、以下のように $p_{model}(x|z)$ を学習できそうである。



- しかしこれだと、 $p_{model}(x|z)$ は、学習用データ  $x$  の潜在変数  $z$  が与えられた時に学習用データ  $x$  を再構成できるだけの器になってしまい、生成モデルとして利用できない。(自己符号化器と同じ)
- 本当に欲しいのは、 $p_{model}(z) = N(\mathbf{0}, I)$  からサンプリングした様々な潜在変数  $z$  を与えた時に新しいデータ  $x$  をうまく生成してくれる  $p_{model}(x|z)$ 。

# 変分自己符号化器の確率モデルとしての解釈

- そこで、 $p_{model}(z|x)$ を求めるのではなく、 $p_{model}(z|x)$ に一定の制約をつけた確率分布 $q(z|x)$ を $p_{model}(z|x)$ の代わりに求めることを考える。



# 変分自己符号化器の定式化

- VAEを最尤推定問題として定式化すると以下のようになる。
- 参照：深層学習,Goodfellow,KADOKAWA,5.5節,20.10.3節

$$\mathcal{L}(q) = \mathbb{E}_{z \sim q(z|x)} \log p_{model}(x|z) - D_{KL}(q(z|x) || p_{model}(z))$$

変分下界  
(これを最大化する)

再構成による対数尤度の期待値。  
誤差が小さいほど大きくなる。  
(この項を大きくする)

エンコーダが出力する潜在変数 $z$ の分布を $z$ のモデル分布に近づけるための項。 $q(z|x)$ は近似された分布。  
(この項を小さくする)

$y_i$ はデコーダの出力。シグモイド関数の計算結果になる。

$$\mathbb{E}_{z \sim q(z|x)} \log p_{model}(x|z) = \sum_{i=1}^D x_i \log y_i + (1 - x_i) \cdot \log(1 - y_i)$$

入力が0-1なので、シグモイド関数の出力値に対して、ベルヌーイ分布の尤度関数を適応させる(Dは出力層のノード数)

$$D_{KL}(q(z|x) || p_{model}(z)) = D_{KL}(N(\mu, \Sigma) || N(\mathbf{0}, I)) = -\frac{1}{2} \sum_j^J (1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)$$

近似された分布  
(多次元正規分布)      モデルの分布  
(多次元正規分布)



$J$ は、正規分布の次元数

多次元正規分布の公式を  $D_{KL}$  に代入して導く

$N(\mu, \Sigma)$  の  $\Sigma$  は対角成分に分散が並び、非対角成分が0の行列

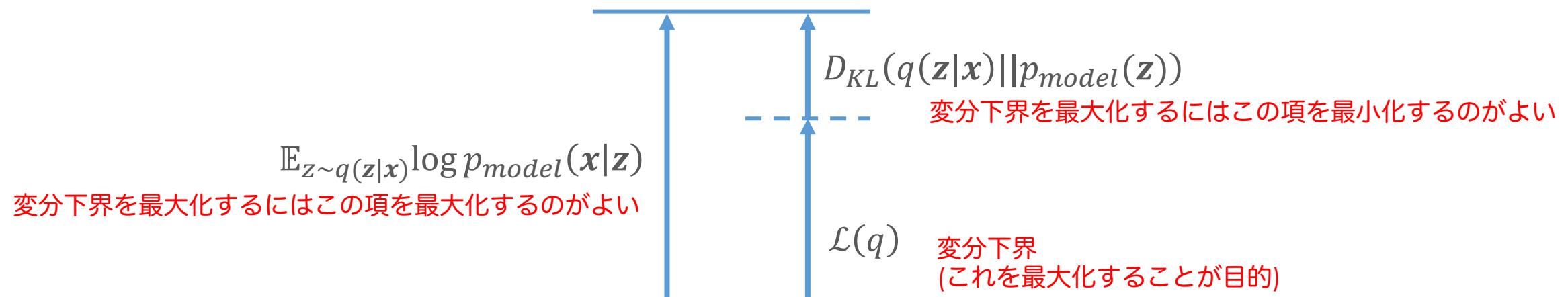
# 変分自己符号化器の定式化

- 変分下界を図示すると以下のようになる。

$$\mathcal{L}(q) = \underbrace{\mathbb{E}_{z \sim q(z|x)} \log p_{model}(x|z)} - D_{KL}(q(z|x) || p_{model}(z))$$

変分下界  
(これを最大化する)  
再構成による対数尤度の期待値。  
誤差が小さいほど大きくなる。  
(この項を大きくする。)

エンコーダが出力する潜在変数 $z$ の  
分布を $z$ のモデル分布に近づけるた  
めの項。 $q(z|x)$ は近似された分布。  
(この項を小さくする)



VAEの学習では、 $q(z|x)$ を  $p_{model}(z)$  に近似させながら、再構成誤差が小さくなるところを探す。

# 変分自己符号化器の損失関数

---

- 前頁で定式化した式を基に損失関数を定義する。

$$L = -\sum_{i=1}^D x_i \log y_i + (1 - x_i) \cdot \log(1 - y_i) - \frac{1}{2} \sum_j^J (1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)$$

---

損失関数	符号を 逆転さ せ、誤 差とし て扱う	ベルヌーイ分布の尤度関数 (出力ノード層毎のクロスエントロピー誤差 に相当する)	KLダーバージェンスに多次元正規 分布の公式を代入した場合の結果
------	---------------------------------	--	-------------------------------------

※  $D$  : 出力層のノード数

※ バッチ処理の場合は、バッチ数N個の平均を計算する。

# カルバック-ライブラーダイバージェンス

---

- カルバック-ライブラーダイバージェンス(Kullback-Leibler divergence, KLダイバージェンス)は、モデルの分布 $q(x)$ が真の分布 $p(x)$ とどれだけ異なっているかを表せる指標。
- 値が大きいほど、2つの分布が異なっていることになる。
- 相対エントロピーとも呼ばれる。

## 連続確率分布の場合

$$D_{KL}(p\|q) = - \int p(x) \ln \frac{q(x)}{p(x)} dx = - \int p(x) \ln q(x) dx - \left( - \int p(x) \ln p(x) dx \right)$$

## 離散確率分布の場合

$$D_{KL}(p\|q) = - \sum p(x) \ln \frac{q(x)}{p(x)}$$

$D_{KL}(p\|q)$  : KLダイバージェンス

$p(x)$  : 真の分布

$q(x)$  : モデルの分布

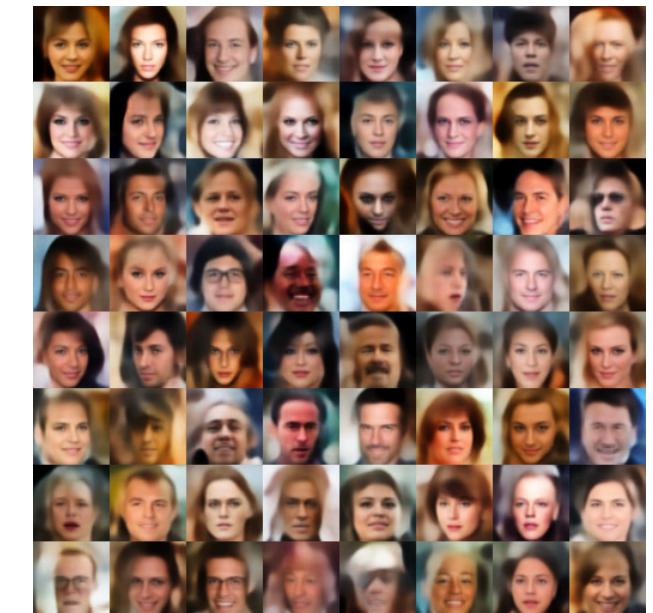
## KLダイバージェンスの性質

- $D_{KL}(p\|q) \geq 0$
- $p(x) = q(x)$  のとき、 $D_{KL}(p\|q) = 0$
- $D_{KL}(p\|q) \neq D_{KL}(q\|p)$

# 変分自己符号化器の例

---

- Morphing Faces
  - 顔画像を自動生成できるモデルをVAEによって構築した例
  - [https://vdumoulin.github.io/morphing\\_faces/](https://vdumoulin.github.io/morphing_faces/)
  - VAEによって、400個の潜在変数を学習
  - 上記デモでは、400個のうち29個を変化させることができる
- vae-celebA
  - celebAデータセットを用いて、有名人の顔画像を生成した例
  - <https://github.com/yzwxx/vae-celebA>



## [演習] VAEによる連続した画像の生成

---

- 5\_2\_VAE.ipynb
  - 変分自己符号化器を訓練し、連続した画像を生成しましょう。
  - TensorFlowを用います。
- 5\_3\_KLdivergence.ipynb
  - カルバック-ライブラーダイバージェンスの計算方法を確認しましょう。

## 敵対的生成ネットワーク

---

# 敵対的生成ネットワークとは

---

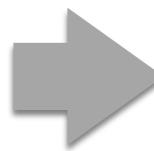
- ・ 敵対的生成ネットワーク(generative adversarial network, GAN) は、以下の論文で提案された生成モデル。
- ・ 原著論文
  - ・ Ian J. Goodfellow et al., Generative Adversarial Networks, <https://arxiv.org/pdf/1406.2661.pdf>
- ・ 参照：深層学習,Goodfellow,KADOKAWA, 20.10.4節
- ・ 二人のプレーヤーが競い合うように学習させるのが特徴。
- ・ 生成器ネットワーク
  - ・ 偽物を作る側 (Generator)
  - ・ Discriminatorを騙すことが目的
  - ・ 本物そっくりの偽物を作るよう学習していく
- ・ 識別器ネットワーク
  - ・ 偽物を見破る側 (Discriminator)
  - ・ Generatorの作った偽物を見破るのが目的
  - ・ ほんのわずかな違いも見破れるよう学習していく

# 敵対的生成ネットワークの概念図

ランダムなデータ  
(正規分布からのサンプリングなど)

正解データ  


入力



鑑定者が本物と間違う程の偽物を作れるように学習が進む



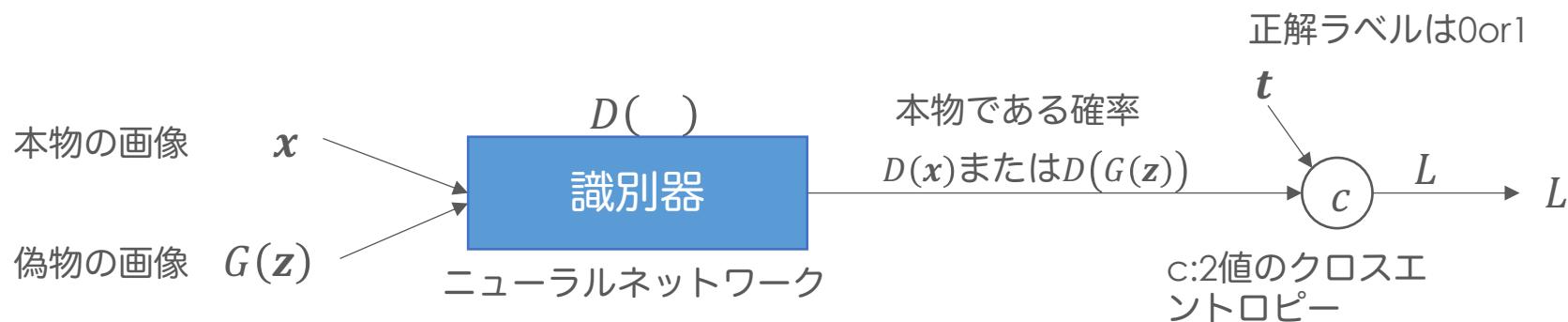
鑑定者が騙される程の偽物



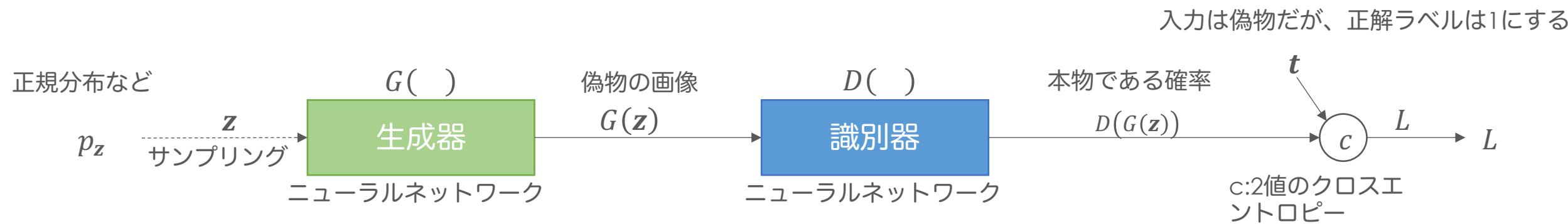
正解データか偽物かを精度良く判定できるように学習が進む

# 敵対的生成ネットワークの学習

- ・ 敵対的生成ネットワークでは、識別器の学習と生成器の学習を繰り返す。
- ・ 理想的に学習できると、識別器は本物と偽物を区別できなくなり、 $D(x) = 0.5$ に行き着く。
- ・ 識別器の学習



- ・ 生成器の学習



# 敵対的生成ネットワーク

- 敵対的生成ネットワークにおける学習を定式化すると以下のようになる。

*Gは評価関数Vを最小にしたい*  
↓      ↓  
*Dは評価関数Vを最大にしたい*

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

*Dが最大化したい部分*      *Gが最小化したい部分*

$\min_G$	$\max_D$	$V(D, G)$	=	$\mathbb{E}_{x \sim p_{data}(x)} [\log D(x)]$	+	$\mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$
<i>評価関数</i>	<i>本物の画像xを D( )に入力し た時の期待値</i>	<i>本物である 確率</i>		<i>偽物の画像G(z) をD( )に入力 した時の期待値</i>		<i>偽物である確率</i>

- Discriminator(D)*は、本物が入力された時に出力される「本物である確率」を最大化しようとする。また、偽物が入力された時に出力される「偽物である確率」を最大化しようとする。
- Generator (G)*は、Discriminatorに生成データを入力した時に出力される「偽物である確率」を最小化しようとする。
- 両者のバランスがとれる点をナッシュ均衡(Nash equilibrium)という。

# 代表的なGAN

---

- DCGAN
- LAPGAN
- infoGAN
- Conditional GAN
- pix2pix
- cycleGAN
- StackGAN

# DCGAN

- CNNを用いて画像を生成するGAN。
- バッチ正規化、逆畳み込み、LeakyReLUが使われている。
- 原著論文
  - Alec Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, <https://arxiv.org/pdf/1511.06434.pdf>



## Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

fractional-strided convolutionsとは、逆畳み込みのこと

## 生成器のネットワーク

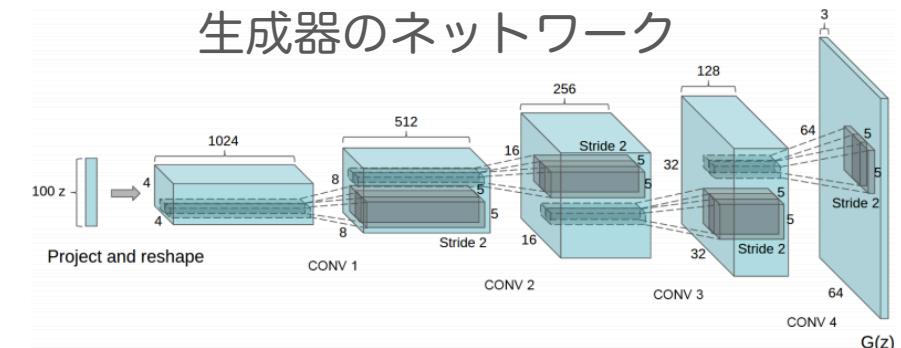
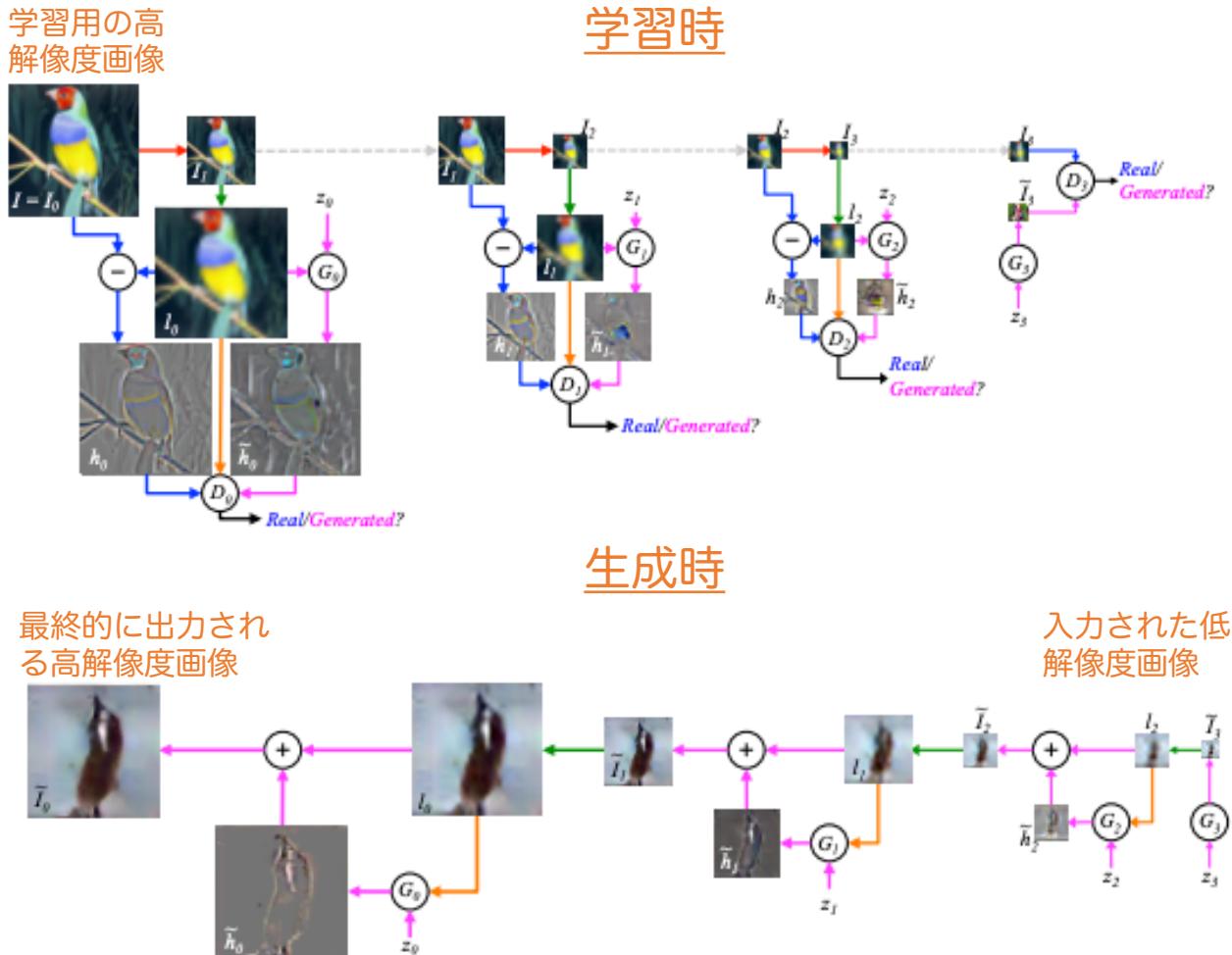


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution  $Z$  is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a  $64 \times 64$  pixel image. Notably, no fully connected or pooling layers are used.

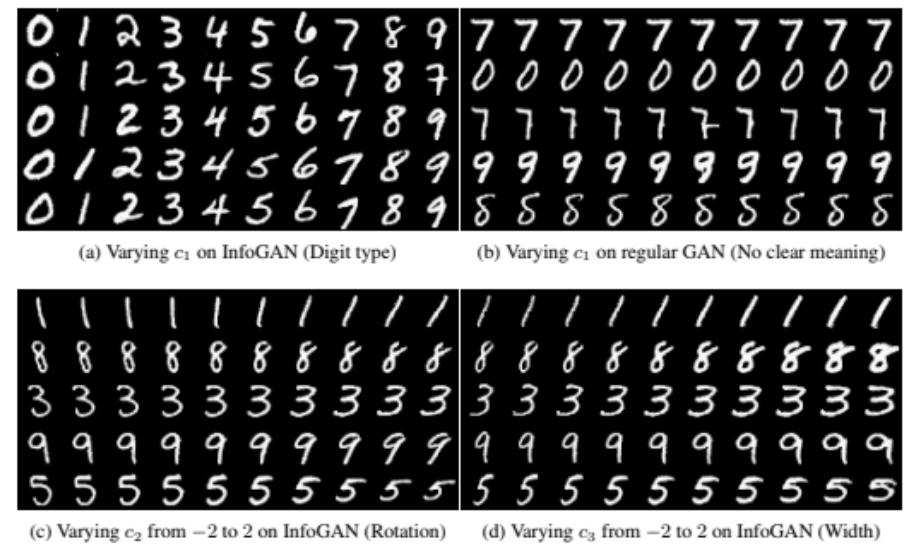
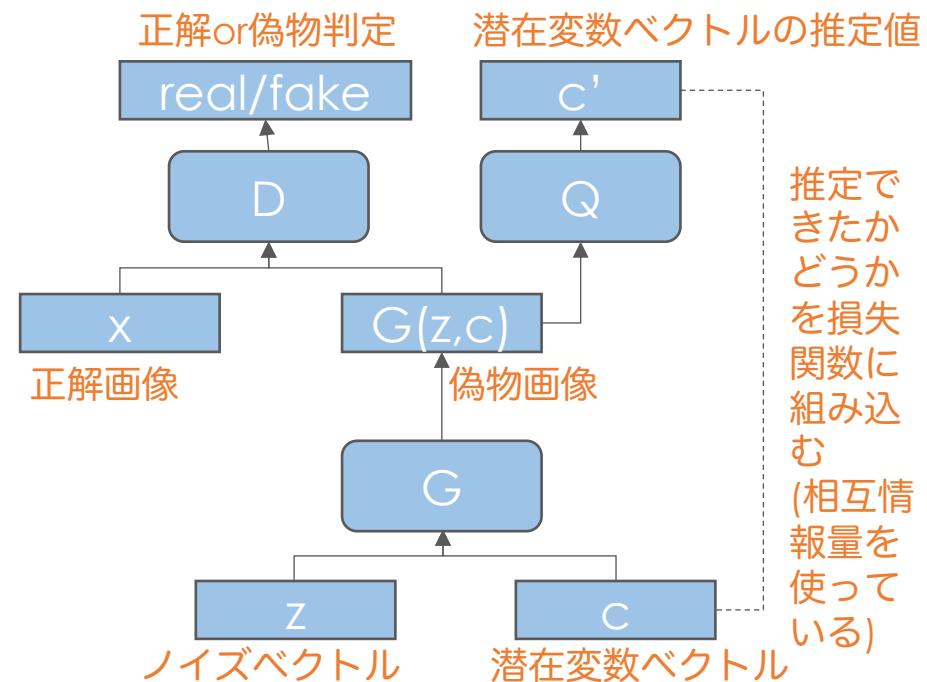
# LAPGAN

- 低解像度の画像から高解像度の画像を生成するモデル。
- 各周波数帯ごとのGANを作り、それらをつなげていくことで解像度を上げていく。
- 原著論文
  - Emily Denton, Soumith Chintala, Arthur Szlam, Rob Fergus. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks.  
<https://arxiv.org/abs/1506.05751>
- Laplacian Pyramidは、古典的な画像処理の方法。以下の論文を参照
  - PETER J. BURT, EDWARD H. ADELSON. The Laplacian Pyramid as a Compact Image Code.[http://persci.mit.edu/pub\\_pdfs/pyramid83.pdf](http://persci.mit.edu/pub_pdfs/pyramid83.pdf)



# infoGAN

- ランダムな入力ベクトルの一部(隠れ符号c)に意味を持たせるGAN。
- DCGANに、隠れ符号cを推測するQを加えている。
- 通常のDCGANでは、ノイズzをどういった値にしたら特定の画像(例えばMNISTであれば手書きの5)を生成できるかは、いろいろ試さないとわからないが、infoGANでは、学習済みの生成モデルにおいて、隠れ符号cを変化させると、回転や幅などが異なる画像を生成することができるようになる。
- 原著論文
  - Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. <https://arxiv.org/pdf/1606.03657.pdf>



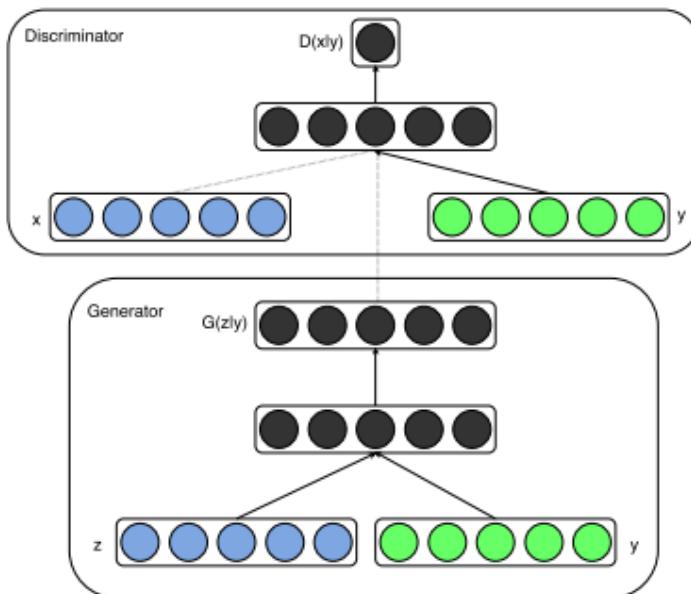
# ConditionalGAN

- 画像とラベルなどの情報がペアになったデータセットを用いて、敵対的学习を行うモデル。
- 原著論文
  - Mehdi Mirza, Simon Osindero.  
Conditional Generative  
Adversarial Nets.  
<https://arxiv.org/pdf/1411.1784.pdf>

*z*: ノイズ  
*y*: ラベルなどの情報

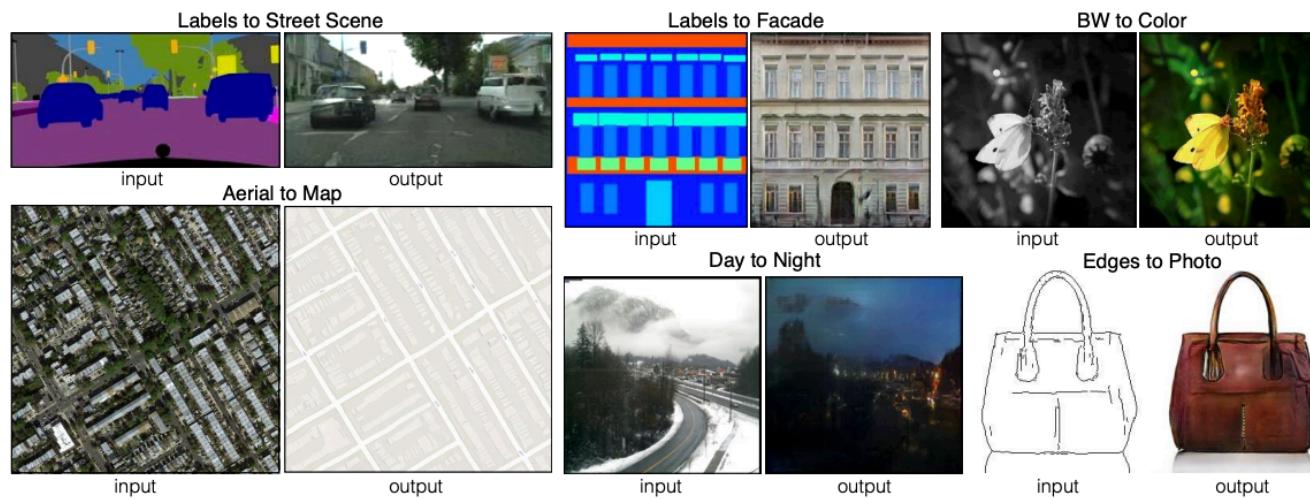


Figure 2: Generated MNIST digits, each row conditioned on one label



# pix2pix

- 画像を変換できるモデル。
- 変換前画像 $x$ と変換後画像 $y$ のペアを用意する必要がある。
- 原著論文
  - Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. <https://arxiv.org/pdf/1611.07004.pdf>



<https://phillipi.github.io/pix2pix/>

$D$ には、変換前画像 $x$ と生成された変換後画像 $G(x)$ を入れる  
 $D$ には、変換前 $x$ と本物の変換後画像 $y$ を入れる

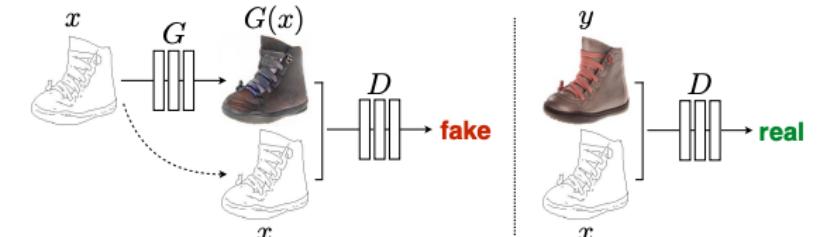


Figure 2: Training a conditional GAN to map edges→photo. The discriminator,  $D$ , learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator,  $G$ , learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

# cycleGAN

- 画像変換を行うためのモデル。
- 画像 $x$ と $y$ がペアになっている必要がない。
- 原著論文

- Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros.  
Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. <https://arxiv.org/pdf/1703.10593.pdf>

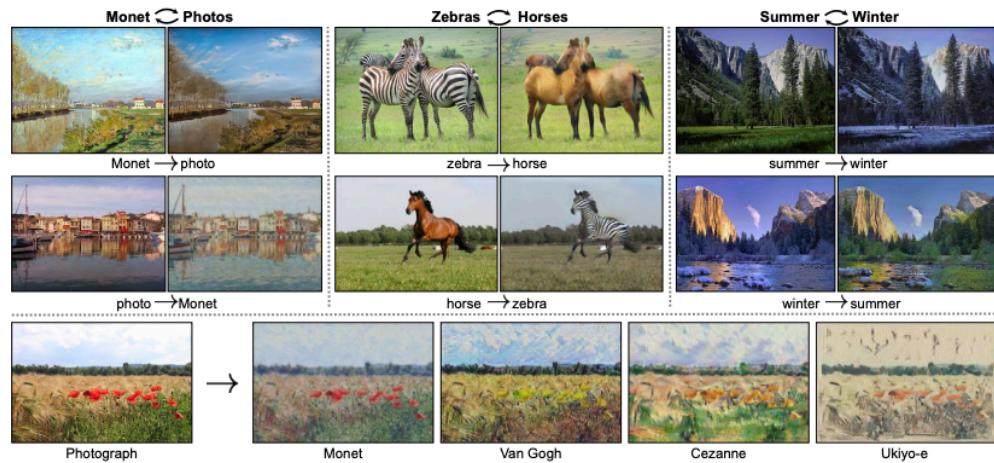


Figure 1: Given any two unordered image collections  $X$  and  $Y$ , our algorithm learns to automatically “translate” an image from one into the other and vice versa: (left) Monet paintings and landscape photos from Flickr; (center) zebras and horses from ImageNet; (right) summer and winter Yosemite photos from Flickr. Example application (bottom): using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

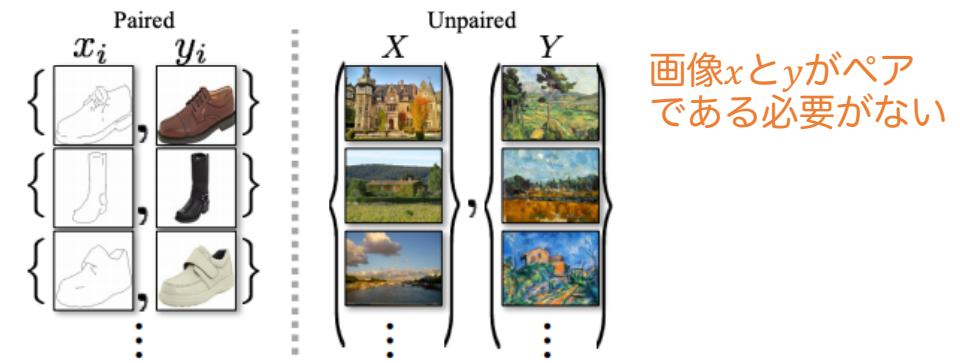


Figure 2: *Paired* training data (left) consists of training examples  $\{x_i, y_i\}_{i=1}^N$ , where the correspondence between  $x_i$  and  $y_i$  exists [22]. We instead consider *unpaired* training data (right), consisting of a source set  $\{x_i\}_{i=1}^N$  ( $x_i \in X$ ) and a target set  $\{y_j\}_{j=1}^N$  ( $y_j \in Y$ ), with no information provided as to which  $x_i$  matches which  $y_j$ .

$G$ と $F$ は画像を変換する生成器。 $D_Y$ と $D_X$ は識別器。生成器は、通常のGANと同じく、識別器を騙せるように学習させる。その際、 $G$ が変換した画像を $F$ が元に戻せたら損失が小さくなるように設計しておく。また、逆も同様。

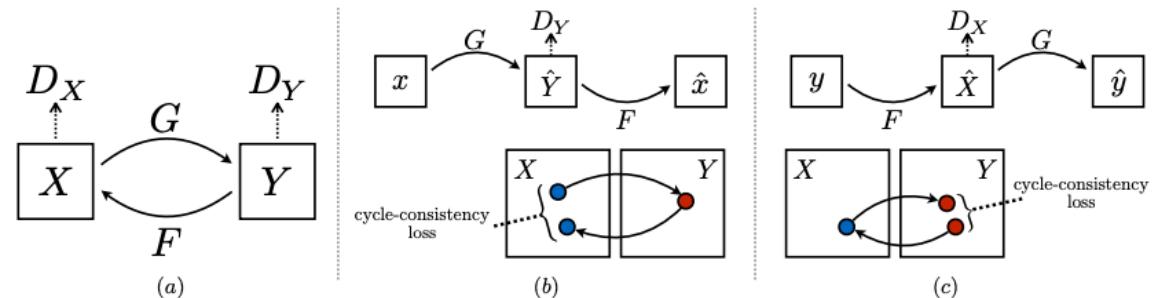


Figure 3: (a) Our model contains two mapping functions  $G : X \rightarrow Y$  and  $F : Y \rightarrow X$ , and associated adversarial discriminators  $D_Y$  and  $D_X$ .  $D_Y$  encourages  $G$  to translate  $X$  into outputs indistinguishable from domain  $Y$ , and vice versa for  $D_X$  and  $F$ . To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss:  $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$ , and (c) backward cycle-consistency loss:  $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

# StackGAN

- 文章の描写に基づいて、高解像度の画像を生成するモデル。
- 原著論文
  - Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. <https://arxiv.org/pdf/1612.03242.pdf>

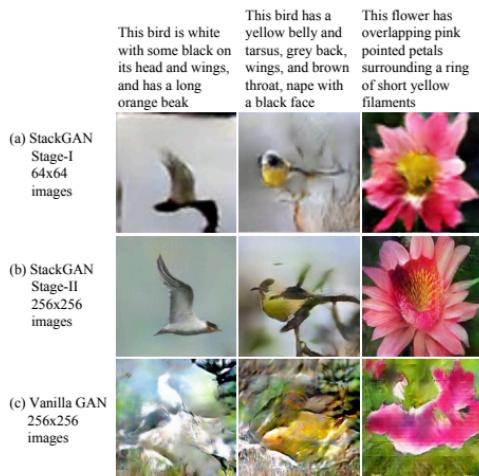


Figure 1. Comparison of the proposed StackGAN and a vanilla one-stage GAN for generating  $256 \times 256$  images. (a) Given text descriptions, Stage-I of StackGAN sketches rough shapes and basic colors of objects, yielding low-resolution images. (b) Stage-II of StackGAN takes Stage-I results and text descriptions as inputs, and generates high-resolution images with photo-realistic details. (c) Results by a vanilla  $256 \times 256$  GAN which simply adds more upsampling layers to state-of-the-art GAN-INT-CLS [26]. It is unable to generate any plausible images of  $256 \times 256$  resolution.

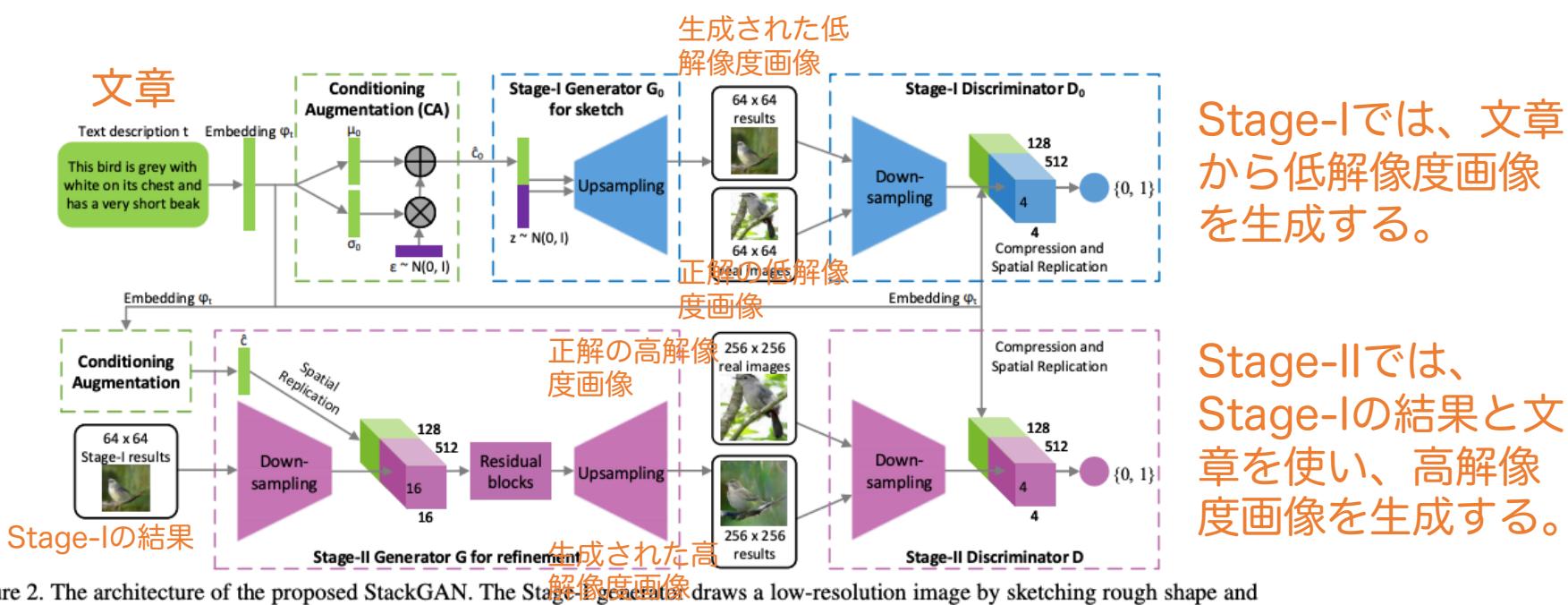


Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects and adds compelling details into Stage-I results, yielding a more realistic high-resolution image.

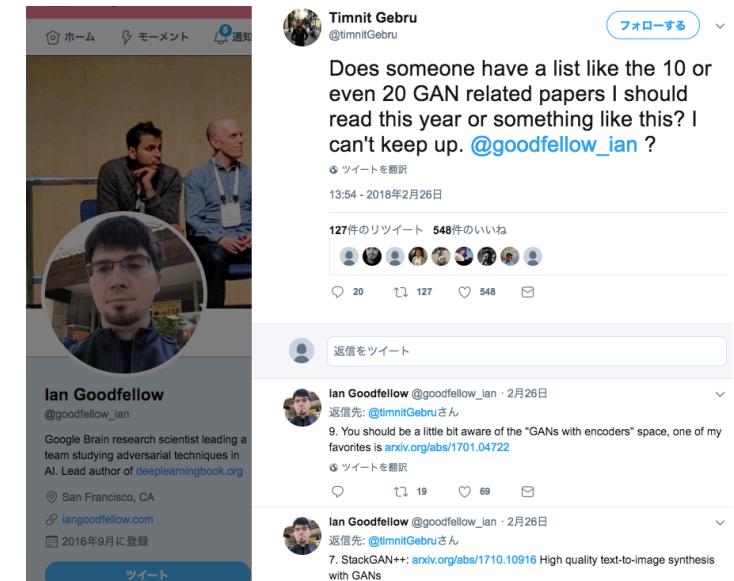
## [演習] GANによる手書き文字の生成

---

- 5\_4\_GAN.ipynb
  - 敵対的生成ネットワークを訓練し、Discriminatorが判別できないような手書き文字をGeneratorに生成させましょう。
  - TensorFlowを用います。

Any Questions?

- GANに関する研究は活発であり、様々なモデルが提案されています。
  - GANまとめサイト
    - <http://urx2.nu/O8c3>
- Ian Goodfellow氏はtwitterにて、GANに関する10の論文を紹介しています。
  - twitterでの投稿
    - <http://urx2.nu/O8bE>



- GANにはどんな派生系があるのか、インターネットを使って調べてみましょう。
- 調べる際、以下の項目に着目してください。
  - モデルの名称
  - モデル名称の由来
  - モデルの特徴
  - 大まかなネットワーク構成
  - 他のGANとの関連性
- 上記を予習段階で各自取り組みましょう。
- 予習段階の取り組み結果をグループで共有します。(45分)
- 最後に、グループごとに発表していただきます。(15分)

## 参考：GANを用いた商用サービス事例

- ・サービス名：クリエイティブAI
- ・事業主：株式会社データグリッド
- ・概要：アイドル画像やキャラクター画像を生成するAI
- ・技術分類：画像処理、生成モデル
- ・URL: <https://datagrid.co.jp/>



## [宿題] 論文調査

---

- ・ 深層学習は、日々新しい手法が開発されている分野です。
- ・ 最新の手法は書籍に掲載されていないことが多く、その場合は原論文を読む必要があります。ほとんどの論文は、 Arxiv.org(<https://arxiv.org>)に登録されています。
- ・ Arxiv.orgのサイトから、通し課題に役立ちそうな論文を探してみましょう。
- ・ 探す時のキーワード
  - ・ CNN
  - ・ Convolutional
  - ・ Recognition
  - ・ Neural network
  - ・ MNIST
  - ・ など

Any Questions?

## 講座の時間が余ったら

---

- ・今回の復習をします。
- ・次の予習をします。

# Appendix

## VAEにおけるKLダイバージェンスの積分計算

---

## 計算過程

---

1次元において、平均 $\mu$ 、分散 $\sigma^2$ 正規分布の確率密度関数を考えると

$$N(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad N(0, 1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

ここで、2つの確率分布 $p, q$ に対するKLダイバージェンスは

$$\begin{aligned} D_{KL}(p||q) &= \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \text{を表す} \\ &= \int_{-\infty}^{\infty} p(x) \log(p(x)) dx - \int_{-\infty}^{\infty} p(x) \log(q(x)) dx \end{aligned}$$

$p(x) = N(\mu, \sigma)$     $q(x) = N(0, 1)$  と置き換えると、

$$D_{KL}(N(\mu, \sigma)||N(0, 1)) = \boxed{\int_{-\infty}^{\infty} N(\mu, \sigma) \log(N(\mu, \sigma)) dx} - \boxed{\int_{-\infty}^{\infty} N(\mu, \sigma) \log(N(0, 1)) dx}$$

**A**      **B**

## A部分の計算

$$A = \int_{-\infty}^{\infty} N(\mu, \sigma) \log(N(\mu, \sigma)) dx = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \cdot \left\{ \left( -\frac{(x-\mu)^2}{2\sigma^2} \right) + \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) \right\} dx$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \cdot \left\{ \left( -\frac{(x-\mu)^2}{2\sigma^2} \right) - \frac{1}{2} \log(2\pi\sigma^2) \right\} dx$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \left( -\frac{1}{2} \right) \int_{-\infty}^{\infty} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \cdot \left\{ \frac{(x-\mu)^2}{\sigma^2} + \log(2\pi\sigma^2) \right\} dx$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \left( -\frac{1}{2} \right) \int_{-\infty}^{\infty} \exp\left(-\frac{t^2}{2\sigma^2}\right) \cdot \left\{ \frac{t^2}{\sigma^2} + \log(2\pi\sigma^2) \right\} dt$$

↓ ガウスの積分公式(右窓)を適用

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \left( -\frac{1}{2} \right) \left\{ \frac{1}{\sigma^2} \frac{2\sigma^2}{2} \sqrt{2\pi\sigma^2} + \log(2\pi\sigma^2) \sqrt{2\pi\sigma^2} \right\}$$

$$= -\frac{1}{2} \{1 + \log(2\pi\sigma^2)\}$$

$x - \mu \rightarrow t$ と置き換え。積分区間は $-\infty \rightarrow +\infty$

$$\frac{dx}{dt} = \frac{d}{dt}(t + \mu) = 1 \text{より}, \quad dx = dt$$

$$\int_{-\infty}^{\infty} \exp\left(-\frac{t^2}{2\sigma^2}\right) dt = \sqrt{\frac{\pi}{1/(2\sigma^2)}} = \sqrt{2\pi\sigma^2}$$

$$\int_{-\infty}^{\infty} \frac{1}{\sigma^2} t^2 \exp\left(-\frac{t^2}{2\sigma^2}\right) dt = \frac{1}{\sigma^2} \frac{(2-1)}{2 * 1/(2\sigma^2)} \sqrt{\frac{\pi}{1/(2\sigma^2)}}$$

ガウスの積分公式

$$\int_{-\infty}^{\infty} e^{-ax^2} dx = \sqrt{\frac{\pi}{a}}$$

$$\int_{-\infty}^{\infty} x^{2n} e^{-ax^2} dx = \frac{(2n-1)!!}{2^n a^n} \sqrt{\frac{\pi}{a}}$$

(2n-1)!! は1から2n-1までの奇数を全てかけ合わせたもの

## B部分の計算

$$B = \int_{-\infty}^{\infty} N(\mu, \sigma) \log(N(0,1)) dx = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \cdot \left\{ \left(-\frac{x^2}{2}\right) + \log\left(\frac{1}{\sqrt{2\pi}}\right) \right\} dx$$

$$= \frac{1}{\sqrt{2\pi}\sigma^2} \int_{-\infty}^{\infty} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \cdot \left\{ \left(-\frac{x^2}{2}\right) - \frac{1}{2} \log(2\pi) \right\} dx$$

$$= \frac{1}{\sqrt{2\pi}\sigma^2} \left(-\frac{1}{2}\right) \left\{ \int_{-\infty}^{\infty} x^2 \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx + \log(2\pi) \int_{-\infty}^{\infty} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx \right\}$$

$$= \frac{1}{\sqrt{2\pi}\sigma^2} \left(-\frac{1}{2}\right) \left\{ \int_{-\infty}^{\infty} (t+\mu)^2 \exp\left(-\frac{t^2}{2\sigma^2}\right) dt + \log(2\pi) \int_{-\infty}^{\infty} \exp\left(-\frac{t^2}{2\sigma^2}\right) dt \right\} \quad \begin{array}{l} x - \mu \rightarrow t \text{と置き換え。積分区間は } -\infty \rightarrow +\infty \\ \frac{dx}{dt} = \frac{d}{dt}(t + \mu) = 1 \text{より、 } dx = dt \end{array}$$

$$= \frac{1}{\sqrt{2\pi}\sigma^2} \left(-\frac{1}{2}\right) \left\{ \int_{-\infty}^{\infty} (t^2 + 2\mu t + \mu^2) \exp\left(-\frac{t^2}{2\sigma^2}\right) dt + \log(2\pi) \int_{-\infty}^{\infty} \exp\left(-\frac{t^2}{2\sigma^2}\right) dt \right\}$$

この項は奇関数になる。積分区間は  $-\infty \rightarrow +\infty$  であり、奇関数を積分すると結果は0になるためこの項は消える

$$= \frac{1}{\sqrt{2\pi}\sigma^2} \left(-\frac{1}{2}\right) \left\{ \frac{2\sigma^2}{2} \sqrt{2\sigma^2\pi} + \mu^2 \sqrt{2\sigma^2\pi} + \sqrt{2\sigma^2\pi} \log(2\pi) \right\}$$

$$= -\frac{1}{2} \{ \sigma^2 + \mu^2 + \log(2\pi) \}$$

ガウスの積分公式を適用

# KLダイバージェンスの計算

$$D_{KL}(N(\mu, \sigma) || N(0, 1)) = \int_{-\infty}^{\infty} N(\mu, \sigma) \log(N(\mu, \sigma)) dx - \int_{-\infty}^{\infty} N(\mu, \sigma) \log(N(0, 1)) dx$$

A

B

$$\begin{aligned} &= -\frac{1}{2} \{1 + \log(2\pi\sigma^2)\} - \left(-\frac{1}{2}\right) \{\sigma^2 + \mu^2 + \log(2\pi)\} \\ &= -\frac{1}{2} \{1 + \log(\sigma^2) - \mu^2 - \sigma^2\} \end{aligned}$$

ここまで1次元空間を想定して $x$ 方向についてのみ積分を計算してきたが、  
J次元ベクトル $z$ を考えるとき、ベクトル $z$ の各成分に対して同様の計算を行うため、  
ベクトル $z$ の $j$ 番目の( $1 \leq j \leq J$ )成分についての平均を $\mu_j$ 、分散を $\sigma_j^2$ として、  
上記の計算結果を足し合わせる。J次元ベクトル $z$ についてのKLダイバージェンスは

$$D_{KL}(N(\mu(z), \sigma(z)) || N(\theta, I)) = -\frac{1}{2} \left\{ J + \sum_{j=1}^J \log(\sigma^2) - \sum_{j=1}^J (\mu_j^2 + \sigma_j^2) \right\}$$