iTHEMS RIKEN interdisciplinary Theoretical & Mathematical Sciences

RIKEN

Introduction to Deep Learning

深層学習の理論的な基礎とその様々な応用

Masato Taki RIKEN, iTHEMS

2017.5/10

@RIKEN

AI

IoT

ビックデータ

シンギュラリティ

AI

ビックデータ

!?

IoT

シンギュラリティ

これら良く分からないものの話はしません

AI

ビックデータ

!?

IoT

シンギュラリティ

こういうバズワードを議論したがる雰囲気を醸成しているのが**深層学習**の飛躍的な進歩です

Glorious Debut of Deep Learning

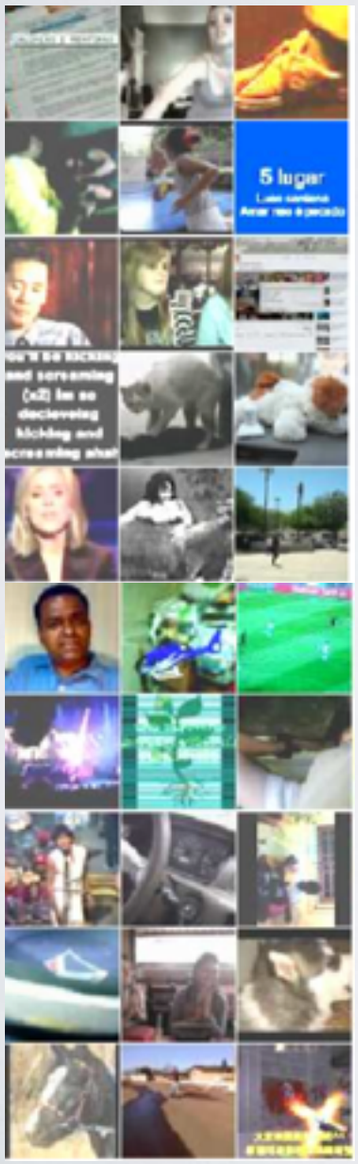
2012年： 二つのエポックメイキングな出来事

① Google's cat (グーグルの猫) の発見

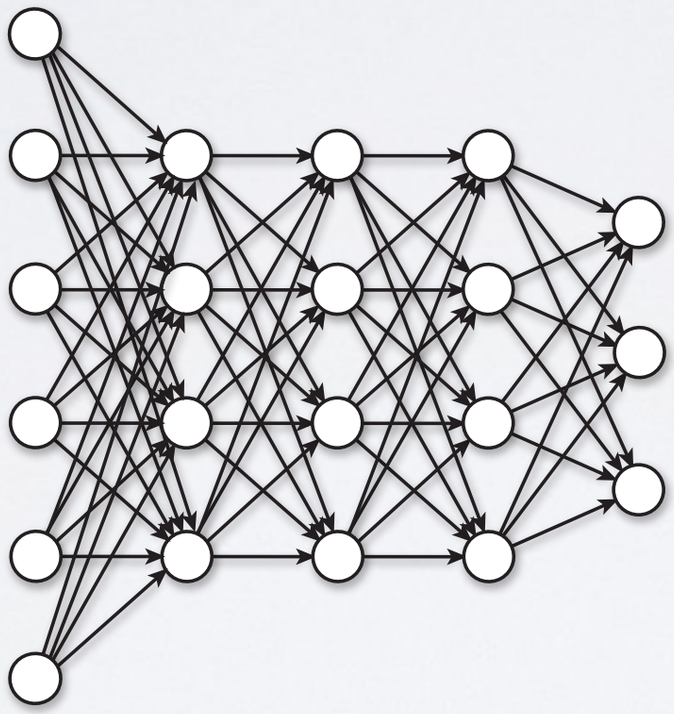
② G.Hinton達のAlexNetがImageNet LSVRC-2012コンペティションで優勝

1. Google's Cat [Google's team(Dean, Ng, et al), 2012]

10 million images
from youtube



16000 processor array

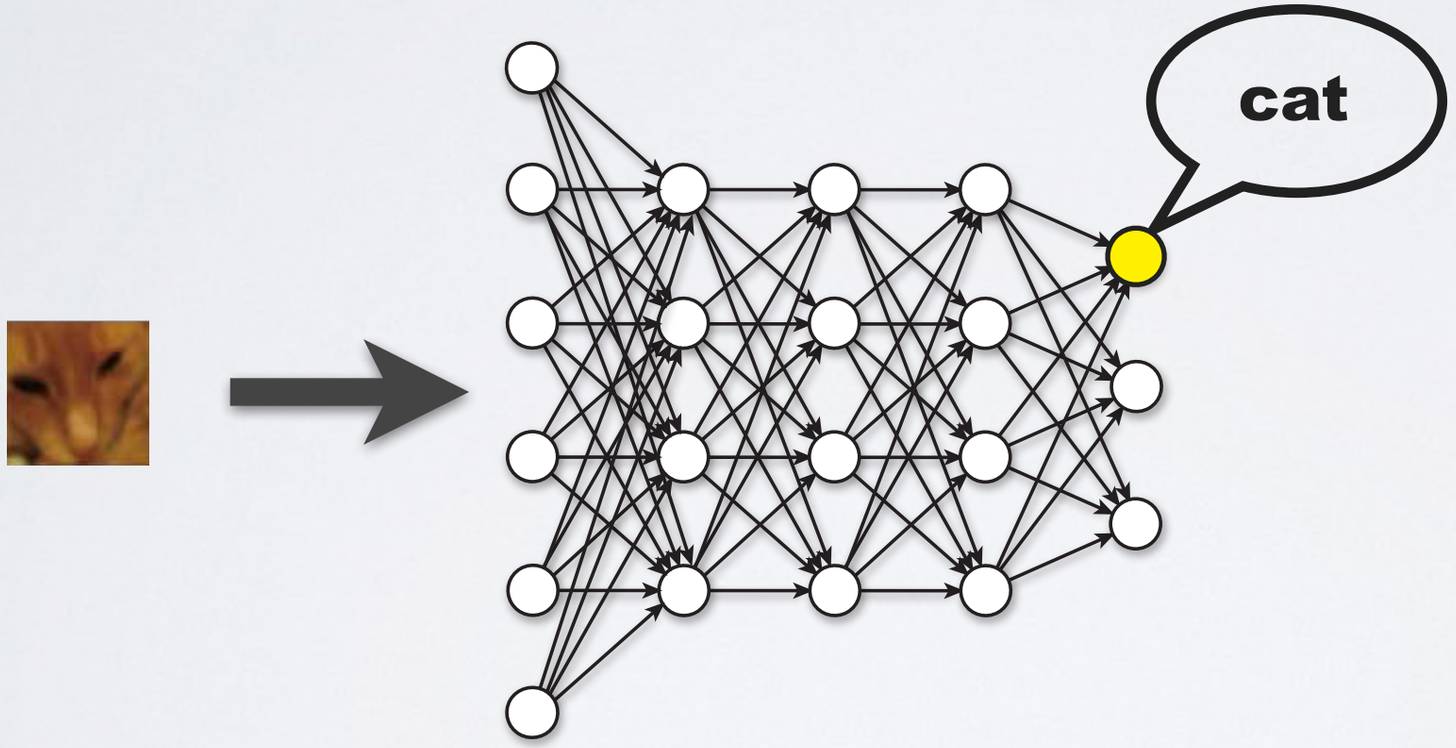


(convolutional net
+ autoencoder)

recognize many objects

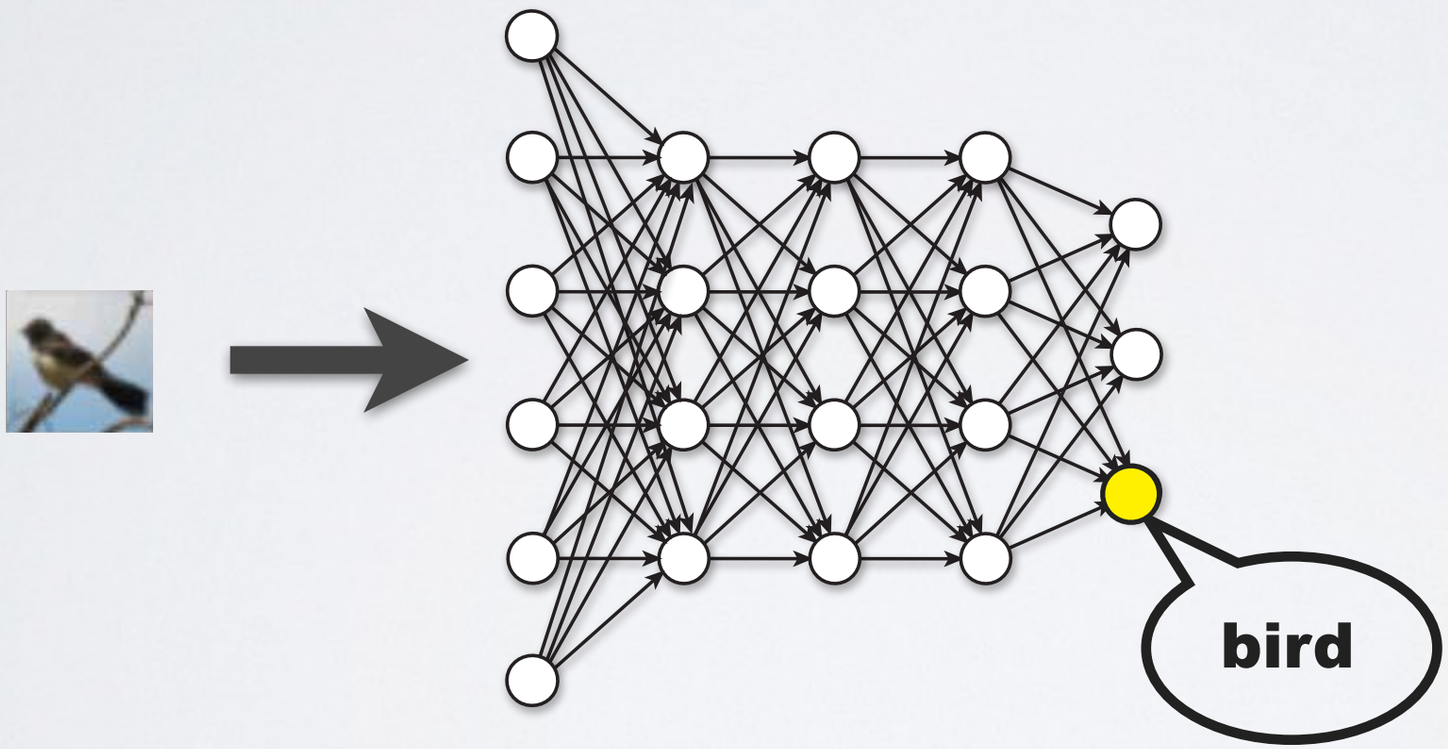
1. Google's Cat [Google's team(Dean, Ng, et al), 2012]

Very high classification performance



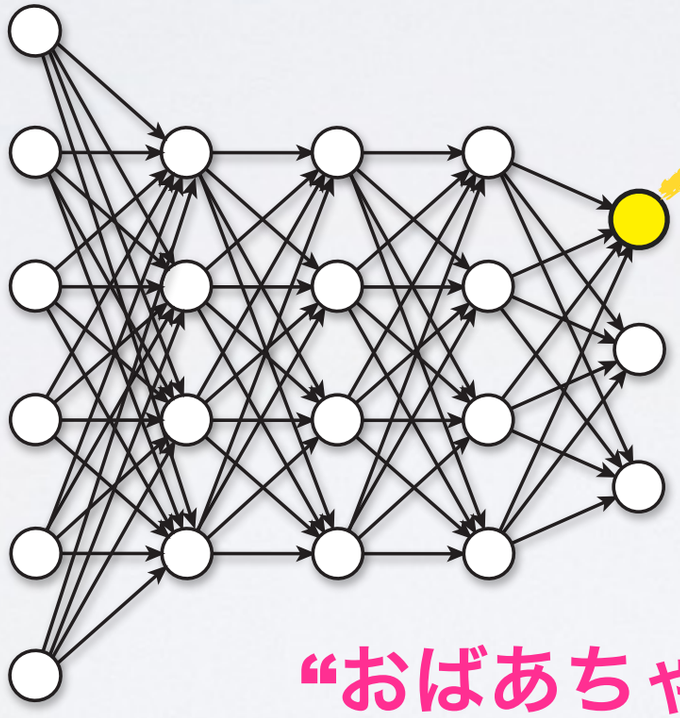
1. Google's Cat [Google's team(Dean, Ng, et al), 2012]

Very high classification performance



1. Google's Cat [Google's team(Dean, Ng, et al), 2012]

reconstructing `pre-image' of corresponding output signal, we can see the system acquired "abstract concept" (deep feature)



"おばあちゃん細胞"

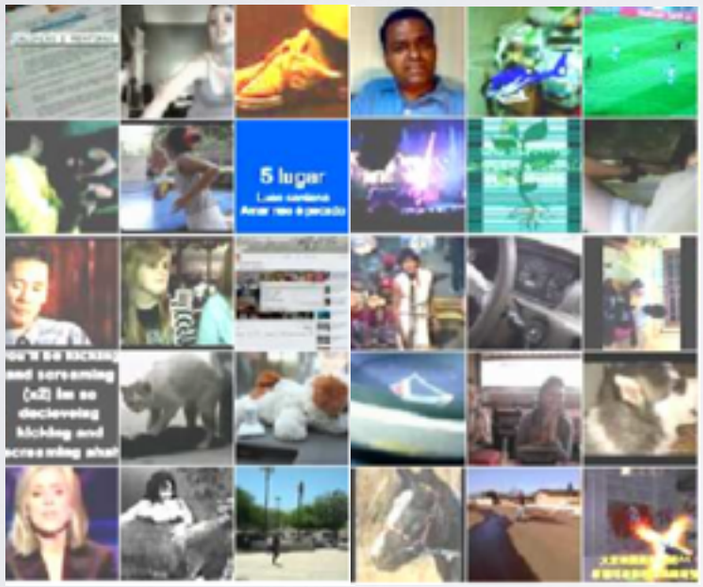
Google's cat

No understanding why & how it works so well

1. 機械学習の基礎

Aim of machine learning (supervised case)

‘observed’ data



explaining how they generated

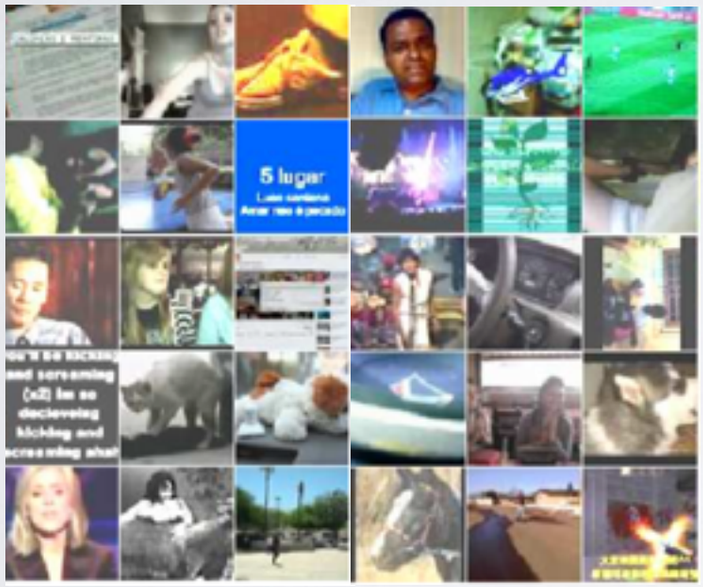
classifying into categories

extracting essences of them

etc

Aim of machine learning (supervised case)

‘observed’ data



explaining how they generated

classifying into categories

extracting essences of them

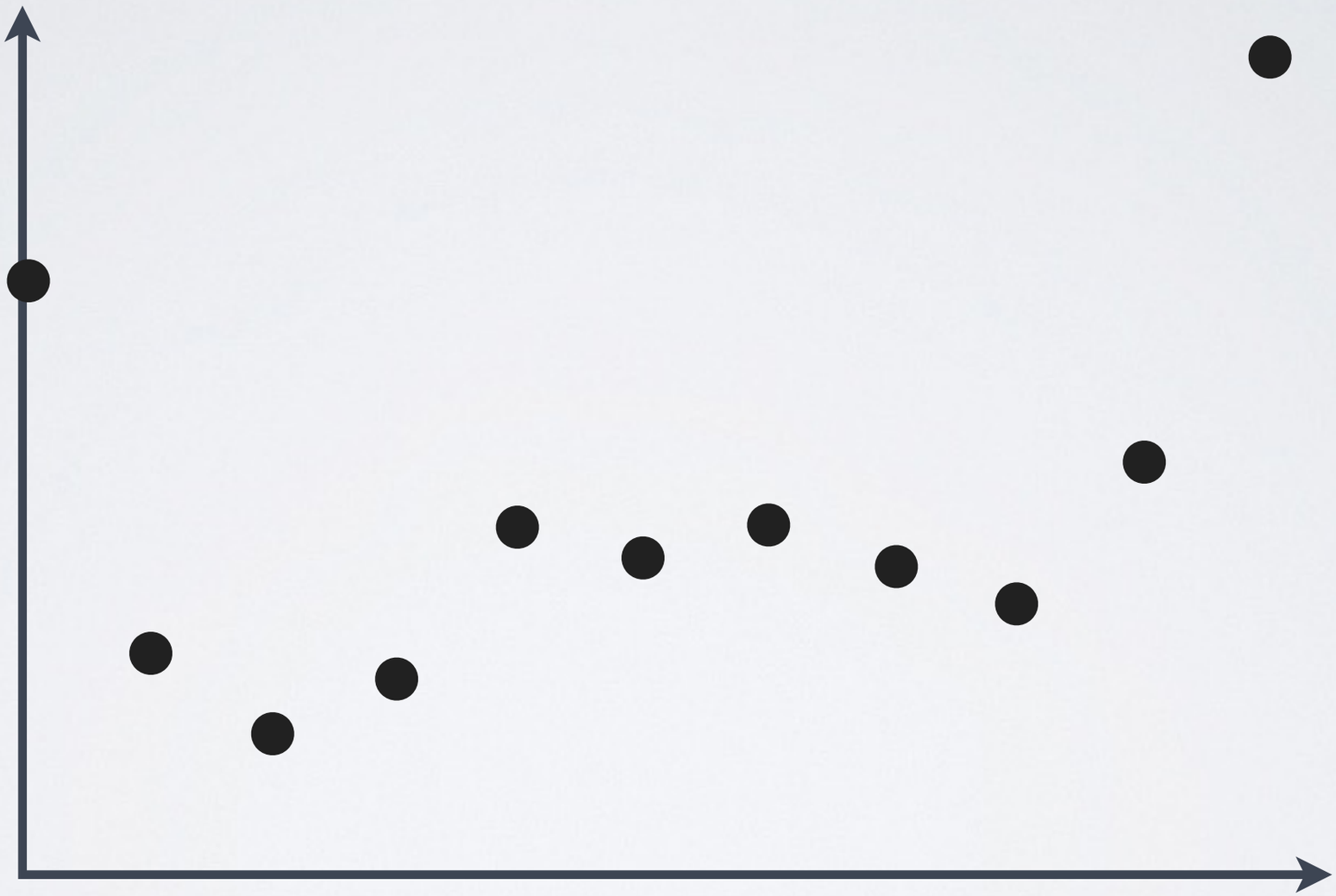
etc

a kind of fitting of ‘observed’ data

未知の世界から法則性／パターンを抽出する作業

Aim of machine learning ((un)supervised case)

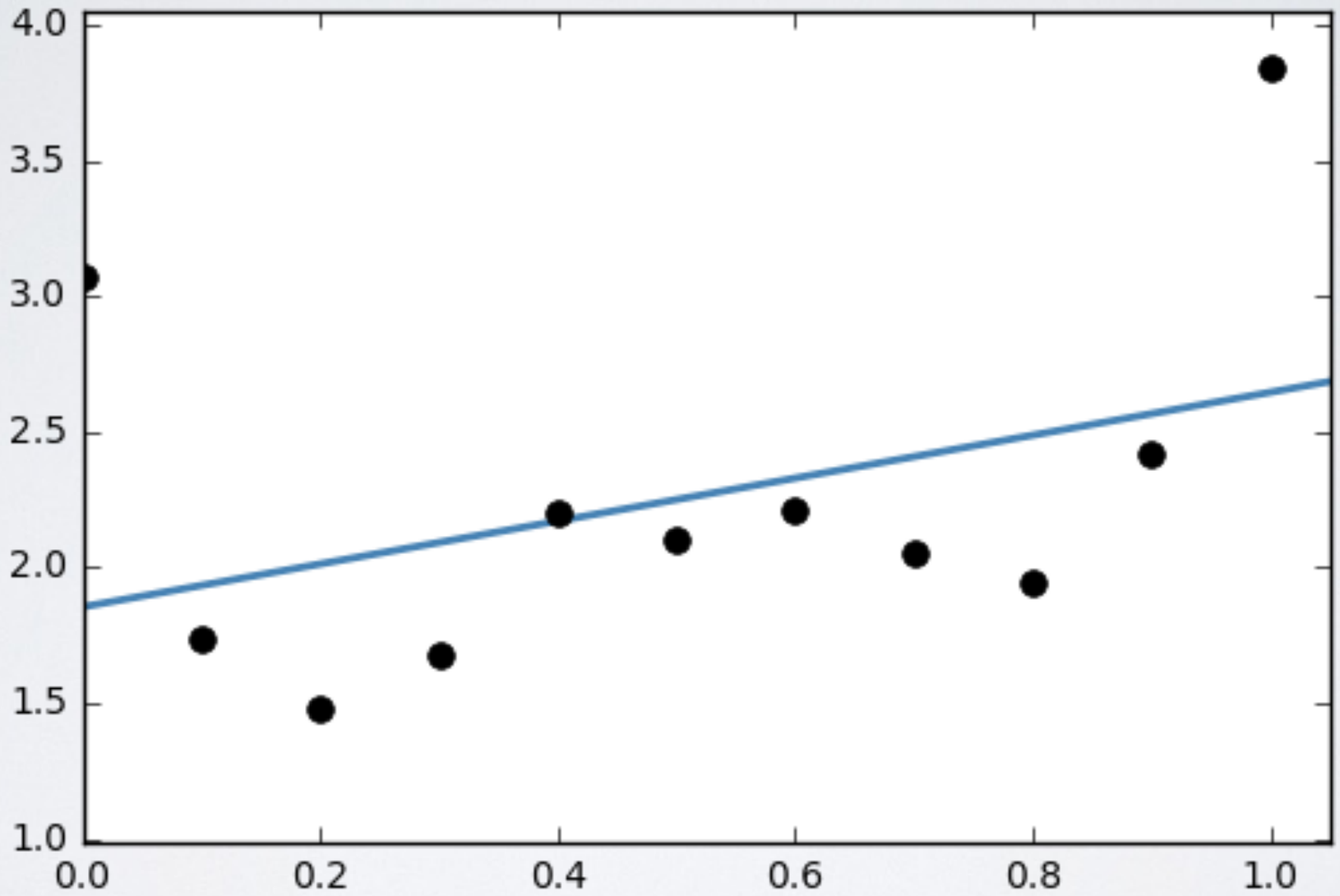
observed data points



a kind of fitting of ‘observed’ data

Aim of machine learning ((un)supervised case)

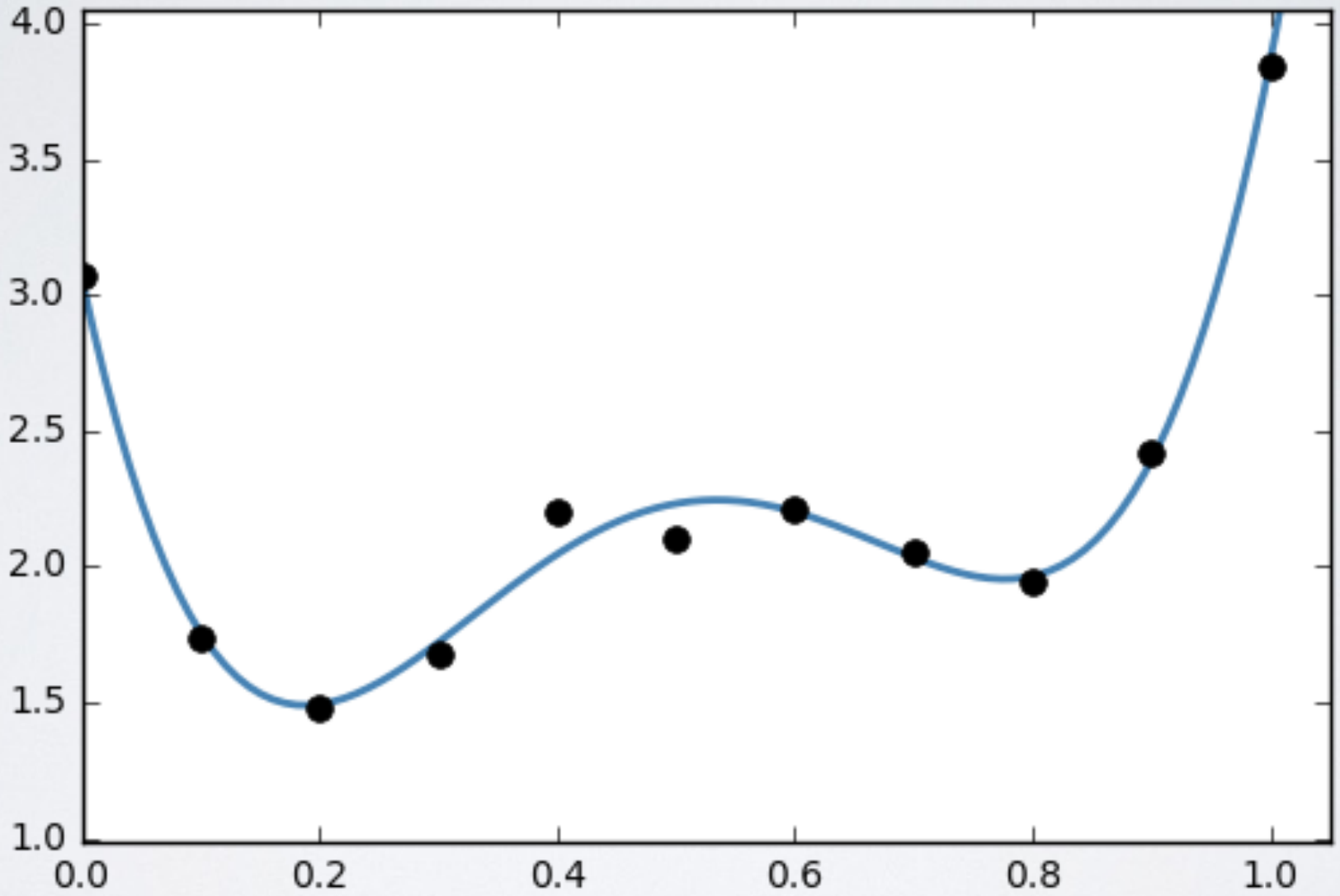
未学習 (パラメータが少なすぎる)



poor performance

Aim of machine learning ((un)supervised case)

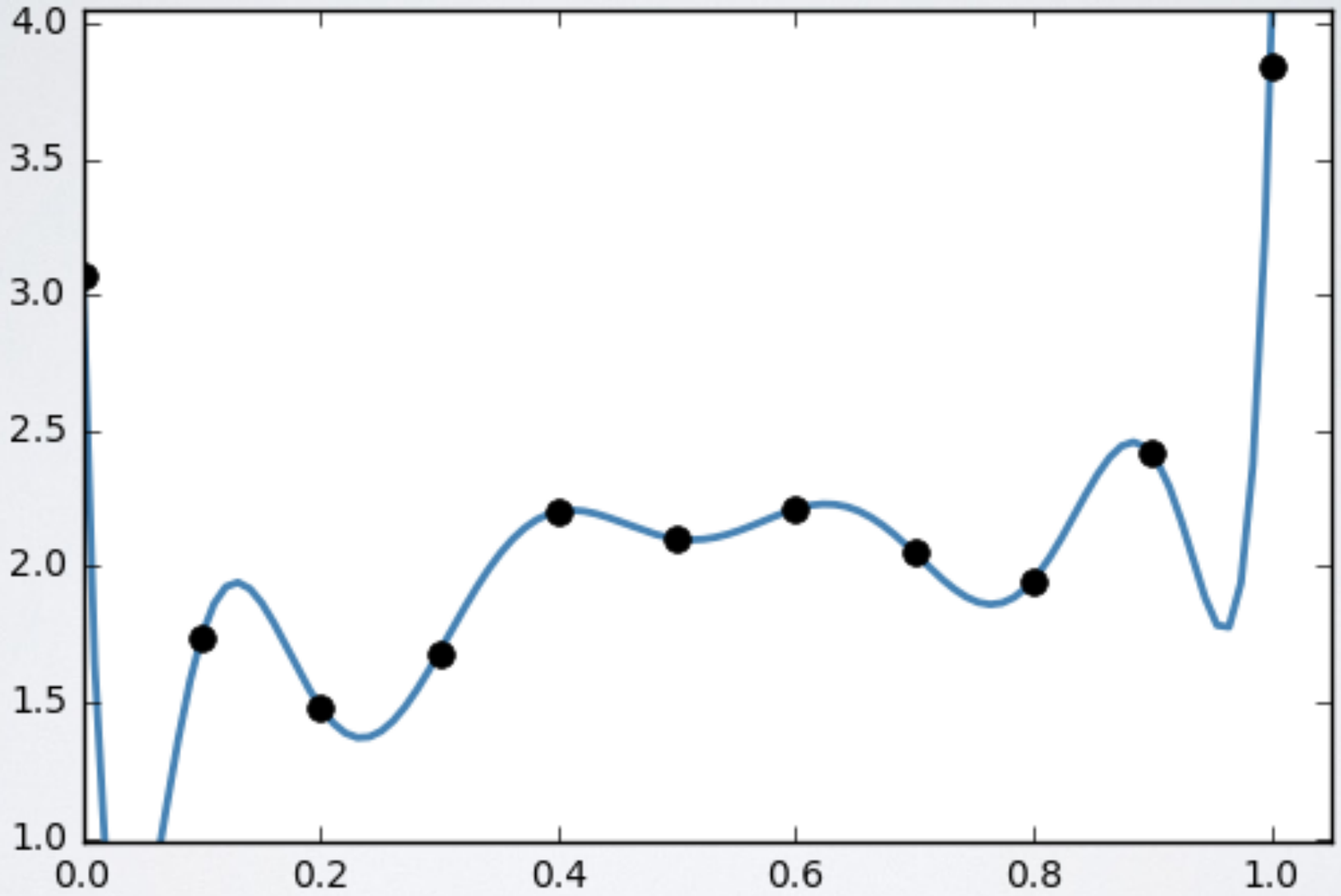
良いフィッティング (道のデータ点もよく予測)



it makes reliable prediction

Aim of machine learning ((un)supervised case)

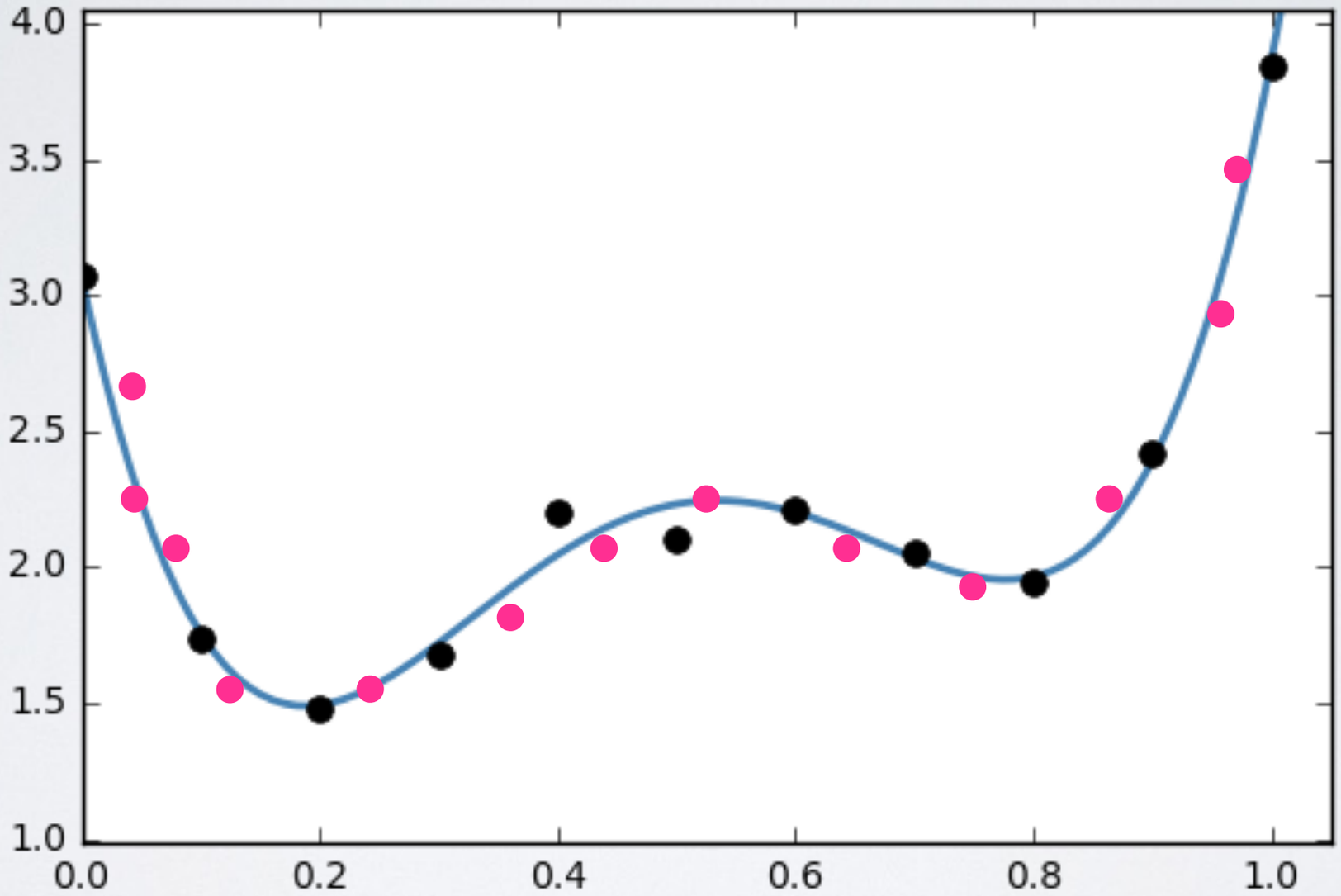
過学習 overfitting (パラメータが多すぎ/データが少なすぎ)



no reliable prediction

Aim of machine learning ((un)supervised case)

まだ観測されていないデータまでうまく説明したい！

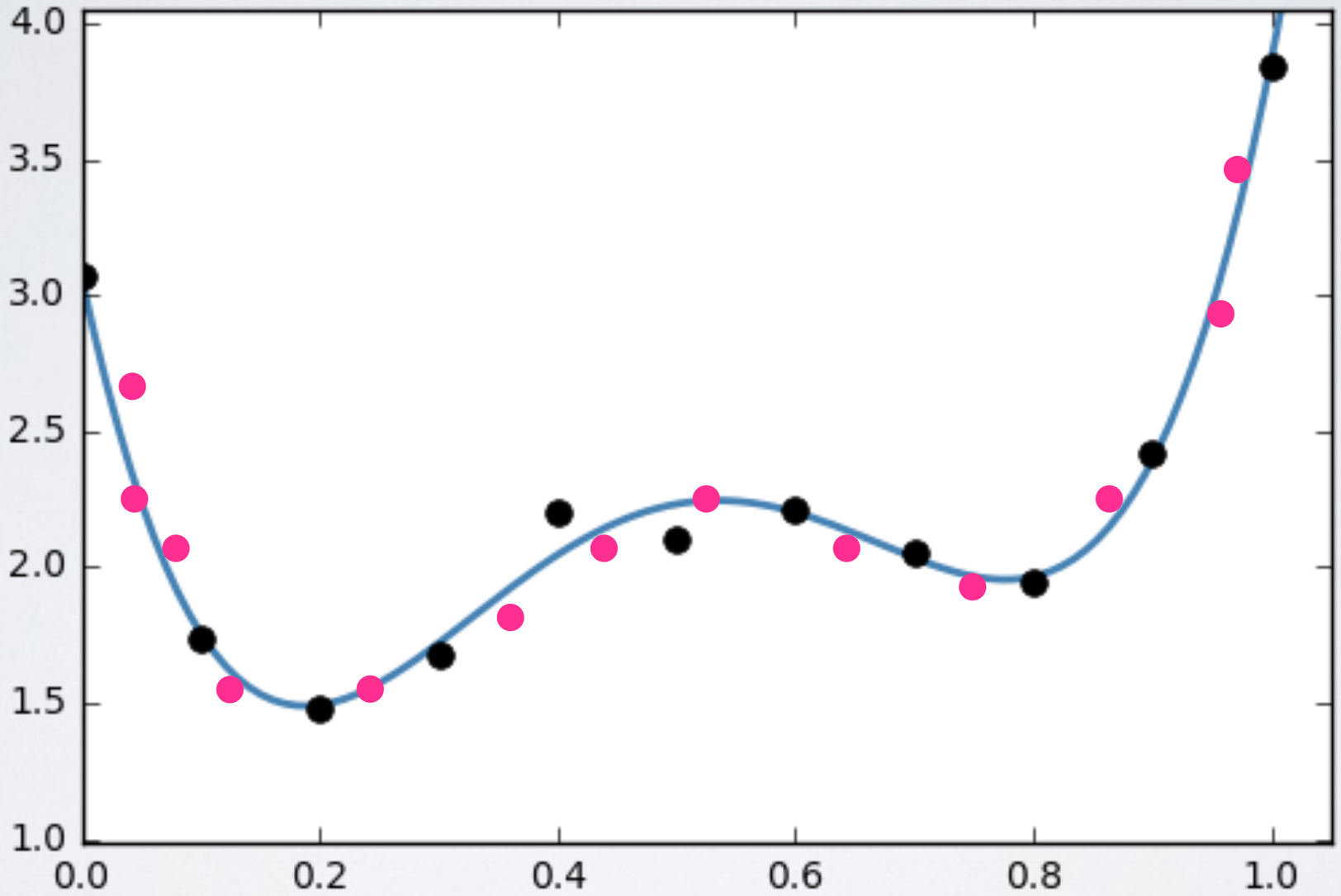


巨大なデータを大きなモデルでうまくフィッティングさせること

汎化性能の実現

Aim of machine learning ((un)supervised case)

まだ観測されていないデータまでうまく説明したい！



汎化性能の為の技術（正則化）を後で紹介

Finding hyper parameter

Basics of machine learning

正解ラベル付き訓練データ集合

$$(\vec{x}^{(n)}, \vec{y}^{(n)})$$

$$n = 1, 2, \cdots, N$$

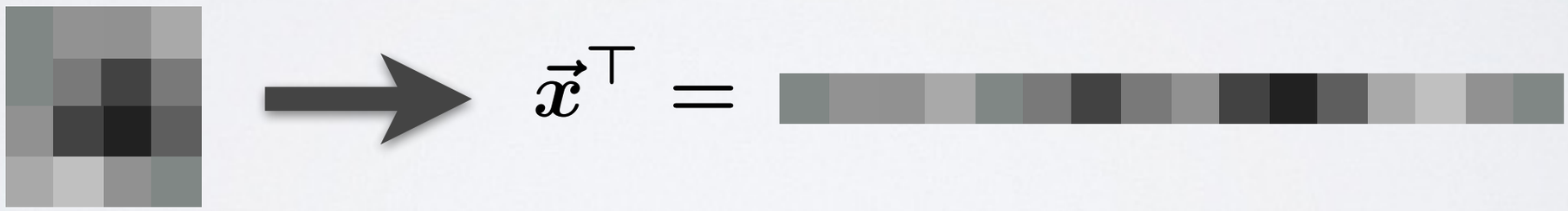
Basics of machine learning

正解ラベル付き訓練データ集合

$$(\vec{x}^{(n)}, \vec{y}^{(n)})$$

$$n = 1, 2, \cdots, N$$

data



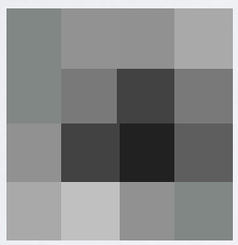
Basics of machine learning

正解ラベル付き訓練データ集合

$$(\vec{x}^{(n)}, \vec{y}^{(n)})$$

$$n = 1, 2, \cdots, N$$

data



$$\vec{x}^{\top} =$$

attribution(label): answer

猫成分　　犬

$$\vec{y}^{\top} = (0, 0, 0, 1, 0, 0, 0, 0, 0)$$

1-of-K 表現 ／ one-hot ベクトル

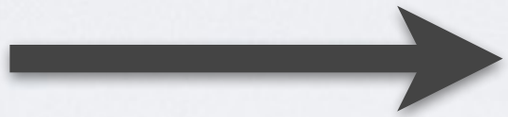
Basics of machine learning: regression

訓練データ

$$(\vec{x}^{(n)}, \vec{y}^{(n)})$$

$$n = 1, 2, \cdots, N$$

推定



$$\vec{y} = \vec{y}(\vec{x})$$

Basics of machine learning: regression

訓練データ

$$(\vec{x}^{(n)}, \vec{y}^{(n)})$$

$$n = 1, 2, \cdots, N$$

推定

$$\vec{y} = \vec{y}(\vec{x})$$

))

パラメトリックモデル

$$\vec{y}(\vec{x}; \vec{w})$$

Basics of machine learning: regression

訓練データ

$$(\vec{x}^{(n)}, \vec{y}^{(n)})$$

$$n = 1, 2, \cdots, N$$

推定

$$\vec{y} = \vec{y}(\vec{x})$$

))

パラメトリックモデル

$$\vec{y}(\vec{x}; \vec{w})$$

Basics of machine learning: regression

訓練データ

$$(\vec{x}^{(n)}, \vec{y}^{(n)})$$

$$n = 1, 2, \cdots, N$$

推定

$$\vec{y} = \vec{y}(\vec{x})$$

))

パラメトリックモデル

$$\vec{y}(\vec{x}; \vec{w})$$

推定の誤差を最小化するパラメータを決める(最適化)

$$\sum_{n=1}^{N} \left(\vec{y}(\vec{x}^{(n)}; \vec{w}) - \vec{y}^{(n)} \right)^2$$

回帰以外のタスク

logistic regression ロジスティック回帰

データxがクラスy=1とクラスy=0のどちらに属するのかを判別
(ここではone-hotとせずに直接y=1,0でクラスを表すものとします)

$$P(y=1|x)=\sigma(u),\quad P(y=0|x)=\sigma(-u)$$

$$\sigma(u)=\frac{e^u}{1+e^u}$$ シグモイド関数

logistic regression ロジスティック回帰

データ**x**がクラス**y=1**とクラス**y=0**のどちらに属するのかを判別

$$P(y=1|x) = \sigma(u), \quad P(y=0|x) = \sigma(-u)$$

$$\sigma(u) = \frac{e^u}{1+e^u}$$ シグモイド関数

$$u = \boldsymbol{w}^\top \boldsymbol{x} + b$$

logistic regression ロジスティック回帰

最尤法で学習

$$E = -\log \prod_{n=1}^{N} \left(P(y=1|x^{(n)})\right)^{k^{(n)}} \left(P(k=0|x^{(n)})\right)^{1-k^{(n)}}$$

これを多クラスに拡張したものが**ソフトマックス回帰**

様々なデータ

さまざまなdata sets

王道タスクである画像認識(クラス分類)に関心を絞りましょう。

さまざまなdata sets

王道タスクである画像認識(クラス分類)に関心を絞りましょう。

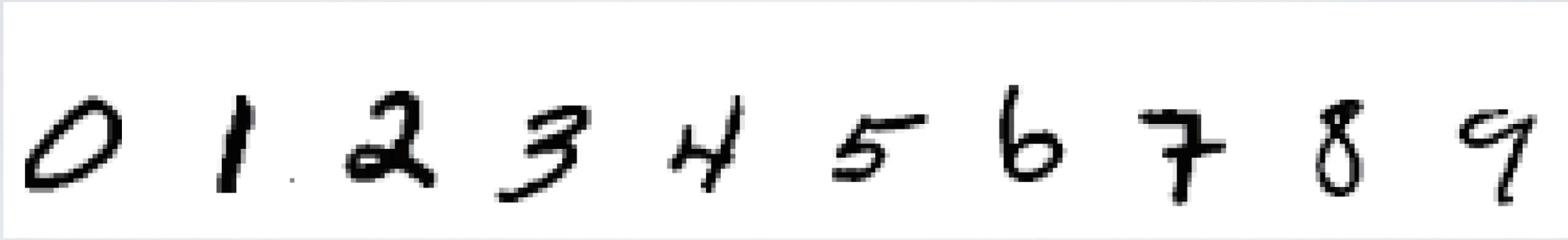
1.手書き数字認識

簡単であるが機械学習の本質が詰まったタスク

MNISTデータセットを用いる

LeCunのホームページから入手可

MNISTデータセット



アメリカ国立標準技術研究所(国立標準局、NIST)が、国勢調査員と高校生から集めた計7万枚の手書き数字データ。機械学習における``Hello World``

バイナリー形式(ビッグエンディアン)で配布

さまざまなdata sets

王道タスクである画像認識(クラス分類)に関心を絞りましょう。

1.手書き数字認識

MNISTデータセットを用いる

2.自然画像認識

機械学習モデルによる画像認識のベンチマークテスト

実用上とても重要

いろいろなデータセット

ImageNet

ImageNet

1400万枚の画像を2万クラスにラベル付け

（人手で！！）

ImageNet

1400万枚の画像を2万クラスにラベル付け

（人手で！！）

2010年からこれを用いたILSVRCというコンテストが開かれている

Caltech101

さすがにImageNetはプロ仕様。もっと穏やかなデータセットは？

Caltech101

さすがにImageNetはプロ仕様。もっと穏やかなデータセットは？

Caltech101: カリフォルニア工科大のFeiFei Li(李飛飛)らが提供

101カテゴリーから計9146枚の画像.

ただしサイズは揃っていない

Caltech101

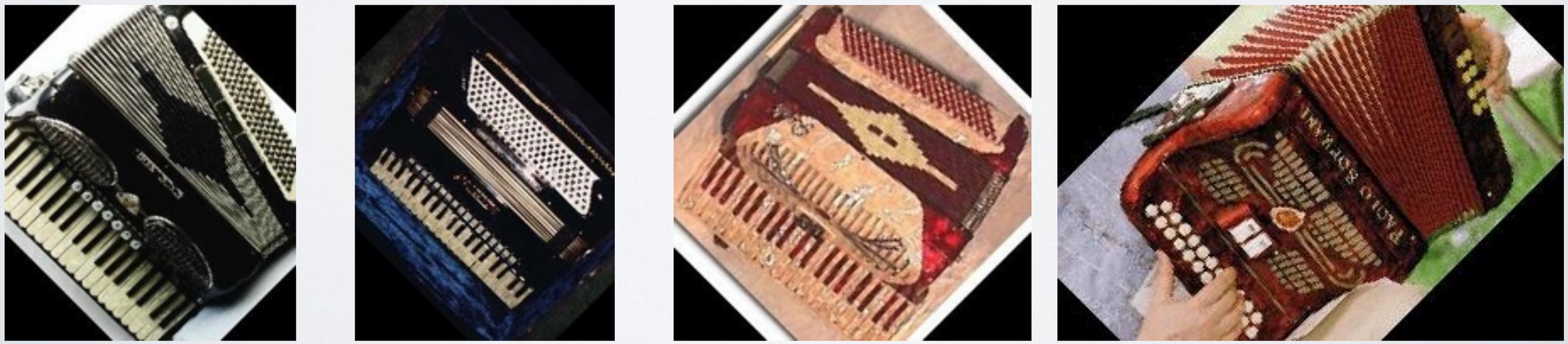
さすがにImageNetはプロ仕様。もっと穏やかなデータセットは？

Caltech101: カリフォルニア工科大のFeiFei Li(李飛飛)らが提供

101カテゴリーから計9146枚の画像.

ただしサイズは揃っていない

クラス1: accordion



Caltech101

さすがにImageNetはプロ仕様。もっと穏やかなデータセットは？

Caltech101: カリフォルニア工科大のFeiFei Li(李飛飛)らが提供

101カテゴリーから計9146枚の画像.

ただしサイズは揃っていない

クラス101: yin-yang



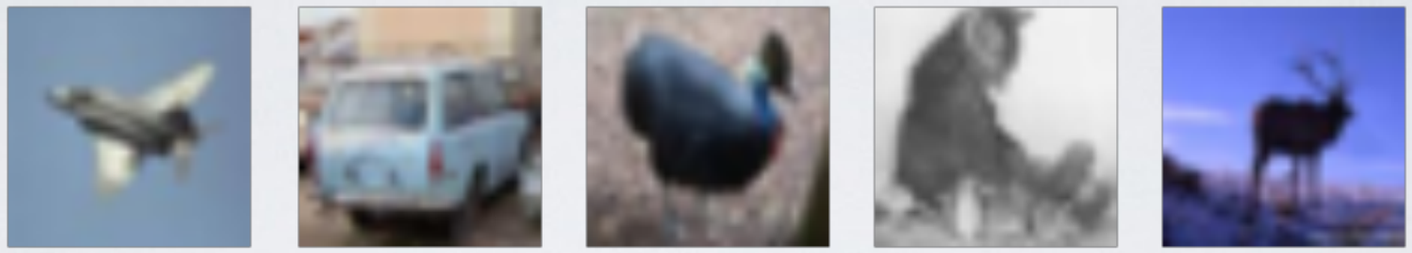

CIFAR-10

CIFAR-10: Google Mountain ViewのAlex Krizhevskyが提供

AlexNetのAlex！

CIFAR-100もあります

CIFAR-10



10クラスの32x32ピクセルカラー画像が約60000枚

ある意味MNISTに似ている

さまざまなdata sets

王道タスクである画像認識(クラス分類)に関心を絞りましょう。

1.手書き数字認識　　MNIST

2.自然画像認識　　ImageNet, CIFAR-10, Caltech101, ...

「猫やら飛行機やらを機械学習したいんじゃない！！」と思われるかもしれませんが、転移学習のところでこれらデータセット(で訓練したモデル)が役に立つことが分かります

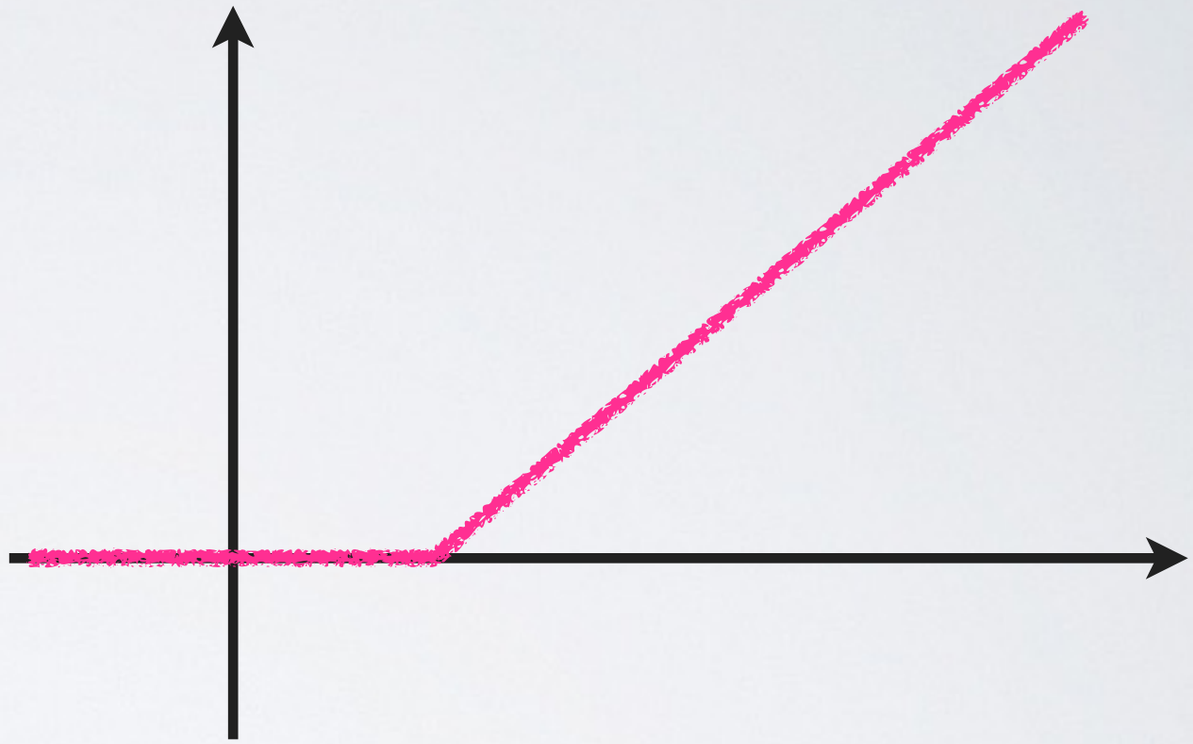
いずれにせよ十分なサイズの訓練データを用意することはとてもコストがかかるので、既存のものをうまく組み合わせることが重要！

2. ニューラルネットと深層学習

neuron system

neuron system

数理モデル化



閾値

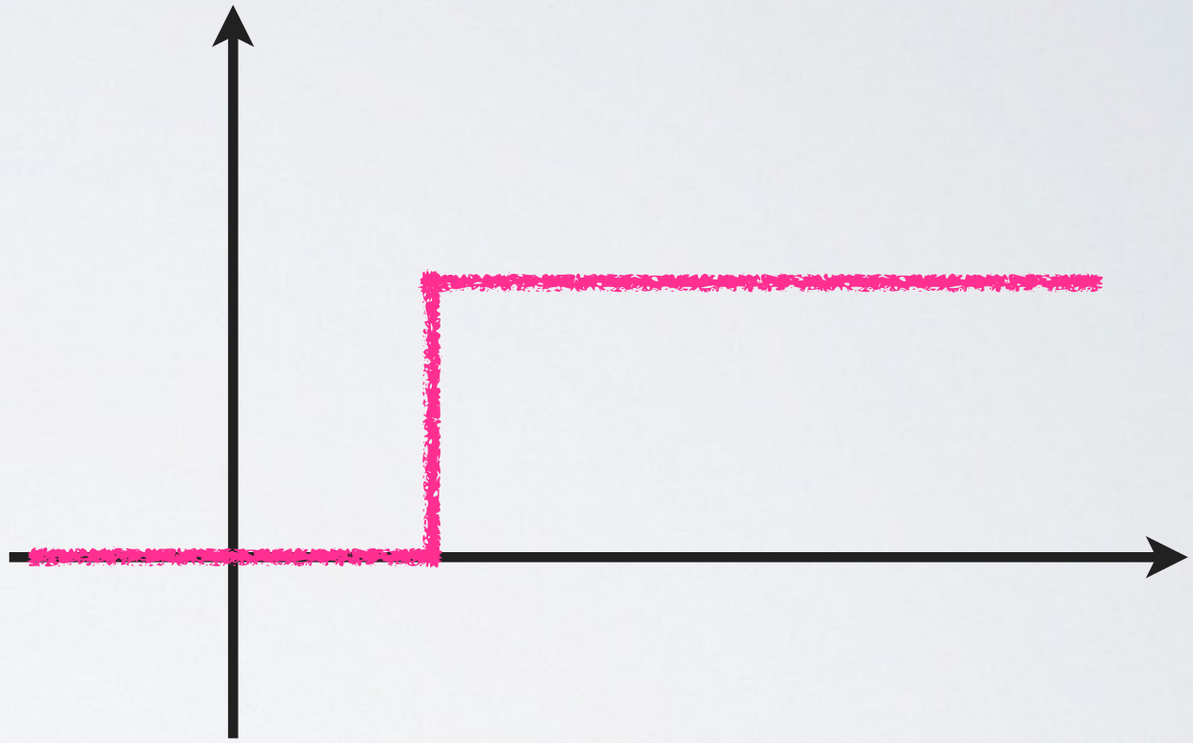
活性化関数

LeRU

neuron system

数理モデル化

閾値



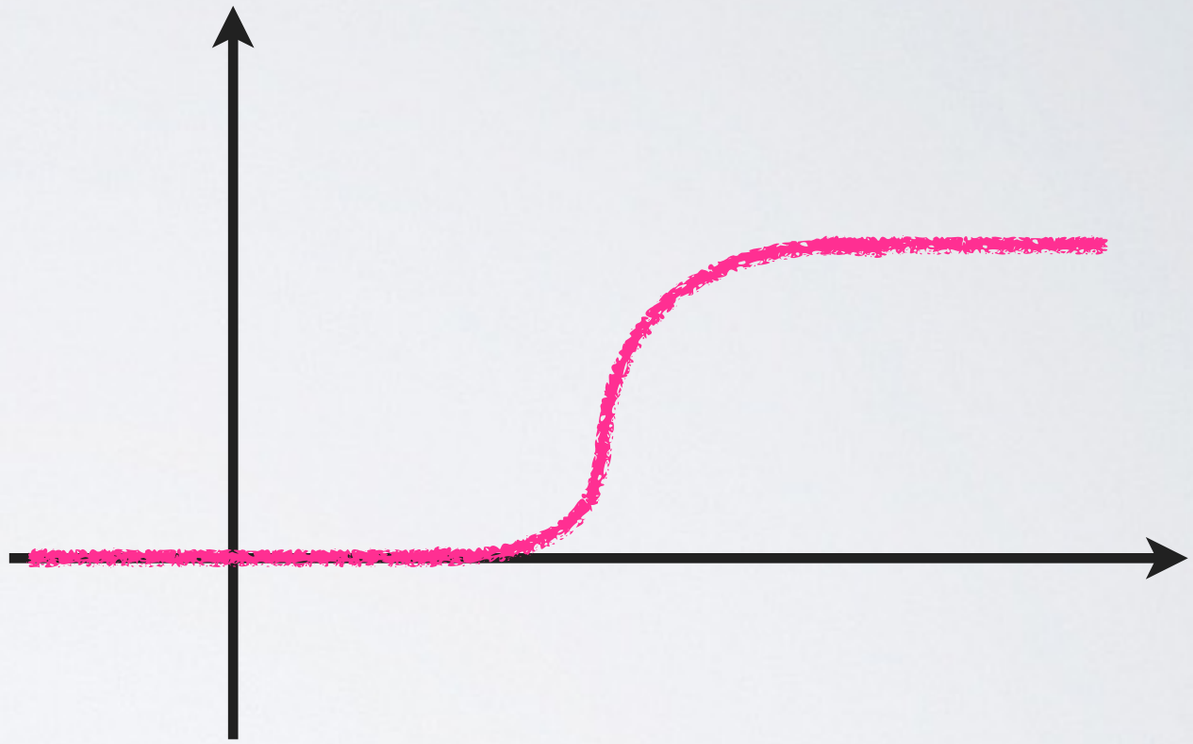
活性化関数

Heviside's step fn.

neuron system

数理モデル化

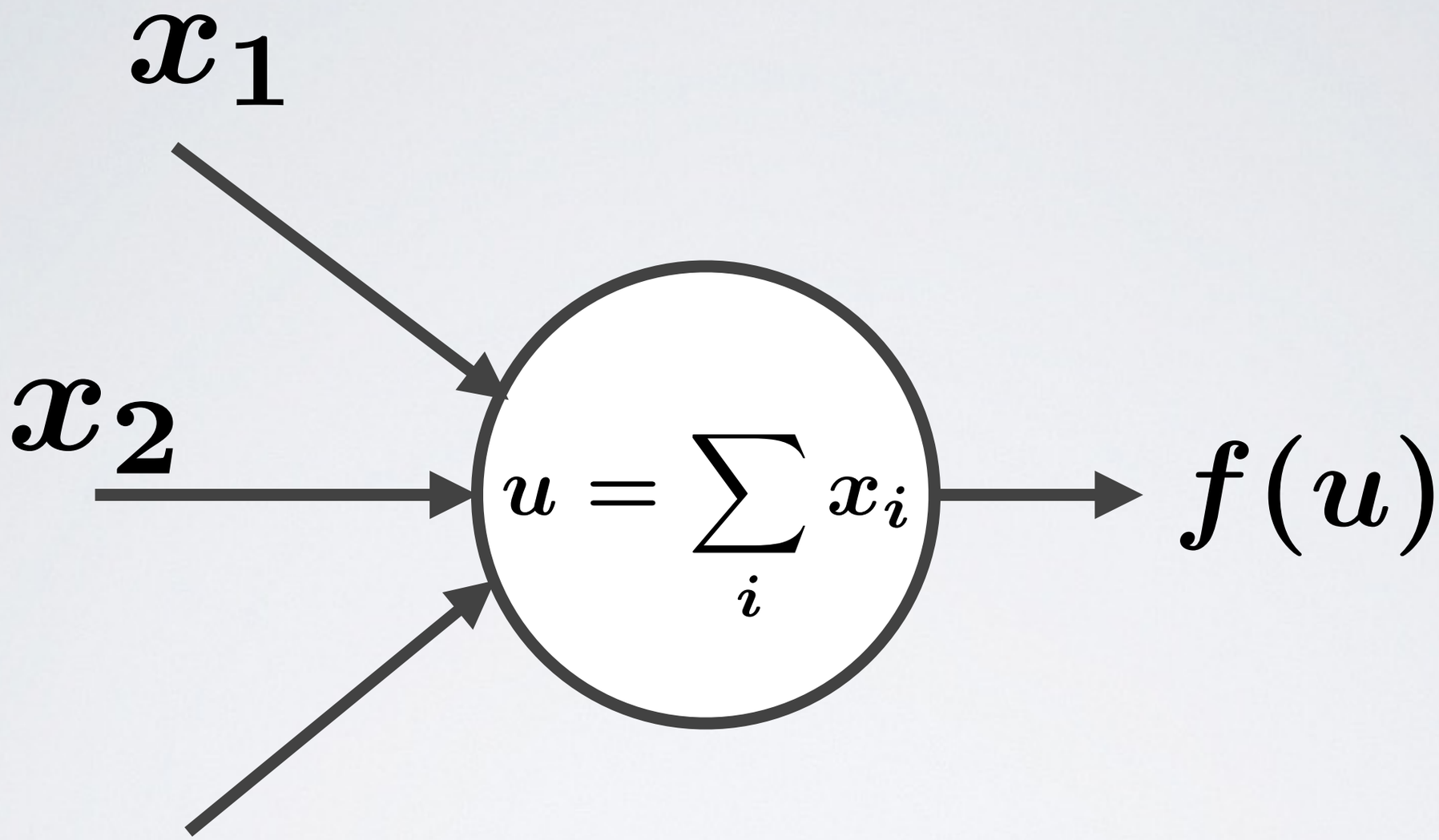
閾値



活性化関数

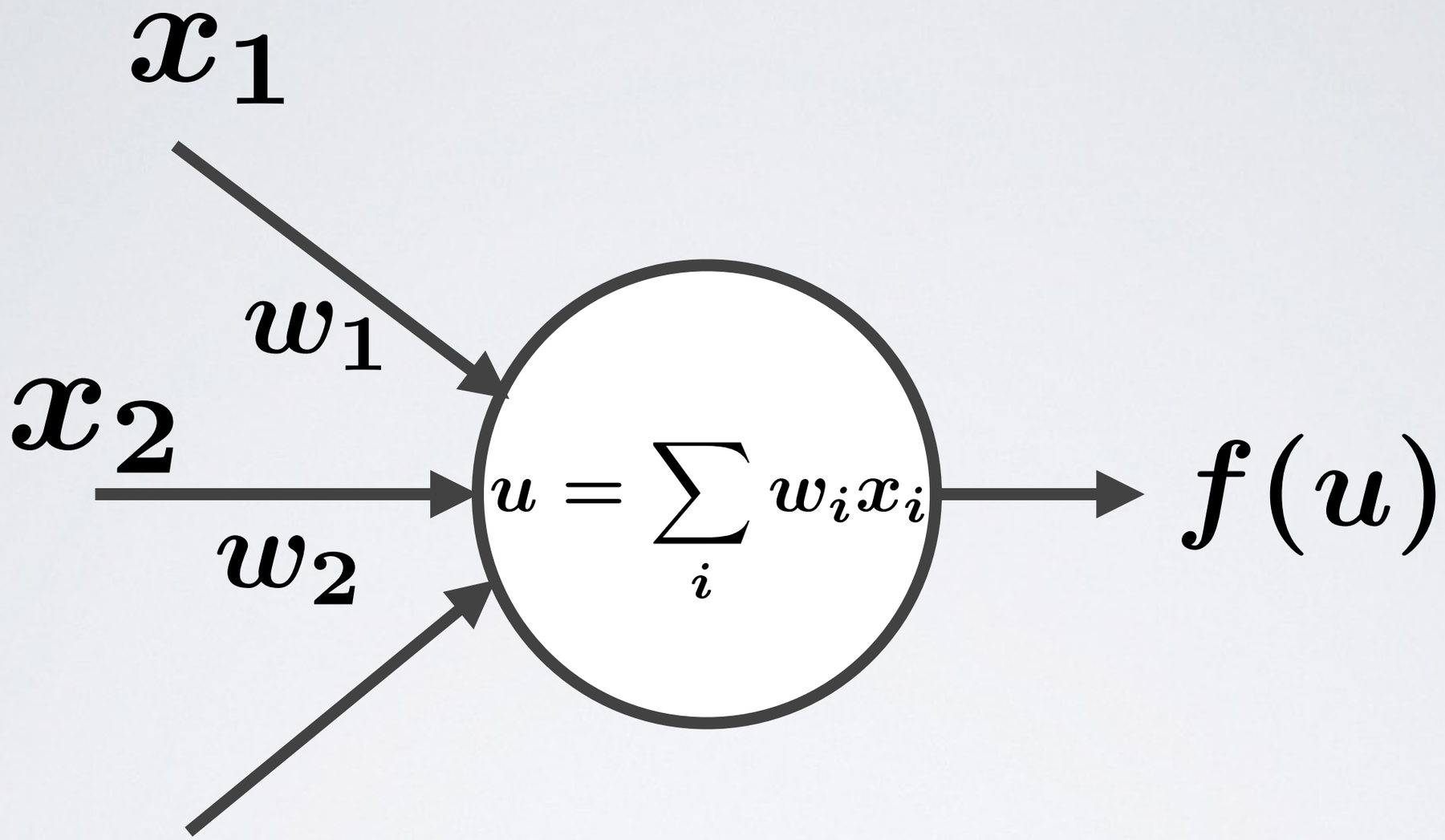
sigmoid

McCulloch-Pitts formal neuron (1943)



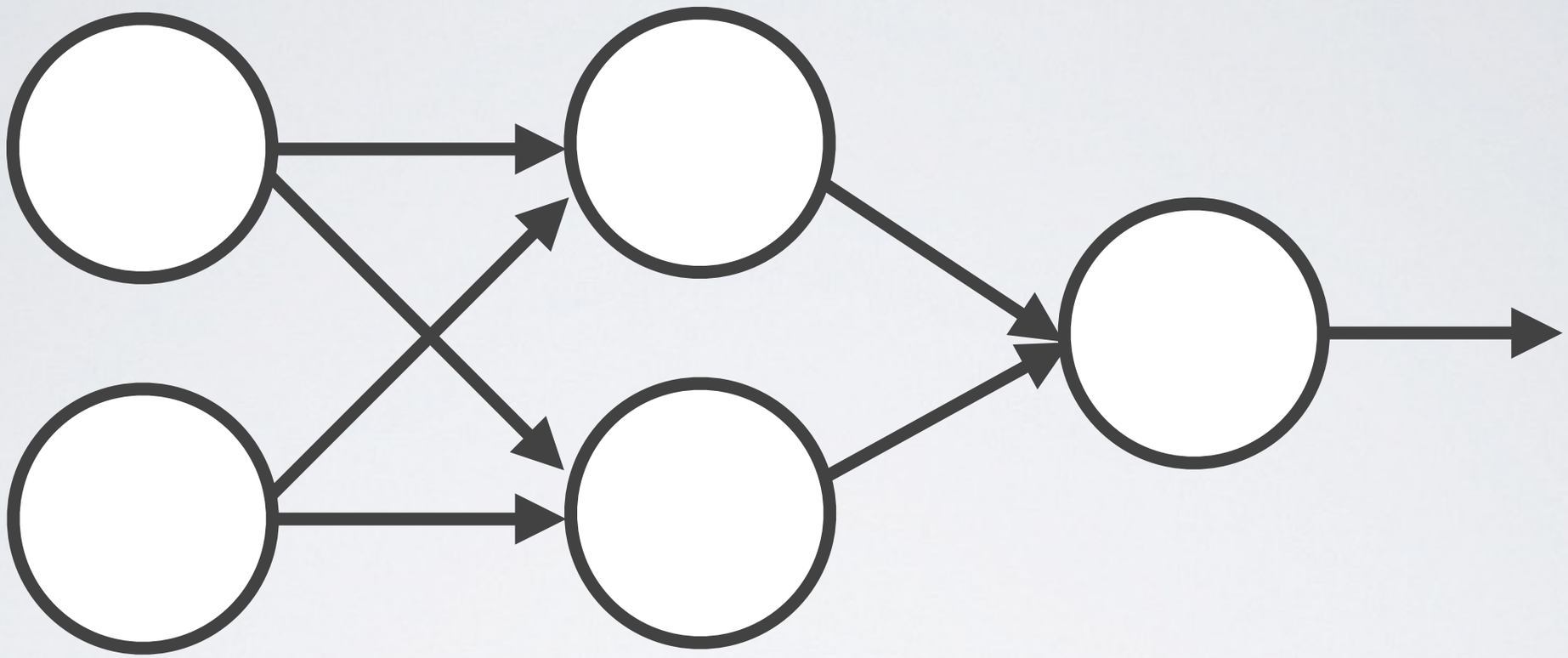
これを素子として、任意の論理演算が実現可能

Rosenblatt's perceptron (1957)



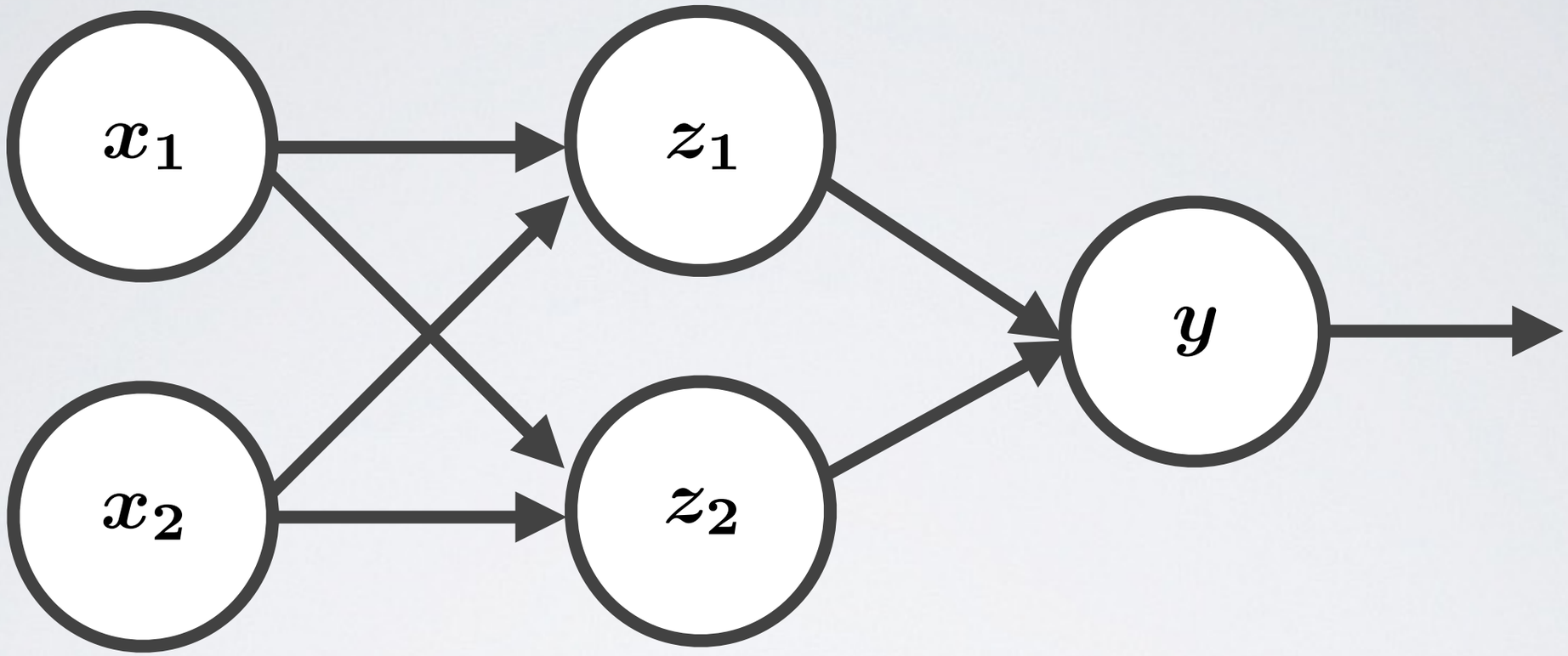
調整可能な結合重みパラメータを導入

Rosenblatt's perceptron (1957)



ニューラルネット

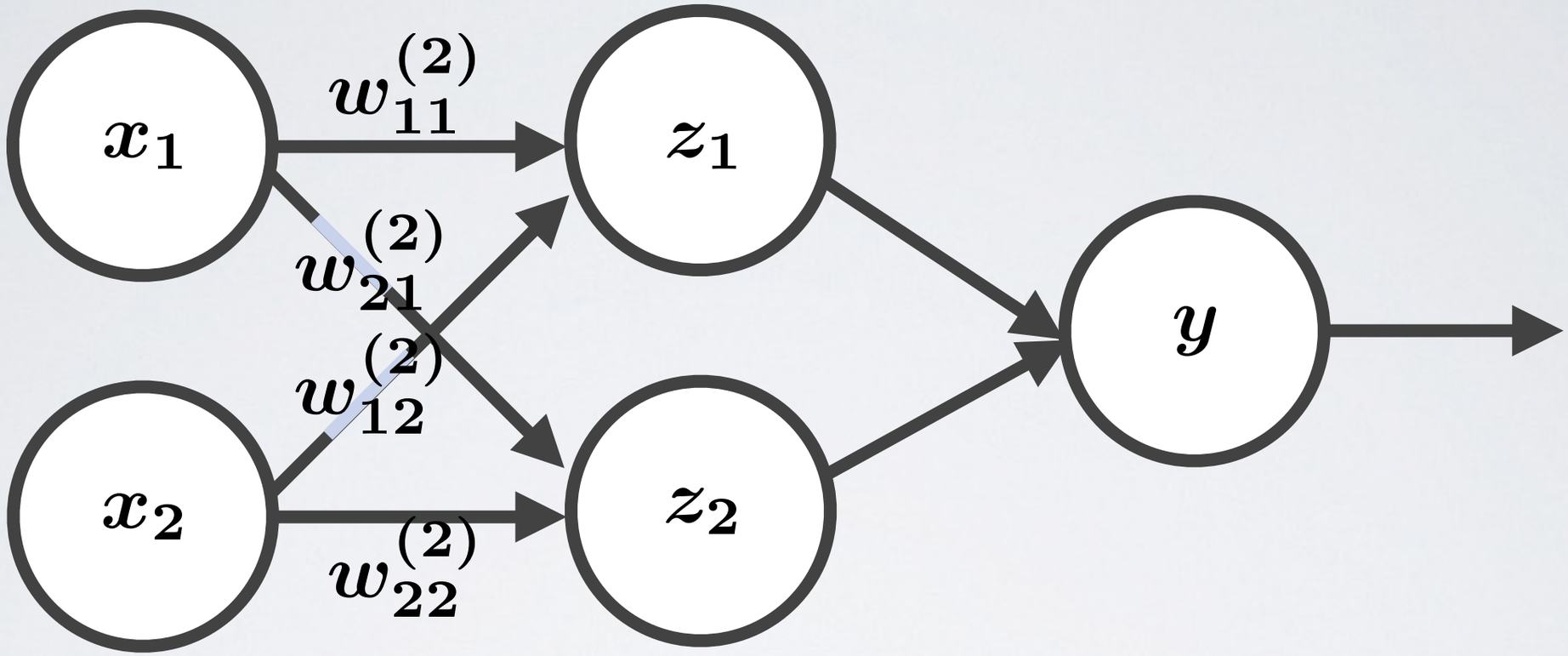
Rosenblatt's perceptron (1957)



$$z_1 = f\left(w_{11}^{(2)} x_1 + w_{12}^{(2)} x_2\right)$$

$$z_2 = f\left(w_{21}^{(2)} x_1 + w_{22}^{(2)} x_2\right)$$

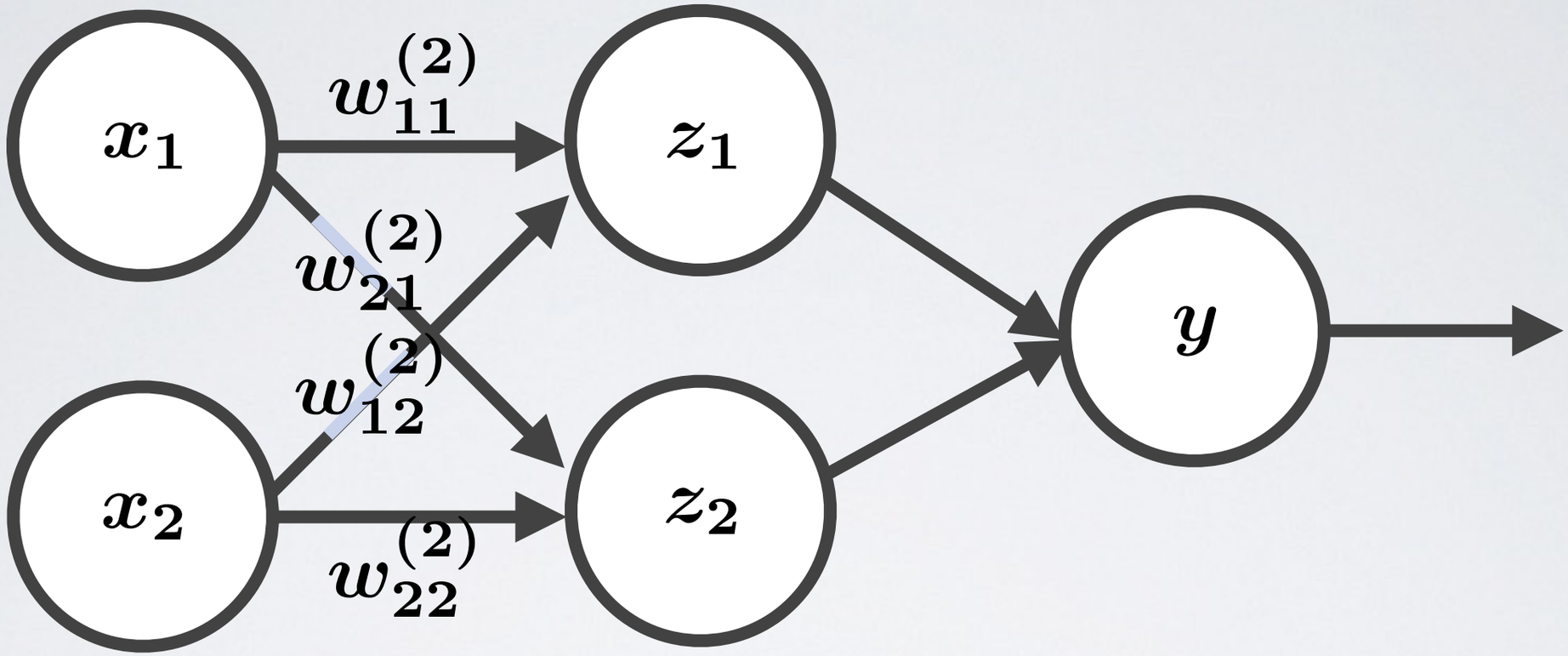
Rosenblatt's perceptron (1957)



$$z_1 = f\left(w_{11}^{(2)} x_1 + w_{12}^{(2)} x_2\right)$$

$$z_2 = f\left(w_{21}^{(2)} x_1 + w_{22}^{(2)} x_2\right)$$

Rosenblatt's perceptron (1957)

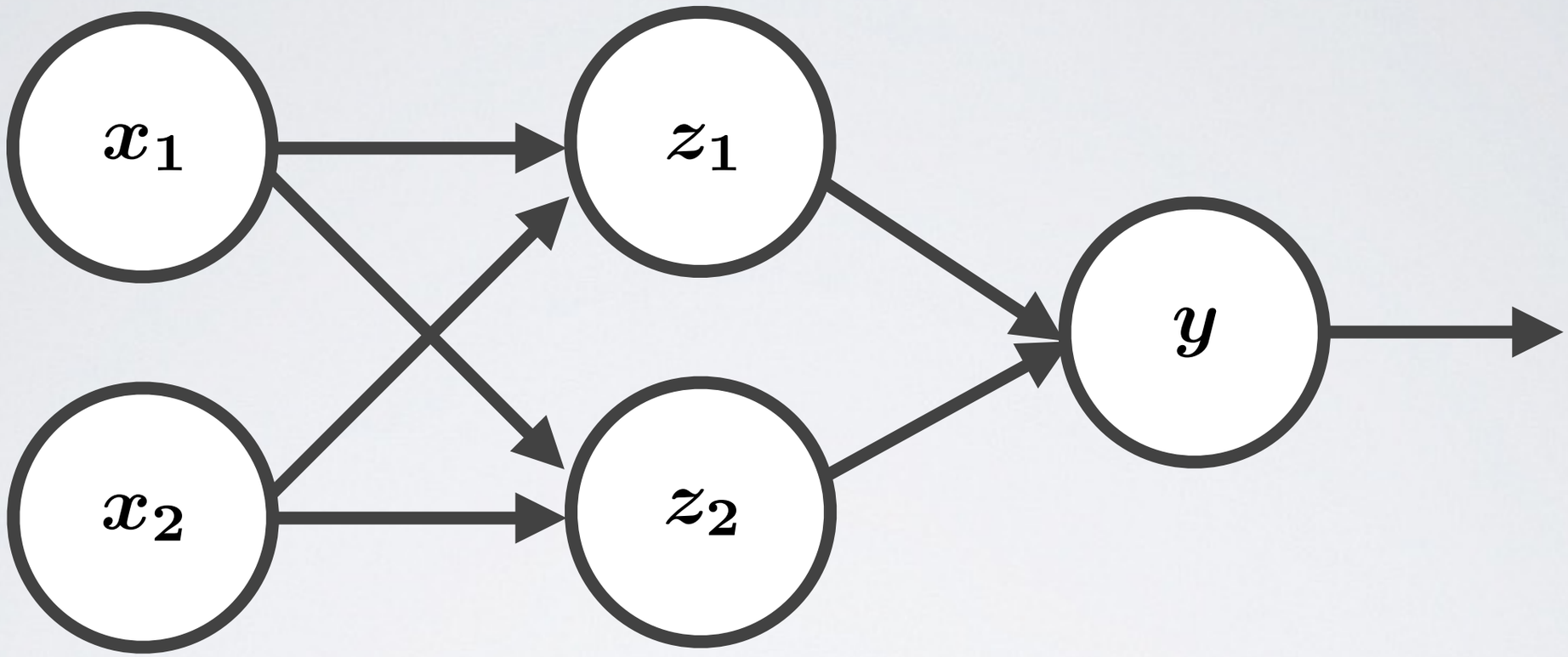


$$z_1 = f\left(w_{11}^{(2)} x_1 + w_{12}^{(2)} x_2\right)$$

$$z_2 = f\left(w_{21}^{(2)} x_1 + w_{22}^{(2)} x_2\right)$$

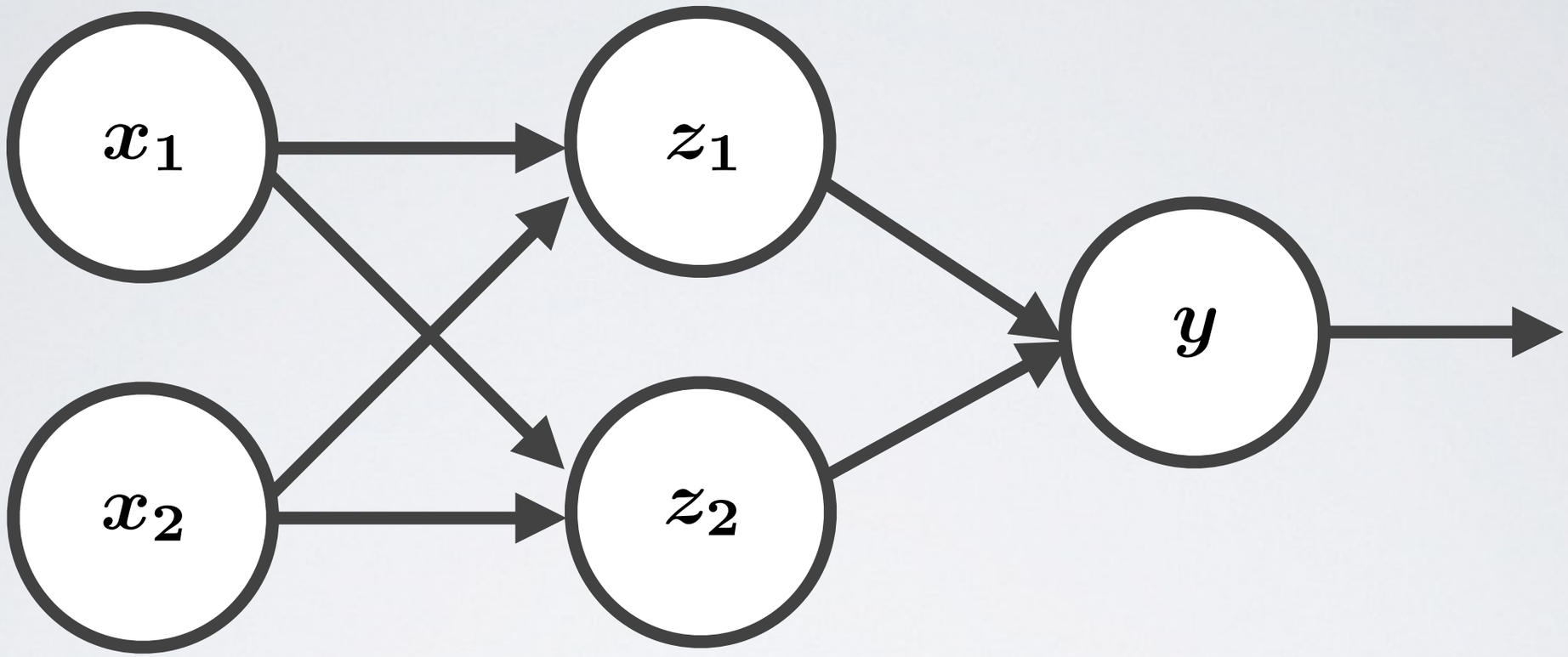
$$\vec{z} = f\left(W^{(2)} \vec{x}\right)$$

Rosenblatt's perceptron (1957)



$$y = f\left(w_{11}^{(3)} z_1 + w_{12}^{(3)} z_2\right)$$

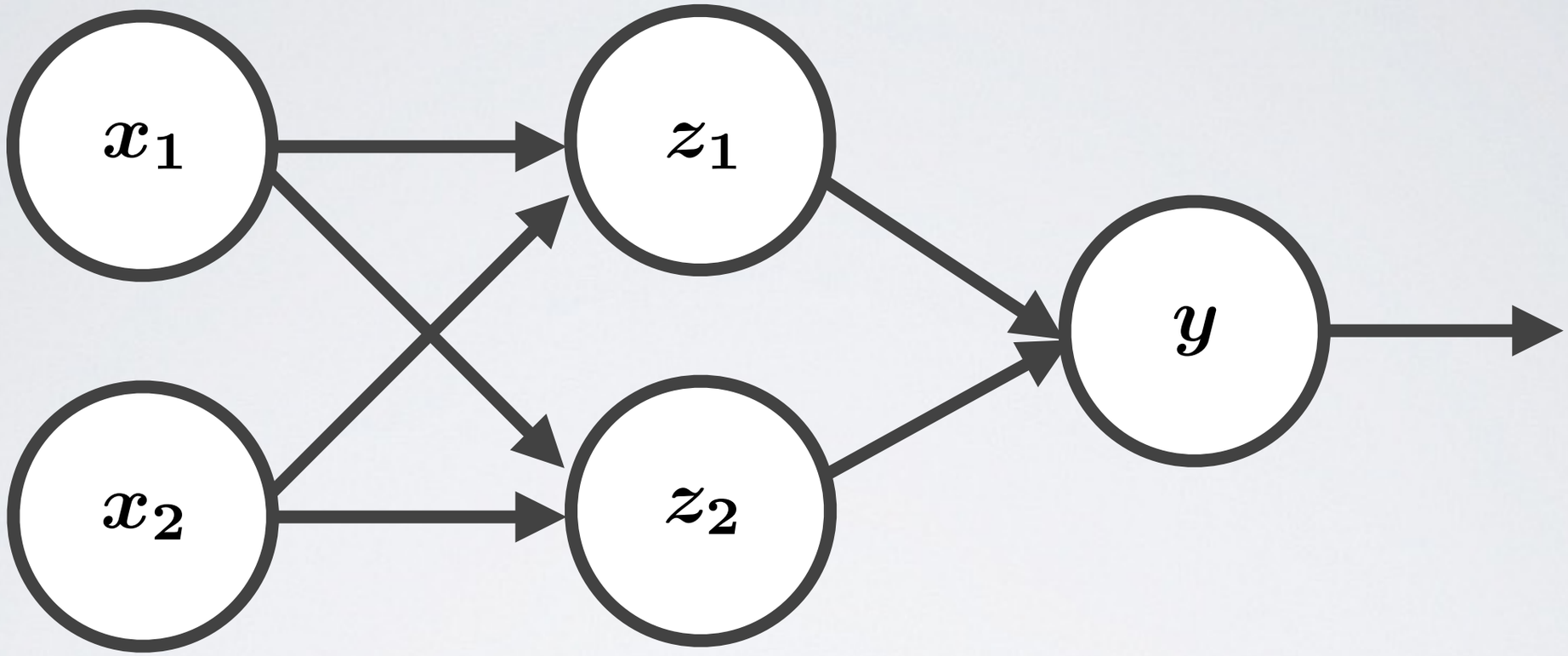
Rosenblatt's perceptron (1957)



$$y = f\left(w_{11}^{(3)} z_1 + w_{12}^{(3)} z_2\right)$$

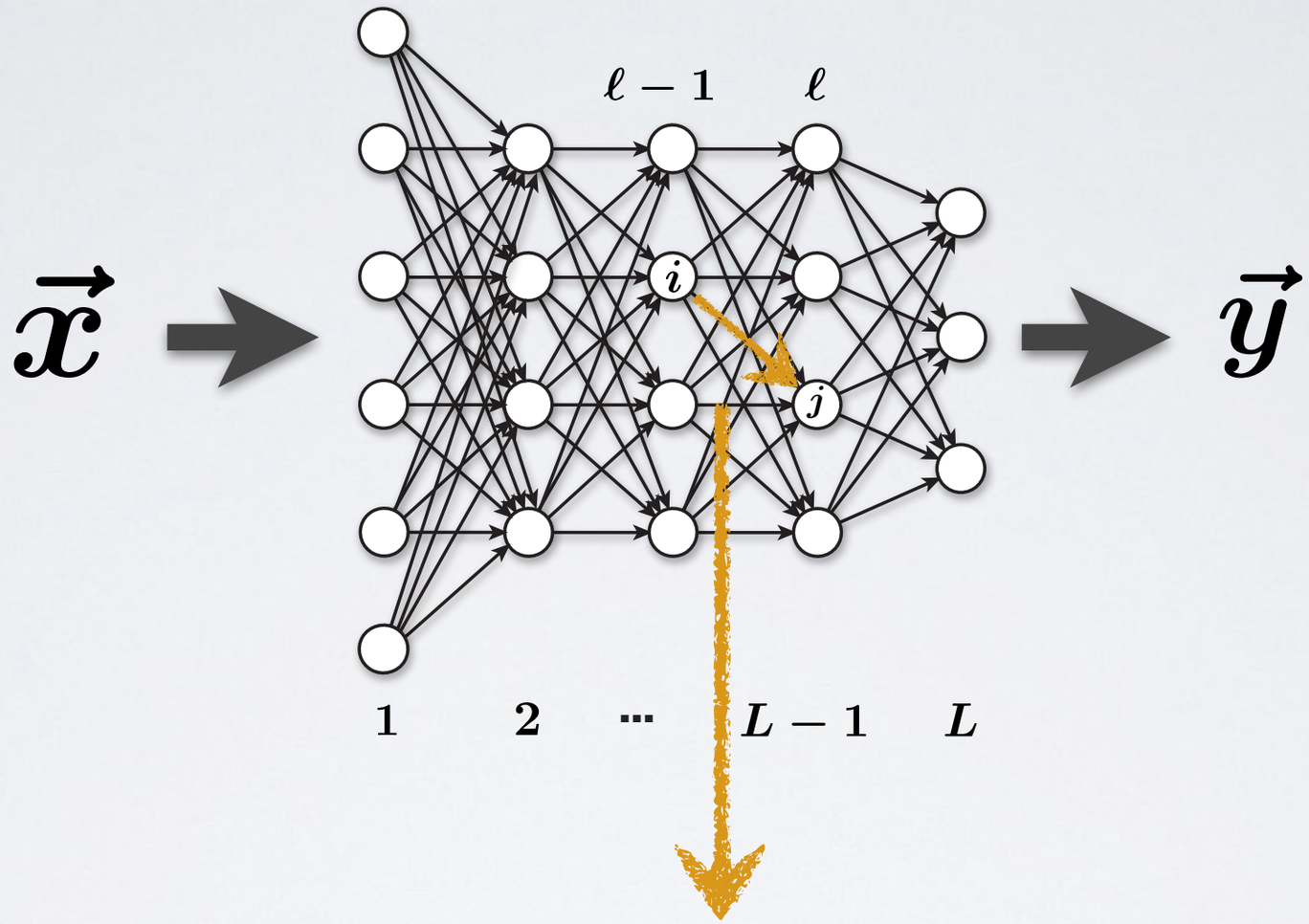
$$\vec{y} = f\left(W^{(3)} \vec{z}\right)$$

Rosenblatt's perceptron (1957)



$$\vec{z} = f\left(\boldsymbol{W}^{(2)}\vec{x}\right) \qquad \vec{y} = f\left(\boldsymbol{W}^{(3)}\vec{z}\right)$$

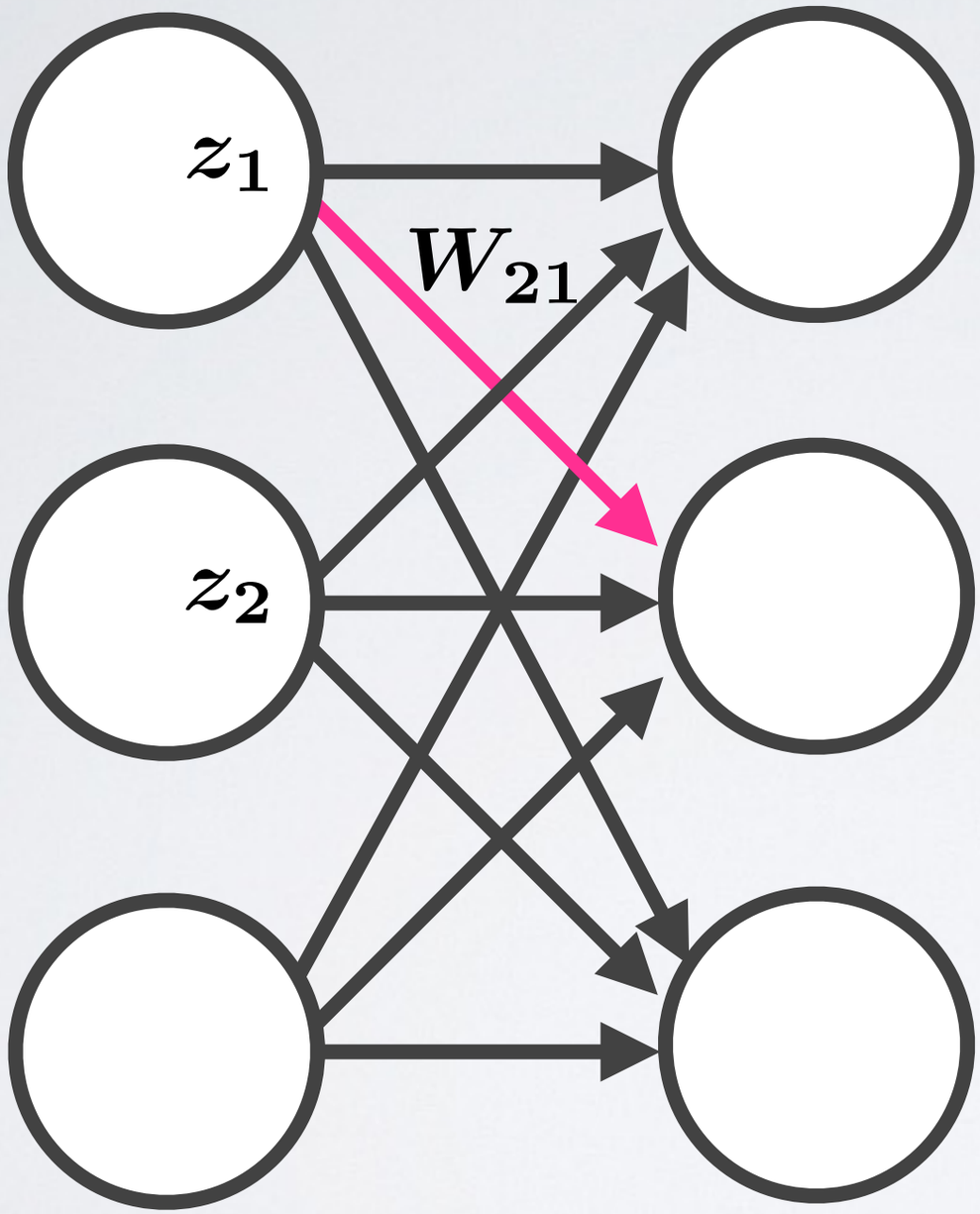
neural network



$$W^{(\ell)} = \left(w_{ji}^{(\ell)}\right)$$

重みパラメータ

neural network

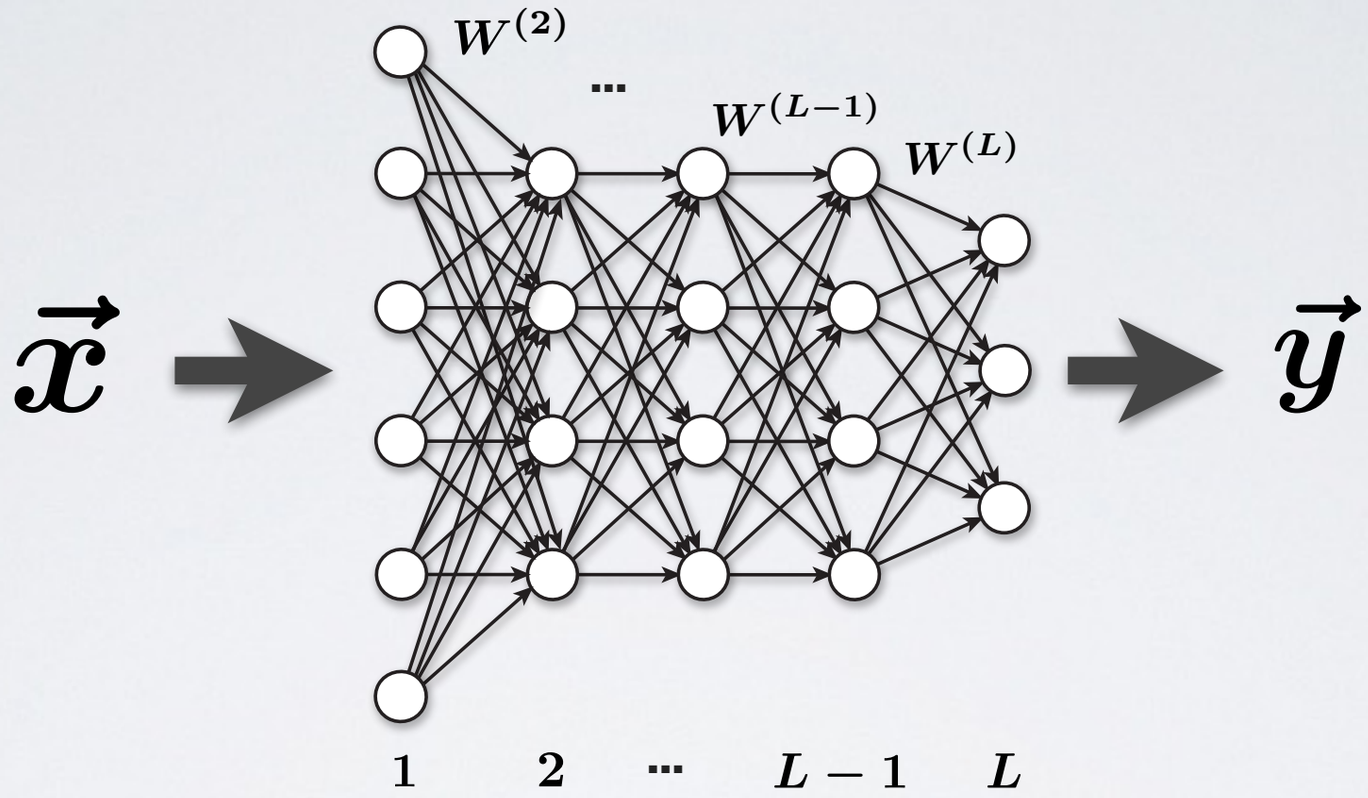


$$\vec{z}$$

$$f(\vec{u}) = \begin{pmatrix} f(u_1) \\ f(u_2) \\ \dots \end{pmatrix}$$

$$\vec{u} = W\vec{z}$$

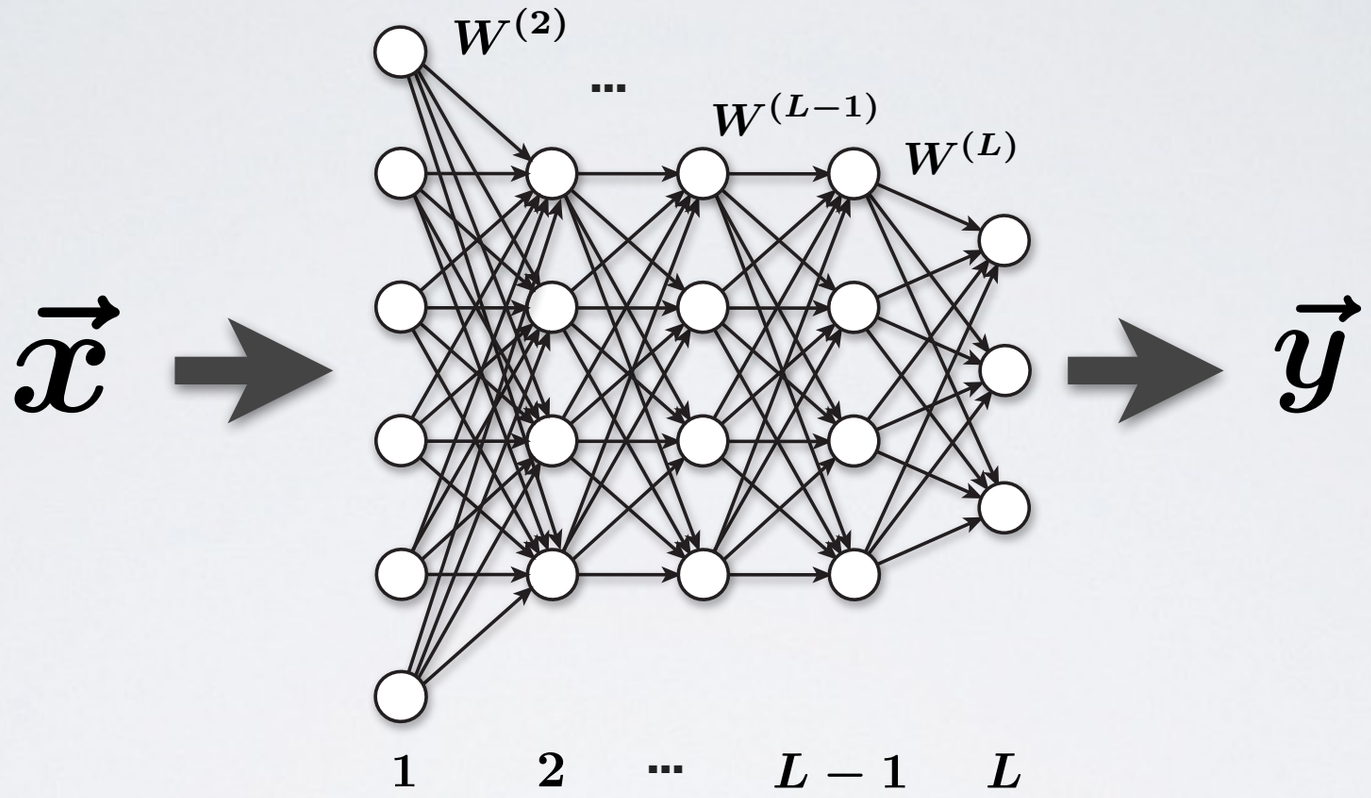
neural network



$$\vec{y} = f\left(W^{(L)} f\left(W^{(L-1)} f\left(W^{(L-2)} \cdots \cdot f\left(W^{(2)} \vec{x}\right) \cdot \cdot\right)\right)\right)$$

f : a nonlinear function (activation)

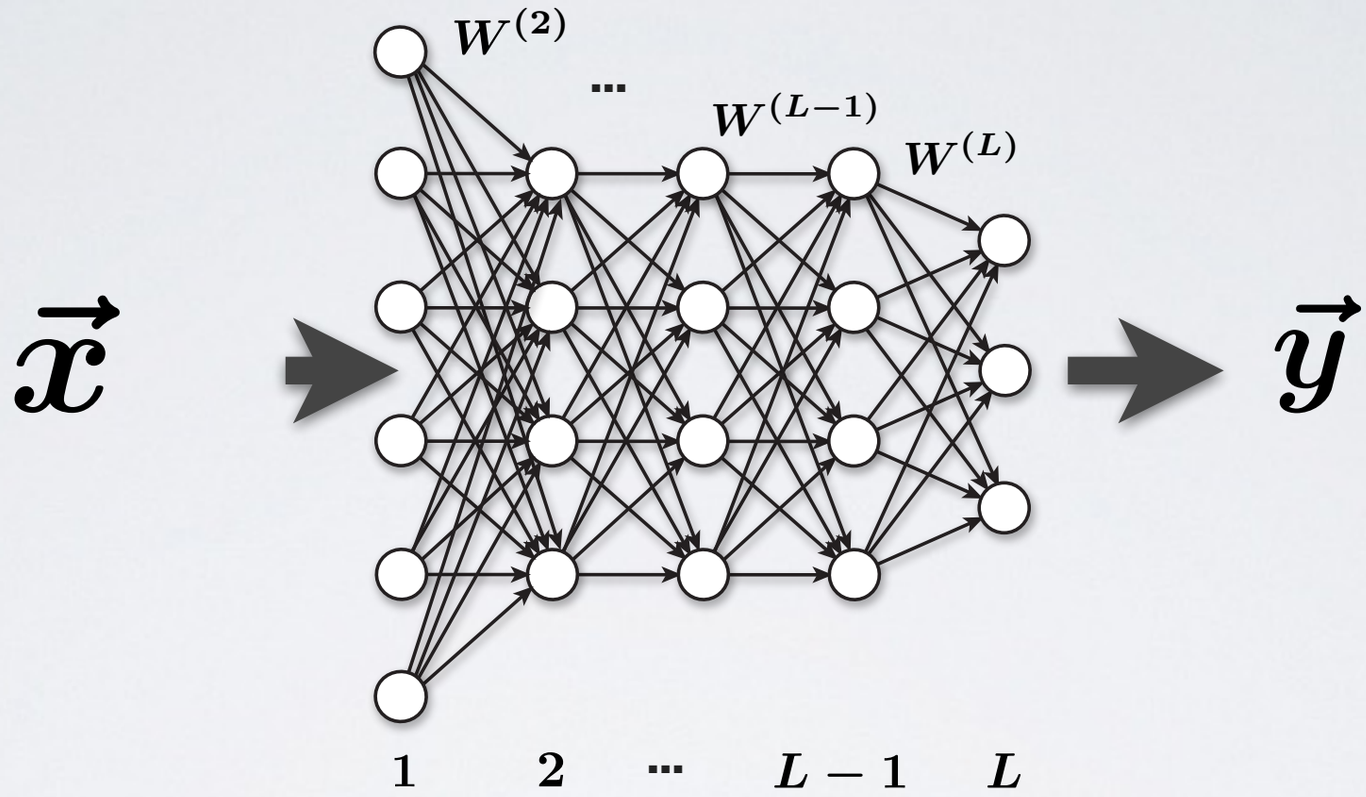
neural network



$$\vec{y} = f\left(W^{(L)} f\left(W^{(L-1)} f\left(W^{(L-2)} \cdots \cdot f\left(W^{(2)} \vec{x}\right) \cdot \cdot\right)\right)\right)$$

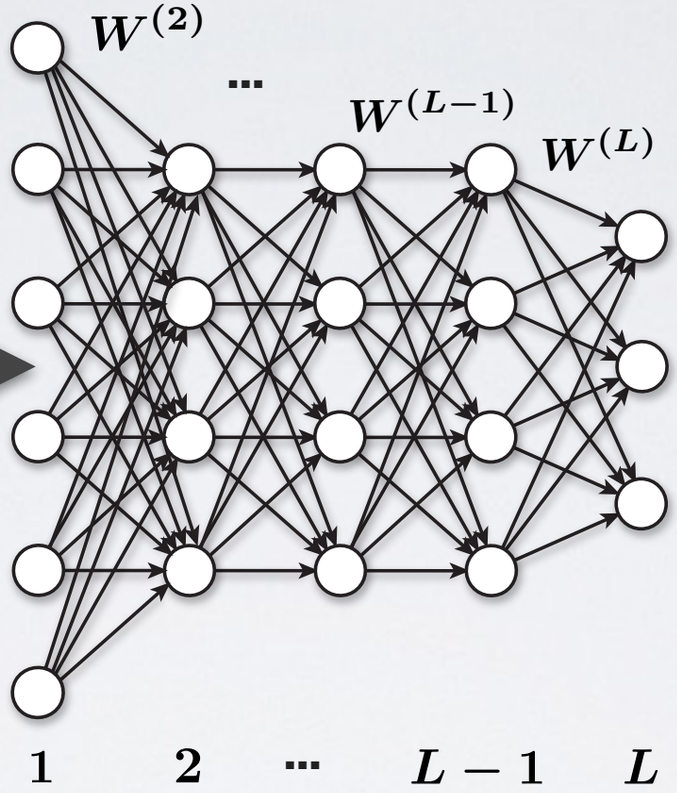
tune (‘train’) them to explain data

training in neural network



training in neural network

$\vec{x}^{(1)}$



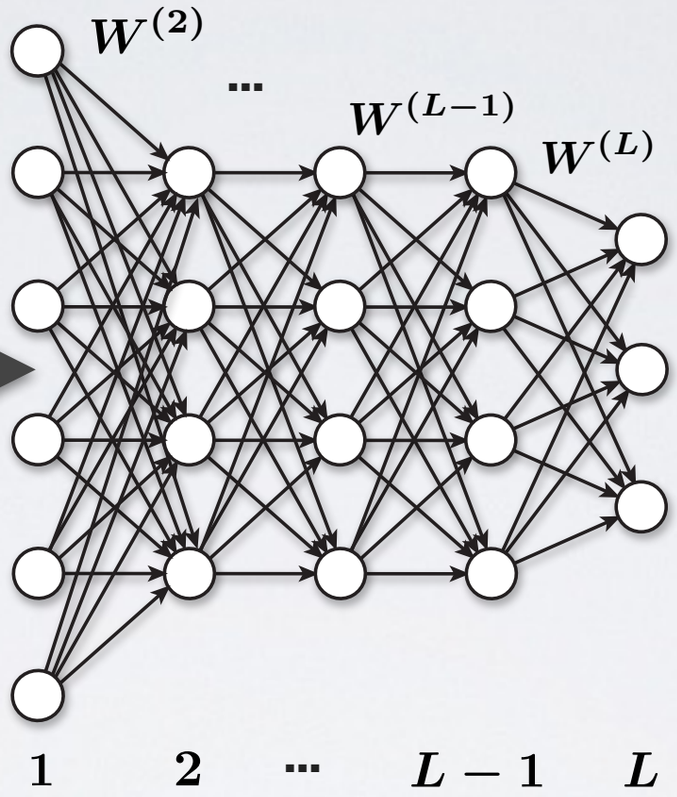
$\vec{y}$

‘cat’ $\vec{y}^{(1)}$

answer

training in neural network

$\vec{x}^{(2)}$



$\vec{y}$

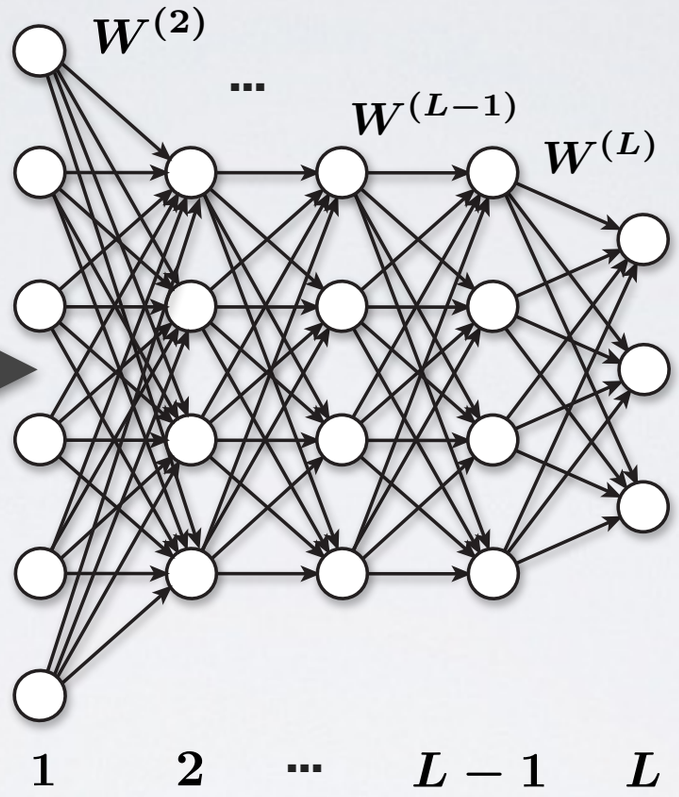
$\vec{y}^{(2)}$

'bird'

answer

training in neural network

$\vec{x}^{(3)}$

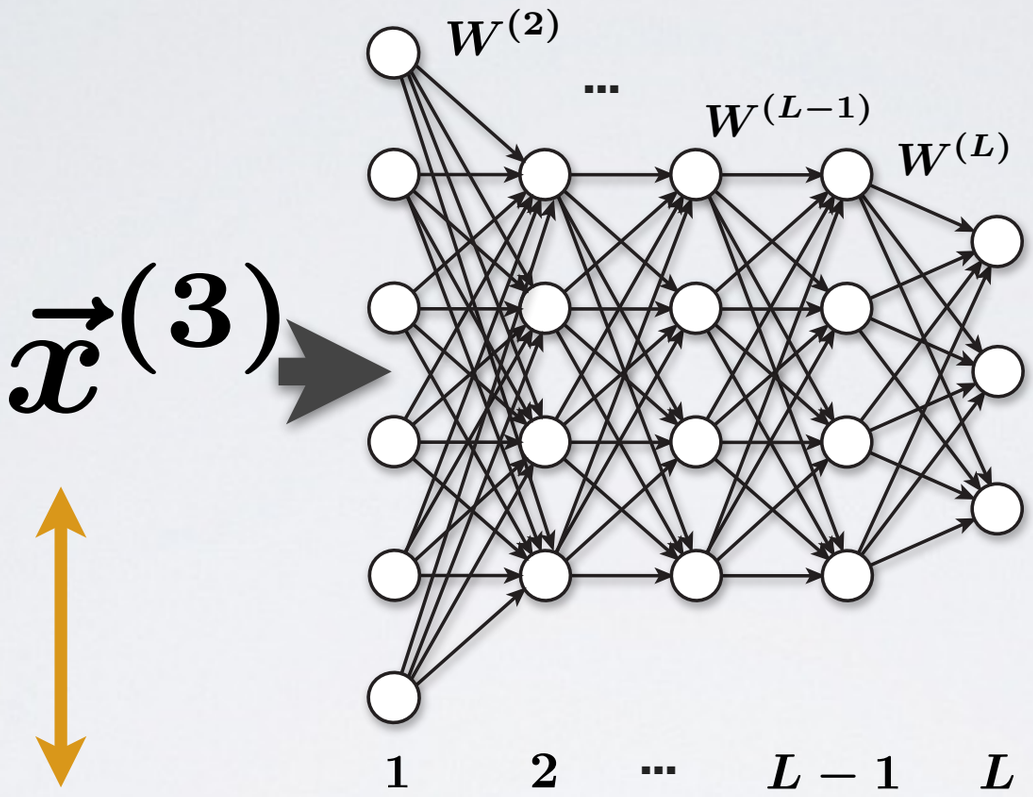


$\vec{y}$

‘cat’ $\vec{y}^{(3)}$

answer

教師あり学習による訓練



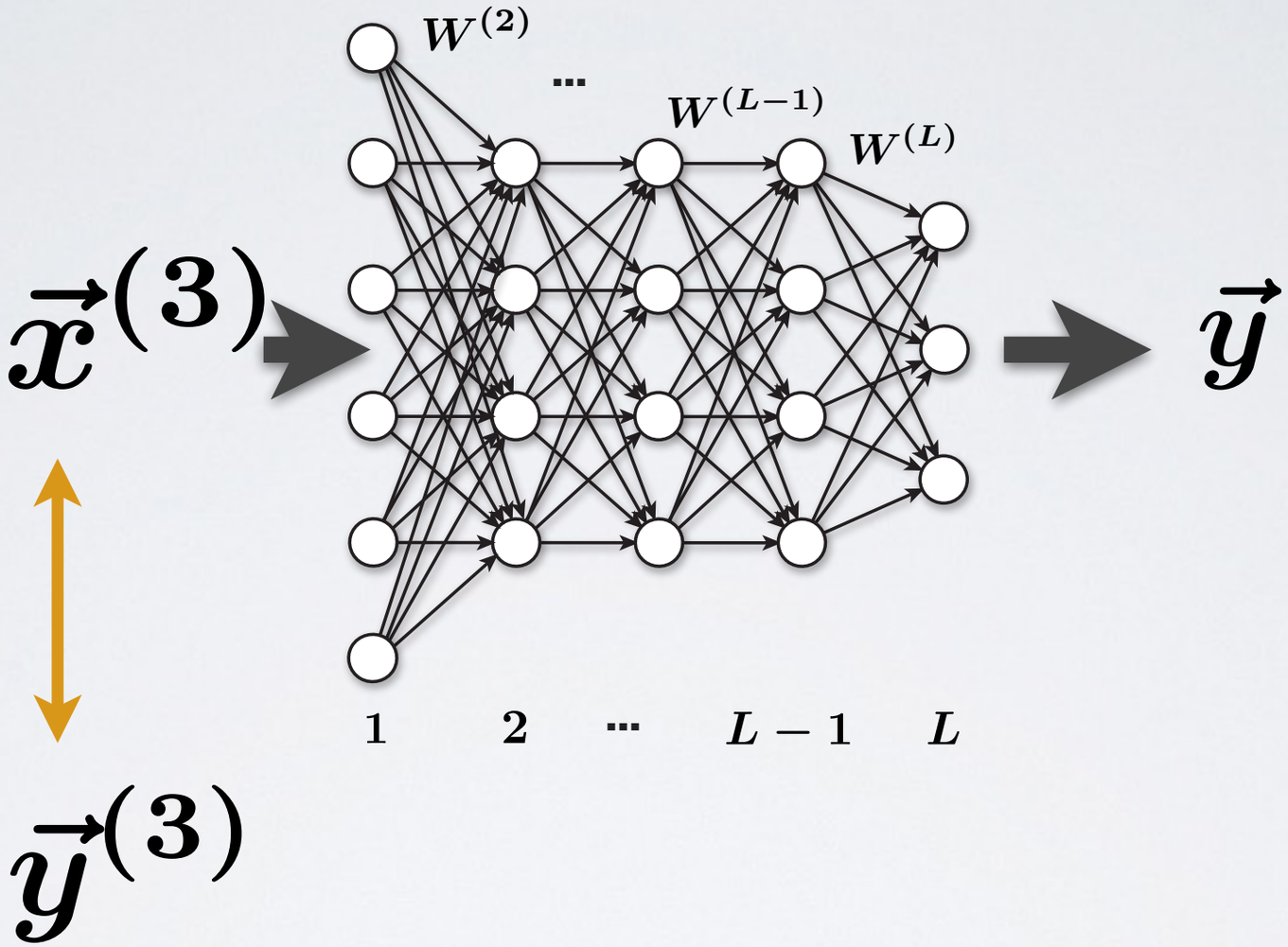
生徒の答案

$\vec{y}^{(3)}$

教師の模範解答

これらのズレ（推定の誤差）
を最小化

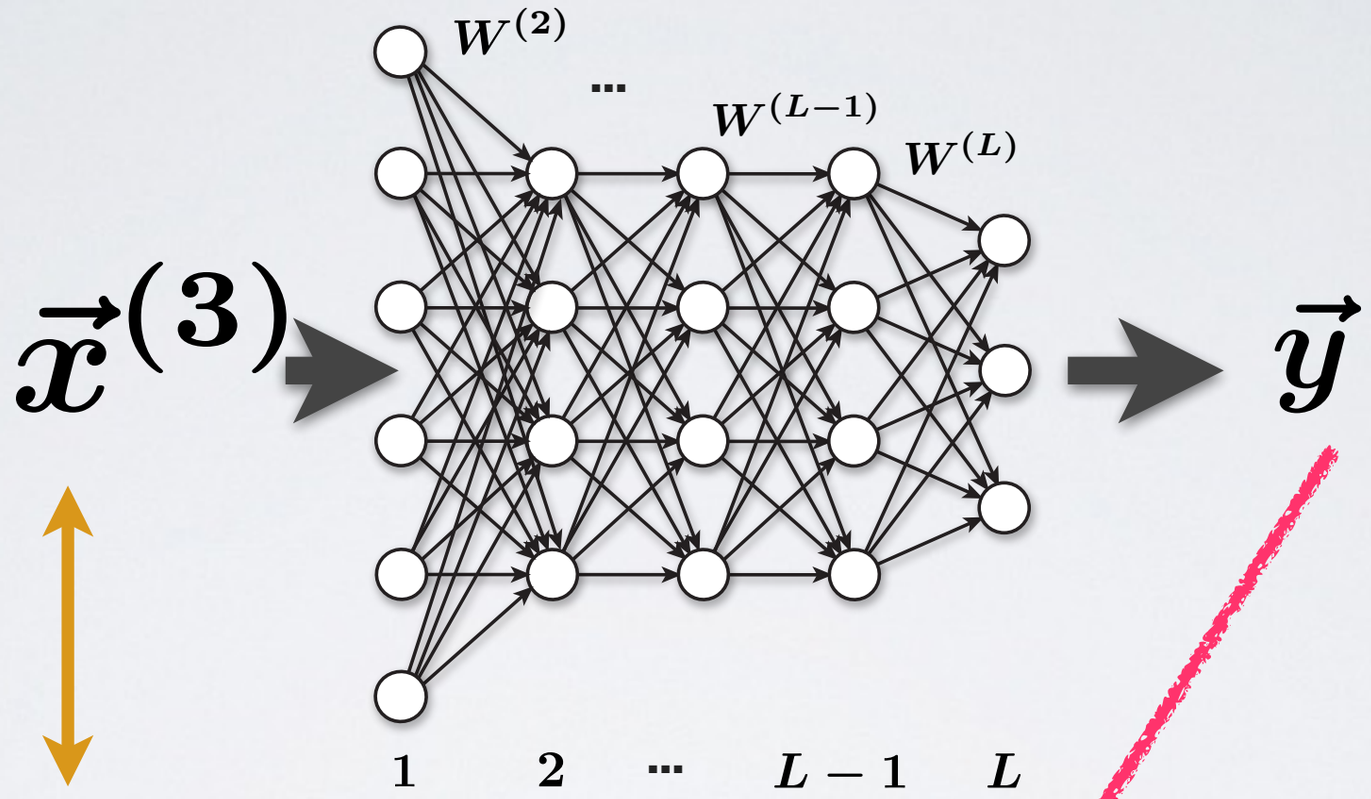
教師あり学習による訓練



誤差関数の最小化

$$E(W) = \frac{1}{2}\sum_{n=1}^{N}\left(\vec{y}(\vec{x}^{(n)};W) - \vec{y}^{(n)}\right)^2$$

教師あり学習による訓練

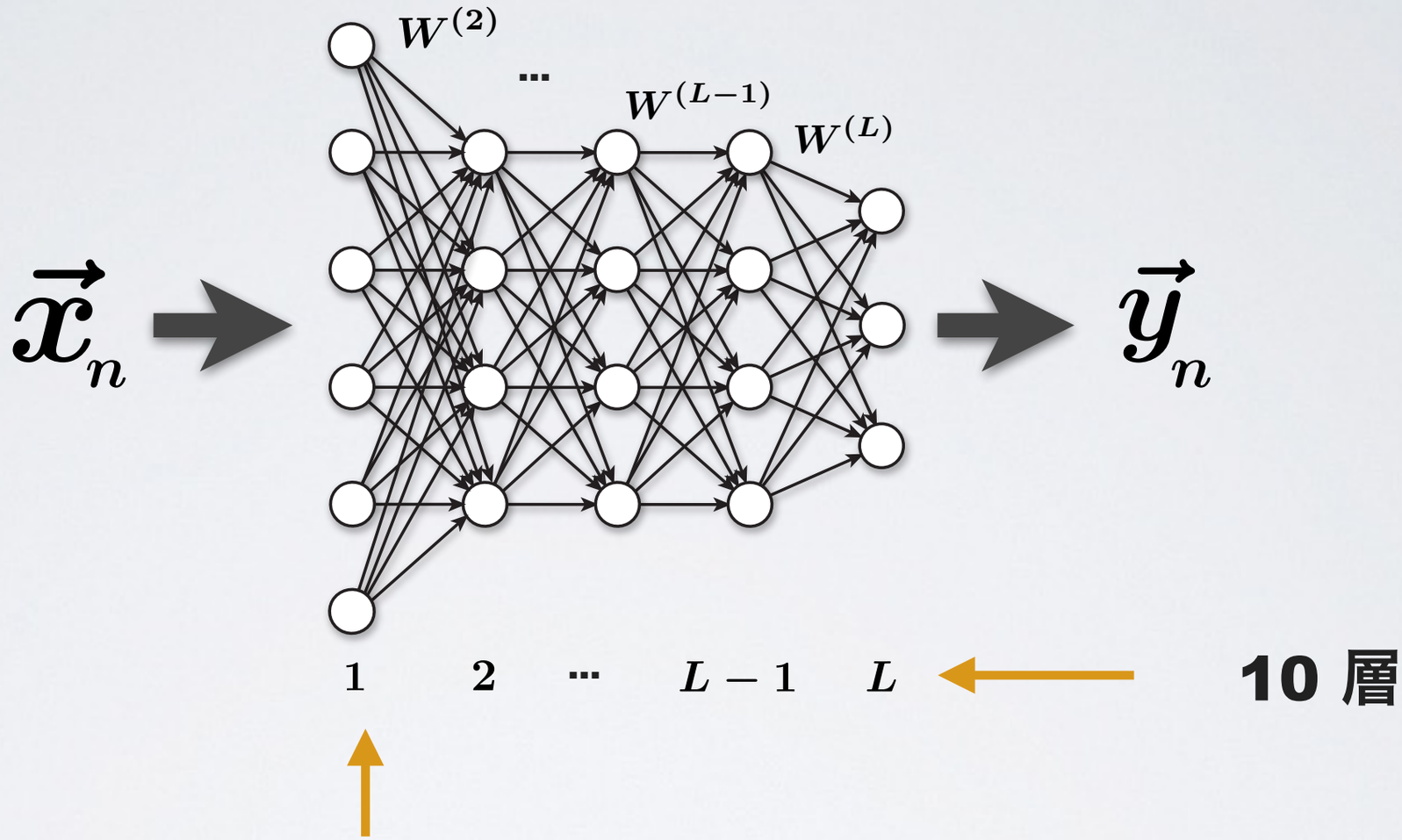


$\vec{y}^{(3)}$

誤差関数の最小化

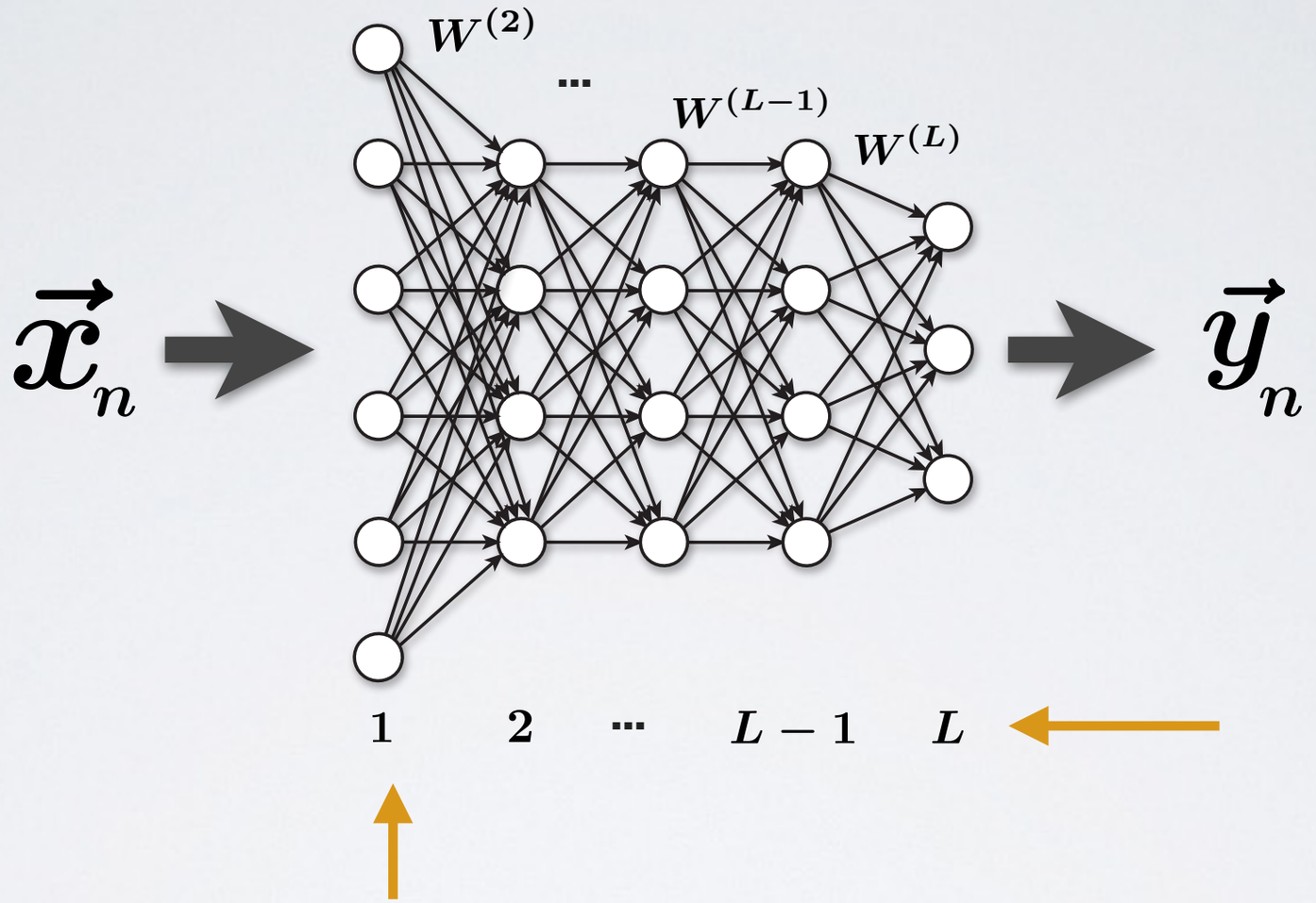
$$E(W) = \frac{1}{2}\sum_{n=1}^{N}\left(\vec{y}(\vec{x}^{(n)};W) - \vec{y}^{(n)}\right)^2$$

教師あり学習による訓練



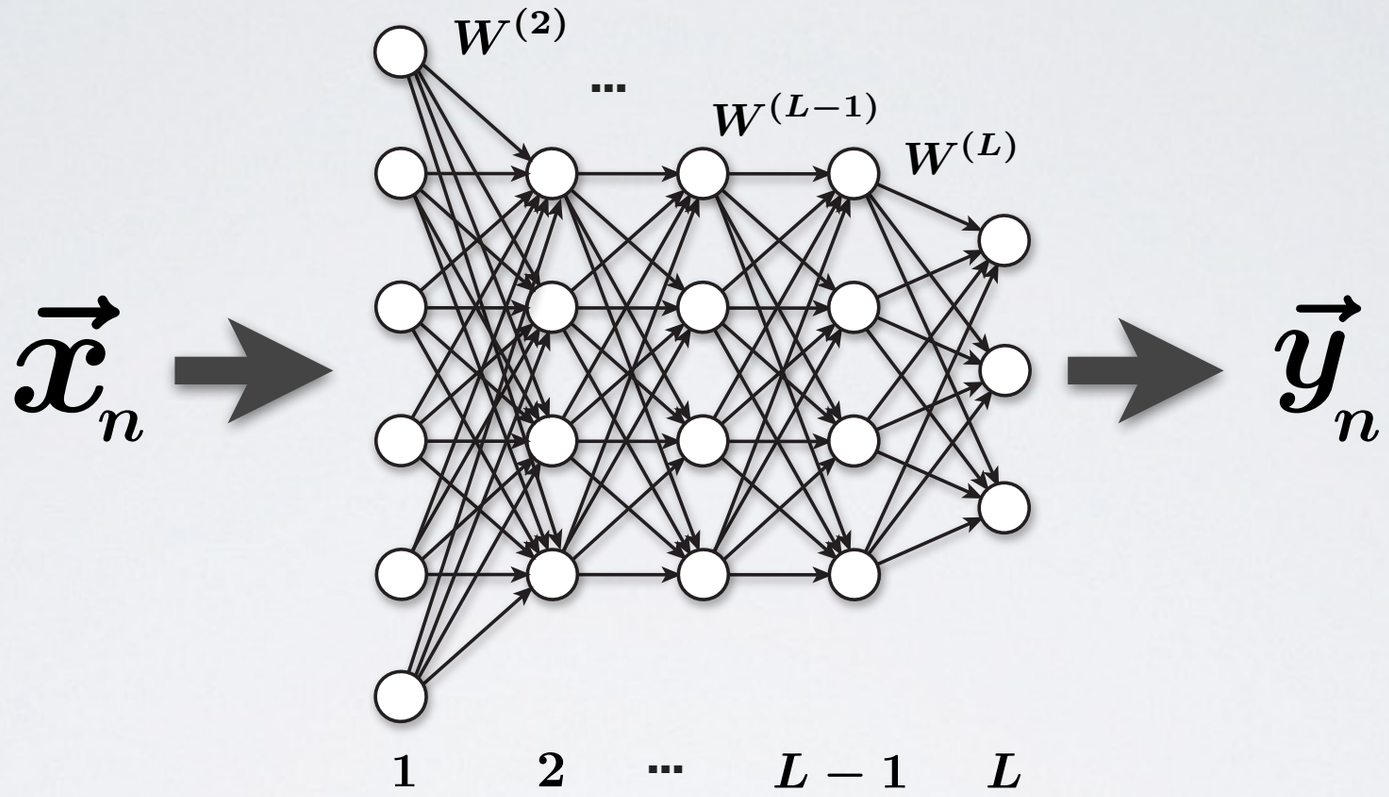
$$E(\boldsymbol{W}) = \frac{1}{2}\sum_{n=1}^{N}\left(\vec{y}(\vec{x}^{(n)};\boldsymbol{W}) - \vec{y}^{(n)}\right)^2$$

教師あり学習による訓練



(100x100)x(100x100)x10=1000000000個のパラメータの調整！！

教師あり学習による訓練

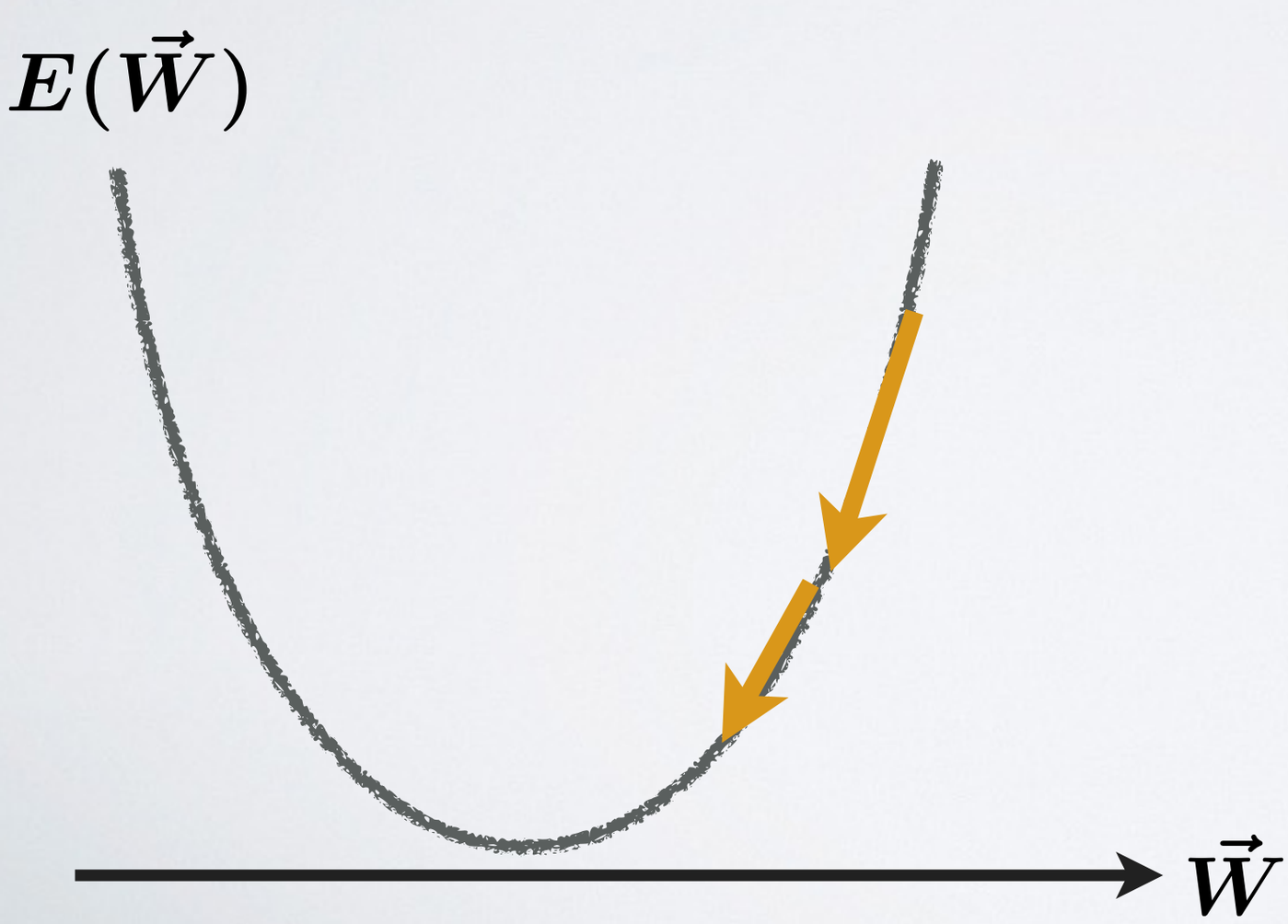


1000000000次元空間上の誤差関数の最小値を探す問題！！

$$E(\boldsymbol{W}) = \frac{1}{2}\sum_{n=1}^{N}\left(\vec{y}(\vec{x}^{(n)};\boldsymbol{W}) - \vec{y}^{(n)}\right)^2$$

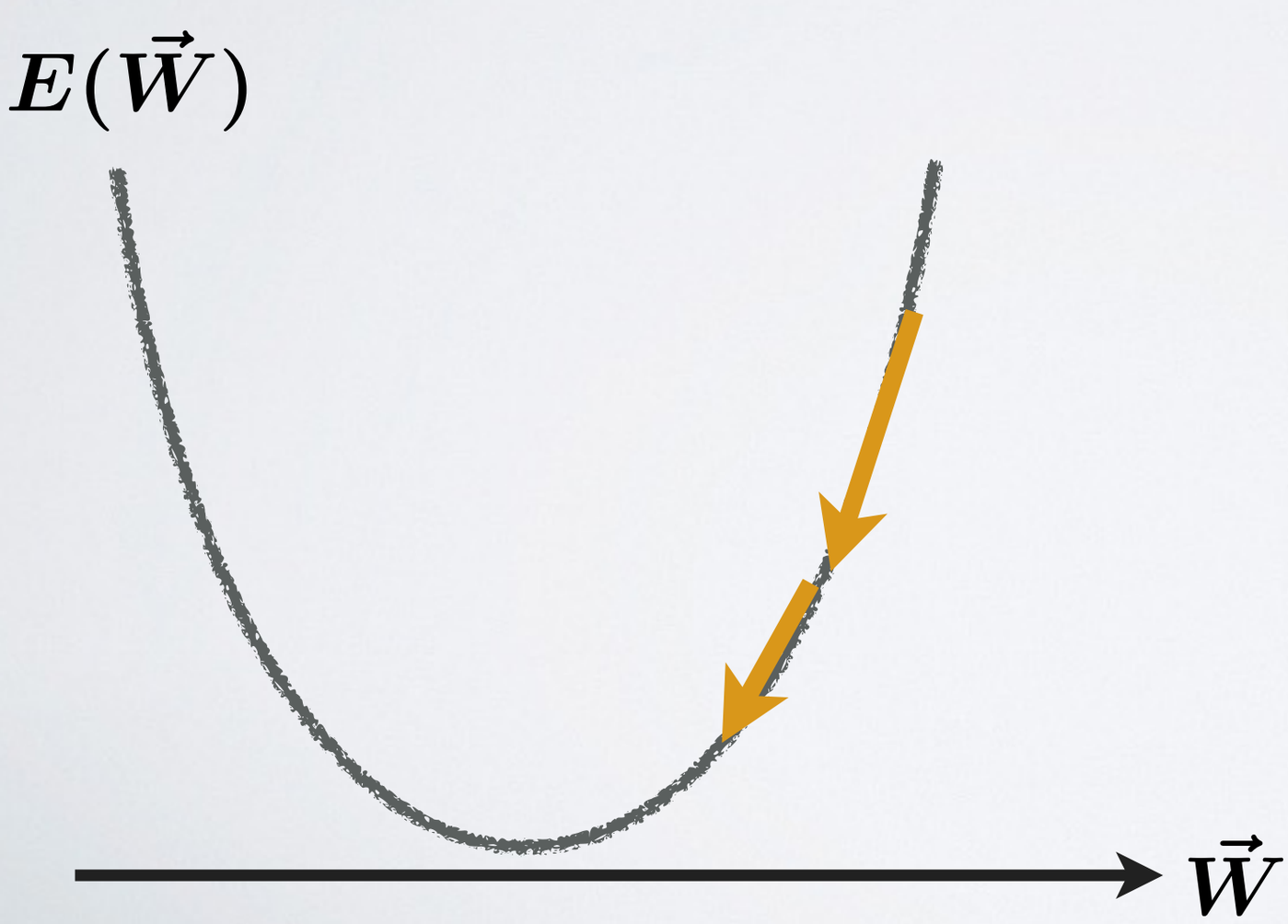
勾配降下法

$$W_{t+1} \leftarrow W_t - \eta \frac{\partial E(W_t)}{\partial W}$$



勾配降下法

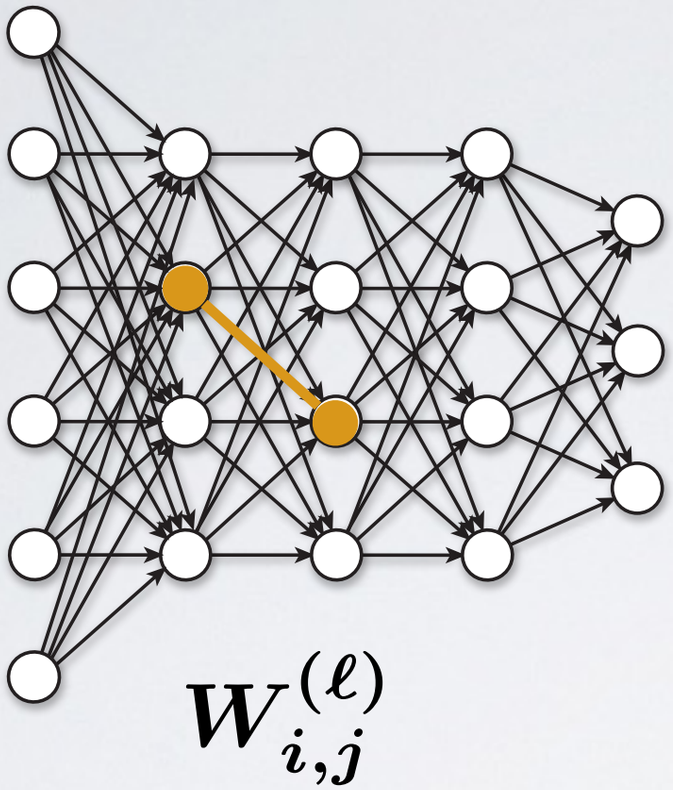
$$W_{t+1} \leftarrow W_t - \eta \frac{\partial E(W_t)}{\partial W}$$



誤差逆伝播法(backprop)*

Backpropagation（誤差逆伝播法）

$\vec{x}_n$

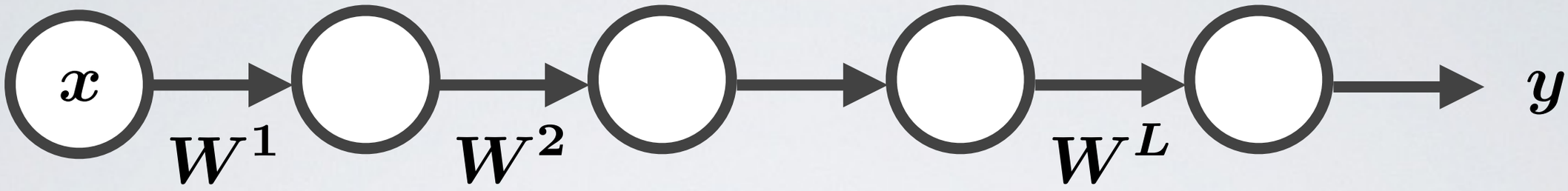


$\vec{y}_n$

$$E(W) = \frac{1}{2}\sum_n \left(\vec{y}_n(W) - \vec{a}_n\right)^2$$

$$\frac{\partial E(\vec{W})}{\partial W_{ij}^{(\ell)}}$$

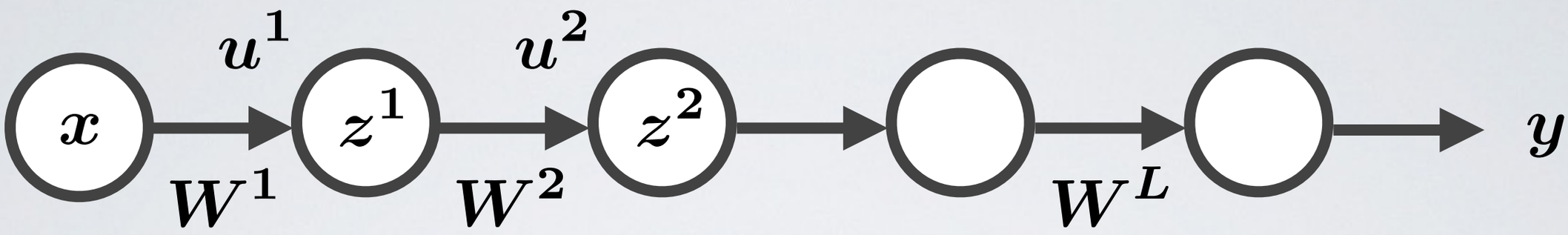
Backpropagation（誤差逆伝播法）



$$y = f\left(W^L f\left(W^{L-1} f\left(\cdots W^2 f\left(W^1 x\right)\right)\right)\right)$$

$$\frac{\partial E(\vec{W})}{\partial W_{ij}^{(\ell)}}$$

Backpropagation（誤差逆伝播法）



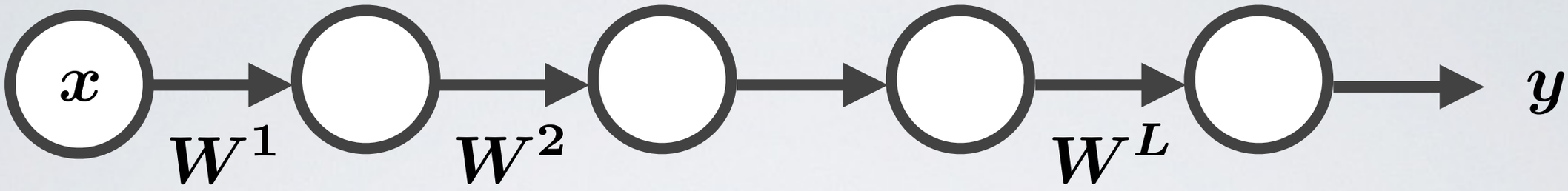
$$y = f\left(W^L f\left(W^{L-1} f\left(\cdots W^2 f\left(W^1 x\right)\right)\right)\right)$$

$$u^\ell = W^\ell z^{\ell-1}$$

$$z^\ell = f\left(u^\ell\right)$$

$$x = z^0, \quad y = z^L$$

Backpropagation（誤差逆伝播法）

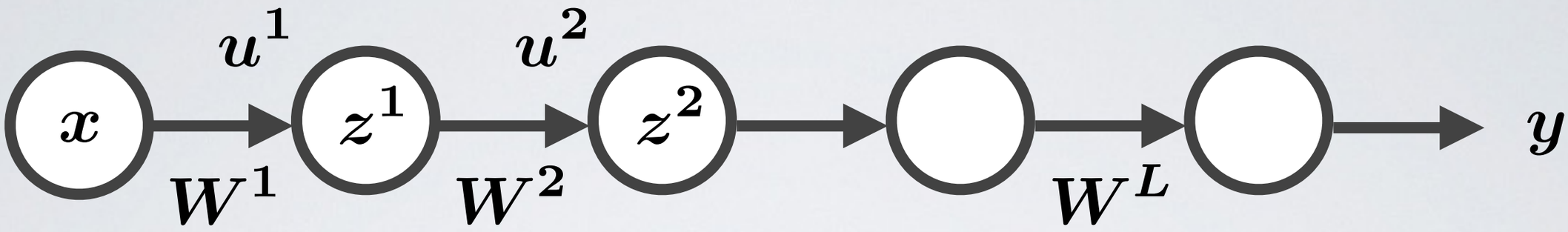


$$y = f(W^L f(W^{L-1} f(\cdots W^2 f(W^1 x))))$$

$$E = E(y(W's))$$

$$\frac{\partial E(W)}{\partial W^2}$$

Backpropagation (誤差逆伝播法)



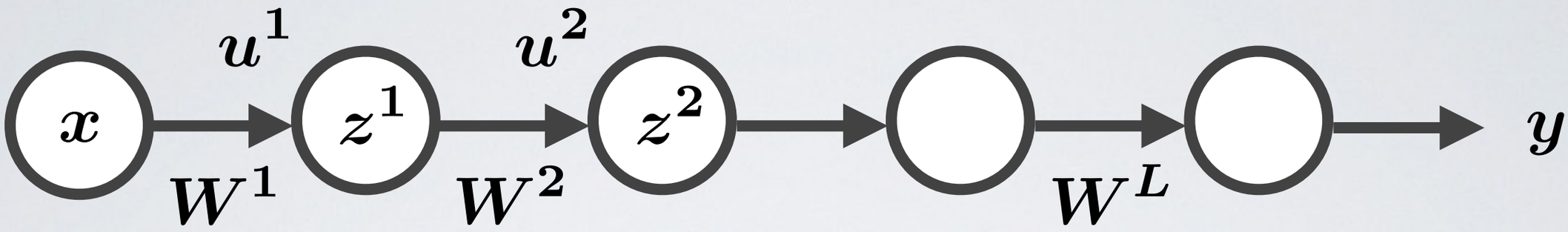
$$u^\ell = W^\ell z^{\ell-1}$$

$$z^\ell = f\left(u^\ell\right)$$

$$x = z^0, \quad y = z^L$$

$$\frac{\partial E}{\partial W^\ell} = \frac{\partial E}{\partial u^\ell}\frac{\partial u^\ell}{\partial W^\ell} = \frac{\partial E}{\partial u^\ell} z^{\ell-1}$$

Backpropagation (誤差逆伝播法)



$$u^\ell = W^\ell z^{\ell-1}$$

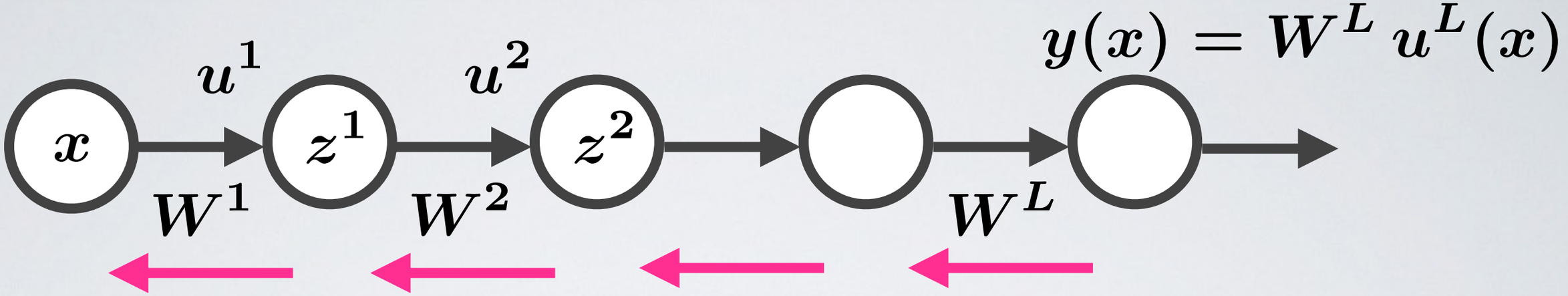
$$z^\ell = f(u^\ell)$$

$$x = z^0, \quad y = z^L$$

$$\frac{\partial E}{\partial W^\ell} = \frac{\partial E}{\partial u^\ell}\frac{\partial u^\ell}{\partial W^\ell} = \frac{\partial E}{\partial u^\ell} z^{\ell-1}$$

$$\frac{\partial E}{\partial u^\ell} = \frac{\partial E}{\partial u^{\ell+1}}\frac{\partial u^{\ell+1}}{\partial u^\ell} = \frac{\partial E}{\partial u^{\ell+1}} W^{\ell+1} f'(u^\ell)$$

Backpropagation（誤差逆伝播法）

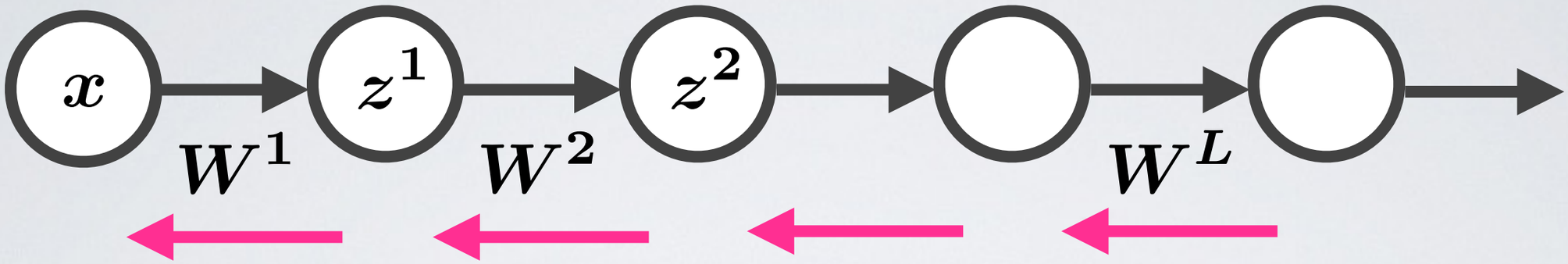


$$\frac{\partial E}{\partial u^L} = \sum_n \left(y(x^{(n)}) - y^{(n)} \right)$$

$$\frac{\partial E}{\partial W^\ell} = \frac{\partial E}{\partial u^\ell} \frac{\partial u^\ell}{\partial W^\ell} = \frac{\partial E}{\partial u^\ell} z^{\ell-1}$$

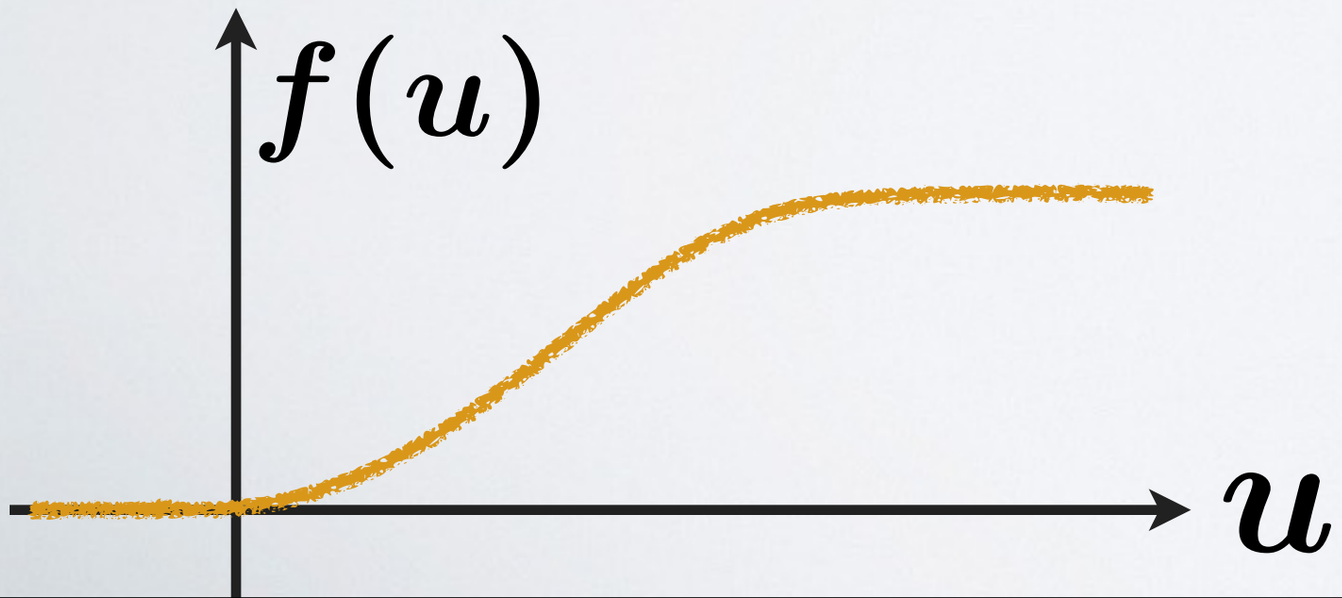
$$\frac{\partial E}{\partial u^\ell} = \frac{\partial E}{\partial u^{\ell+1}} \frac{\partial u^{\ell+1}}{\partial u^\ell} = \frac{\partial E}{\partial u^{\ell+1}} W^{\ell+1} f'(u^\ell)$$

Backpropagation（誤差逆伝播法）

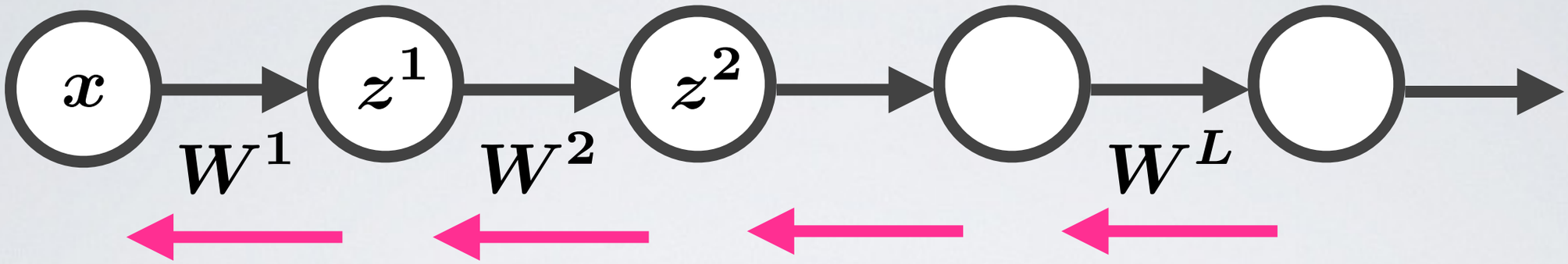


$$\frac{\partial E}{\partial u^{\ell}} = \frac{\partial E}{\partial u^{\ell+1}} W^{\ell+1} f'(u^{\ell})$$

逆伝播による誤差情報の消失(勾配消失問題)

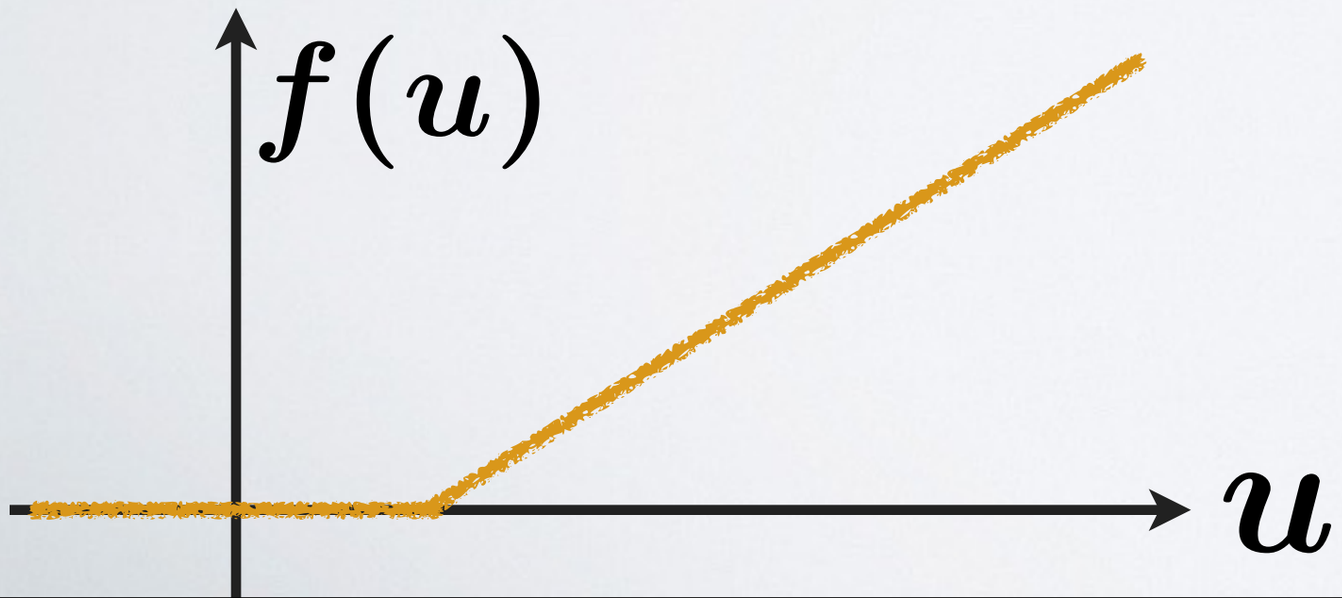


Backpropagation（誤差逆伝播法）



$$\frac{\partial E}{\partial u^{\ell}} = \frac{\partial E}{\partial u^{\ell+1}} W^{\ell+1} f'(u^{\ell})$$

逆伝播による誤差情報の消失(勾配消失問題)

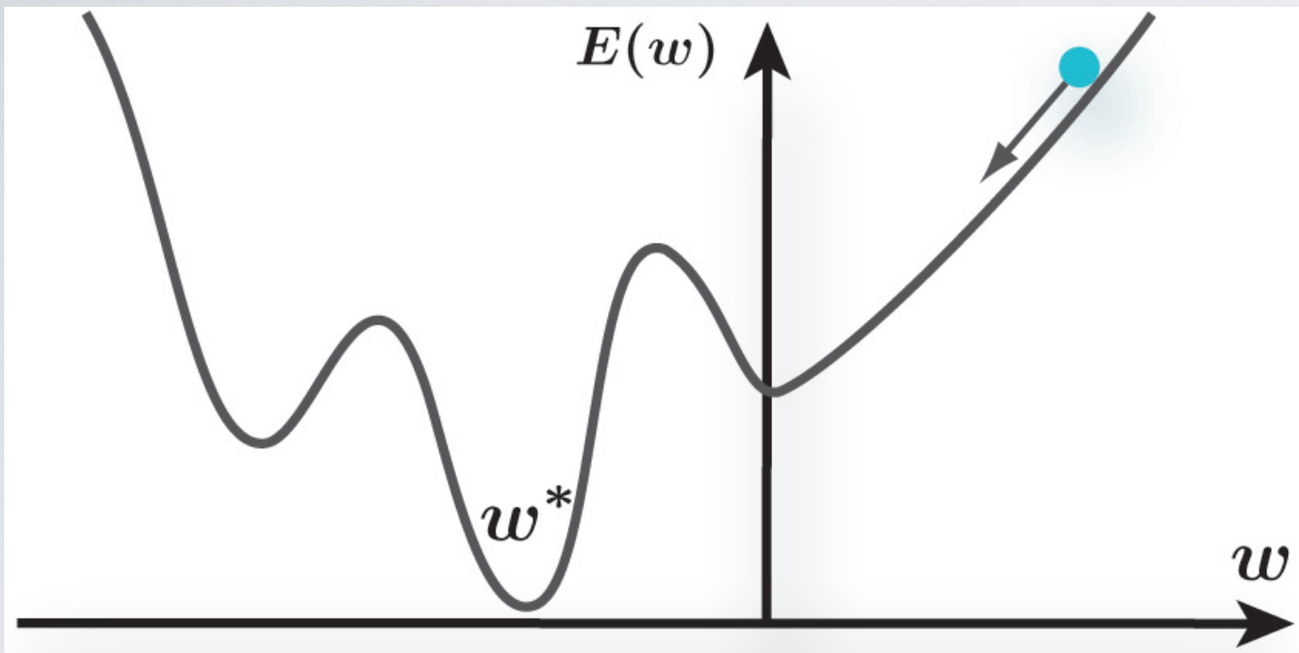


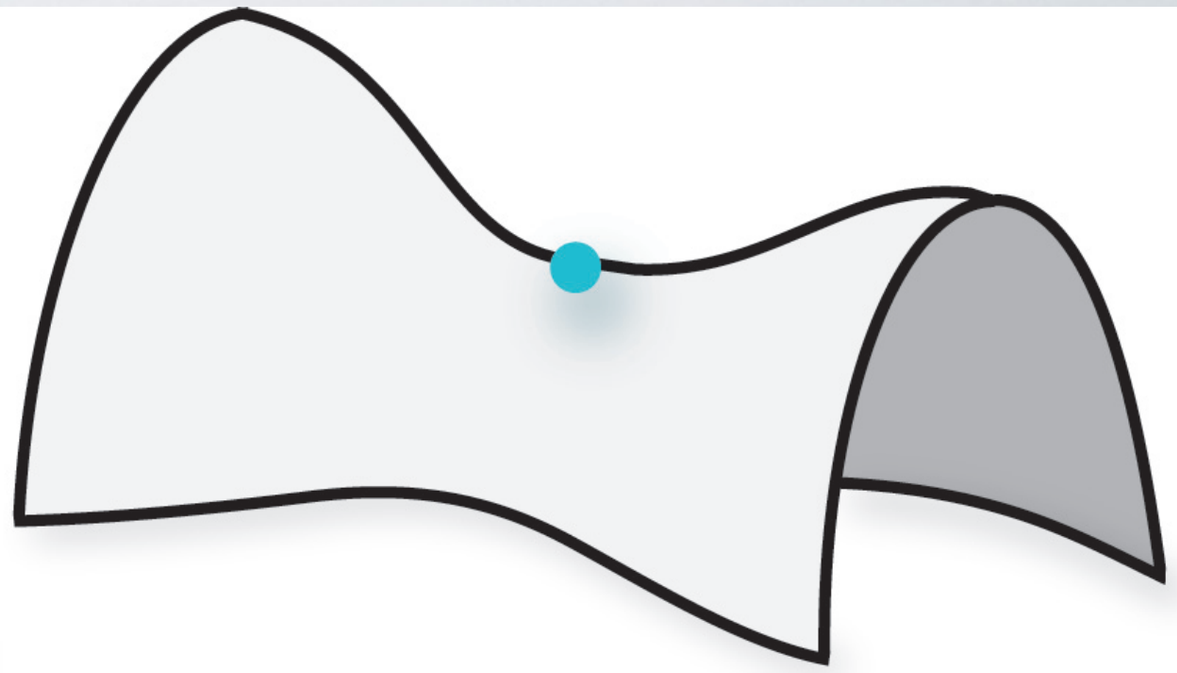
事前学習

ReLU活性化関数

mini-batch学習

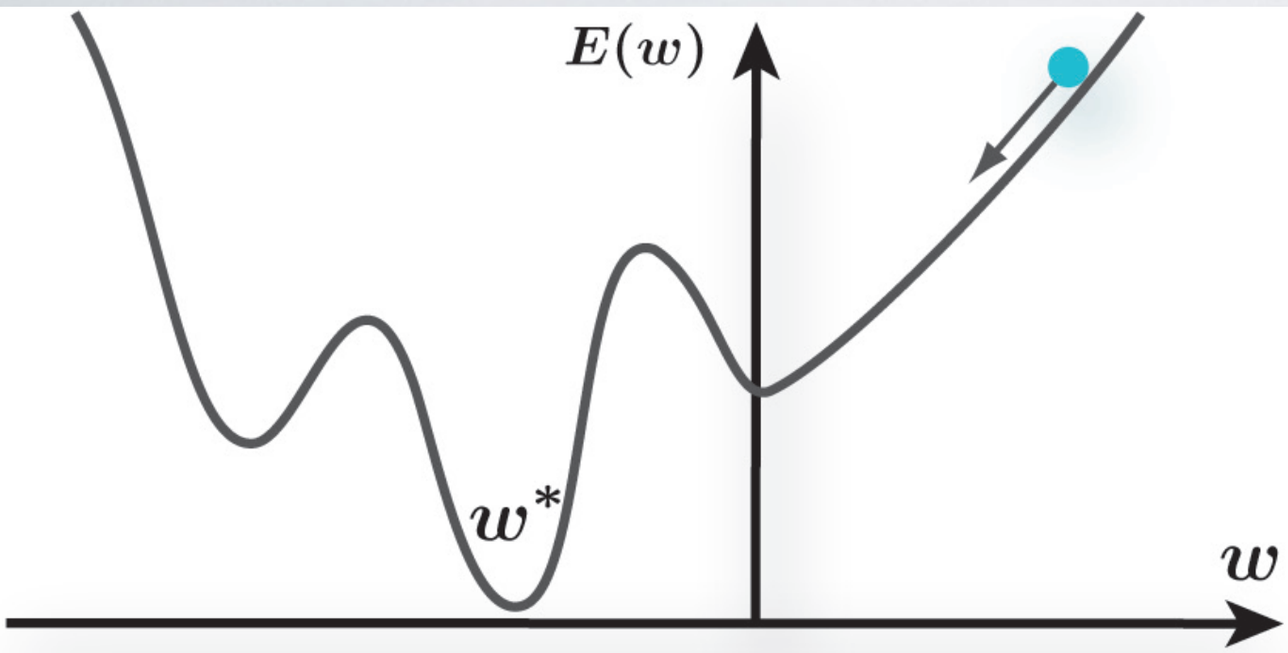
mini-batch

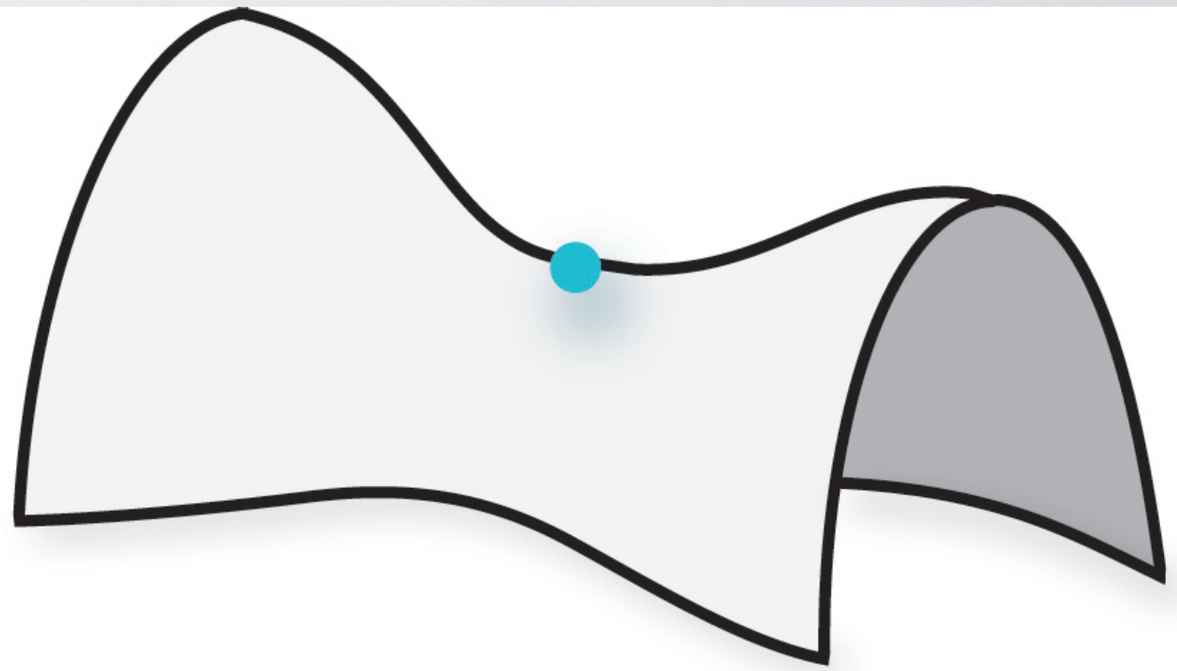




局所最適解や鞍点にはまりかねない。というか必ずはまる。

mini-batch





局所最適解や鞍点にはまりかねない。というか必ずはまる。

少しでも悪い場所にはまり込むのを防ぐために、ランダムな要素を入れて、臨界点から弾き出す。

mini-batch

$$E(W) = \frac{1}{N} \sum_{n=1}^{N} e(x_n; W)$$

mini-batch

$$E(W) = \frac{1}{N}\sum_{n=1}^{N} e(x_n; W)$$

$\mathcal{D} = \{x_1, x_2, \cdots, x_N\}$ → $\mathcal{B}_t \subset \mathcal{D}$

ミニバッチをランダムに取り出しておく

mini-batch

$$E(W) = \frac{1}{N}\sum_{n=1}^{N} e(x_n; W)$$

$\mathcal{D} = \{x_1, x_2, \cdots, x_N\}$ → $\mathcal{B}_t \subset \mathcal{D}$

ミニバッチをランダムに取り出しておく

$$E_t(W) = \frac{1}{|\mathcal{B}_t|}\sum_{m \in \mathcal{B}_t} e(x_m; W)$$

mini-batch

$$E(W) = \frac{1}{N}\sum_{n=1}^{N} e(x_n; W)$$

$\mathcal{D} = \{x_1, x_2, \cdots, x_N\}$ → $\mathcal{B}_t \subset \mathcal{D}$

ミニバッチをランダムに取り出しておく

$$E_t(W) = \frac{1}{|\mathcal{B}_t|}\sum_{m \in \mathcal{B}_t} e(x_m; W)$$

$$W_{t+1} \leftarrow W_t - \eta \frac{\partial E_t(W_t)}{\partial W}$$

mini-batch

$$E(W) = \frac{1}{N}\sum_{n=1}^{N} e(x_n; W)$$

$\mathcal{D} = \{x_1, x_2, \cdots, x_N\}$ → $\mathcal{B}_t \subset \mathcal{D}$

ミニバッチをランダムに取り出しておく

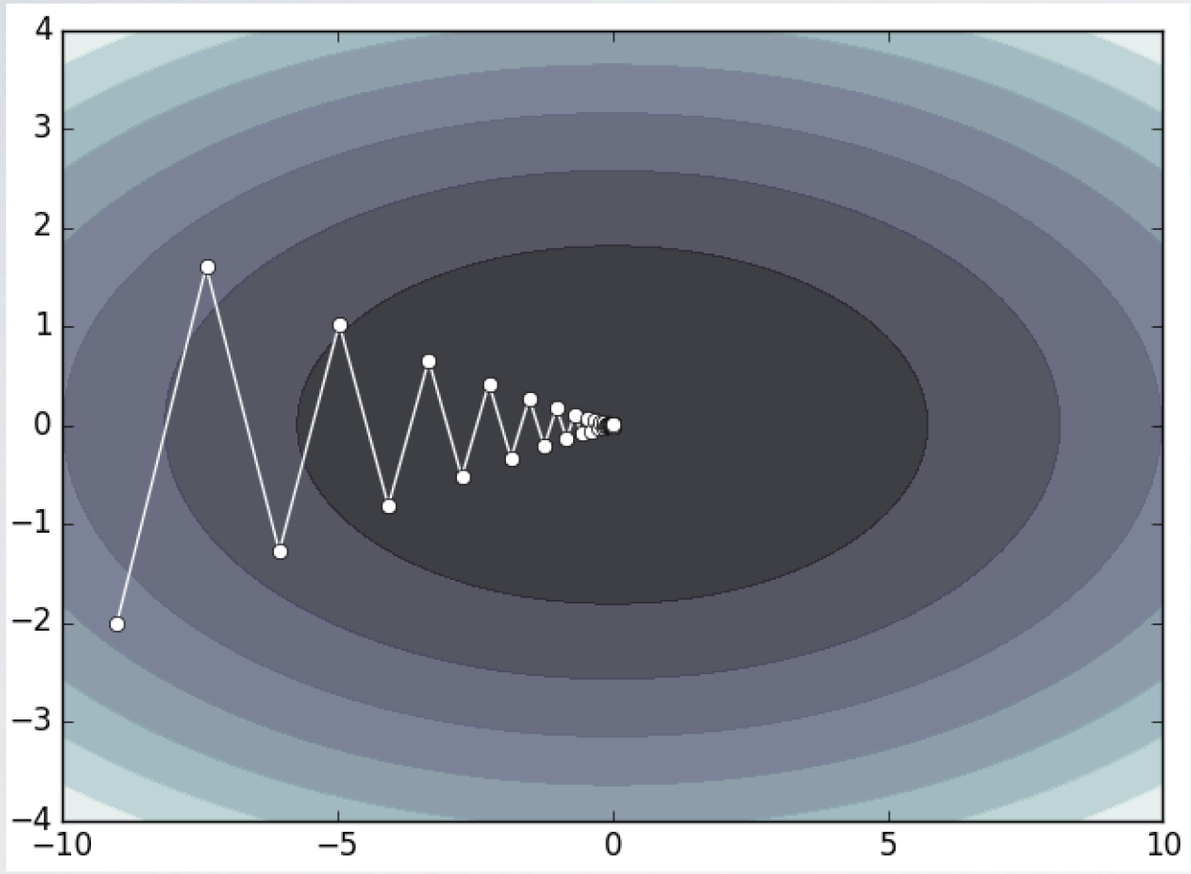
$$E_t(W) = \frac{1}{|\mathcal{B}_t|}\sum_{m \in \mathcal{B}_t} e(x_m; W)$$

$$W_{t+1} \leftarrow W_t - \eta \frac{\partial E_t(W_t)}{\partial W}$$

データ並列化の観点からも有効。

Optimizers

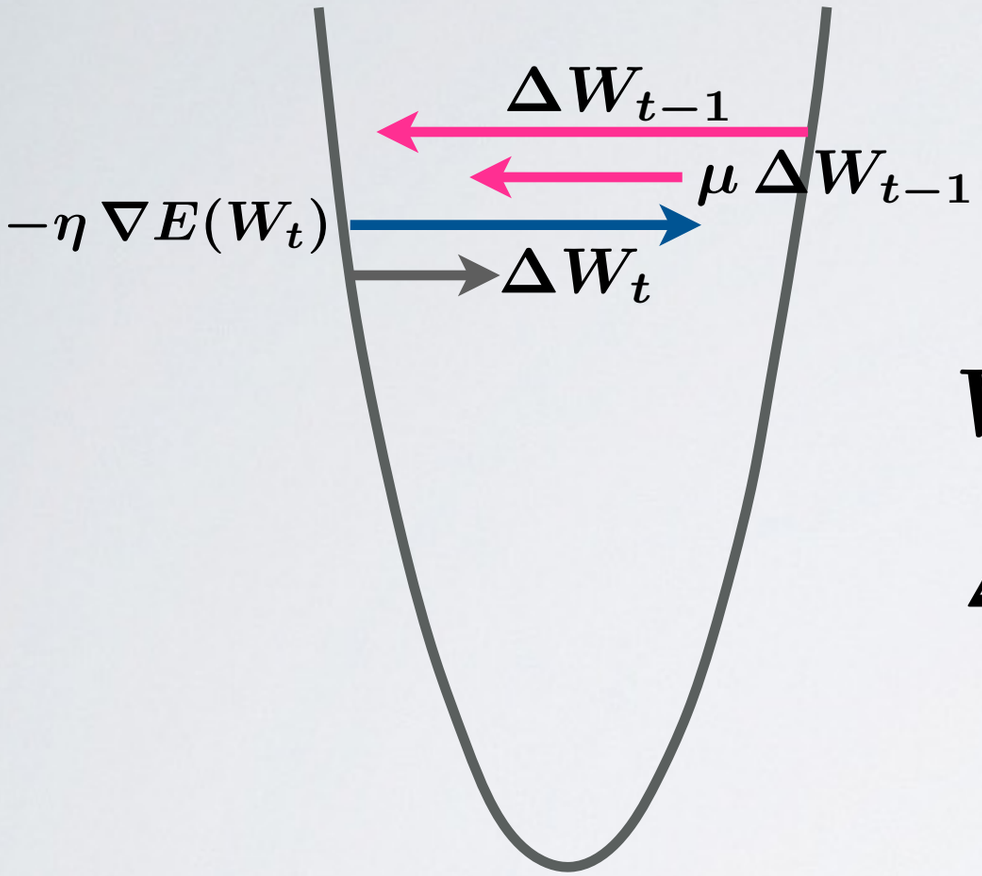
momentum



谷底での振動が収束を遅くする。

勾配の緩やかな方向にも探索を進めたい。

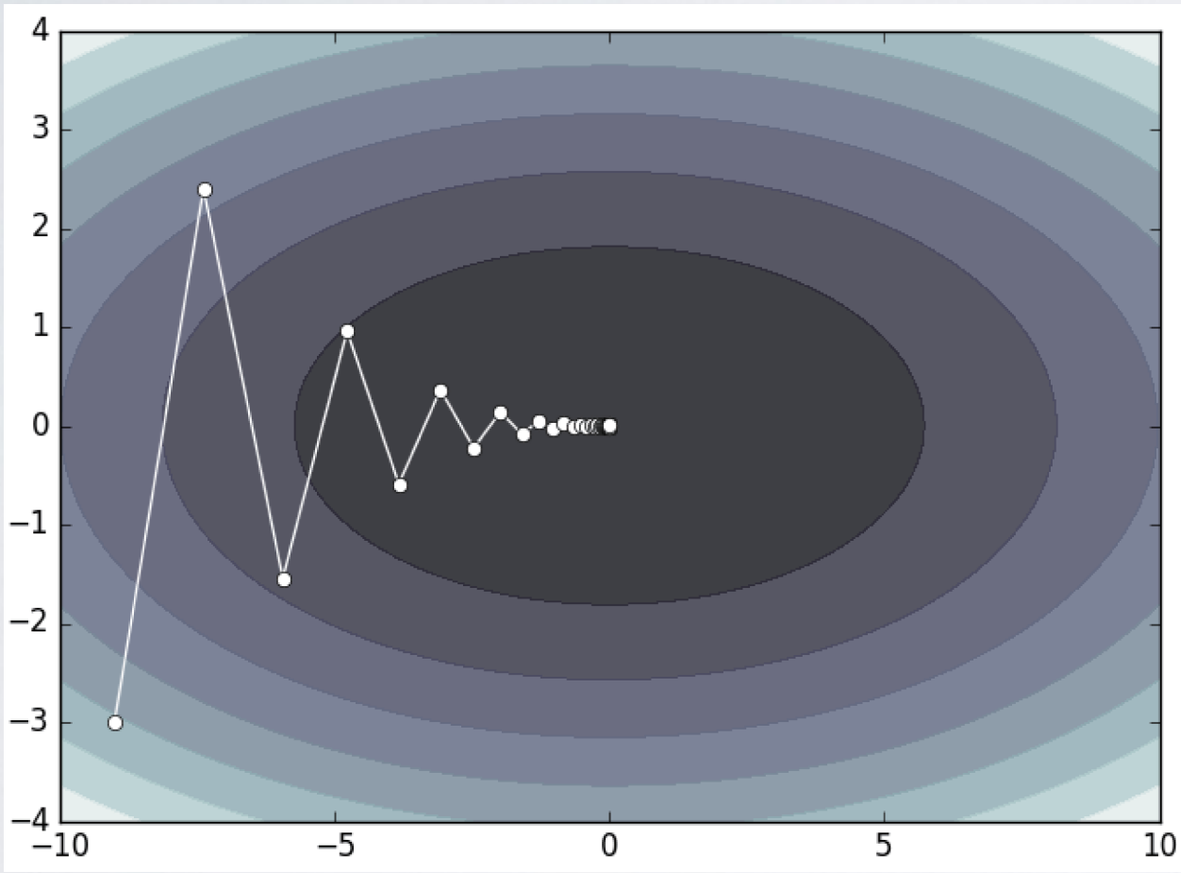
momentum

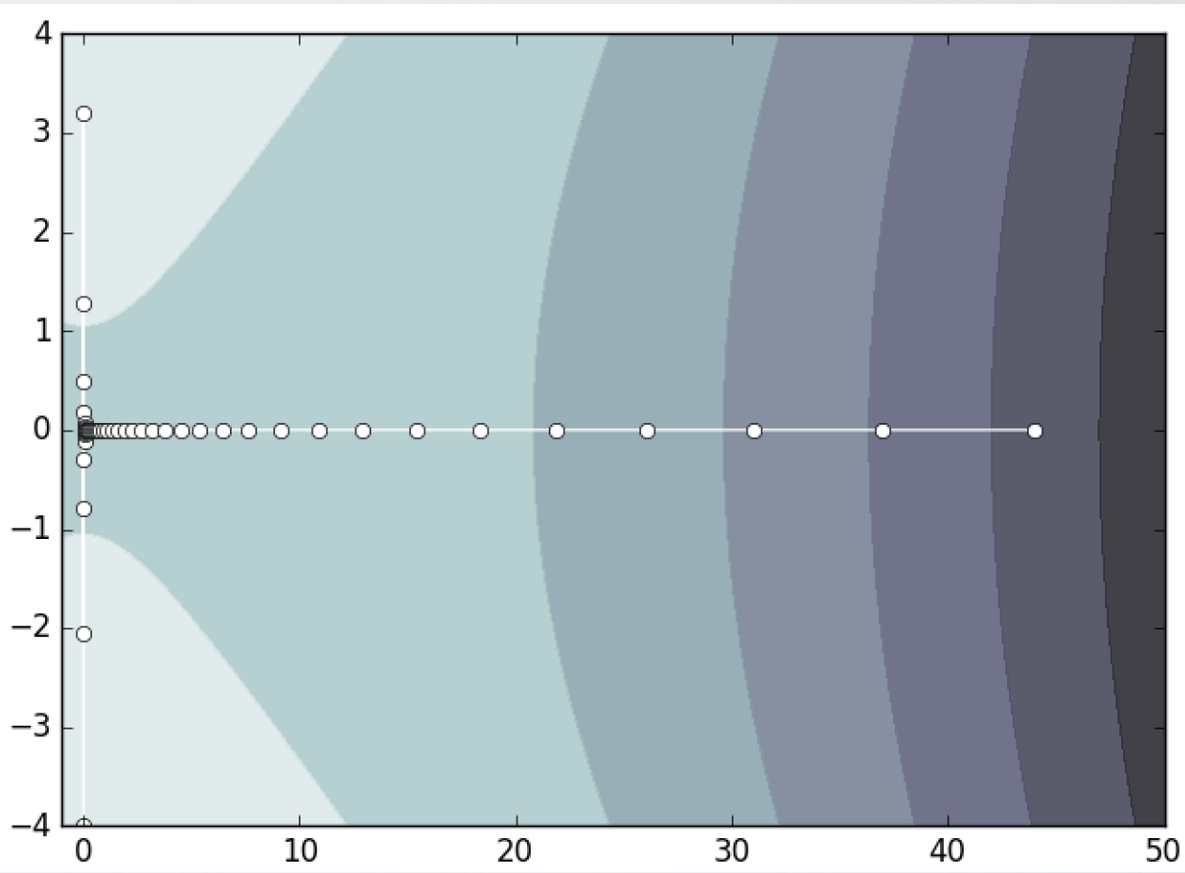


$$W_{t+1} = W_t + \Delta W_t$$

$$\Delta W_t = -\eta\,\nabla E(W_t) + \mu\,\Delta W_{t-1}$$

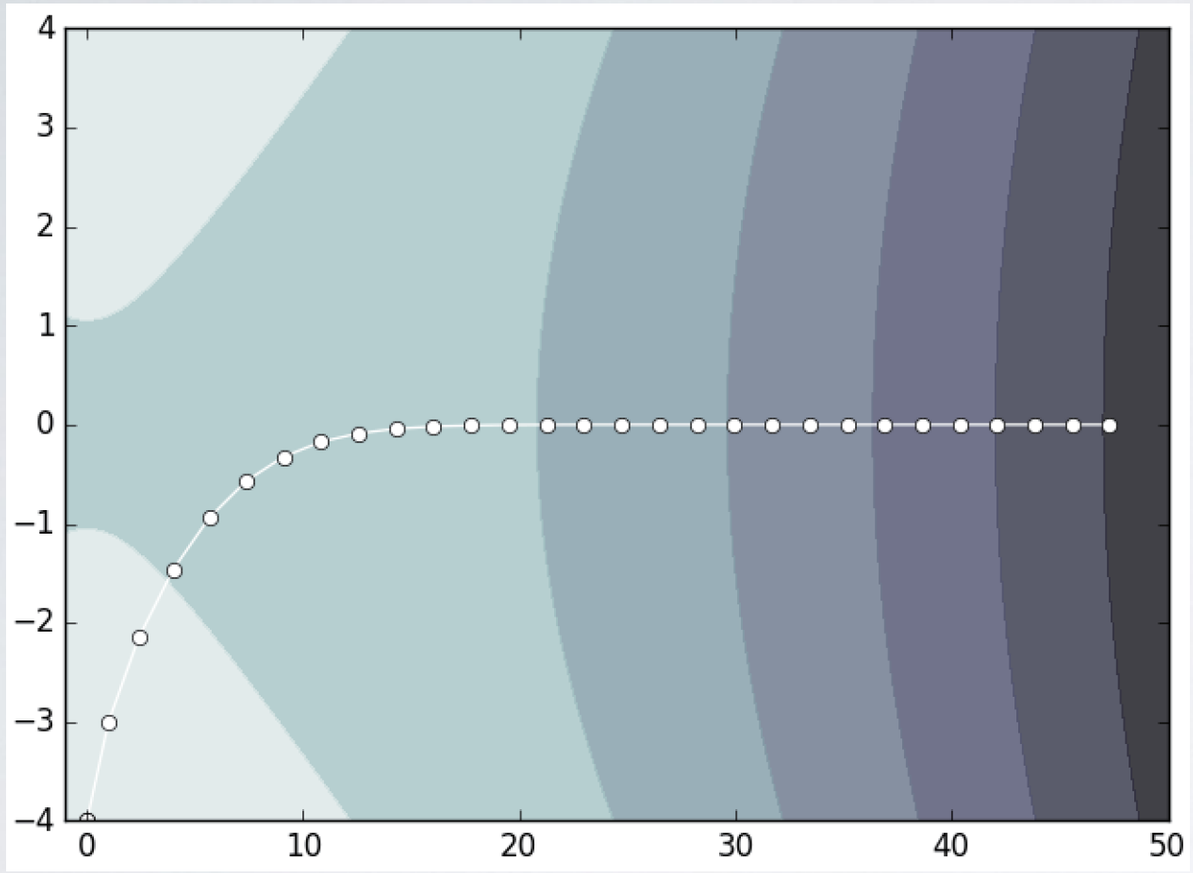
momentum





鞍点を抜け出すのには時間が掛かる。

Adam



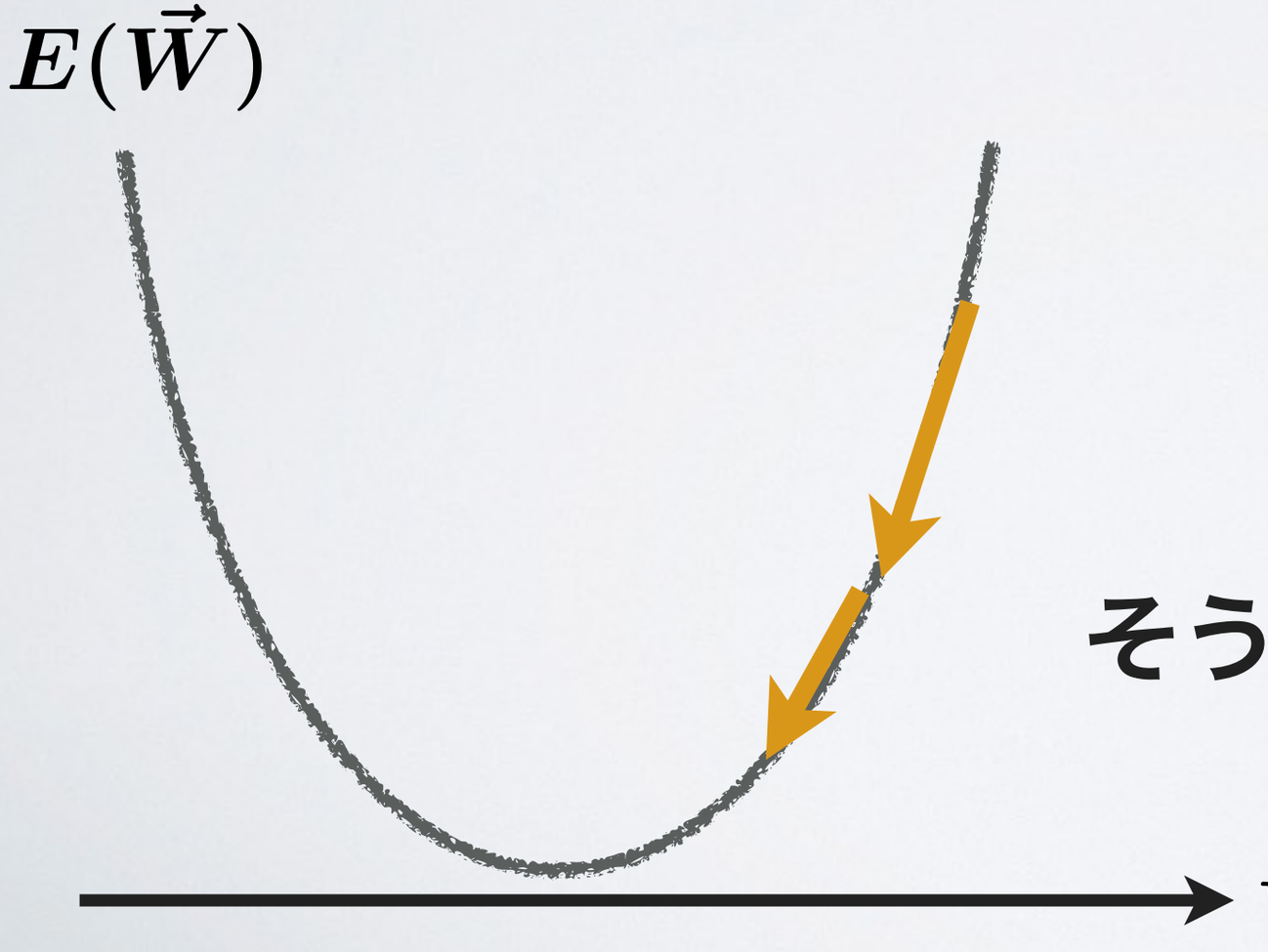
$$\Delta W_{t,i} = -\eta \frac{\widehat{\nabla_i E(W_t)}}{\sqrt{\widehat{(\nabla_i E(W_t))^2}}}$$

局所最適解問題の解消の謎*

ー深層学習のtrainabilityー

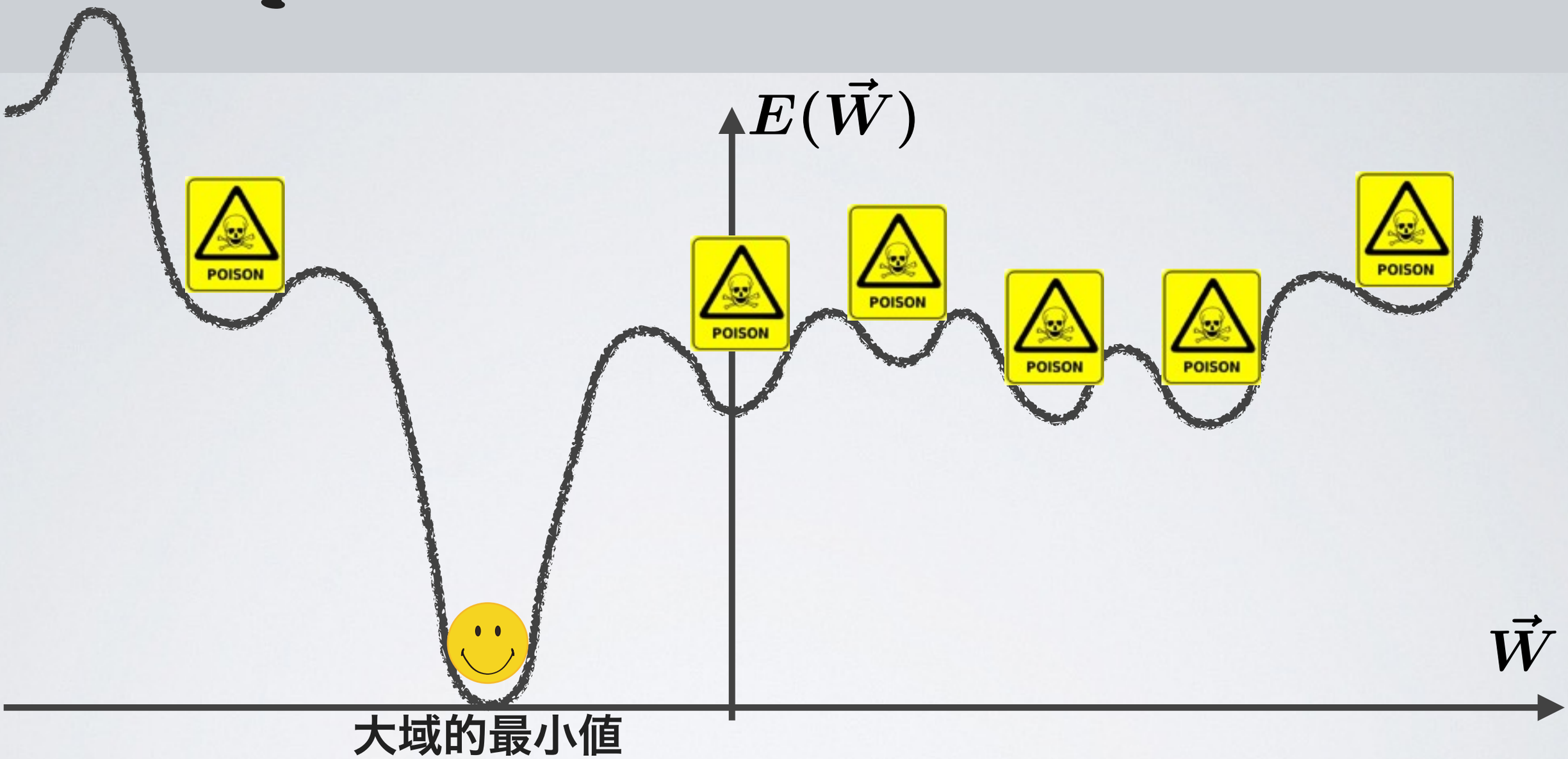
勾配が求まったので勾配降下方で底を探せる?

$$\vec{W} \leftarrow \vec{W} - \eta \nabla_{\vec{W}} E(\vec{W})$$



そうはいかない(late 80's〜)

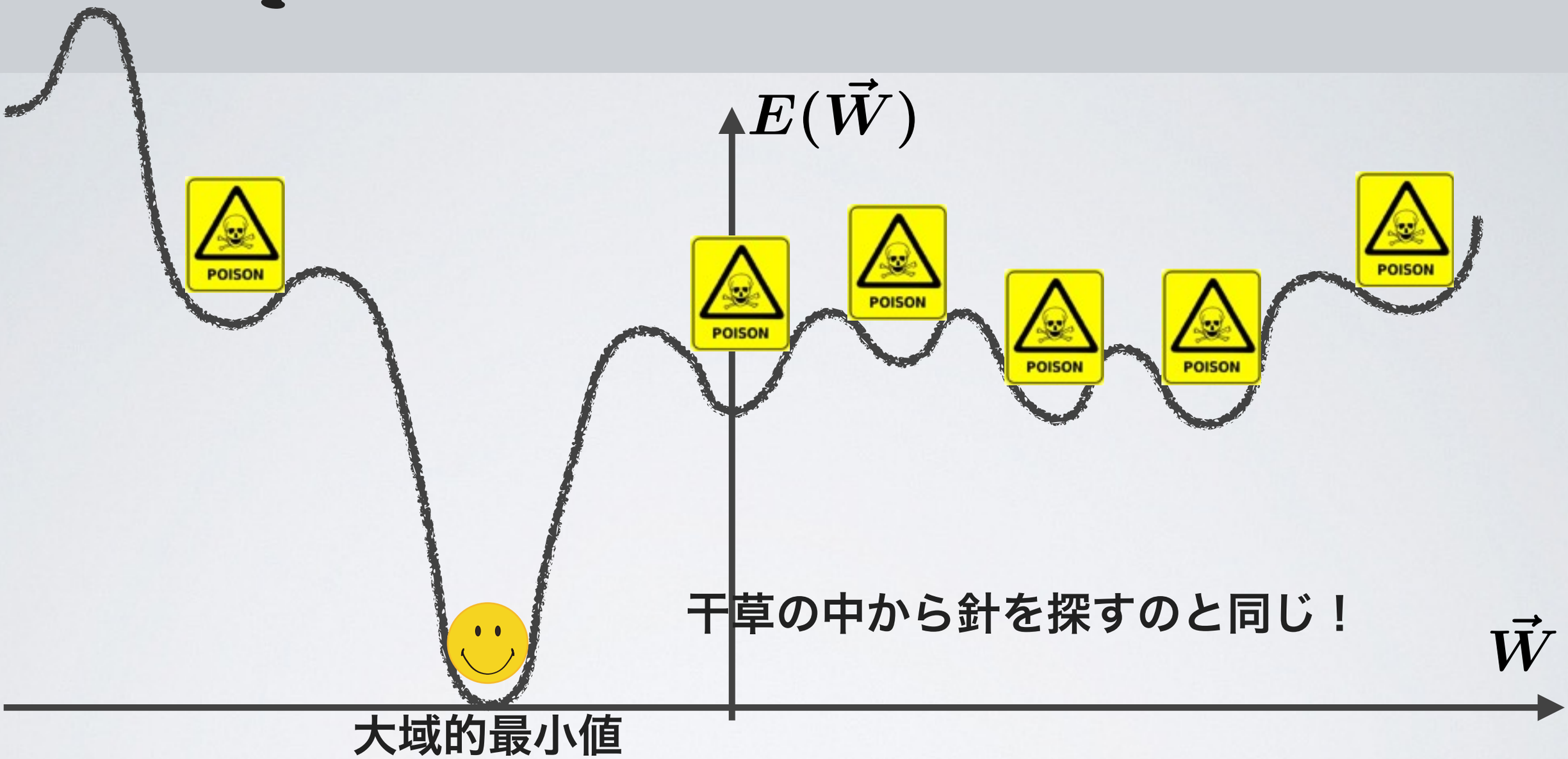
landscape of minima



非凸関数の最適化なので、多量の極小値(局所最適解)

非凸は使い物にならないと普通は考えられる

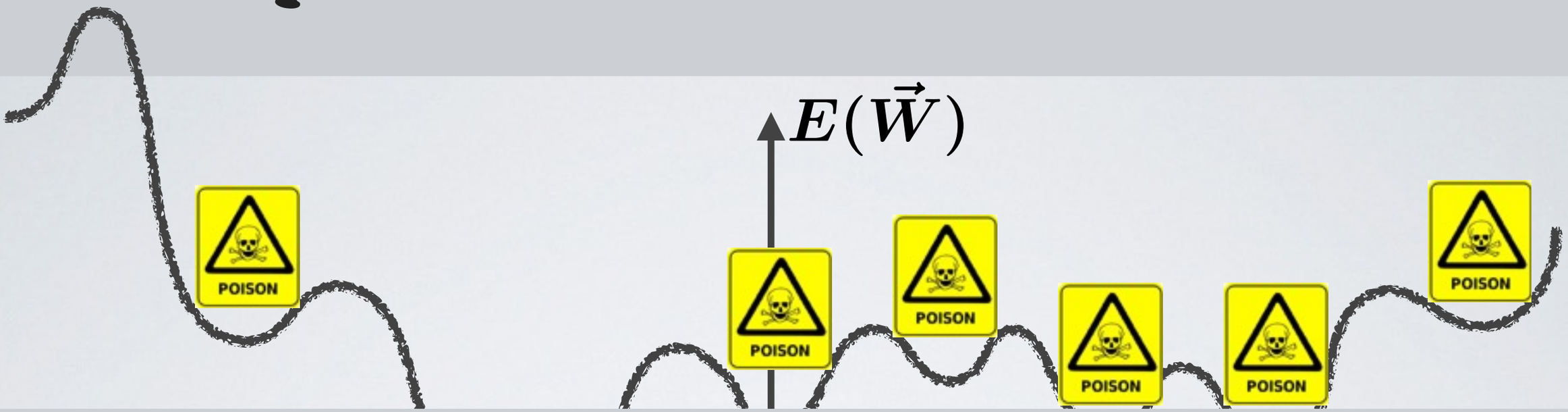
landscape of minima



非凸関数の最適化なので、多量の極小値(局所最適解)

非凸は使い物にならないと普通は考えられる

landscape of minima



でも、深層学習はあんなに上手くいっている！！

非凸関数の最適化なので、多量の極小値(局所最適解)

非凸は使い物にならないと普通は考えられる

trainability of deep learning (observation)

For deep learning with huge parameters,

all local minimum show almost the same nice performance (generalization)

we shouldn't seek global minimum. It's just overfitting.

this makes deep learning outstanding ML system

正則化の不思議
（深層学習の核となる技法（魔法））

Machine Learning and Generalization

本来最小化すべき目的関数: 汎化誤差

$$E(W) = \mathbb{E}_{x \sim P_{data}} \left[e(x; W) \right]$$

Machine Learning and Generalization

本来最小化すべき目的関数: 汎化誤差

$$E(W) = \mathbb{E}_{x \sim P_{data}} \left[e(x; W) \right]$$

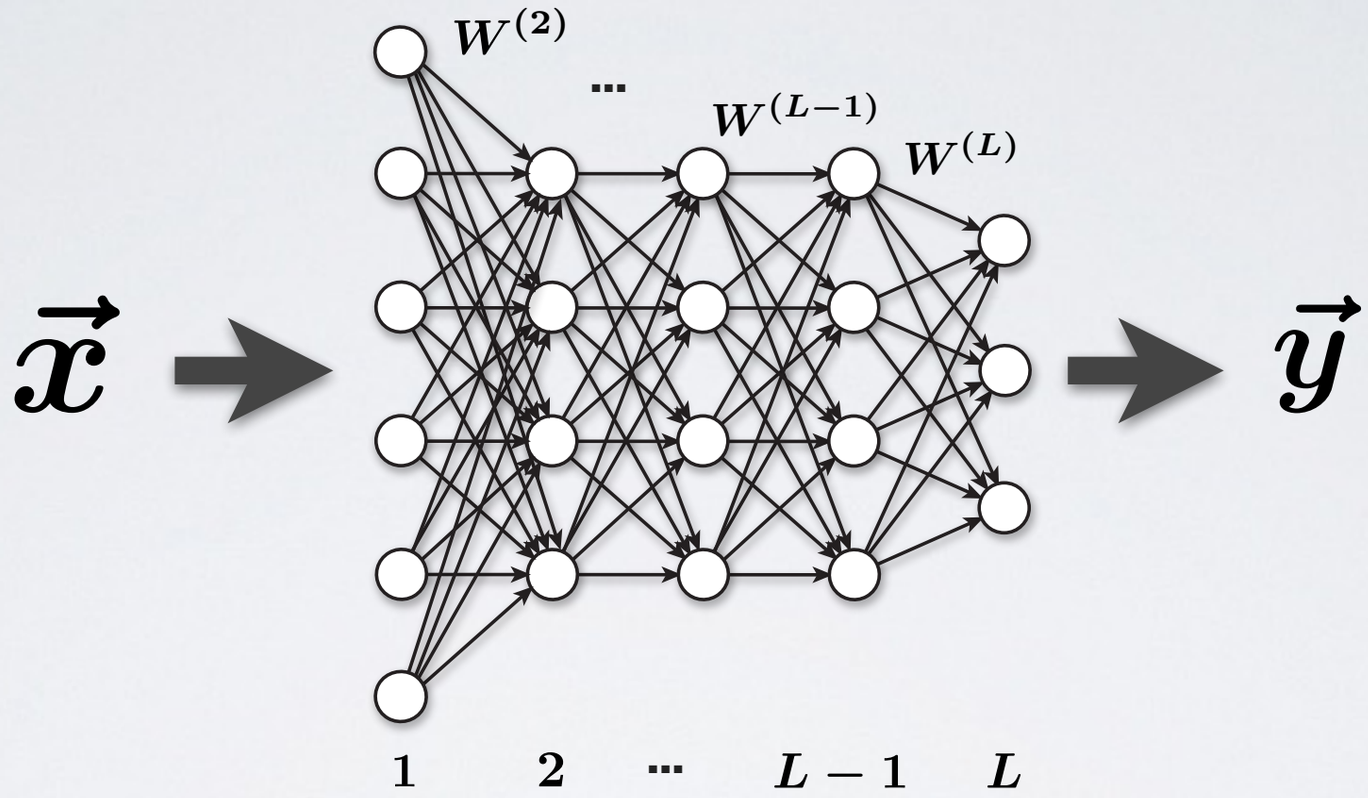
我々が使える目的関数: 訓練誤差

$$E_{train}(W) = \mathbb{E}_{x \sim P_{empirical}} \left[e(x; W) \right]$$

$$= \frac{1}{N} \sum_{n=1}^{N} e(x_n; W)$$

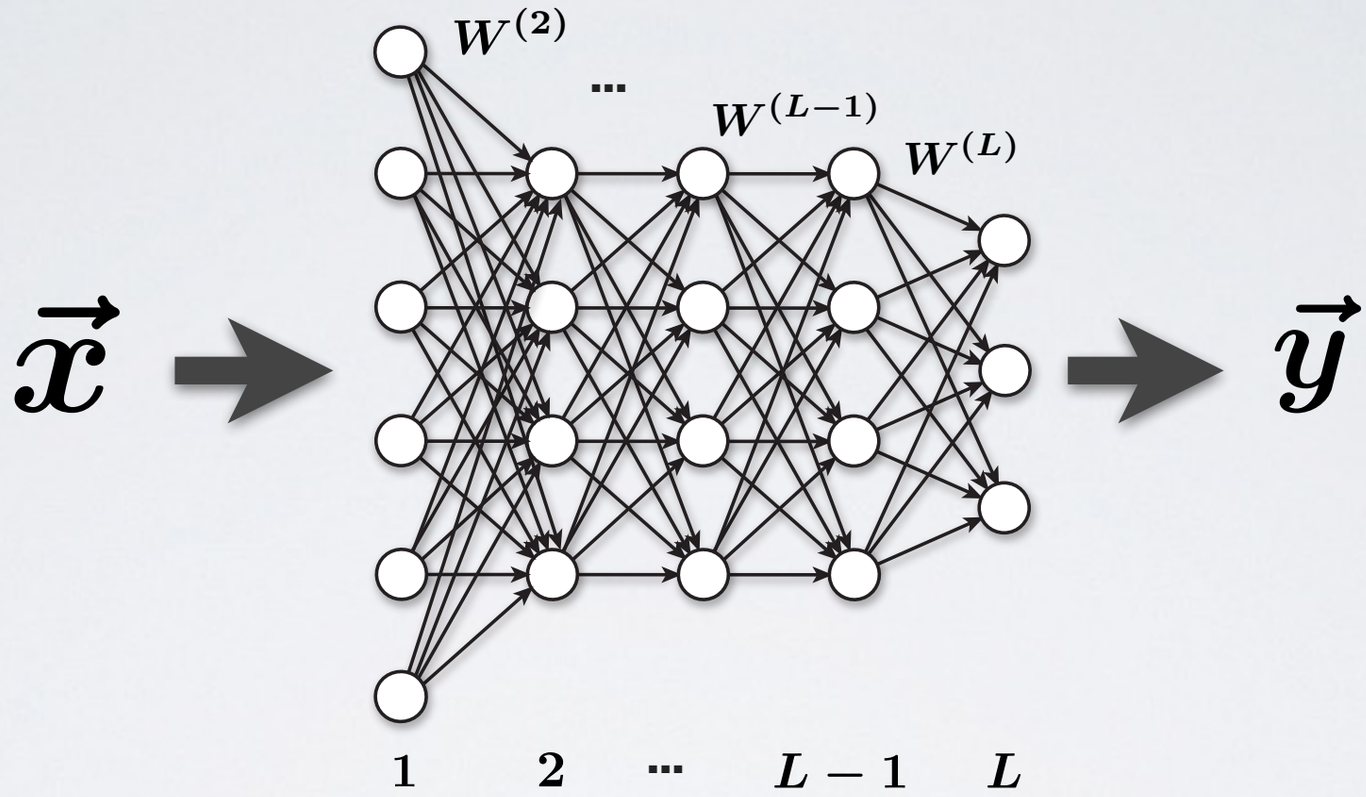
このズレが過学習を引き起こす

deep neural network



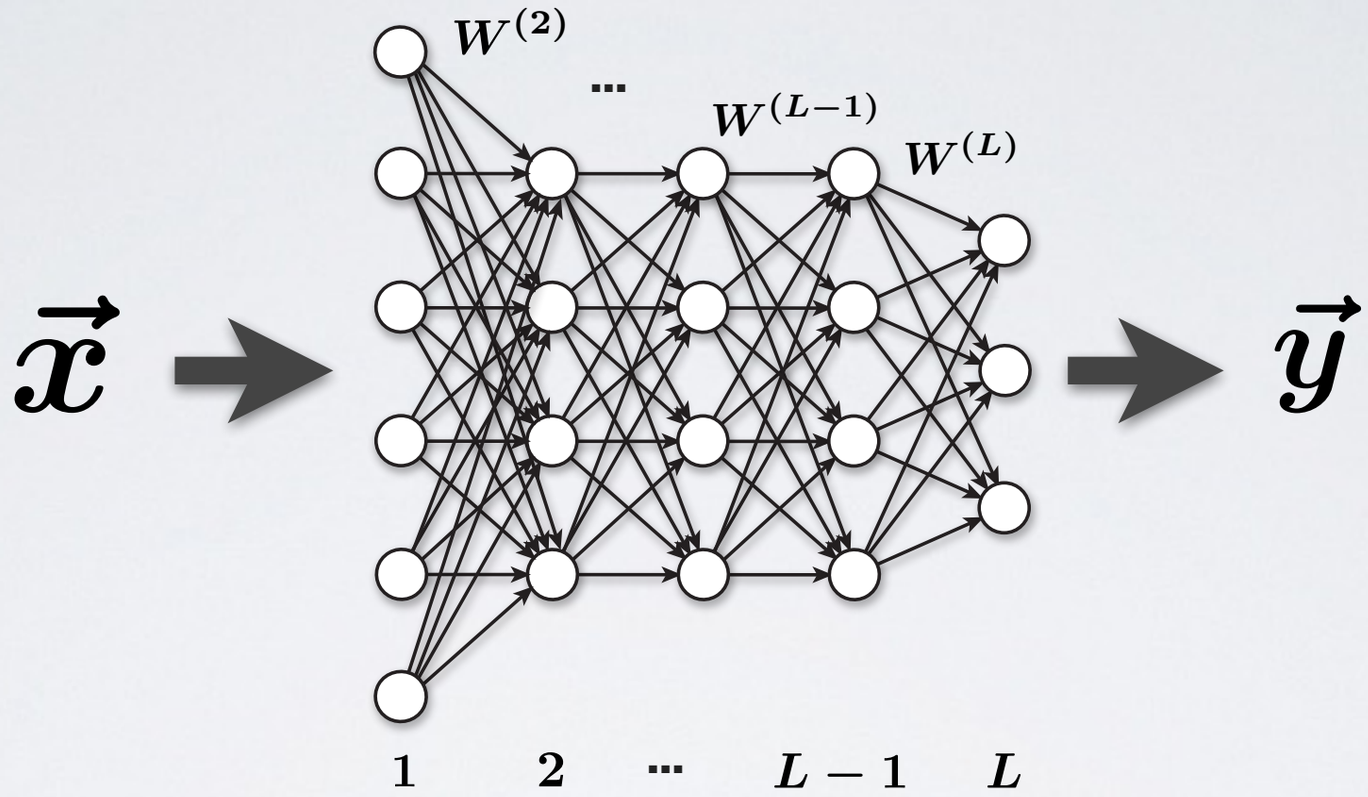
eg. (100x100)x(100x100)x10=10^9 parameters

deep neural network



eg. (100x100)x(100x100)x10=10^9 parameters
recent DL's have > 100 layers with >1000 units

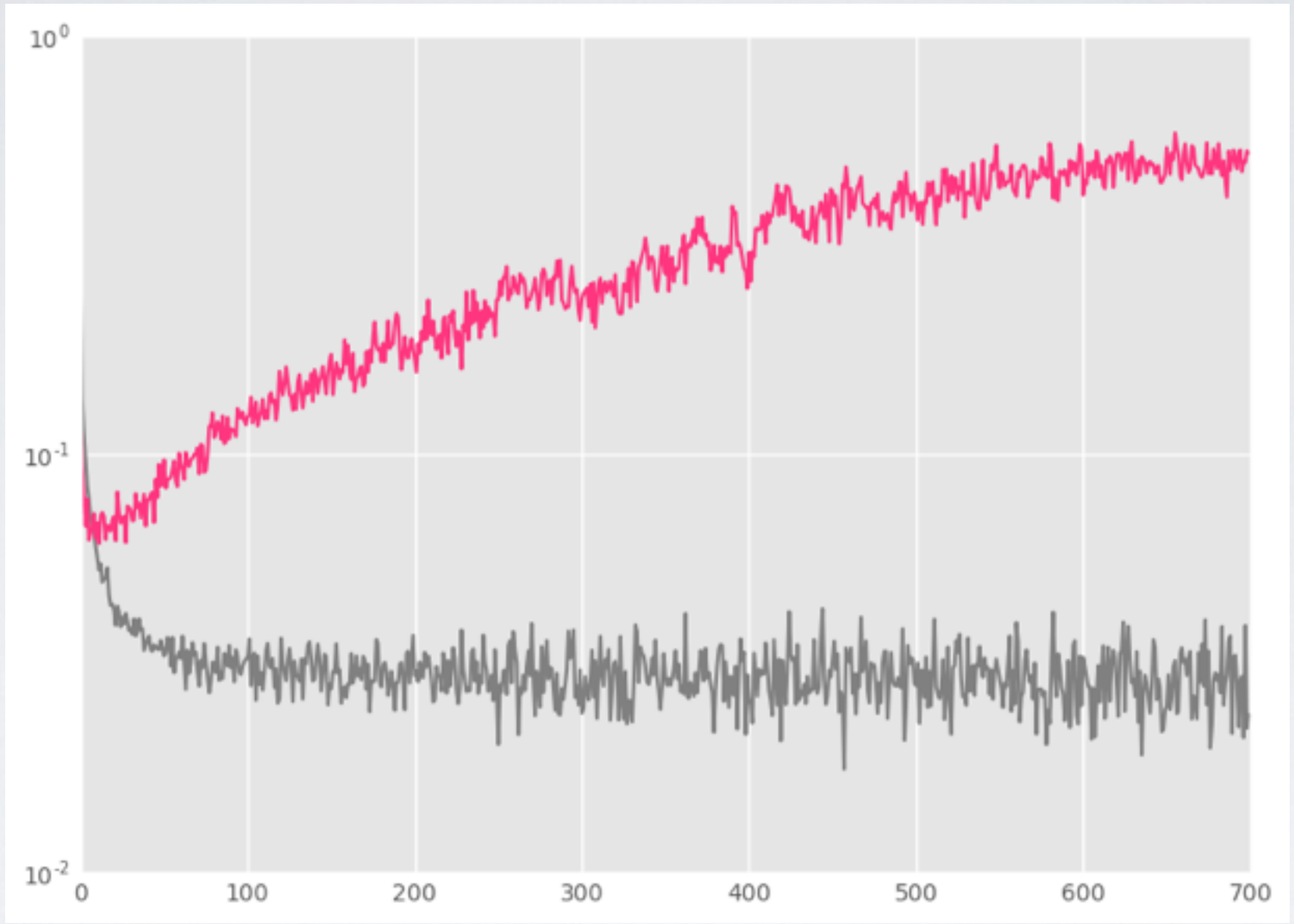
deep neural network



eg. (100x100)x(100x100)x10=10^9 parameters
recent DL's have > 100 layers with >1000 units

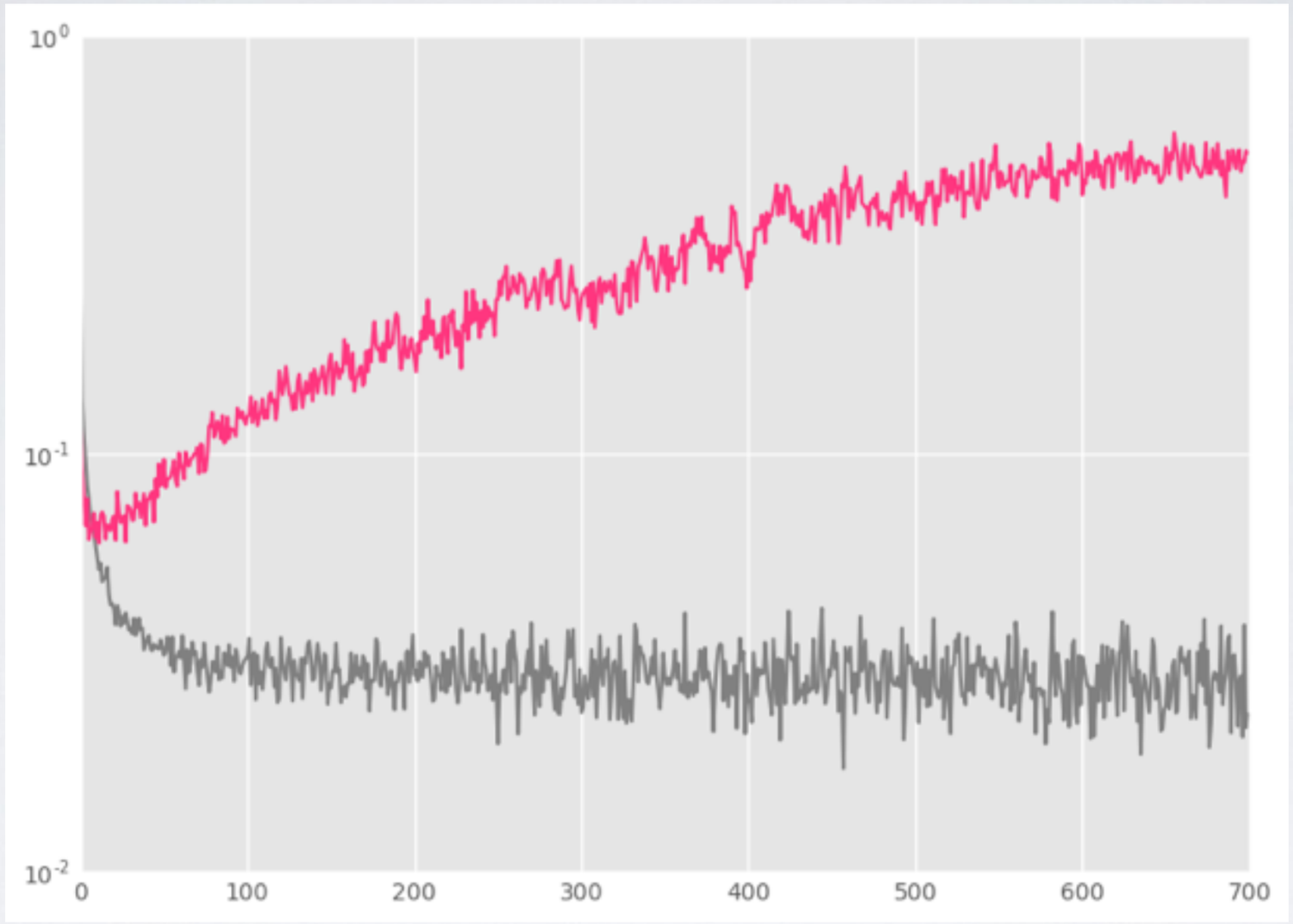
naive fitting leads to overfitting

deep neural network



naive fitting leads to overfitting

deep neural network



naive fitting leads to overfitting

→ **regularization of the network system**

deep neural network

実は、はじめは巨大なモデルを用意して、そのパラメータを正則化で抑制しながら学習させられることが、深層学習を他の手法と比べて飛び抜けた方法にしている

→ **regularization of the network system**

weight decay (old fashioned)

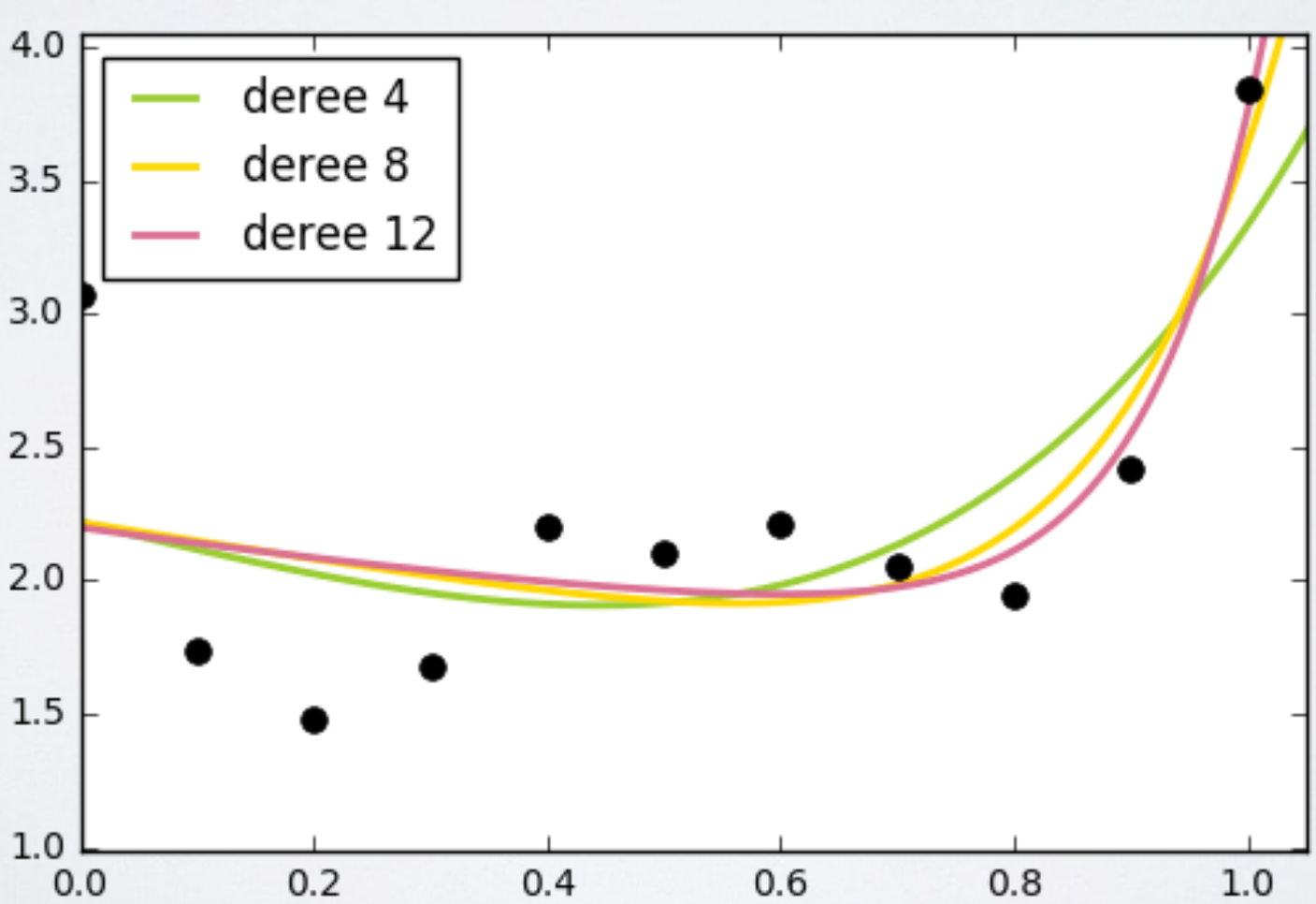
$$E(W) \rightarrow E(W) + \lambda \sum \left(W_{ij}^{(\ell)}\right)^2$$

preference of model

weight decay (old fashioned)

$$E(W) \rightarrow E(W) + \lambda \sum \left(W_{ij}^{(\ell)}\right)^2$$

preference of model



weight decay (old fashioned)

$$E(W) \rightarrow E(W) + \lambda \sum \left(W_{ij}^{(\ell)}\right)^2$$

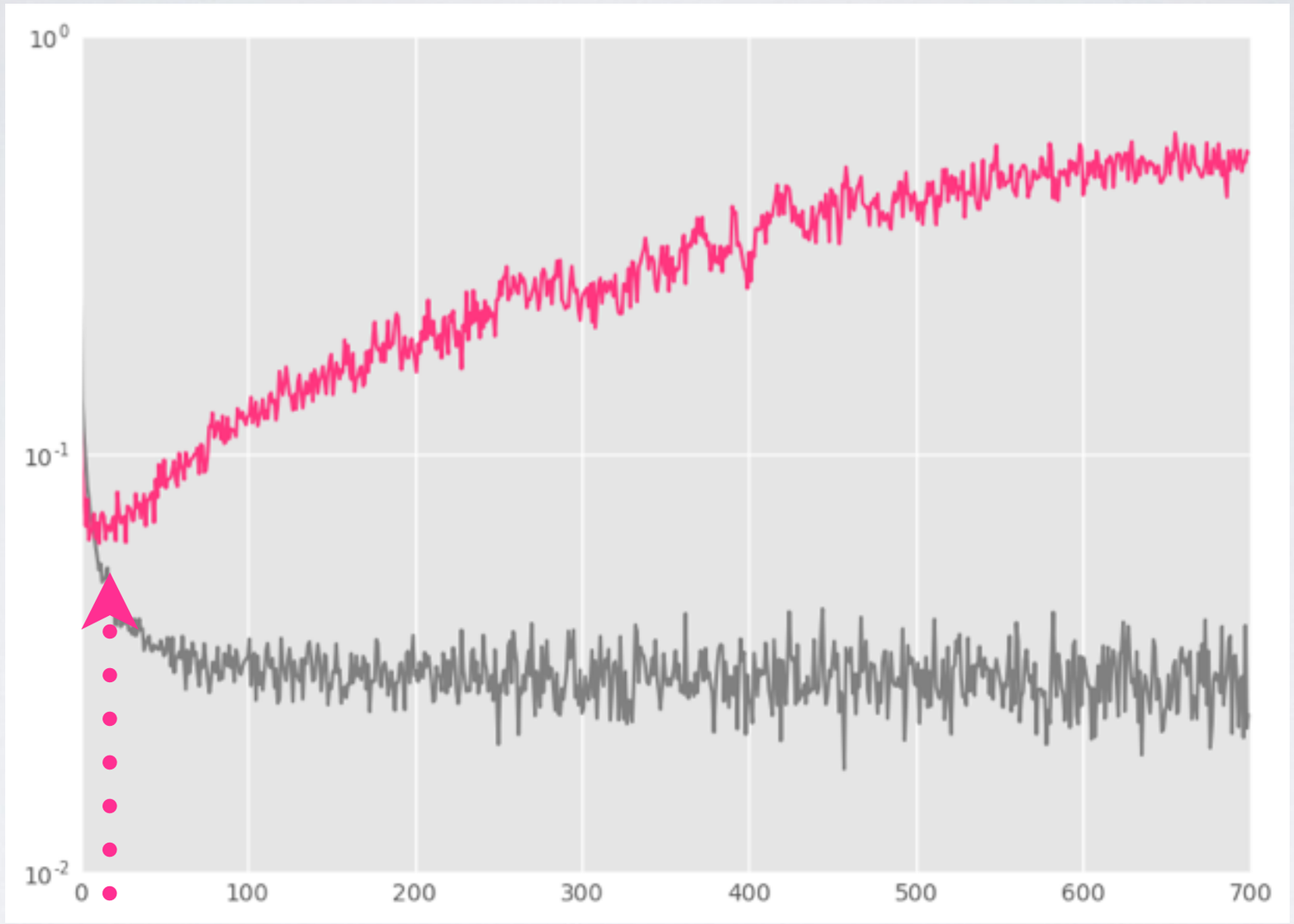
preference of model

sparse regularization

$$E(W) \rightarrow E(W) + \lambda \sum |W_{ij}^{(\ell)}|$$

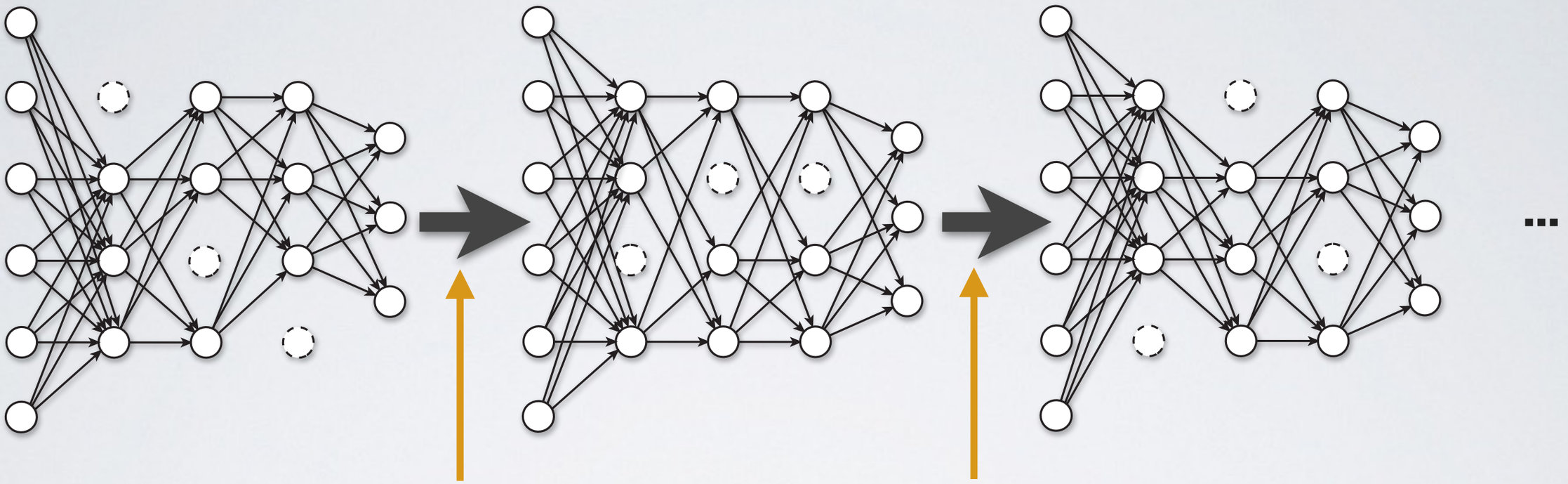
etc

Early Stopping（早期終了）



stop!

drop out method



train by data

$\vec{x}_1$

train by data

$\vec{x}_2$

...

...

avoiding overfitting & realizing generalization

3. 深層学習の成果

ILSVRC (ImageNet Large Scale Visual Recognition Challenge)

ImageNet データセットの一部を用いた画像認識のコ ンペティション(2010-)

様々な大学や企業が機械学習の先端技術を競う, 業界ではとても有名な世界的イベント. 100 万枚ほどの訓練用自然画像 により, 画像を 1000 カテゴリ程に分類する画像認識モデルを訓練し てその性能を互いに競い合う.

ILSVRC (ImageNet Large Scale Visual Recognition Challenge)

2010年　　28%

2011年　　26%

ILSVRC (ImageNet Large Scale Visual Recognition Challenge)

2010年　28%

2011年　26%

pre-DL

2012年 トロント大　16%

ILSVRC (ImageNet Large Scale Visual Recognition Challenge)

2010年　28%

2011年　26%

pre-DL

2012年 トロント大　16%

2013年 ニューヨーク大 11.7%

ILSVRC (ImageNet Large Scale Visual Recognition Challenge)

2010年　28%

2011年　26%

pre-DL

2012年 トロント大　16%

2013年 ニューヨーク大 11.7%

2014年 Google　6.6%

ILSVRC (ImageNet Large Scale Visual Recognition Challenge)

2010年　28%

2011年　26%

pre-DL

2012年 トロント大　16%

2013年 ニューヨーク大 11.7%

2014年 Google　6.6%

人間の誤答率5.1%

2015年 Microsoft　3.57%

2016年 Trimps-Soushen　2.99%

ILSVRC (ImageNet Large Scale Visual Recognition Challenge)

2010年　28%

2011年　26%

pre-DL

2012年 トロント大　16%　8層

2013年 ニューヨーク大 11.7%　8層

2014年 Google　6.6%　22層

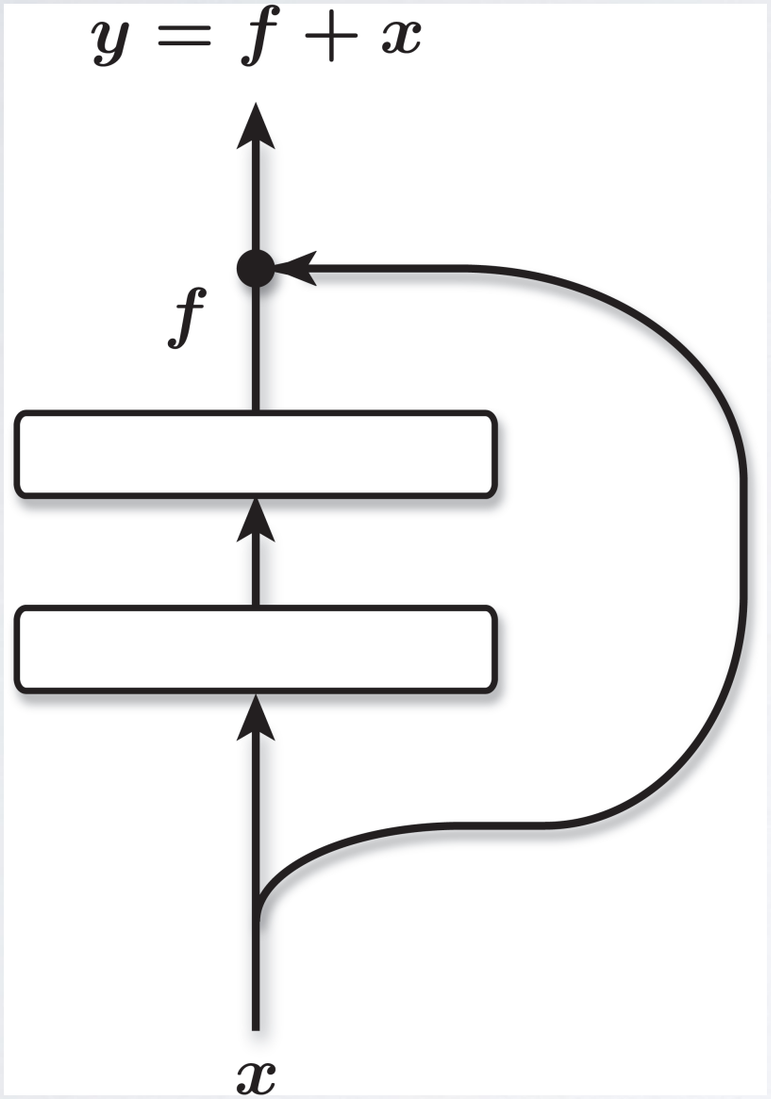
人間の誤答率5.1%

2015年 Microsoft　3.57%　152層

2016年 Trimps-Soushen　2.99%

超深層ネット — ResNet*

ResNet



Information is lost as propagating through the layers.
bypath prevents losing information through propagation

ResNet

we can compute 1000-layer Deep Learning!

画像からキャプション生成*

[Google, 2014]

A person riding a motorcycle on a dirt road.

A group of young people playing a game of frisbee.

A herd of elephants walking across a dry grass field.

Two dogs play in the grass.

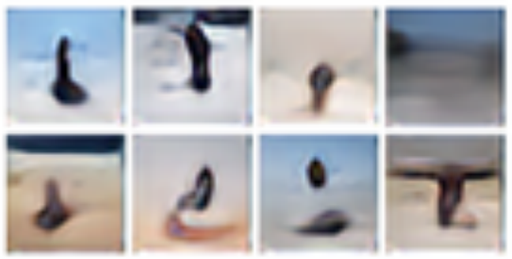
キャプションから画像生成*

Text2Image [Mansimov et al, 2015]

A stop sign is flying in blue skies.

A herd of elephants flying in the blue skies.

A toilet seat sits open in the grass field.

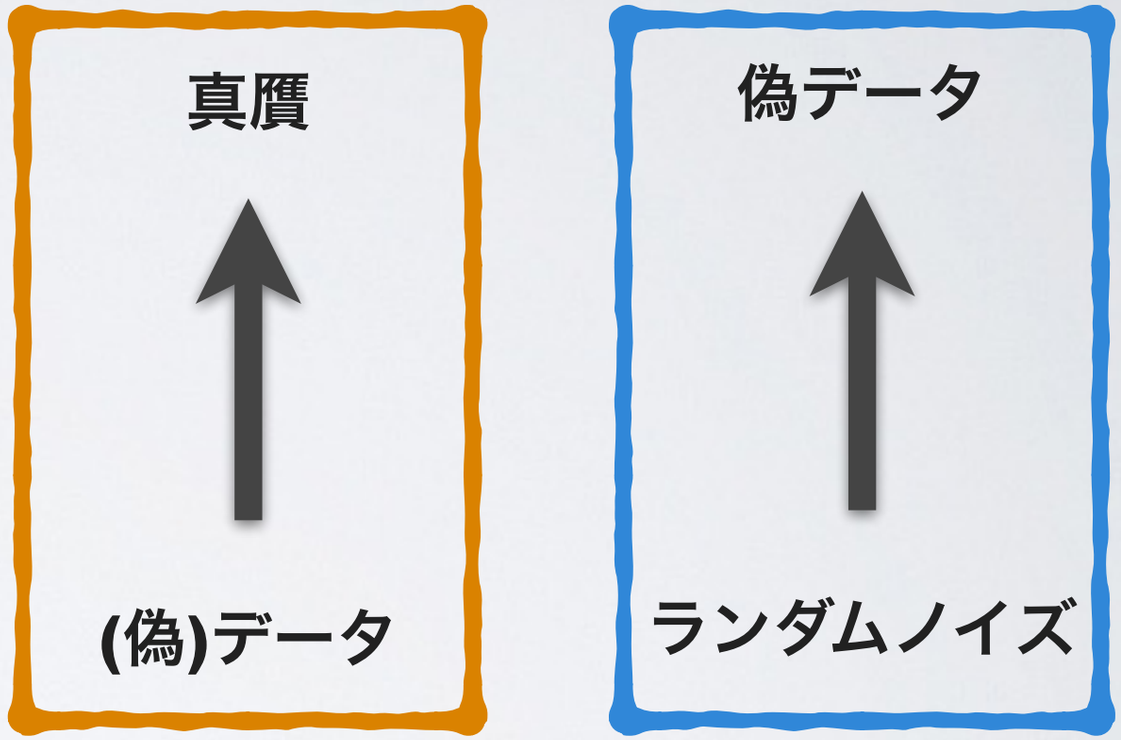


A person skiing on sand clad vast desert.

Figure 1: Examples of generated images based on captions that describe novel scene compositions that are highly unlikely to occur in real life. The captions describe a common object doing unusual things or set in a strange location.

敵対的生成ネットワーク

贋金造りと警官の「追いかけっこ」ゲーム



discriminator V.S.generator

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathrm{E}_{x \sim P_{data}}\big[\log D(x)\big] + \mathrm{E}_{z \sim P_{noise}}\big[\log\big(1 - D(G(z))\big)\big]$$

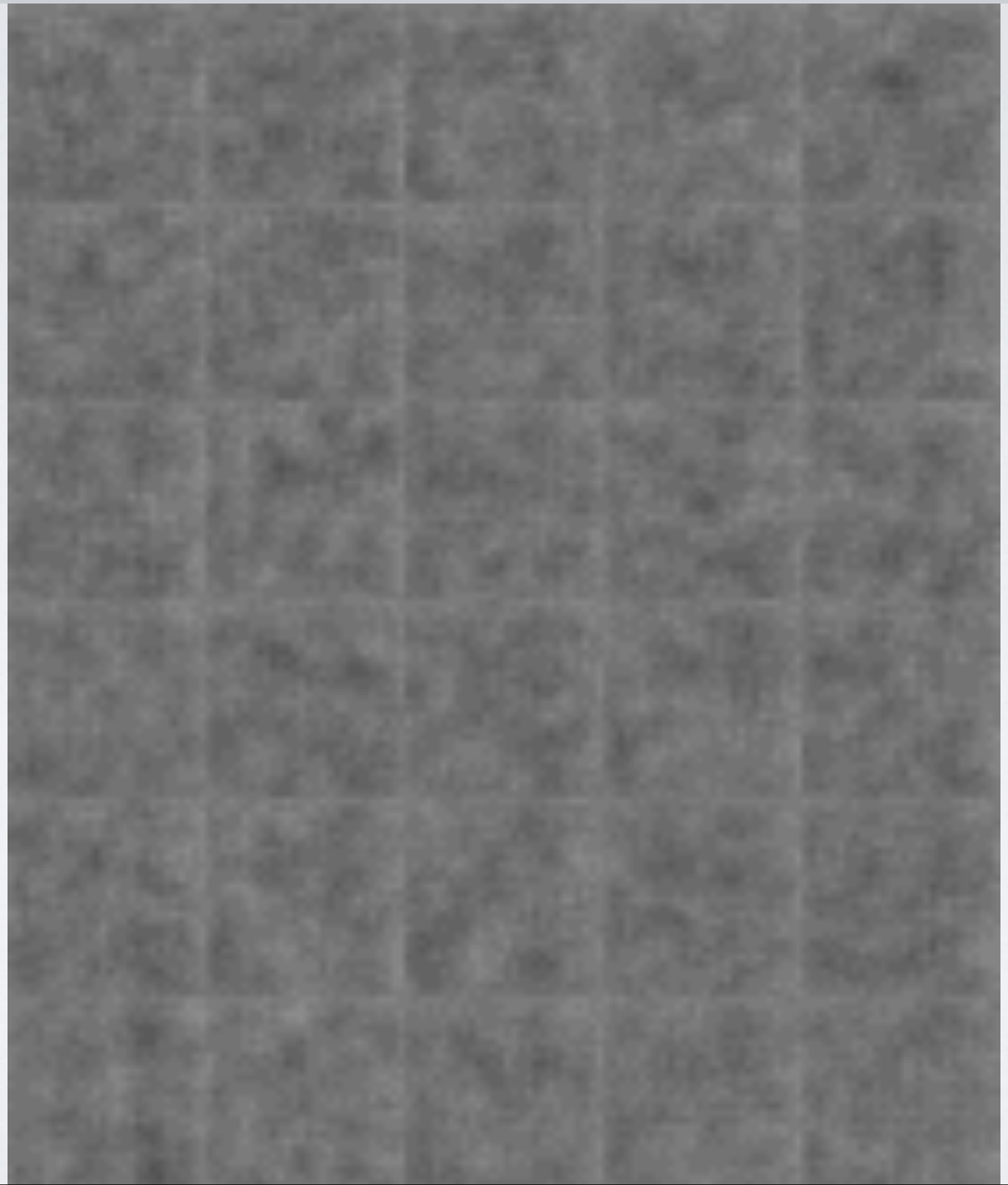
Generative adversarial net (DCGAN)

CPUで数時間かけて実験した結果

写真でうまくやるのは大変なので、MNISTで

(lap topでやったことに意味は何もないので、(試されたい方はラボにあるでしょうし)是非GPGPUでやってください。)

CPUで数時間かけて実験した結果



CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

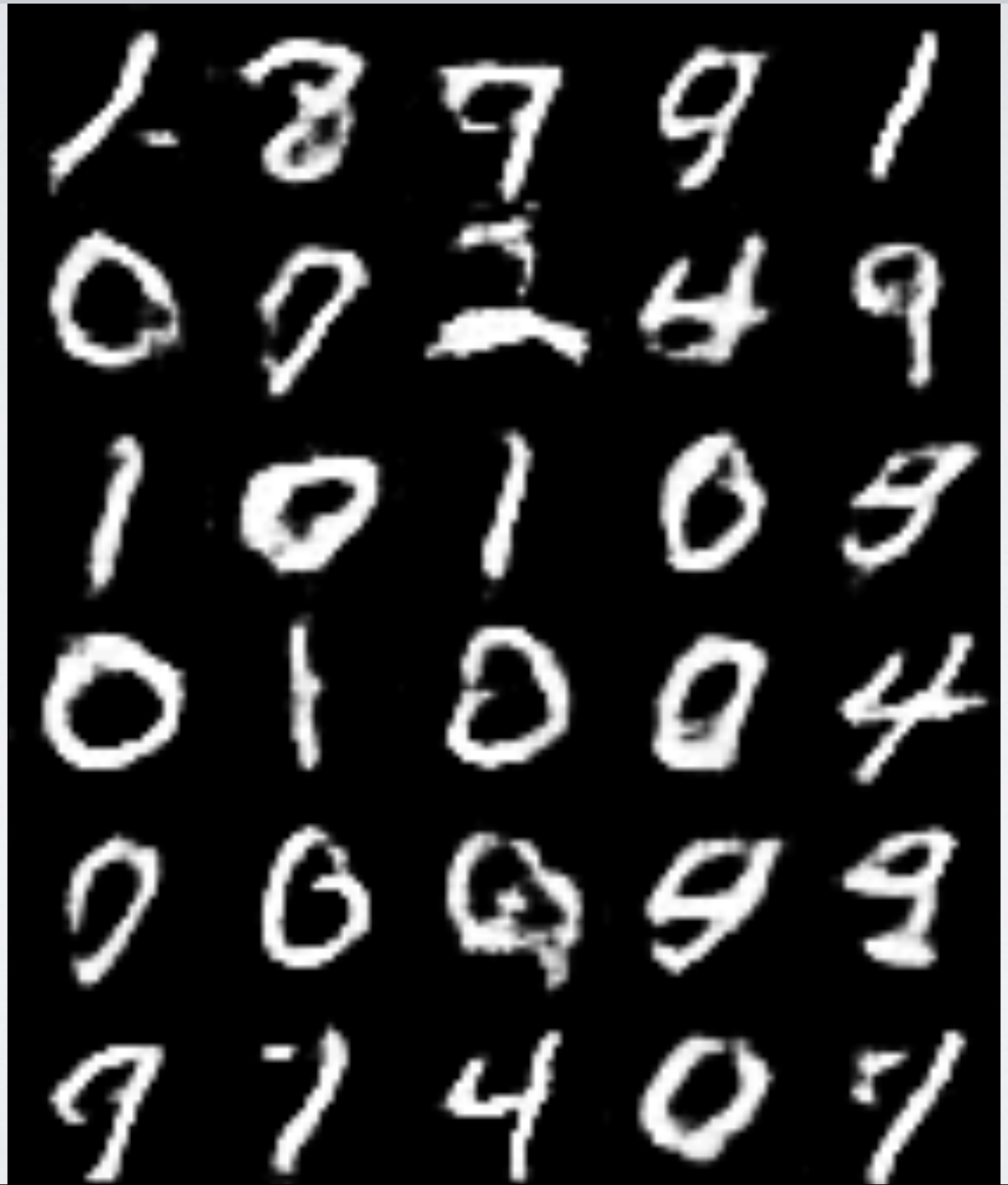
CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果



CPUで数時間かけて実験した結果

と、やっていくうちに...

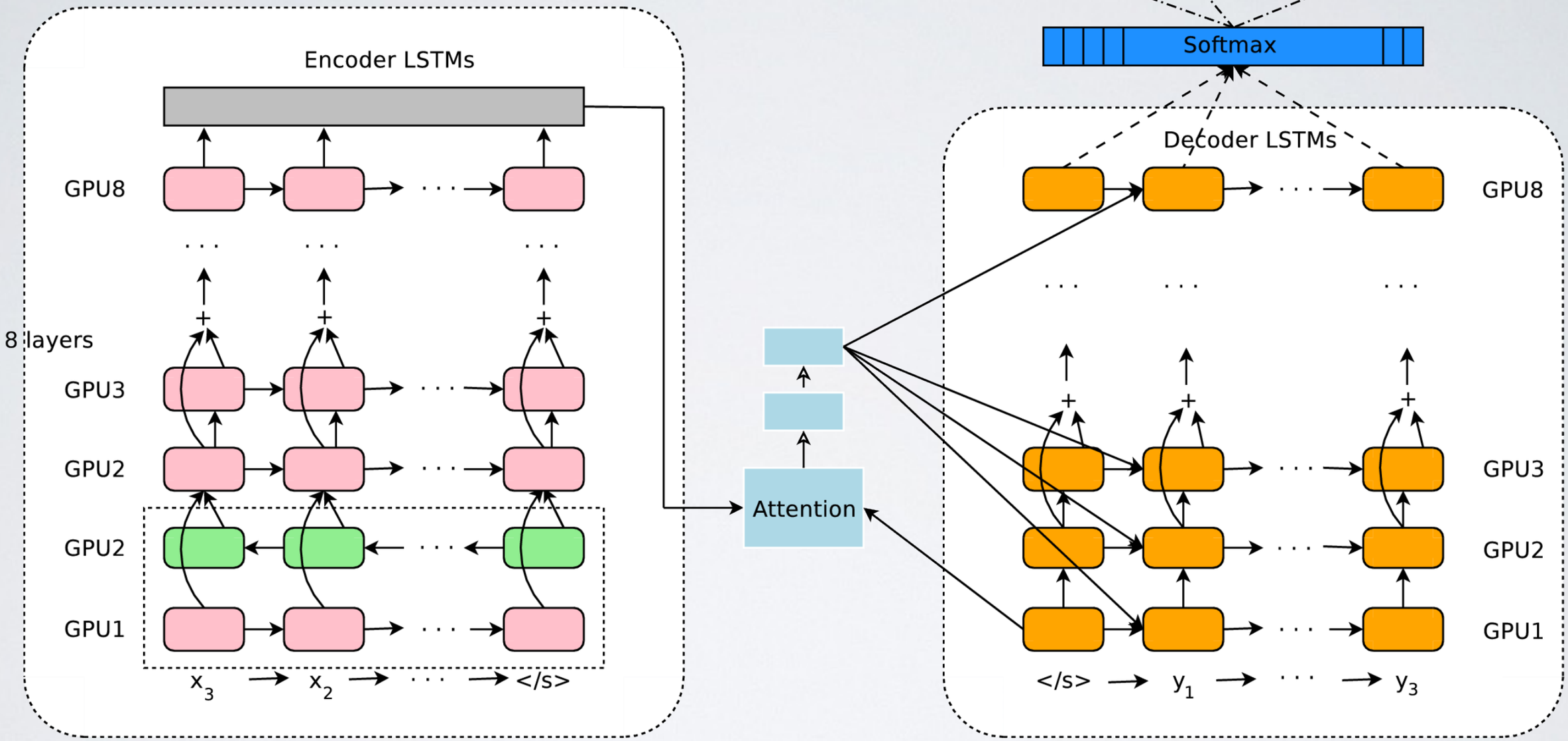
CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

CPUで数時間かけて実験した結果

Google Translation *

Google Translation



Google Translation

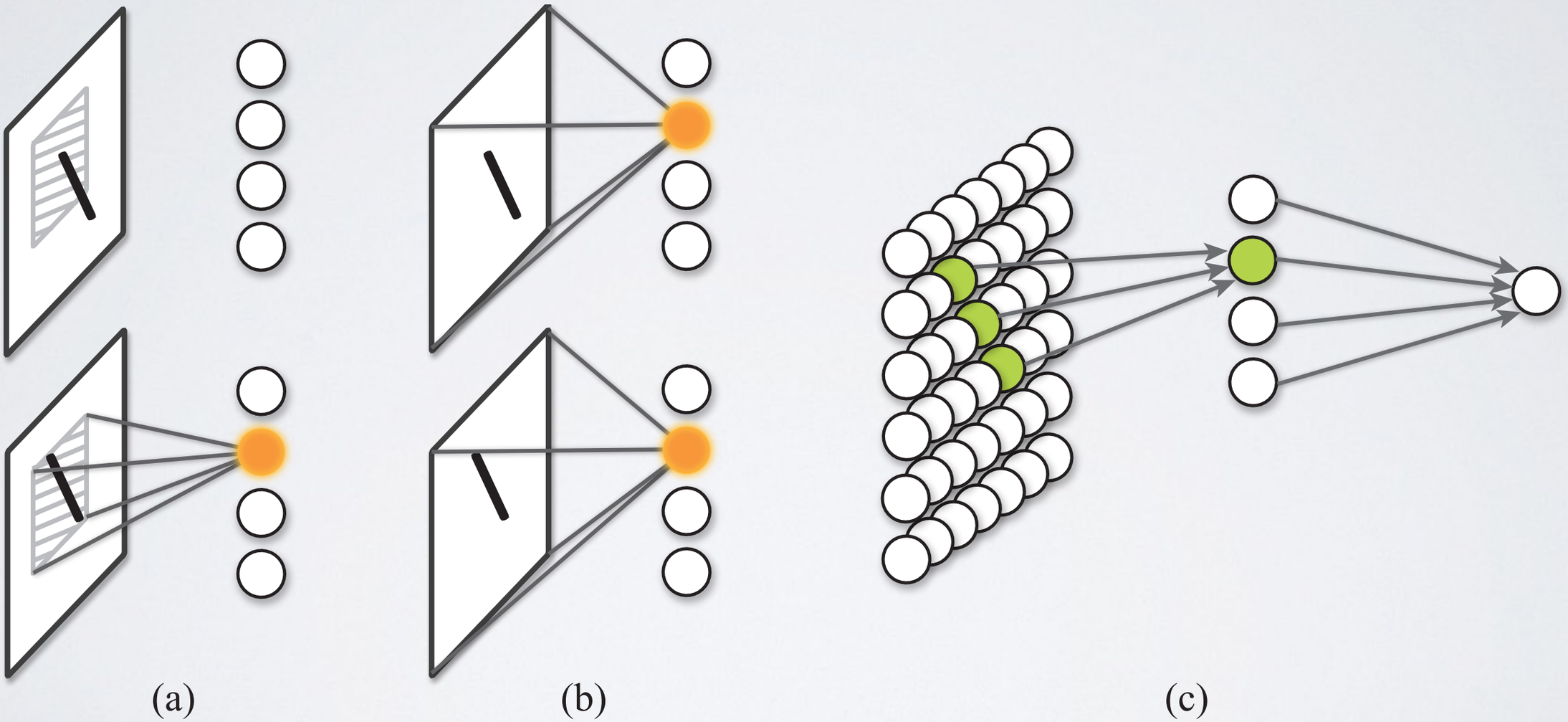
ニューラルマシン翻訳（NMT）は、従来のフレーズベースの翻訳システムの弱点の多くを克服する可能性のある、自動翻訳のためのエンドツーエンドの学習アプローチです。残念なことに、NMTシステムは、訓練と翻訳推論の両方で計算上高価であることが知られています。非常に大規模なデータセットや大型モデルの場合は時折禁止されることがあります。いくつかの著者は、特に入力文にまれな単語が含まれている場合、NMTシステムは堅牢性に欠けていると請求しています。これらの問題は、精度とスピードの両方が重要である実用的な展開やサービスでのNMTの使用を妨げています。この研究では、これらの問題の多くに取り組むGoogleの神経機械翻訳システムGNMTを紹介します。

4. 畳み込みニューラルネット

ヒューベル・ウィーゼルの階層仮説(1958)

ヒューベル・ウィーゼルの階層仮説(1958)

一次視覚野V1でのパターン認識においては



(a)　(b)　(c)

単純型細胞　複雑型細胞　階層仮説

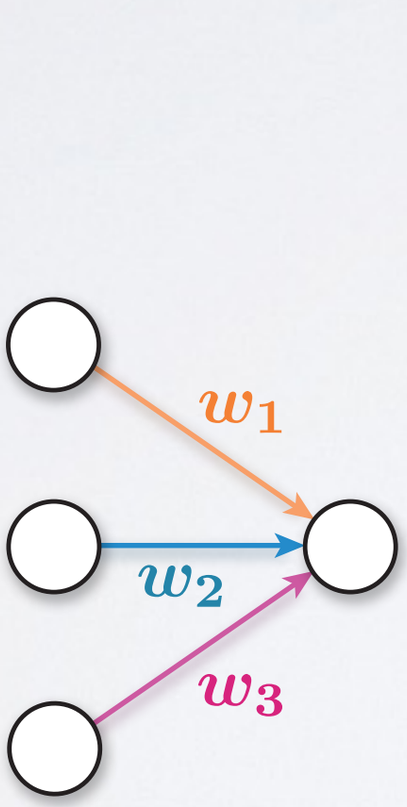
ネオコグニトロン(1979)からLeNet(1989)へ

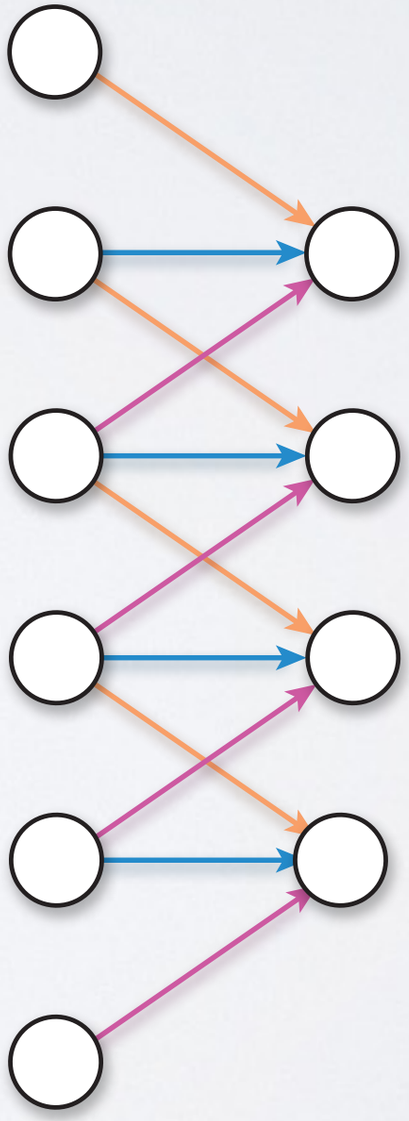
ニューラルネットの結合も、階層仮説を満たすように、単純型細胞の層（畳み込み層）と複雑型細胞層（プーリング層）の形に作り込む

ネオコグニトロン(1979)からLeNet(1989)へ

ニューラルネットの結合も、階層仮説を満たすように、単純型細胞の層（畳み込み層）と複雑型細胞層（プーリング層）の形に作り込む

1. 畳み込み層

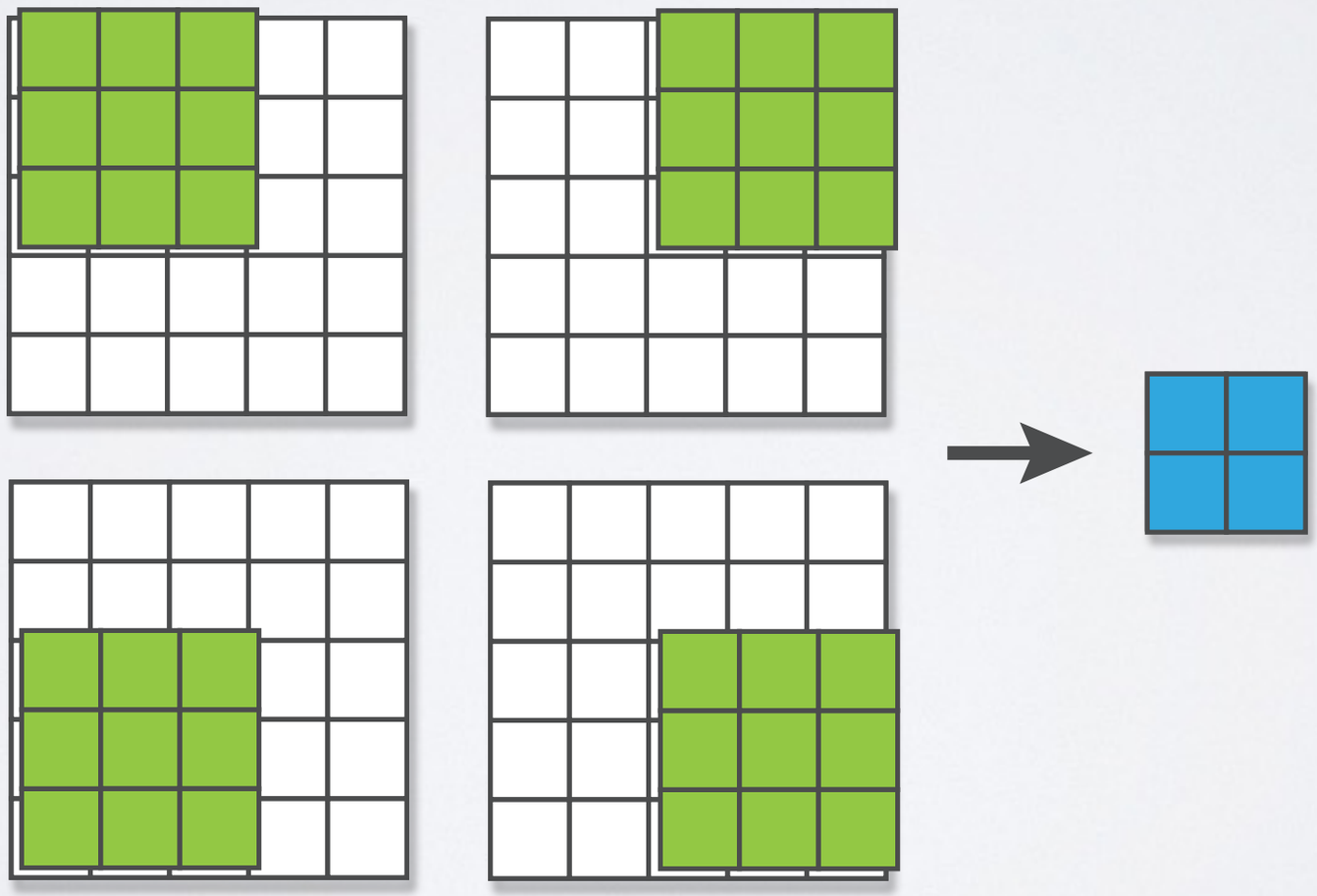




ネオコグニトロン(1979)からLeNet(1989)へ

ニューラルネットの結合も、階層仮説を満たすように、単純型細胞の層（畳み込み層）と複雑型細胞層（プーリング層）の形に作り込む

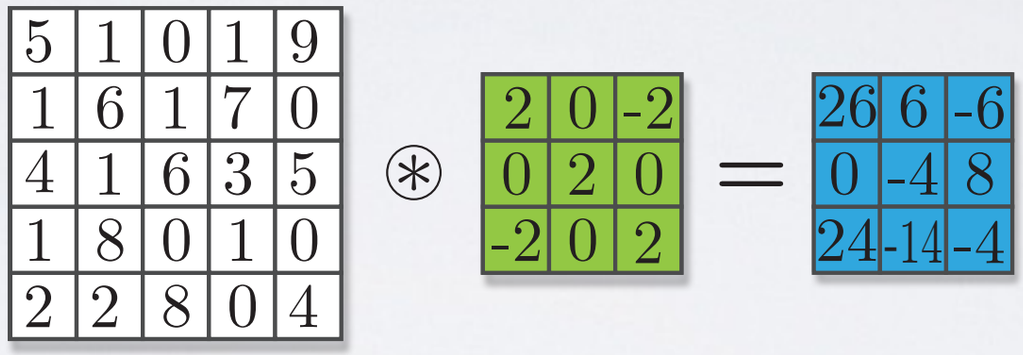
1. 畳み込み層

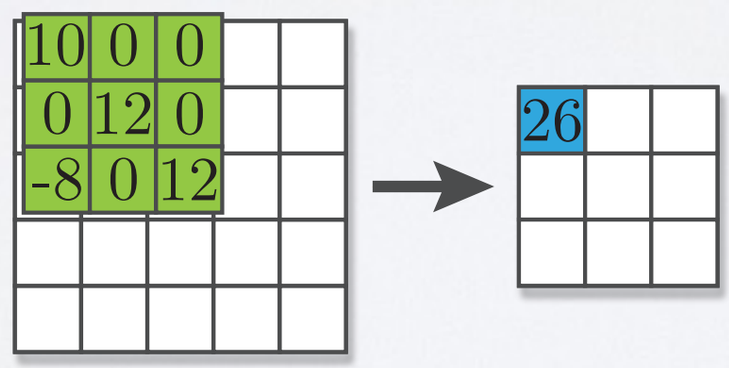


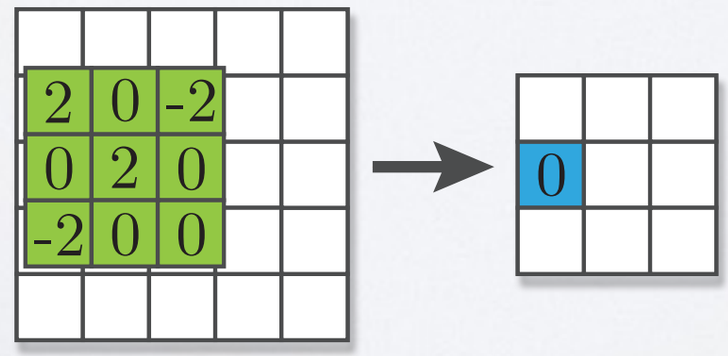
ネオコグニトロン(1979)からLeNet(1989)へ

ニューラルネットの結合も、階層仮説を満たすように、単純型細胞の層（畳み込み層）と複雑型細胞層（プーリング層）の形に作り込む

1. 畳み込み層



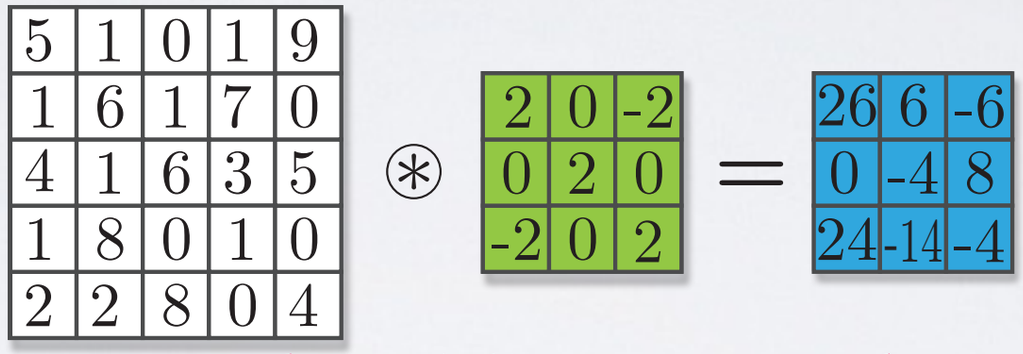




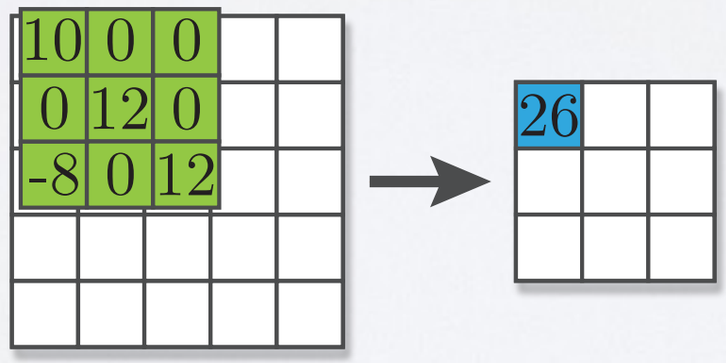
ネオコグニトロン(1979)からLeNet(1989)へ

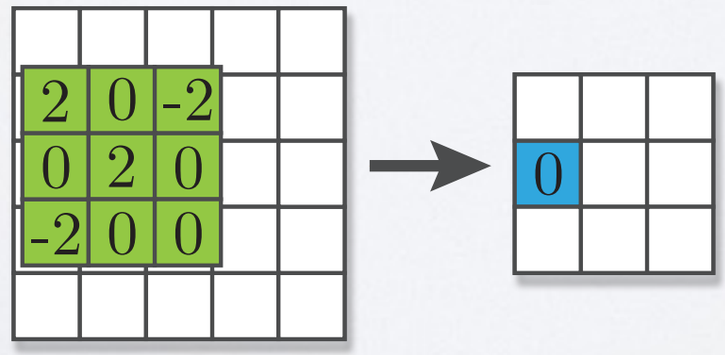
ニューラルネットの結合も、階層仮説を満たすように、単純型細胞の層（畳み込み層）と複雑型細胞層（プーリング層）の形に作り込む

1. 畳み込み層



このフィルターの重みが学習すべき重みパラメータ

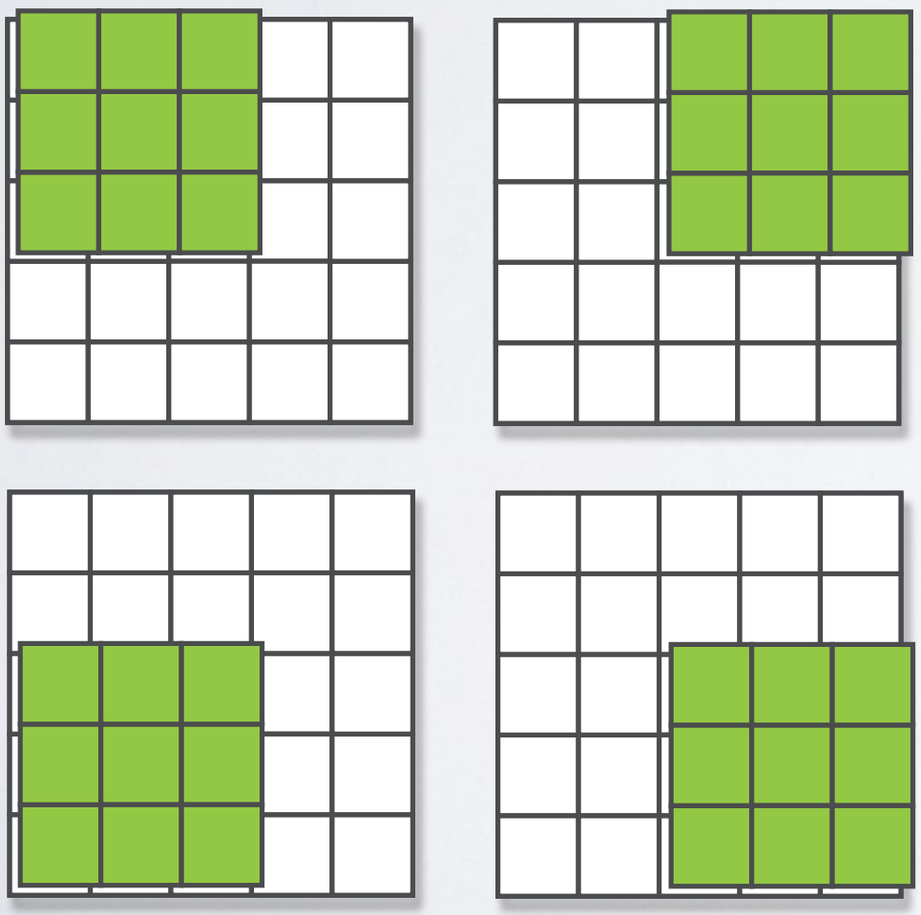




ネオコグニトロン(1979)からLeNet(1989)へ

ニューラルネットの結合も、階層仮説を満たすように、単純型細胞の層（畳み込み層）と複雑型細胞層（プーリング層）の形に作り込む

2. プーリング層



$$z^{(2)} = \frac{1}{2^2} \sum_{i,j} z_{ij}^{(1)}$$

$$z^{(2)} = \max z_{ij}^{(1)}$$

ネオコグニトロン(1979)からLeNet(1989)へ

1979年 福島邦彦（NHK放送科学基礎研究所）

ネオコグニトロンを提唱

学習アルゴリズムがまだ現代的ではない

ネオコグニトロン(1979)からLeNet(1989)へ

1979年 福島邦彦（NHK放送科学基礎研究所）

ネオコグニトロンを提唱

学習アルゴリズムがまだ現代的ではない

1989年 LeCun（AT&T Bell研究所）

LeNetを提唱

誤差逆伝播法を用い、現在の畳み込みニューラルネットそのもの

多層化に成功し、手書き数字認識で高い性能を実現

ネオコグニトロン(1979)からLeNet(1989)へ

現代の畳み込みニューラルネットによる深層学習は、自然画像なども扱えるように高性能化させた**LeNet/Neocognitron**に他ならない

1989年 **LeCun**（AT&T Bell研究所）

LeNetを提唱

誤差逆伝播法を用い、現在の畳み込みニューラルネットそのもの

多層化に成功し、手書き数字認識で高い性能を実現

畳み込みニューラルネット(CNN)のための工夫

勾配消失の回避

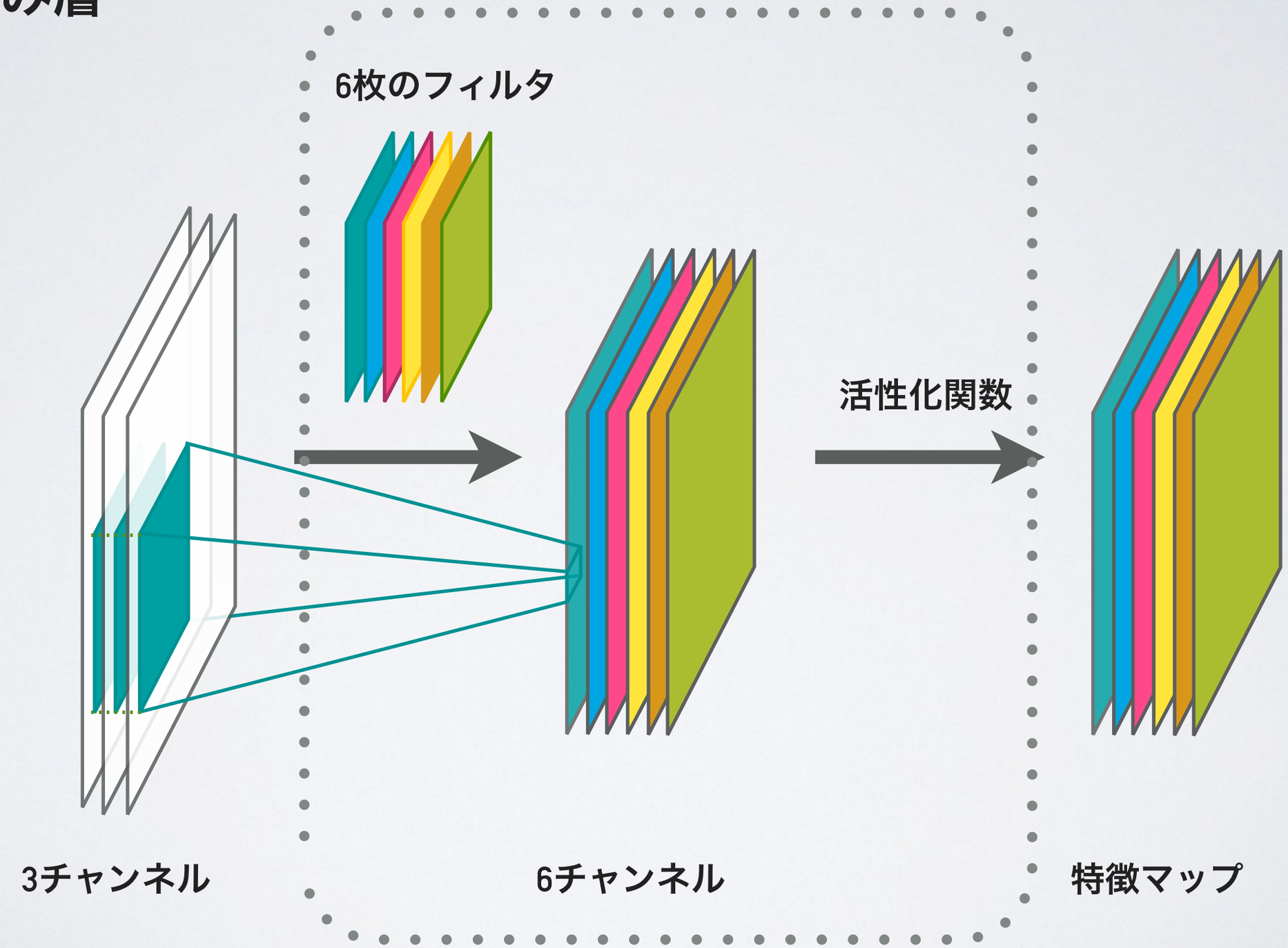
- 活性化関数を**LeRU**など良いものに取る
- 事前学習・転移学習を用いる（後述）
- **GPGPU**での並列化

過学習の抑制

- 学習用データの大規模化、高品質化、標準化、水増し
- ドロップアウトなどの正則化
- アーキテクチャの工夫（インセプション）

畳み込みニューラルネットCNN

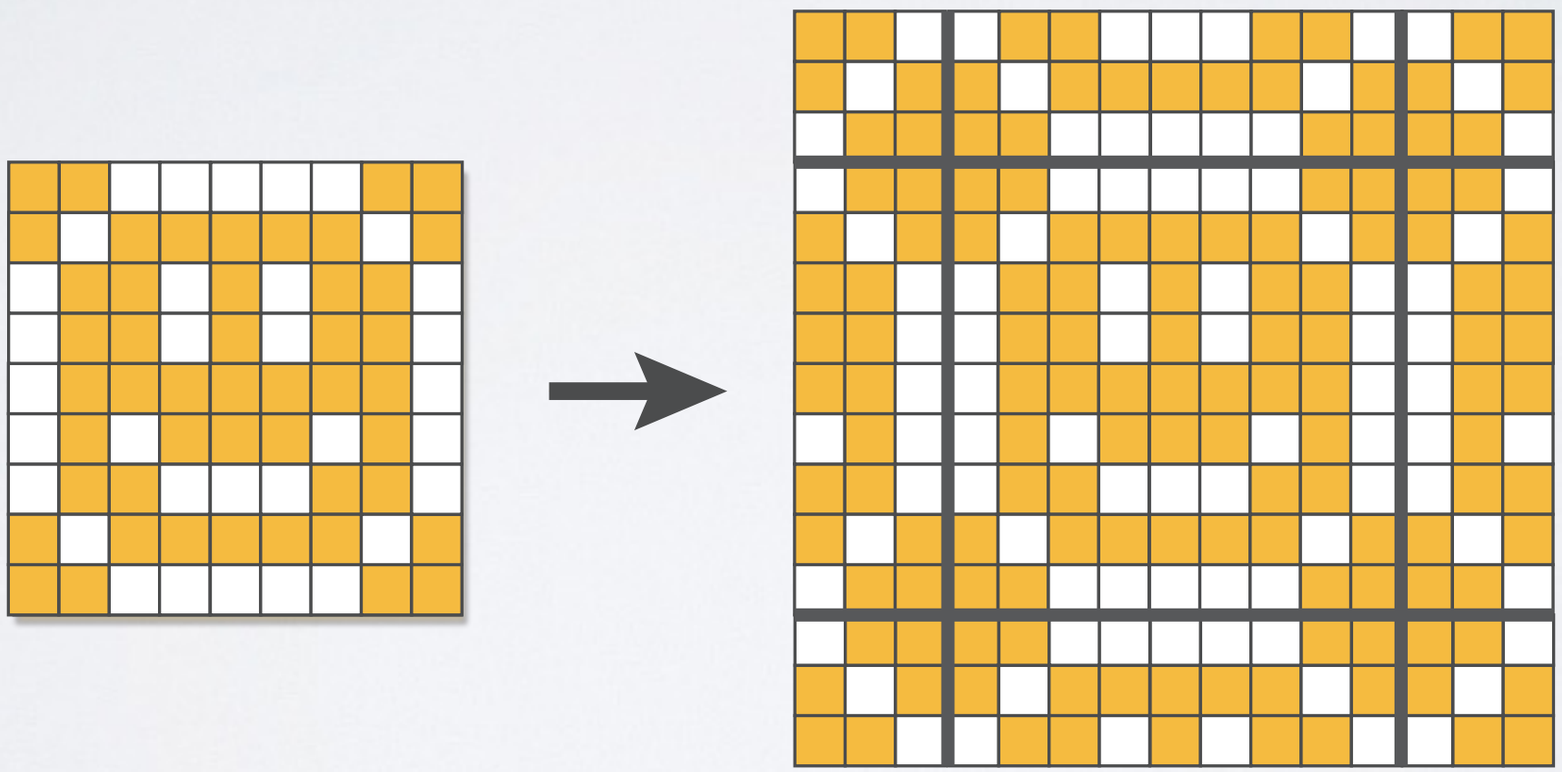
1. 畳み込み層



畳み込みニューラルネットCNN

1. 畳み込み層

パディング: 画像の縮小を防ぐ



畳み込みニューラルネットCNN

1. 畳み込み層

パディング: 画像の縮小を防ぐ

sameパディング: 入出力が同じサイズのパディング

fullパディング: 出力が最大になるようなパディング

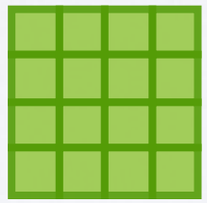
畳み込みニューラルネットCNN

1. 畳み込み層

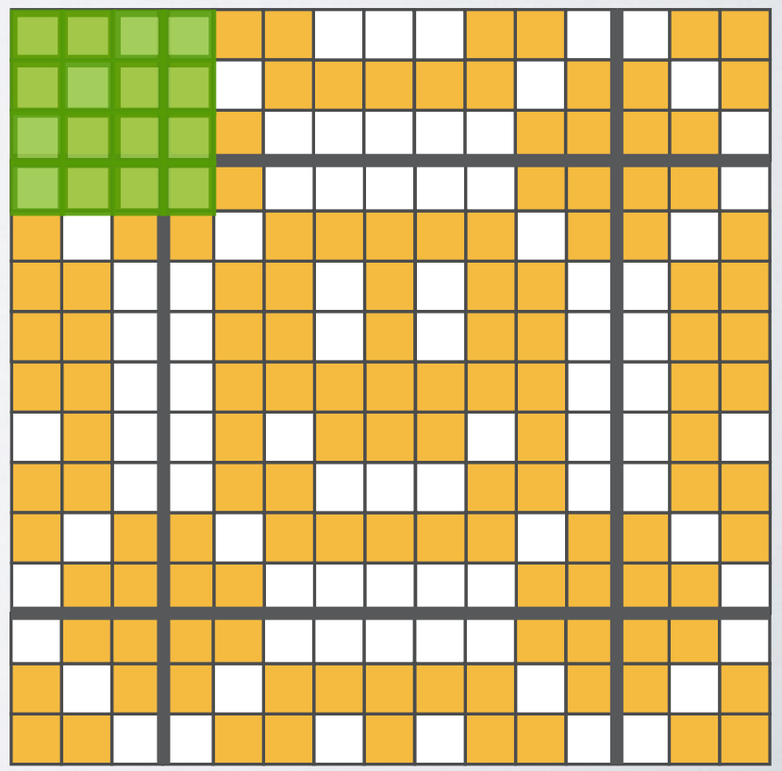
パディング: 画像の縮小を防ぐ

sameパディング: 入出力が同じサイズのパディング

fullパディング: 出力が最大になるようなパディング



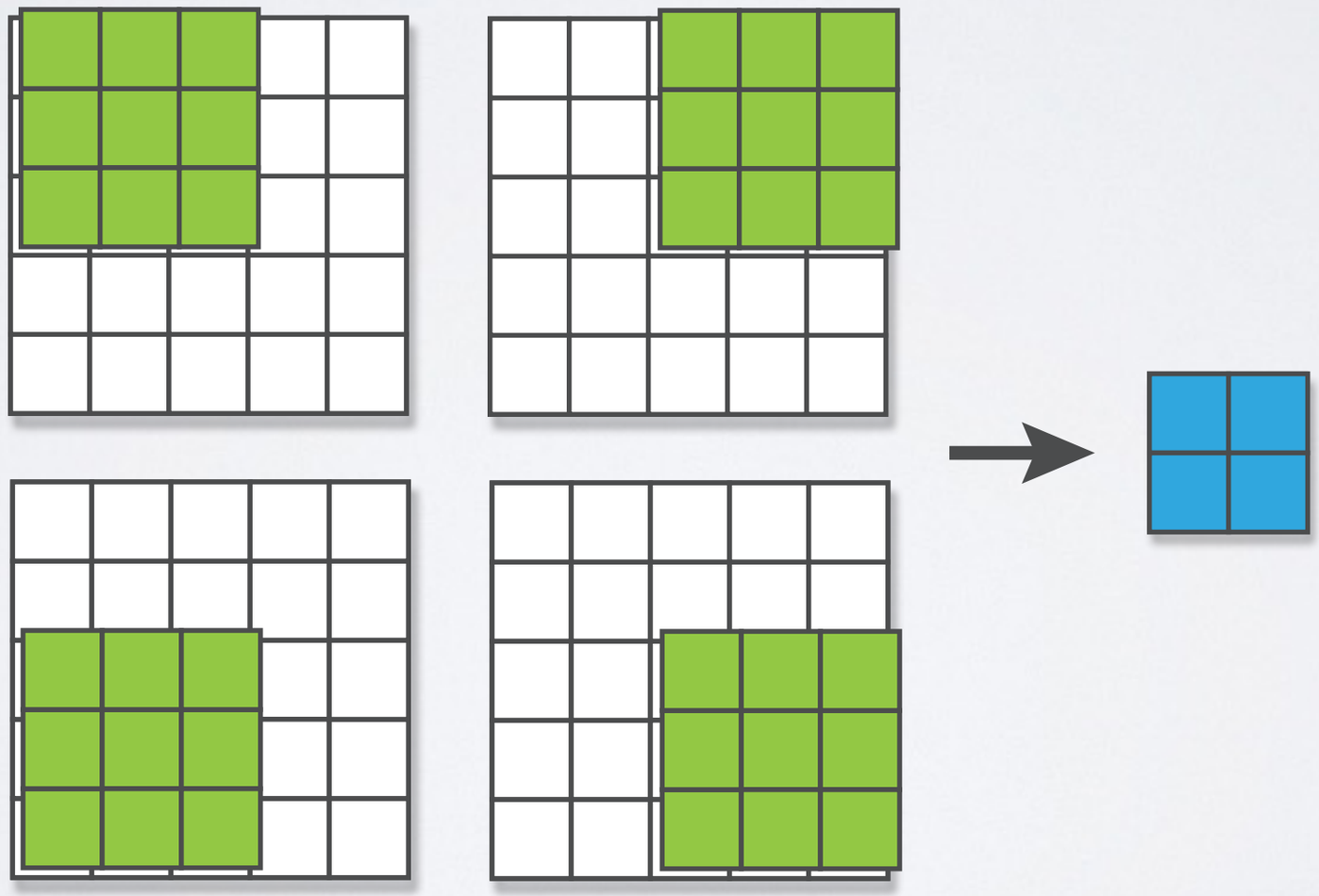
フィルタが4*4の時は、
端に厚み3だけパディング



畳み込みニューラルネットCNN

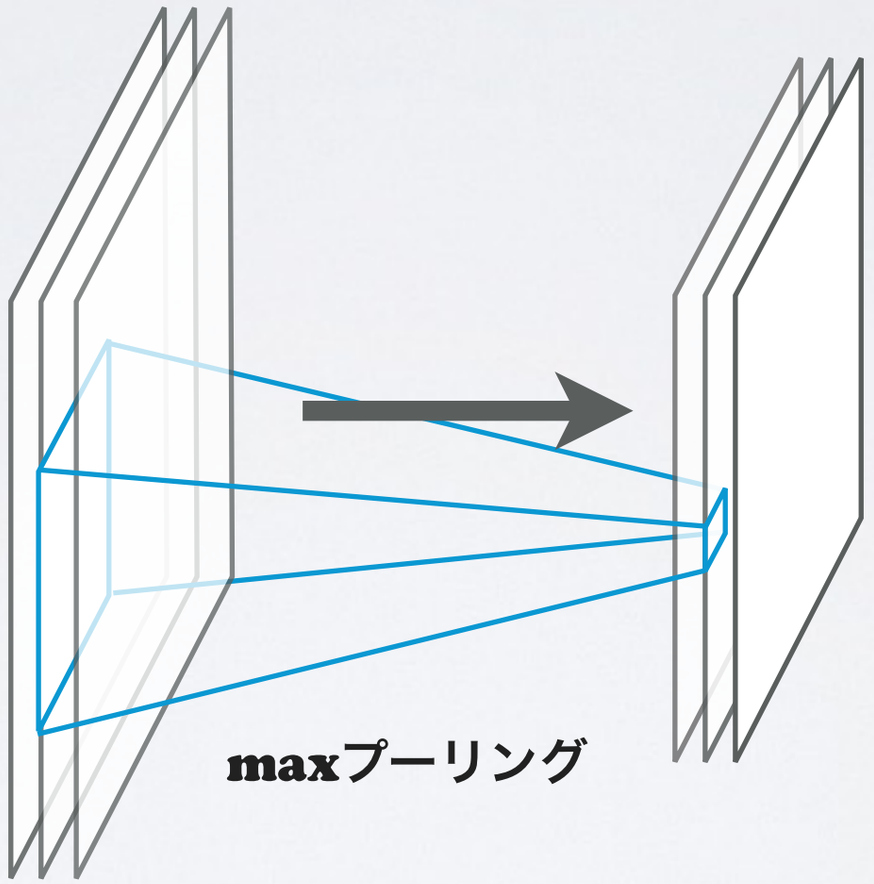
1. 畳み込み層

ストライド: 畳み込みを若干スキップ(下はストライド2)



畳み込みニューラルネットCNN

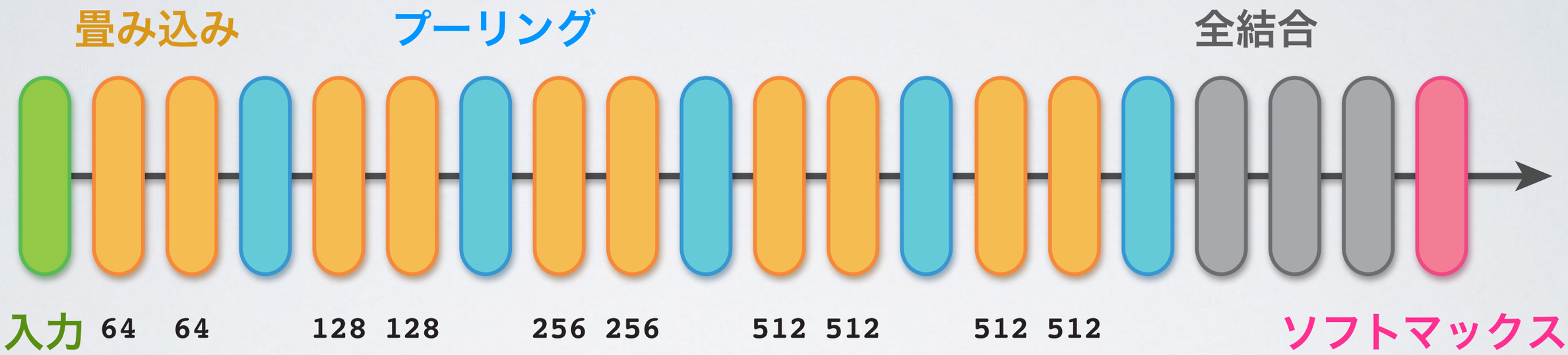
2. プーリング層



maxプーリング

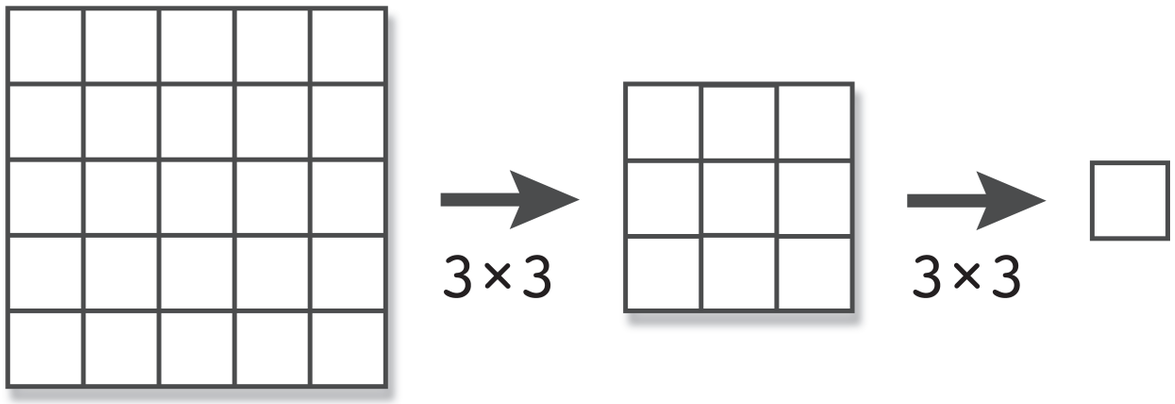
畳み込みニューラルネットCNN

3. モデル全体の例

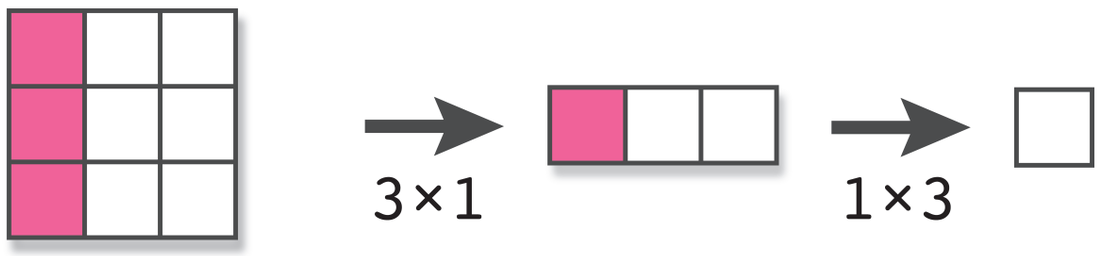


VGG

因子化した畳み込み層 factorized convolution layer



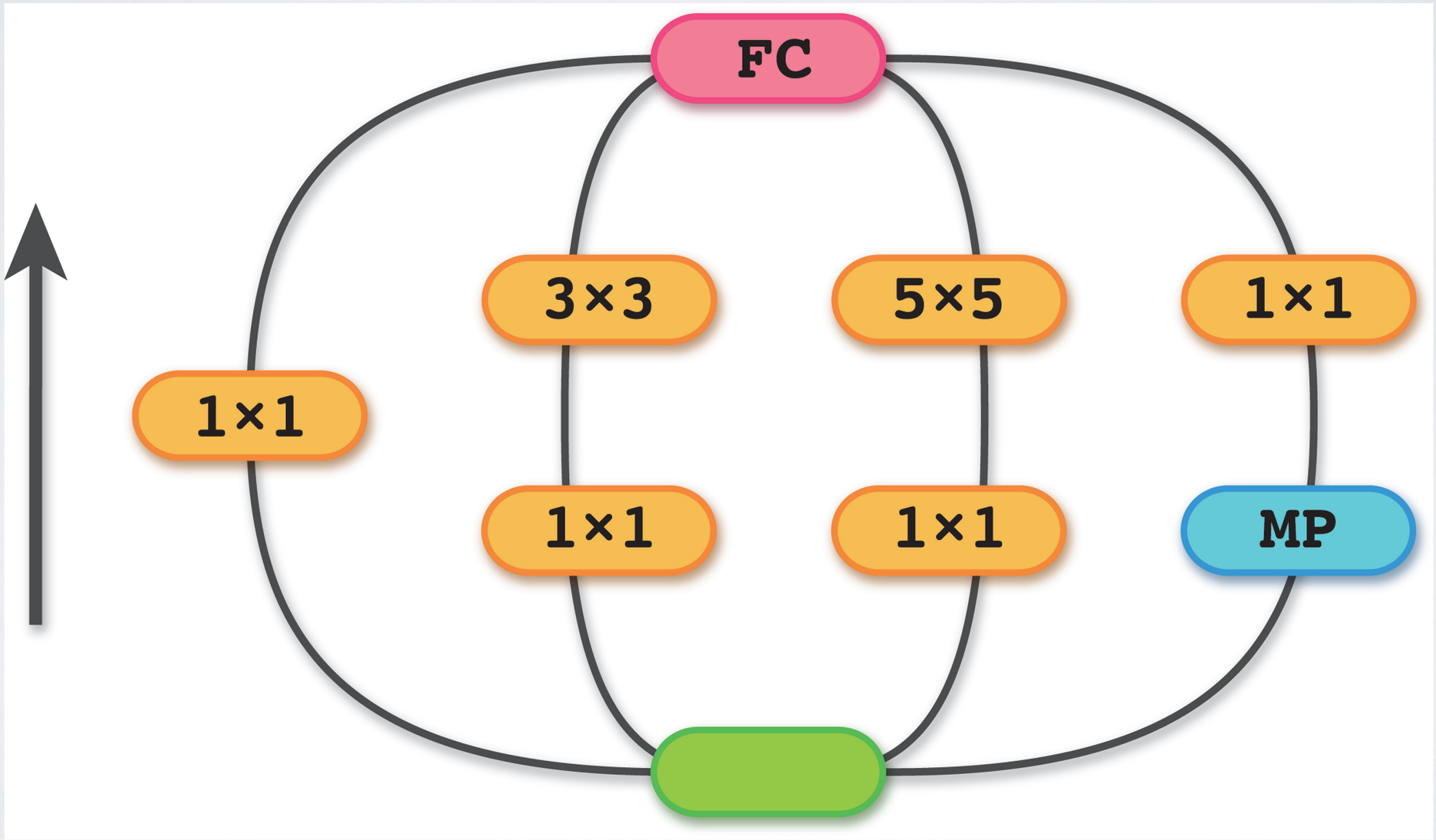
$$\Rightarrow 100 \times \frac{2 \times (3 \times 3)}{5 \times 5} = 72\%$$



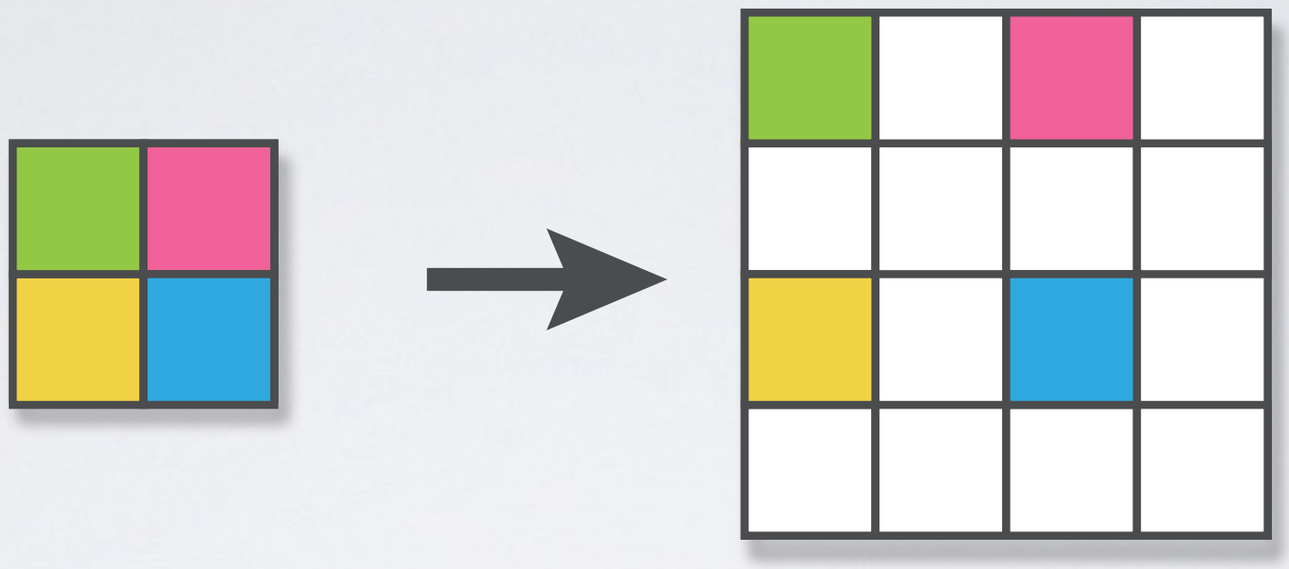
$$\Rightarrow 100 \times \frac{3 \times 1 + 1 \times 3}{3 \times 3} \approx 67\%$$

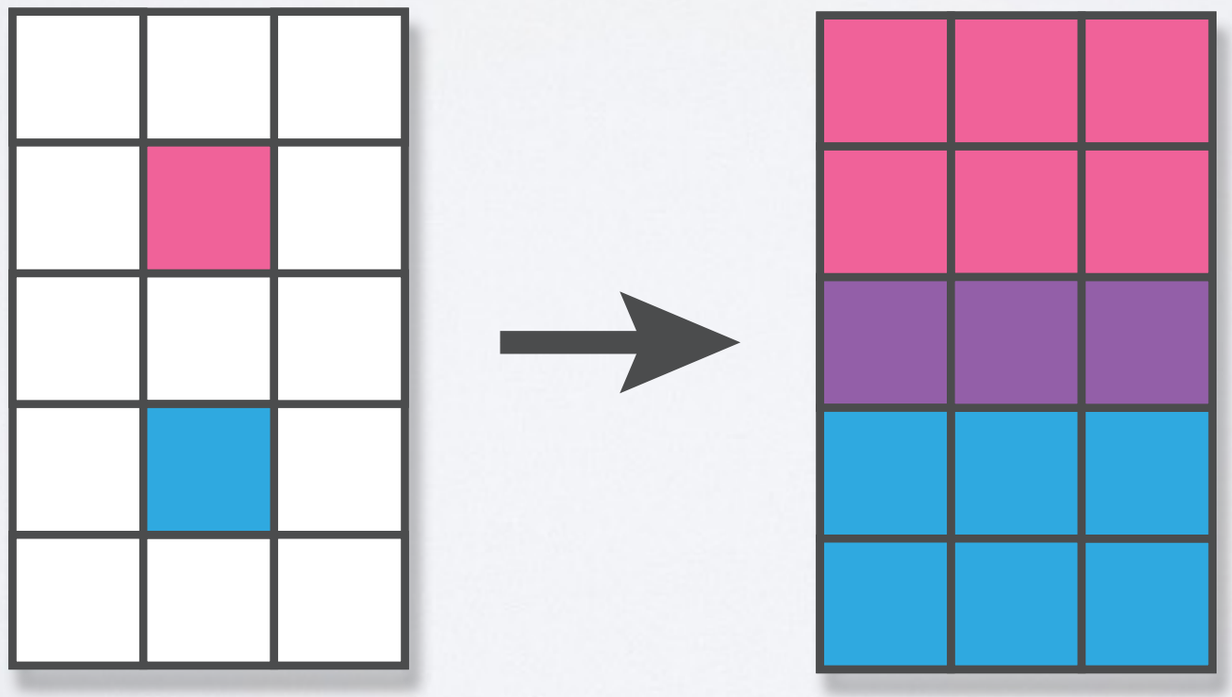
インセプション・モジュール

GoogLeNetで用いられた



脱畳み込みネットワーク deconvolution





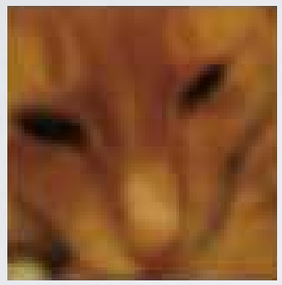
事前学習*

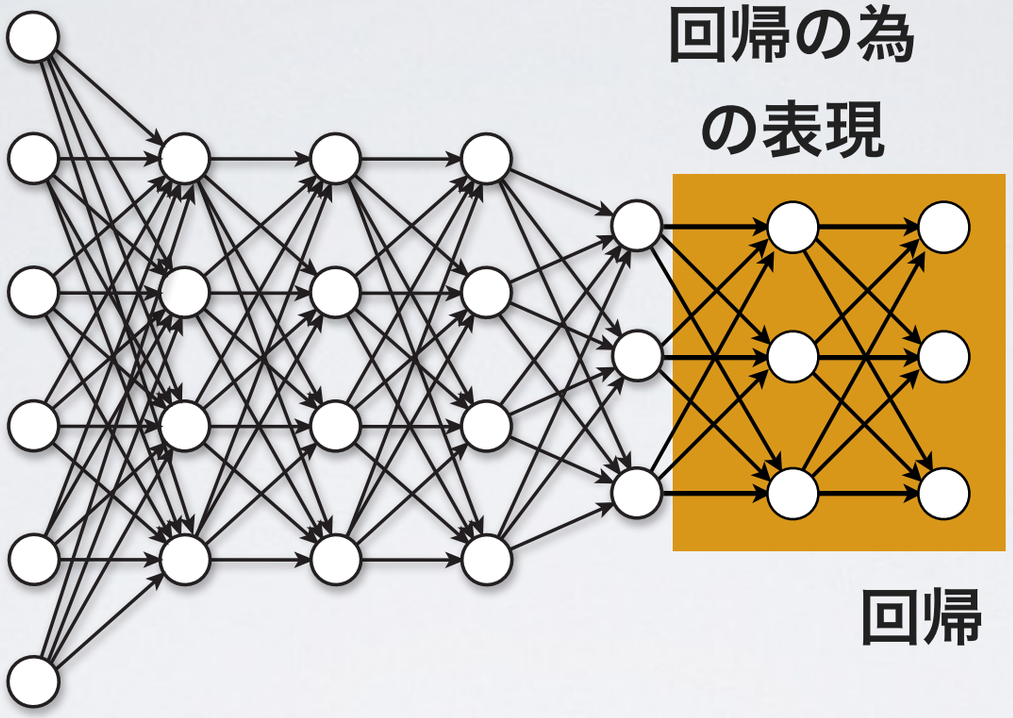
教師あり学習の前に、ネットワークの重みを良い値にしておく（良い初期値からスタートするのが重要）

教師無し学習による「層ごとの貪欲学習法」

後で紹介する自己符合化器を用いる

転移学習



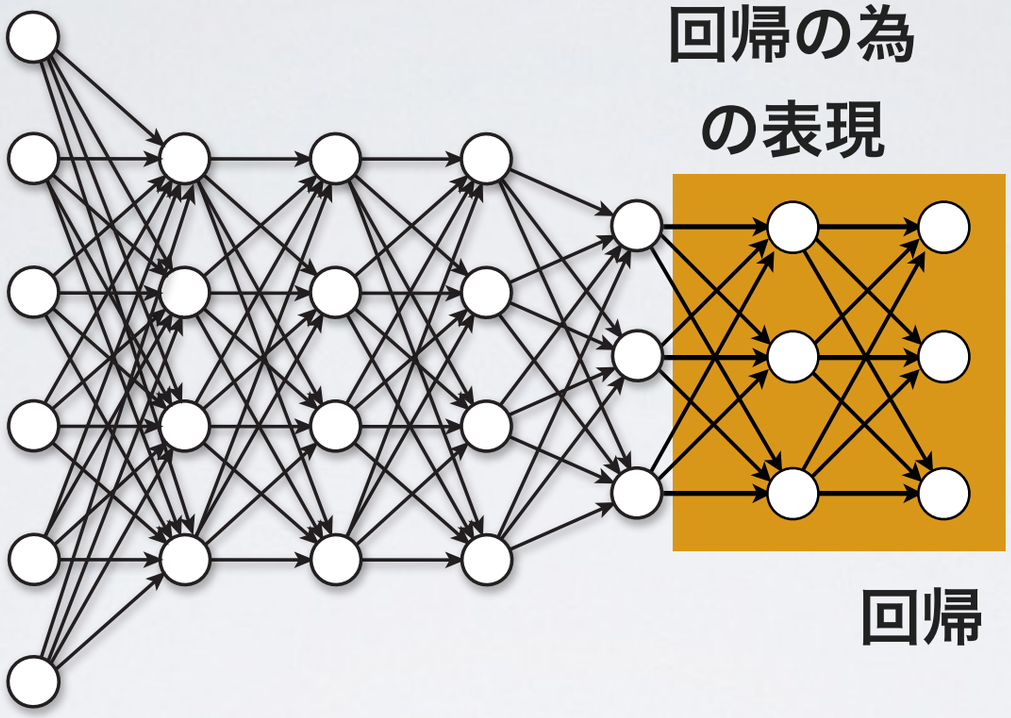


回帰の為
の表現

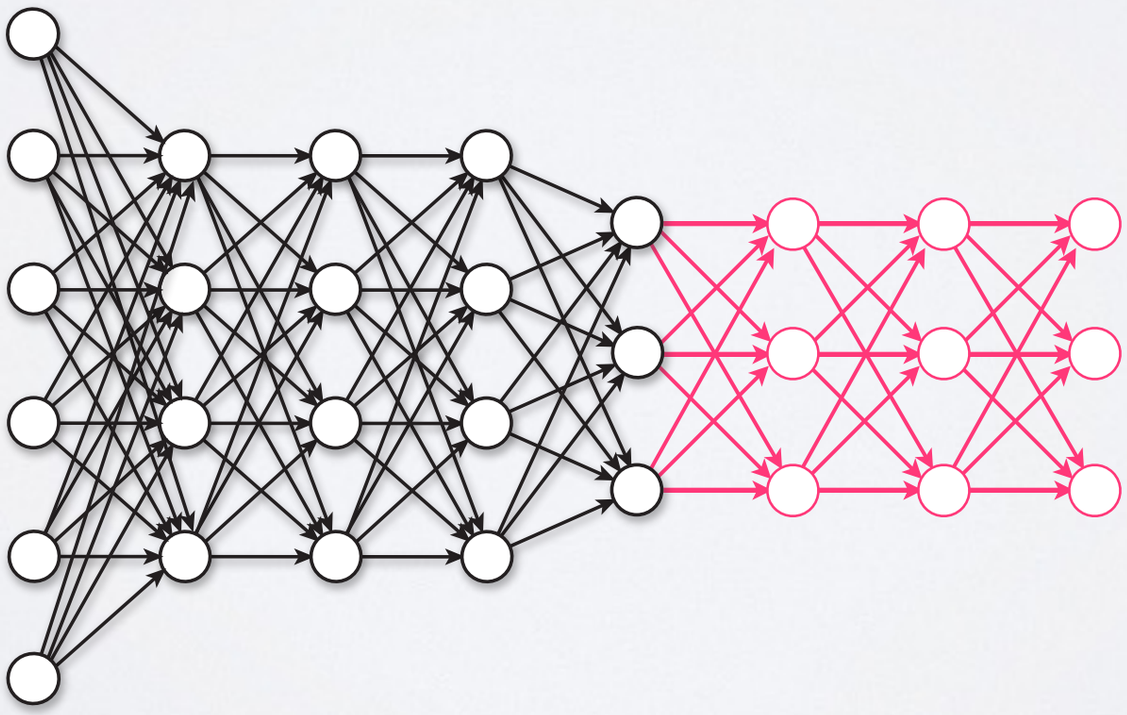
回帰

標的 $\mathbf{y}$=猫

転移学習

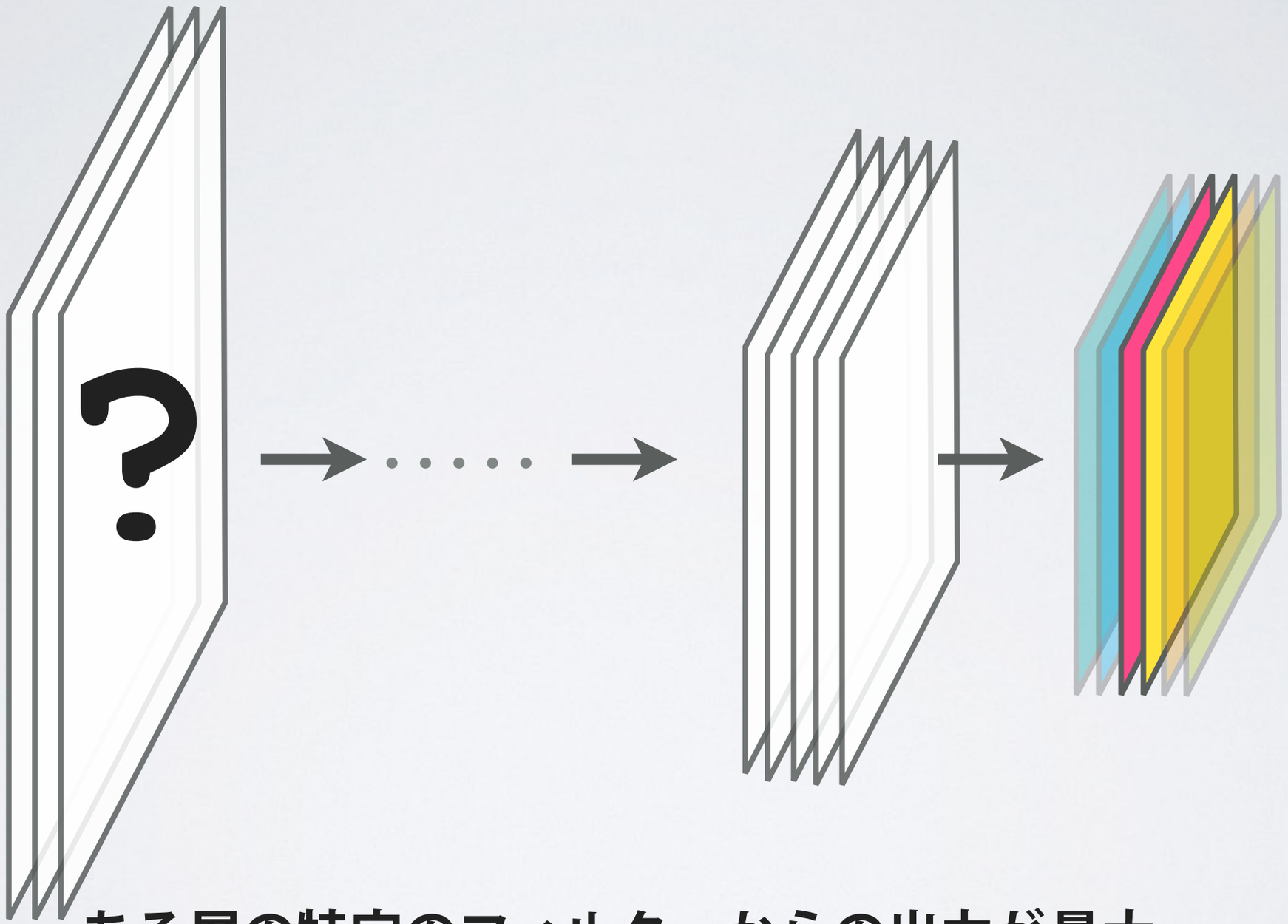


標的 y=猫



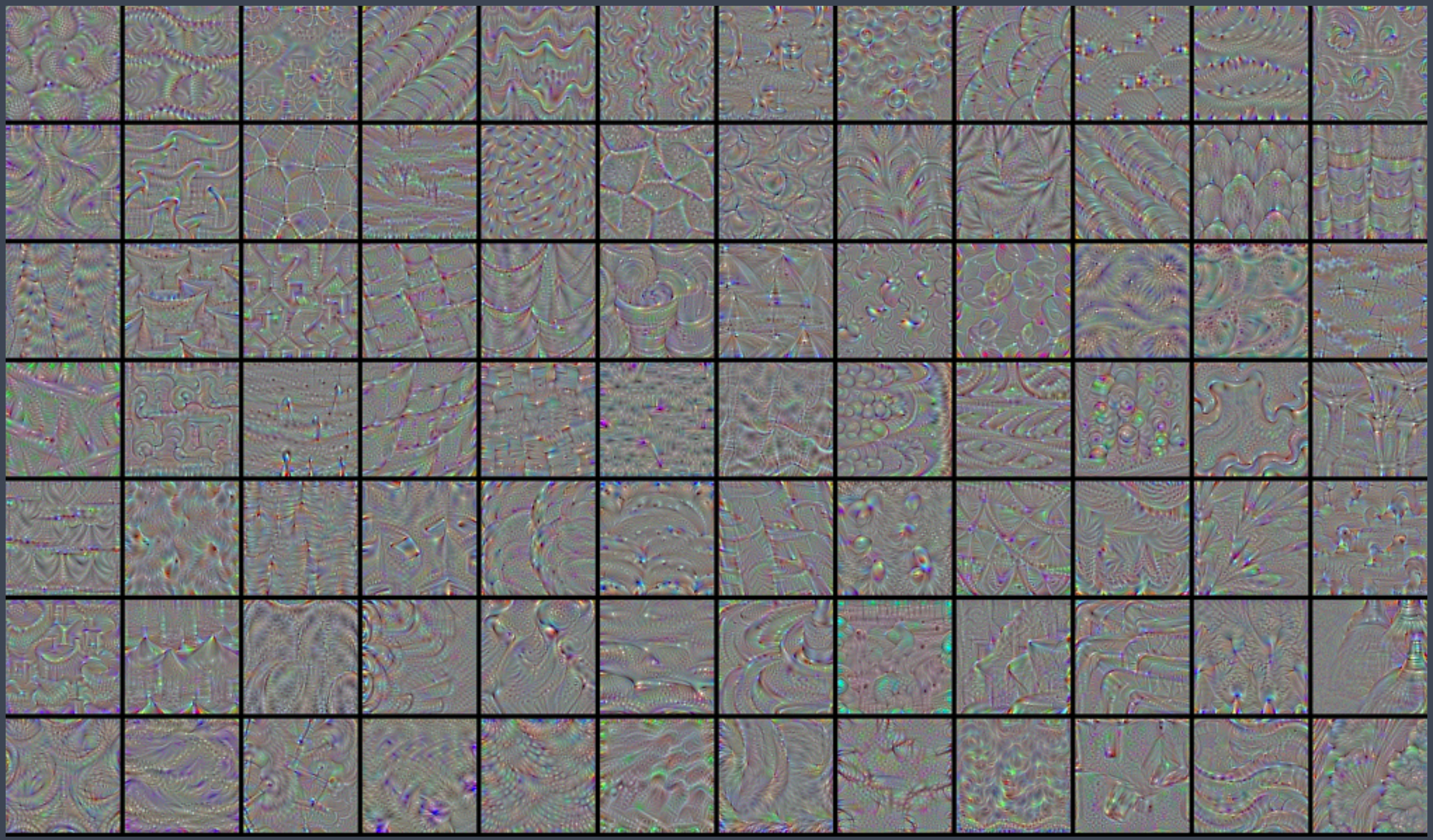
標的 y=哺乳類が1匹、魚類は1匹、…

深層学習が見る世界: いったいどんなパターンを抽出しているのか？

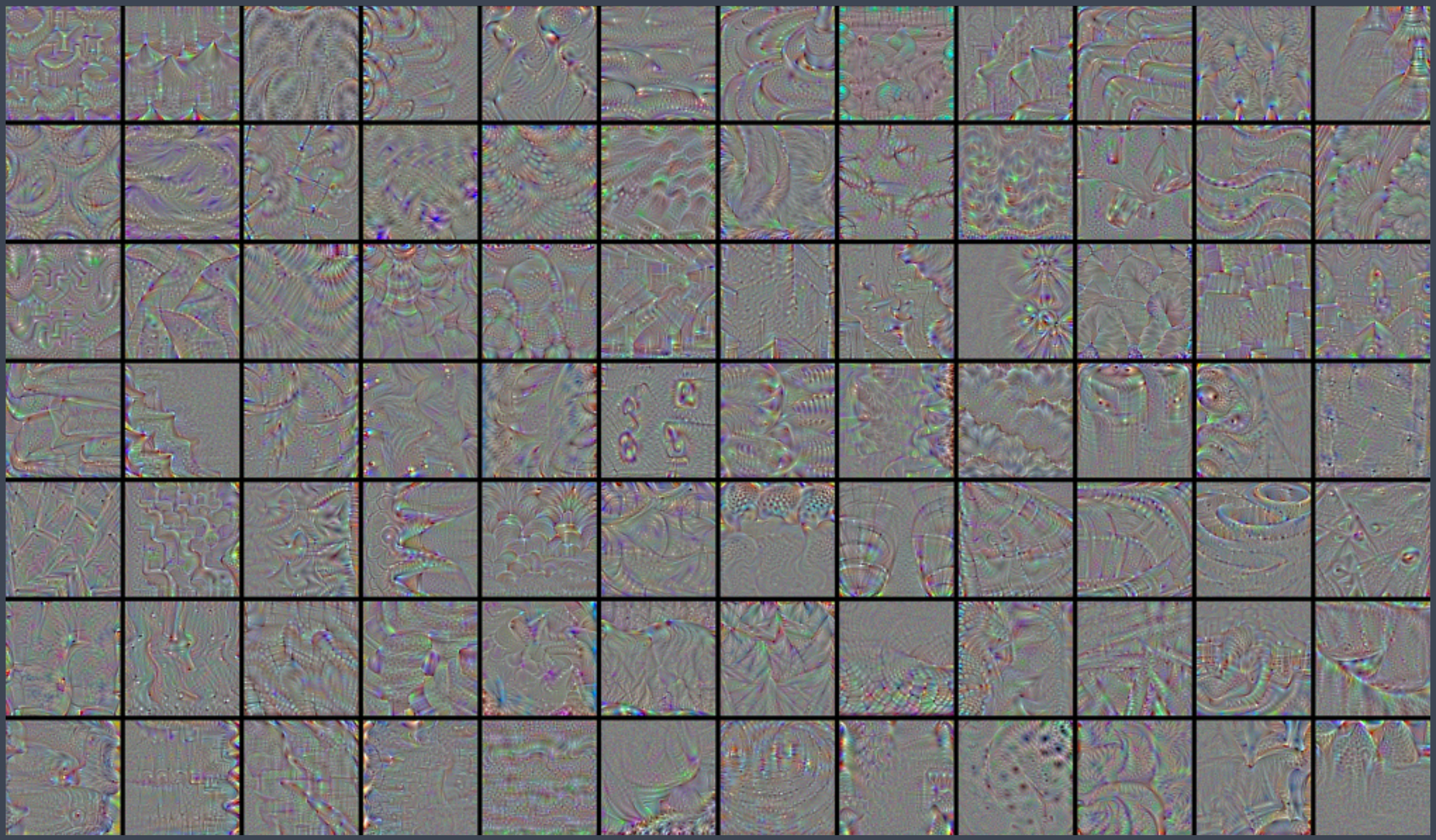


ある層の特定のフィルターからの出力が最大化されるような入力画像はどのようなものか？

深層学習が見る世界: いったいどんなパターンを抽出しているのか？

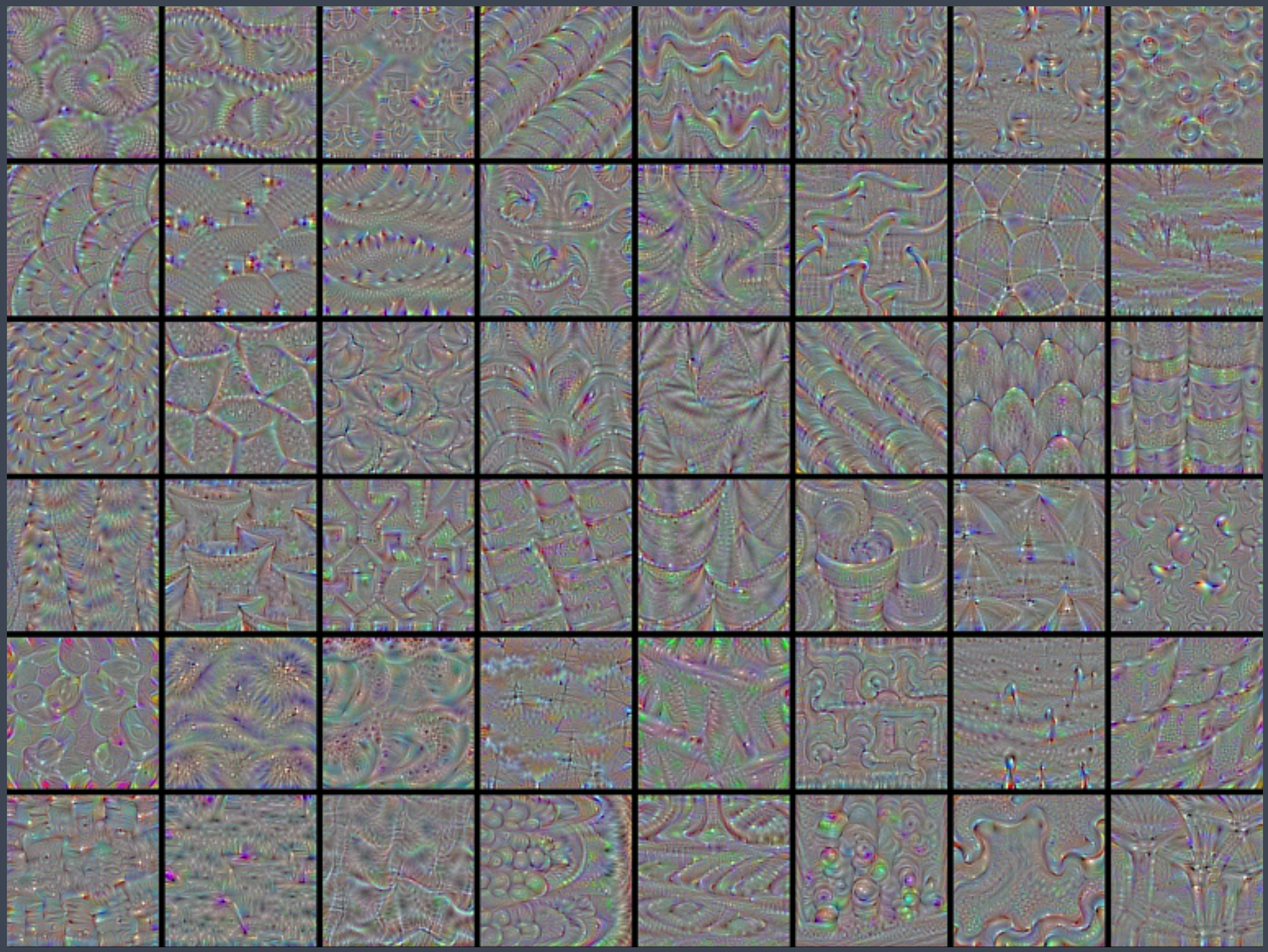


深層学習が見る世界: いったいどんなパターンを抽出しているのか？



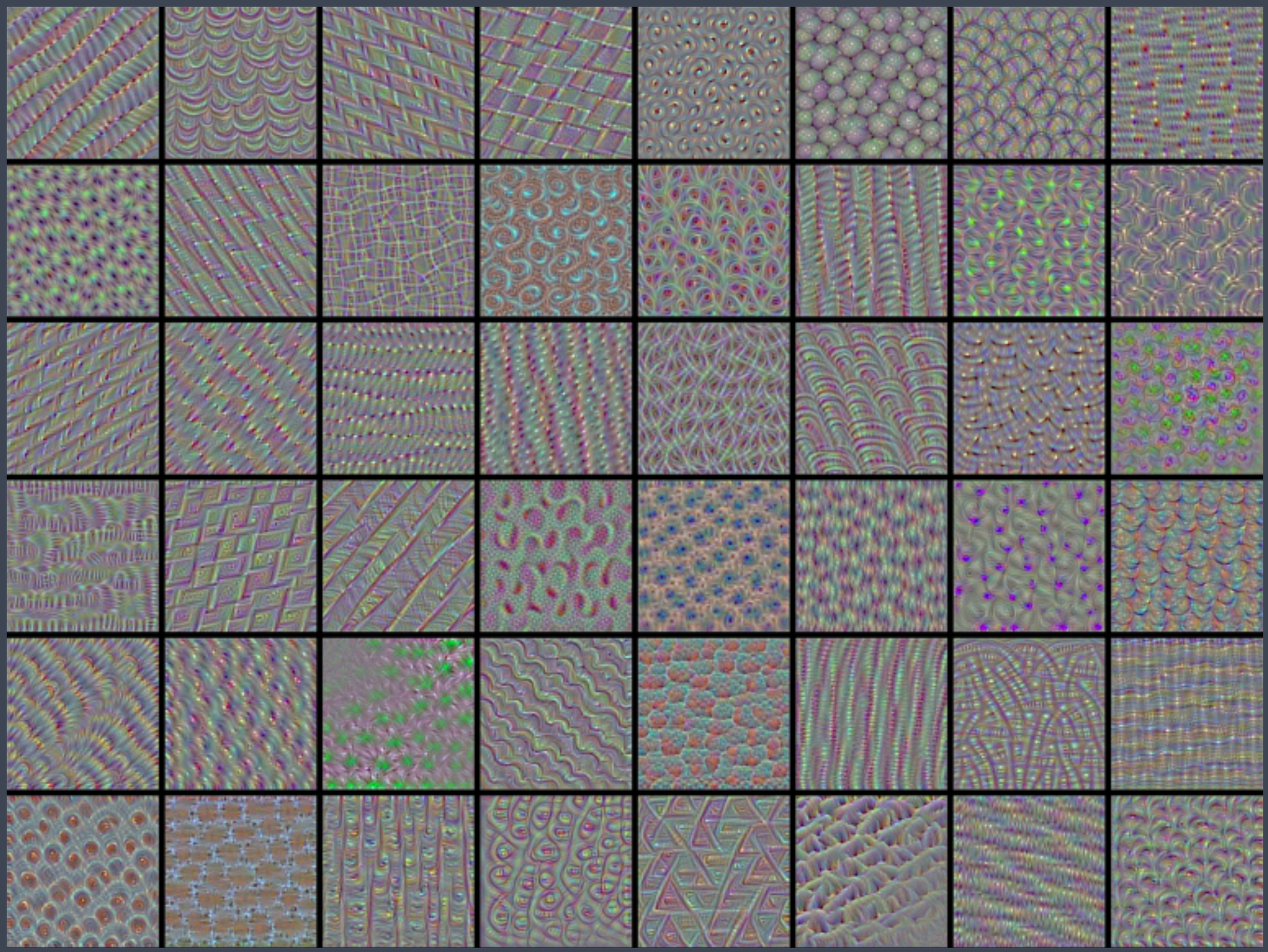
深層学習が見る世界: いったいどんなパターンを抽出しているのか？

Conv5



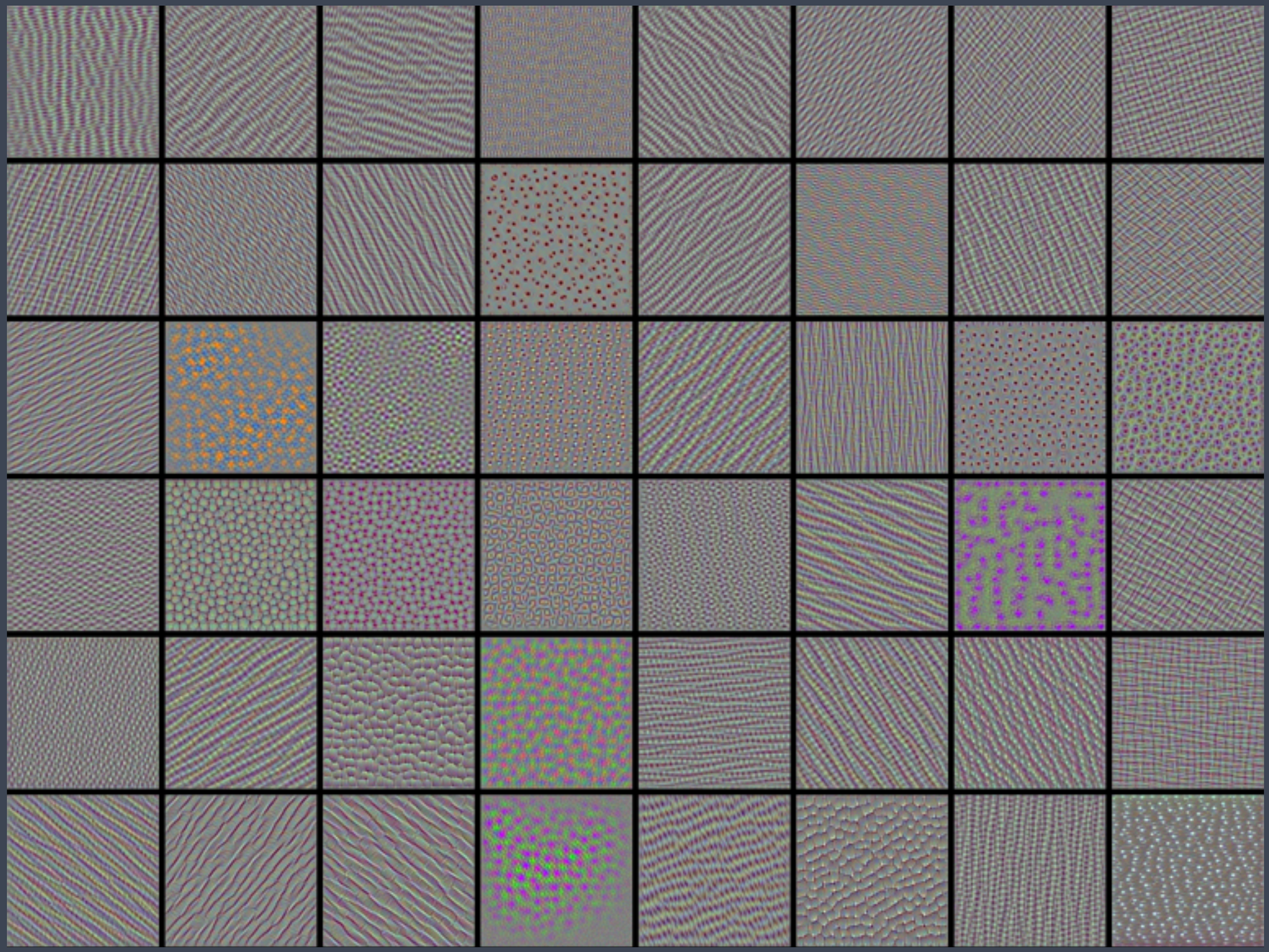
深層学習が見る世界: いったいどんなパターンを抽出しているのか？

Conv4



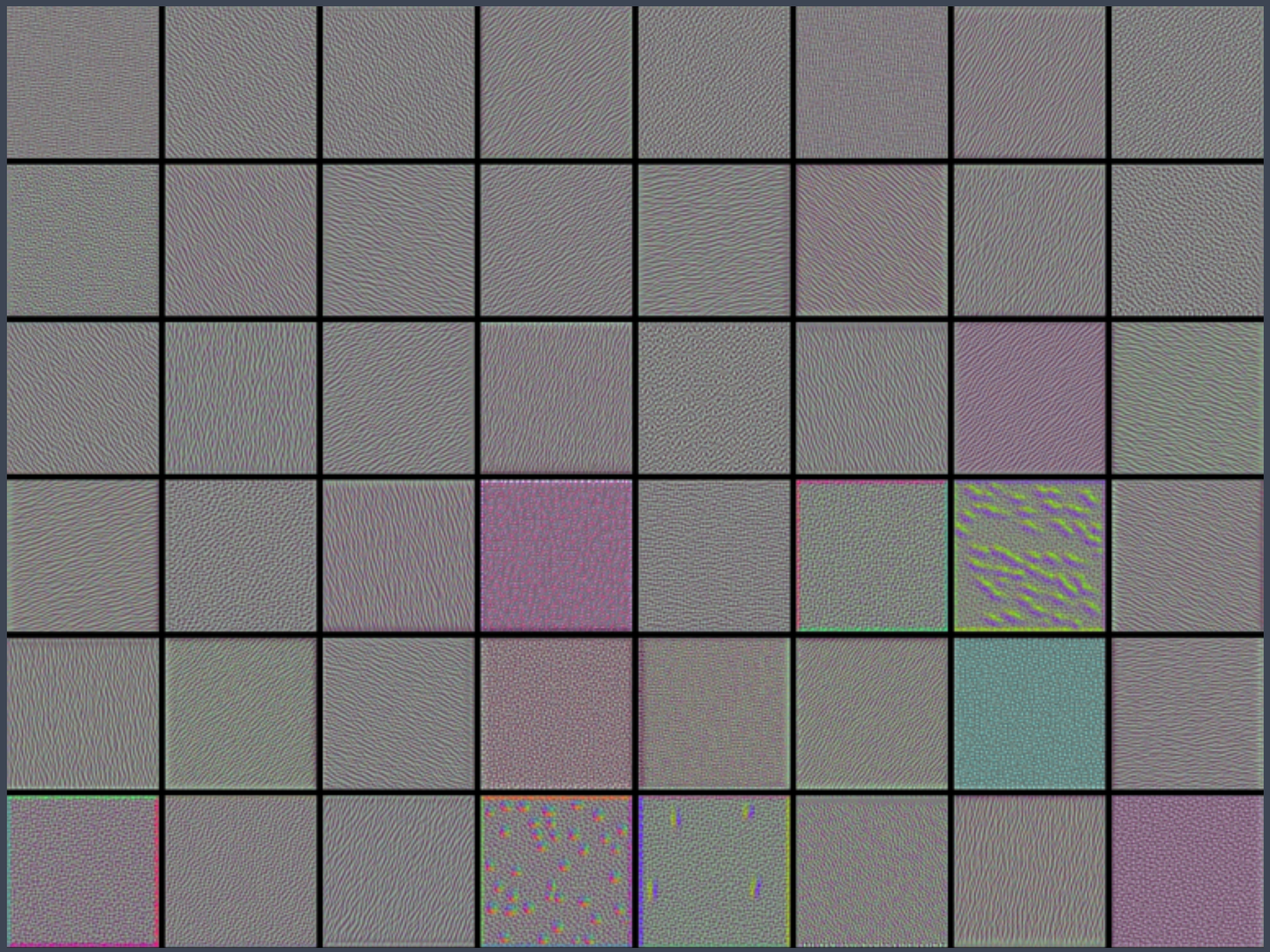
深層学習が見る世界: いったいどんなパターンを抽出しているのか？

Conv3



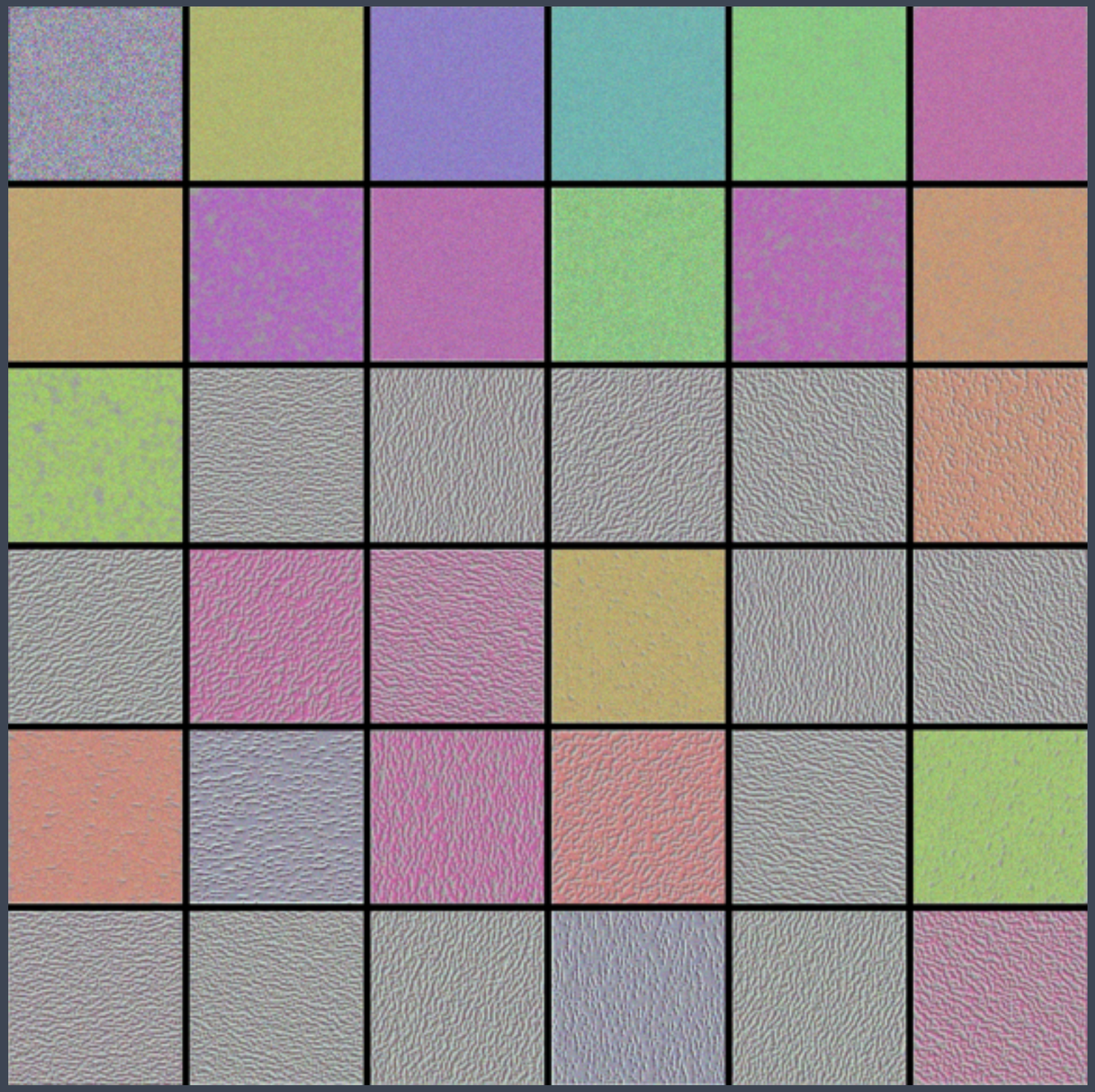
深層学習が見る世界: いったいどんなパターンを抽出しているのか？

Conv2



深層学習が見る世界: いったいどんなパターンを抽出しているのか？

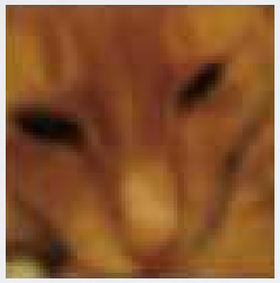
Conv1



6. 深層学習はなぜうまくいく？

表現学習と深層表現

回帰による画像認識



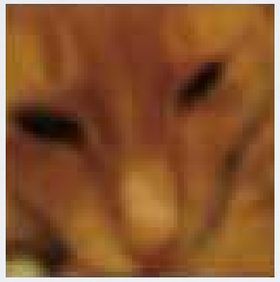
特徴量ベクトル
（表現）
handcraft（SHIFT ets）

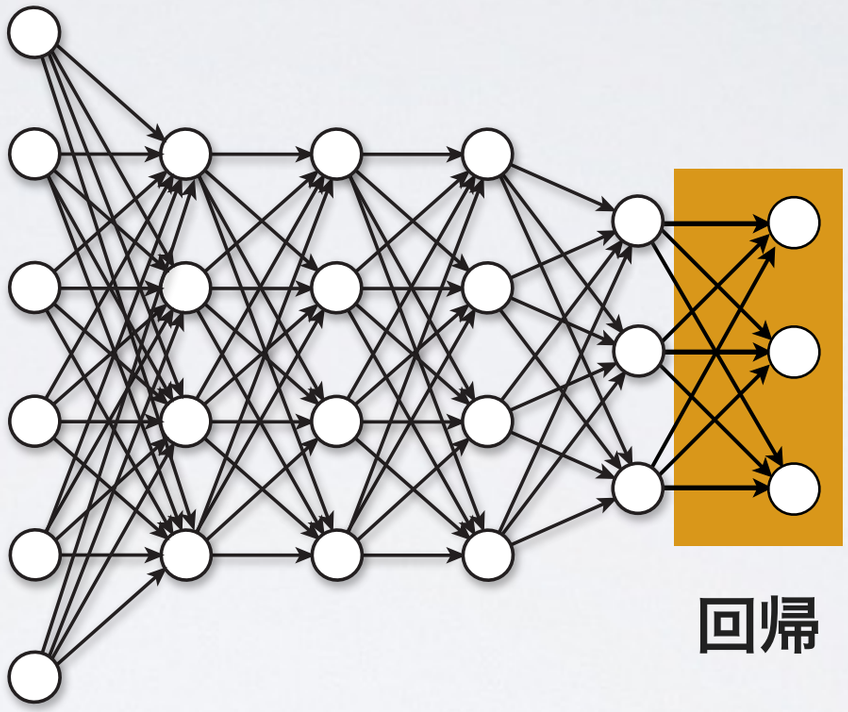
回帰

高等なタスクに有用な特徴量を設計するのは困難

表現学習と深層表現

深層表現学習でのスキーム





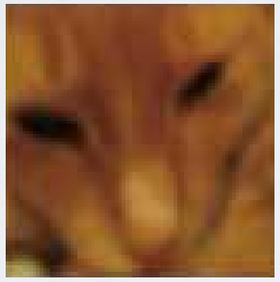
標的 $\mathbf{y}$=猫

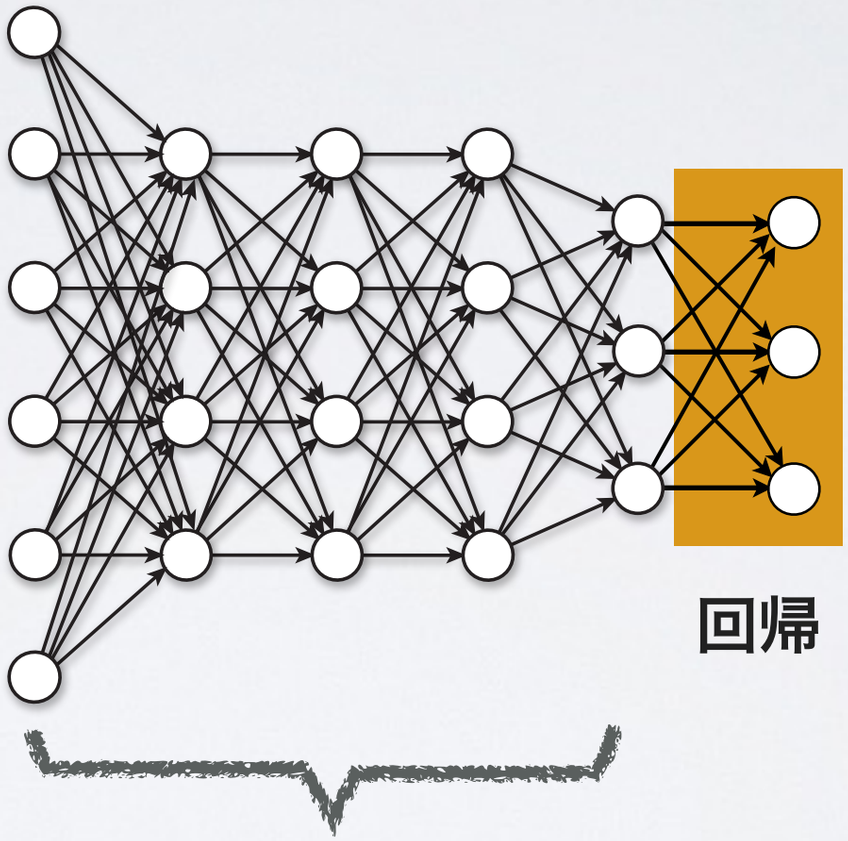
回帰

回帰の為
の表現

表現学習と深層表現

深層表現学習でのスキーム





低次の表現から高次の概念までが階層的に獲得される

特徴量の再利用が可能で、多くのタスクに対し汎用的

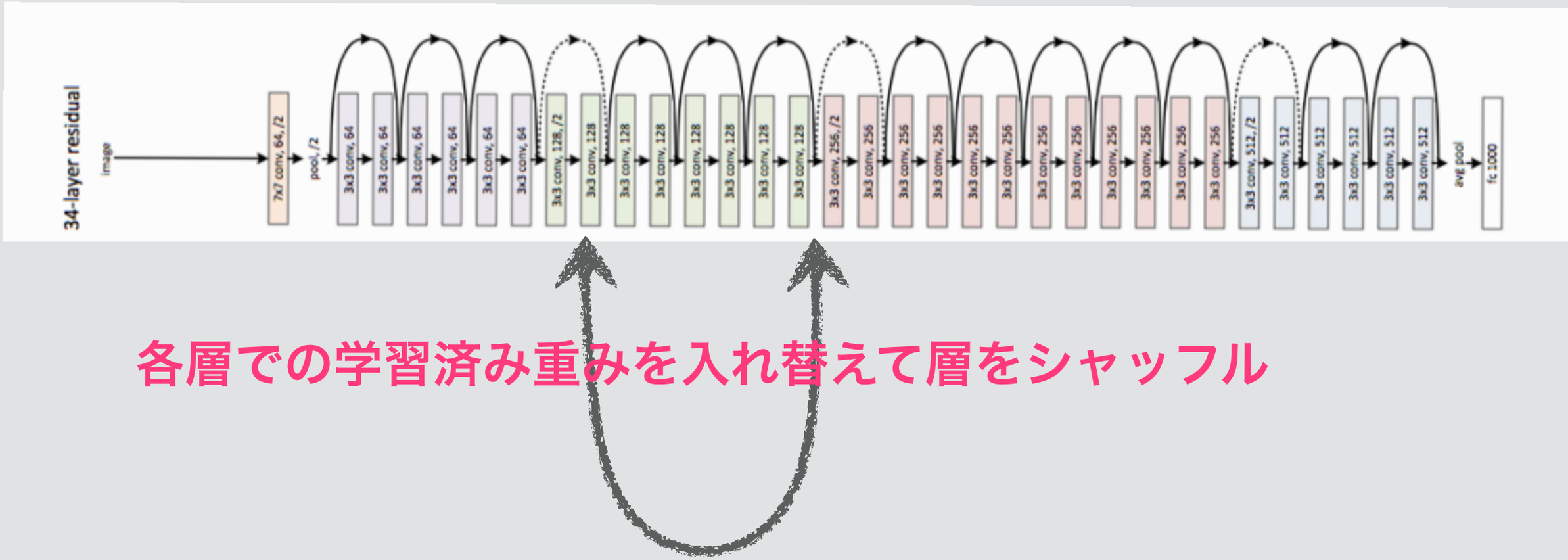
以上が良く引き合いに出される説明
(folklore)

そこらへんの本に書いてあるこれら見てきたような説明の一部は正しくない

数多くの非自明な現象が発見されつつある

ResNetにおける非階層的表現

学習済みResidual Net

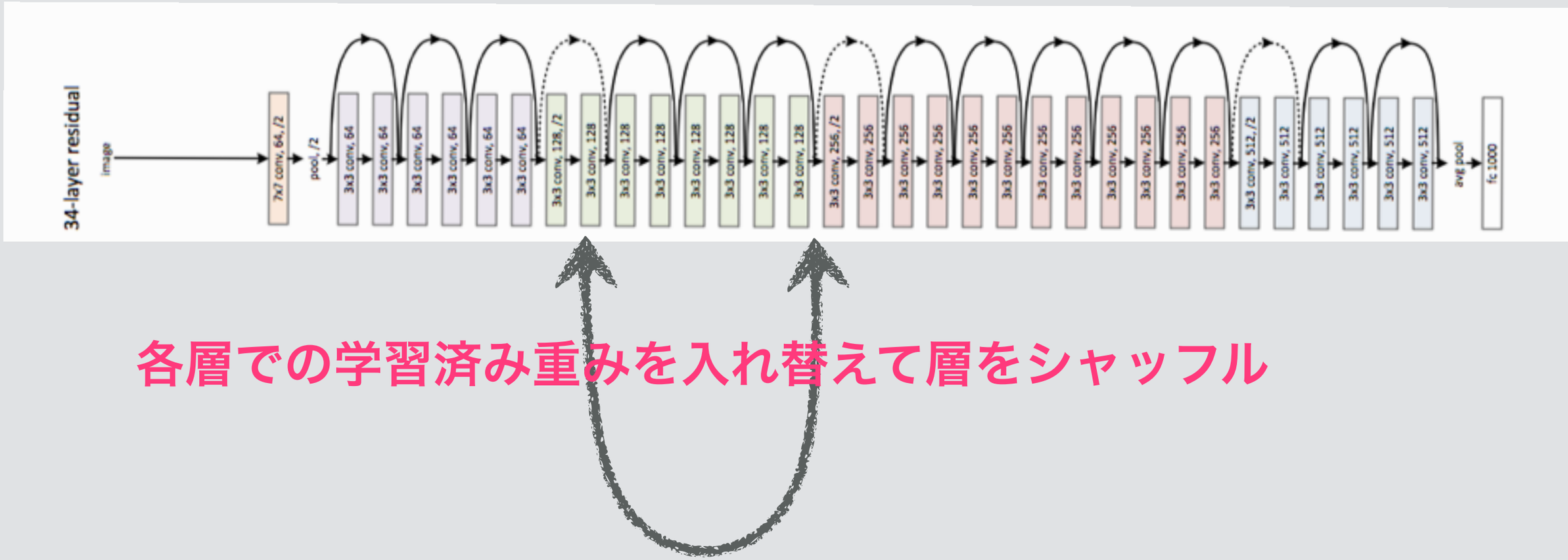


各層での学習済み重みを入れ替えて層をシャッフル

→ ほとんどエラー率は悪化せず。

ResNetにおける非階層的表現

学習済みResidual Net



各層での学習済み重みを入れ替えて層をシャッフル

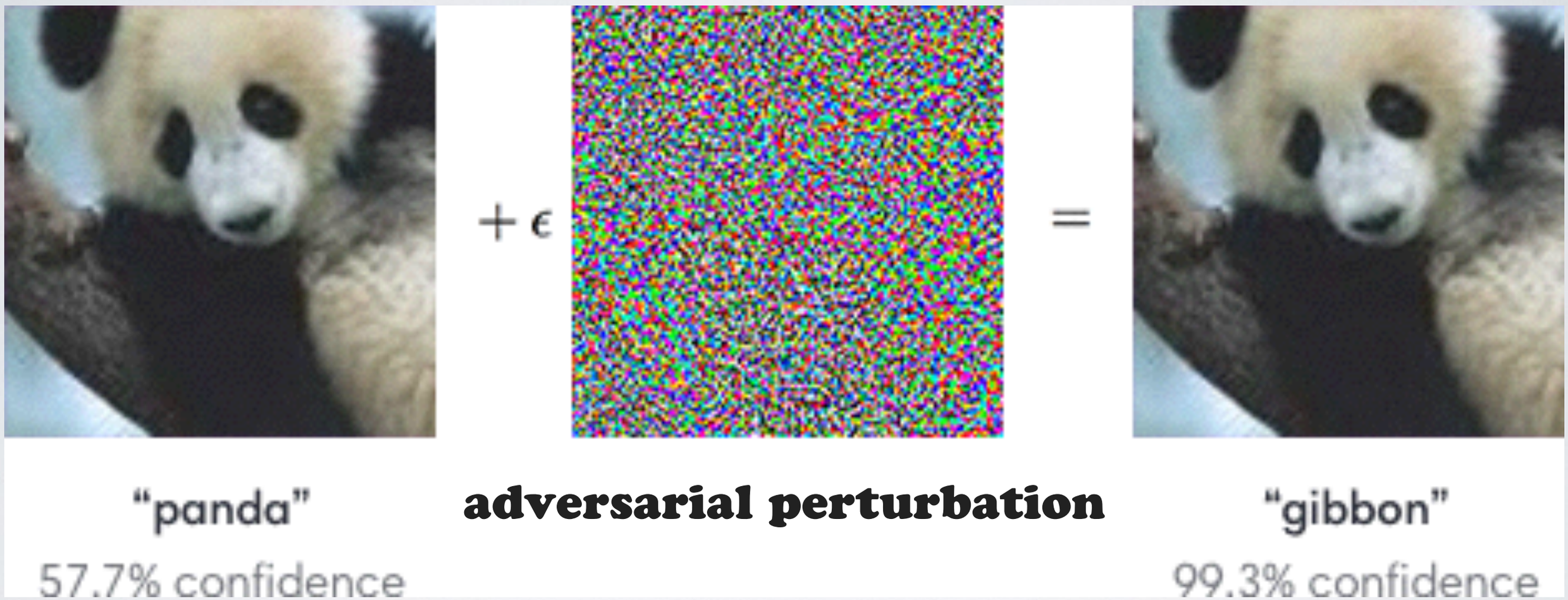
→ ほとんどエラー率は悪化せず。

どの層も、最終層で一気に実現される高次表現を単調に徐々に獲得

Adversarial Examples

左右の画像はなんでしょう？

Adversarial Examples

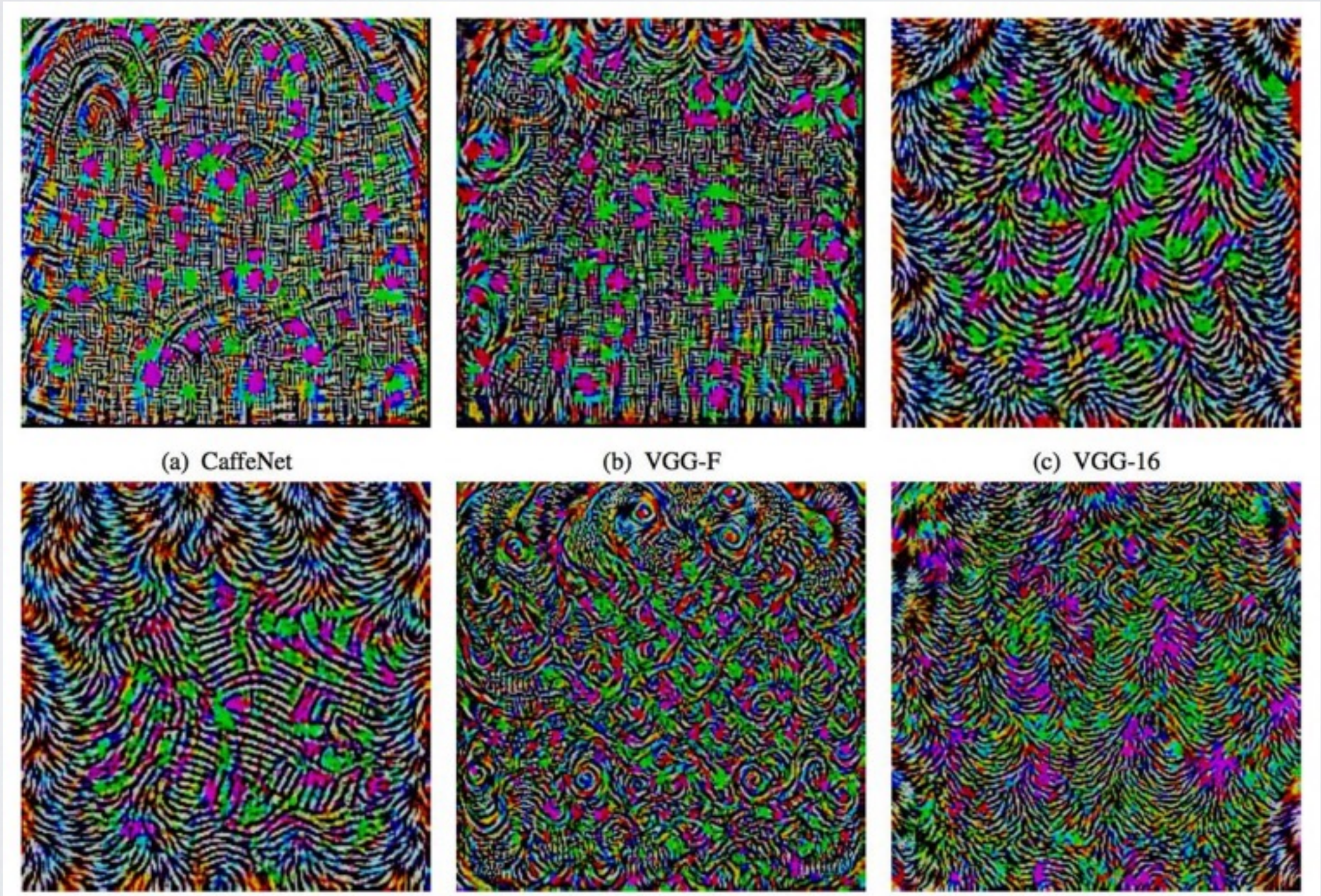


gibbon : テナガザル

これが学習済みモデル(高性能)の識別結果

Universal adversarial perturbations

敵対的エグザンプルは各モデルを超えて普遍的な現象



(a) CaffeNet (b) VGG-F (c) VGG-16

(d) VGG-19 (e) GoogLeNet (f) ResNet-152

学習済みモデルとノイズ

state-of-the-artレベルのモデルなどでも、
特別なノイズに関して、変な反応を示す。

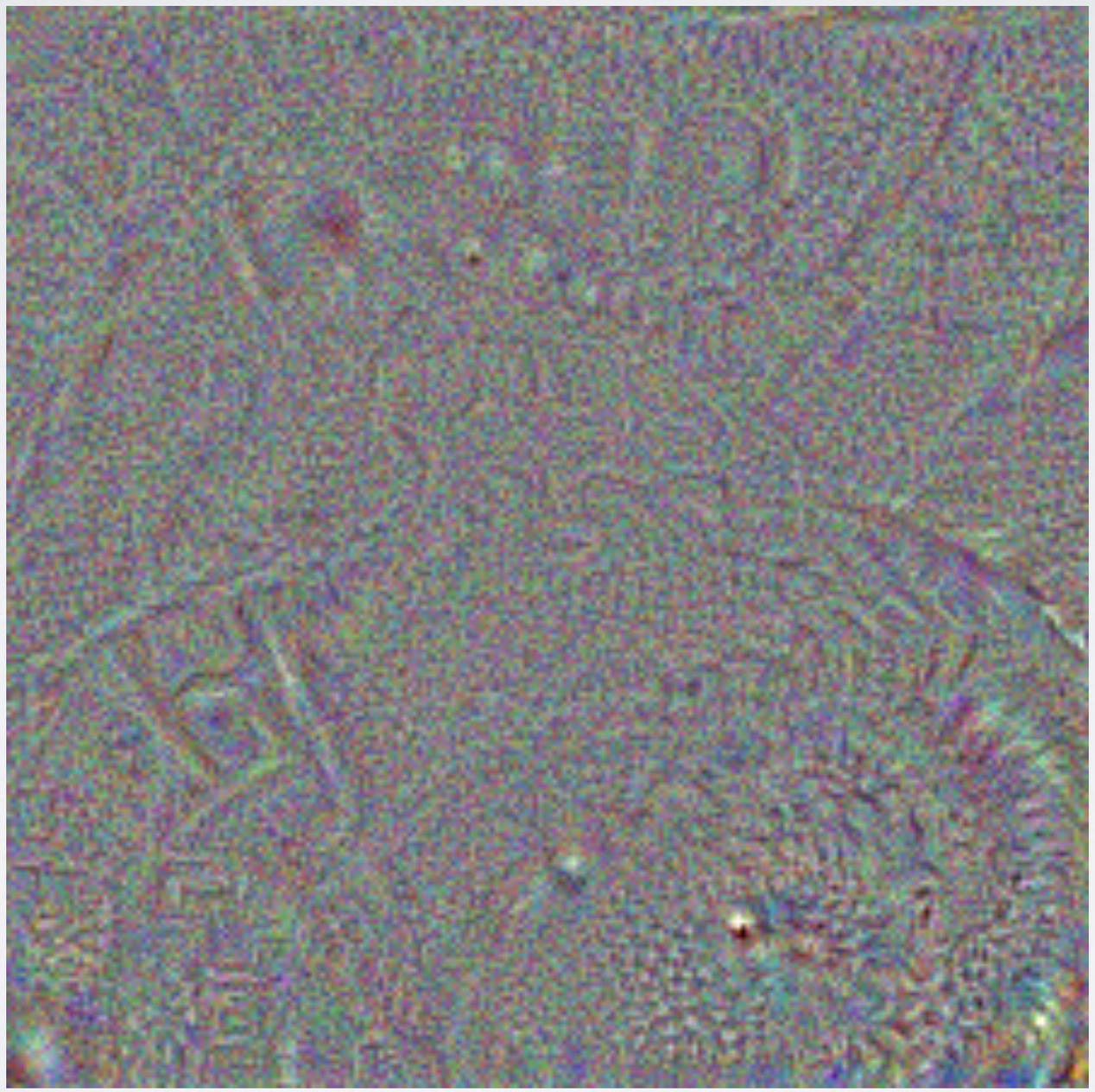
学習済みVGG16モデルで作ってみた

そろそろコーヒーが欲しくなる時間。

そこで深層学習を使って（ノイズを挽いて）
エスプレッソを「淹れて」みた

学習済みVGG16モデルで作ってみた

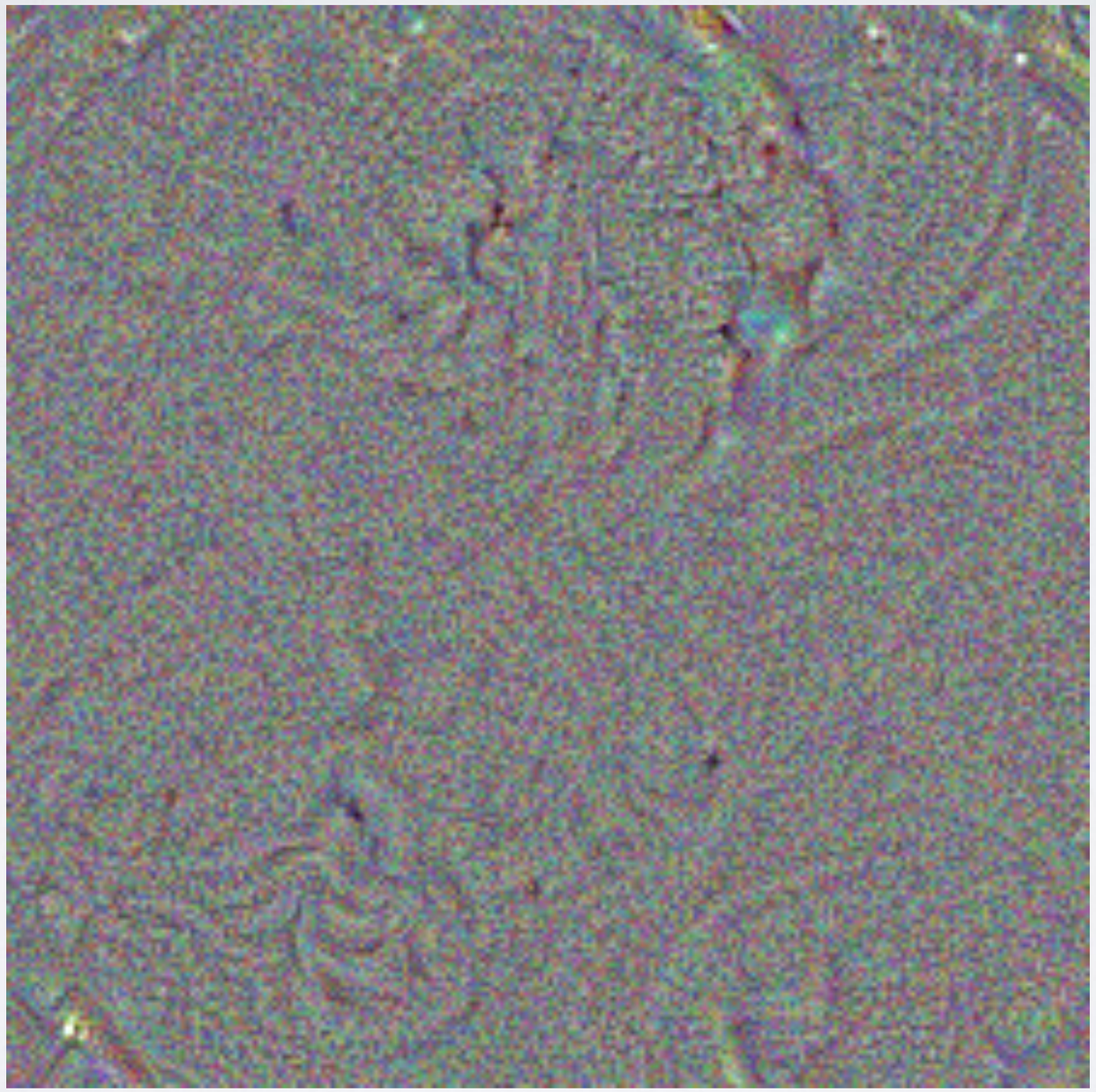
99.99%で識別される



class 967: espresso

学習済みVGG16モデルで作ってみた

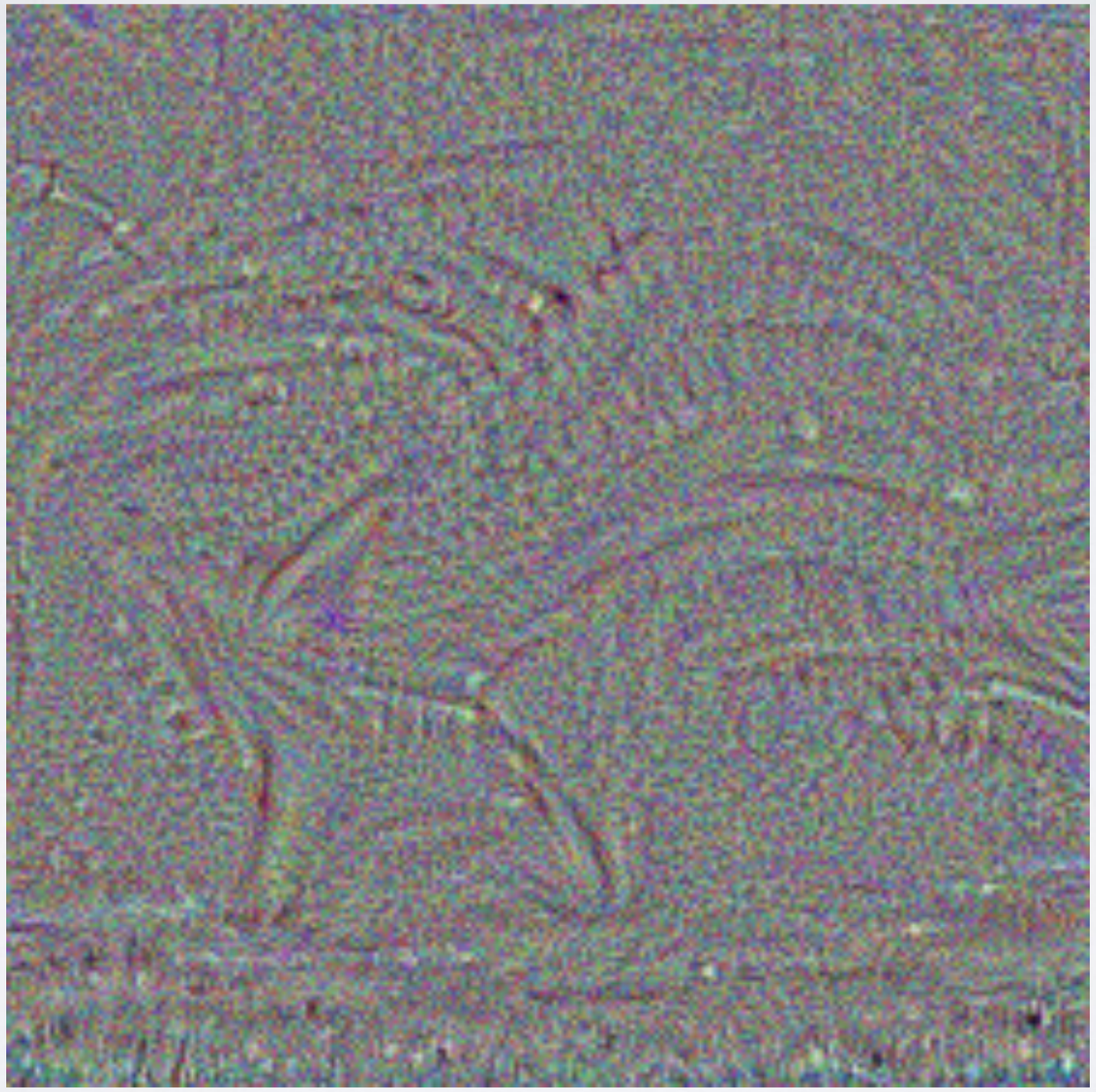
99.99%で識別される



class 928: ice-cream

学習済みVGG16モデルで作ってみた

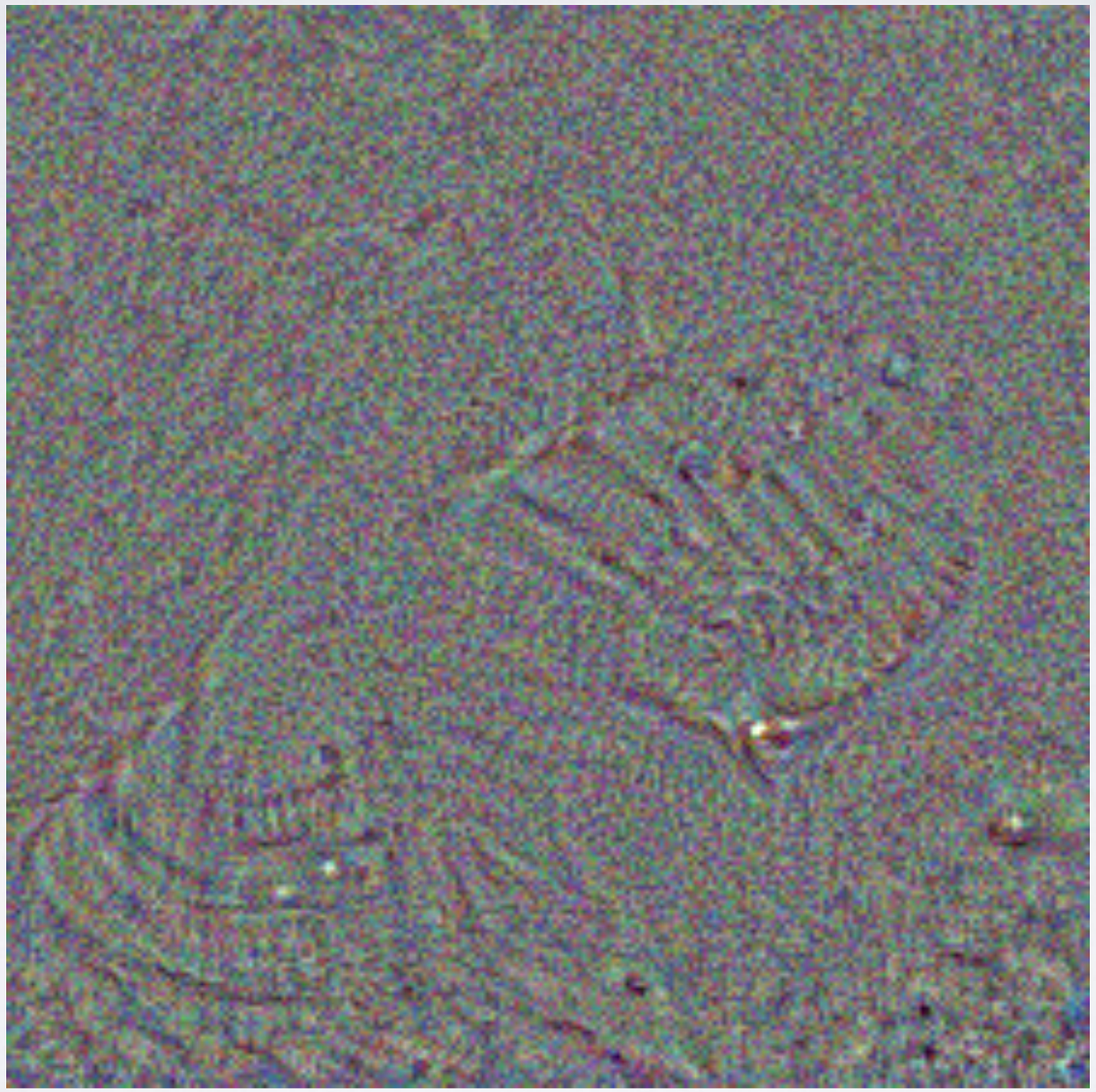
99.99%で識別される



class 340: zebra

学習済みVGG16モデルで作ってみた

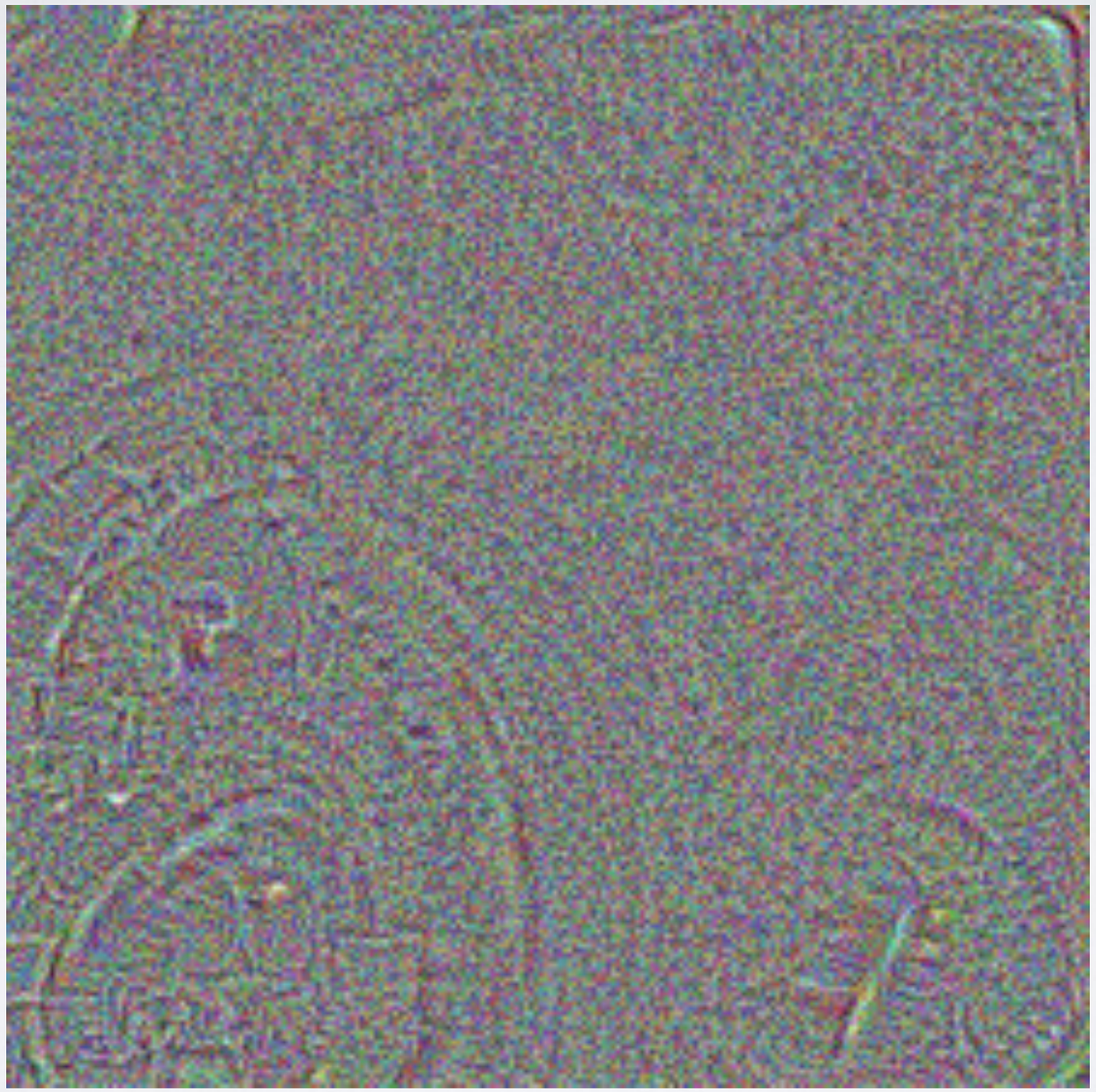
99.99%で識別される



class 69: trilobite

学習済みVGG16モデルで作ってみた

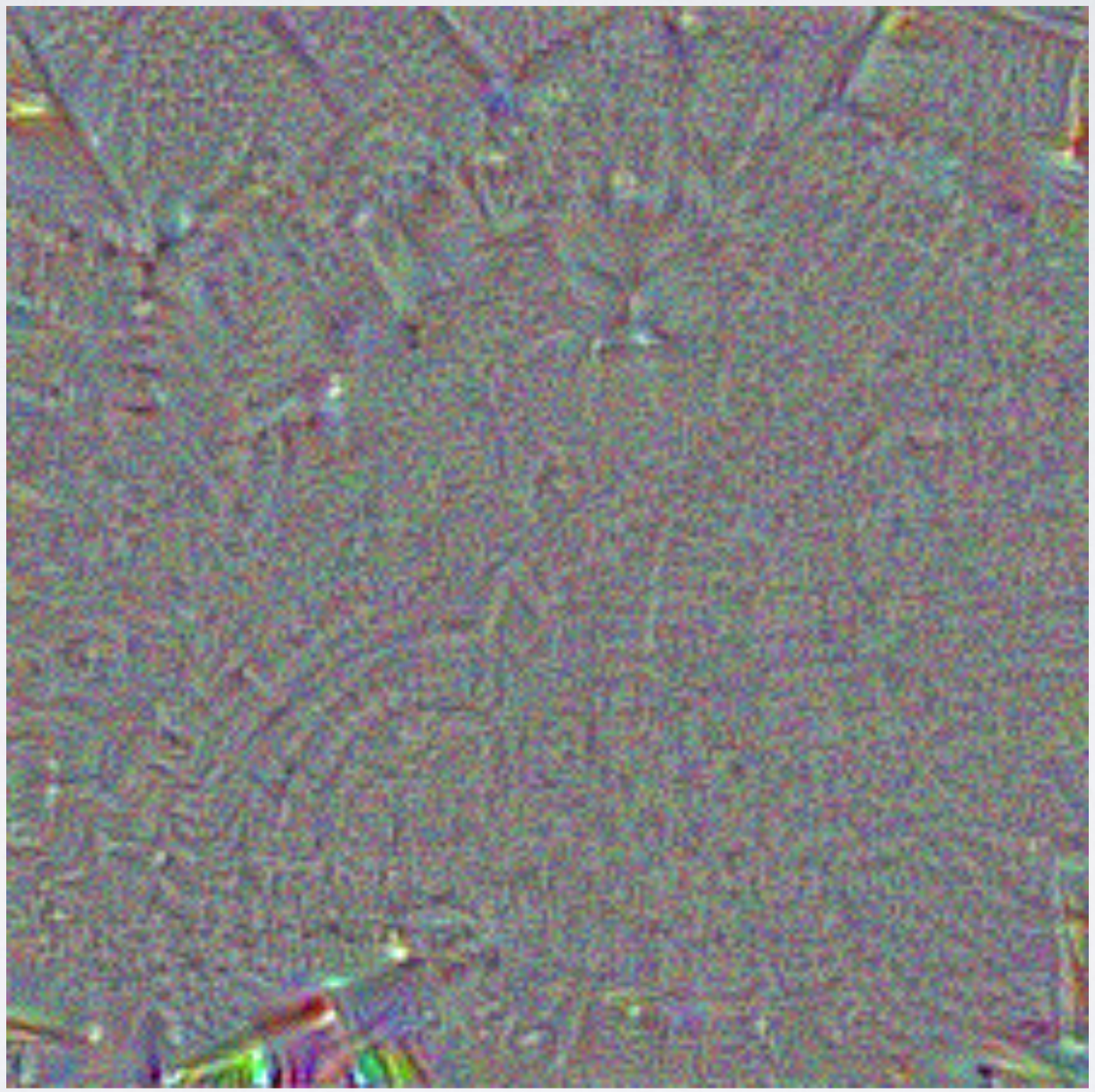
99.99%で識別される



class 605: iPod

学習済みVGG16モデルで作ってみた

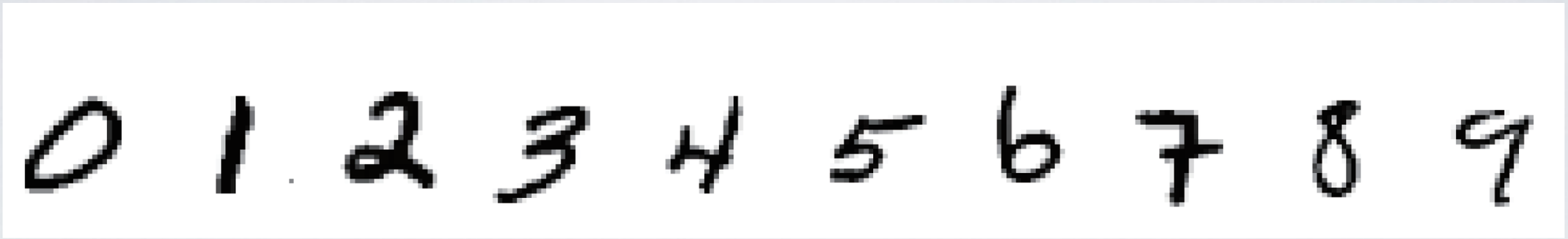
99.99%で識別される



class 624: library

深層学習における汎化とは何か？

[Zhang et al, Nov. ‘16]の実験



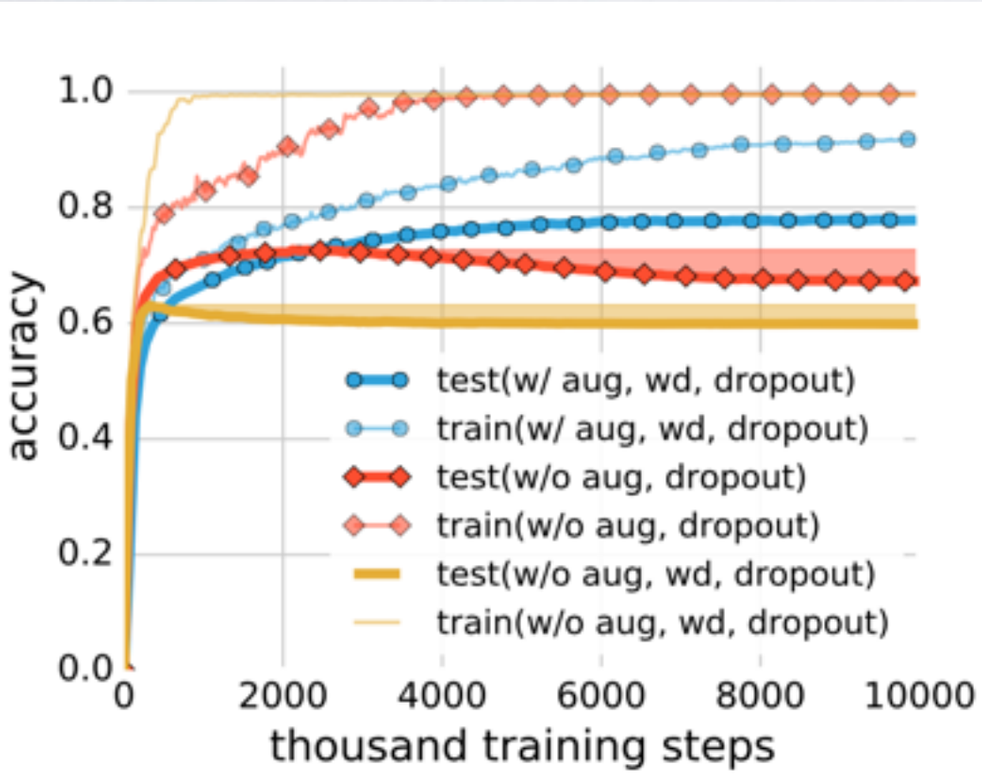
正解ラベルをランダムに付与した擬似データセット

→ 学習すべきパターンがないにもかかわらず、極めて低い訓練誤差まで学習できる

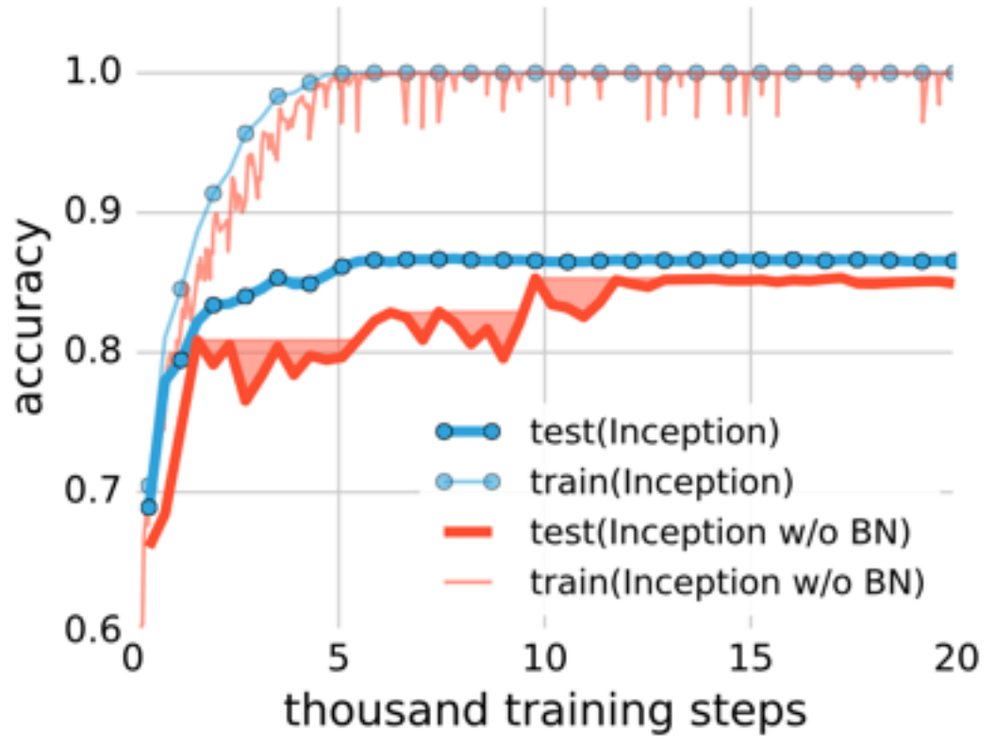
訓練データを「丸暗記」もしている！？

正則化はそんなに強力か？

[Zhang et al, Nov. ‘16]の実験



(a) Inception on ImageNet



(b) Inception on CIFAR10

understanding deep learning requests rethinking generalization

結局何もよくわかっていない

前からの問題

trainability

generalizability

representability

加えて多層ニューラルネットの学習の仕組みはわれわれが思い描いてたようなものではない可能性。単純な議論を拒むほど深い系。

ただし、人類が手にした最強のフィッティング関数である事は確か！

理論の部終わり

実験の時間

Googleが提供するオープンソースの機械学習用フレームワーク(ライブラリー)

Google Brainが**2011**年から内部で用いていた**DistBelief**の発展版(多分)

Apache 2.0オープンソースラインセンス！！

いろんな機械学習フレームワーク(ライブラリ)

Theano

Pylearn2

Caffe

Chainer

ThensorFlow

Keras

Lasagne

・・・

基本的にはすべてPython  **(か、ごく一部C++)**

NumPy

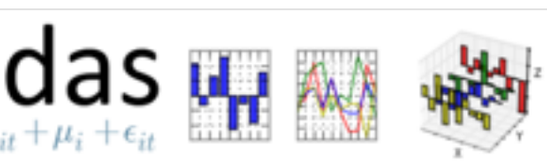
SciPy

matplotlib

jupyter

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



scikit learn

python