京都大学集中講義

機械学習と深層学習の数理と応用 (2)

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
Japan Digital Design

2018年12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

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

統計的学習

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

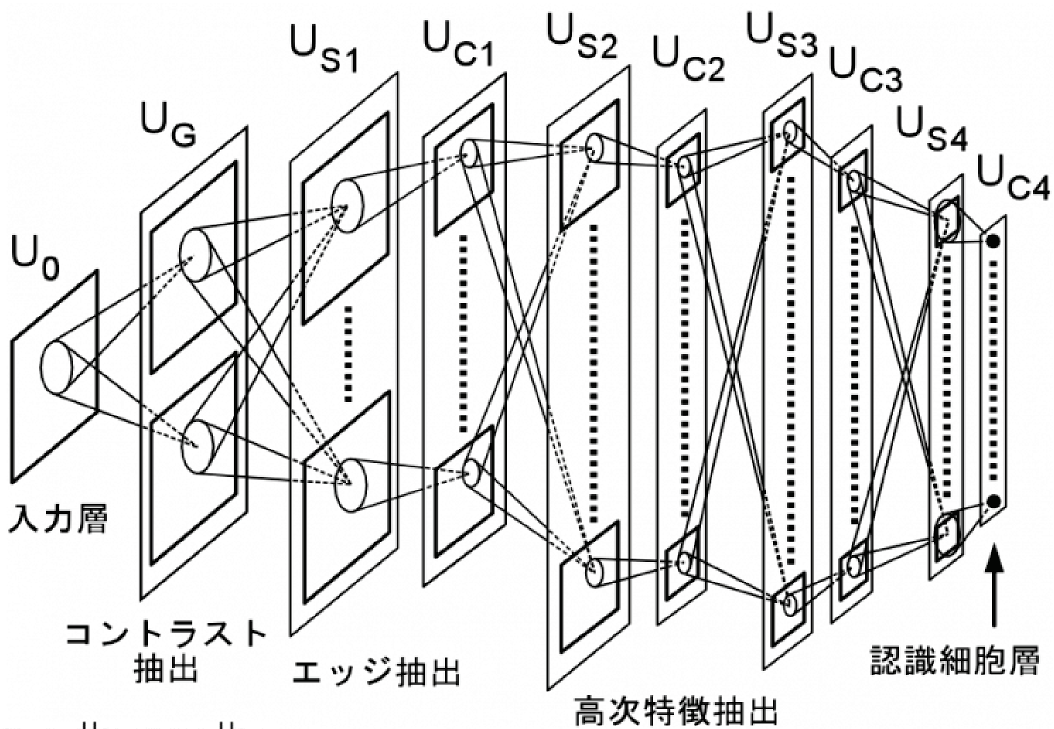
非凸性の問題

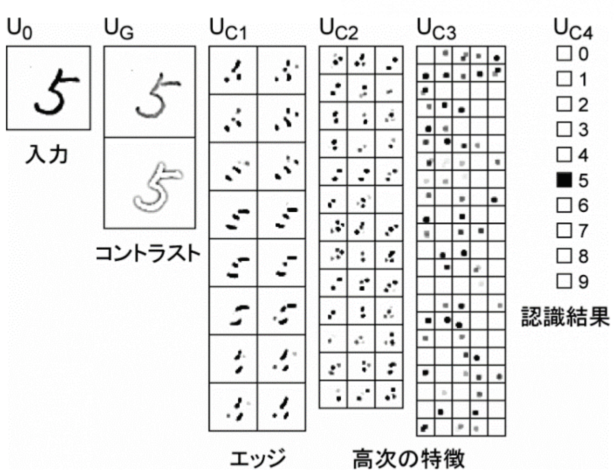
1992: 非線形サポートベクトルマシン
(カーネル法)
1996: スパース学習 (Lasso)
2003: トピックモデル (LDA)
2012: Supervision (Alex-net)
第三次ニューラルネットワークブーム

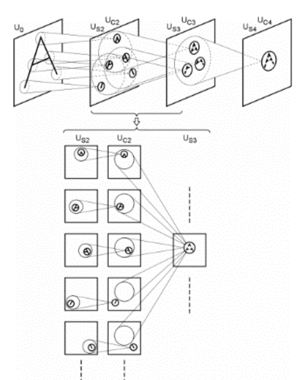
データの増加
＋計算機の強化

ネオコグニトロン

[福島,79]



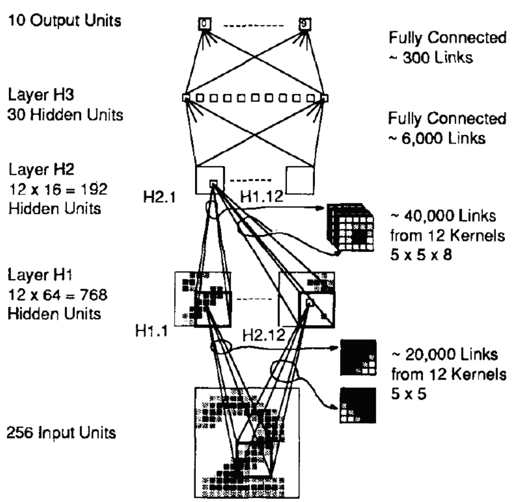




- 人間の脳を模倣
- 畳み込みネットの初期型
- 自己組織型学習
  →素子を足してゆく

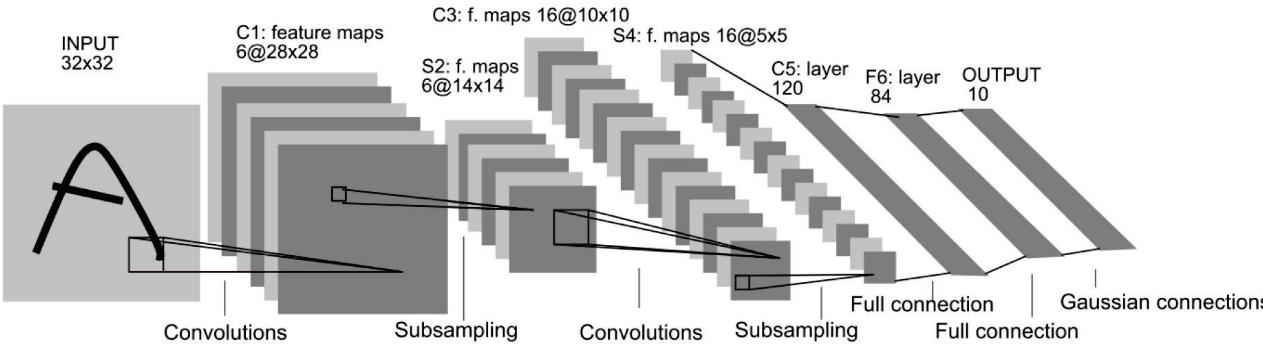
LeNet

[LeCun+etal,89]



LeNet-5

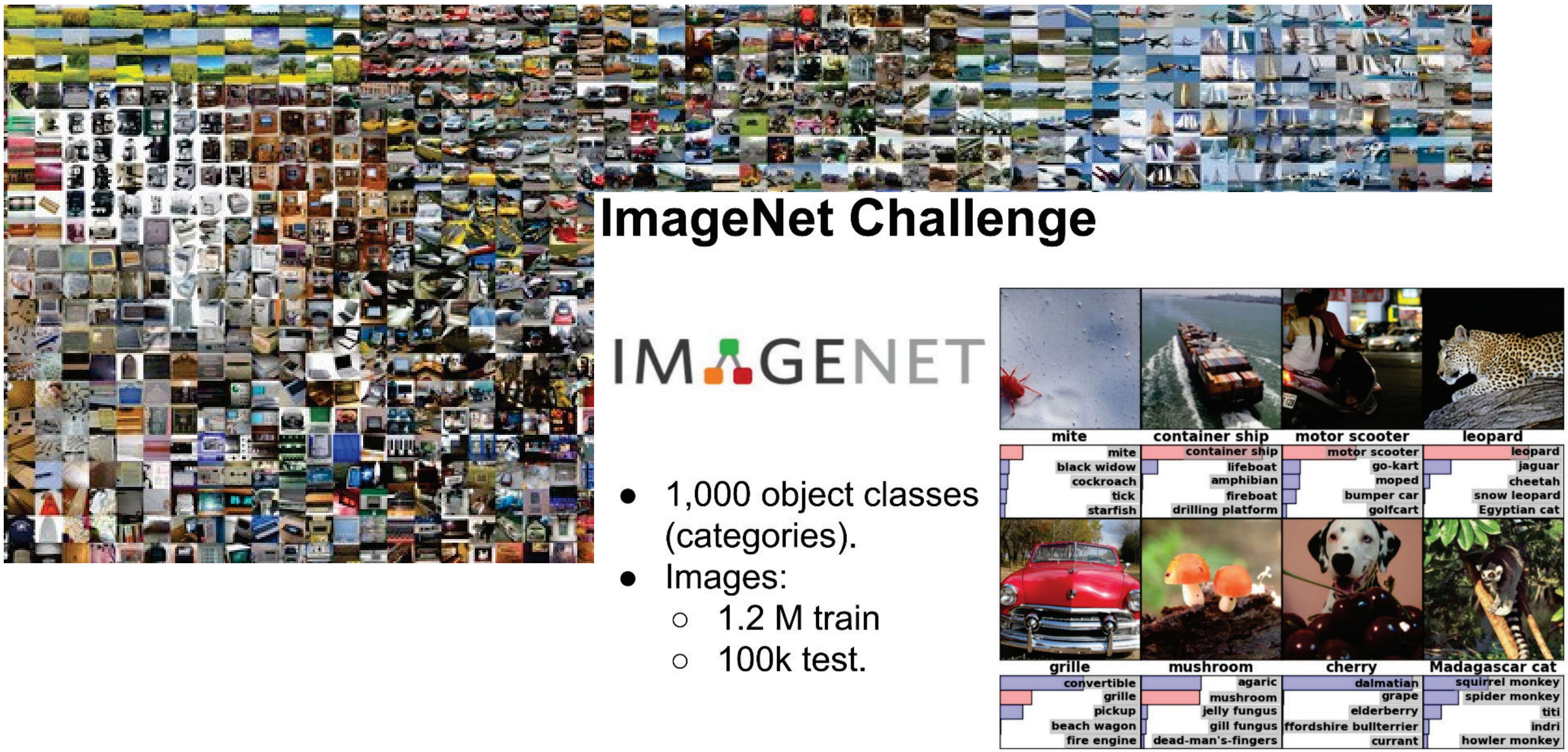
[LeCun etal,98]



- <u>畳み込み</u>＋<u>プーリング</u>：現在も使われている構造
- <u>誤差逆伝搬法</u>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

Deep Learning
深層学習

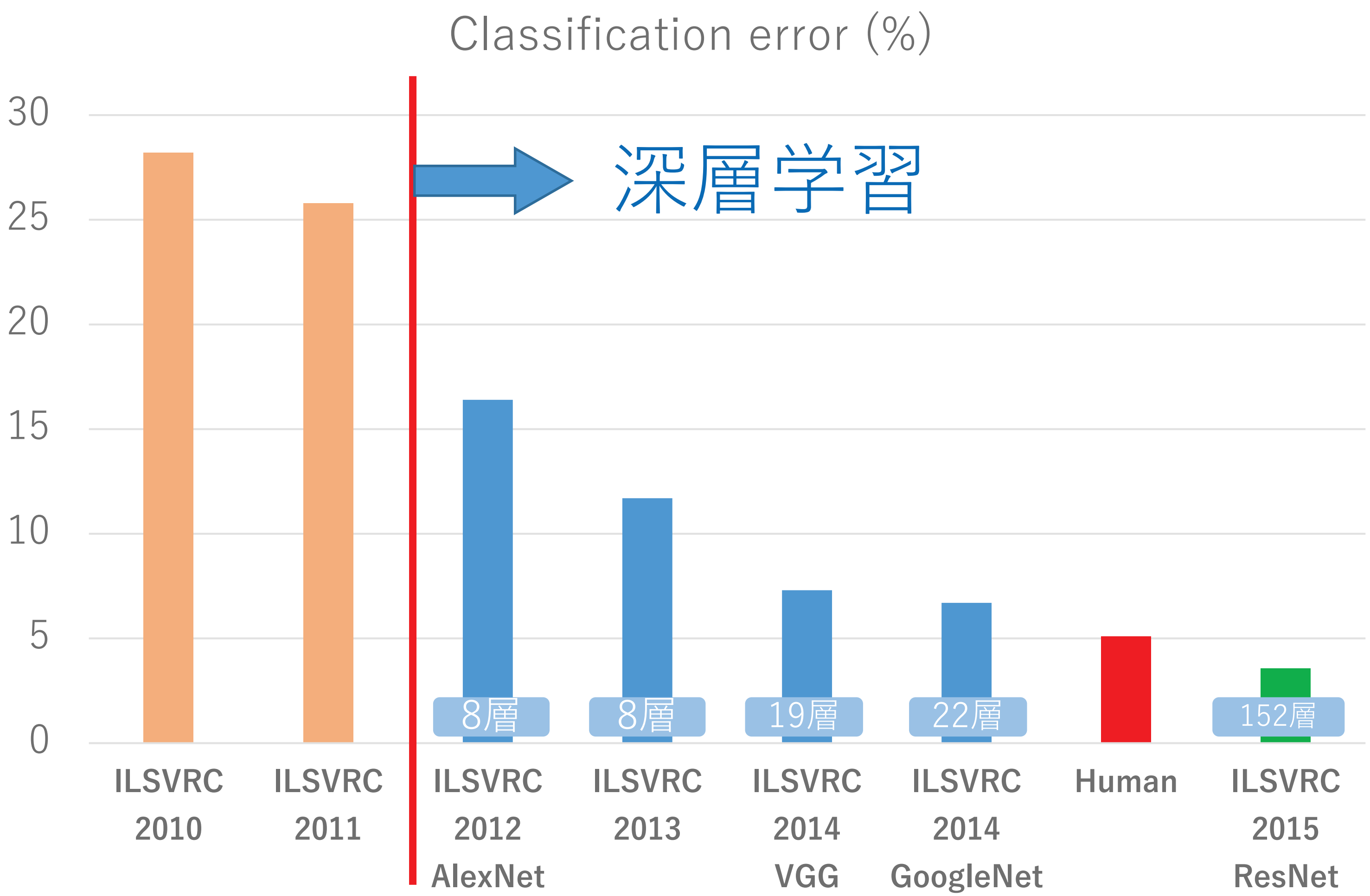
ImageNet



ImageNet: 1,000カテゴリ，約120万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

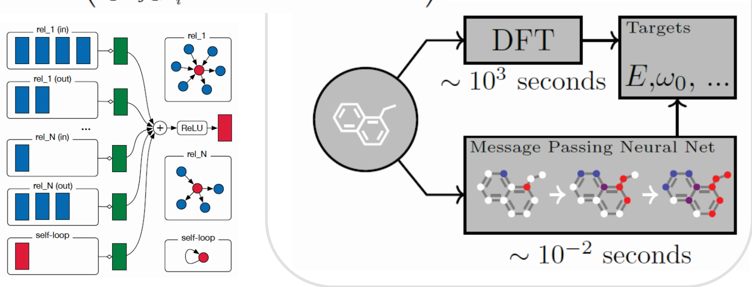
諸分野への波及

ロボット

[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]

量子化学計算，分子の物性予測

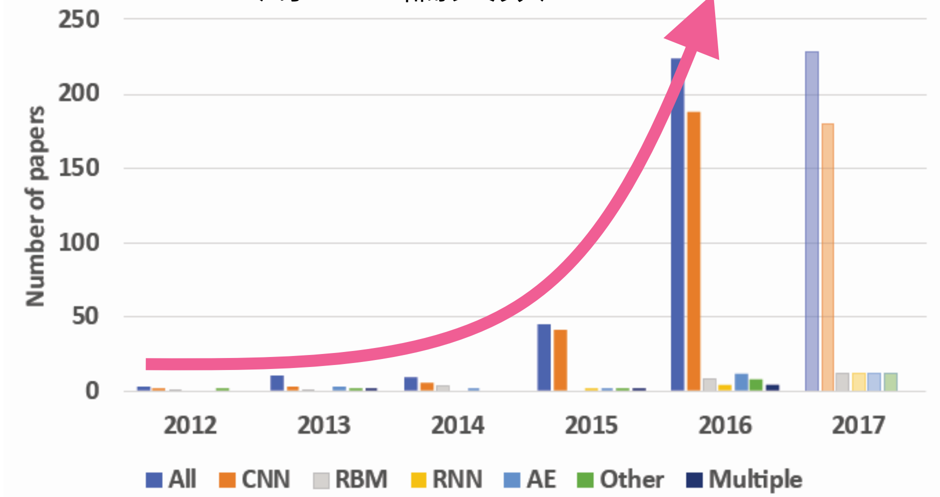
$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right)$$



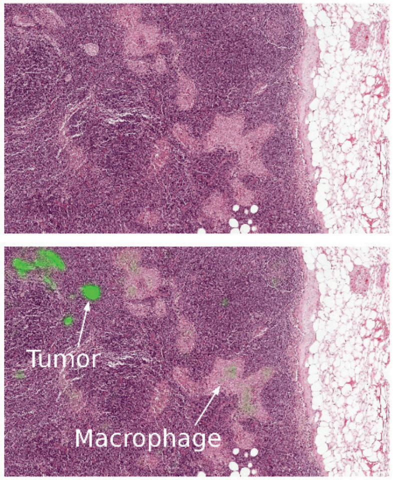
[Niepert, Ahmed&Kutzkov: Learning Convolutional Neural Networks for Graphs, 2016]
[Gilmer et al.: Neural Message Passing for Quantum Chemistry, 2017]
[Faber et al.:Machine learning prediction errors better than DFT accuracy, 2017.]

医療

医療分野における「深層学習」を用いた論文数



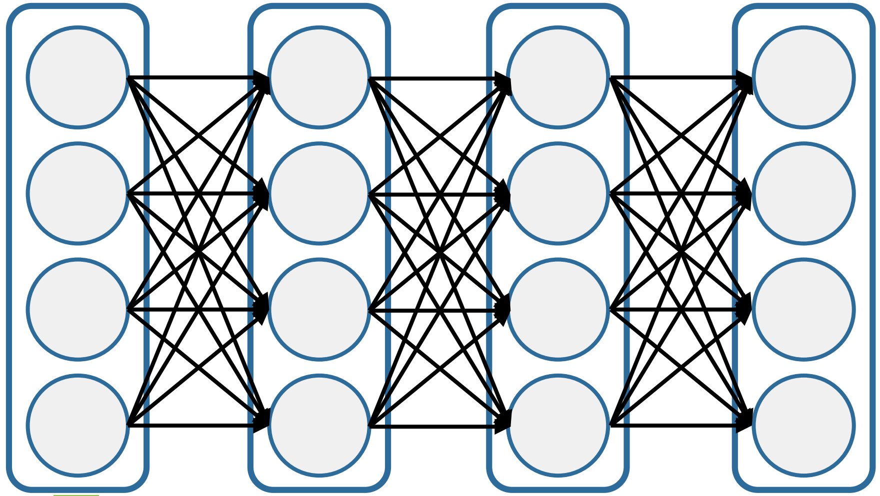
[Litjens, et al. (2017)]



- 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]

深層学習の構造



x W_1 h_1 W_2 h_2 W_3 h_3

基本的に「線形変換」と「非線形活性化関数」の繰り返し．

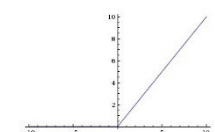
x → $W_1 x$ → $h_1(W_1 x)$ → $W_2 h_1(W_1 x)$ → $h_2(W_2 h_1(W_1 x))$

入力　線形変換　非線形変換　線形変換　非線形変換

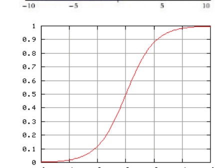
$$h_1(u) = [h_{11}(u_1), h_{12}(u_2), \ldots, h_{1d}(u_d)]^T$$

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)： $h(u) = \max\{u, 0\}$



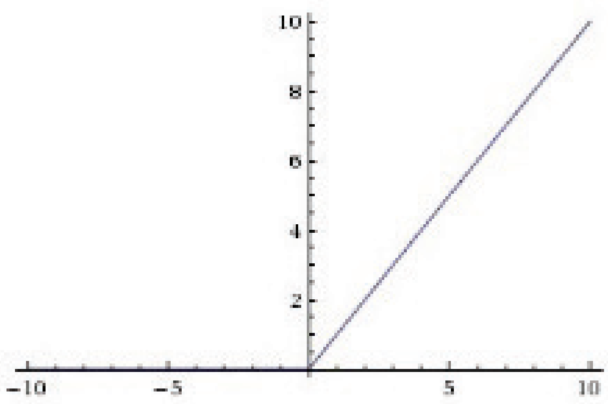
- シグモイド関数： $h(u) = \dfrac{1}{1 + e^{-u}}$



活性化関数の例

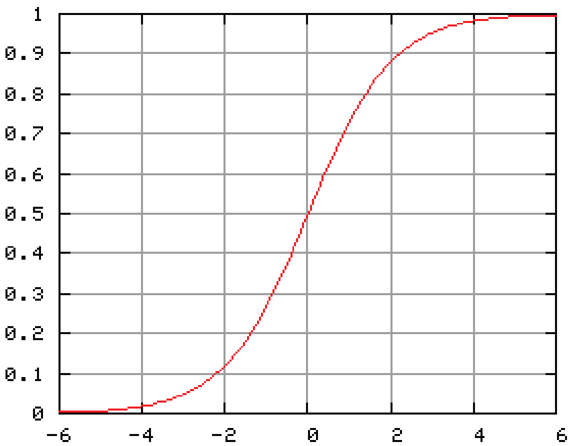
☆ReLU (Rectified Linear Unit)

$$h(u) = \max\{u, 0\}$$



シグモイド関数

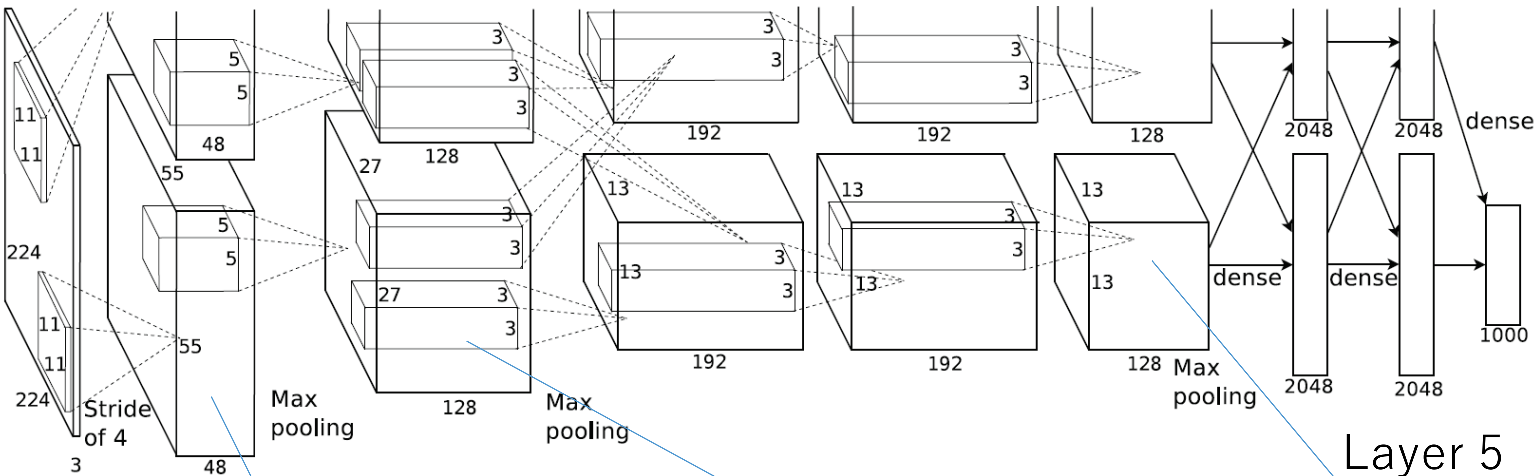
$$h(u) = \frac{1}{1 + e^{-u}}$$

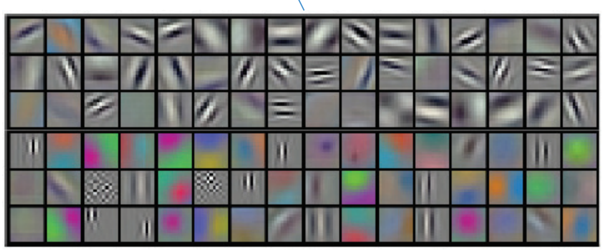


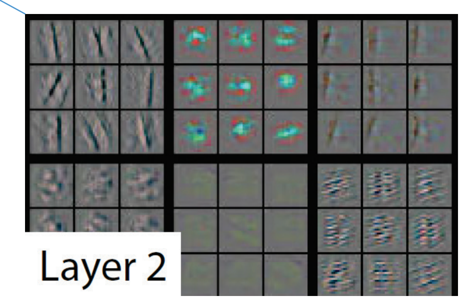
深層NNの構造
（主に畳み込みネット）

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

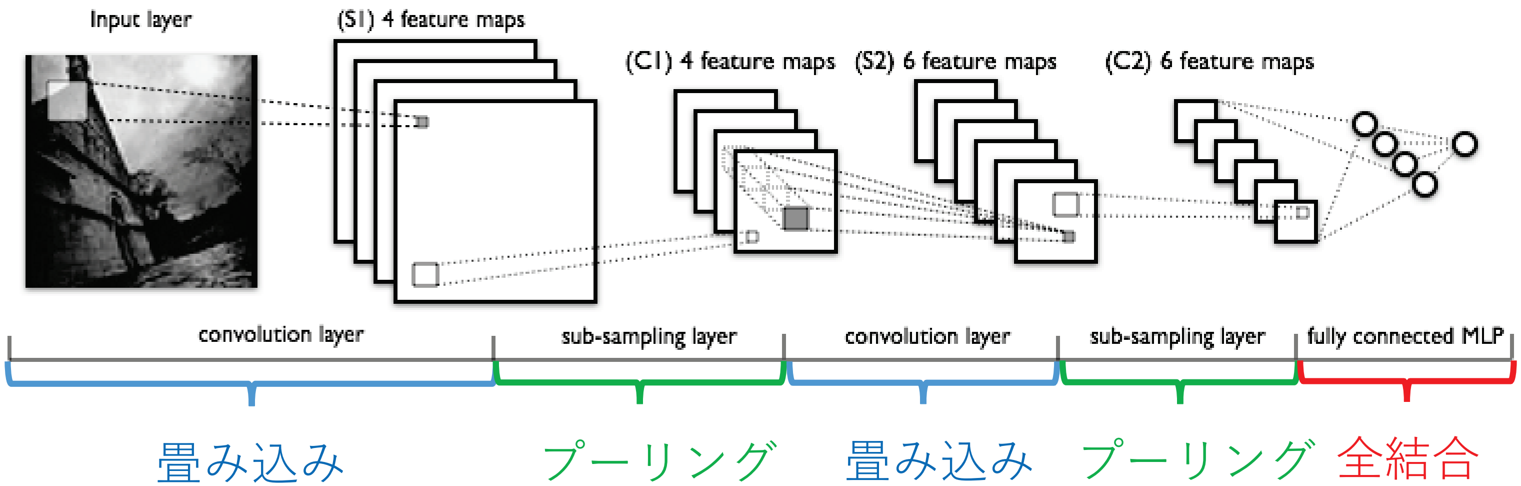
人の顔

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

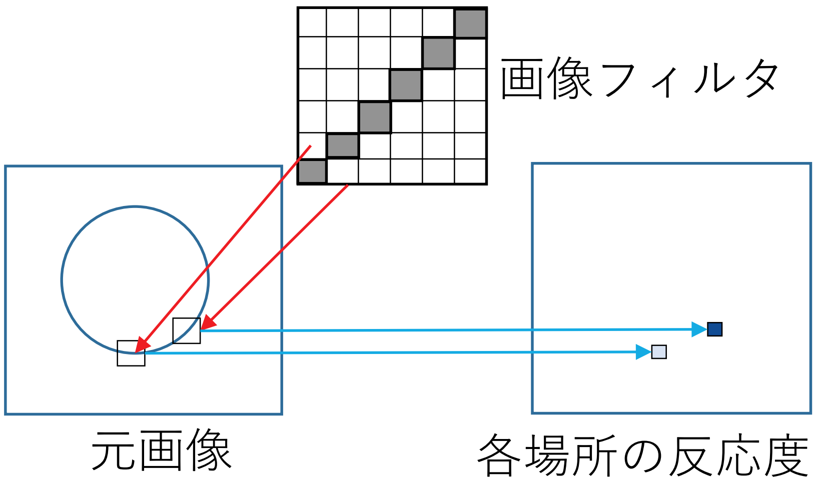
畳み込みニューラルネット (Convolutional Neural Network, CNN)

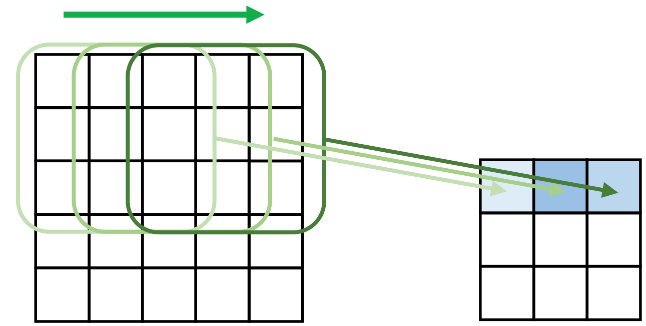


- 畳み込みとプーリングを交互に積み重ねる
- 最後に全結合層を重ねる.
  - 畳み込み (Convolution)：パターンの抽出
  - プーリング (Pooling)：移動不変性を獲得
  - 全結合 (Fully-connected)：最終的な判別器を構成

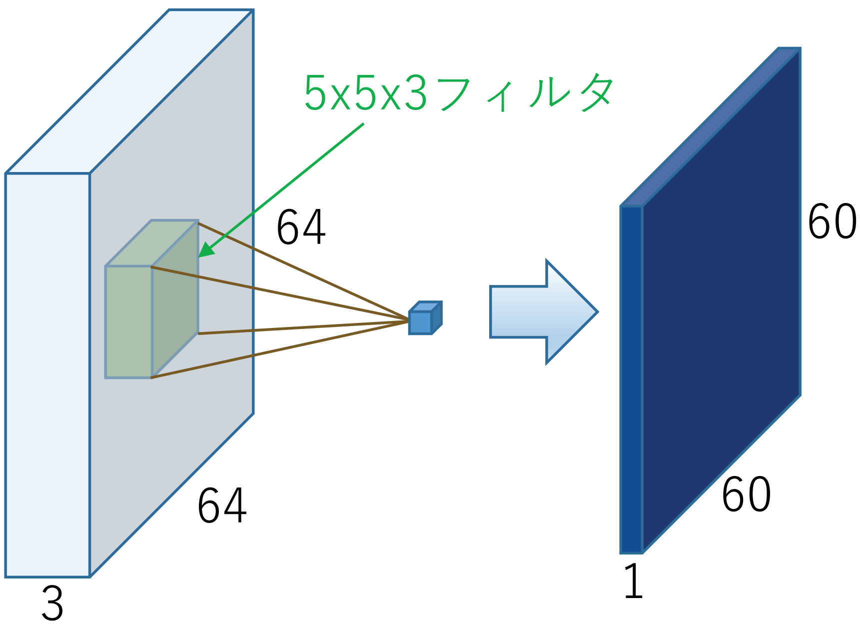
畳み込み層

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

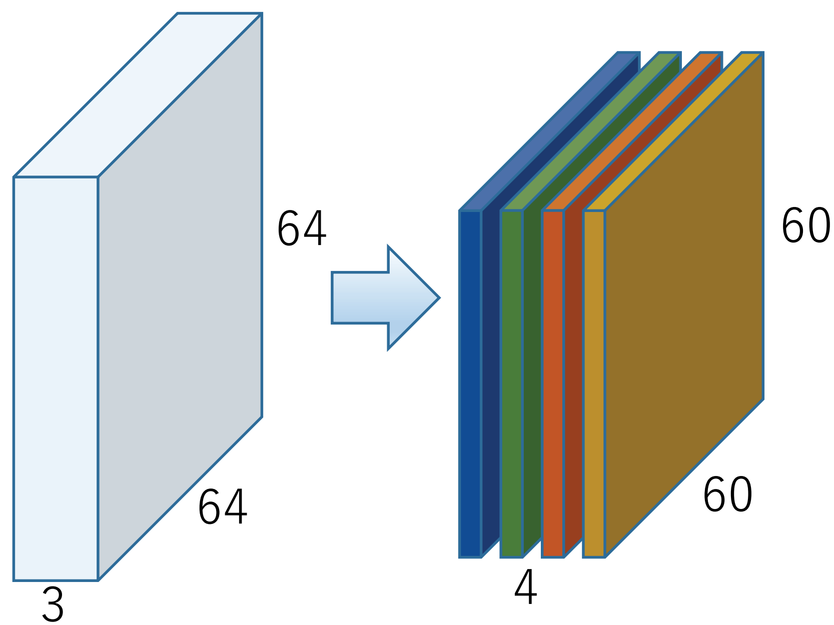




- 画像フィルタをずらしながら畳み込む.
- 複数のフィルタを用意して特徴量を構成.



- 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

0-パディング

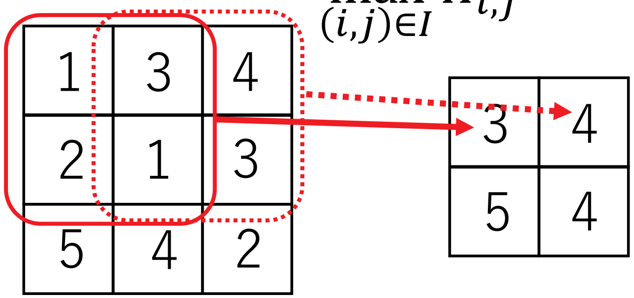
- 畳み込むと画像のサイズが小さくなってしまう.
- 周辺に0を埋めてサイズが変わらないようにする.

0	0	0	0	0	0	0	0
0	35	25	41	23	31	34	0
0	26	24	23	84	14	65	0
0	24	45	42	6	7	3	0
0	34	32	17	98	5	4	0
0	36	21	56	67	56	23	0
0	41	51	45	81	92	17	0
0	0	0	0	0	0	0	0

プーリング層

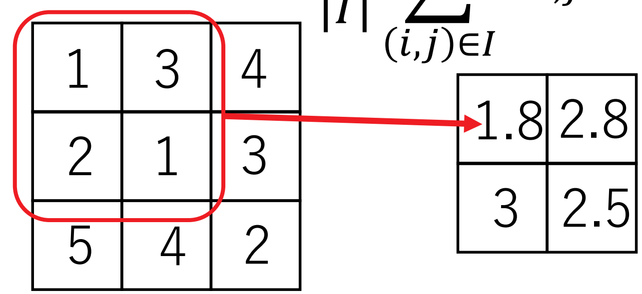
- ある特徴量に選択的に発火している箇所がその領域にあるか検出.
- 少しのずれを吸収する作用→移動不変性

$$\max_{(i,j)\in I} X_{i,j}$$

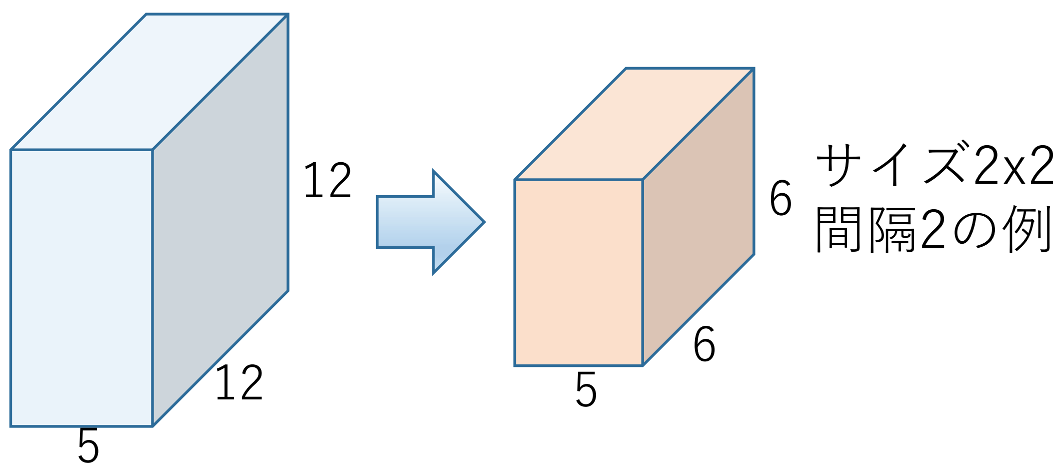


Max-プーリング

$$\frac{1}{|I|}\sum_{(i,j)\in I} X_{i,j}$$



Average-プーリング



深層NNの学習

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）　$(y, y' \in \mathrm{R})$

$\ell(y, y') = -\sum_{k=1}^{K} y_k \log(y'_k)$ 　Cross-entropy損失（多値判別）
$(y_k \in \{0,1\}, y'_k \in [0,1]$, ともに和が1)

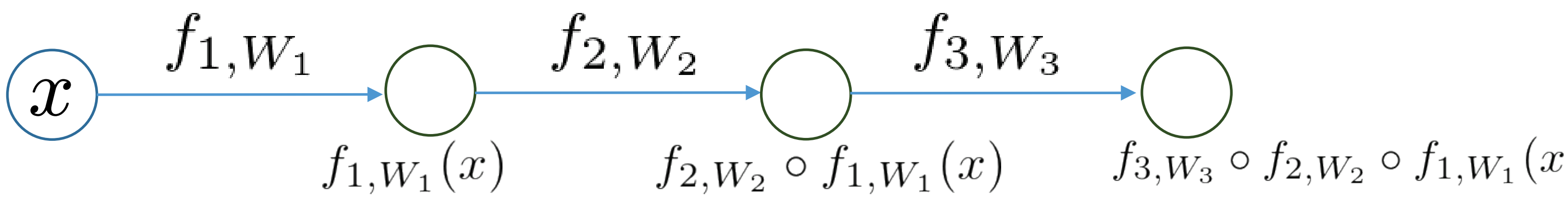
$$\min_{W} L(W)$$

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

- 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



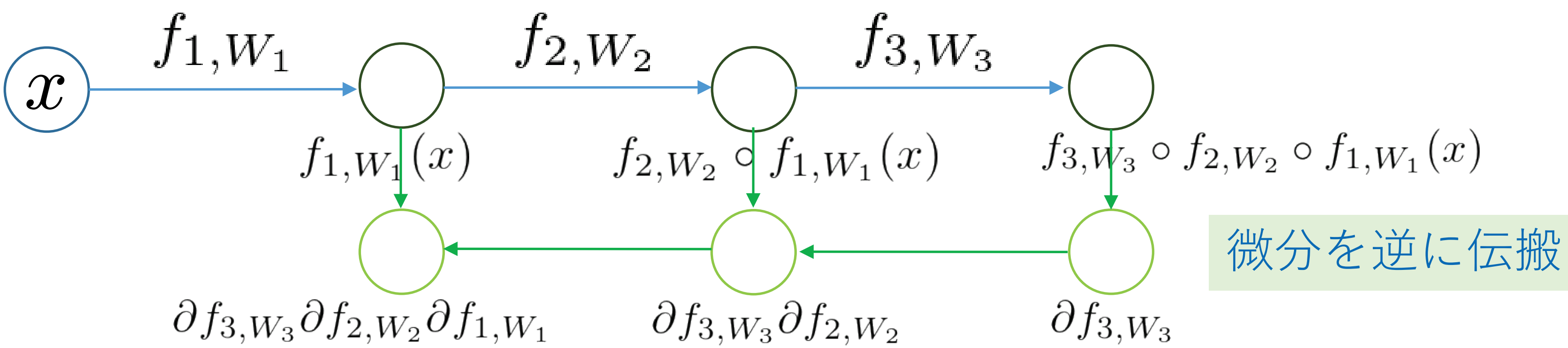
例： $f_{1,W_1}(x) = h(W_1 x)$

合成関数

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

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

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

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

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

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

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

大きな問題を分割して個別に処理

普通の勾配降下法：

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

確率的勾配降下法 (SGD)

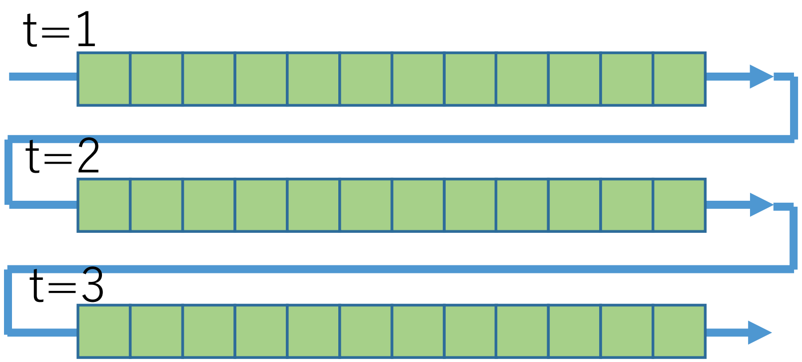
(Stochastic Gradient Descent)

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

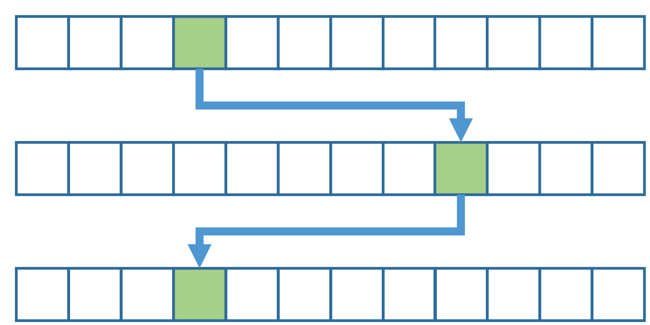
普通の勾配降下法：



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

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

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

大きな問題を分割して個別に処理

- 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：

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

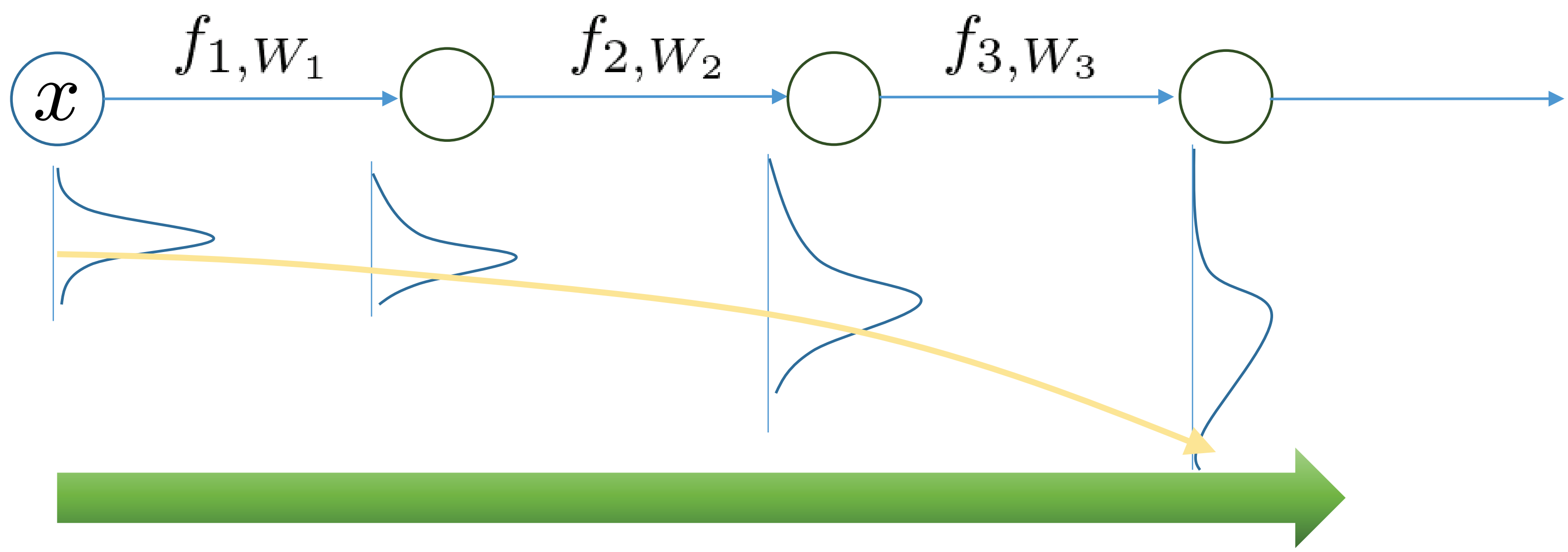
← O(1)の計算量

- 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.
- 鞍点を抜けやすいという副産物もある.

バッチ正規化 (batch normalization)



- 深くなるにつれ，平均と分散が発散or縮小してゆき，学習が安定しない.
- バッチ正規化はこの揺れによる自由度を解消し，学習を安定化させる.

※ReLUを用いている限りスケール不変なのでスケールは固定したほうが良い.

各座標ごとに
平均と分散を正規化

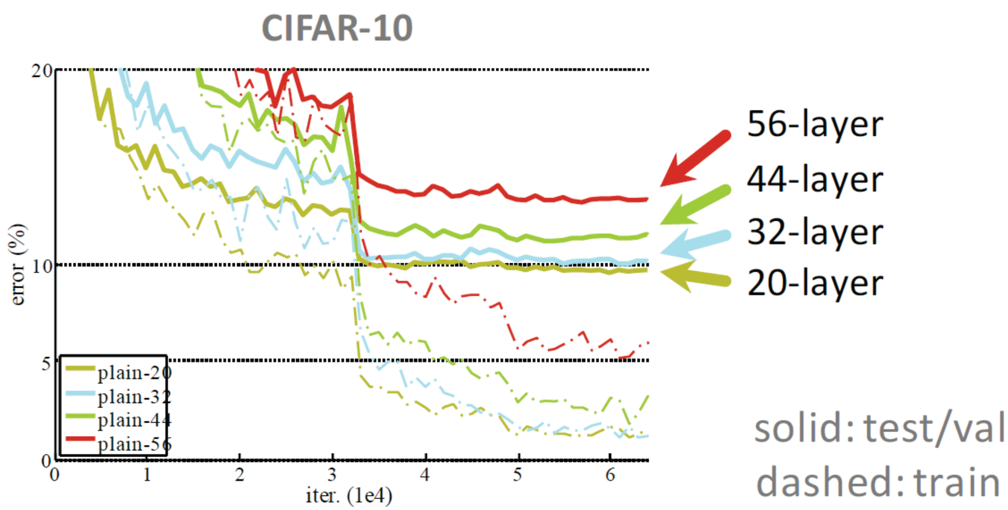
$$\hat{x} \leftarrow \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

μ_B: ミニバッチ内の平均，σ_B^2: ミニバッチ内の分散

ResNet

深ければ良い？

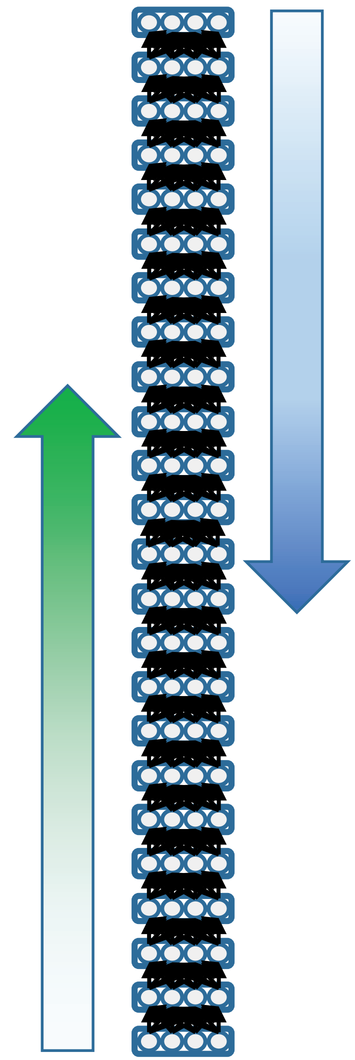
A ：必ずしもそうではない.



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

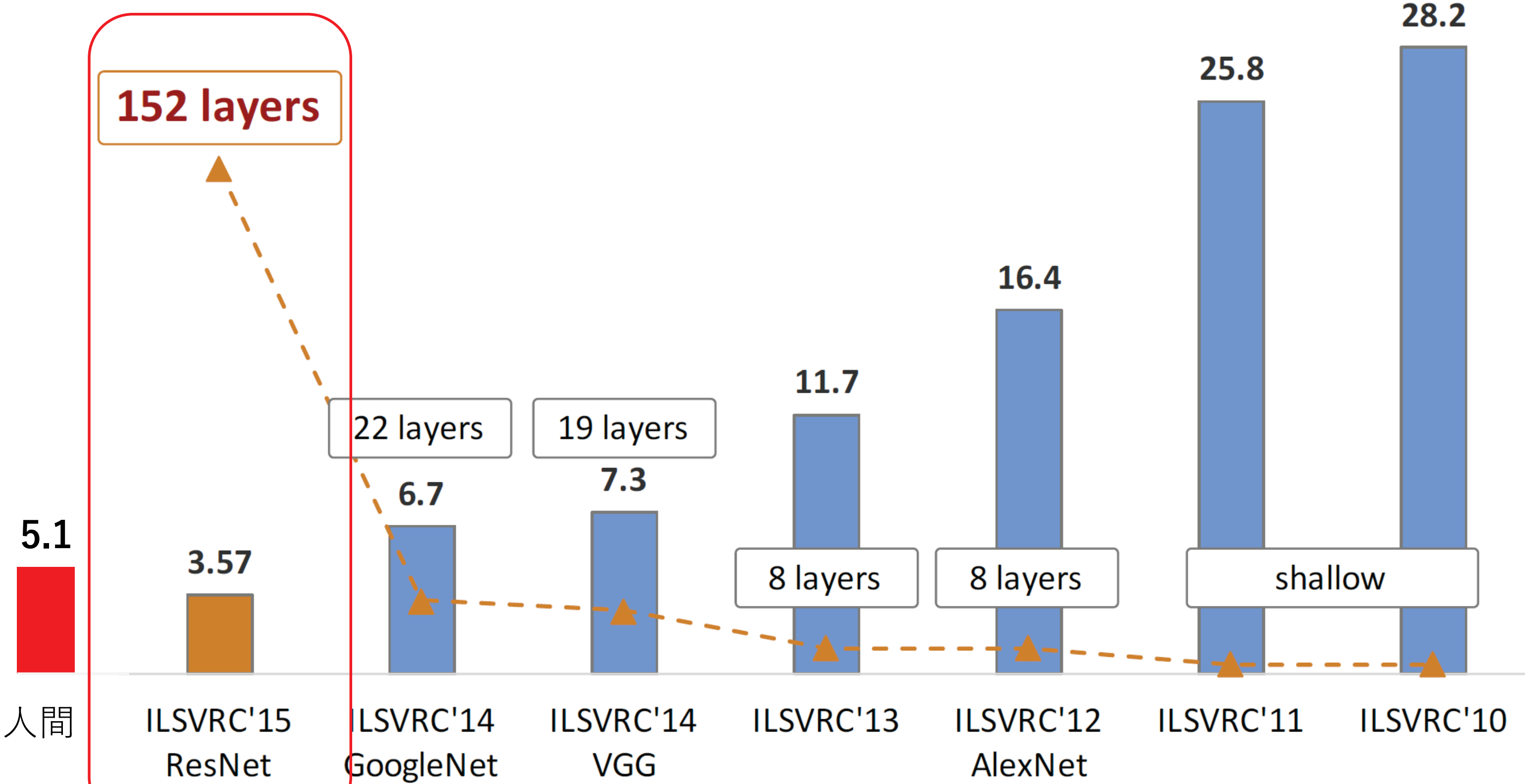
He. “Deep Residual Network”. ICML2016 tutorial.

誤差の情報が伝わらない

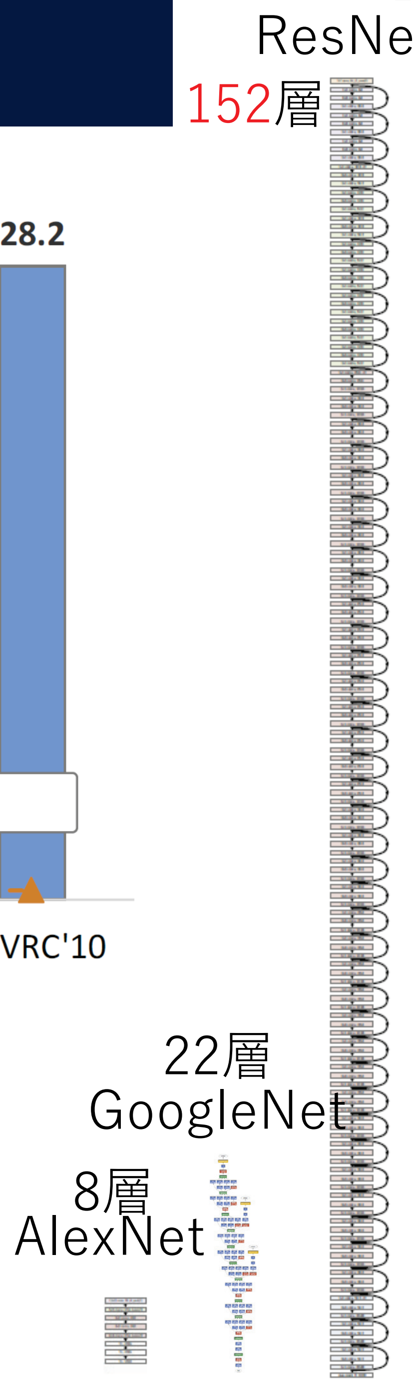


入力の情報が伝わらない

ResNet (Deep Residual Net)



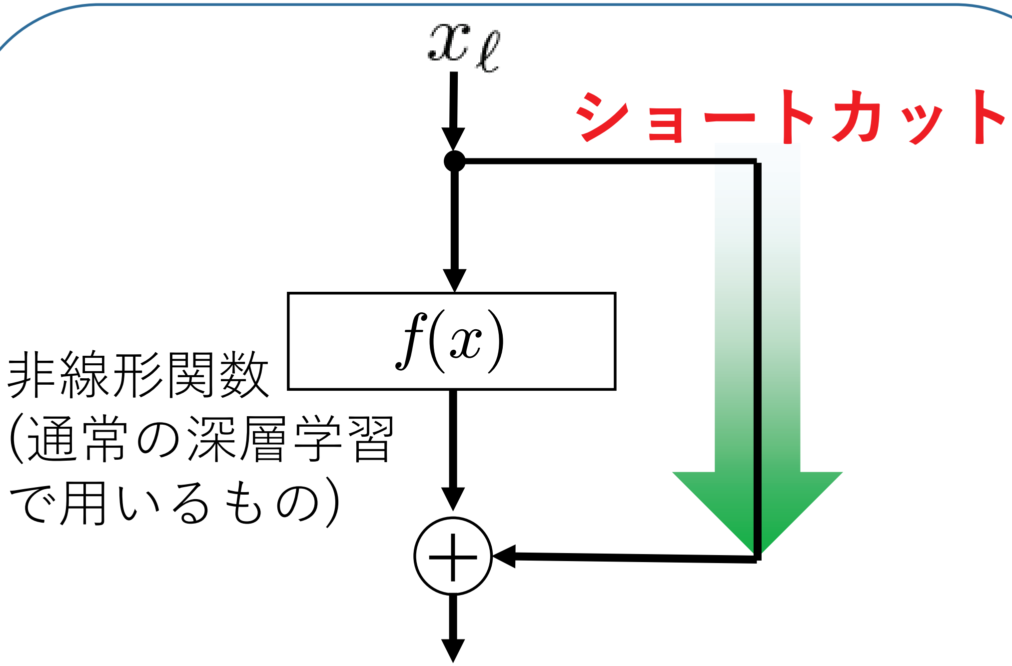
ImageNet Classification top-5 error (%)



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



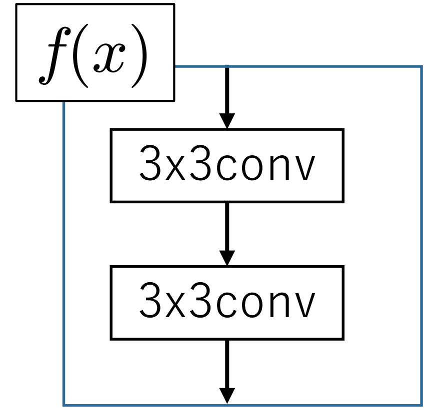
$$x_{\ell+1} = x_\ell + f(x_\ell)$$

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

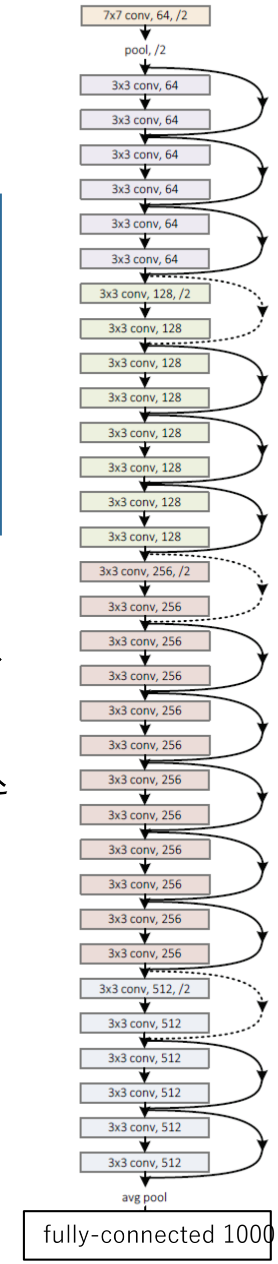
$$\vdots$$

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

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.

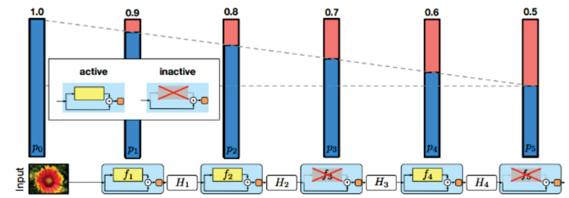
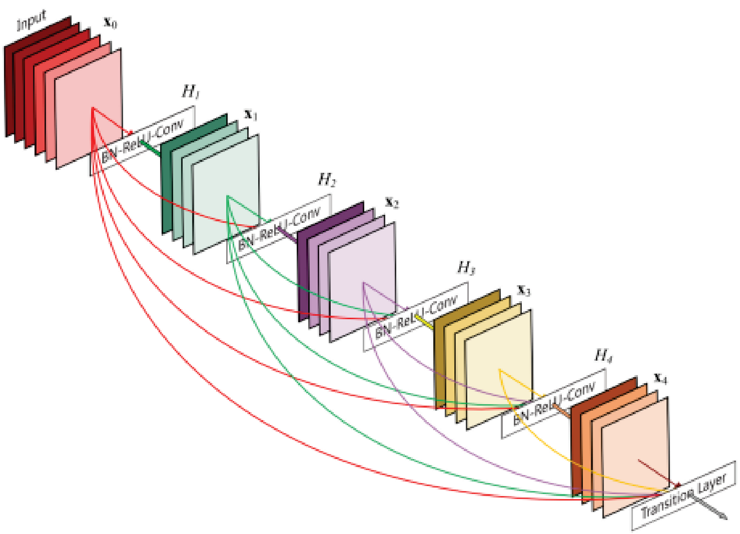


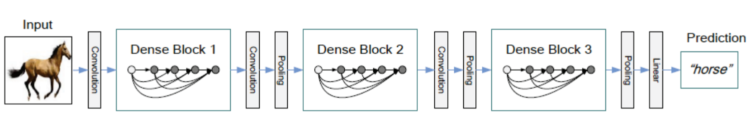
Fig. 2. The linear decay of p_ℓ illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is always active.

- 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

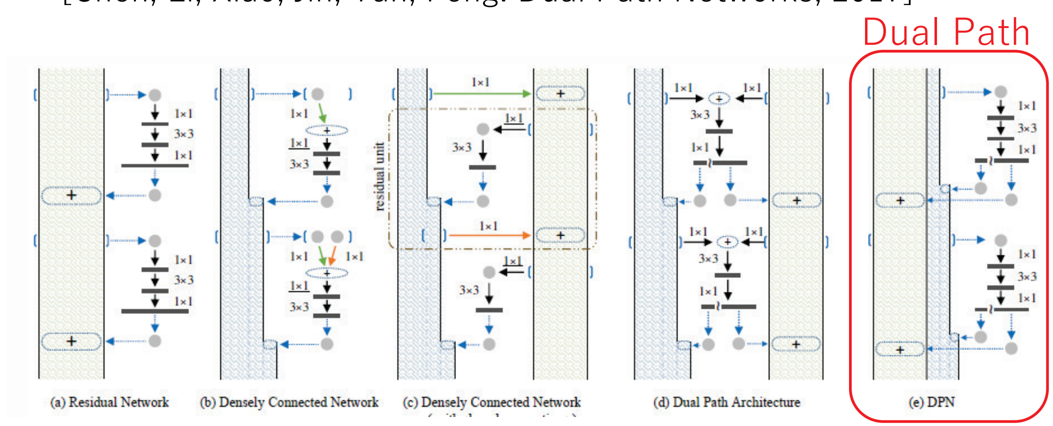




DenseNetの様子

- Dual Path Networks

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



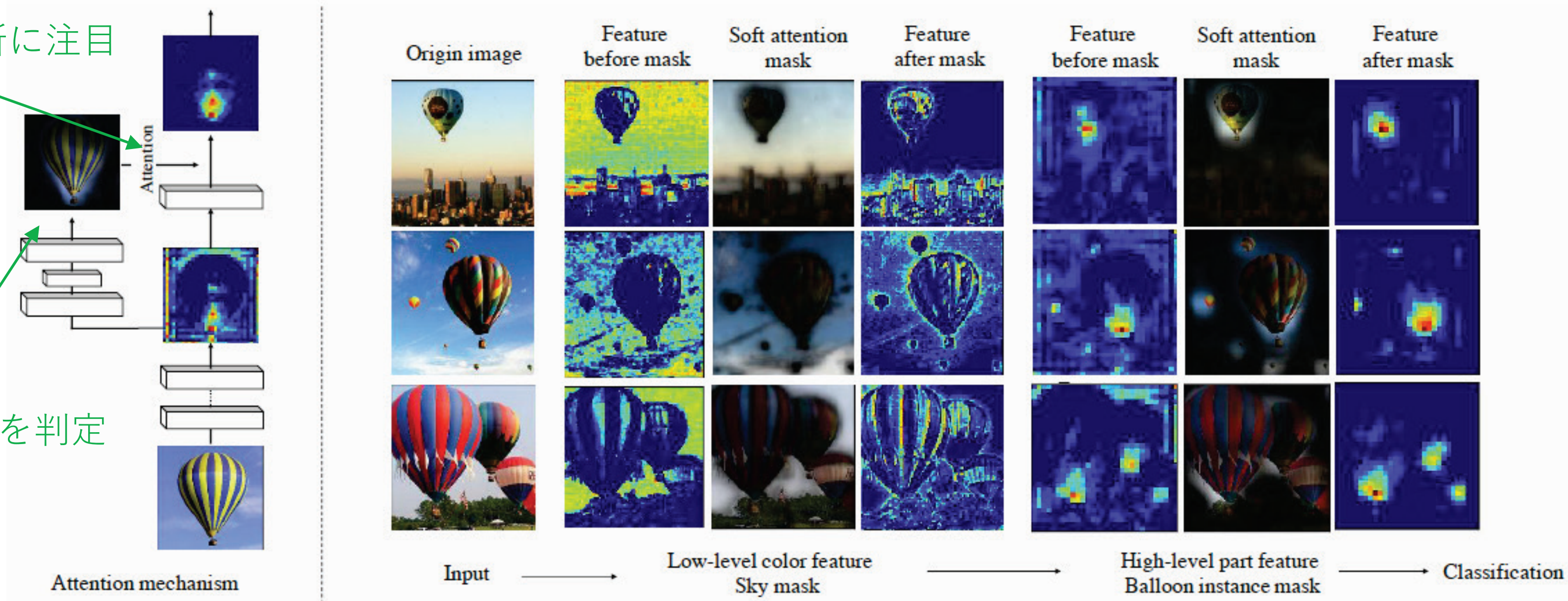
ResNetとDenseNetの良い部分を組み合わせ.
ILSVRC2017のObject localization部門で１位.

Residual Attention Network

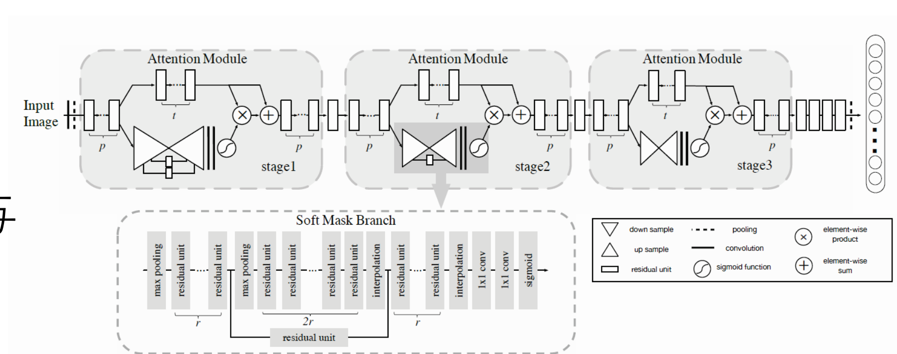
ILSVRC2017のObject detection部門１位

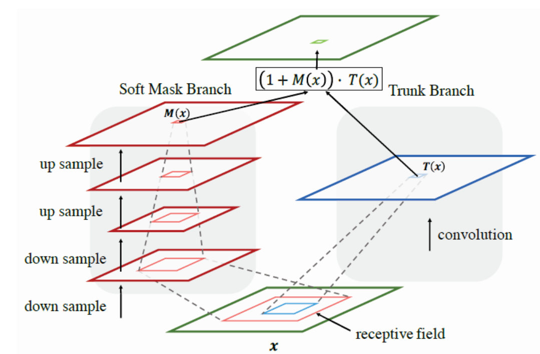
重要そうな場所に注目

重要そうな場所を判定



ResNetに選択的
注意の機構を付与



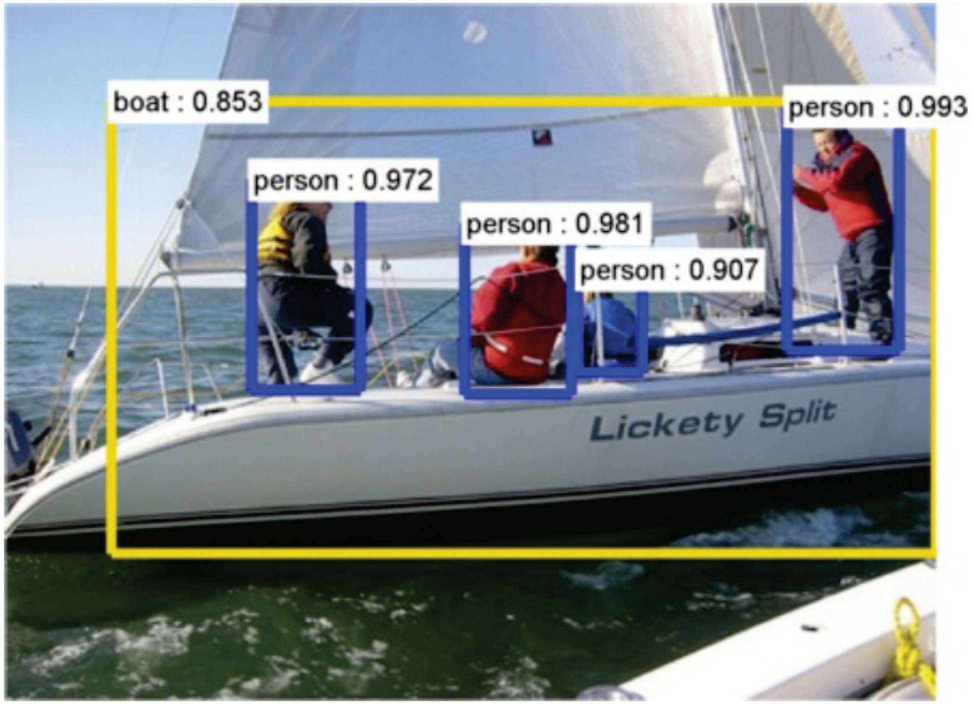


[Wang F, Jiang M, Qian C, et al. Residual Attention Network for Image Classification. arXiv:1704.06904, 2017]

種々の応用

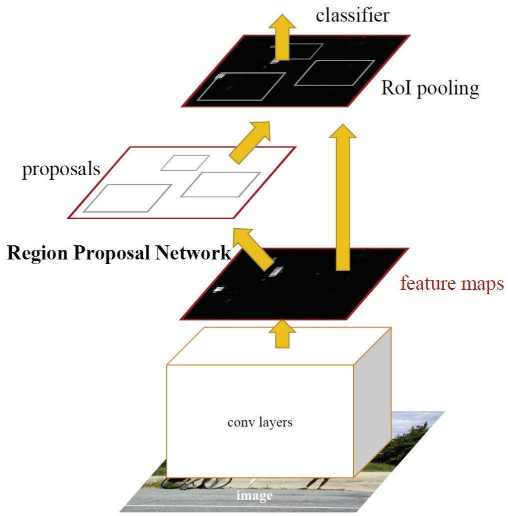
物体認識→物体検出

Faster R-CNN (2015)



「どこに」＋「なにが」
写っているか

物体認識より難しい.
物体認識：「なにが」のみ



	mAP(%)
ResNet	**48.4**
VGG	41.5

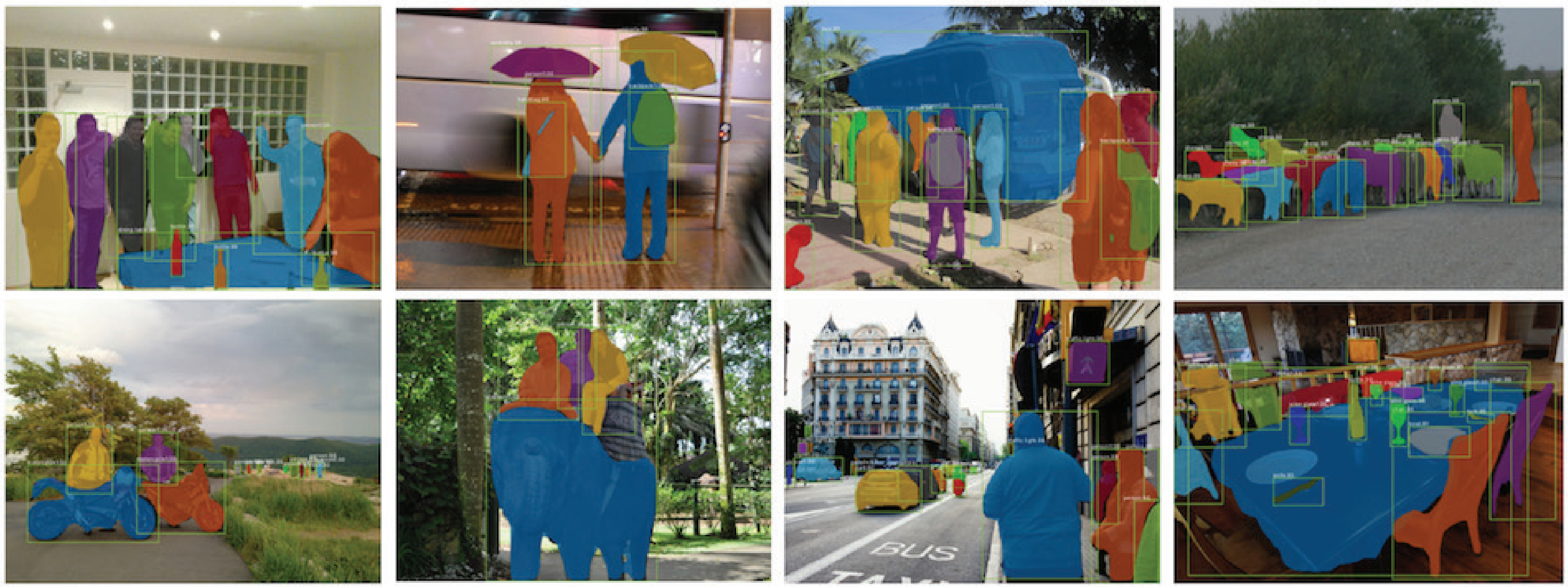
(従来)

COCO 2015 dataset

Ren, He, Girshick, & Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. NIPS 2015.

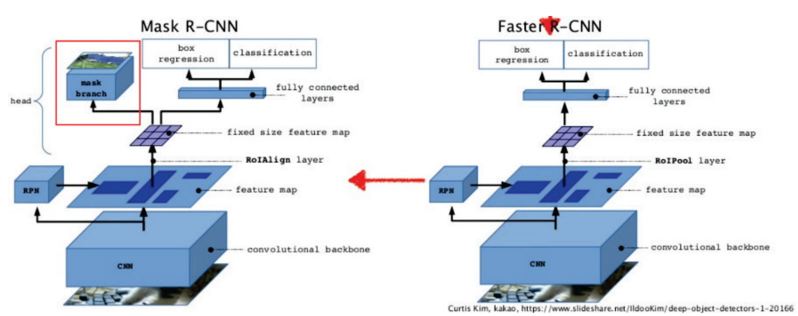
物体検出→マスキング

Mask R-CNN [He, Gkioxari, Dollár, Girshick, ICCV2017]
https://arxiv.org/abs/1703.06870



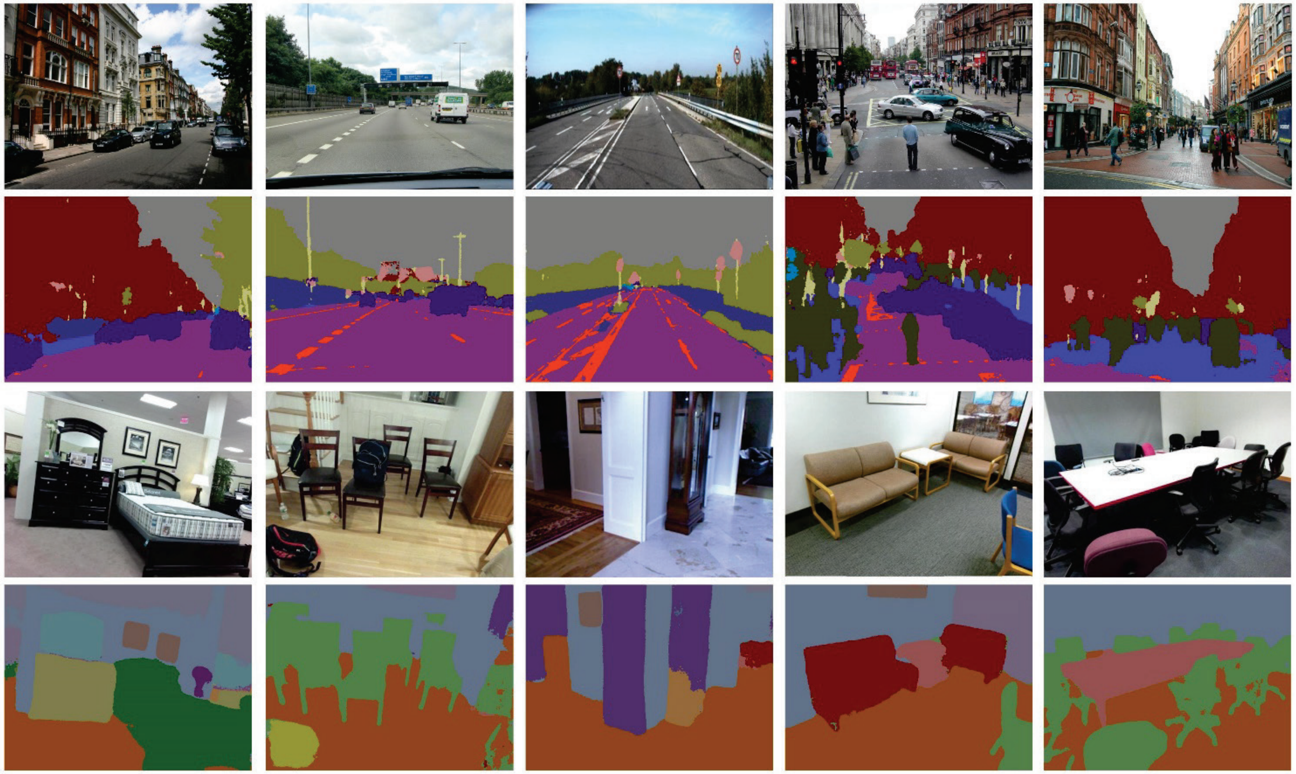
四角で囲むだけでなくマスキングも実行

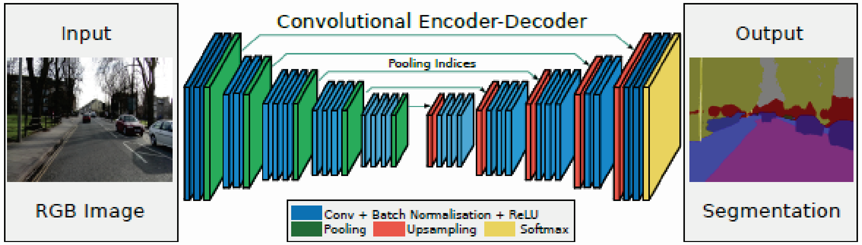
人物姿勢推定も可能



(動画も紹介)

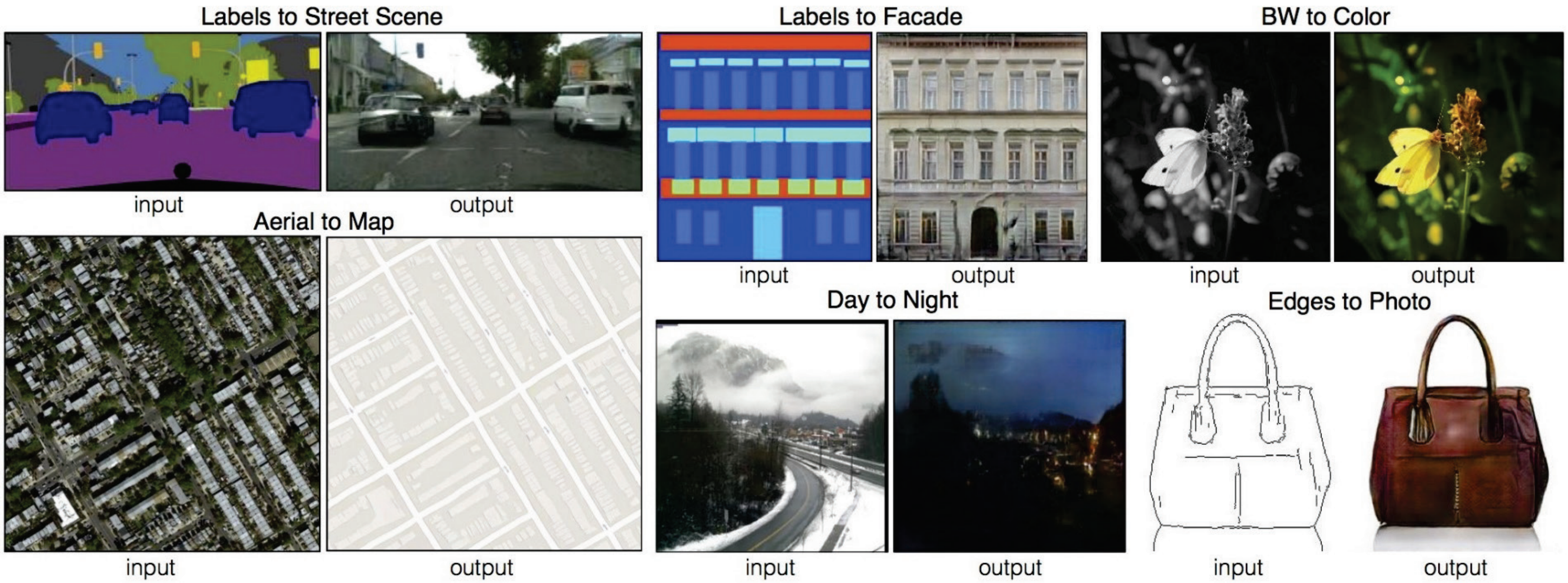
SegNet





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

Pix2Pix



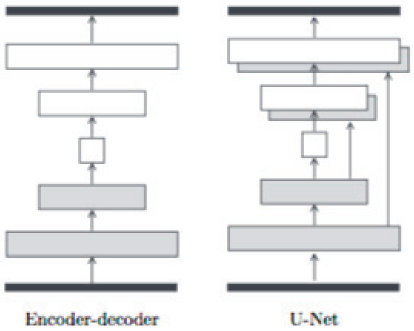
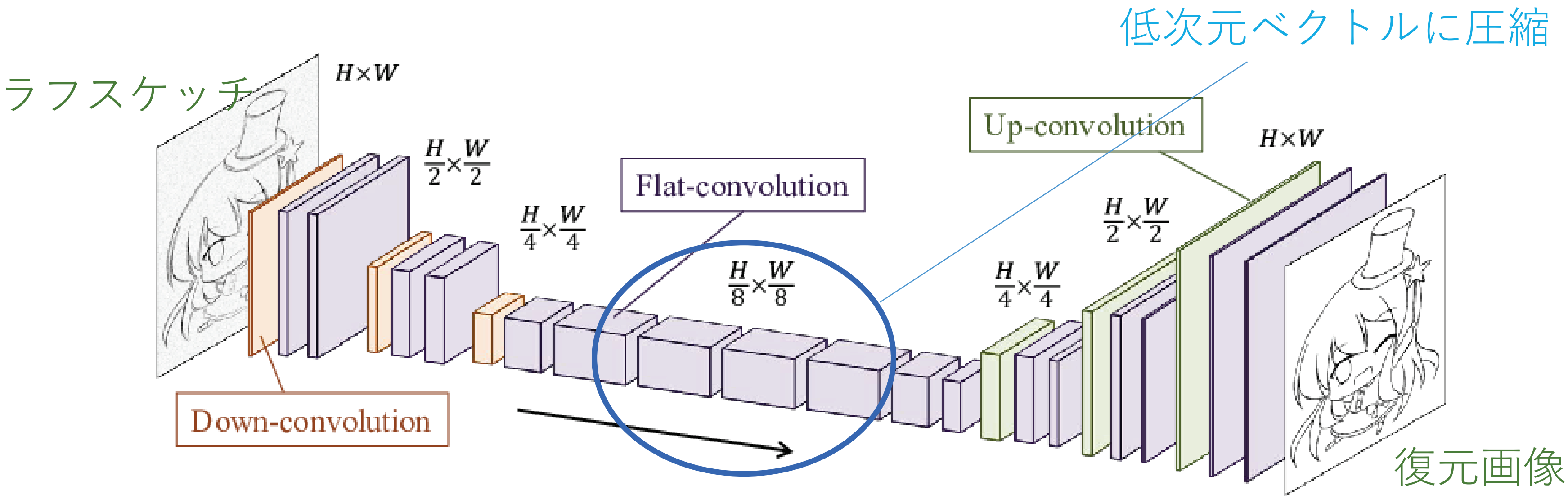


Figure 3: Two choices for the architecture of the generator. The "U-Net" [34] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

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



(c) Masks (d) Book (e) Standing girl

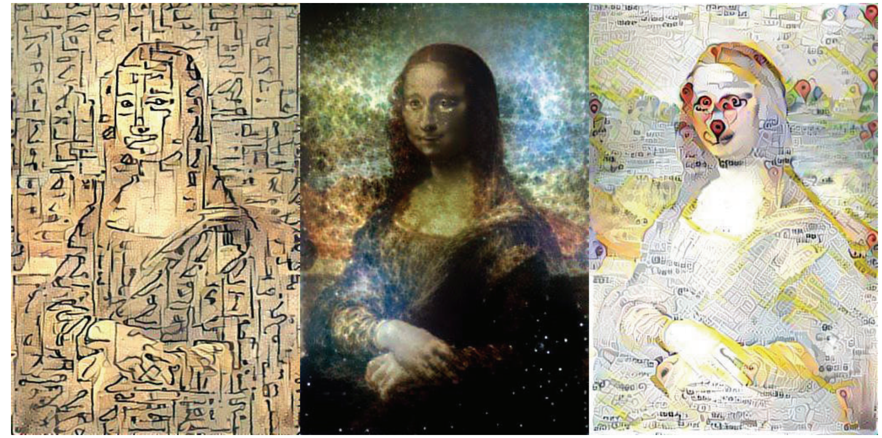
[Simo-Serra et al., SIGGRAPH2016]

スタイル変換

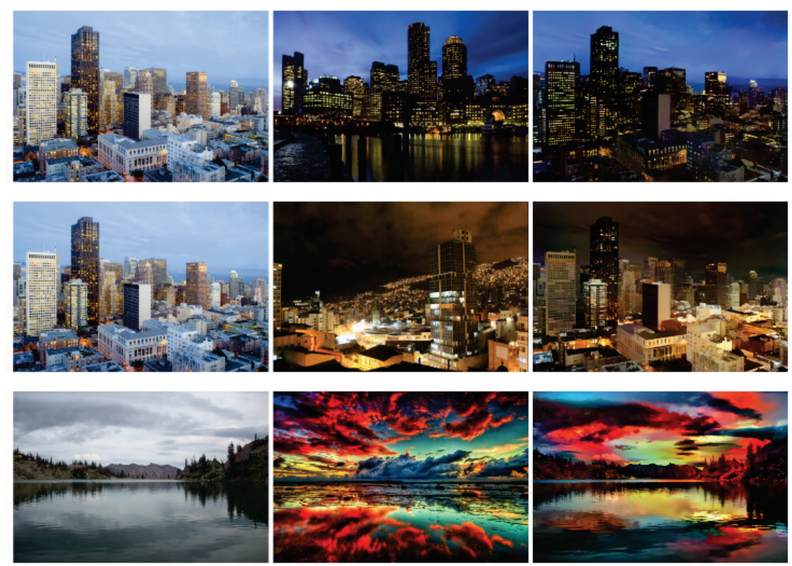
[Gatys, Ecker, Bethge : Image Style Transfer Using Convolutional Neural Networks, CVPR2016]

元画像

[Gene Kogan: Experiments with style transfer, 2015]



http://genekogan.com/works/style-transfer/



[Luan, Paris, Shechtman, Bala: Deep Photo Style Transfer, 2017] https://arxiv.org/abs/1703.07511

画像診断への応用

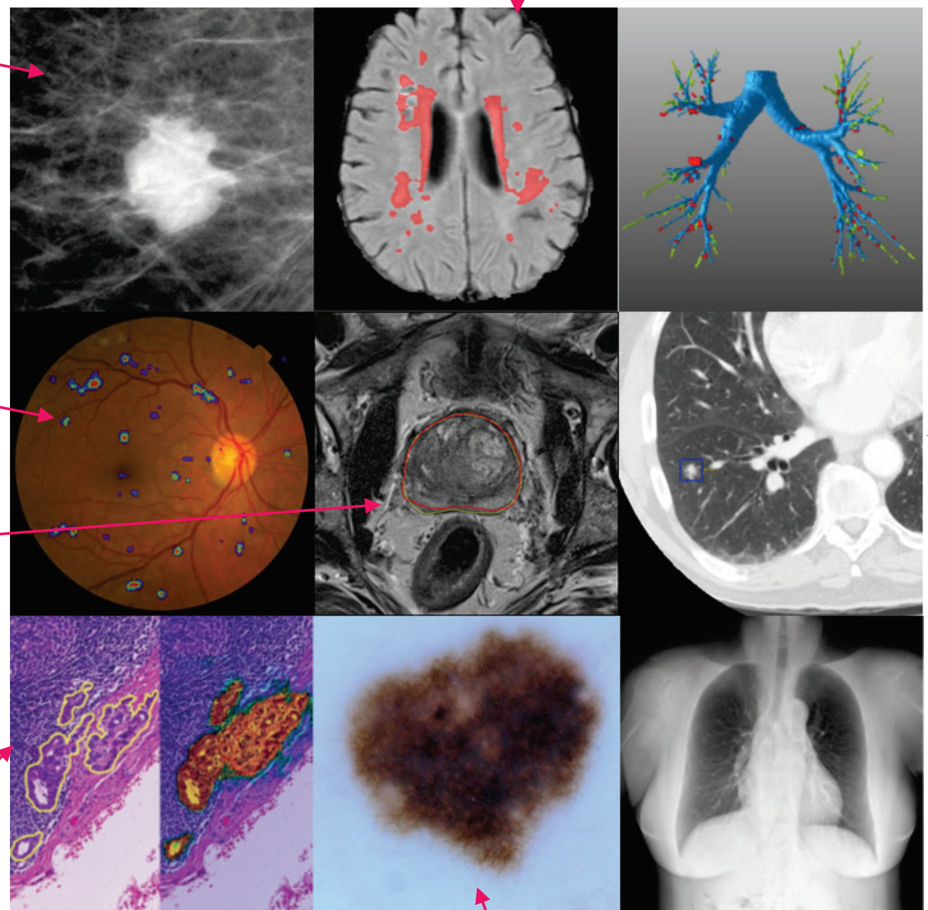
マンモグラム分類
[Kooi et al., 2016]

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]

前立腺セグメンテーション
[PROMISE12 challenge]



小結節分類
[LUNA16 challenge]

肺癌転移検出
[CAMELYON16]

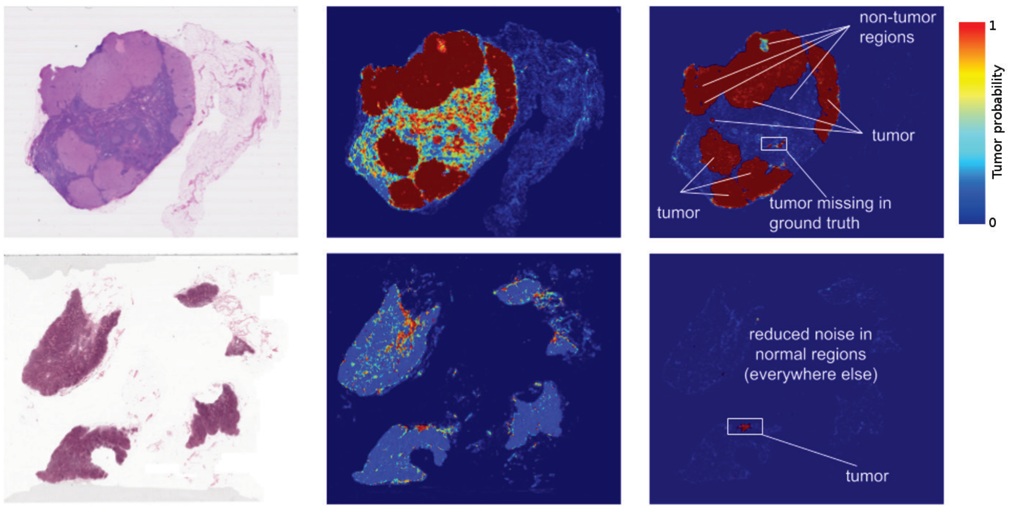
胸部骨減弱処理
[Yang et al., 2016]

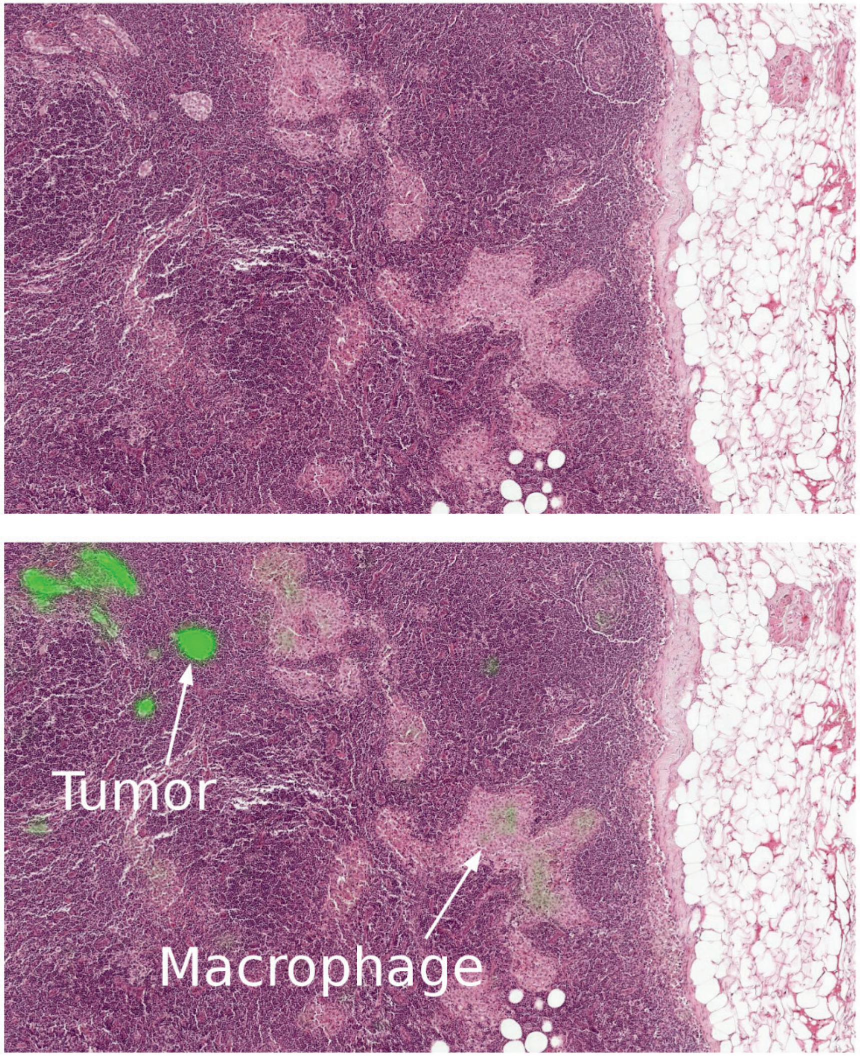
皮膚損傷分類
[Esteva et al., 2017]

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

[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

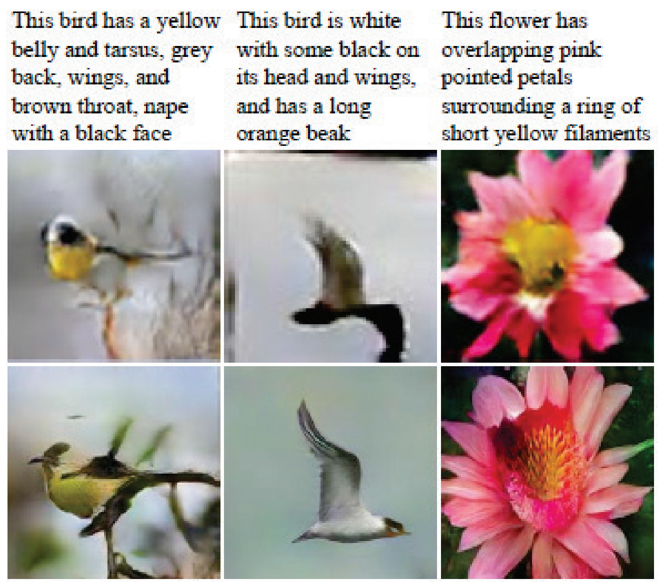
本物らしいデータを生成したい

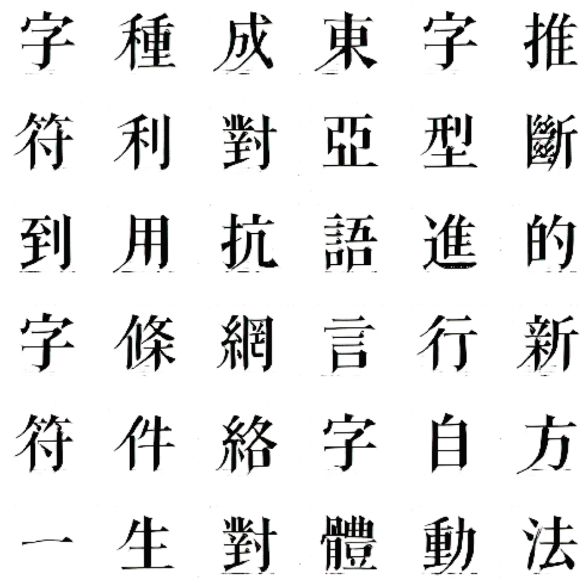
深層学習が生成した画像

生成データ　訓練データ

(a) Stage-I images

(b) Stage-II images



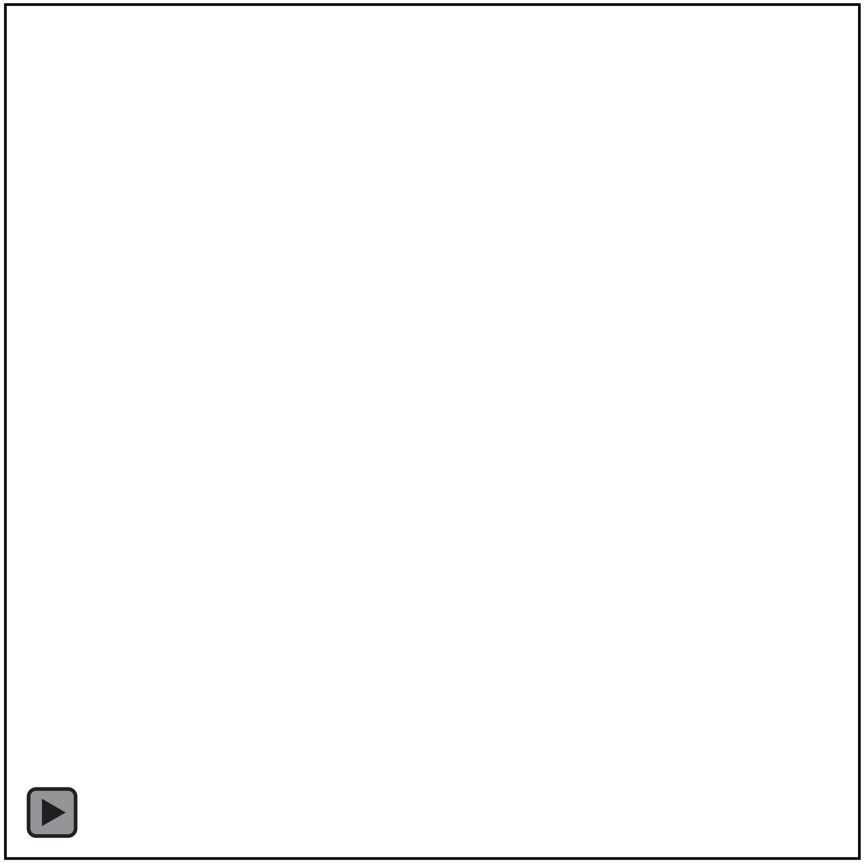


[Tian: zi2zi, Master Chinese Calligraphy with Conditional Adversarial Networks,2017]

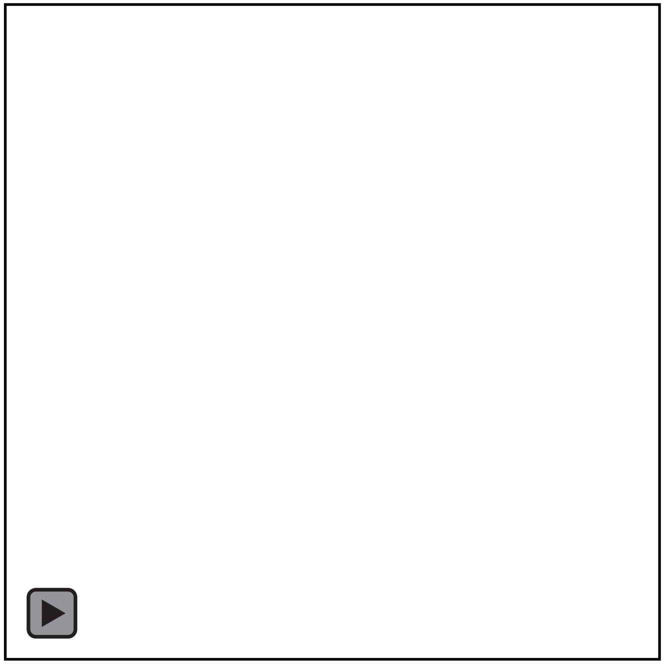
CycleGAN

[Zhu et al.: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. 2017]

More applications



[Glow: Generative Flow with Invertible 1x1 Convolutions. Kingma and Dhariwal, 2018]



Crypko, 2018

Generative Adversarial Network

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: $D(x) = P(x$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

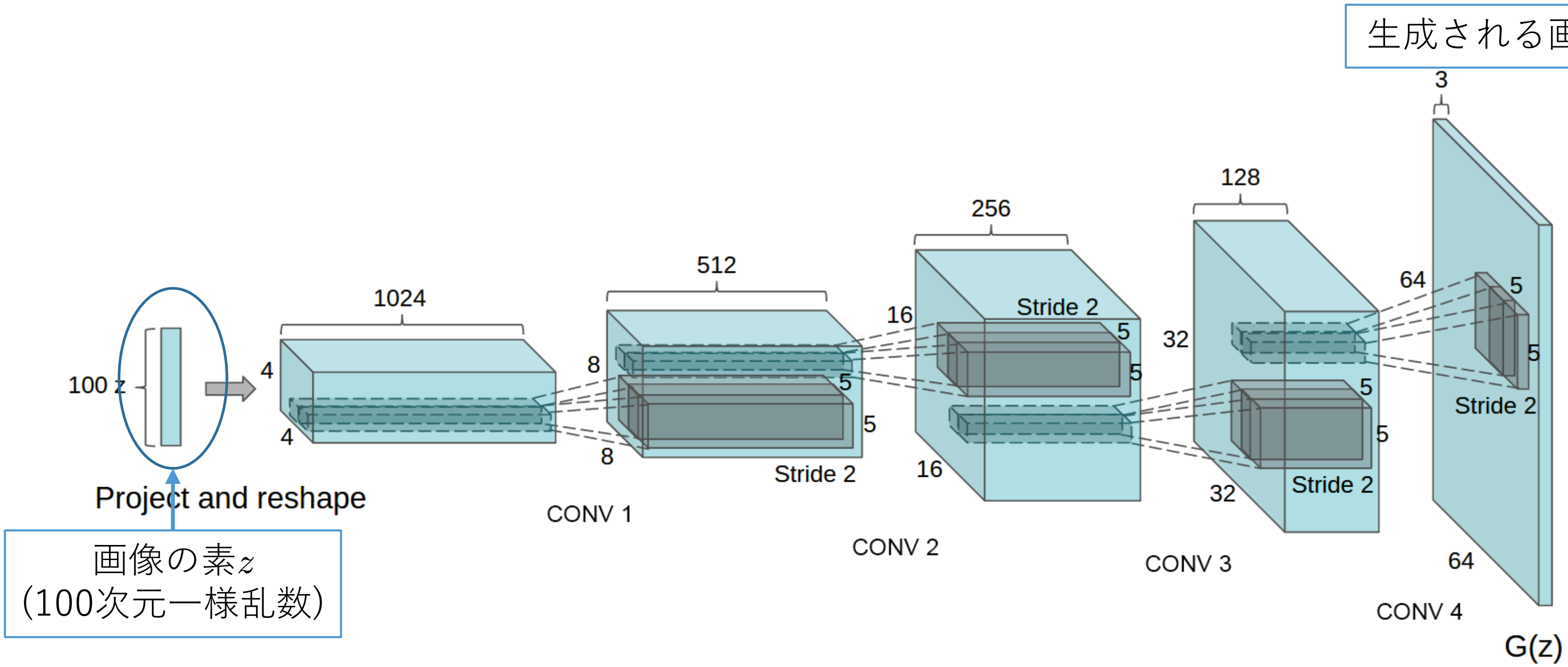
$$\min_G \max_D \underbrace{\mathrm{E}_{x\sim p_{\mathrm{data}}}[\log D(x)]}_{\text{本当の画像を本物と判別する確率}} + \underbrace{\mathrm{E}_{z\sim p_z}[\log(1-D(G(z)))]}_{\text{偽物の画像を偽物と判別する確率}}$$

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.

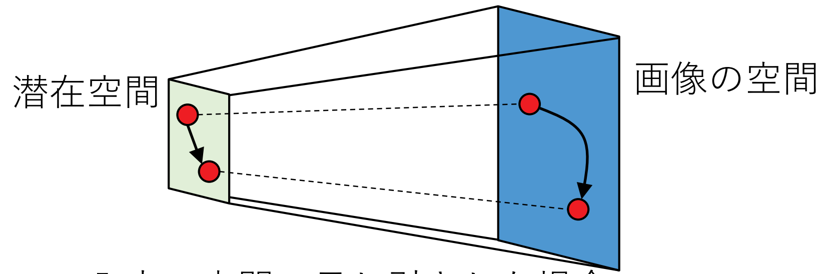
DCGAN (Deep Convolutional GAN)
畳み込みネットを用いたGAN

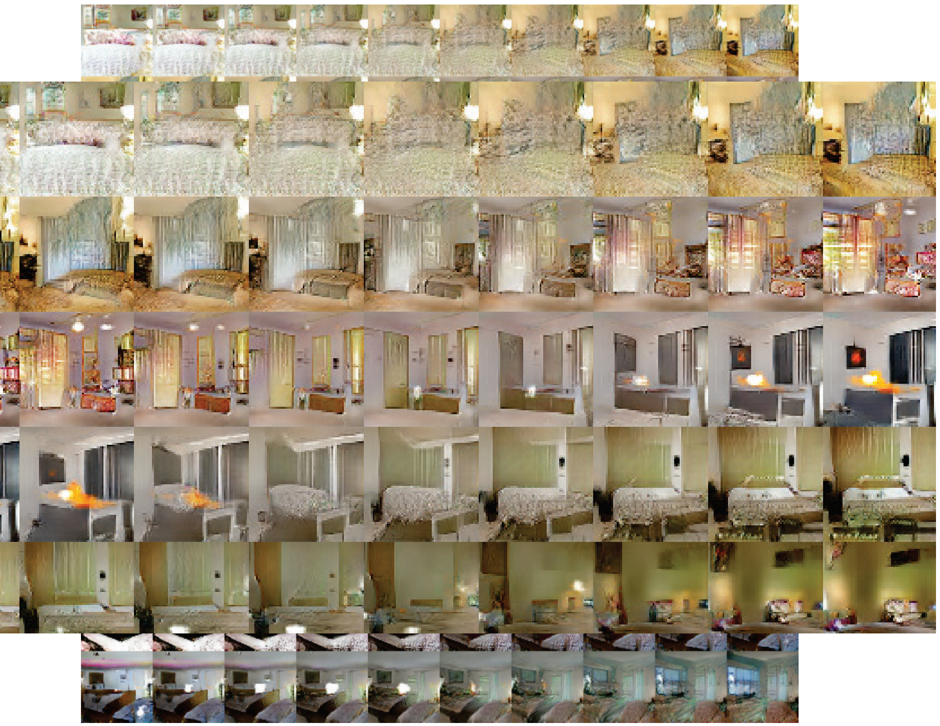


DCGANのGenerator

- 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.

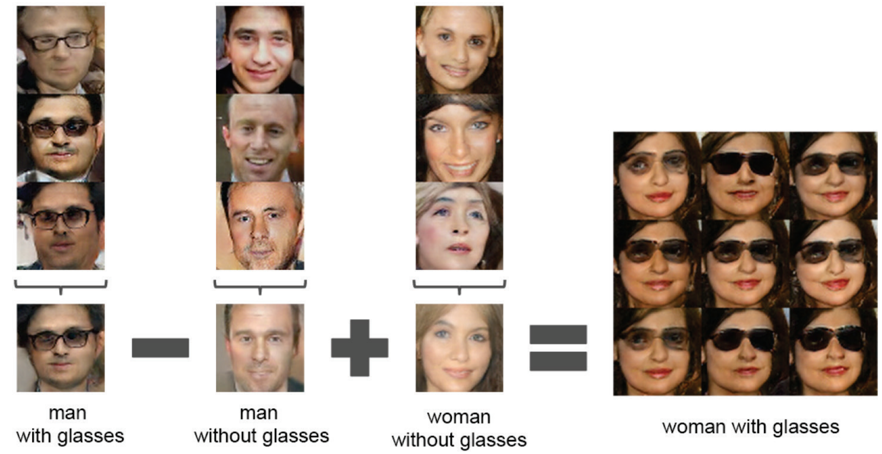
Radford, Metz & Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks.” ICLR2016.



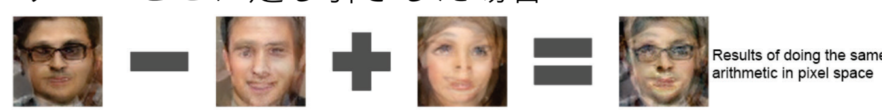


生成されたベッドルーム画像

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



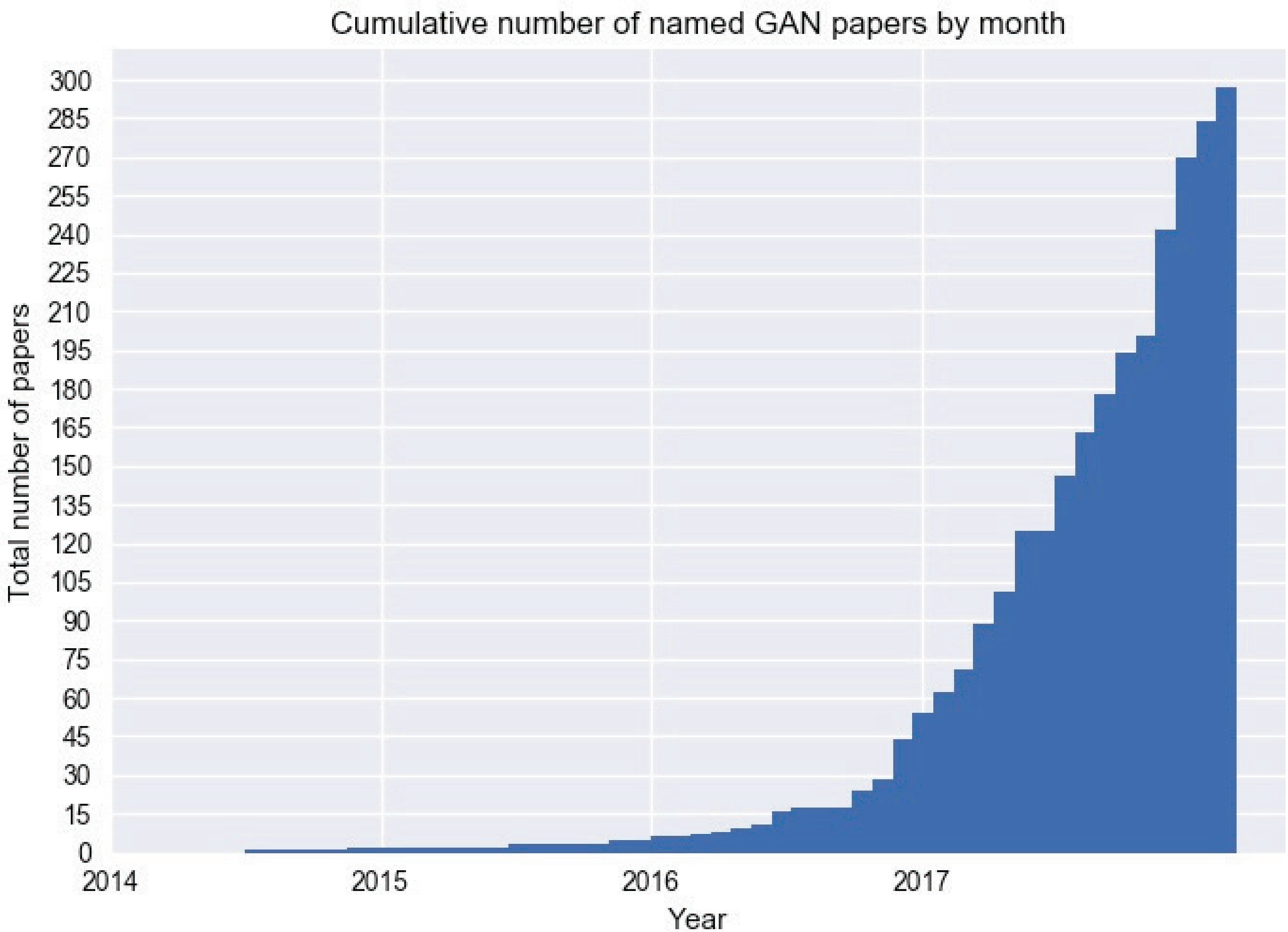
生成されたベッドルーム画像
入力zの凸結合で中間的画像が
得られる.

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

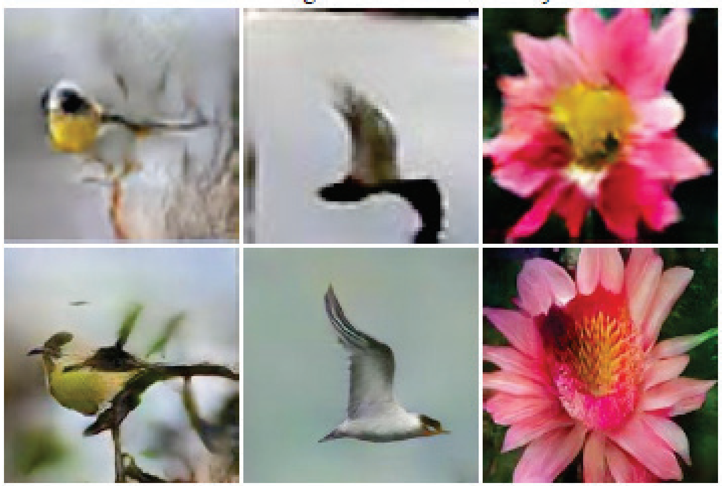
StackGAN [Zhang+etal.2016]　荒い画像を生成してからそれを高精細に修正（超解像）

入力文章

This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face	This bird is white with some black on its head and wings, and has a long orange beak	This flower has overlapping pink pointed petals surrounding a ring of short yellow filaments

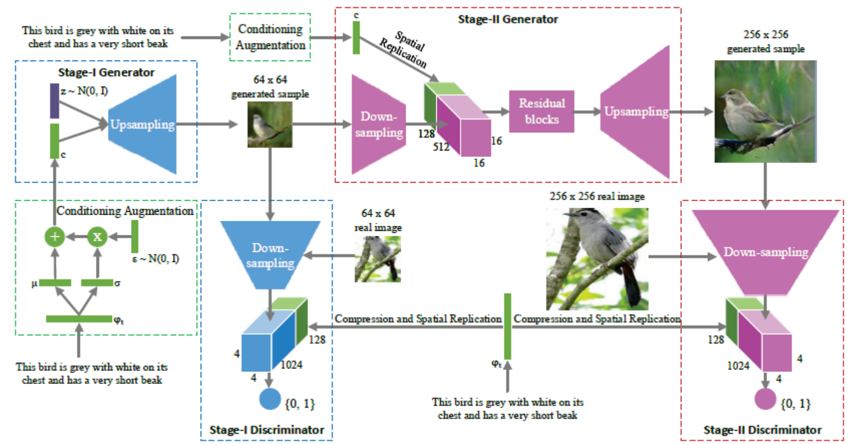
(a) Stage-I images

荒い画像を生成



(b) Stage-II images

さらにこうなる



Text description	This flower has petals that are white and has pink shading	This flower has a lot of small purple petals in a dome-like configuration	This flower has long thin yellow petals and a lot of yellow anthers in the center	This flower is pink, white, and yellow in color, and has petals that are striped	This flower is white and yellow in color, with petals that are wavy and smooth	This flower has upturned petals which are thin and orange with rounded edges	This flower has petals that are dark pink with white edges and pink stamen

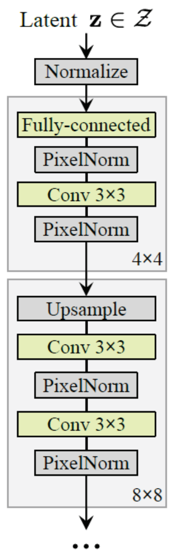
64x64 GAN-INT-CLS [22]　既存手法

256x256 StackGAN　StackGAN

Style-Based Generator

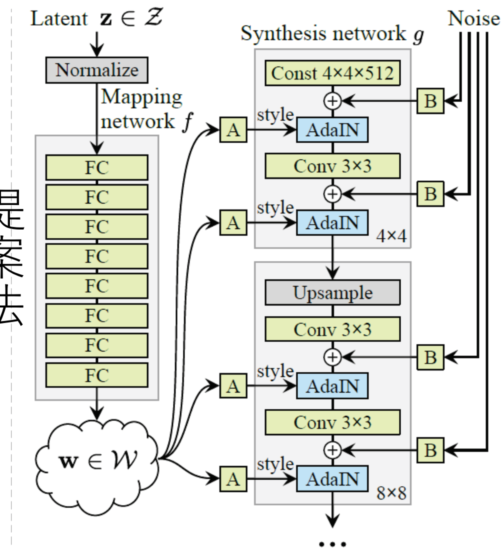
<u>中間層にもノイズ</u>を入れて細かいスタイルのバリエーションを確保

従来法



(a) Traditional

提案法

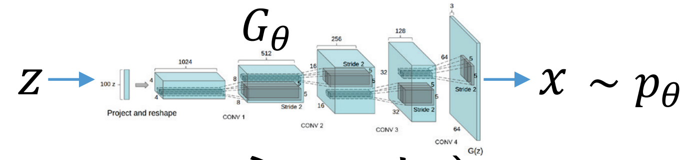


(b) Style-based generator

[Karras, Laine, Aila: A Style-Based Generator Architecture for Generative Adversarial Networks. arXiv:1812.04948]

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

近づけたい

xの分布p_θ ⟷ 真の分布q

---

f-divergenceの最小化

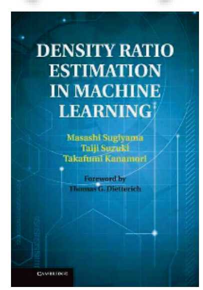
f-divergence:
分布の距離(のようなもの)

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

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.]

f-divergence

- f-divergence

$$D_f(p||q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

fは凸関数

- KL-divergence

$$D_{\mathrm{KL}}(p||q) = \int p(x) \log\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

$$(f(x) = x\log(x))$$

- Jensen-Shannon divergence

$$D_{\mathrm{JS}}(p||q) = \frac{1}{2} D_{\mathrm{KL}}(p||(p+q)/2) + \frac{1}{2} D_{\mathrm{KL}}(q||(p+q)/2)$$

$$(f(x) = x\log(x) - (1+x)\log((1+x)/2))$$

GANとの関係：

$$\max_D \mathrm{E}_{x\sim p^*}[\log(D(x))] + \mathrm{E}_{x\sim q}[\log(1 - D(x))] = D_{\mathrm{JS}}(p^*||q)$$

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

Wasserstein距離とその仲間

- 積分型確率測度距離

$$D(P||Q) = \sup_{f \in \mathcal{F}} \mathrm{E}_P[f] - \mathrm{E}_Q[f]$$

- f(x)としてf(x)=xのみを用いれば平均値の差を見ていることになる.
- f(x)として，f(x)=xおよびf(x)=x^2も考えれば二次モーメントの差も考慮できる.
- Fとしてもっと広い関数の集合を考えれば分布の“距離”になる.

- 1-Wasserstein距離

  $\mathcal{F}$ =1-リプシッツ連続な関数の集合

$$\sup_{x,y} \frac{|f(x) - f(y)|}{\|x - y\|} \leq 1$$

- MMD (Maximum Mean Discrepancy)

  $\mathcal{F}$ =ある再生核ヒルベルト空間の単位球

Wasserstein距離について

- 「輸送距離」とも言われる

$$\inf_{\pi \in \Pi(\mu,\nu)} \int c(x,y) \mathrm{d}\pi(x,y)$$

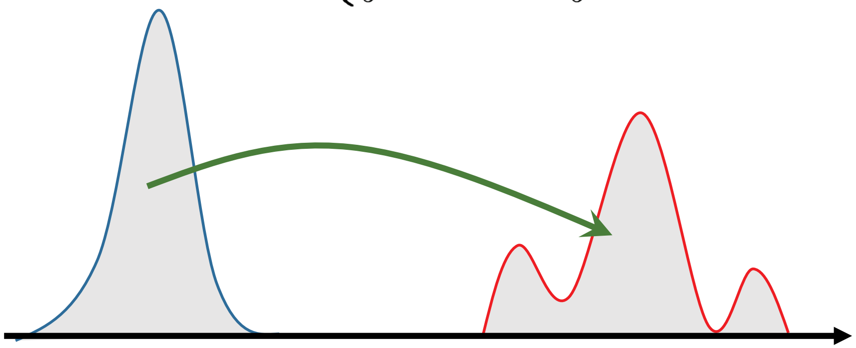
周辺分布を固定した同時分布の中で最小化

$$(W_1\colon\ c(x,y) = \|x - y\|)$$

- 分布のサポートがずれていてもwell-defined
- 底空間の距離が反映されている

（双対表現）

$$\inf_{\pi \in \Pi(\mu,\nu)} \int c(x,y) \mathrm{d}\pi(x,y) = \sup \left\{ \int \psi \mathrm{d}\mu + \int \phi \mathrm{d}\nu \mid \psi(x) + \phi(y) \leq c(x,y) \right\}$$



自然言語処理

キャプション生成

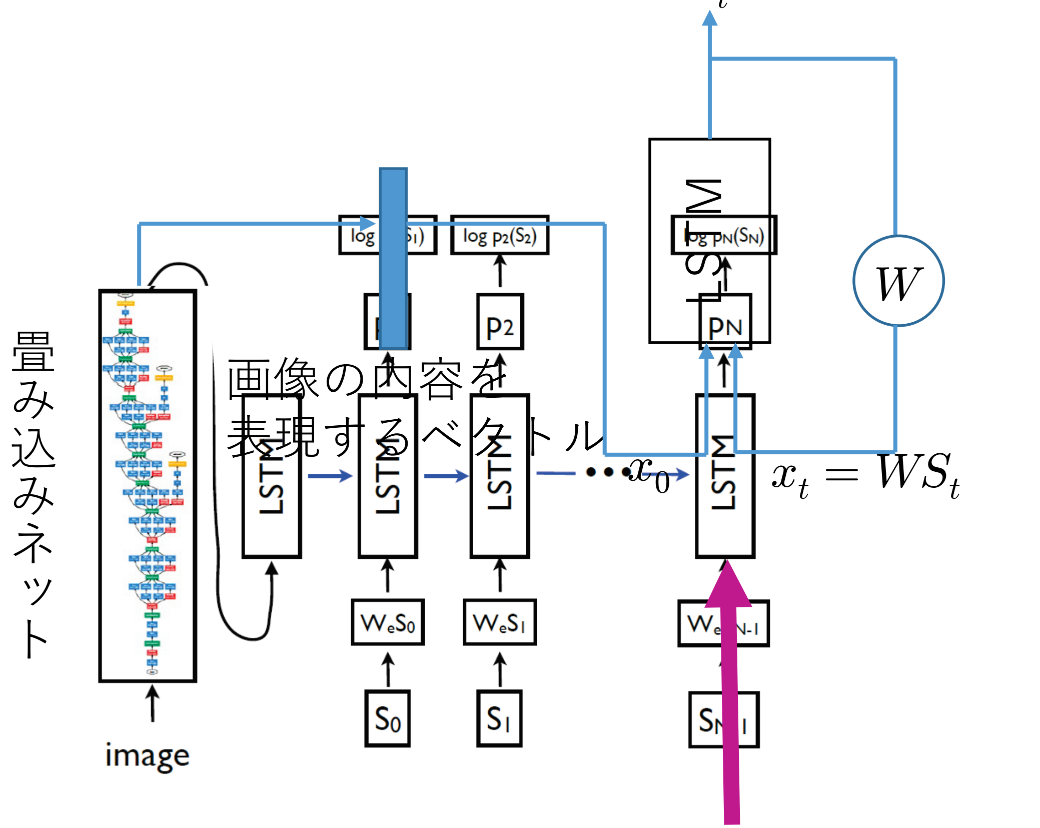
入力画像

自動生成された説明文

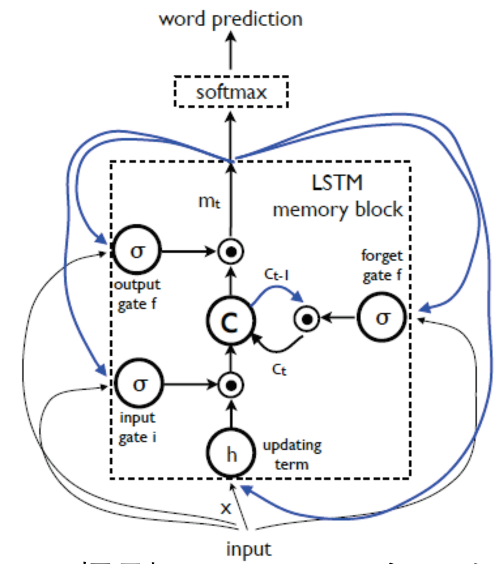
Google による画像説明文章の自動生成

[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)

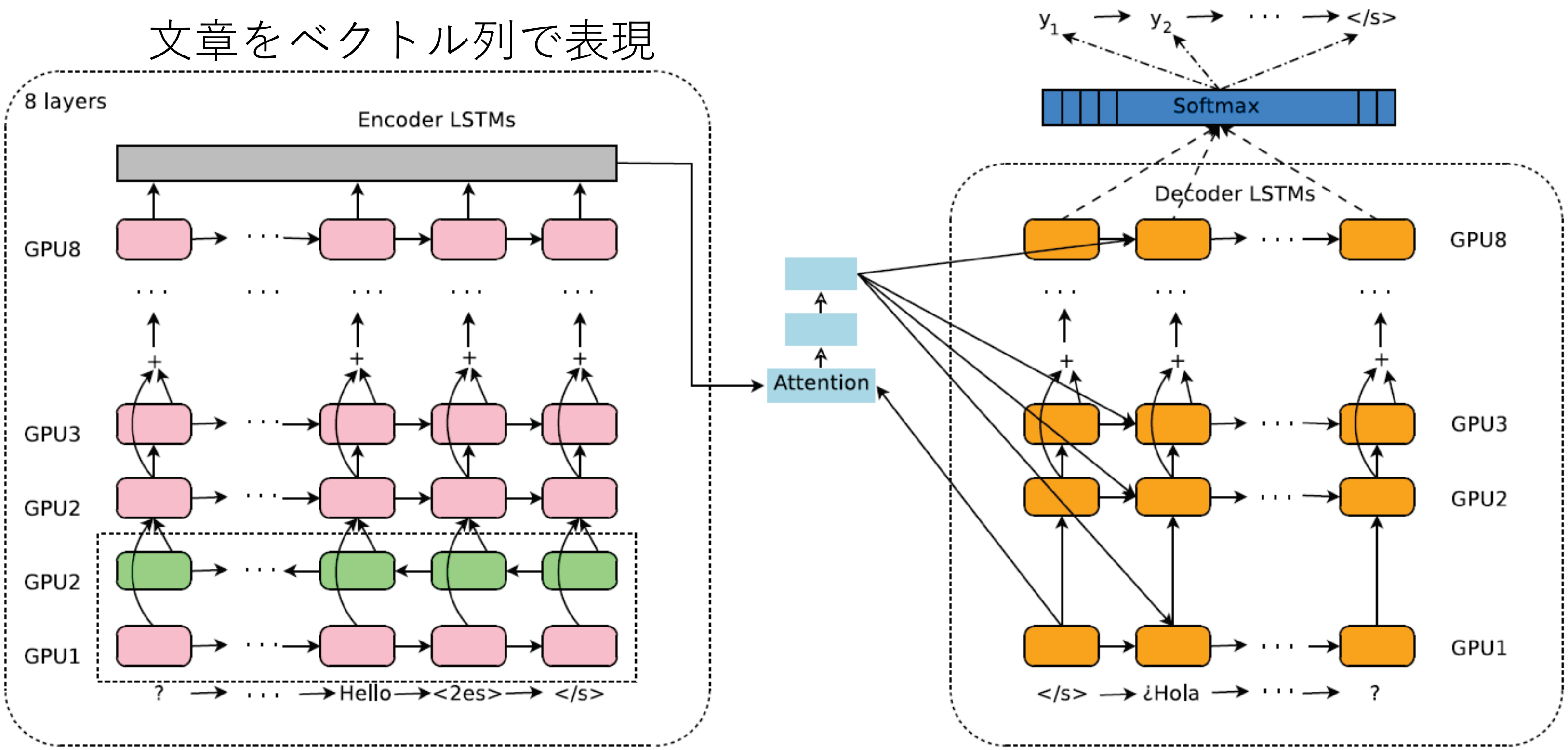


- 再帰型ニューラルネット
- 言葉の列等を生成できる
- 内部状態，前の出力，今の入力で次の出力が決まる

- 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

文章をベクトル列で表現



入力文　　　　　　　　　　出力文

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

日本語　　英語

深層学習は機械学習の一手法ですか？

Shinsō gakushū wa kikai gakushū no ichishuhōdesu ka?

Is depth learning a method of machine learning?

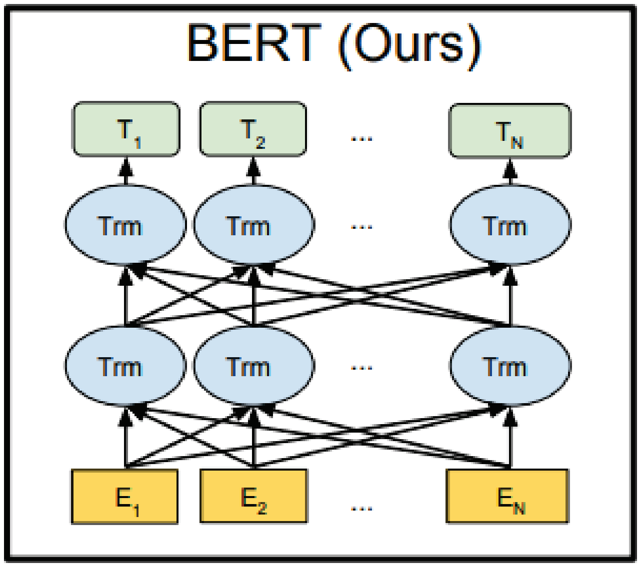
Google 翻訳で開く　　フィードバック

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?

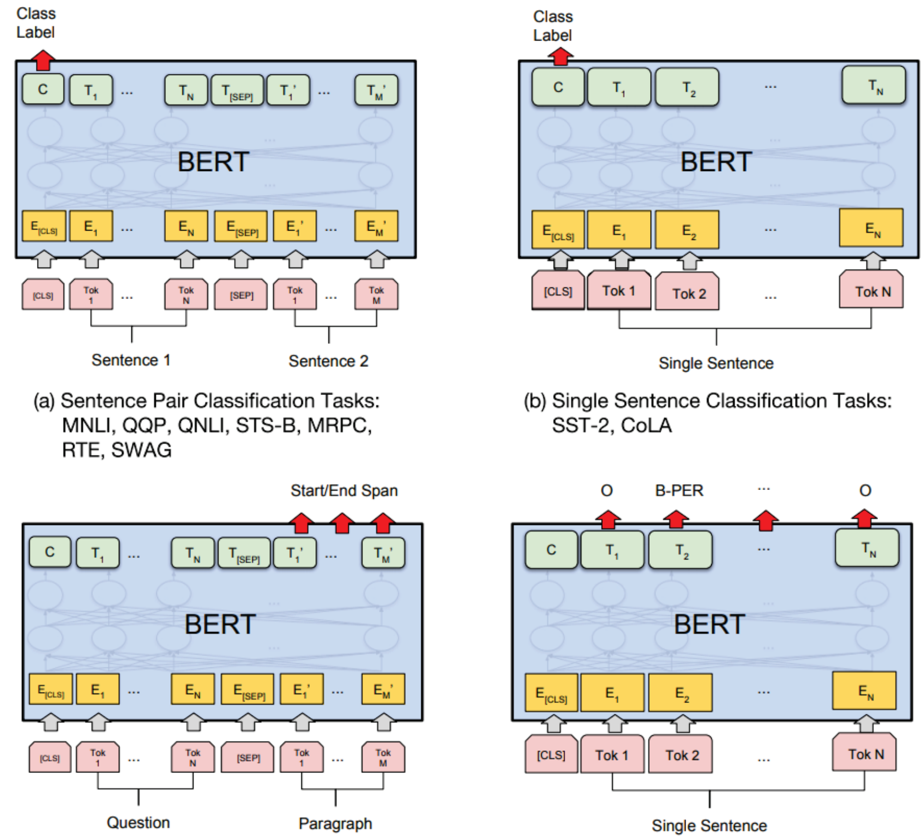
BERT

- 自然言語処理における事前学習モデル.
- 文章・単語の表現を学習.
- 各種タスクはその表現を用いてfine tuningすればよい.
- 多くの問題で精度を改善.



System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\mathrm{BERT_{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\mathrm{BERT_{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); $\mathrm{BERT_{BASE}}$ = (L=12, H=768, A=12); $\mathrm{BERT_{LARGE}}$ = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from https://gluebenchmark.com/leaderboard and https://blog.openai.com/language-unsupervised/.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

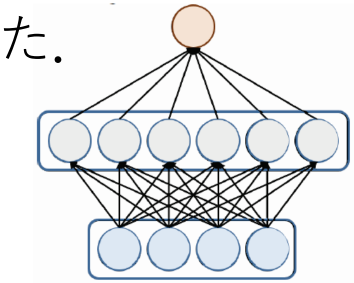
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

[Devlin, Chang, Lee, Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805]

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？

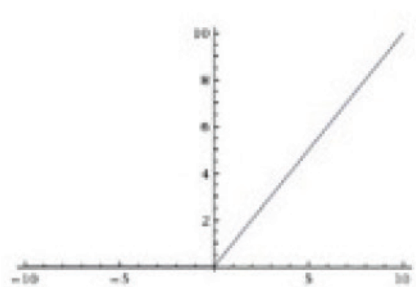
（「任意の関数」や「任意の精度」の意味はどのような関数空間を考えるかに依存）

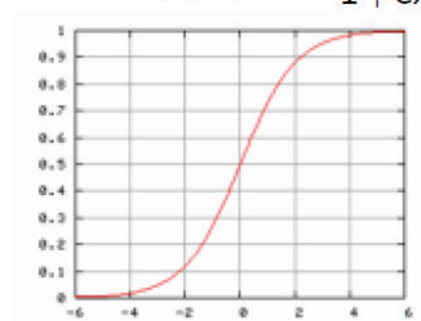
hがシグモイド関数やReLUなら万能性を有する．

年		基底関数	空間

Activation functions:

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





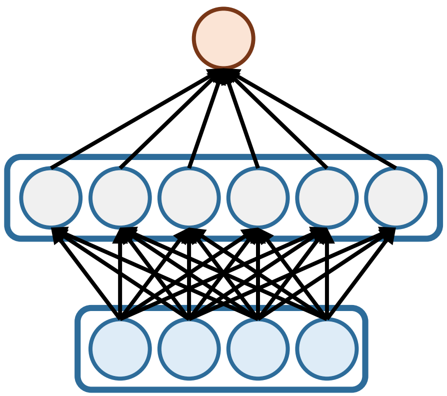
Kは任意のコンパクト集合

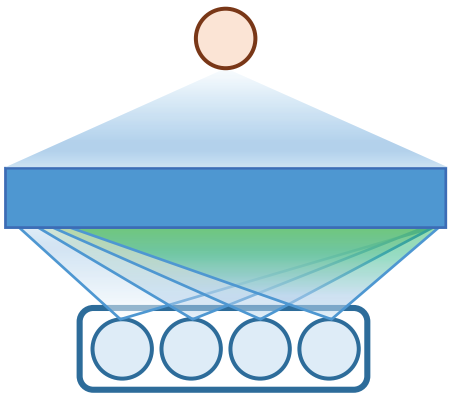
参考：園田, “ニューラルネットの 積分表現理論”, 2015.

積分表現

有限和近似（３層NN）　　　　　　　真の関数

$$\hat{f}(x) = \sum_{j=1}^{m} v_j \eta(w_j^\top x + b_j) \quad \simeq \quad f^\circ(x) = \int h^\circ(w,b)\eta(w^\top x + b)\mathrm{d}w\mathrm{d}b$$



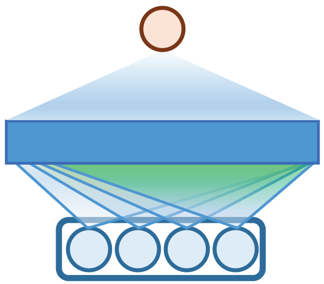


(Sonoda & Murata, 2015)

- 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

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



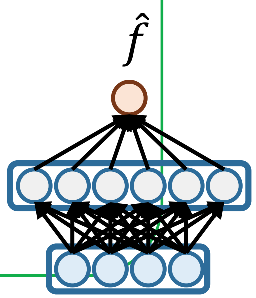
仮定： $\displaystyle \int_{\mathbb{R}^d} \|w\| |\tilde{f}(w)| \mathrm{d}w < \infty$

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

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



活性化関数の条件： $\eta(-z) = 1 - \eta(z)$

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

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

(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？

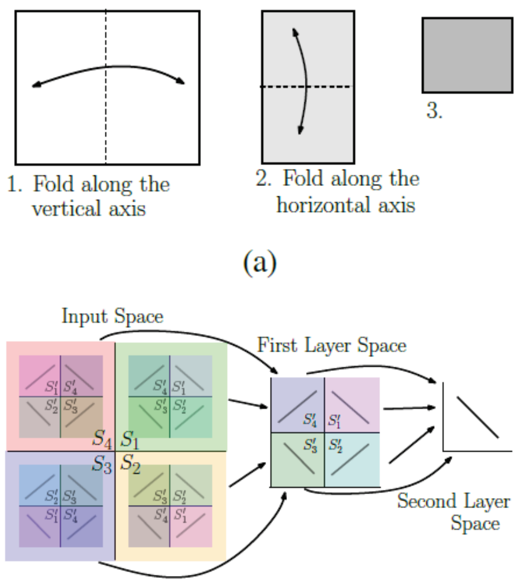
- 層の数に対して表現力は**指数的**に上がる.

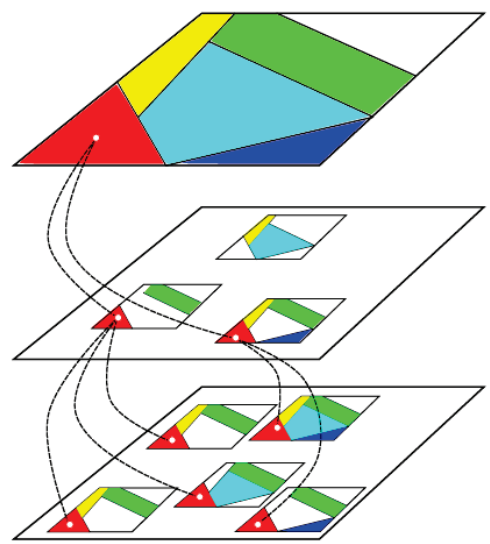
$$\left(\frac{n}{n_0}\right)^{L-1}\sum_{j=0}^{n_0}\binom{n}{j}$$

- 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





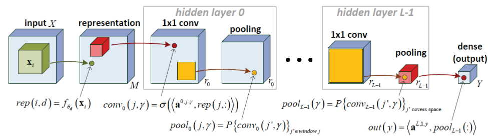
折り紙のイメージ

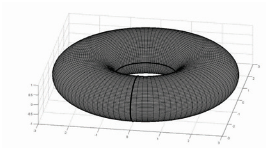
Montufar, Guido F., et al. "On the number of linear regions of deep neural networks." 2014.

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

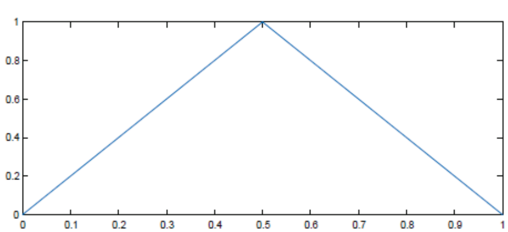
- **多項式展開，テンソル解析** [Cohen et al., 2016; Cohen & Shashua, 2016]
  単項式の次数
- **代数トポロジー** [Bianchini & Scarselli, 2014]
  ベッチ数(Pfaffian)
- **リーマン幾何 + 平均場理論** [Poole et al., 2016]
  埋め込み曲率



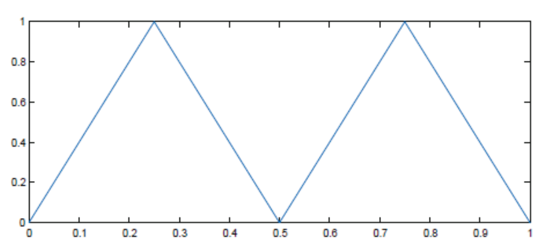


対称性の高い関数は，特に層を深くすることで得をする．

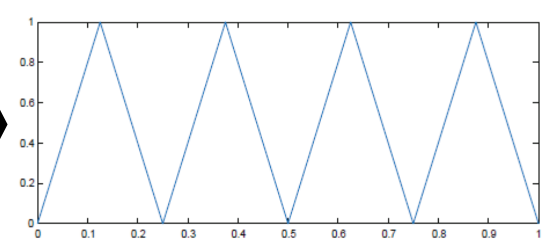
$$h(x) = \begin{cases} 2x & (0 \leq x \leq 1/2) \\ 2(1-x) & (1/2 \leq x \leq 1) \\ 0 & (\text{otherwise}). \end{cases}$$



$h(x)$



$h \circ h(x)$

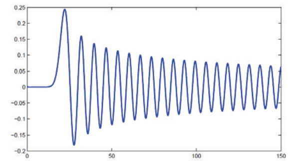


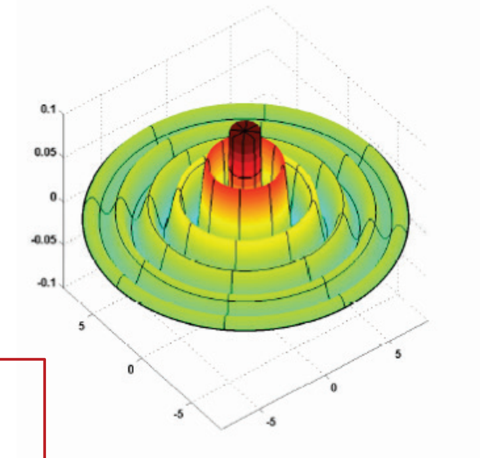
$h \circ h \circ h(x)$

多層が得する例

$$g(\|x\|^2) = g(x_1^2 + x_2^2 + \cdots + x_{d_x}^2)$$

gはBessel関数を元に構成





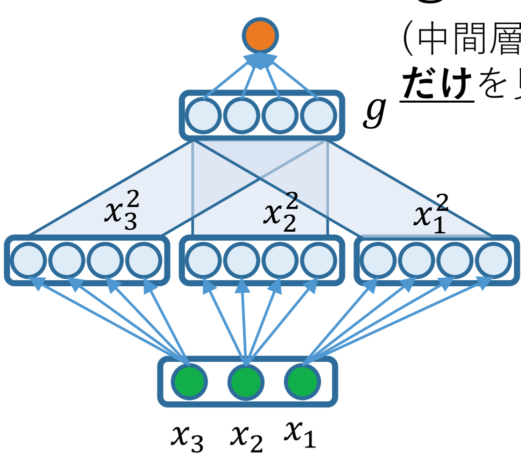
四層（中間層二層）：$O(\mathrm{poly}(d_x))$ノードで十分
三層（中間層一層）：$\Omega(\exp(d_x))$ノードが必要

（Eldan&Shamir, 2016）

四層

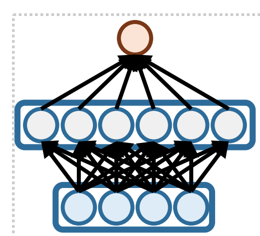
まず二乗和$x_1^2 + \cdots + x_{d_x}^2$を作ってから$g$を作用.

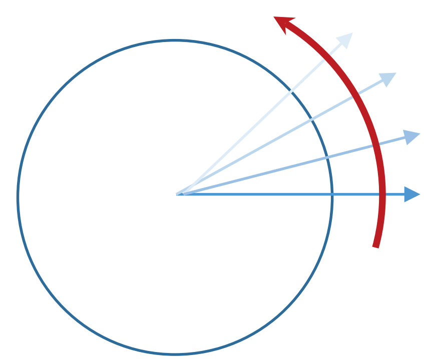
(中間層で<u>**座標軸方向だけ**</u>を見ればよい)



三層

全方向をケアする必要がある
（座標軸方向だけではダメ）

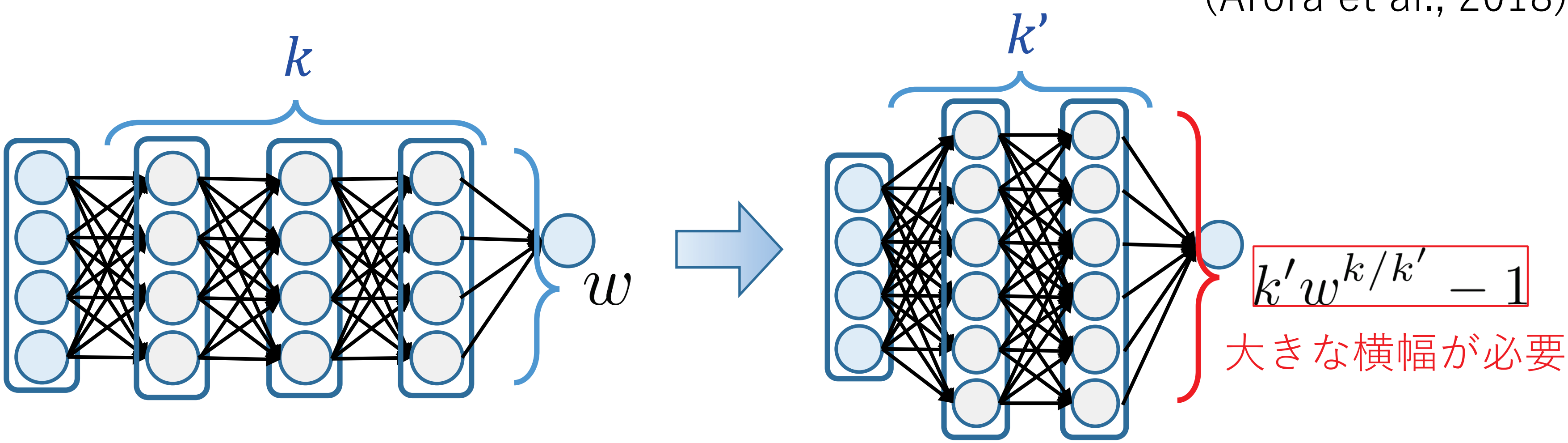




区分線形関数の表現

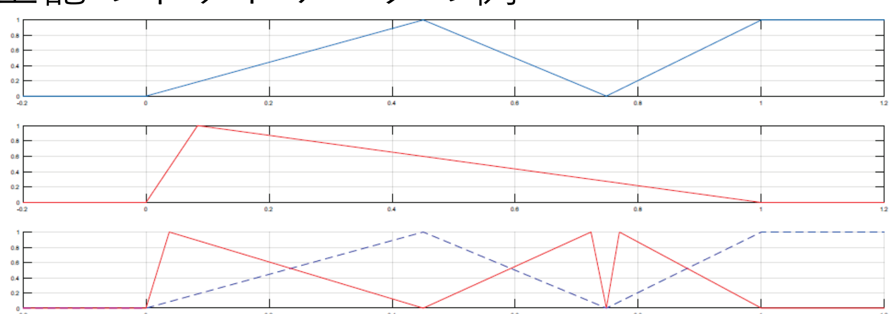
- 任意の区分線形関数($\mathbb{R}^d \to \mathbb{R}$)は深さ$\lceil \log_2(d+1) \rceil$のReLU-DNNで表現可能
- ある横幅w，縦幅kのReLU-DNNが存在して，それを縦幅$k'(< k)$のネットワークで表現するには横幅$k' w^{k/k'} - 1$が必要．

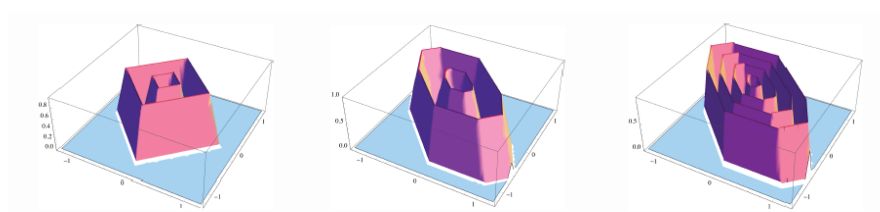
(Arora et al., 2018)



やはり層の深さに対し指数関数的に表現力が増加

上記のネットワークの例





(a) $H_{\frac{1}{2},\frac{1}{2}} \circ N_{\ell_1}$ (b) $H_{\frac{1}{2},\frac{1}{2}} \circ \gamma_{Z(\mathbf{b}^1,\mathbf{b}^2,\mathbf{b}^3,\mathbf{b}^4)}$ (c) $H_{\frac{1}{2},\frac{1}{2},\frac{1}{2}} \circ \gamma_{Z(\mathbf{b}^1,\mathbf{b}^2,\mathbf{b}^3,\mathbf{b}^4)}$

有理関数の近似

- 有理関数をReLU-DNNで近似

$$p : [0,1]^d \to [-1,+1],\ q : [0,1]^d \to [2^{-k},+1]$$ ：r次多項式

p/q をReLU-DNNで近似したい

あるReLU-DNNfが存在してノード数と近似誤差が次のように抑えられる：

ノード数

$$\mathrm{O}(\mathrm{poly}(k,r,d)\underline{\mathrm{poly}(\log(1/\epsilon))})$$

近似誤差

$$\sup_{x\in[0,1]^d}\left|f(x)-\frac{p(x)}{q(x)}\right|\le\epsilon$$

- ReLU-DNNを有理関数で近似

k-層で各層のノード数mの任意のReLU-DNNfに対しては,
次数と近似誤差が以下で抑えられる有理関数p/qが存在：

次数（分母qと分子pの次数の最大値）

$$\mathrm{O}(\underbrace{\log(k/\epsilon)^k m^k}_{\text{深さに対して指数的に増大}})$$

近似誤差

$$\sup_{x\in[0,1]^d}\left|f(x)-\frac{p(x)}{q(x)}\right|\le\epsilon$$

- ReLU-DNNを多項式で近似： $\Omega(\mathrm{poly}(1/\epsilon))$ の次数が必要

→有理関数に比べて表現力が低い

有理関数の近似

- 有理関数をReLU-DNNで近似

$$p:[0,1]^d \to [-1,+1],\ q:[0,1]^d \to [2^{-k},+1]$$ ：r次多項式

p/q をReLU-DNNで近似したい

あるReLU-DNNfが存在してノード数と近似誤差が次のように抑えられる：

ノード数
$$\mathrm{O}(\mathrm{poly}(k,r,d)\mathrm{poly}(\log(1/\epsilon)))$$

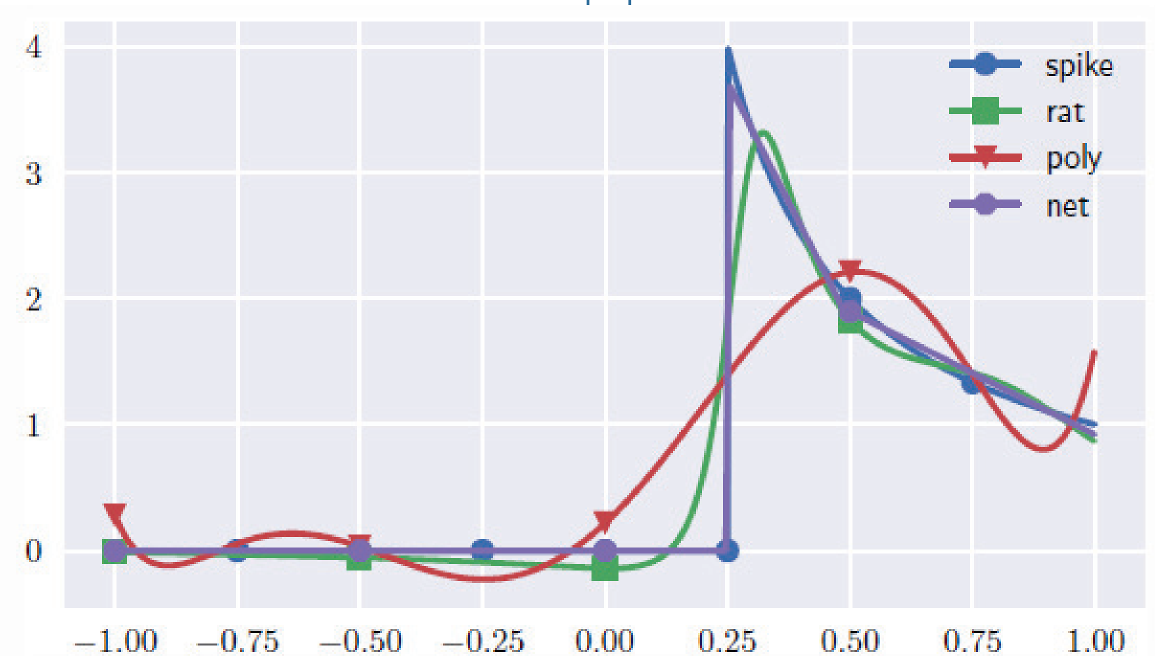
近似誤差
$$\sup_{x\in[0,1]^d}\left|f(x)-\frac{p(x)}{q(x)}\right| \le \epsilon$$

- ReLU-DNI

k-層で各
次数と近

次数（分母
O(l

$$\frac{p(x)}{q(x)}\Bigg| \le \epsilon$$



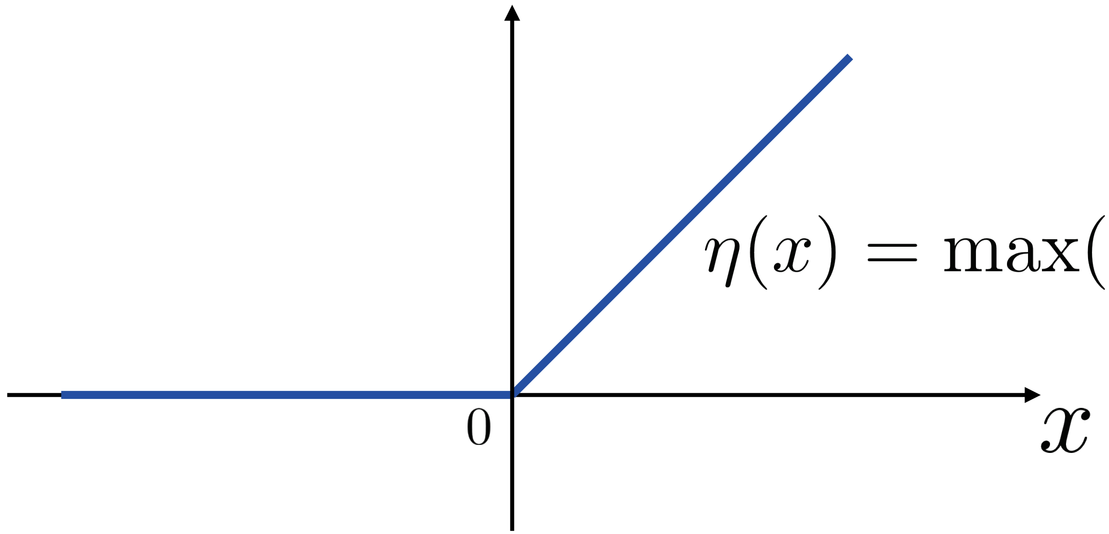
- ReLU-DNI 必要

→有理関数に比べて表現力が低い

滑らかな関数の近似 by ReLU

ReLUの表現力

- ReLU活性化関数



$$\eta(x) = \max(x, 0) = (x)_+$$

- 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

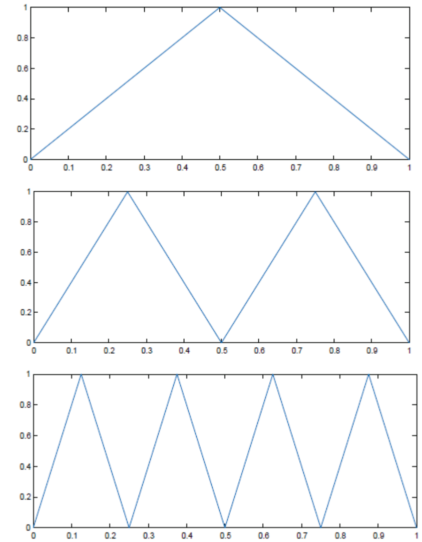
滑らかな関数の近似 (Yarotsky, 2016)

- 二次関数の構成

<u>これが全ての基本</u>

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

$$R_k(x) = \underbrace{h \circ h \circ \cdots \circ h}_{k \text{ times}}(x).$$



$h(x)$

$h \circ h(x)$

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

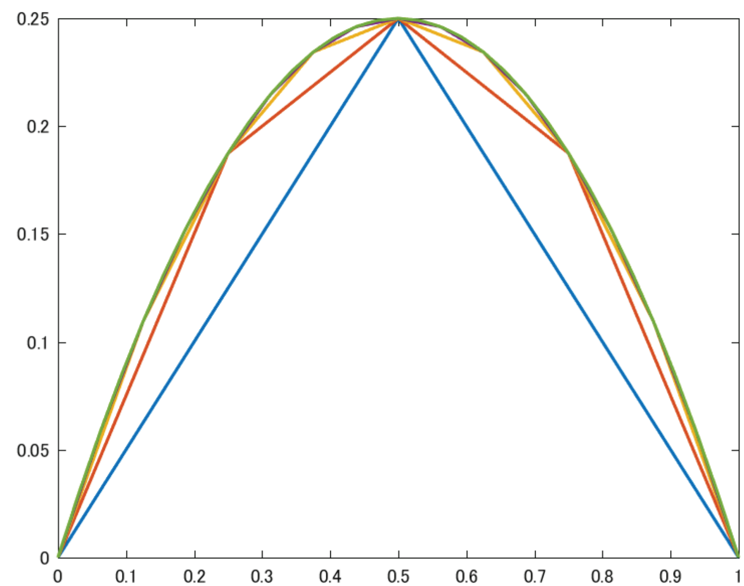
2 次関数の近似 (Telgarsky, 2015)

$$\boxed{\left| x(1-x) - \sum_{k=1}^{m} (2^{-2k}) R_k(x) \right| \le 2^{-m}}$$

層を重ねることで<u>指数的に</u>誤差が減少

層の数，横幅，ユニット数：$\mathrm{O}(\log(1/\epsilon))$

中間層 1 層の場合：　$\Omega(1/\sqrt{\epsilon})$



改善（多項式オーダ→logオーダ）

多項式の構成

- 二次関数→掛け算

$$(x+y)^2 - x^2 - y^2 = 2xy$$

（足し算はReLUで実現可能）

- 掛け算→多項式

$$x^m = \underbrace{x \times (x \times (x \times (\cdots)))}_{m\mathrm{times}}$$

（二次関数から構成した
掛け算を繰り返し適用）

$$\Rightarrow \sum_{|\alpha| \leq m} c_\alpha (x - x_0)^\alpha$$

（足し算と合わせて多項式を構成）

→ 滑らかな関数の近似に利用

滑らかな関数の構成

$\beta \in (0,\infty),\ m = \lfloor \beta \rfloor$

滑らかな関数のクラス（Holder class）

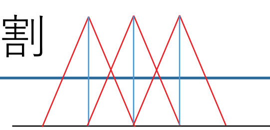
$$\mathcal{F}^{\beta}(K) = \left\{ f \mid \max_{|\alpha|\le m} \|\partial^{\alpha} f\|_{\infty} + \max_{|\alpha| = m} \sup_{x\in[-1,1]^d} \frac{|\partial^{\alpha} f(x) - \partial^{\alpha} f(y)|}{|x-y|^{\beta - m}} \le K \right\}$$

- 滑らかな関数の局所的近似（テイラー展開）

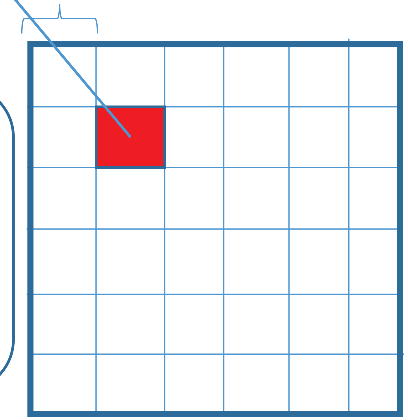
$$\sup_{x:\|x-x_0\|_\infty \le \delta} \left| f(x) - \underbrace{\sum_{|\alpha|\le m} \frac{\partial^{\alpha} f(x_0)}{\alpha!} \underline{(x-x_0)^{\alpha}}}_{m\text{次多項式}\ (R_{x_0} f(x))} \right| \le C\,(d\delta)^{\beta}$$

- 全体の近似

$$\sup_{x\in[-1,1]^d} \left| f(x) - \sum_{x_0\in D(\delta)} \underbrace{\prod_{j=1}^{d} (1 - \delta^{-1}|x_j - x_{0,j}|)_{+}}_{1\text{の分割}} R_{x_0} f(x) \right| \le C\,(d\delta)^{\beta}$$



$\delta \simeq \epsilon^{1/\beta} d^{-1}$



(Yarotsky, 2016; Liang&Srikant, 2017)

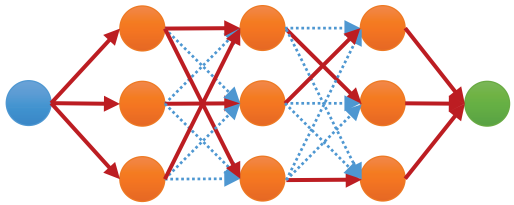
推定理論

- パラメータ数

$$\underbrace{O\left(\frac{1}{\delta^d}\right)}_{\text{領域分割の数}} \times \underbrace{O(\log(1/\epsilon))}_{\text{分割ごとのパラメータ数}} = O(\epsilon^{-\frac{d}{\beta}}\log(1/\epsilon))$$

横幅 :$O(\epsilon^{-\frac{d}{\beta}}\log(1/\epsilon))$　縦幅 : $O(\log(1/\epsilon))$ のネットワークに埋め込める

$\mathcal{F}(L,w,s)$: 縦幅L，横幅w，非ゼロパラメータ数s の深層NNモデルの集合



深層学習の汎化誤差（Schmidt-Hieber, 2017）

縦幅$L=O(\log(n))$，横幅$w=O(n^{\frac{d}{2\beta+d}}\log(n))$，非ゼロ要素$s=O(n^{\frac{d}{2\beta+d}}\log(n))$

$$\hat{f} = \mathop{\arg\min}_{f\in\mathcal{F}(L,w,s)} \sum_{i=1}^{n}(y_i - f(x_i))^2$$

$$\mathrm{E}[\|\hat{f}-f^*\|_{L_2(P(X))}^2] \le O(n^{-\frac{2\beta}{2\beta+d}}\log(n)^2)$$

ミニマックス最適レート

$$\mathrm{E}[\|\hat{f}-f^*\|_{L_2(P(X))}^2] \le O\left(\underbrace{\frac{\epsilon^{-d/\beta}\log(\epsilon)^2}{n}}_{\text{バリアンス}} + \underbrace{\epsilon^2}_{\text{バイアス}}\right)$$

$\epsilon = n^{-\frac{\beta}{2\beta+d}}$ でバランス

補足

- ３層NNは$\beta = 2$までしか最適レートを達成しない.
- 横幅を広げることで縦幅をnに依存しない定数にすることも可能.
- 関数の合成$f = g_1 \circ g_2$を考えることで，以下のモデルも表現できる.
  - テンソル積モデル $f(x) = \sum_{\ell=1}^{N} a_\ell \prod_{r=1}^{k} f_{r,\ell}(x_r)$
  - 一般化加法モデル $f(x) = h(\sum_{r=1}^{d} f_r(x_r))$
- 同様の議論を拡張することで区分滑らかな関数の推定も可能 (最適性も示せる) (Petersen&Voigtlaender, 2017; Imaizumi&Fukumizu, 2018)
- 浅層ネットワークのパラメータ数の下界も知られている (Liang&Srikant, 2017) $\Omega(\mathrm{poly}(1/\epsilon))$

深層学習の最適化

勾配降下法

深層学習における最適化

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

（確率的）勾配降下法：

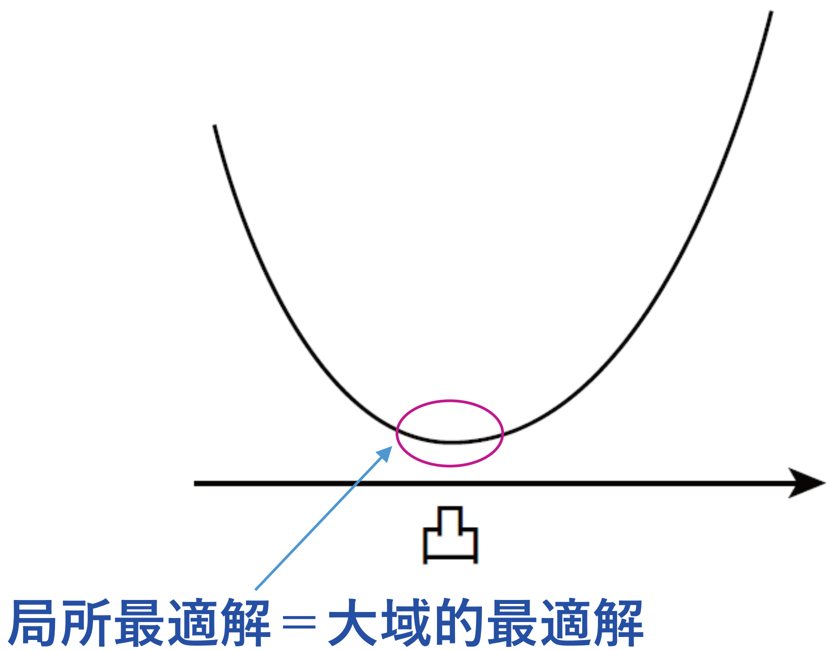
GD $$W^t = W^{t-1} - \alpha \nabla L(W)$$

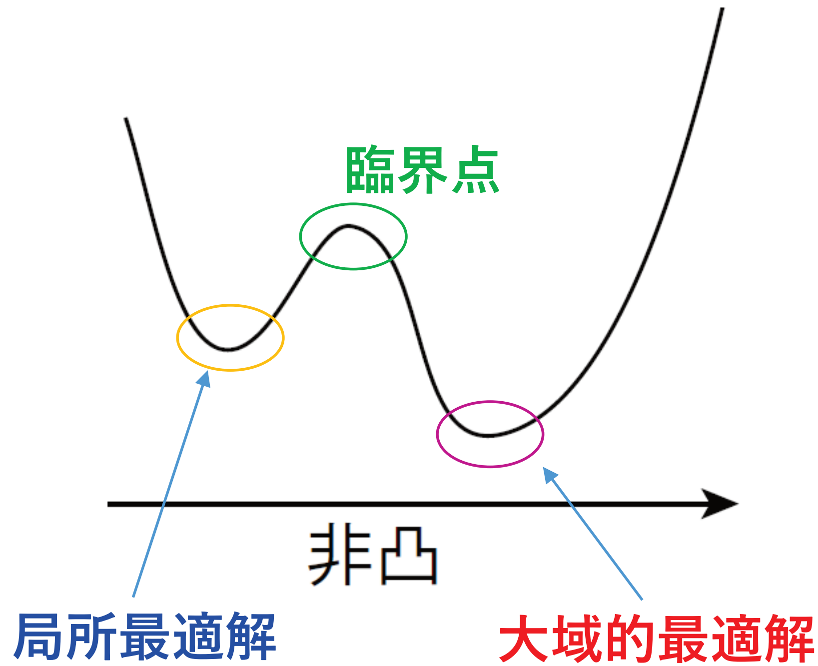
SGD $$W^t = W^{t-1} - \alpha \frac{1}{|I_B|}\sum_{i \in I_B} \nabla \ell(z_i, W)$$

問題点

目的関数が非凸関数

凸関数 $\boxed{\theta f(x) + (1-\theta) f(y) \geq f(\theta x + (1-\theta) y) \quad (\forall x, y \in \mathbb{R}^p, \theta \in [0,1])}$

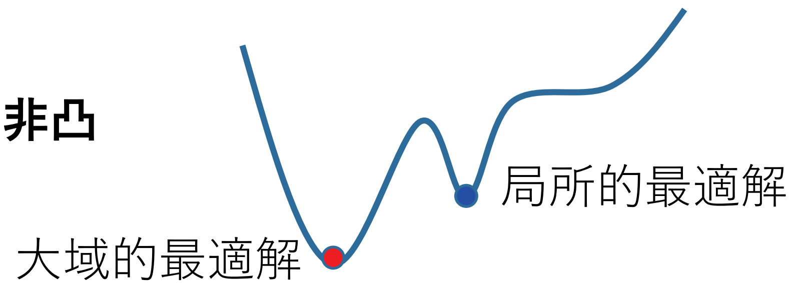




局所最適解や鞍点にはまる可能性あり

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)

こちらのみ動かす

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

- 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.
    (線形パーセプロ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.

非凸最適化一般論

- **<u>ある条件を満たす最適化法はほとんど確実に鞍点にはまらない</u>**
  - ➢ 安定多様体を用いた議論
  - ➢Strict Saddleの仮定 (やや強い)
  - ➢勾配法，鏡像降下法，近接点法，座標降下法等は条件を満たす．

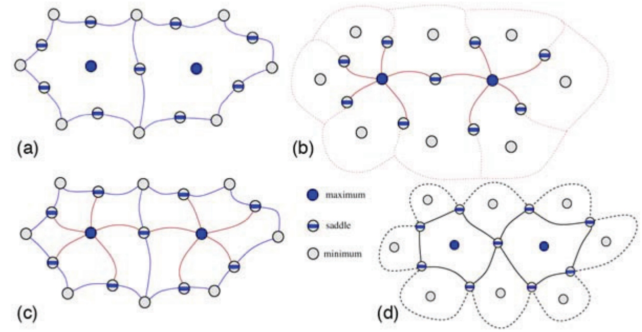


Fig. by Staadt, Natarajan, Weber, Wiley, Hamann (2007)

Lee, Panageas, Piliouras, Simchowitz, Jordan, Recht (2017): First-order Methods Almost Always Avoid Saddle Points.
Lee, Simchowitz, Jordan, Recht (2016): Gradient Descent Only Converges to Minimizers.

- <u>非確率的</u>勾配法は鞍点にはまると抜け出すのに次元に関して指数時間かかる．

Du, Jin, Lee, Jordan, Poczos, Singh (2017): Gradient Descent Can Take Exponential Time to Escape Saddle Points.

- わざとノイズを乗せることで鞍点から抜けられる．
  - ➢SGD, 加速法などに適用可

Jin, Ge, Netrapalli, Kakade, Jordan (2017): How to Escape Saddle Points Efficiently.
Jin, Netrapalli, Jordan: Accelerated Gradient Descent Escapes Saddle Points Faster than Gradient Descent.

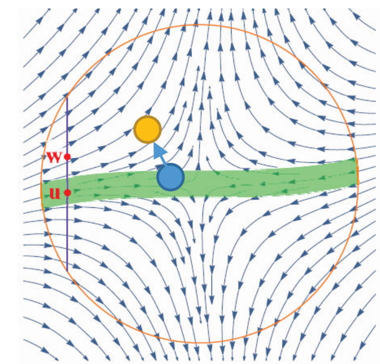
鞍点回避の方法

- 単純な勾配法への適用

ノイズを乗せるだけ
(鞍点脱出)

$$
\begin{aligned}
&\textbf{for } t = 0, 1, \ldots \textbf{ do} \\
&\quad \textbf{if } \text{perturbation condition holds } \textbf{then} \\
&\qquad \mathbf{x}_t \leftarrow \mathbf{x}_t + \xi_t, \qquad \xi_t \text{ uniformly } \sim \mathbb{B}_0(r) \\
&\quad \mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta \nabla f(\mathbf{x}_t)
\end{aligned}
$$

(普通の勾配法)



Jin, Ge, Netrapalli, Kakade, Jordan (2017): How to Escape Saddle Points Efficiently.

- 加速勾配法への適用

$$
\begin{aligned}
&\mathbf{v}_0 \leftarrow 0 \\
&\textbf{for } t = 0, 1, \ldots, \textbf{ do} \\
&\quad \textbf{if } \|\nabla f(\mathbf{x}_t)\| \le \epsilon \text{ and } \textit{no perturbation in last } \mathscr{T} \textit{ steps } \textbf{then} \\
&\qquad \mathbf{x}_t \leftarrow \mathbf{x}_t + \xi_t \qquad \xi_t \sim \mathrm{Unif}(\mathbb{B}_0(r)) \\
&\quad \mathbf{y}_t \leftarrow \mathbf{x}_t + (1-\theta)\mathbf{v}_t \\
&\quad \mathbf{x}_{t+1} \leftarrow \mathbf{y}_t - \eta \nabla f(\mathbf{y}_t) \\
&\quad \mathbf{v}_{t+1} \leftarrow \mathbf{x}_{t+1} - \mathbf{x}_t \\
&\quad \textbf{if } f(\mathbf{x}_t) \le f(\mathbf{y}_t) + \langle \nabla f(\mathbf{y}_t), \mathbf{x}_t - \mathbf{y}_t \rangle - \frac{\gamma}{2}\|\mathbf{x}_t - \mathbf{y}_t\|^2 \textbf{ then} \\
&\qquad (\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) \leftarrow \text{Negative-Curvature-Exploitation}(\mathbf{x}_t, \mathbf{v}_t, s)
\end{aligned}
$$

Perturbation

鞍点にはまっていそうならノイズを乗せて脱出

加速勾配法

AGD

Negative curvature exploitation

凸っぽくなければある方法で降下方向を発見

Jin, Netrapalli, Jordan: Accelerated Gradient Descent Escapes Saddle Points Faster than Gradient Descent.

SGLD

- Stochastic Gradient Langevin Dynamics

GLD: $$X_{t+1} = X_t - \eta \nabla F(X_t) + \sqrt{2\eta\beta^{-1}}\xi_t$$ (Euler-Maruyama scheme)

$$\xi_t \sim \mathrm{N}(0, I)$$

SGLD: $$X_{t+1} = X_t - \eta \underbrace{\frac{1}{|I_B|}\sum_{i \in I_B} \nabla f_i(X_t)}_{\text{Stochastic}} + \sqrt{2\eta\beta^{-1}}\xi_t$$

[Welling and Teh, 2011]

⬇ 連続化

$$\mathrm{d}X_t = -\nabla F(X_t)\mathrm{d}t + \sqrt{2\beta^{-1}}\mathrm{d}B_t$$ (Langevin dynamics)

定常分布： $\pi \propto \exp(-\beta F(X))$

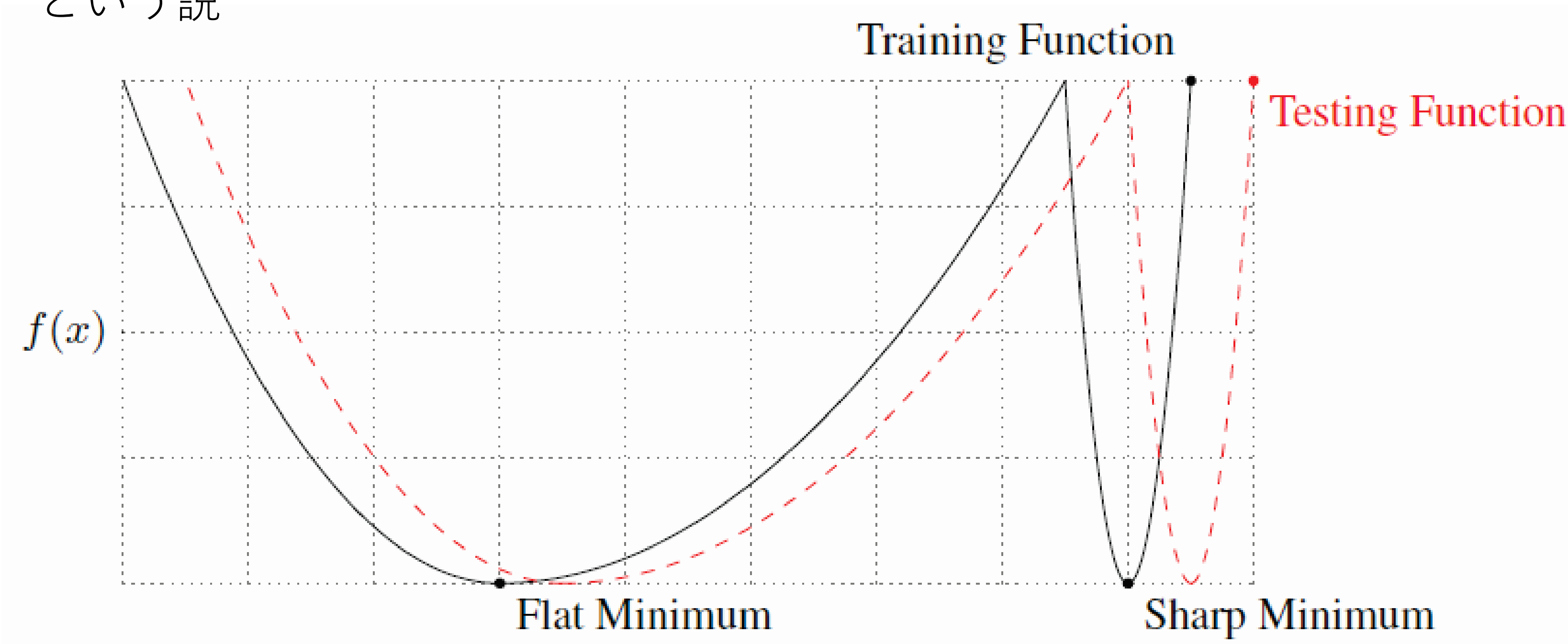
定理 [Xu et al., arXiv:1707.06618]　Fの適当な条件のもと（非凸でOK）, GLDは
（滑らか，散逸的）

$$T = O\left(\frac{de^d}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\right)$$ 回の更新で**<u>大域的最適解</u>**との誤差が ε 以下
(w.h.p.)

SGLDも多項式時間で収束.

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.

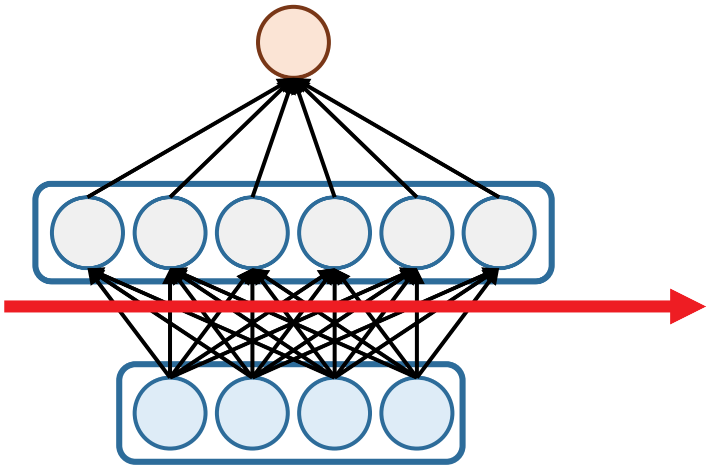
$$\theta_t = \theta_{t-1} - \underbrace{\alpha_b \left(\frac{1}{b} \sum_{j=1}^{b} \nabla_\theta \ell(z_{i_j}; \theta) \right)}_{\cong \text{正規分布}}$$

→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)

Over-parametrization

- 横幅が広いと局所最適解が大域的最適解になる.



横幅を十分広くとってランダム初期化すれば,
経験誤差0の解へ**線形収束**する.

[Du et al., 2018; Allen-Zhu, Li & Song, 2018; Li & Liang, 2018]

誤差 $\leq C' \exp(-cT)$

- 横幅が十分広いと最適解の近くに初期値が来る.
- 目的関数が凸的になる.

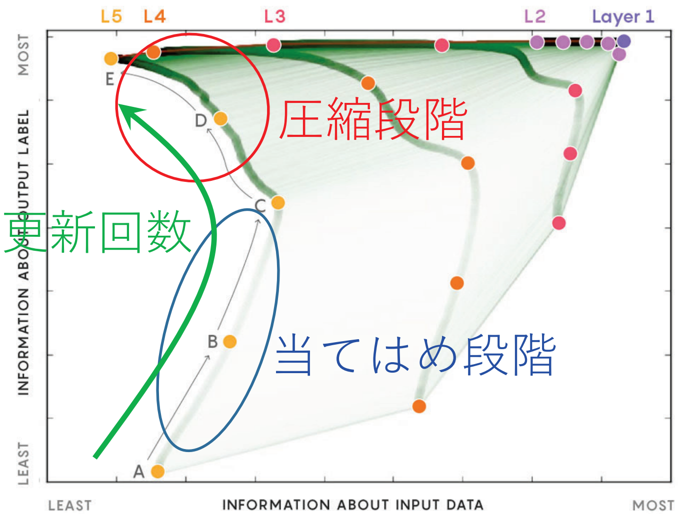
その他

Information Bottleneck

中間層と入力および出力との相互情報量

Inside Deep Learning

New experiments reveal how deep neural networks evolve as they learn.



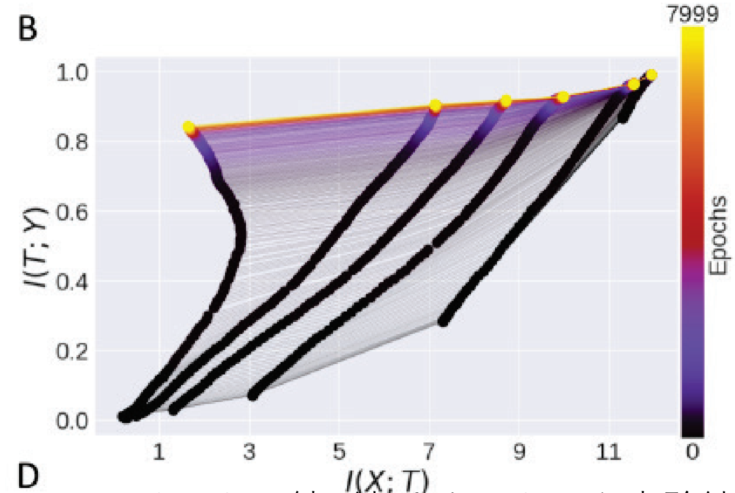
A **INITIAL STATE:** Neurons in Layer 1 encode everything about the input data, including all information about its label. Neurons in the highest layers are in a nearly random state bearing little to no relationship to the data or its label.

B **FITTING PHASE:** As deep learning begins, neurons in higher layers gain information about the input and get better at fitting labels to it.

C **PHASE CHANGE:** The layers suddenly shift gears and start to "forget" information about the input.

D **COMPRESSION PHASE:** Higher layers compress their representation of the input data, keeping what is most relevant to the output label.

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

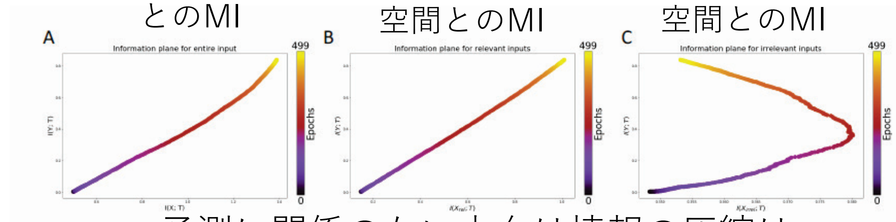


ReLUにすると圧縮が起きないという実験結果も.

入力全体とのMI

関係のある部分空間とのMI

関係のない部分空間とのMI



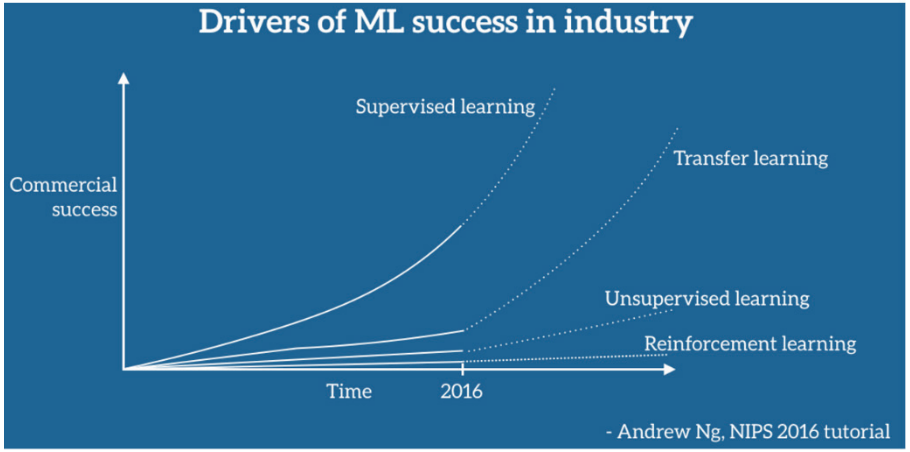
予測に関係のない方向は情報の圧縮は進んでいるという実験結果.

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.

転移学習・メタ学習

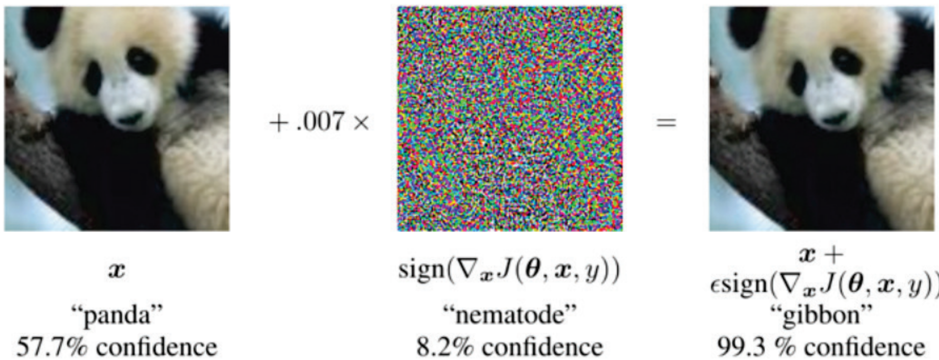
いかに少ないデータで学習するか？
大量のデータで学習しておき，興味のある問題にその知識を「転移」させる．

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, <u>深層学習の信頼性評価はまだ難しい</u>

社会的課題

- 少数データの問題
  - 転移学習・メタ学習・ドメイン適合
  - シミュレーション・CG
- 深層学習の信頼性保証
  - 敵対的入力へのロバスト性
- 学習結果の解釈可能性 (accountability)
- プライバシー保護
- 差別的振る舞いの抑制 (fairness)

まとめ

- 深層学習のモデル
  - 畳み込みネットワーク
  - ResNet
- 各種応用
  - 物体認識・物体検出・マスキング
  - スタイル変換
  - 自然言語処理・Q&Aタスク
  - 生成モデル
- 深層学習の理論
  - 表現力は深くすることで指数的に増大
  - 最適化理論
    - オーバーパラメトライズ・ノイズ付加(SGD)で大域的最適解へ