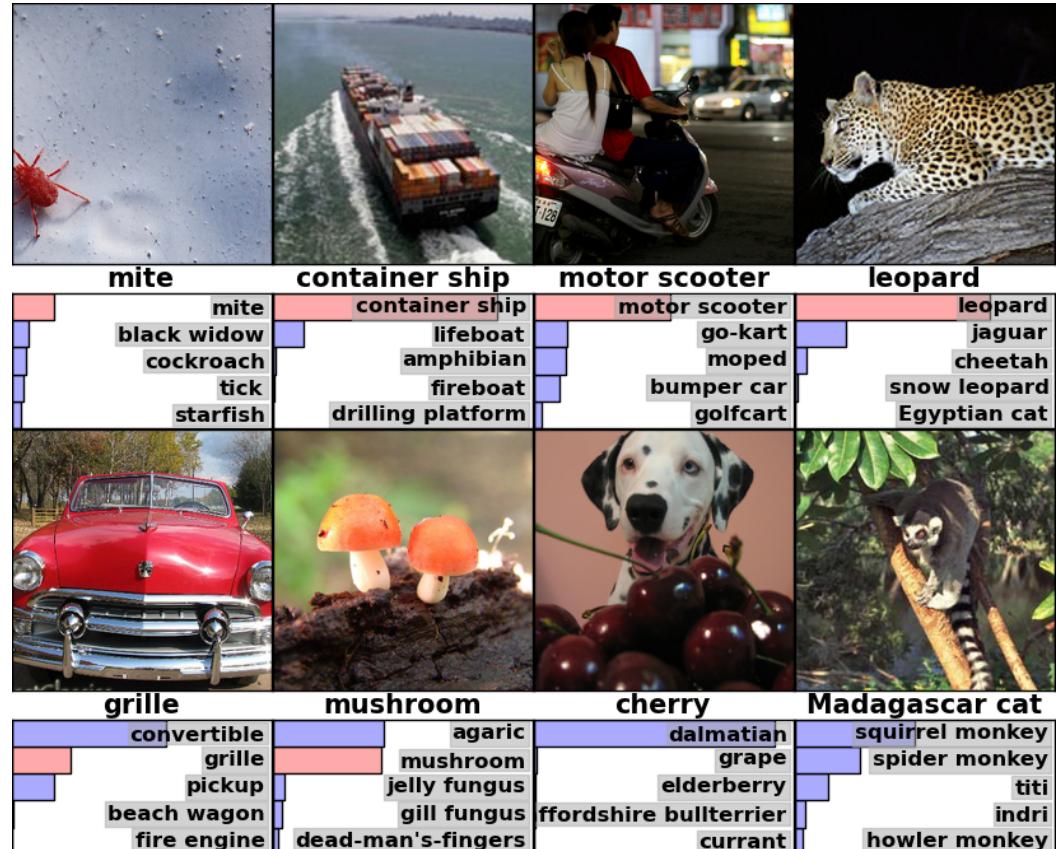


# 深層学習技術の進展

- ・画像認識
- ・音声認識
- ・自然言語処理
- ・機械翻訳

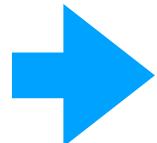
深層学習技術は、これらの分野において特に圧倒的な強みを見せている

ImageNet Classification

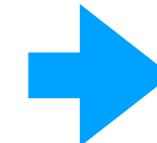


cited from: ``ImageNet Classification with Deep Convolutional Neural Networks'', Alex Krizhevsky et al.

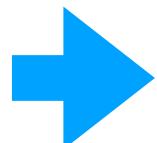
# こんな関数を作りたい！



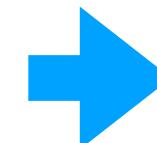
$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$



0: 猫



$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$



1: 犬



関数の形状を制御するパラメータを導入する

$$f_{\Theta} : \mathbb{R}^n \rightarrow \mathbb{R}$$



関数の形状をコントロールするパラメータ群



訓練データを準備する



, 1}



, 1}



, 1}



, 0}



, 0}



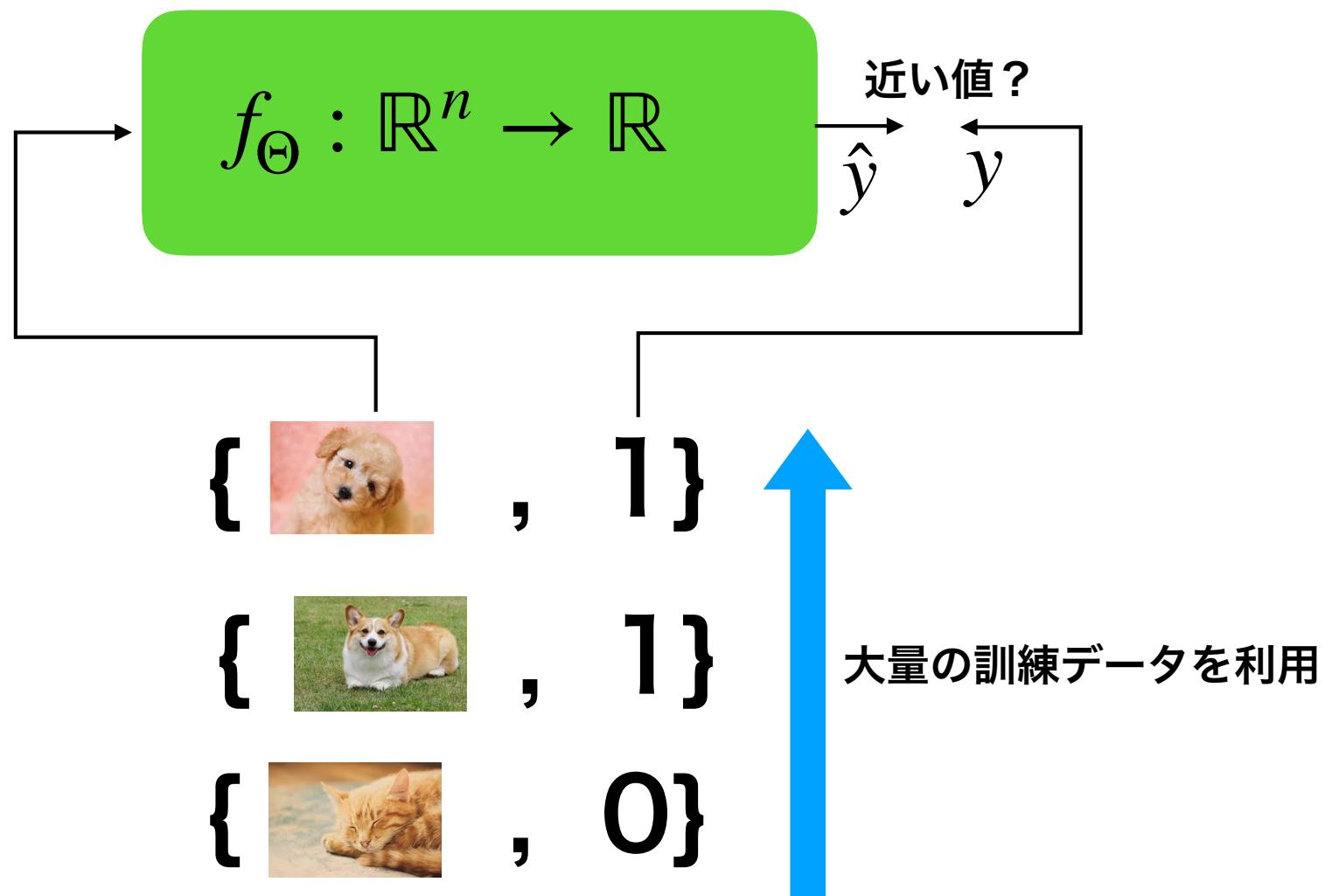
, 0}





## パラメータを更新する(訓練・学習)

出力と教師ラベルとの間の**食い違い**がなるべく小さくなるように  
パラメータを更新する

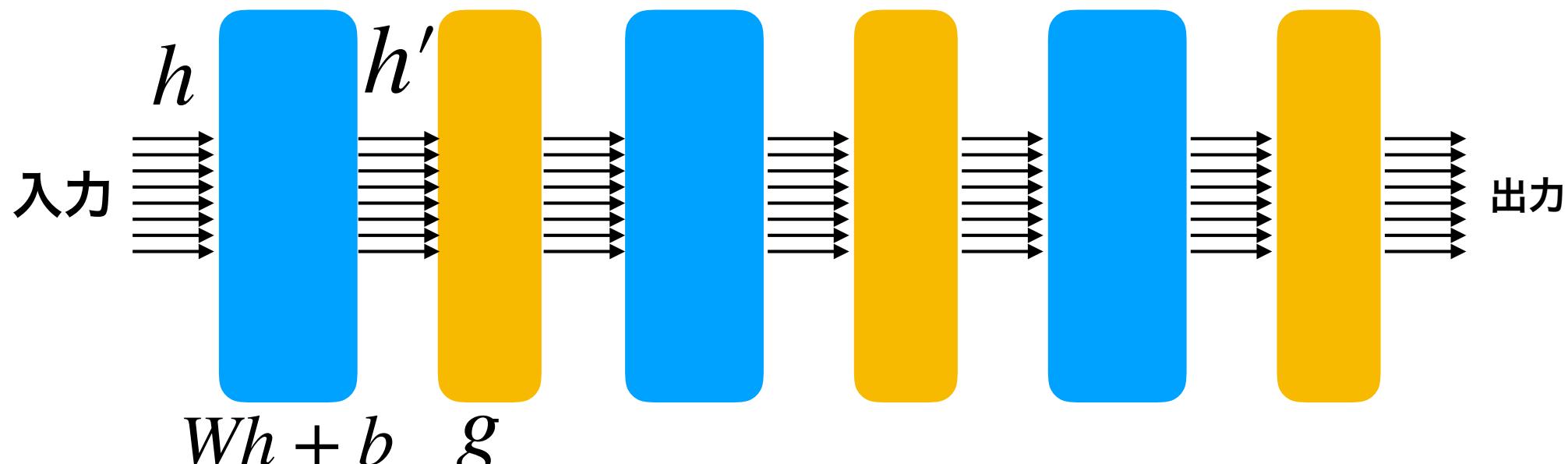


# 深層ネットワークモデル

$$f_{\Theta} : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

線形層 活性化関数

$$\Theta = \{W_1, b_1, W_2, b_2, \dots\}$$



$$h' = \begin{matrix} W \\ h + b \end{matrix}$$

係数行列 バイアス

活性化関数

# アフィン変換

$$h' = W h + b$$

シグモイド関数

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

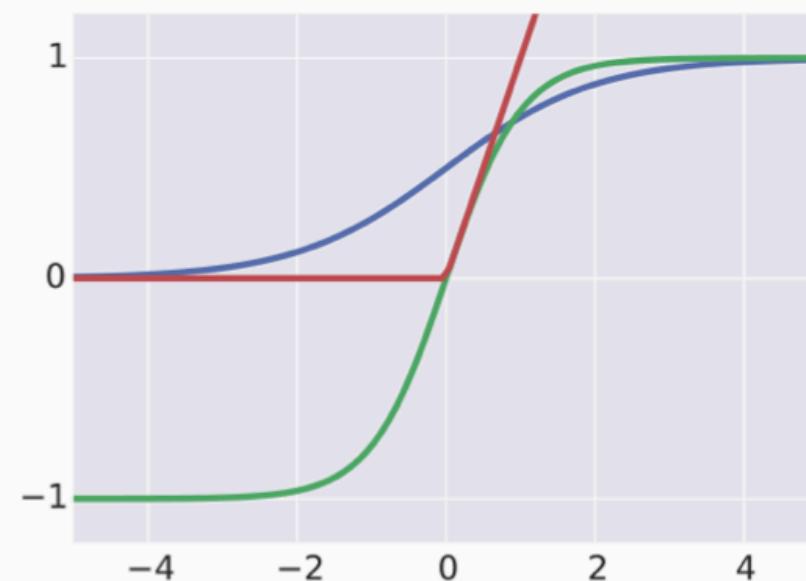
双曲線正接関数

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

ランプ関数

$$\text{ReLU}(u) = \max(0, u)$$

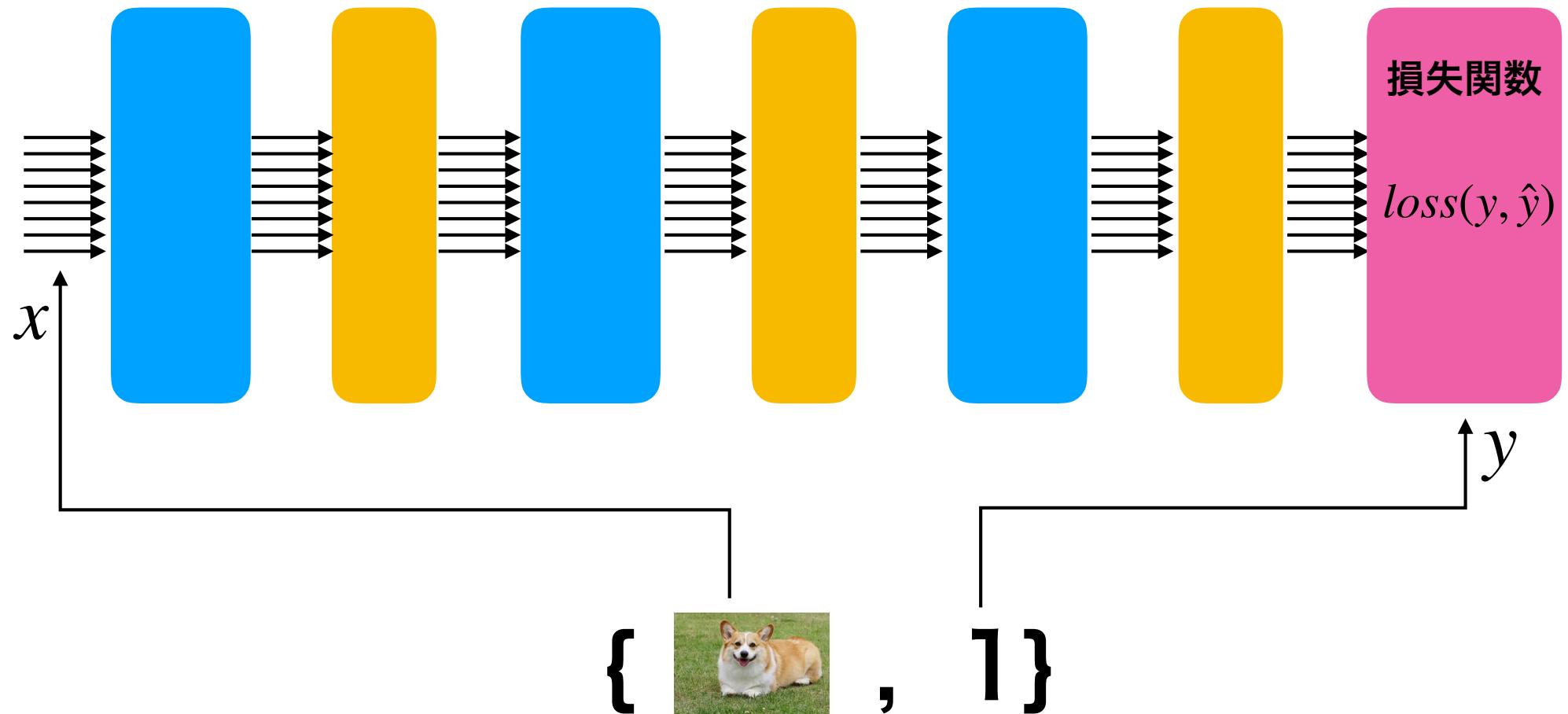
## 活性化関数



# 深層ネットワークの訓練

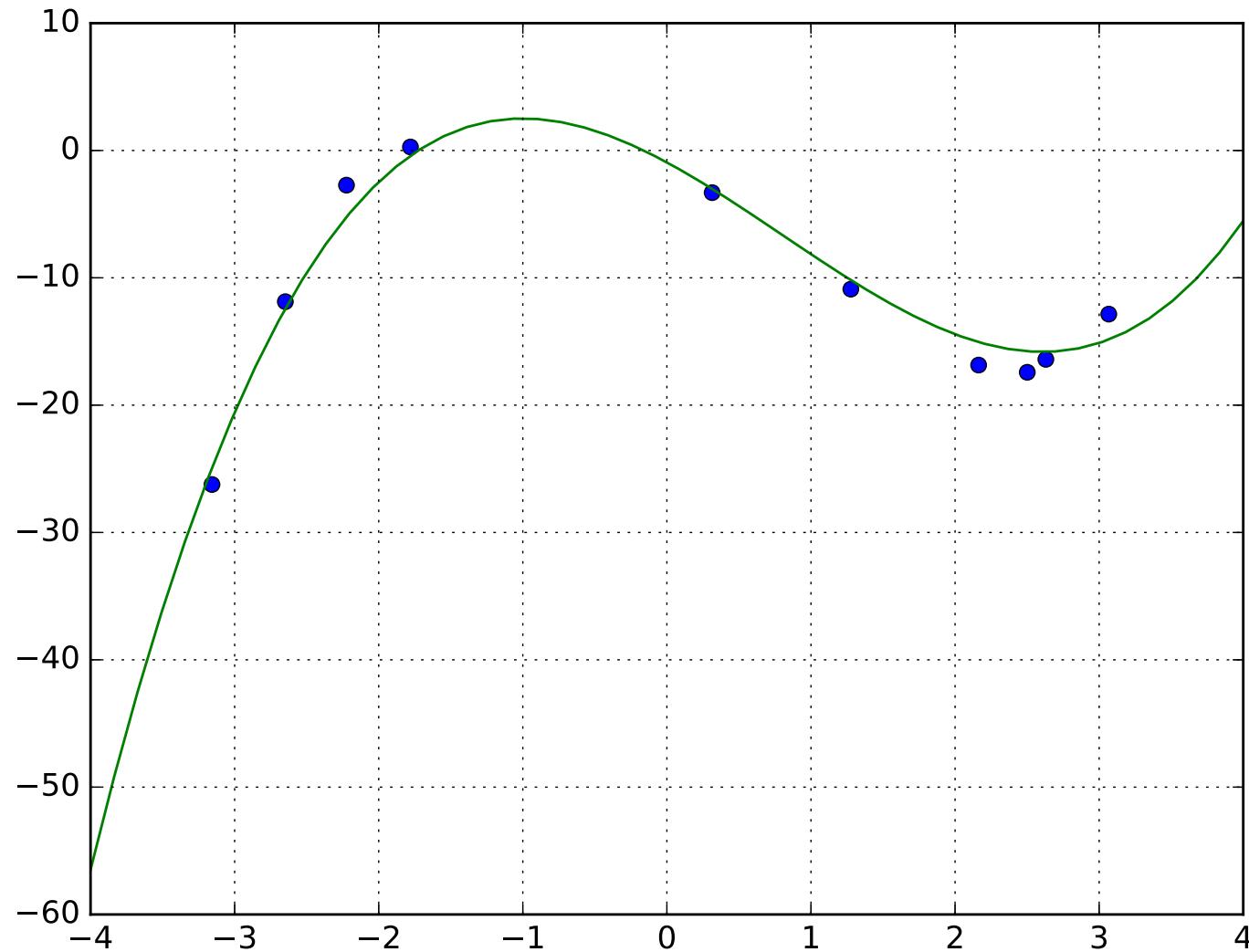
$$\Theta = \{W_1, b_1, W_2, b_2, \dots\}$$

$$\hat{y} = f_{\Theta}(x)$$



訓練・学習過程では、損失関数值を最小化するようにパラメータを変更する

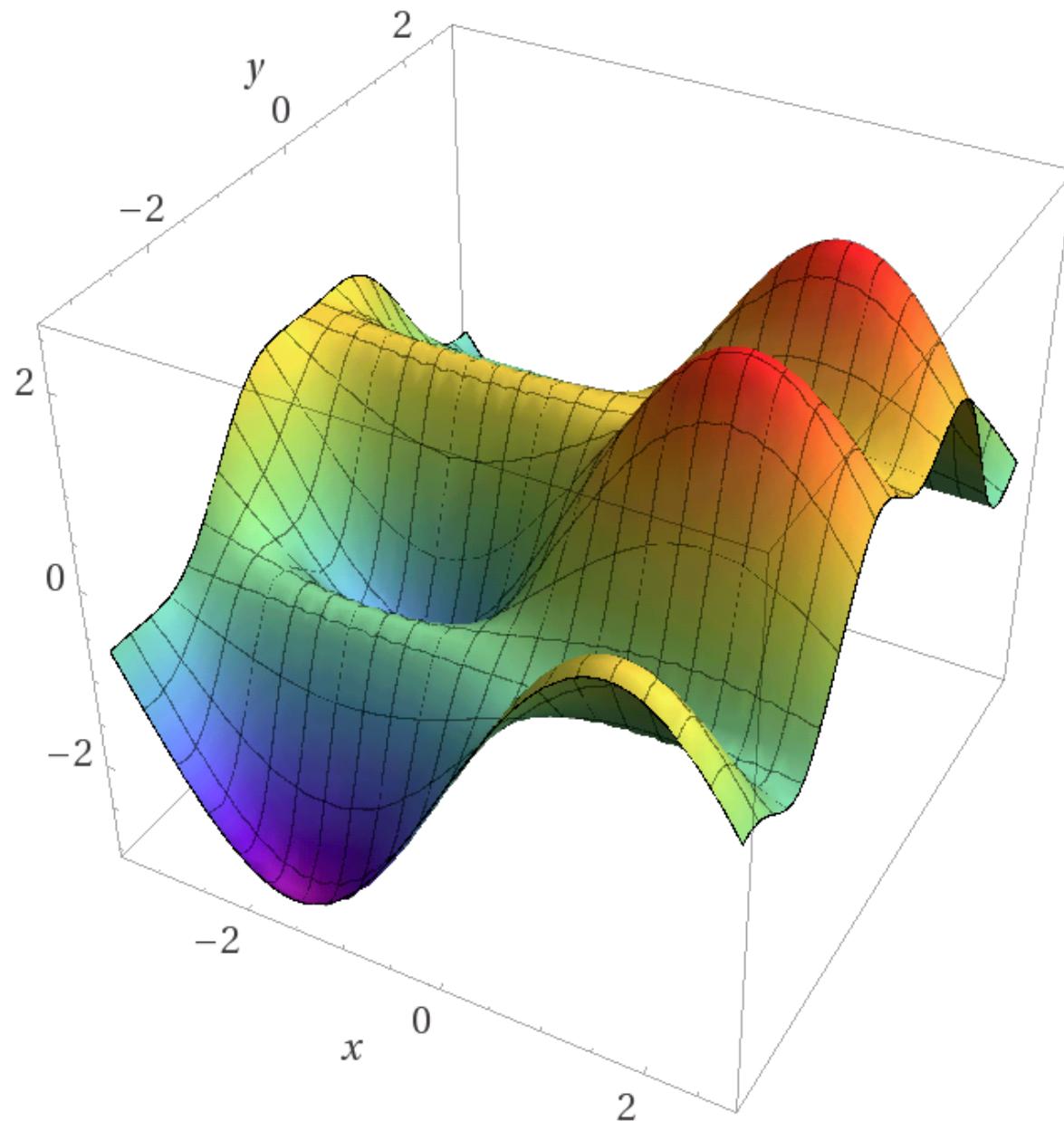
# 深層学習 = 最小二乗法の親玉？



# 補足

-  最適化技法として確率的勾配法を利用
-  パラメータの勾配ベクトルの効率的な計算には誤差逆伝播法(back prop)を利用
-  よい汎化性能を得るためにには、大量の訓練データが必要
-  一般には、非凸最適化となる

# 非凸関数のイメージ



# 確率的勾配法(SGD)

訓練データ  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_T, y_T)\}$

ミニバッチ  $B = \{(x_{b1}, y_{b1}), (x_{b2}, y_{b2}), \dots, (x_{bK}, y_{bK})\}$

目的関数  $G_B(\Theta) = \frac{1}{K} \sum_{k=1}^K loss(y_{bk} - f_\Theta(x_{bk}))$

## 確率的勾配法に基づく最小化

Step 1 (初期点設定)  $\Theta := \Theta_0$

Step 2 (ミニバッチ取得)  $B$  をランダムに生成

Step 3 (勾配ベクトルの計算)  $g := \nabla G_B(\Theta)$

Step 4 (探索点更新)  $\Theta := \Theta - \alpha g$

Step 5 (反復) Step 2 に戻る

## バリエーション

Momentum

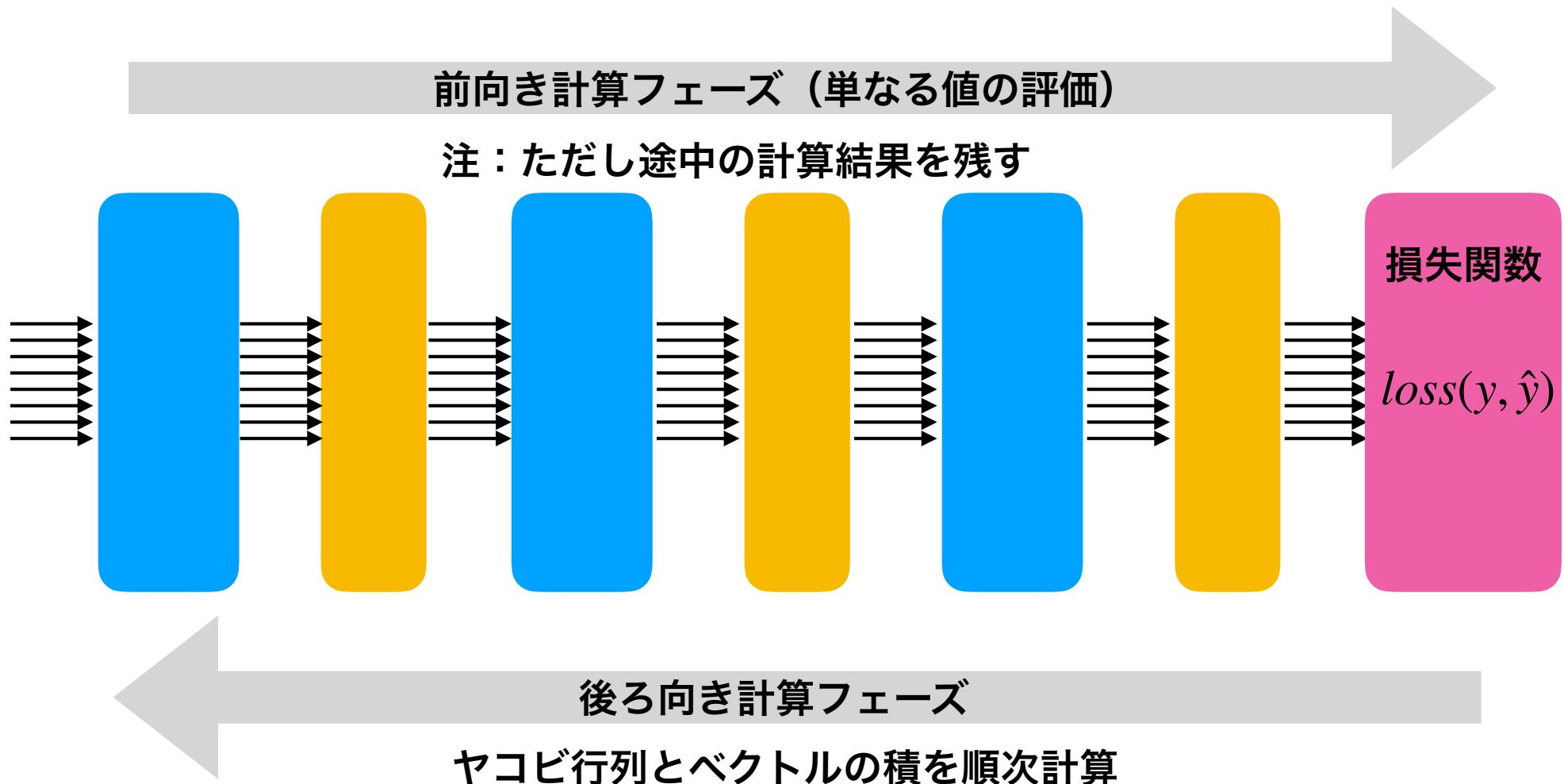
AdaDelta

RMSprop

Adam

# 誤差逆伝播法(backprop)

- ・パラメータの勾配ベクトルを効率良く求めることが目的
- ・微分の連鎖律の利用(BCJRアルゴリズムにとても良く似ている)

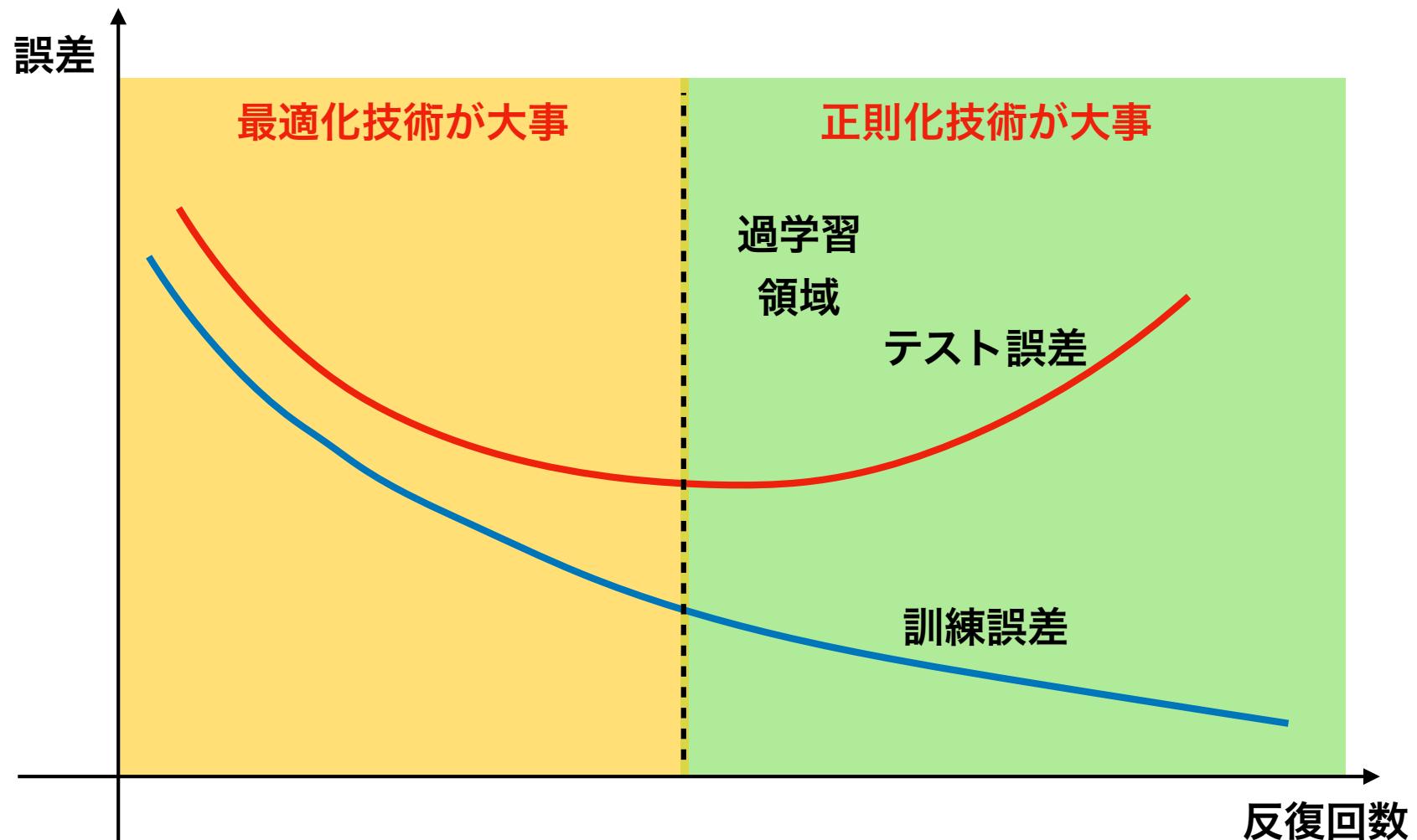


# DL計算環境について

	コスト	メリット	デメリット
Google Colabatory	無料	環境構築不要	12時間以上の連続計算ができない
AWS, GCPなどクラウドを借りる	有料	環境構築ボタンひとつ・メンテ不要	GPU付きインスタンスを借りるとそこそこのお値段
GPUつきマシンを買う	初期投資が必要	手元に実行環境があるのは色々快適	GPUの稼働率はそれほど高くない。。。
CPU	手元のCPUを利用すればタダ	気軽に試せる	遅い(相対的に)

# 汎化誤差

訓練データに含まれない初見のデータに対して適切な出力を与えることが望ましい



# 深層学習技術の分類

A. 関数近似のための  
最適化技術

確率的勾配法  
誤差逆伝播法  
勾配消失を抑制  
する活性化関数

B. 汎化誤差を小さくする  
ための技術

正則化  
重み共有  
ドロップアウト  
バッチ正規化  
ビッグデータ  
データ拡張

C. 表現学習の  
ための技術

畳み込み  
ネットワーク  
重み共有  
ネットワーク構造  
埋め込み

# DLフレームワーク

- PyTorchによるプログラム例



```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(2, 2)
        self.fc2 = nn.Linear(2, 2)
    def forward(self, x):
        x = F.sigmoid(self.fc1(x))
        x = F.sigmoid(self.fc2(x))
        return x
```

ネットワークの定義部

順方向計算のみを記述すればよい。

誤差逆伝播法の後ろ向き計算フェーズは明示的にユーザが書く必要ない

GPUの利用も非常に簡単。  
深層学習以外の分野でも有用！

model = Net() ネットワークのインスタンス化

loss\_func = nn.MSELoss() 損失関数の指定

optimizer = optim.Adam(model.parameters(), lr=0.1) オプティマイザの指定

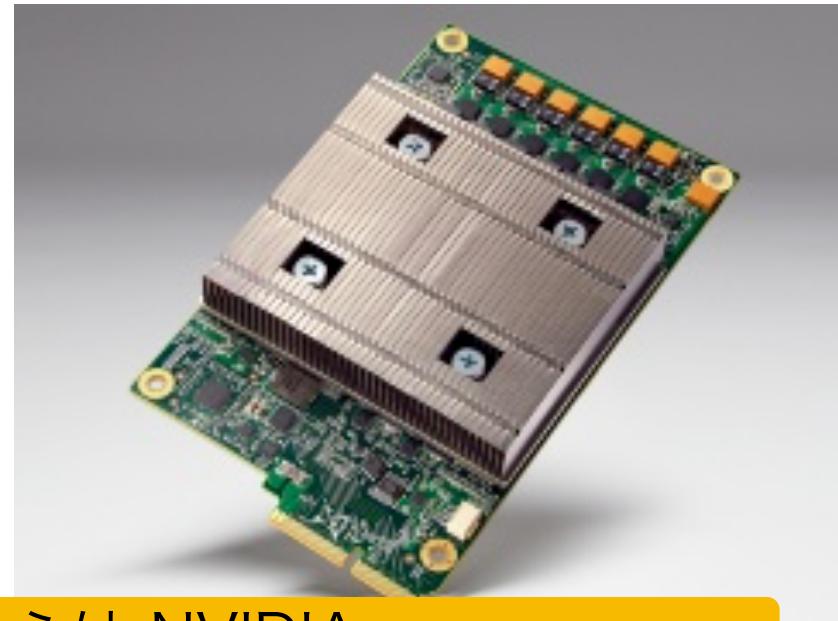
# GPU/ASIC

NVIDIA TESLA GPU



出典 <http://www.nvidia.co.jp/object/tesla.html>

Google Tensor Processing Unit  
(TPU)



コスパの点からは NVIDIA  
GeForce1080Ti あたりがオススメ!  
(スーパーリッチな人はV100 \* 16個の  
NGX-2 (半精度2PFlops, 4000万超) を)

01464/

# オススメの本

