社会人向け講座「データ分析者養成コース」

# 機械学習技術とその数理基盤
(第2部)

鈴木大慈
東京大学大学院情報理工学系研究科数理情報学専攻
理研AIP

2018年4月4日/4月18日

# 機械学習と人工知能の歴史



1946： ENIAC，高い計算能力
フォン・ノイマン「俺の次に頭の良い奴ができた」
1952： A. Samuelによるチェッカーズプログラム

統計的学習

ルールベース

1960年代前半：
ELIZA（イライザ），
擬似心理療法士

1980年代：
エキスパートシステム

人手による学習ルールの作りこみの限界
「膨大な数の例外」

Siriなどにつながる

1957：Perceptron，ニューラルネットワークの先駆け
第一次ニューラルネットワークブーム
1963：線形サポートベクトルマシン

線形モデルの限界

1980年代：多層パーセプトロン，誤差逆伝搬，
畳み込みネット
第二次ニューラルネットワークブーム

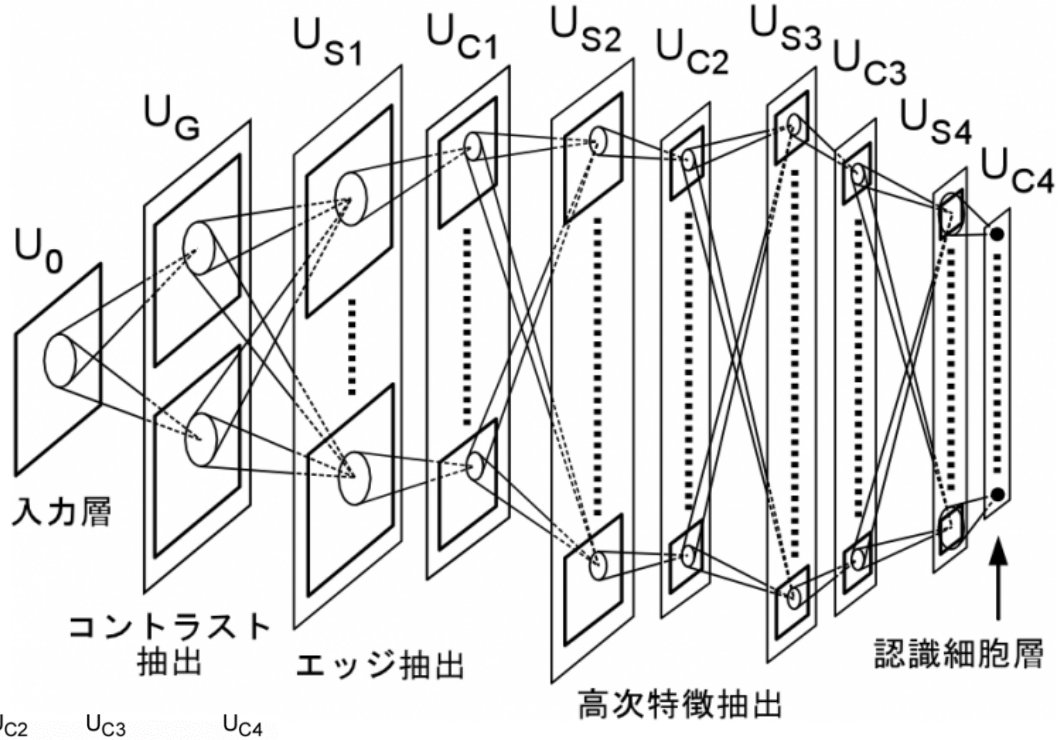
非凸性の問題

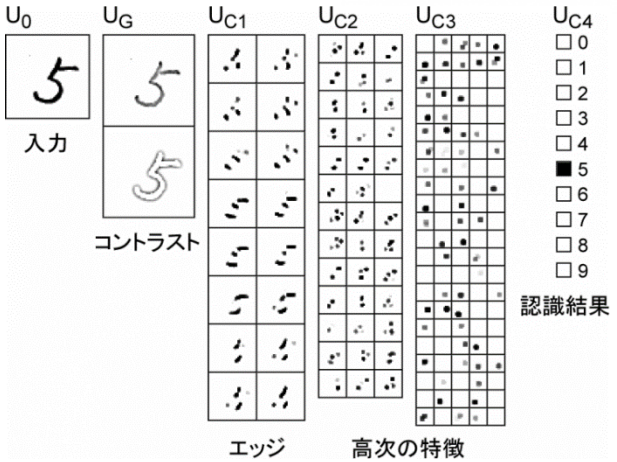
1992： 非線形サポートベクトルマシン
（カーネル法）
1996： スパース学習（Lasso）
2003： トピックモデル（LDA）
2012： Supervision（Alex-net）
第三次ニューラルネットワークブーム

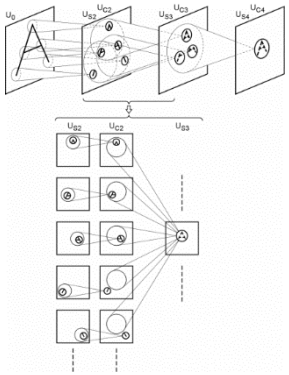
データの増加
＋計算機の強化

# ネオコグニトロン

[福島,79]



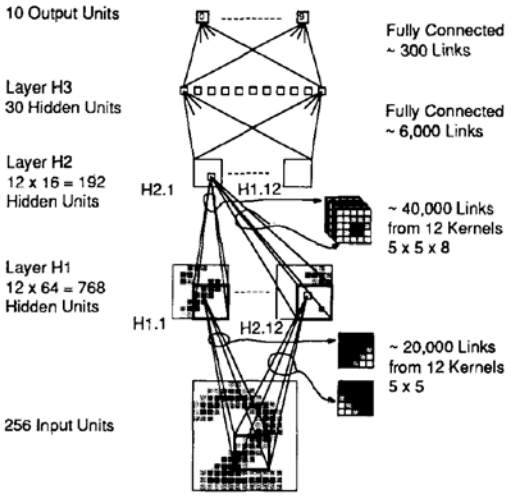



- 人間の脳を模倣
- 畳み込みネットの初期型
- 自己組織型学習
  →素子を足してゆく

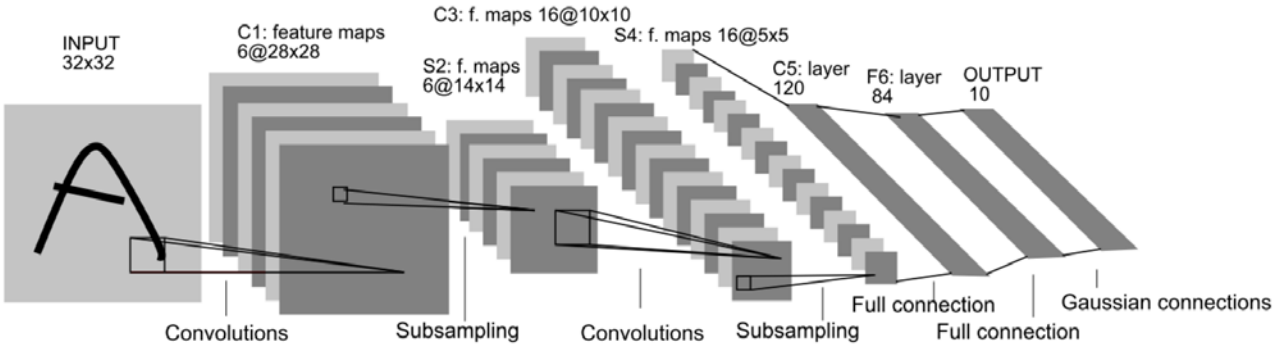
# LeNet



[LeCun+etal,89]




LeNet-5
[LeCun etal,98]




- 畳み込み＋プーリング：現在も使われている構造
- 誤差逆伝搬法でパラメータを更新
- 手書き文字認識データセット（MNIST）で９９％の精度を達成

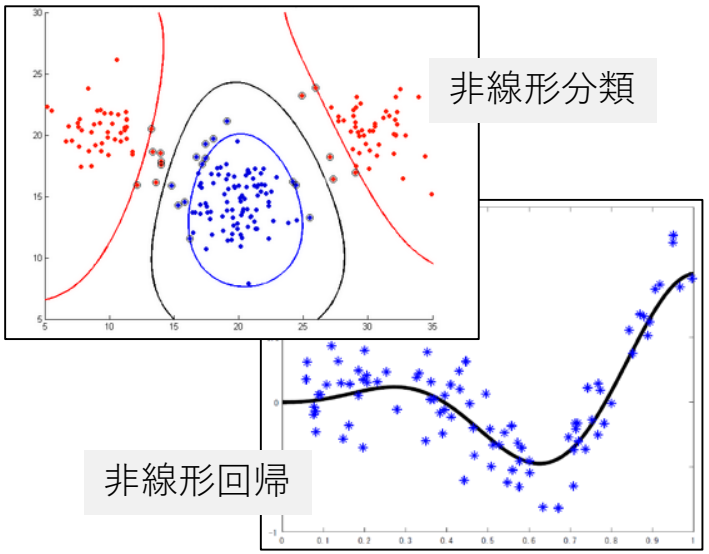
# カーネル法

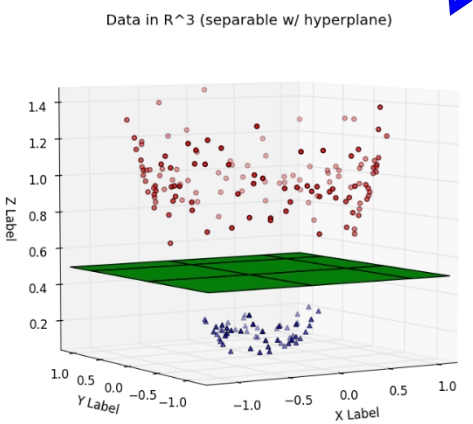
$$\min_{\alpha_i, b} \sum_{i=1}^{n} \max\left\{1 - y_i\left(\sum_{j=1}^{n} k(x_j, x_i)\alpha_j + b\right), 0\right\} + C\sum_{i,j=1}^{n} \alpha_i \alpha_j k(x_i, x_j)$$

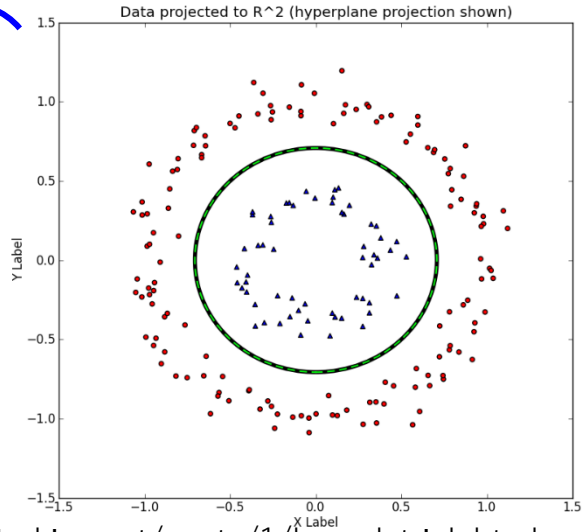
カーネルトリック

$$k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$$

非線形写像 $\phi$




http://wiki.eigenvector.com/index.php?title=Svmda

http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

## 関数解析：再生核ヒルベルト空間の理論

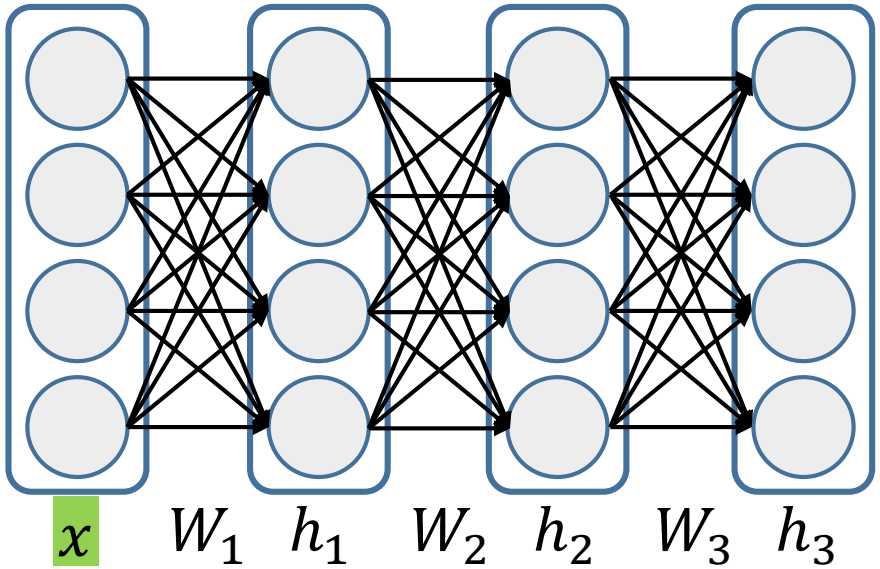
- **凸最適化問題**で解ける.
  - ✓ 効率的な最適化手法が存在.
  - ✓ 解は一つ. 誰が解いても同じ答えが返ってくる.
- **VC理論・経験過程の理論**による汎化誤差の保証.

$$\|\hat{f} - f_0\|_{L_2}^2 \leq O_p(n^{-\frac{1}{1+s}})$$

Vladimir Vapnik

# Deep Learning
# 深層学習

# 深層学習の構造




基本的に「線形変換」と「非線形活性化関数」の繰り返し．

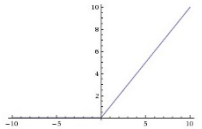
$x$ → $W_1x$ → $h_1(W_1x)$ → $W_2h_1(W_1x)$ → $h_2(W_2h_1(W_1x))$

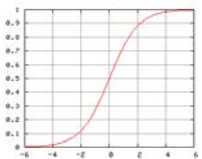
入力　線形変換　非線形変換　線形変換　非線形変換

$$h_1(u) = [h_{11}(u_1), h_{12}(u_2), \ldots, h_{1d}(u_d)]^T$$

活性化関数は通常要素ごとにかかる．Poolingのように要素ごとでない非線形変換もある．

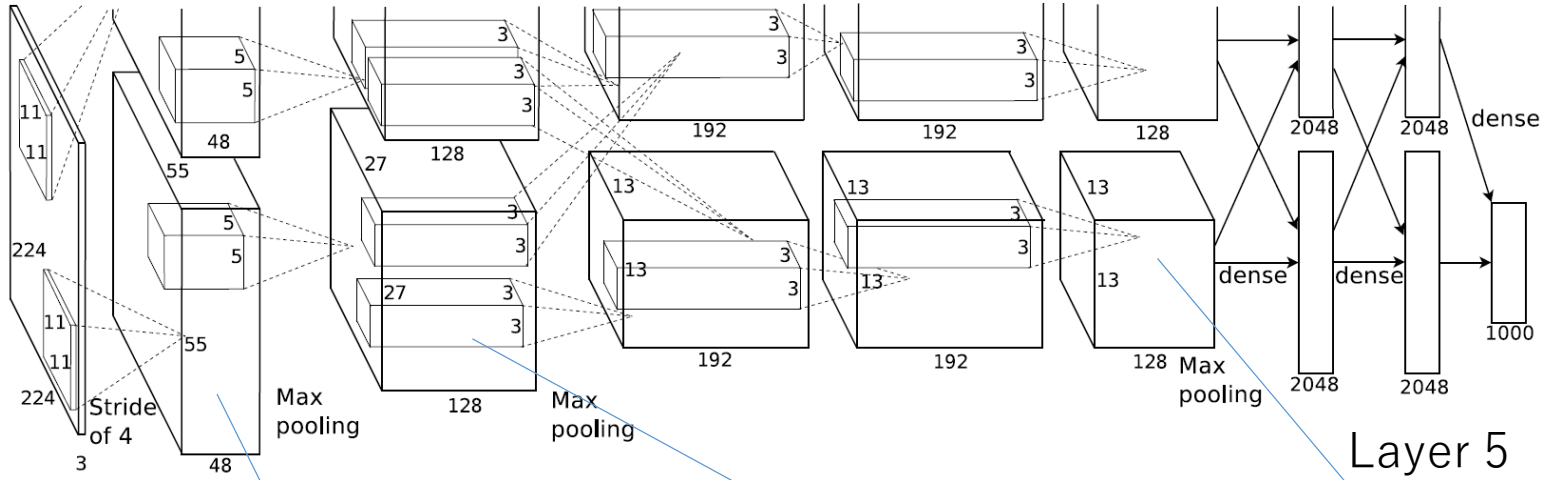
- ☆ReLU (Rectified Linear Unit)：　$h(u) = \max\{u, 0\}$
- シグモイド関数：　$h(u) = \dfrac{1}{1+e^{-u}}$

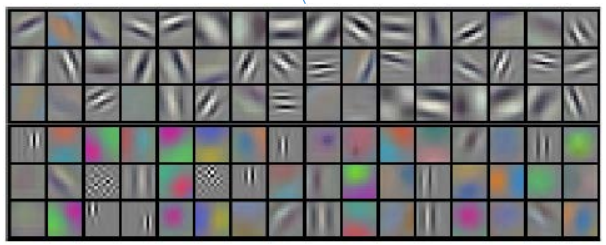


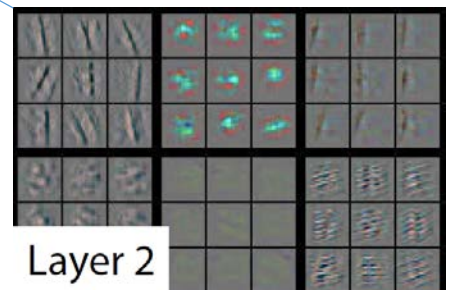


# Alex-net [Krizhevsky, Sutskever + Hinton, 2012]

畳み込みニューラルネットを 5 層積み重ね（+pooling+3層の全結合層）






Layer 5

人の顔

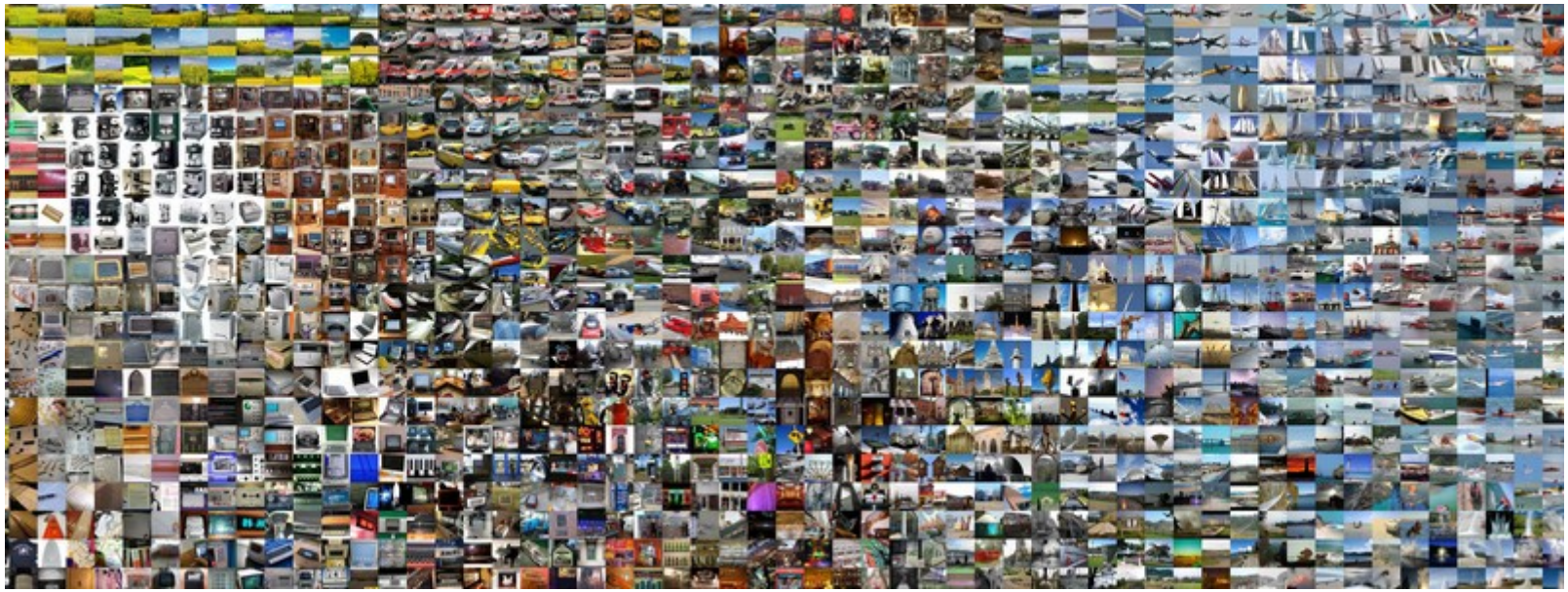
猫の顔

イメージパッチのようなものが学習されている
⇒　特徴量の自動学習

中間層ではより抽象的な情報がコードされる

# ImageNet



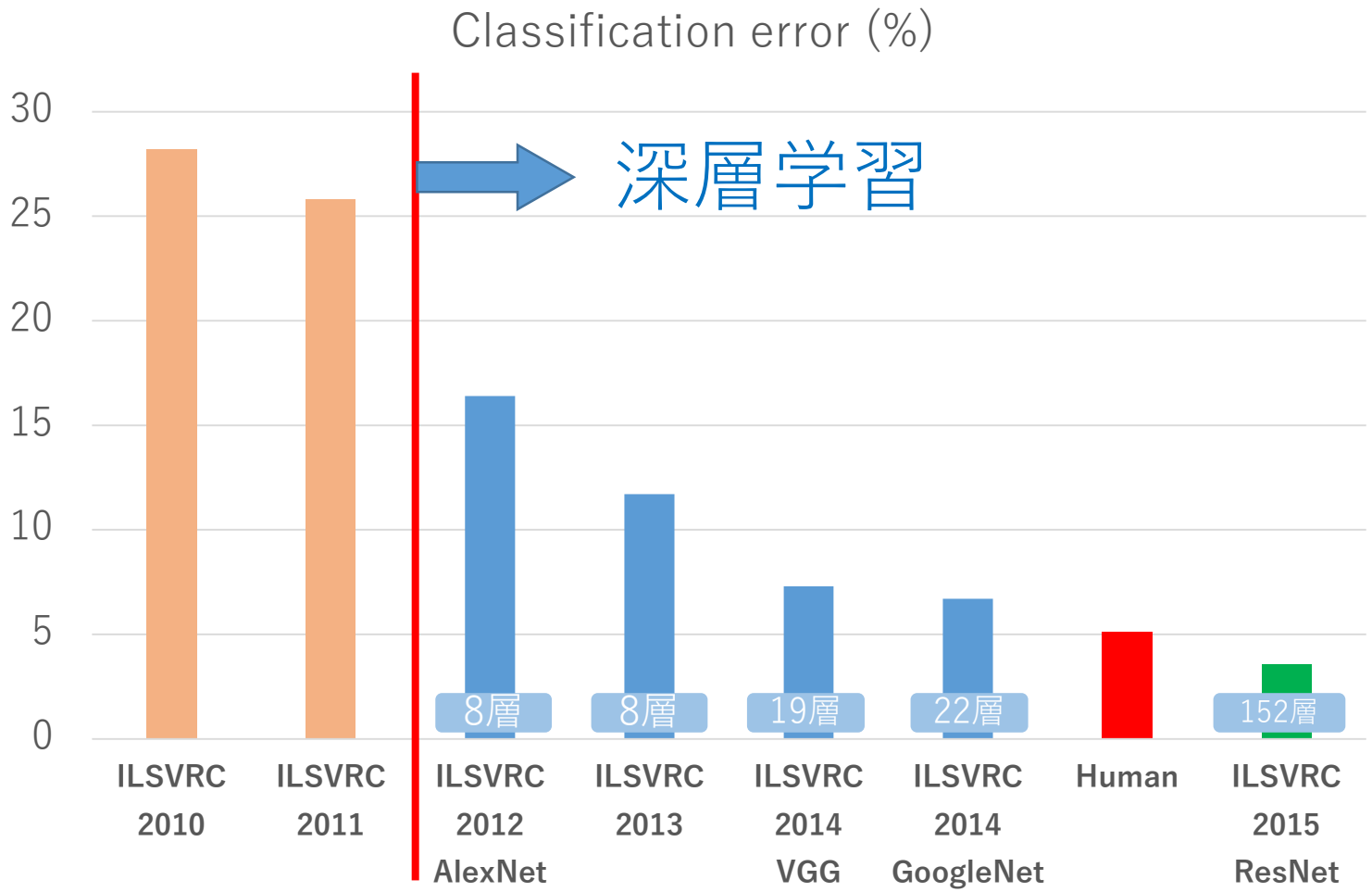


ImageNet: 21,841カテゴリ，14,197,122枚の訓練画像データ

[J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei.
ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009.]

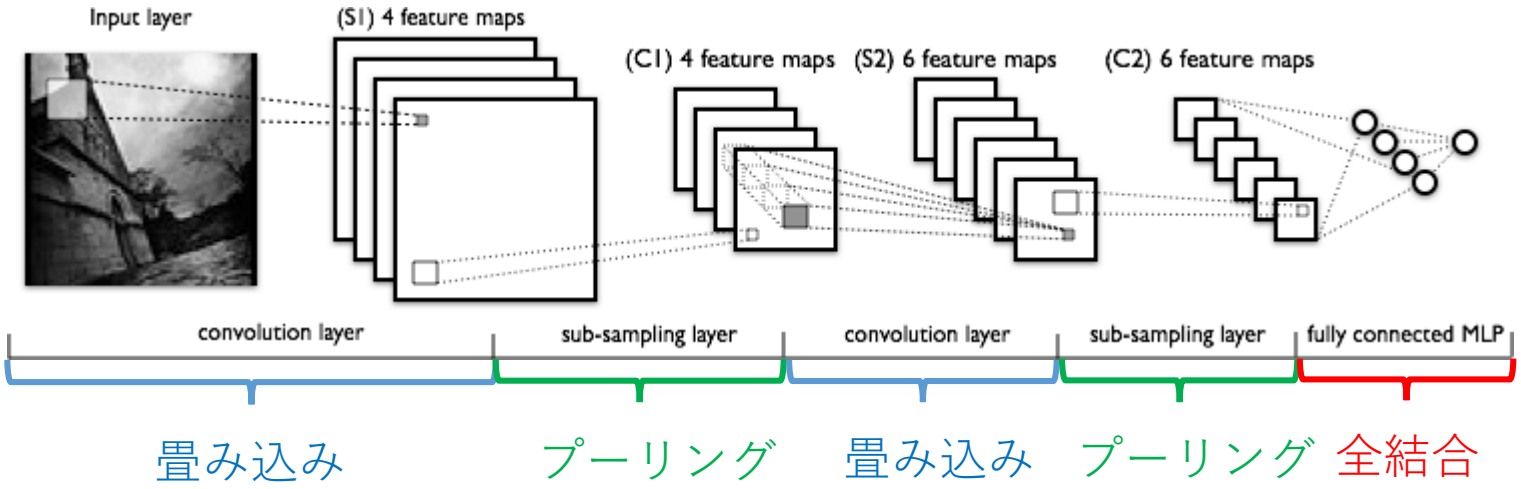
# ImageNetデータにおける識別精度の変遷






ImageNet: 21841クラス，14,197,122枚の訓練画像データ
そのうち1000クラスでコンペティション

# 畳み込みニューラルネット (Convolutional Neural Network, CNN)

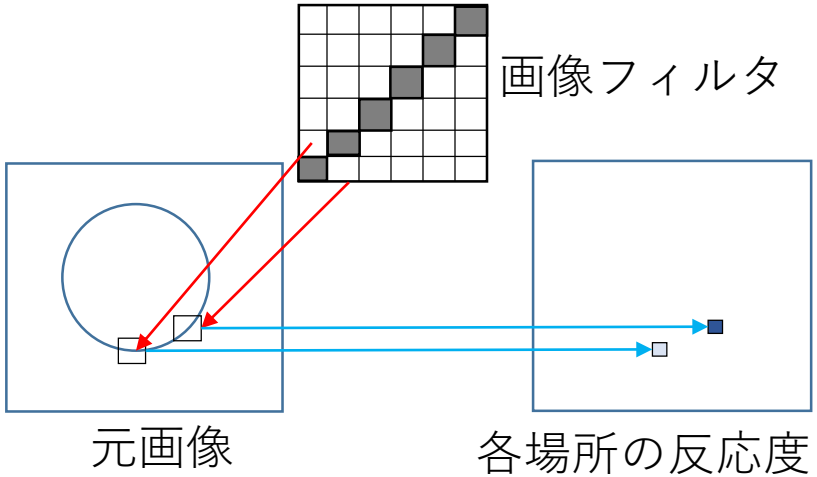


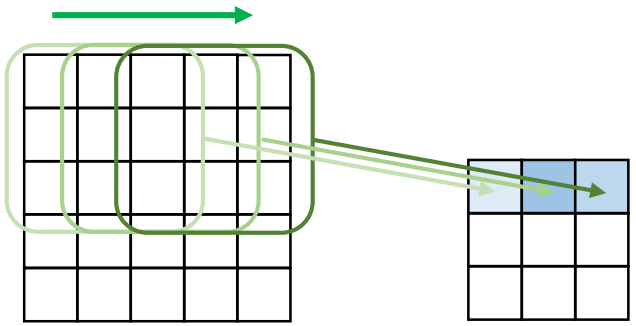

- 畳み込みとプーリングを交互に積み重ねる
- 最後に全結合層を重ねる.
  - 畳み込み (Convolution)：パターンの抽出
  - プーリング (Pooling)：移動不変性を獲得
  - 全結合 (Fully-connected)：最終的な判別器を構成

# 畳み込み層

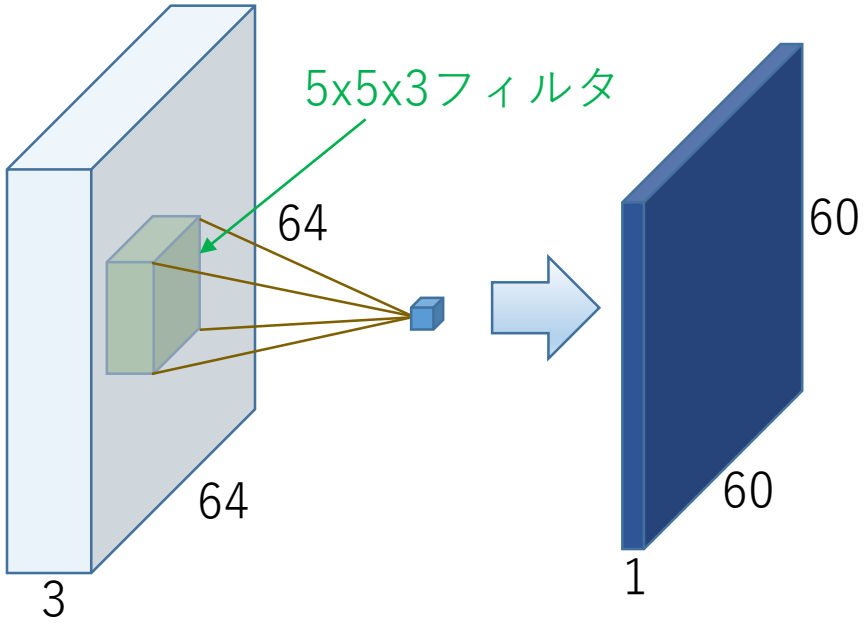


$$X'_{i,j} = \sum_{k,l} X_{i+k,j+l} F_{k,l}$$

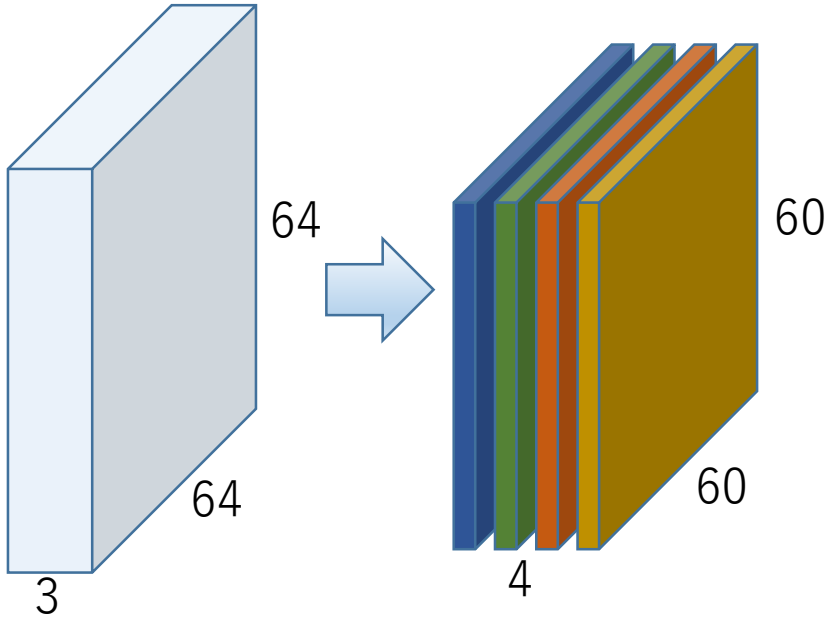


- 画像フィルタをずらしながら畳み込む.
- 複数のフィルタを用意して特徴量を構成.




- RGBカラー画像は「深さ3」のデータ.
- 奥行きも入れたフィルターで畳み込み.

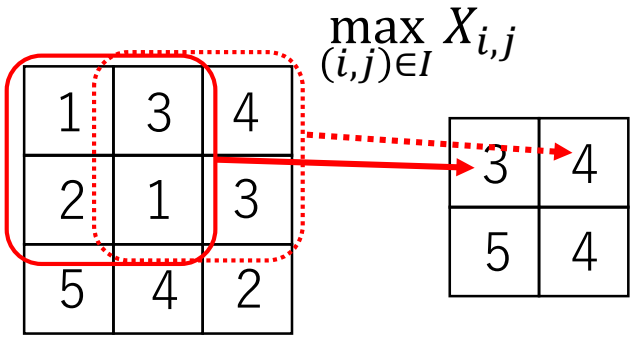



複数フィルタ

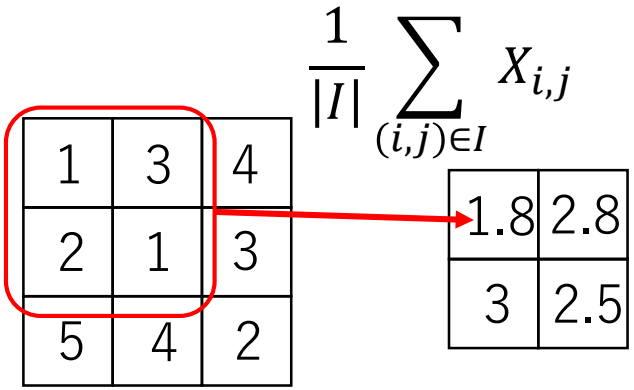
# プーリング層



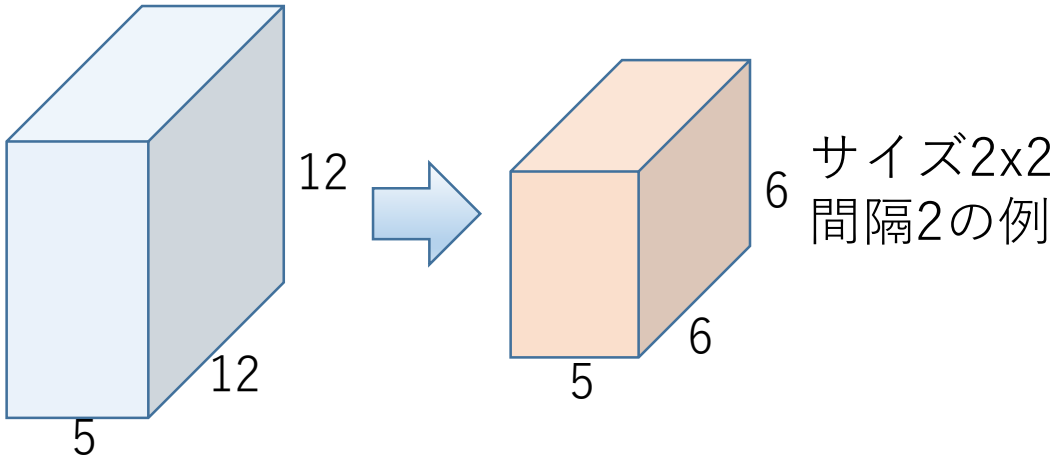
- ある特徴量に選択的に発火している箇所がその領域にあるか検出.
- 少しのずれを吸収する作用→移動不変性




Max-プーリング




Average-プーリング



# 損失関数最小化



経験損失（訓練誤差）

$$L(W) = \sum_{i=1}^{n} \ell(y_i, f(x_i, W))$$

$\ell(y, y') = (y - y')^2$ 二乗損失（回帰）

$\ell(y, y') = -\sum_{k=1}^{K} y_k \log(y'_k)$ Cross-entropy損失（多値判別）

$$\min_W L(W)$$

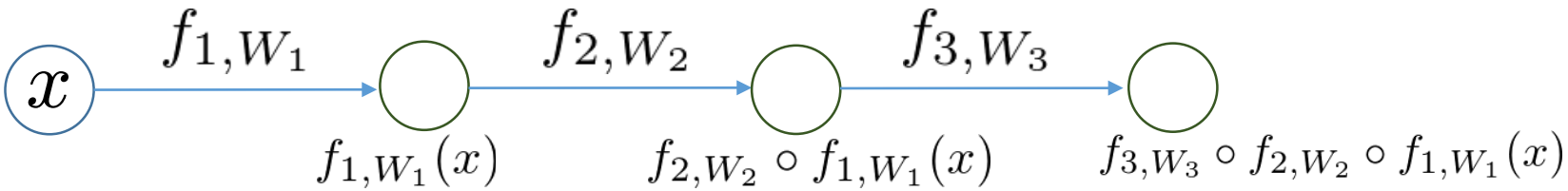
$$W^t = W^{t-1} - \eta \partial_W L(W)$$

- 基本的には確率的勾配降下法 (SGD) で最適化を実行
- AdaGrad, Adam, Natural gradientといった方法で高速化

微分はどうやって求める？ → 誤差逆伝搬法

# 誤差逆伝搬法





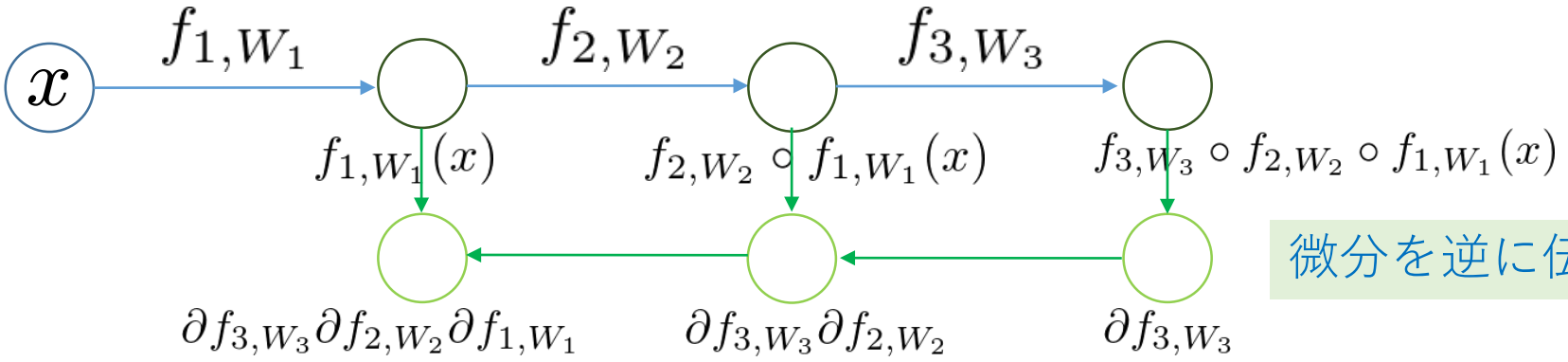

例： $f_{1,W_1}(x) = h(W_1 x)$

合成関数

$$f(x;W) = f_{3,W_3}(f_{2,W_2}(f_{1,W_1}(x)))$$
$$= f_{3,W_3} \circ f_{2,W_2} \circ f_{1,W_1}(x)$$

合成関数の微分

$$\frac{\partial f}{\partial W_1}(x) = \frac{\partial f_{3,W_3}}{\partial f_{2,W_2}} \frac{\partial f_{2,W_2}}{\partial f_{1,W_1}} \frac{\partial f_{1,W_1}}{\partial W_1}(x)$$




連鎖律を用いて微分を伝搬

$$\frac{\partial f}{\partial W_3}(x) = \frac{\partial f_{3,W_3}}{\partial W_3}\left(f_{2,W_2} \circ f_{3,W_3}(x)\right)$$

$$\frac{\partial f}{\partial W_2}(x) = \frac{\partial f_{3,W_3}}{\partial f_{2,W_2}} \frac{\partial f_{2,W_2}}{\partial W_2}(f_{3,W_3}(x))$$

$$\frac{\partial f}{\partial W_1}(x) = \frac{\partial f_{3,W_3}}{\partial f_{2,W_2}} \frac{\partial f_{2,W_2}}{\partial f_{1,W_1}} \frac{\partial f_{1,W_1}}{\partial W_1}(x)$$

パラメータによる微分と入力による微分は違うが，情報をシェアできる．

$f_{1,W}(x) = h(Wx)$ の場合　　$u = Wx$

$$\frac{\partial f_{1,W}}{\partial W_{ij}}(x) = \frac{\partial h}{\partial u_i}(u) x_j \qquad \frac{\partial f_{1,W_1}}{\partial x_j}(x) = \sum_i \frac{\partial h}{\partial u_i}(u) W_{ij}$$

# 確率的勾配降下法 (SGD)

(Stochastic Gradient Descent)

沢山データがあるときに強力

$$\min_W \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{重い}} \ell(z_i, W)$$

大きな問題を分割して個別に処理

**普通の勾配降下法：**

$$W^t = W^{t-1} - \alpha \nabla L(W)$$

$$= W^{t-1} - \alpha \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{全データの計算}} \nabla \ell(z_i, W)$$

# 確率的勾配降下法 (SGD)

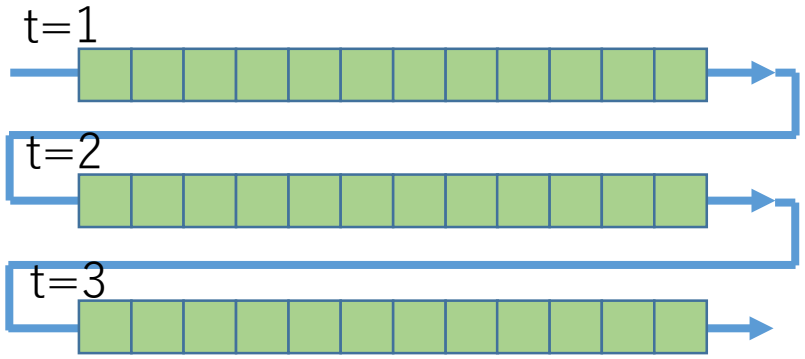
(Stochastic Gradient Descent)

沢山データがあるときに強力

$$\min_W \underbrace{\frac{1}{n}\sum_{i=1}^{n}\ell(z_i, W)}_{\text{重い}}$$

大きな問題を分割して個別に処理

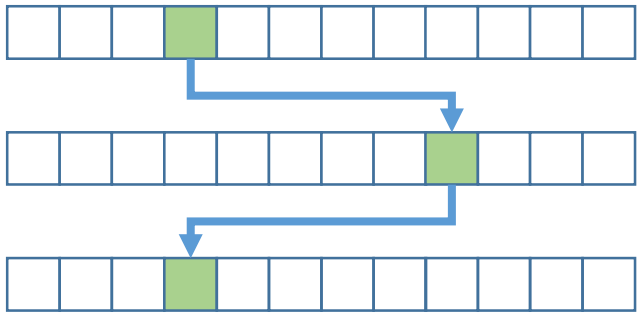
**普通の勾配降下法：**




$$W^t = W^{t-1} - \alpha\frac{1}{n}\sum_{i=1}^{n}\nabla\ell(z_i, W)$$

**確率的勾配降下法：**

毎回の更新でデータを一つ(または少量)しか見ない



$$W^t = W^{t-1} - \alpha\nabla\ell(z_i, W)$$

# 確率的勾配降下法 (SGD)

(Stochastic Gradient Descent)

沢山データがあるときに強力

$$\min_W \frac{1}{n} \underbrace{\sum_{i=1}^{n} \ell(z_i, W)}_{\text{重い}}$$

大きな問題を分割して個別に処理

- ランダムに一つのデータ $z_{i_t}$ を観測.
- 選択した一つのデータで勾配を計算：

$$g_t = \nabla_W \ell(z_i, W^{(t-1)})$$

← O(1)の計算量

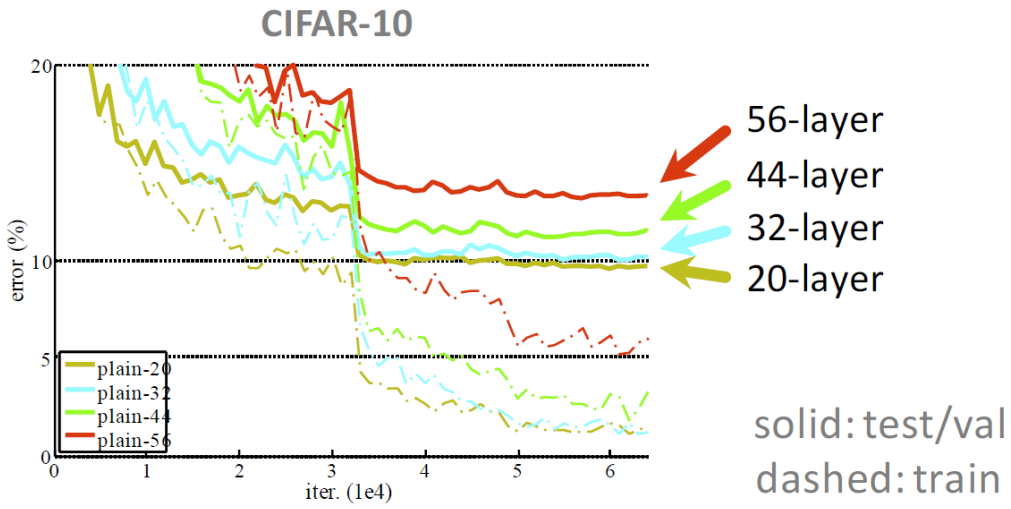
- 勾配方向へ更新（近接勾配法と同じ更新式）：

$$W^{(t)} = W^{(t-1)} - \alpha g_t$$

理論的にも実験的にもトータルの計算量で得することが知られている.

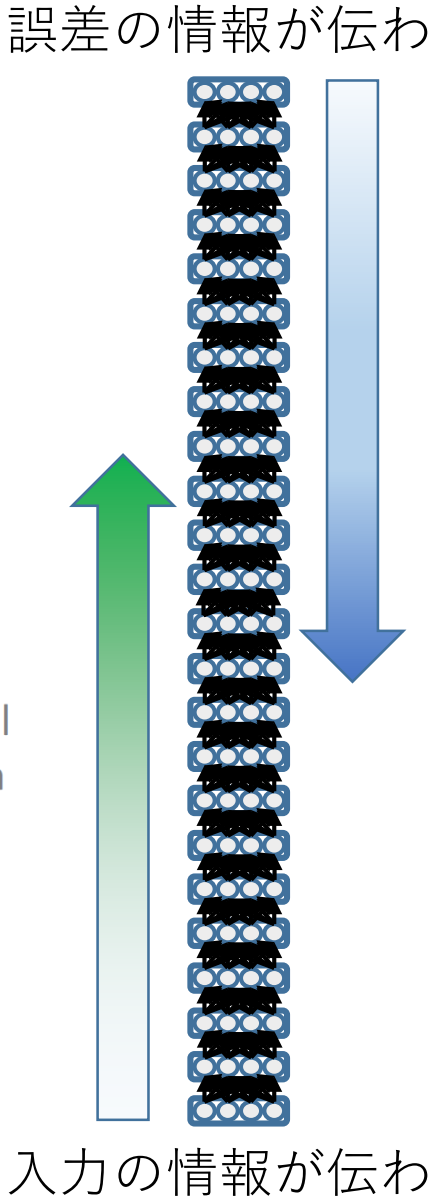
# 深ければ良い？

A：必ずしもそうではない.

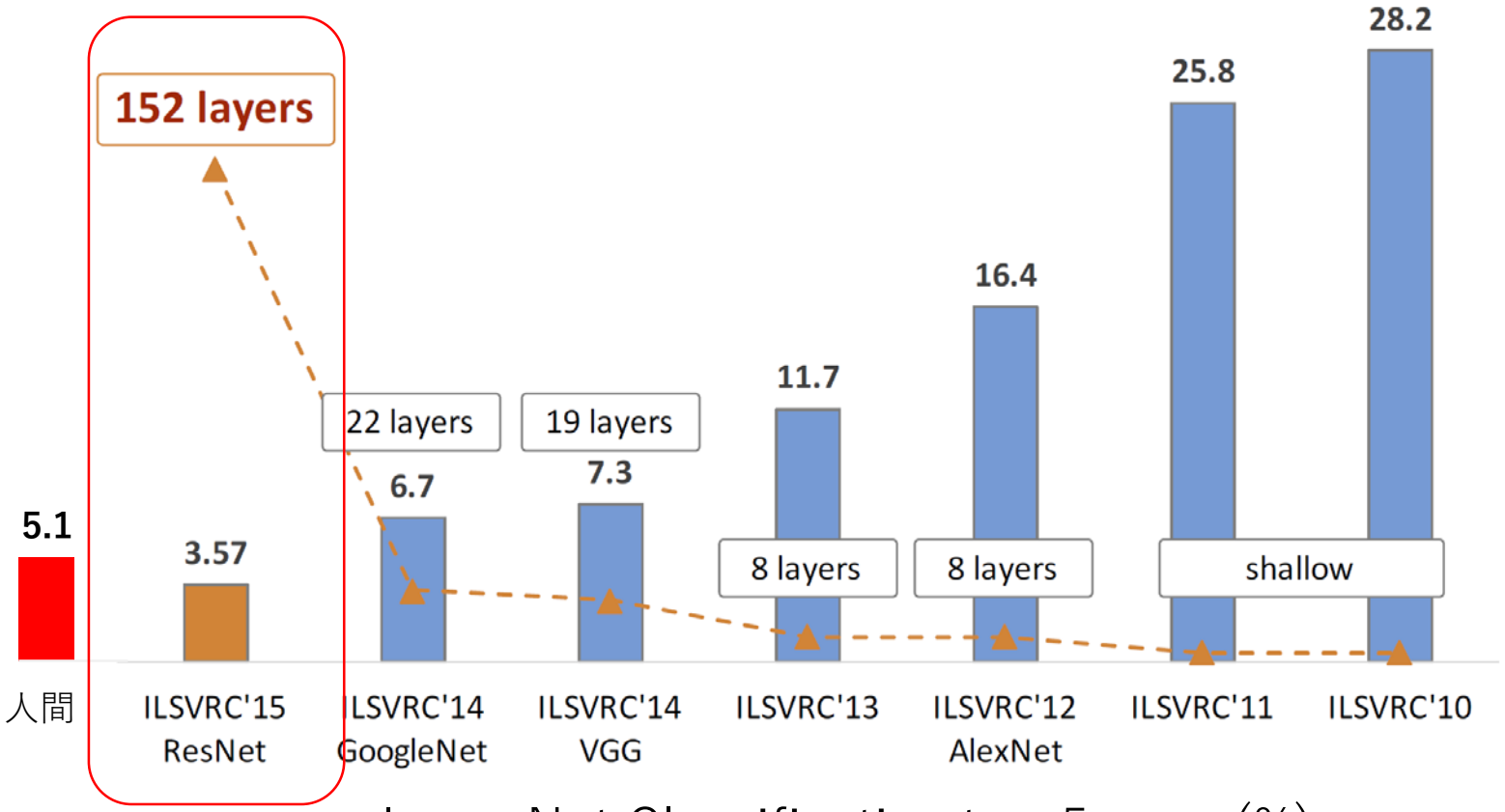



He, Zhang, Ren, & Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

He. “Deep Residual Network”. ICML2016 tutorial.



# ResNet (Deep Residual Net)




ImageNet Classification top-5 error (%)

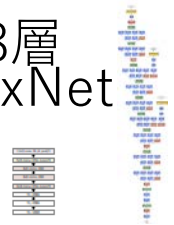
He, Zhang, Ren, & Sun. "Deep Residual Learning for Image Recognition". CVPR 2016. (CVPR2016 best paper award)

He. "Deep Residual Network". ICML2016 tutorial.

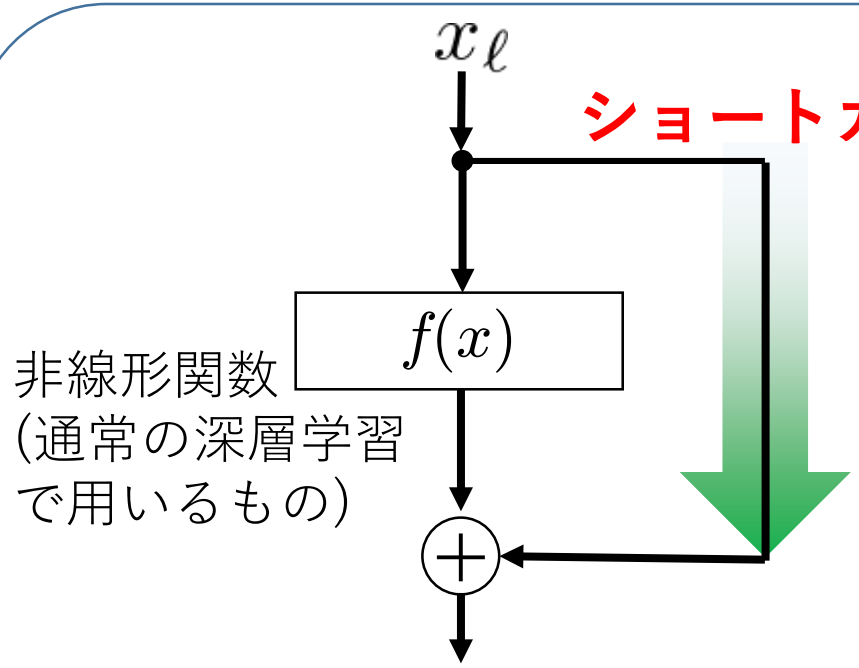
ResNet
152層

22層
GoogleNet

8層
AlexNet



# ResNetの構造



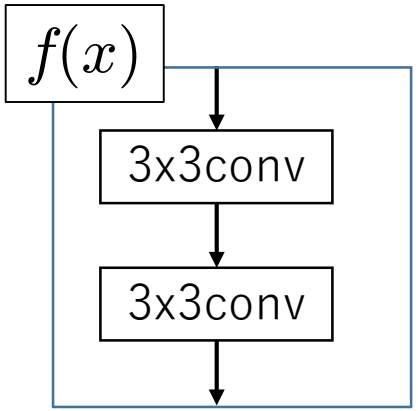

$$x_{\ell+1} = x_\ell + f(x_\ell)$$

$$x_{\ell+2} = x_\ell + f(x_\ell) + f(x_{\ell+1})$$

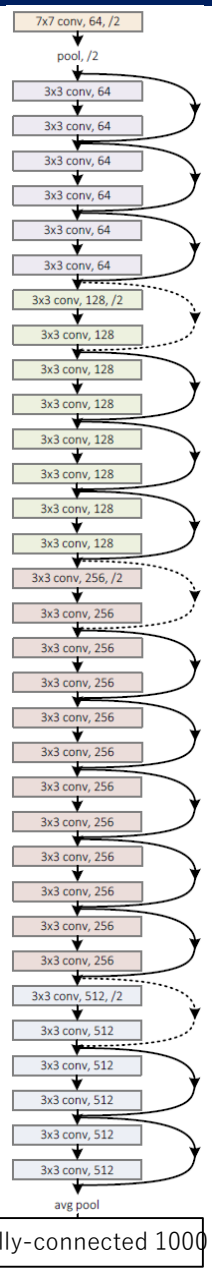
$$\vdots$$

$$x_L = x_\ell + \sum_{k=\ell}^{L-1} f(x_k)$$

情報が減衰せずに伝わる




CIFAR-10などの画像認識タスクでは$f(x)$として2層の畳み込み層を用いたものが良かった.




1000層を超えるものもある

# ResNetの変種

- **Stochastic Depth**

  [Huang,Sun,Liu,Sedra,Weinberger: Deep Networks with Stochastic Depth, 2016]

  学習中に接続を確率的に切る.

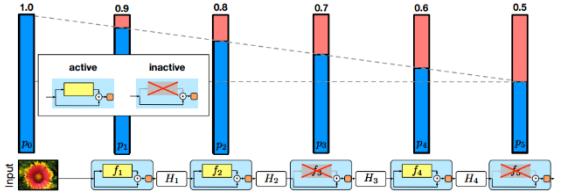
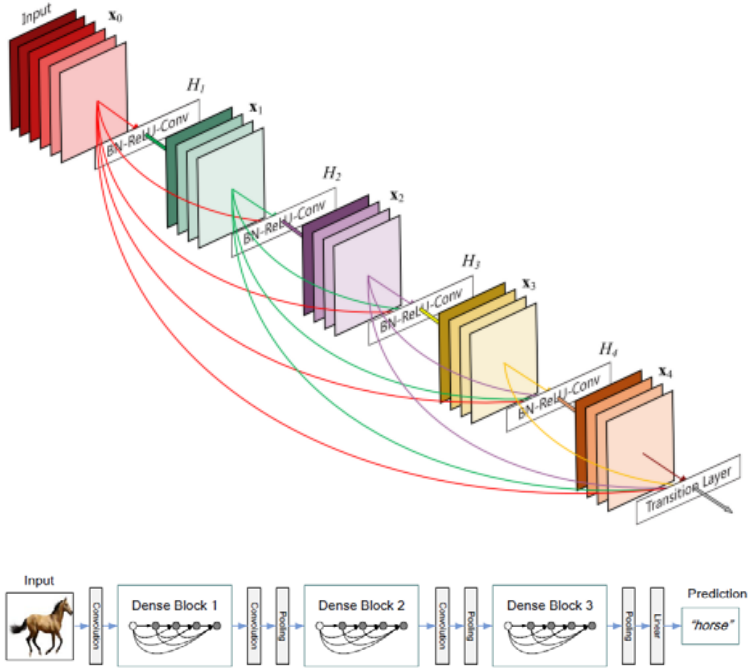



**Fig. 2.** The linear decay of $p_\ell$ illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as $H_0$, which is always active.

- **DenseNet** [Huang, Liu, Weinberger, van der Maaten: Densely Connected Convolutional Networks, 2016]
  (CVPR2017 best paper award)

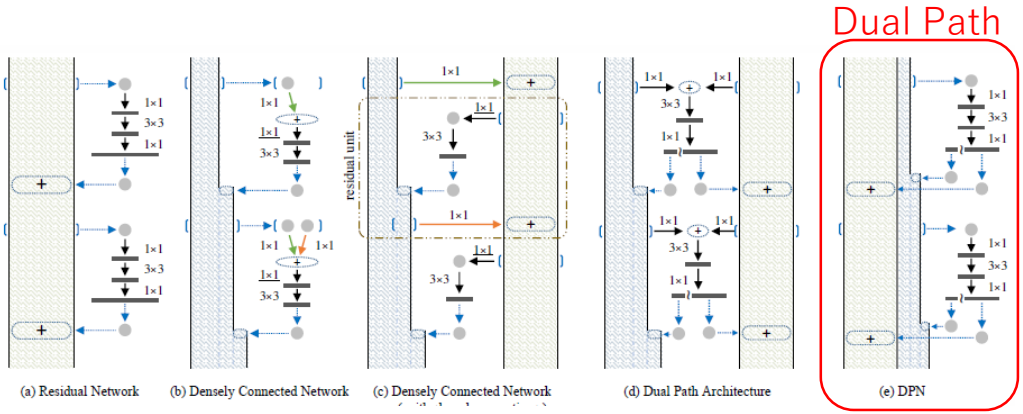
長いスキップを用いて密な結合を用いる




DenseNetの様子

- **Dual Path Networks**

[Chen, Li, Xiao, Jin, Yan, Feng: Dual Path Networks, 2017]




ResNetとDenseNetの良い部分を組み合わせ.
ILSVRC2017のObject localization部門で１位.

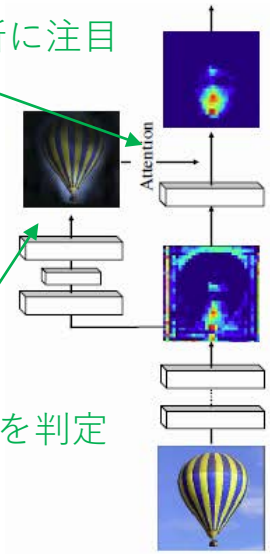
# Residual Attention Network



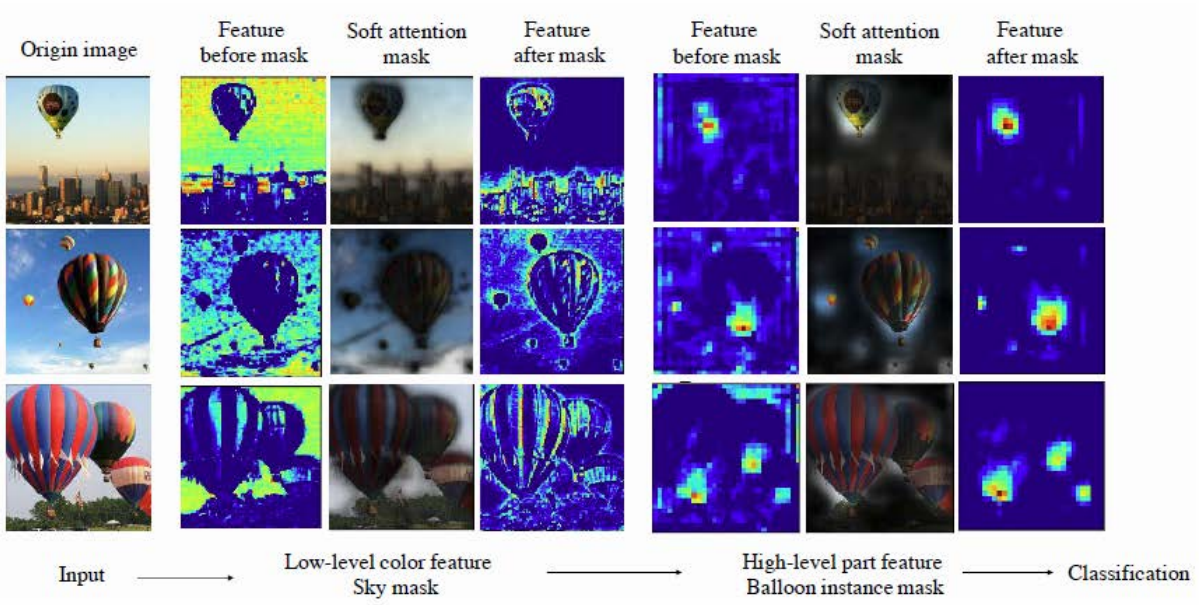
ILSVRC2017のObject detection部門１位

重要そうな場所に注目

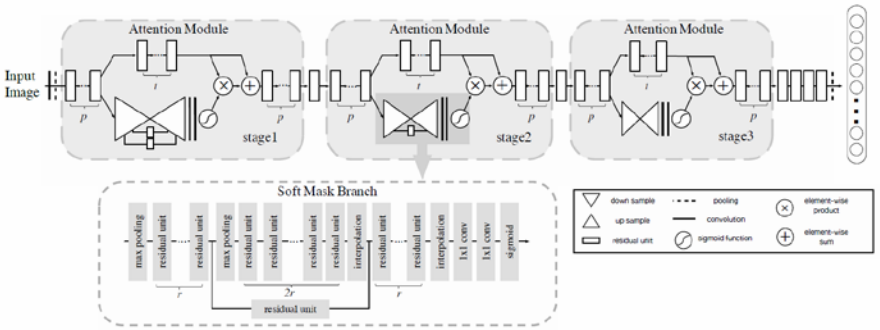
重要そうな場所を判定

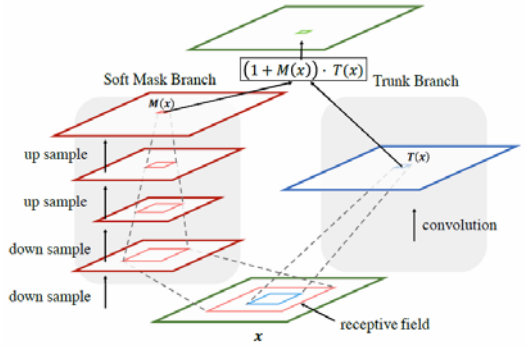



Attention mechanism




ResNetに選択的注意の機構を付与

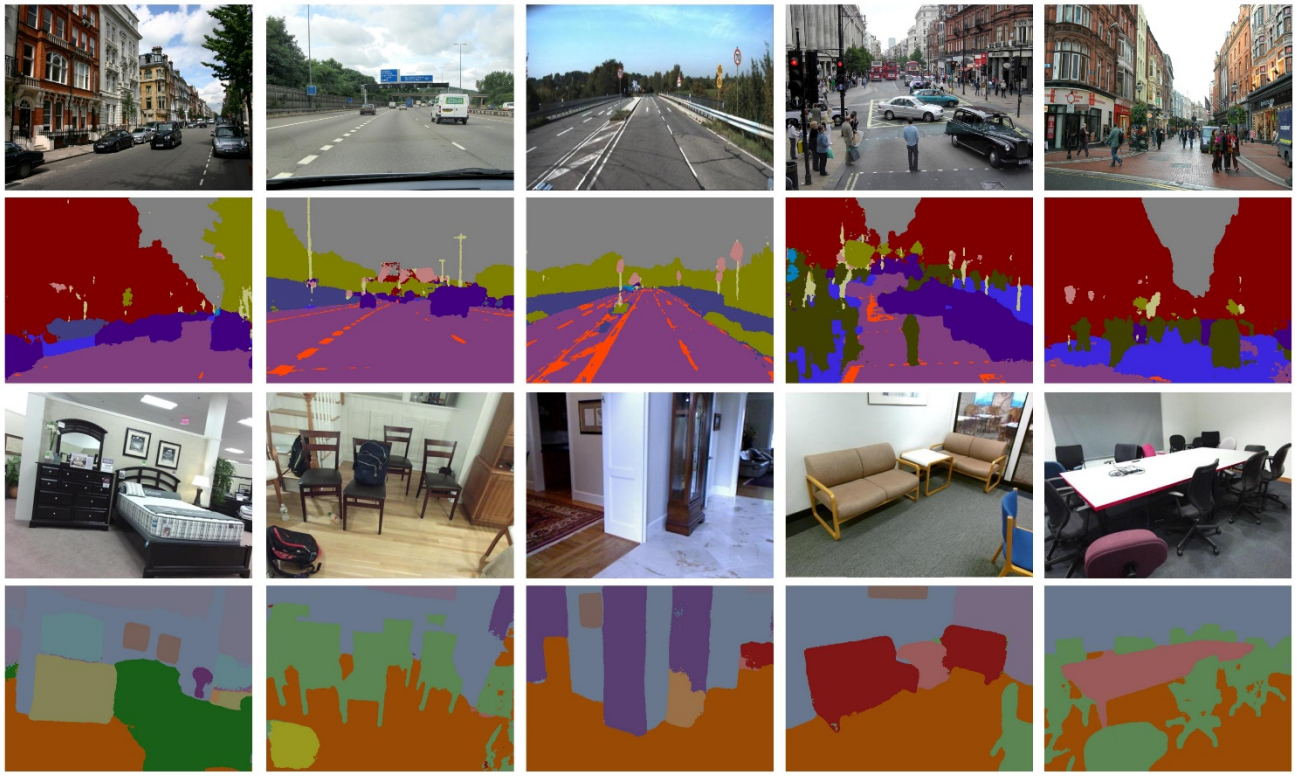



[Wang F, Jiang M, Qian C, et al. Residual Attention Network for Image Classification. arXiv:1704.06904, 2017]

# SegNet

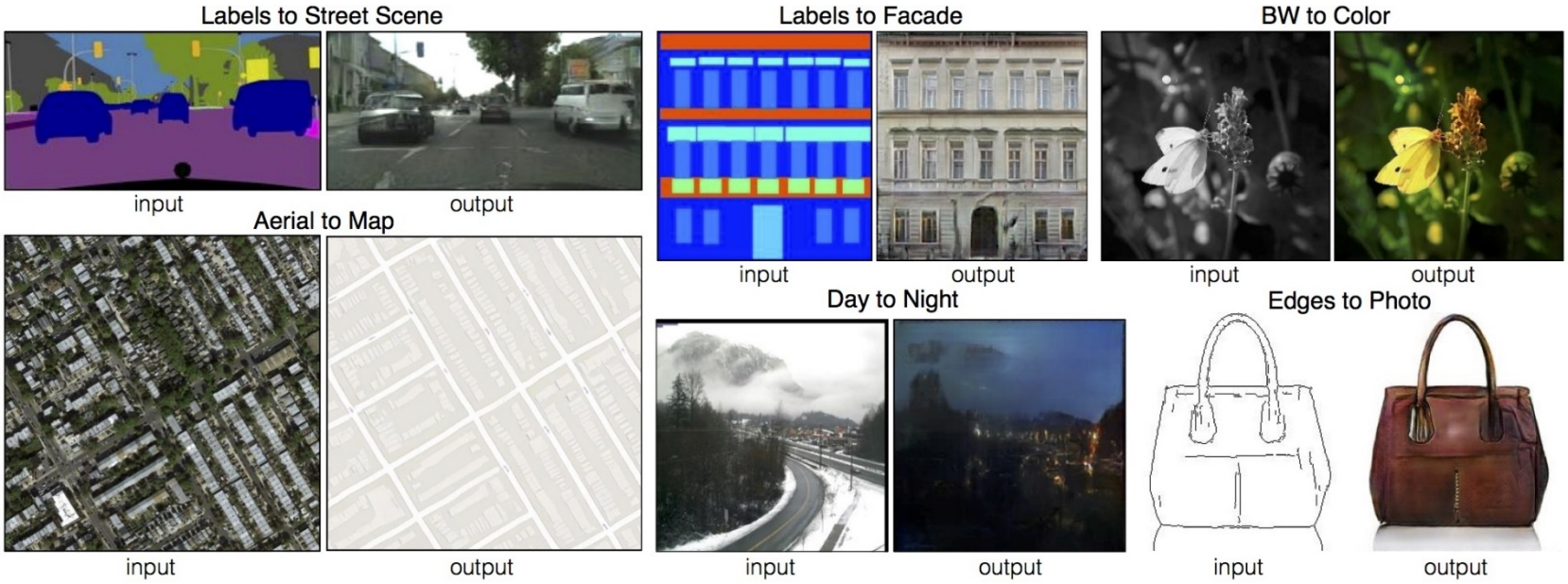





Badrinarayanan, Kendall, Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. 2015.

# Pix2Pix





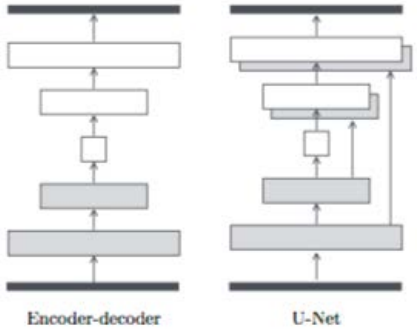
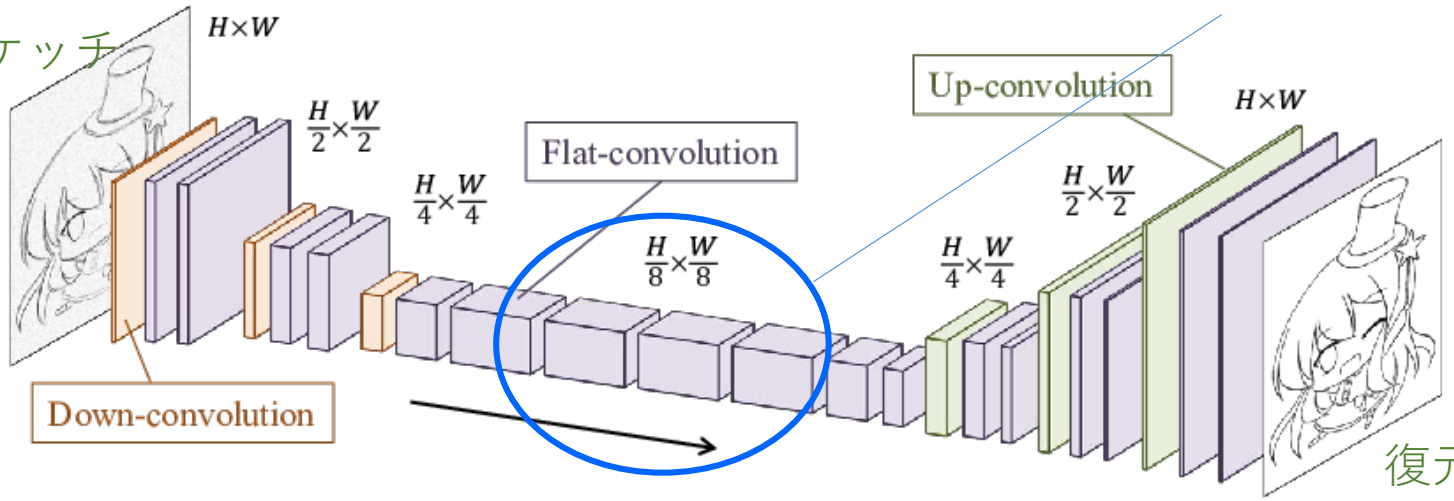




Figure 3: Two choices for the architecture of the generator. The "U-Net" [34] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

Chainerによる自動彩色

Isola, Zhu, Zhou, and Efros :Image-to-Image Translation with Conditional Adversarial Networks. 2016

# ラフスケッチの自動線画化





[ Simo-Serra et al., SIGGRAPH2016]

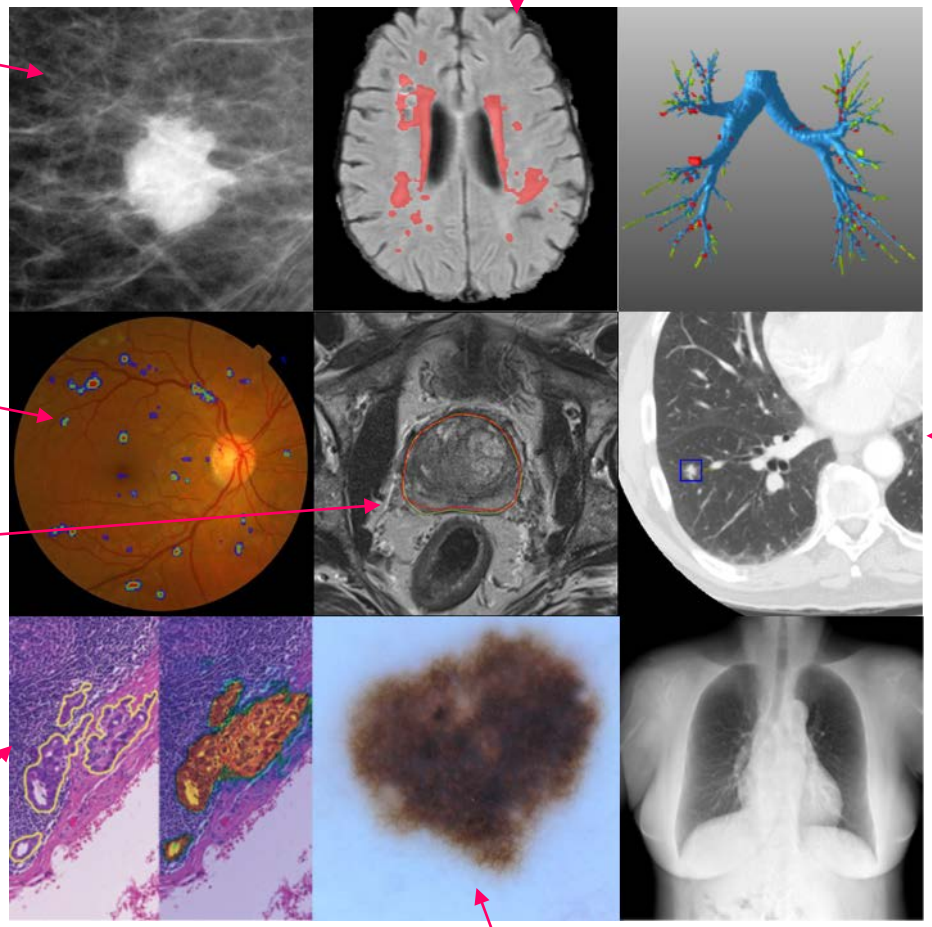
# 画像診断への応用

マンモグラム分類
[Kooi et al., 2016]

脳損傷セグメンテーション
[Ghafoorian et al., 2016]

気管のセグメンテーションと損傷の検出
[Charbonnier et al., 2017]

糖尿病網膜症分類
[Kaggle, and van Grinsven et al. 2016]



小結節分類
[LUNA16 challenge]

前立腺セグメンテーション
[PROMISE12 challenge]

肺癌転移検出
[CAMELYON16]

胸部骨減弱処理
[Yang et al., 2016]

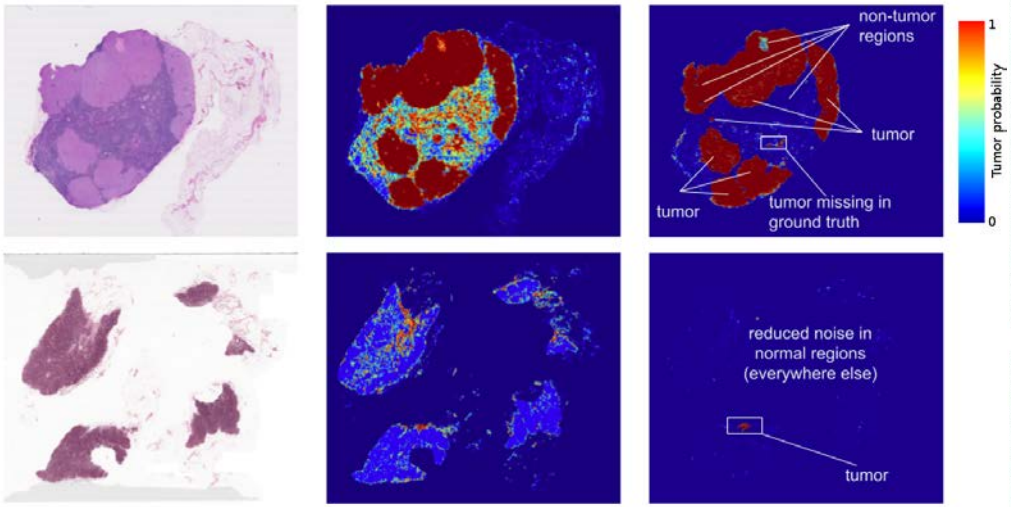
皮膚損傷分類
[Esteva et al., 2017]

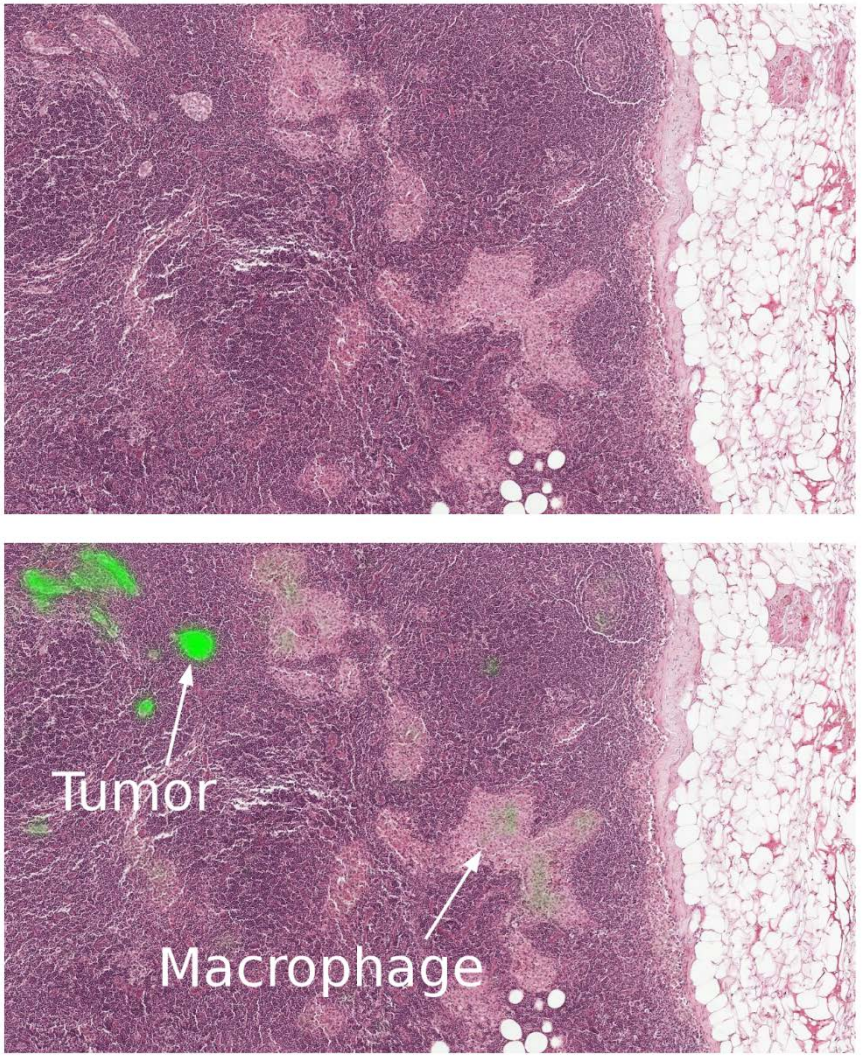
画像はLitjens, et al. (2017)より

# 高精細画像の処理



ギガピクセル画像の認識もできつつある.



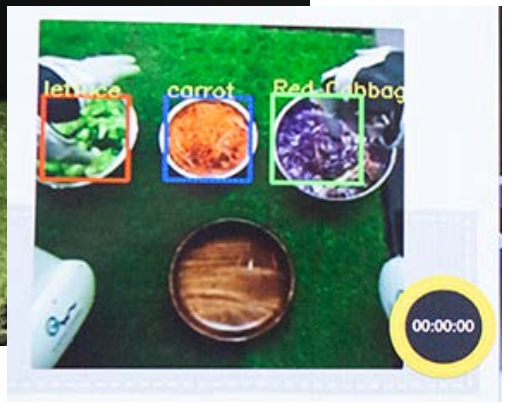

- 人を超える精度
  （FROC73.3% -> 87.3%）
- 悪性腫瘍の場所も特定

[Detecting Cancer Metastases on Gigapixel Pathology Images: Liu et al., arXiv:1703.02442, 2017]

# ロボット



[DENSO, 2017]




[タオル畳み、サラダ盛り付け　「指動く」ロボット初公開, ITMedia: http://www.itmedia.co.jp/news/articles/1711/30/news089.html]

[【ここまできた！】初公開の「汎用」マルチモーダルAIロボットアームはここが凄い！深層学習と予測学習を使い、VRでティーチング！, ロボスタ: https://robotstart.info/2017/11/29/denso-mmaira.html]

# 生成モデル

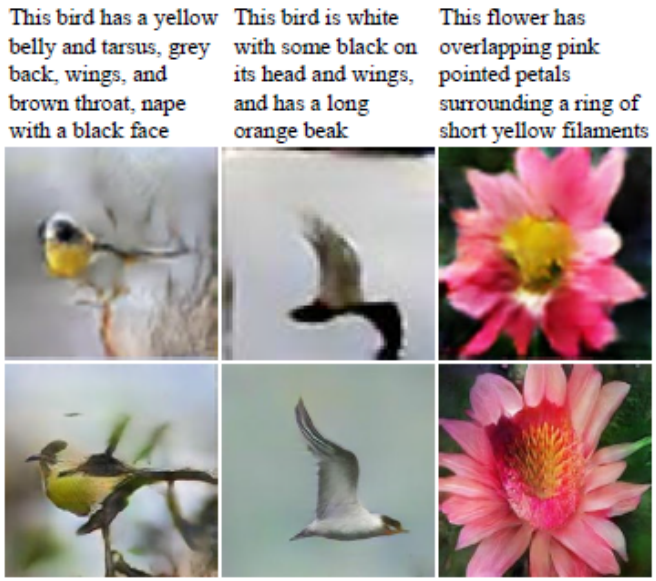
# 本物らしいデータを生成したい

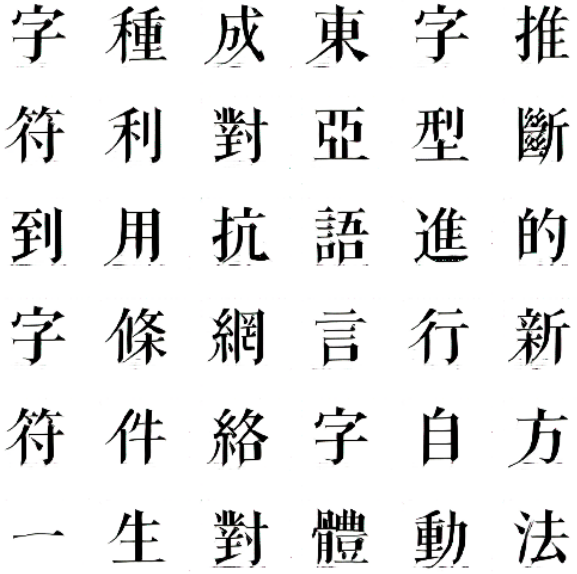


生成データ　　訓練データ

(a) Stage-I images

(b) Stage-II images



[Tian: zi2zi, Master Chinese Calligraphy with Conditional Adversarial Networks,2017]

深層学習が生成した画像

# 生成モデル



目標：本物らしい画像を生成したい.

- GAN (Generative Adversarial Network) [Goodfellow+et al., 2014]

2つの構成要素

Generator: $x = G(z)$

Discriminator: $D(x) = P(x\text{が本物})$

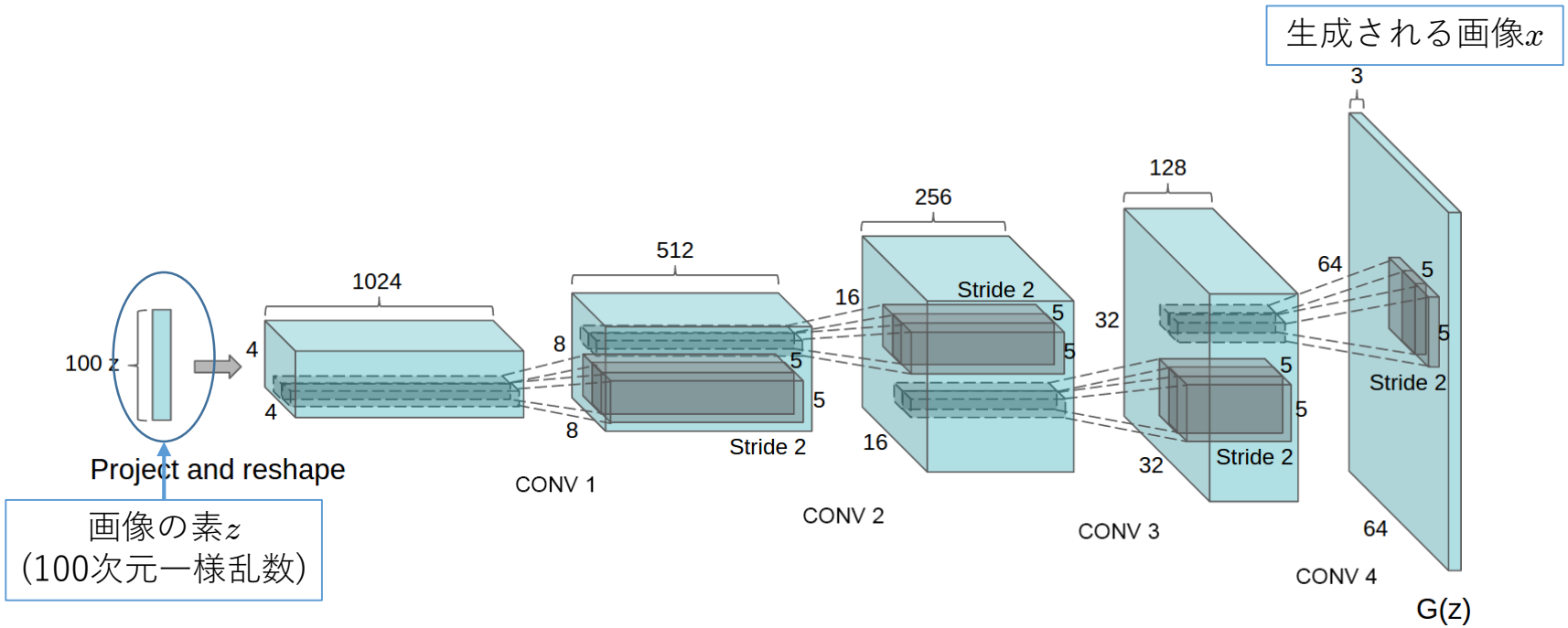
$G$: 画像の素$z$ (乱数) から偽画像$x$を生成. $D$を騙そうとする.
$D$: 画像$x$が本物か偽物か判別. $G$に騙されないようにする.

最適化問題

$$\min_G \max_D \underbrace{\mathrm{E}_{x\sim p_{\mathrm{data}}}[\log D(x)]}_{\text{本当の画像を本物と判別する確率}} + \underbrace{\mathrm{E}_{z\sim p_x}[\log(1 - D(G(z)))]}_{\text{偽物の画像を偽物と判別する確率}}$$

2016-2017にかなり流行
GANの変種まとめ：https://github.com/hindupuravinash/the-gan-zoo
※GANの他にもVAE (Variational Auto-Encoder)と呼ばれる方法もよく用いられている.

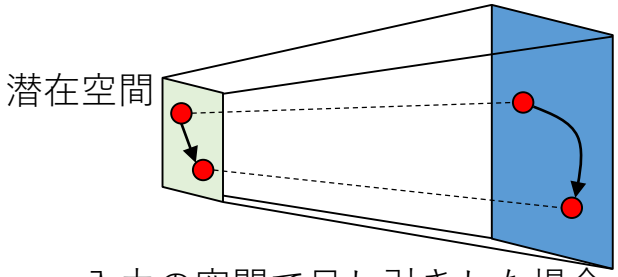
# DCGAN (Deep Convolutional GAN)
## 畳み込みネットを用いたGAN

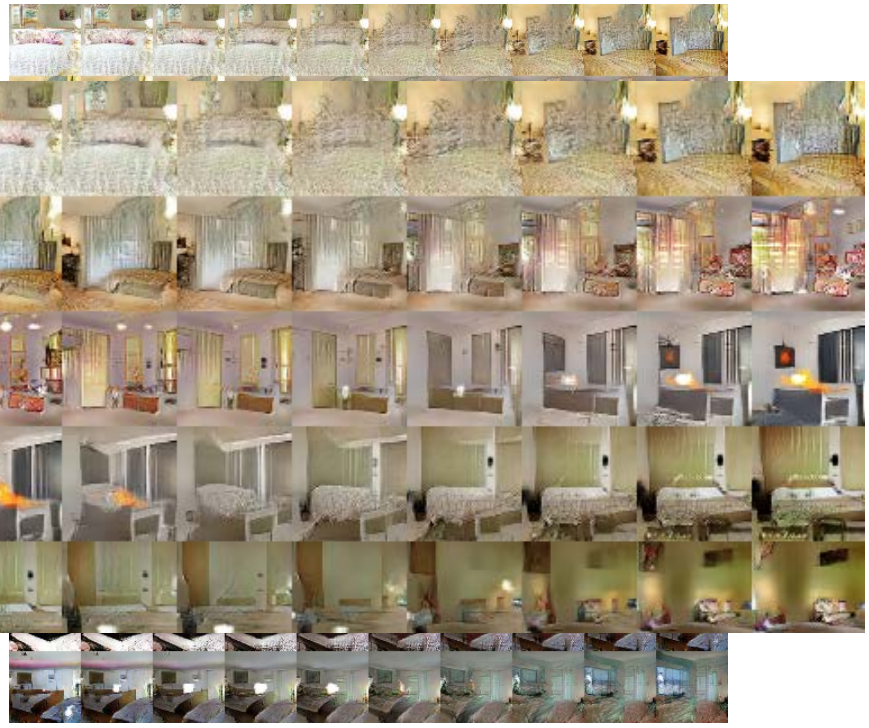



DCGANのGenerator

- 入力$z$は画像の低次元ベクトル表現にもなっている.
- Discriminatorも畳み込みネットを用いる.

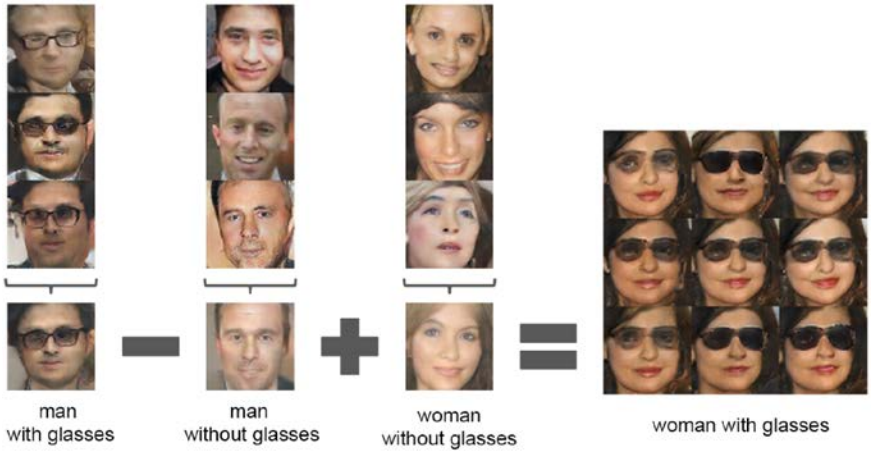
Radford, Metz & Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks.” ICLR2016.



生成されたベッドルーム画像

入力の空間で足し引きした場合




ピクセルごとに足し引きした場合


生成されたベッドルーム画像
入力zの凸結合で中間的画像が
得られる.

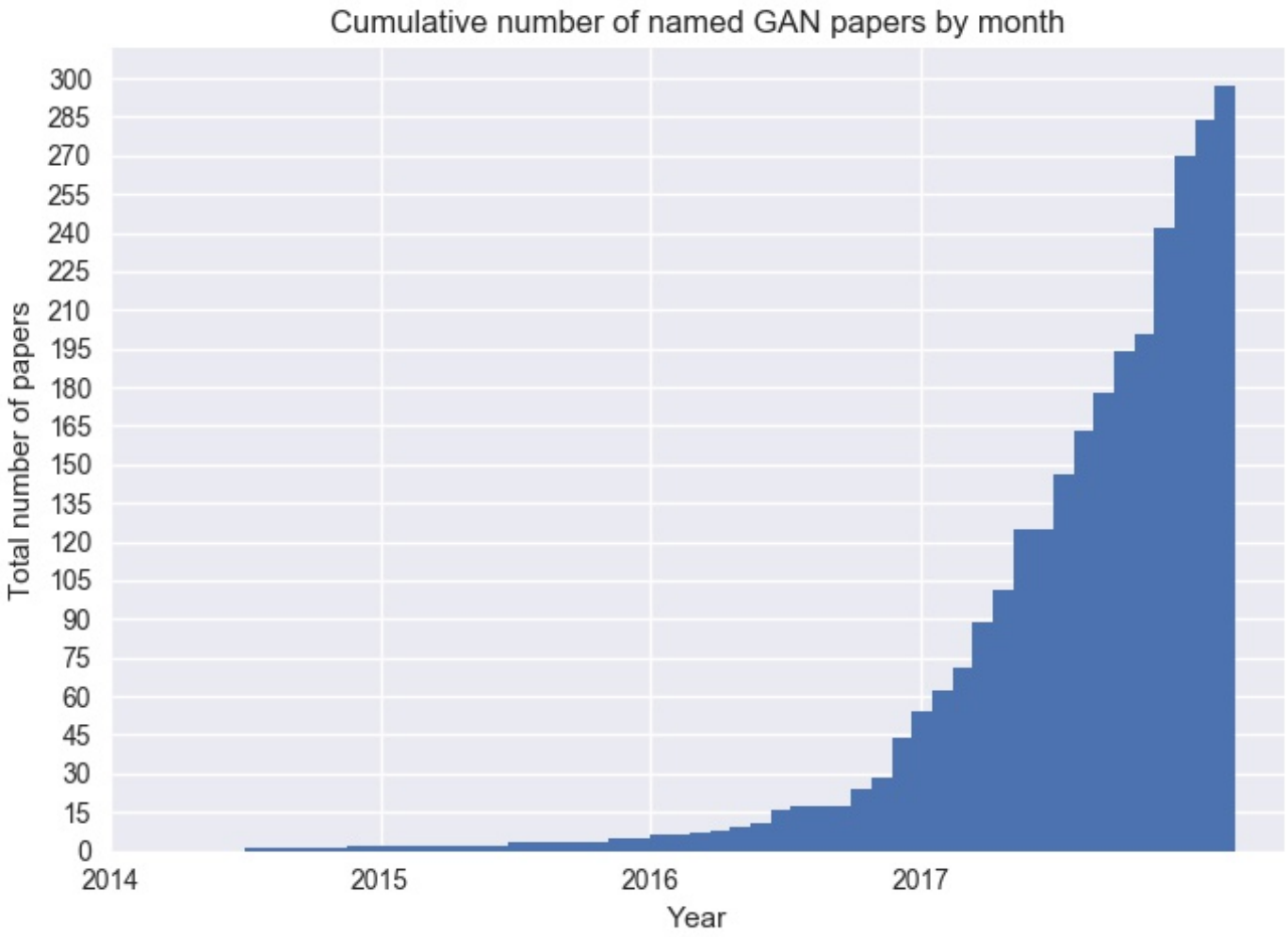
入力zを足し引きすることで意味
の足し引きが実現されている.
cf. word2vec.

Radford, Metz & Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks.” ICLR2016.

# GAN Zoo



「○○-GAN」




[https://github.com/hindupuravinash/the-gan-zoo]

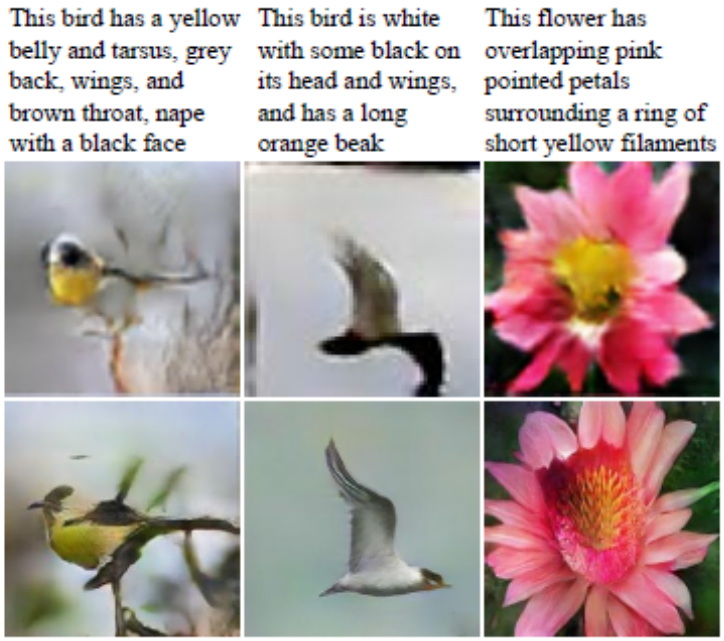
# StackGAN

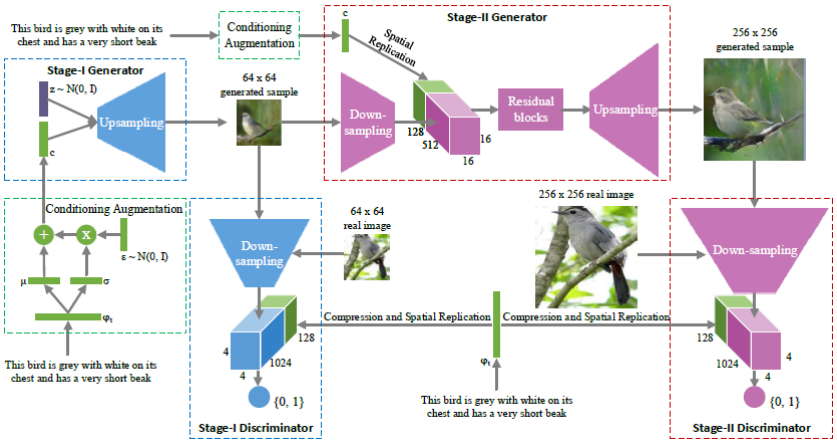
StackGAN [Zhang+etal.2016]　荒い画像を生成してからそれを高精細に修正（超解像）

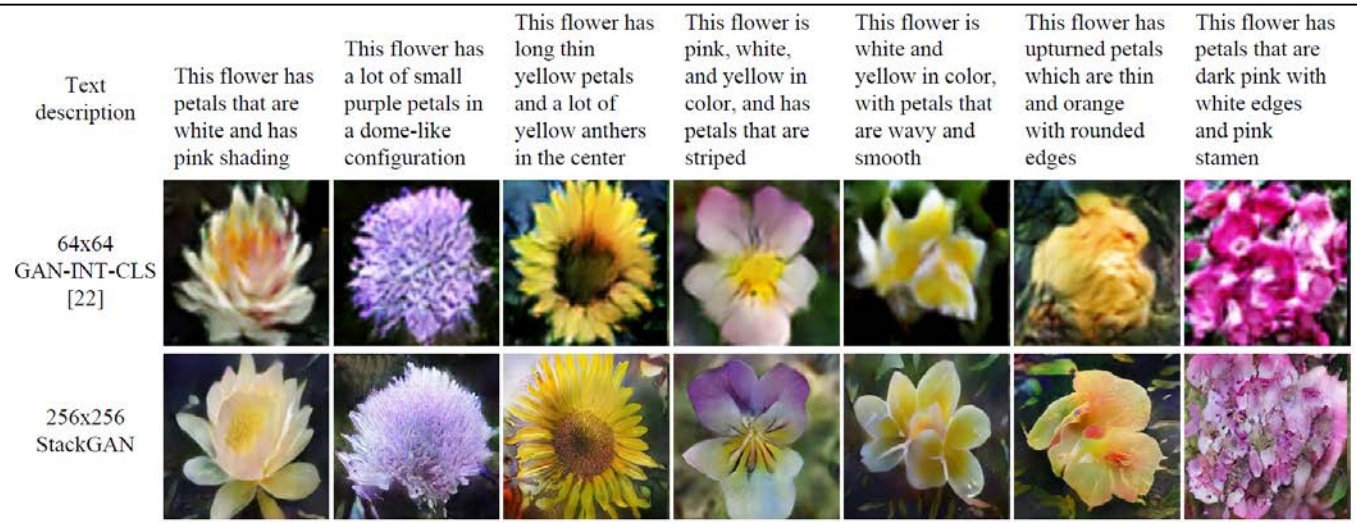
入力文章

荒い画像を生成

さらにこうなる



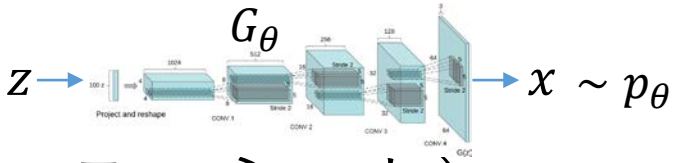

既存手法

StackGAN

# GANの仕組み



- $z$: 乱数（一様分布など）
- $x = G_\theta(z)$　（変数変換 by ニューラルネット）




適当な乱数を変数変換して目的の乱数（画像など）を生成

$x$の分布$p_\theta$ ⟷（近づけたい）真の分布$q$

**$f$-divergenceの最小化**

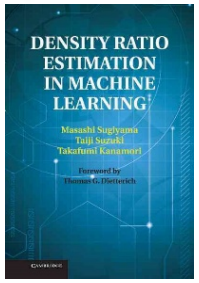
$f$-divergence:
分布の距離(のようなもの)

$$\int q(x) f\left(\frac{p_\theta(x)}{q(x)}\right) \mathrm{d}x$$

GANはJensen-Shannon divergenceに対応

f-GAN [Nowozin, Cseke, Tomioka, 2016]

**双対の関係**




密度比推定の方法論

**密度比$p_\theta(x)/q(x)$を1に近づける**

Bregman-divergence:

$$\mathrm{BR}_f(\hat{r}) = \int q(x)\{f'(\hat{r}(x)) - f(\hat{r}(x)) + f(r_\theta(x))\}\mathrm{d}x - \int p_\theta(x) f'(\hat{r}(x))\mathrm{d}x$$

真の密度比とのBregman-divergenceを最小化して密度比を推定

B-GAN [Uehara+et al., 2016]

[Sugiyama, Suzuki, Kanamori: Density ratio matching. 2012.]

# その他のアプローチ



- 分布間の距離が定義できれば何を用いても良い

- Wasserstein GAN (Metz et al., 2016)
  - Wasserstein距離を利用
  - 二つの分布のサポートがずれていてもwell-defined
  - 安定した学習

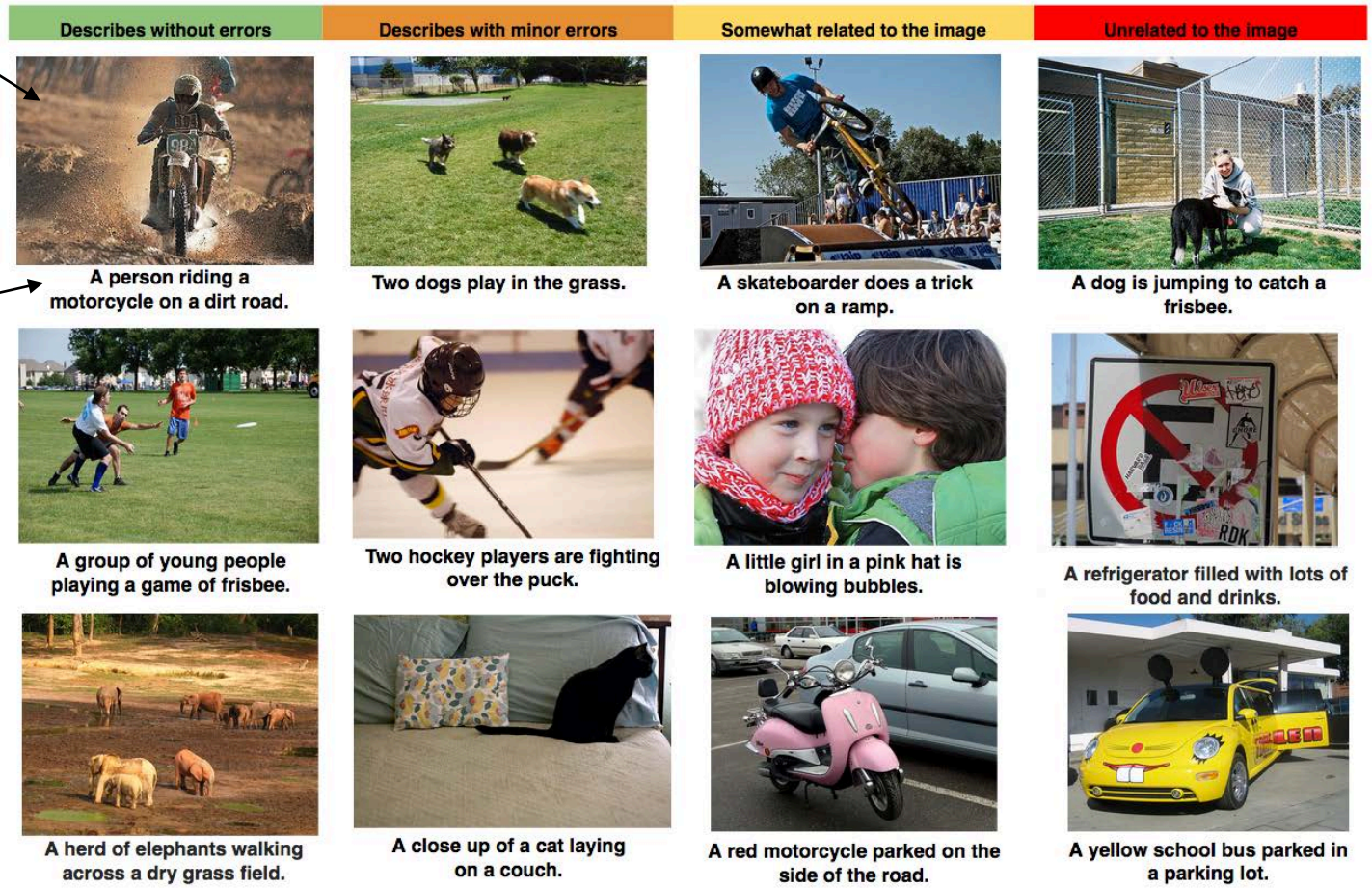
- MMD GAN (Li et al., 2017)
  - カーネル法による分布間距離 (Maximum Mean Discrepancy) を利用

# 自然言語処理

# キャプション生成

入力画像

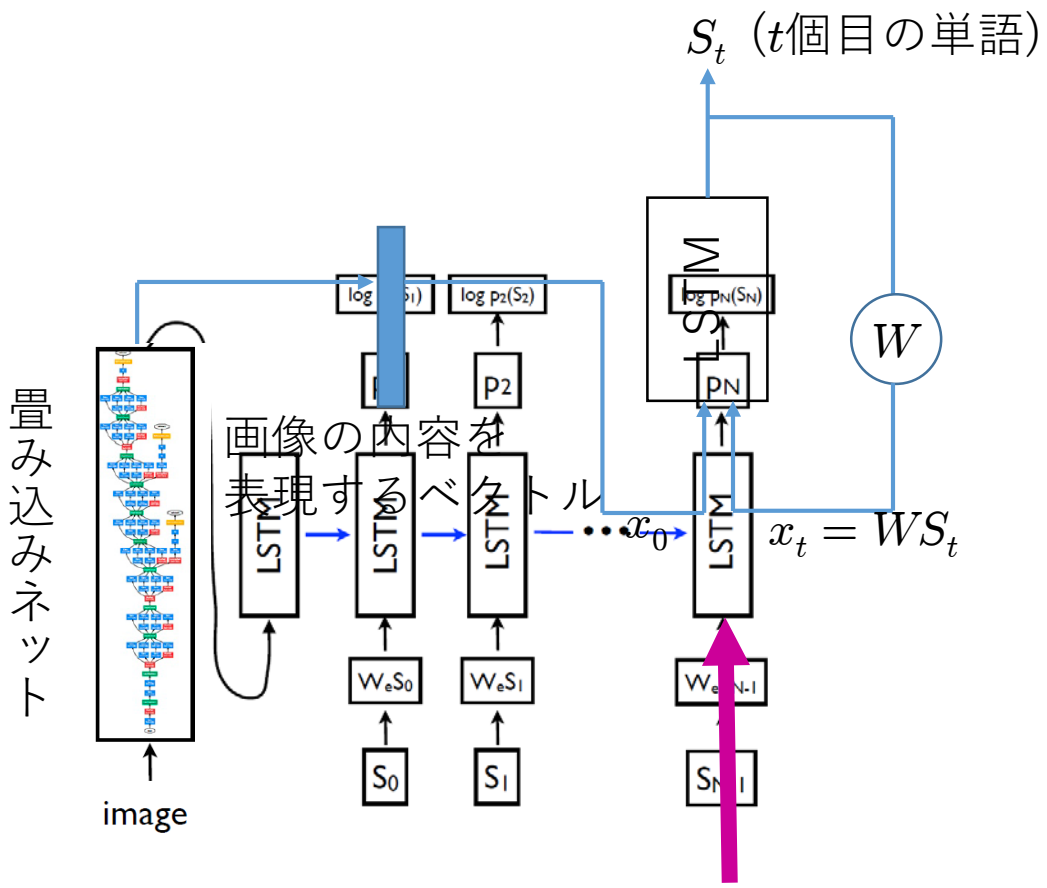
自動生成された説明文



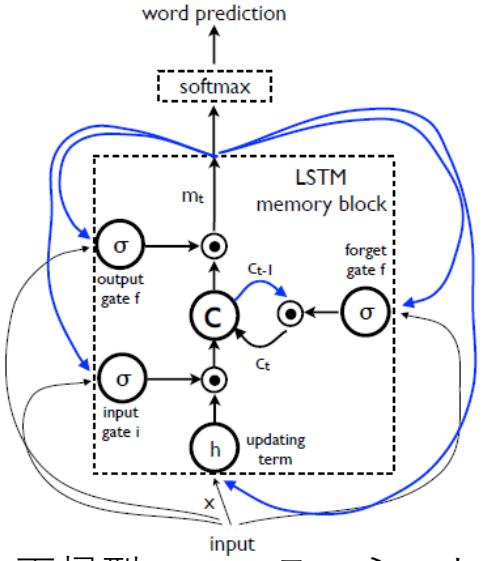

**Google** による画像説明文章の自動生成

[Vinyals et al., Show and tell: A neural image caption generator. CVPR, 2015]

# 深層学習を用いたキャプション生成モデル



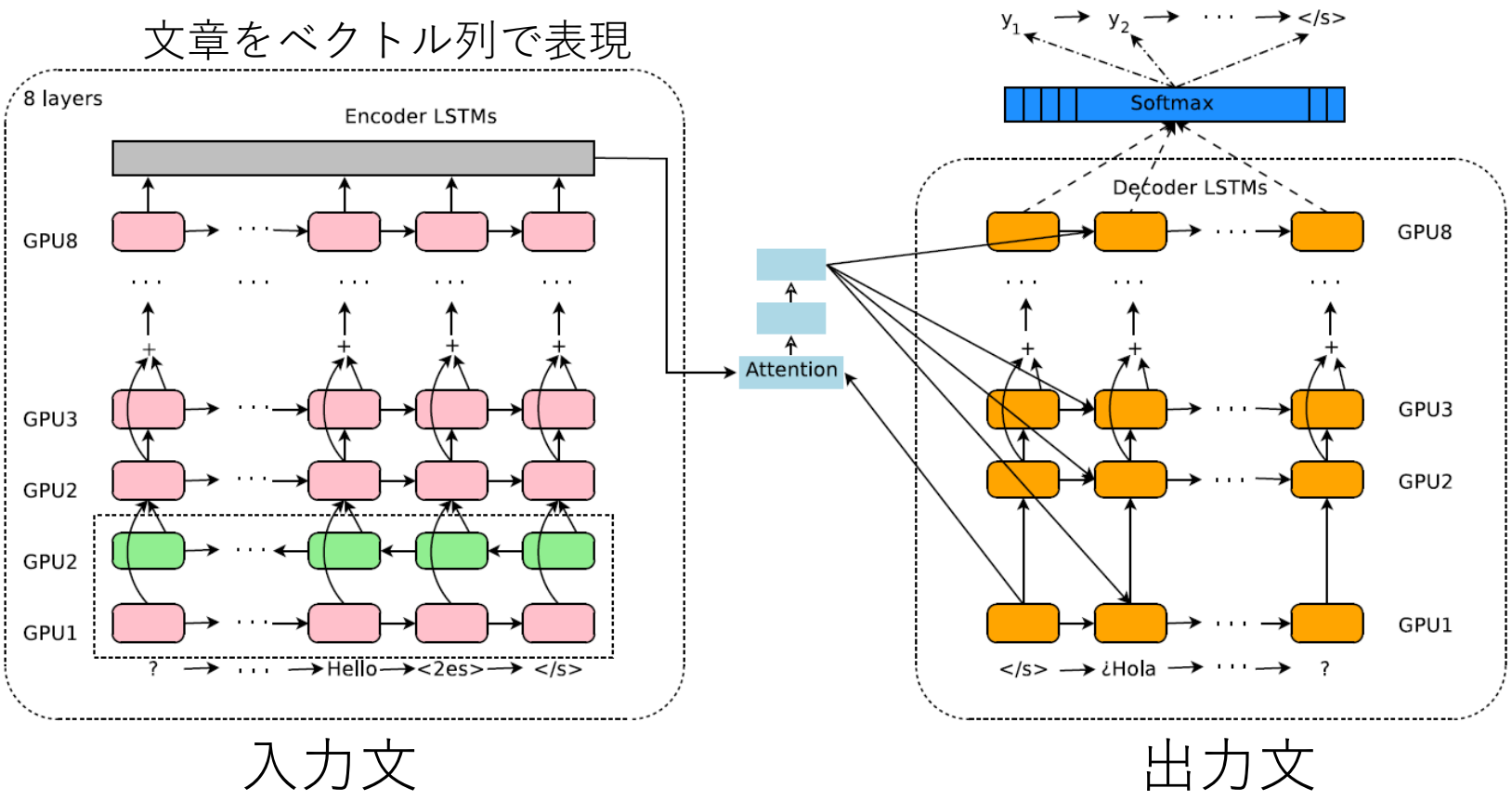

LSTM
(Long Short Term Memory)




- 再帰型ニューラルネット
- 言葉の列等を生成できる
- 内部状態，前の出力，今の入力で次の出力が決まる

- 最初の時刻だけ画像を表現するベクトルを入力
- 次の時刻からは前の時刻に自分の生成した単語を入力
- 画像の意味をベクトルで表すことが本質的

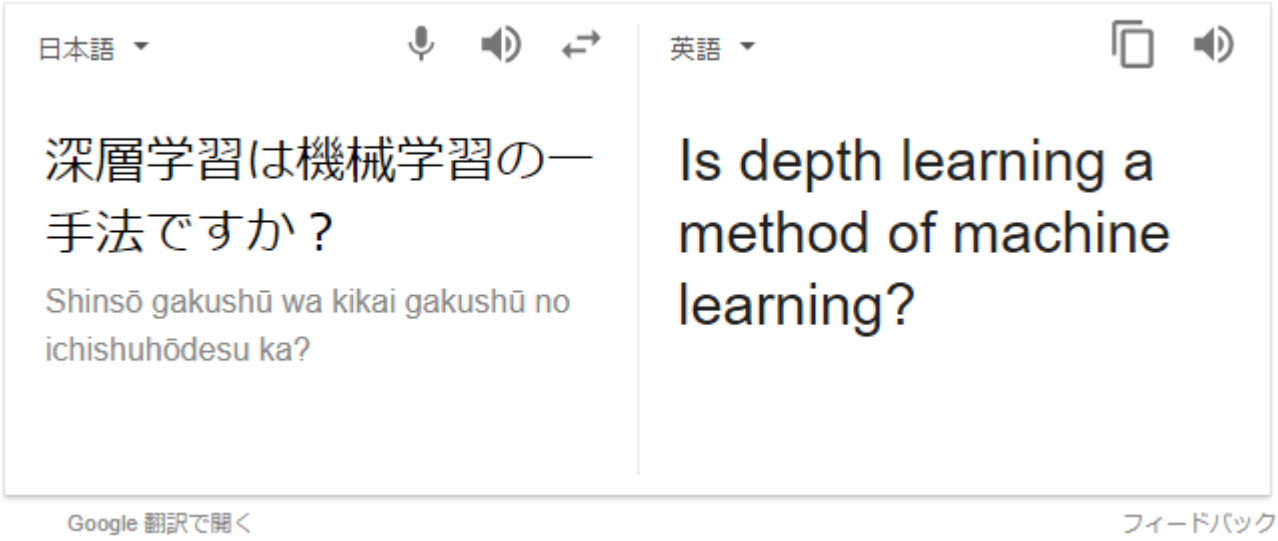
# 翻訳




Googleの翻訳システム

- 深層学習（再帰型ニューラルネットワーク）を利用
- 複数言語にも対応
- 「データのない言語対」の翻訳も可能に

# 翻訳の例



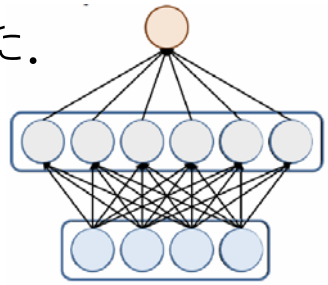

| | |
|---|---|
| この先生きのこるにはどうすればよいのか 編集 | How can I make this teacher mushrooms |

| | |
|---|---|
| この先，生きのこるにはどうすればよいのか 編集 | How can we live ahead? |

# 深層学習の理論

# 万能近似能力

ニューラルネットの関数近似能力は80年代に盛んに研究された.



$$f(x) = \sum_{j=1}^{m} v_j h(w_j^\top x + b_j)$$

なる関数が$m \to \infty$で任意の関数を任意の精度で近似できるか？

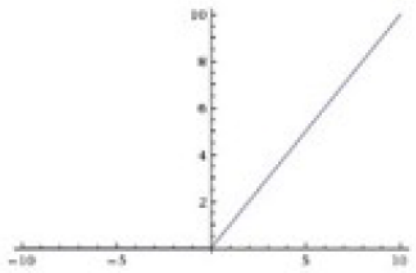
（「任意の関数」や「任意の精度」の意味はどのような関数空間を考えるかに依存）

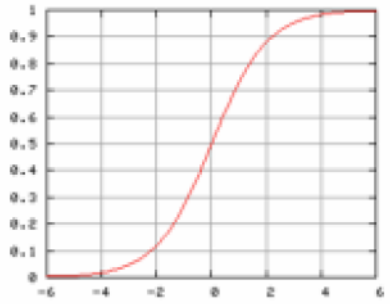
$h$がシグモイド関数やReLUなら万能性を有する.

| 年 | | 基底関数 | 空間 |
|---|---|---|---|

Activation functions:

**ReLU:** $\eta(u) = \max\{u, 0\}$ **Sigmoid:** $\eta(u) = \frac{1}{1+\exp(-u)}$





$K$は任意のコンパクト集合

参考：園田, “ニューラルネットの 積分表現理論”, 2015.

# 積分表現



有限和近似（３層NN）

$$\hat{f}(x) = \sum_{j=1}^{m} v_j \eta(w_j^\top x + b_j)$$

$\simeq$

真の関数

$$f^{\circ}(x) = \int h^{\circ}(w, b)\eta(w^\top x + b)\mathrm{d}w\mathrm{d}b$$

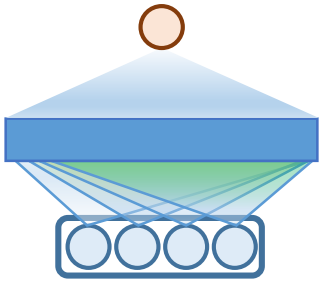
(Sonoda & Murata, 2015)

- Ridgelet変換による解析（Fourier変換の親戚）
- ３層NNはridgelet変換で双対空間（中間層）に行ってから戻ってくる（出力層）イメージ

# 三層ニューラルネットの汎化誤差



$$f(x) = \int_{\mathbb{R}^d} e^{iw^\top x} \tilde{f}(w) \mathrm{d}w$$ (Fourier変換)



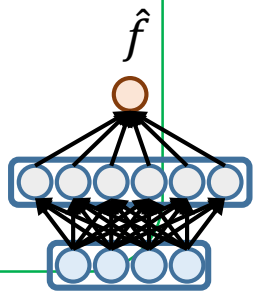
仮定： $$\int_{\mathbb{R}^d} \|w\| |\tilde{f}(w)| \mathrm{d}w < \infty$$

$(x_i, y_i)_{i=1}^n$ : i.i.d.　　$f(x) = \mathrm{E}[Y|X=x]$　　(例：$y_i = f(x_i) + \epsilon_i$ )

**三層ニューラルネットワークの汎化誤差 (Barron 1991, 1993)**

三層ニューラルネットワークのある種の正則化推定量$\hat{f}$が存在して次を満たす：

$$\mathrm{E}\left[\|\hat{f} - f\|^2_{L_2(P(X))}\right] \leq O\left(\sqrt{\frac{d \log(n)}{n}}\right)$$




活性化関数の条件： $\eta(-z) = 1 - \eta(z)$

$\|\eta'\|_\infty < \infty$

$\lim_{z \to -\infty} \sup \eta(z)/|z|^p < \infty$

(MDL, PAC-Bayes的解析)

理論より三層パーセプトロンでも中間層のユニット数を
無限に増やせば任意の関数を任意の精度で近似できる.

歴史的には後にSVMの理論に繋がってゆく.

(例：Gaussian kernelの万能性)

**Q**：ではなぜ深い方が良いのか？

**A**：深さに対して**指数的に表現力が増大**するから.

# 表現力と層の数



NNの“表現力”：領域を何個の多面体に分けられるか？

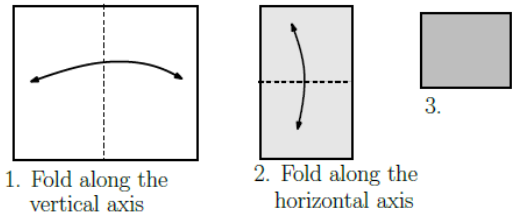
- 層の数に対して表現力は**指数的**に上がる.

$$\left(\frac{n}{n_0}\right)^{L-1}\sum_{j=0}^{n_0}\binom{n}{j}$$

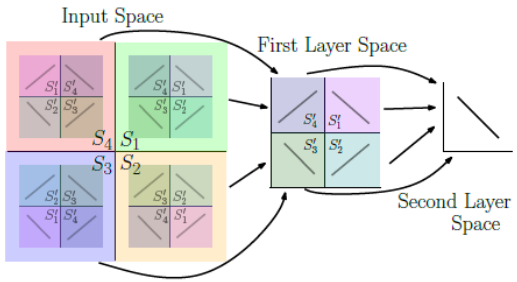
- 中間層のユニット数 (横幅) に対しては**多項式的**.

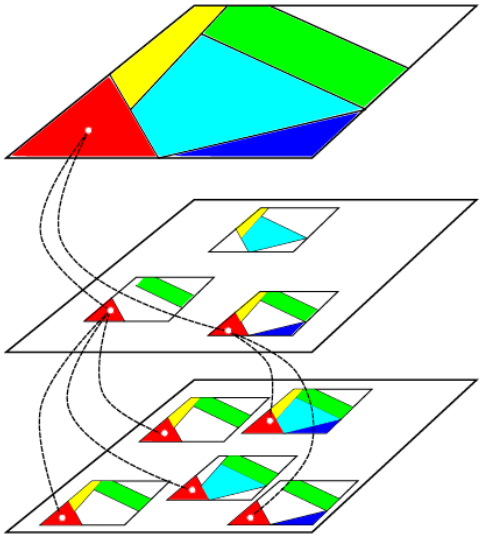
$$\sum_{j=0}^{n_0}\binom{n}{j}$$

$L$：層の数
$n$：中間層の横幅
$n_0$：入力の次元




(a)



折り紙のイメージ

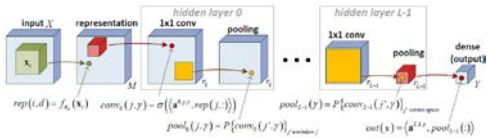
Montufar, Guido F., et al. "On the number of linear regions of deep neural networks." 2014.

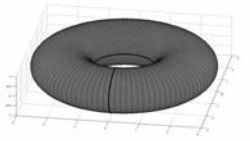
# 多層で得する理由



他にも同様の結論を出している論文多数

- **多項式展開，テンソル解析** [Cohen et al., 2016; Cohen & Shashua, 2016]
  単項式の次数
- **代数トポロジー** [Bianchini & Scarselli, 2014]
  ベッチ数(Pfaffian)
- **リーマン幾何 + 平均場理論** [Poole et al., 2016]
  埋め込み曲率

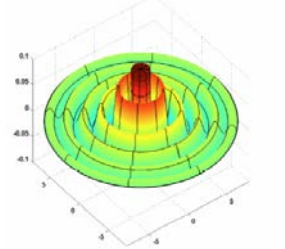


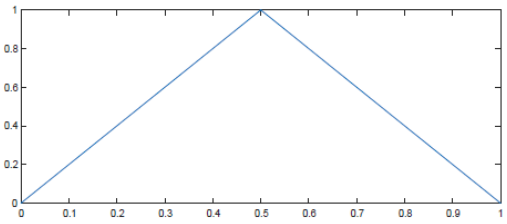
対称性の高い関数は，特に層を深くすることで得をする．

$$h(x) = \begin{cases} 2x & (0 \le x \le 1/2) \\ 2(1-x) & (1/2 \le x \le 1) \\ 0 & (\text{otherwise}). \end{cases}$$

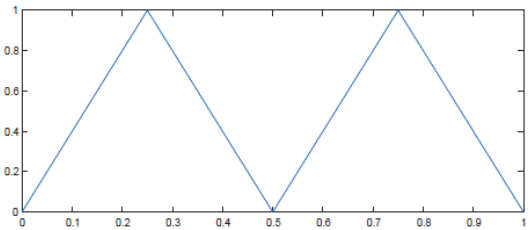
多層が得する例



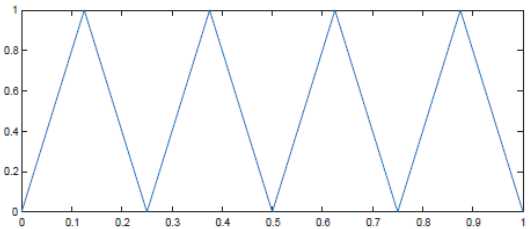
[Eldan,Shamir,ALT2016]



$h(x)$



$h \circ h(x)$

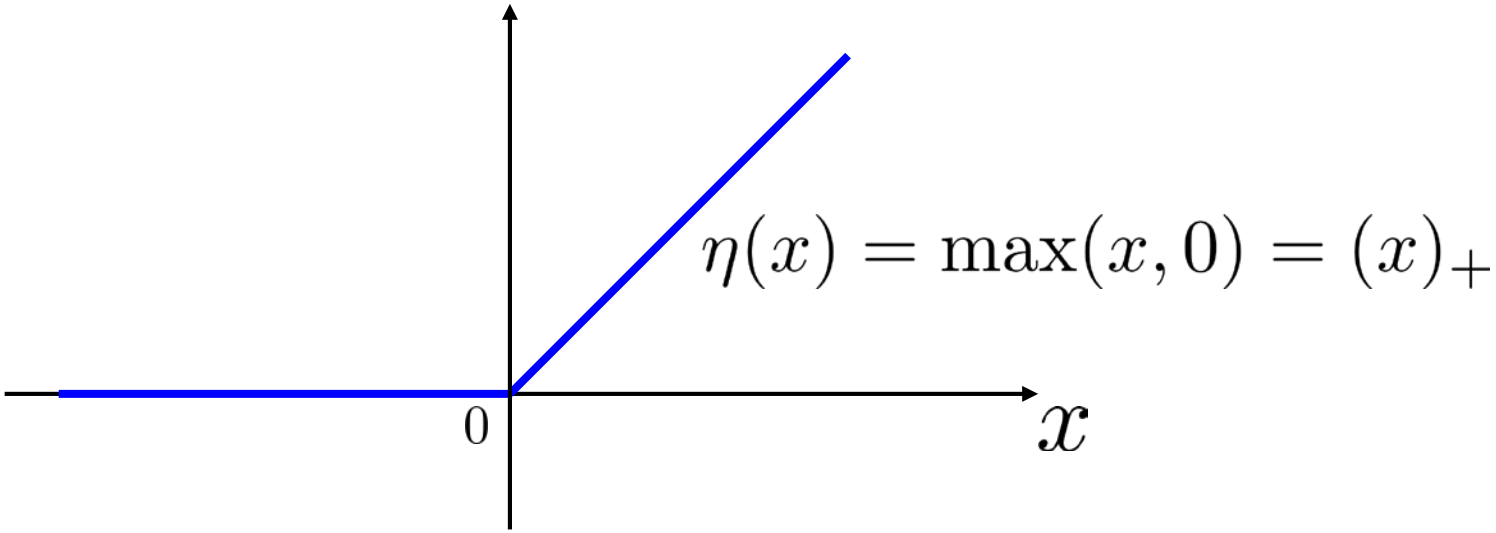


$h \circ h \circ h(x)$

# ReLUの表現力



- ReLU活性化関数




- 現在広く使われている
(LeakyReLUなどの亜種もあるがかなりスタンダード)

- 統計的性質も解明されつつある
  - 万能近似能力あり
  - (区分的) 滑らかな関数の推定
  - 区分線形関数の表現
  - 有理関数の表現
  - 関数のテンソル積，合成関数の表現

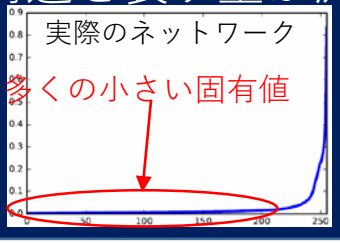
# 深層学習の汎化誤差理論

[T. Suzuki. Fast learning rate of deep learning via a kernel perspective. arXiv:1705.10182, 2017.]

## 深層学習のネットワーク構造を決定する指針はないか？

- 「自由度」という深層ネットワークの構造を表す量が汎化誤差に影響

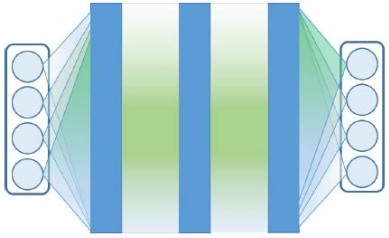
「自由度」は中間層の出力の分散共分散行列の固有値に対応して決まる




小さい固有値が多ければ横幅は狭くて良い

深層学習の汎化誤差を決める要素は何か？
→**自由度**が重要

深層NNの積分表現（真の関数）

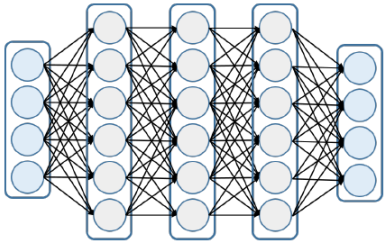


$$F_\ell(\tau, x) = \int_{\mathcal{T}_\ell} \underbrace{h_\ell^o(\tau, \tau')}_{\text{Weight}} \eta(F_{\ell-1}(\tau', x)) \mathrm{d}Q_\ell(\tau') + \underbrace{b_\ell^o(\tau)}_{\text{Bias}}$$

有限近似

求積法の理論

有限次元モデル



汎化誤差=真の関数とモデルのずれ（バイアス）＋モデルの複雑さ（バリアンス）

- 真の関数を表すために積分表現を導入
- 積分表現から各層に再生核ヒルベルト空間を定義
- 空間に付随した「自由度」を定義

自由度 $N_\ell(\lambda) = \sum_{j=1}^{\infty} \frac{\mu_j^{(\ell)}}{\mu_j^{(\ell)} + \lambda}$

ただし$\mu_j^{(\ell)}$は各層に対応するカーネルの固有値

（各層の実質的次元）

**定理** 第$\ell$層の横幅$m_\ell$が$m_\ell \geq N_\ell(\lambda_\ell)$ならば

$$\|\hat{f} - f^*\|_{L^2}^2 \leq 2(\hat{\delta}^2 + \epsilon_n^2)$$

$$\hat{\delta} = \sum_{\ell=2}^{L} 2\sqrt{\hat{c}_\delta^{L-\ell-1}} R^{L-\ell} \sqrt{\lambda_\ell}$$ バイアス

$$\epsilon_n = C\sigma \sqrt{\frac{\sum_{\ell=1}^{L} m_{\ell+1} m_\ell}{n} \log(n)}.$$ バリアンス

一般の深層NN $\sum_{\ell=1}^{L} n^{-\frac{1}{1+2s_\ell}} \log(n)$

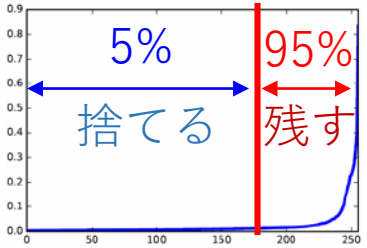
3深NN（カーネル法） $n^{-\frac{1}{1+s_1}} \log(n)$

# 理論の応用

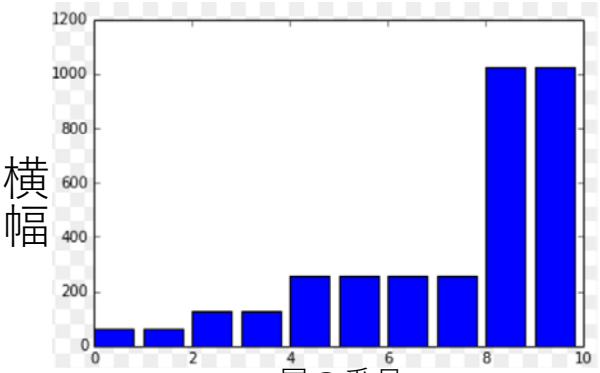
- **構造の自動決定**：各レイヤーの横幅をデータから決定できる.
  →**ネットワークの圧縮に利用：予測値計算の高速化+メモリ効率化**

構造の自動決定：各層の横幅（チャネル数）


VGG13:
よく使われるモデル



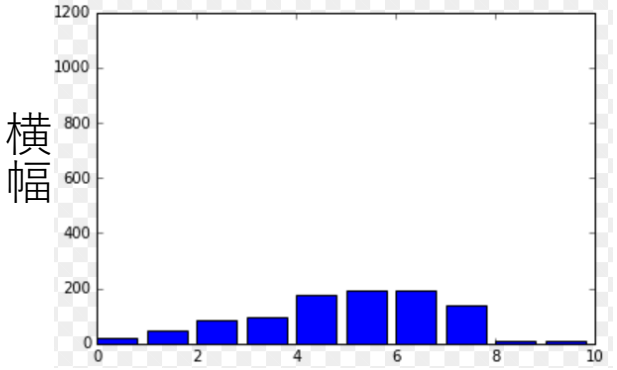

64,64,128, 128,256,256,256,256,1024,1024




精度：0.9374

提案手法でサイズを自動決定
**結果的に大きくサイズを圧縮**

37,58,114,117, 229,236,234,201, 10,10



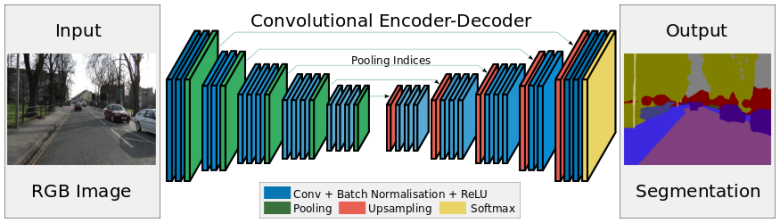

精度：0.9375

データ：cifar10

モデルを圧縮しても精度は損なわず

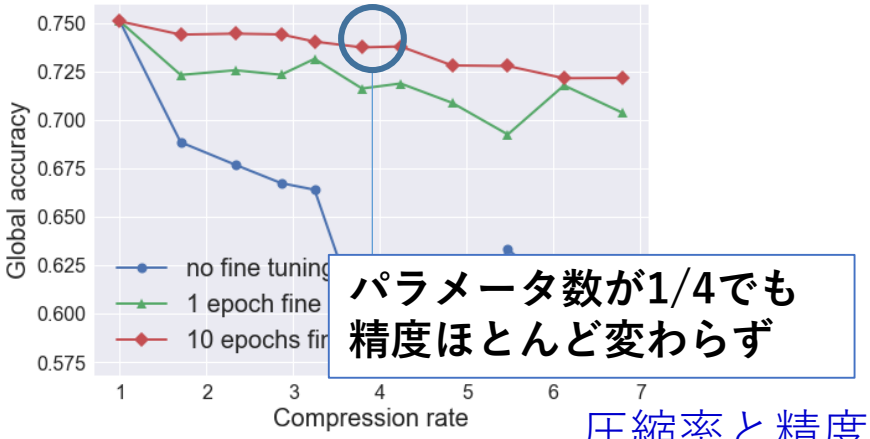
# SegNet(-Basic)の圧縮





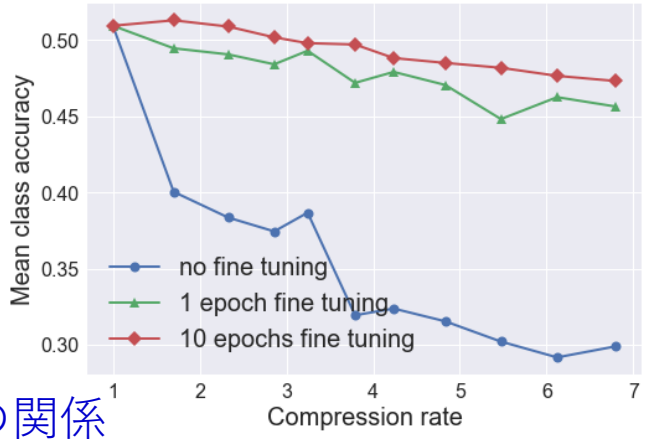

SegNet-Basicに応用

Global Accuracy




**パラメータ数が1/4でも**
**精度ほとんど変わらず**

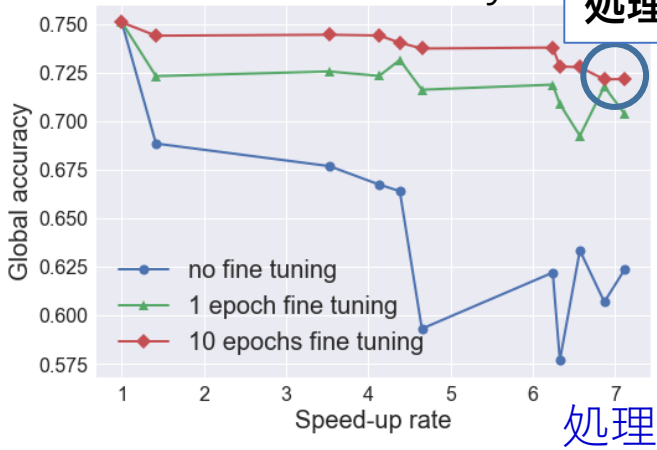
Mean Class Accuracy



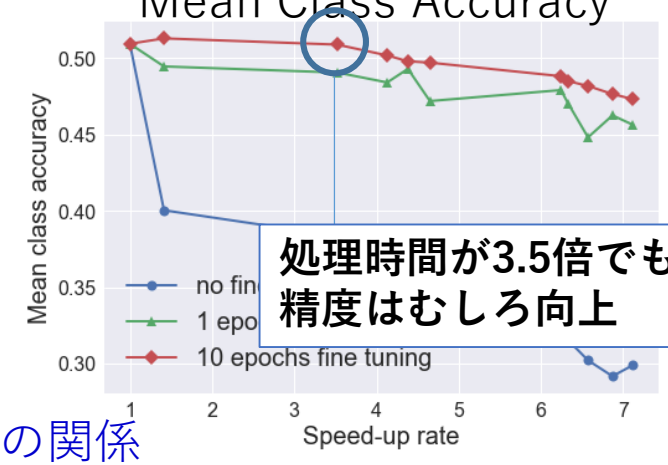

圧縮率と精度の関係

Global Accuracy

**処理時間7倍**




Mean Class Accuracy



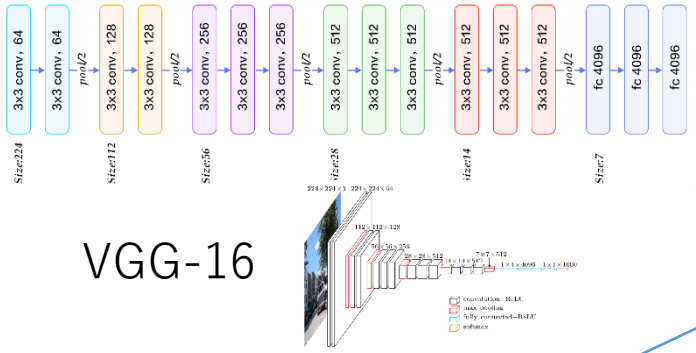

**処理時間が3.5倍でも**
**精度はむしろ向上**

処理時間と精度の関係

# ImageNetデータセットでの実験



- 1,300万枚の訓練画像
- 1,000クラスへの分類




VGG-16

VGG-16ネットワークの圧縮

| Model | Top-1 | Top-5 | # Param. | FLOPs |
|---|---|---|---|---|
| Original VGG | 68.34% | 88.44% | 138.34M | 30.94B |
| SqueezeNet | 57.67% | 80.39% | 1.24M | 1.72B |
| ThiNet-Conv | 69.80% | 89.53% | 131.44M | 9.58B |
| ThiNet-GAP | 67.34% | 87.92% | 8.32M | 9.34B |
| ThiNet-Tiny | 59.34% | 81.97% | 1.32M | 2.01B |
| **Ours-Conv1** | 72.23% | 91.17% | 132.75M | 22.41B |
| **Ours-Conv2** | 69.61% | 89.34% | 113.92M | 9.71B |
| **Ours-Conv-FC** | 68.66% | 88.90% | 45.77M | 9.58B |
| **Ours-GAP** | 67.09% | 87.90% | 8.40M | 12.5B |
| **Ours-Tiny** | 60.10% | 82.89% | 2.31M | 2.07B |

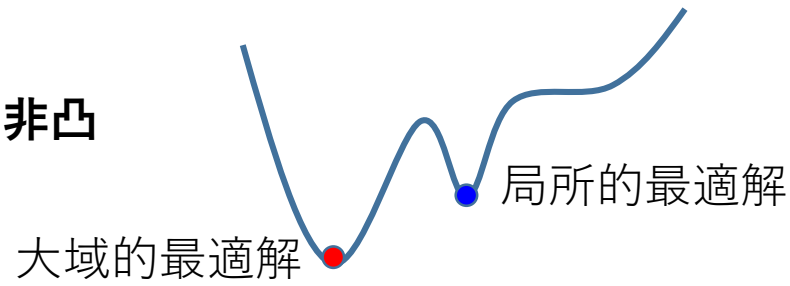
精度向上

元モデルの1/3程度のサイズでむしろ精度向上　我々の提案手法

# 局所最適性



**深層学習の目的関数は非凸**




- 深層NNの局所的最適解は全て大域的最適解：
  Kawaguchi, 2016; Lu&Kawaguchi, 2017.

  **※ただし対象は<u>線形NN</u>のみ.**

  → 臨界点が大域的最適解であることの条件も出されている
  (Yun, Sra&Jadbabaie, 2018)

- 低ランク行列補完の局所的最適解は全て大域的最適解：
  Ge, Lee&Ma, 2016; Bhojanapalli, Neyshabur&Srebro, 2016.

$$\min_{U \in \mathbb{R}^{M \times k}} \sum_{(i,j) \in E} (Y_{i,j} - (UU^\top)_{i,j})^2$$

# 3層NN-非線形活性化関数-



## 二層目の重みを固定する設定

(Tian, 2017; Brutzkus and Globerson, 2017; Li and Yuan, 2017; Soltanolkotabi, 2017; Soltanolkotabi et al., 2017; Shalev-Shwartz et al., 2017; Brutzkus et al., 2018)

固定 こちらのみ動かす

$$y = \sum_{j=1}^{k} v_j \eta(w_j^\top x + b_j)$$

- Li and Yuan (2017): ReLU，入力はガウス分布を仮定
  - SGDは多項式時間で大域的最適解に収束
  - 学習のダイナミクスは2段階
    → 最適解の近傍へ近づく段階 + 近傍での凸最適化的段階
- Soltanolkotabi (2017): ReLU，入力はガウス分布を仮定
  - 過完備（横幅>サンプルサイズ）なら勾配法で最適解に線形収束
    (Soltanolkotabi et al. (2017)は二乗活性化関数でより強い帰結)
- Brutzkus et al. (2018): ReLU
  - 線形分離可能なデータなら過完備ネットワークで動かしたSGDは大域的最適解に有限回で収束し，過学習しない．
    (線形パーセプトロンの理論にかなり依存)

Li and Yuan (2017): Convergence Analysis of Two-layer Neural Networks with ReLU Activation.
Soltanolkotabi (2017): Learning ReLUs via Gradient Descent.
Brutzkus, Globerson, Malach and Shalev-Shwartz (2018): SGD learns over parameterized networks that provably generalized on linearly separable data.

# 3層NN-非線形活性化関数-



## 二層目の重みも動かす設定

両方動かす

$$y = \sum_{j=1}^{k} v_j \eta(w_j^\top x + b_j)$$

※細かい強い仮定が置かれているので文章から結果を鵜呑みにできないことに注意.

- Du et al. (2017): CNNを解析
  - **勾配法は局所最適解があっても非ゼロの確率で回避可能**
    → ランダム初期化を複数回行えば高い確率で大域解へ
  - ガウス入力を仮定
- Du, Lee & Tian (2018): CNN, vj固定だが非ガウス入力で大域的最適解への収束を保証.

**学習ダイナミクスとセットで議論**

## その他のアプローチ

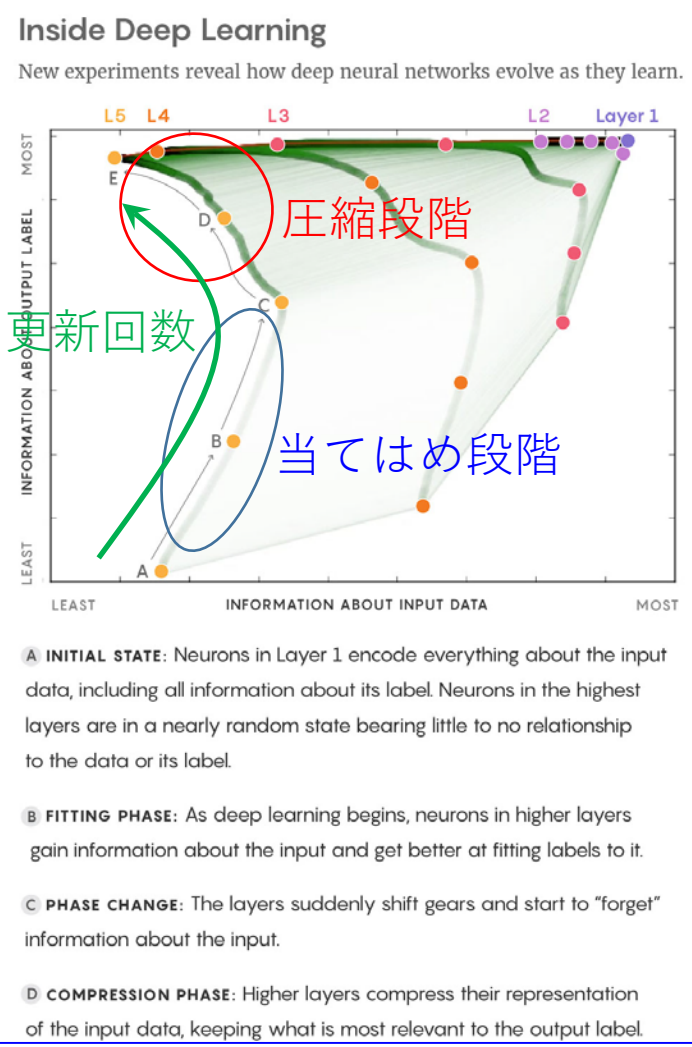
- テンソル分解を用いた大域的最適性の議論: Ge, Lee & Ma (2018).
- カーネル法的解釈+Frank-Wolfe法による最適化：Bach (2017).

Du, Lee, Tian, Poczos & Singh (2017): Gradient Descent Learns One-hidden-layer CNN: Don't be Afraid of Spurious Local Minima.
Du, Lee, Tian (2018): When is a convolutional filter easy to learn?
Ge, Lee & Ma (2018): Learning one-hidden-layer neural networks with landscape design.
Bach (2017): Breaking the Curse of Dimensionality with Convex Neural Networks.

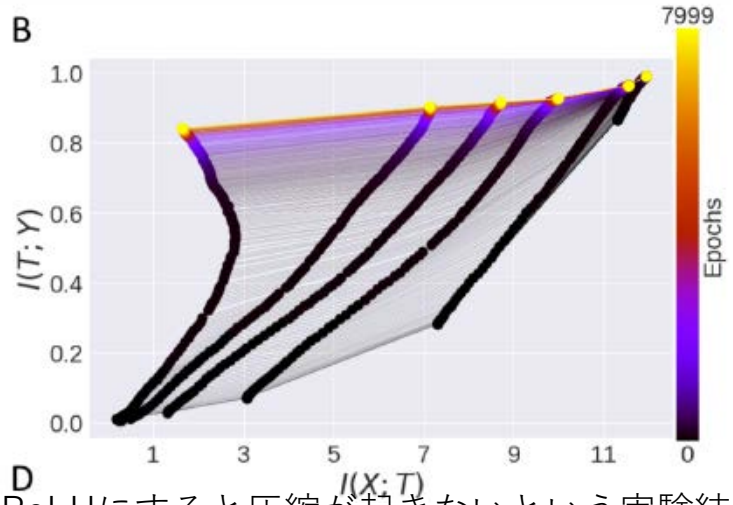
# Information Bottleneck



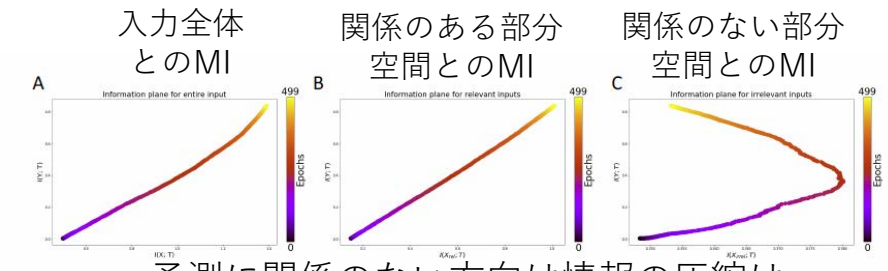
中間層と入力および出力との相互情報量




SGDによる最適化の間，まずデータへの当てはまりを良くしてから，無駄な情報の圧縮が始まる，という説




ReLUにすると圧縮が起きないという実験結果も．



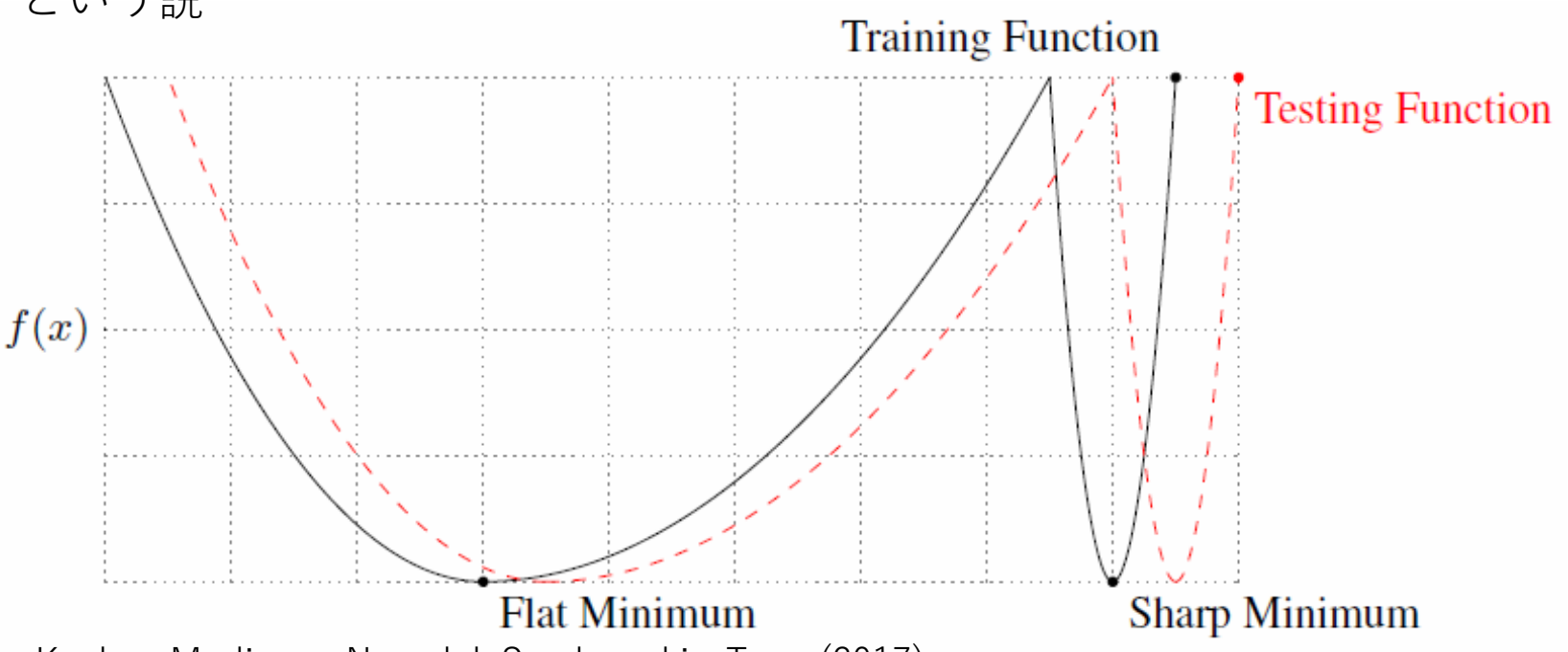

予測に関係のない方向は情報の圧縮は進んでいるという実験結果．

Tishby, Pereira, Bialek (2000): The information bottleneck method.
Tishby, Zaslavsky (2015): Deep learning and the information bottleneck principle.
Schwartz-iv, Tishby (2017): Opening the black box of Deep Neural Networks via Information.
Saxe, Bansal, Dapello, Advani, Kolchinsky, Tracey, Cox (2018): On the information bottleneck theory of deep learning.

# Sharp minima vs flat minima



**SGDは「フラットな局所最適解」に落ちやすい→良い汎化性能を示す**
という説




Keskar, Mudigere, Nocedal, Smelyanskiy, Tang (2017):
On large-batch training for deep learning: generalization gap and sharp minima.

$$\theta_t = \theta_{t-1} - \alpha_b \underbrace{\left( \frac{1}{b} \sum_{j=1}^{b} \nabla_\theta \ell(z_{i_j}; \theta) \right)}_{\cong \text{正規分布}}$$

→ランダムウォークはフラットな領域にとどまりやすい

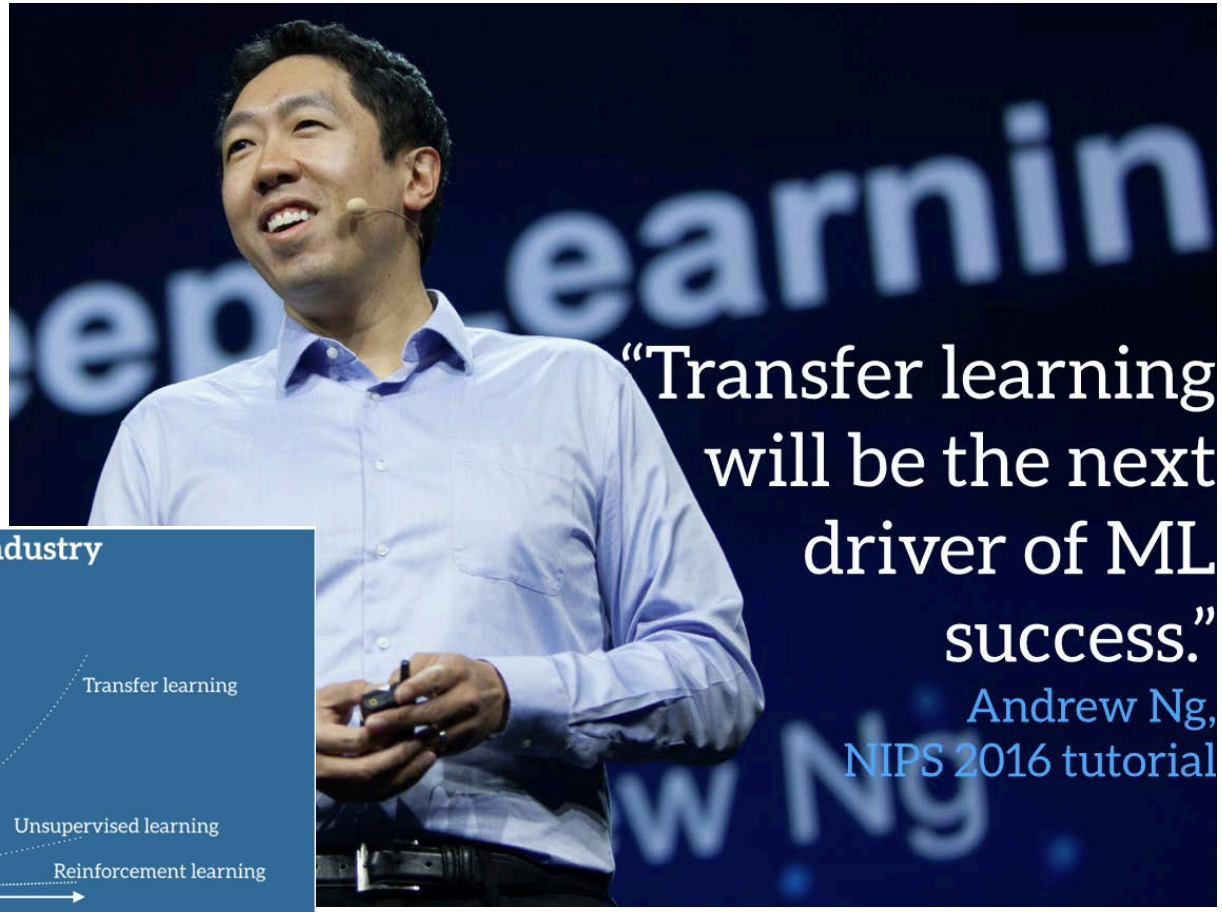
- 「フラット」という概念は座標系の取り方によるから意味がないという批判. (Dinh et al., 2017)
- PAC-Bayesによる解析 (Dziugaite, Roy, 2017)

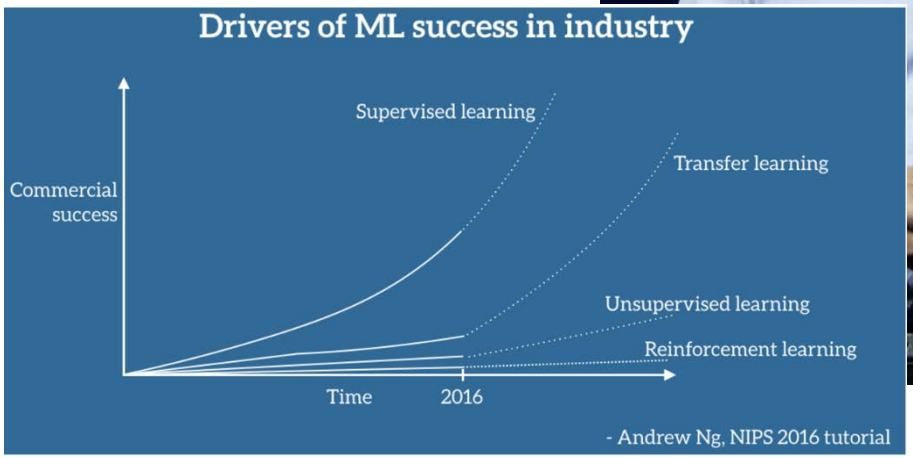
# 転移学習・メタ学習



いかに少ないデータで学習するか？
大量のデータで学習しておき，興味のある問題にその知識を「転移」させる．

転移学習
教師無し学習
メタ学習
ワンショット学習
Learn-to-learn

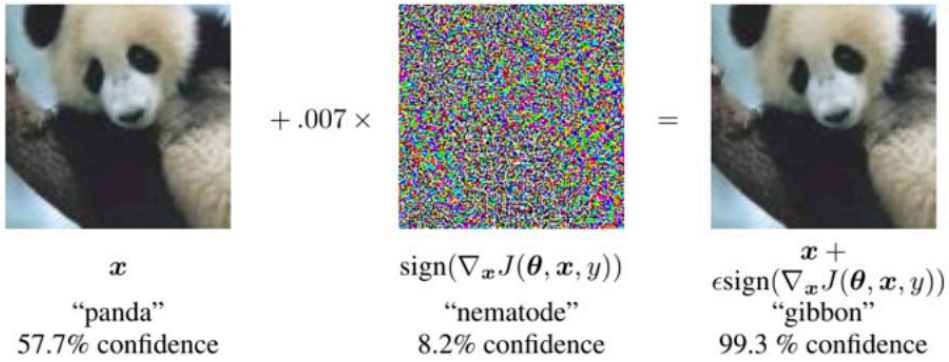


# 深層学習の脆さ



- Adversarial example




少し作為的ノイズを入れただけでパンダをテナガザルと間違える.
しかもかなり強い自信をもって間違える.

[Szegedy et al.: Intriguing properties of neural networks. ICLR2014.]


標識をハックすることで誤認識を誘発.

「STOP」を「スピード制限時速45mile」と誤認識

[Evtimov et al.: Robust Physical-World Attacks on Machine Learning Models. 2017]

敵対的入力 (adversarial example) に関する研究は現在盛り上がってる.
様々な対処法 (dropoutやVirtual Adversarial Trainingなど) も提案されている.
しかし，深層学習の信頼性評価はまだ難しい

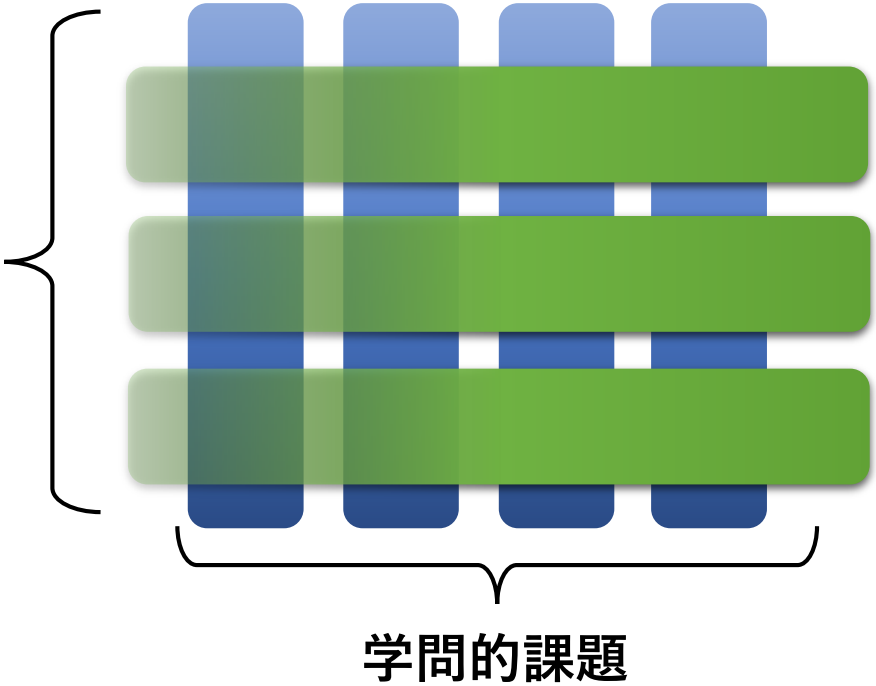
# 最後に



- 機械学習基礎研究による各種方法論の正しい理解
- 社会のニーズに応える産業界と時代によらない普遍的価値を求めるアカデミアの交流

「機械学習は両業界の距離が近い」

横糸と縦糸の関係



ご清聴ありがとうございました.