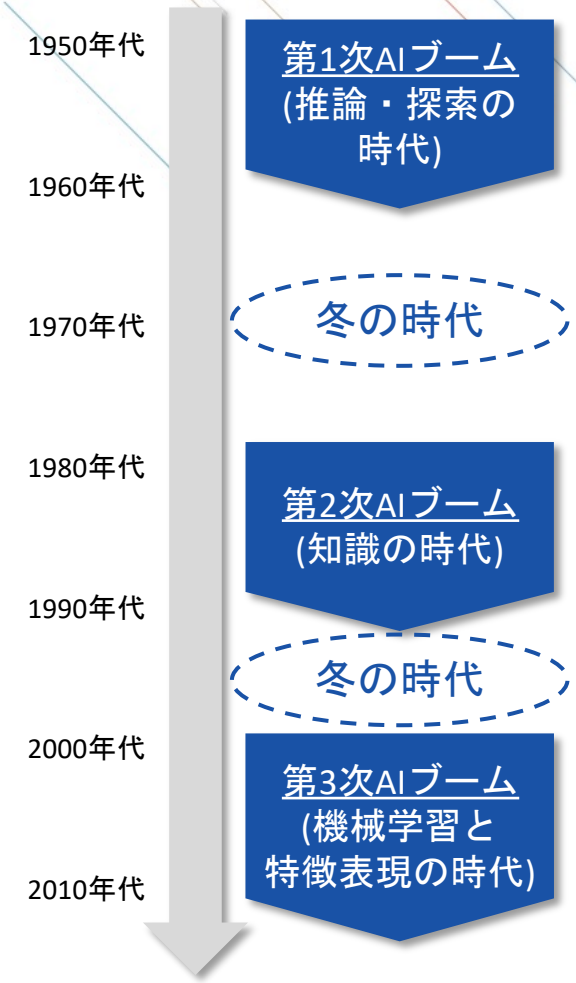
ディープラーニングとは？

人工知能の歴史とディープラーニング

人工知能（AI）の歴史

AIはこれまで「ブーム」と「冬の時代」を繰り返している。
現在は、機械学習とDeep Leaningの登場により、第3次AIブームを迎えている。



- ✓ 1950年代以降、コンピューターによる「推論」や「探索」が可能となり、特定の問題に対して解を提示できるようになった。
- ✓ しかし、迷路の解き方や定理の証明のような単純な仮説の問題を扱うことはできても、様々な要因が絡み合っているような現実社会の課題を解くことはできないことが明らかになり、一転して冬の時代を迎えた。

- ✓ 1980年代に入り、「知識」を与えることでAIが実用可能な水準に達し、多数のエキスパートシステムが生み出された。
- ✓ しかし、知識そのものは、コンピュータが理解可能なように人が記述する必要があったことから、実際に活用可能な知識量には自ずと限界があることがわかり、1995年頃から冬の時代を迎えた。

- ✓ 「ビッグデータ」と呼ばれているような大量のデータを用いることでAI自身が知識を獲得する「機械学習」が実用化され、次いで知識を定義する要素(特徴表現)をAIが自ら習得する「Deep Learning」が登場した。

AI/機械学習/Deep Learningの関係

AIを実現する技術/アルゴリズムの一つとして、Deep Learningが注目を集めている。

人工知能: 機械に人間の知能を持たせる技術の総称

(Artificial Intelligence, AI)

構成要素

- 自然言語処理
- インテリジェントエージェント
- **機械学習**
- コンピュータビジョン
- 自立制御機械・ロボット
- IoT etc.

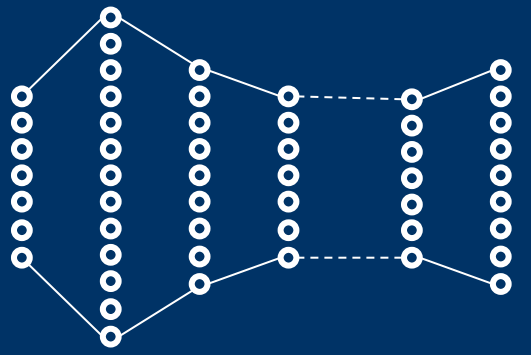
機械学習: AIを実現するための要素技術の一つ

(Machine Learning, ML)

アルゴリズム例

- 線形回帰
- 決定木
- ランダムフォレスト
- **Deep Learning**
- 強化学習 etc.

深層学習: Deep Learning

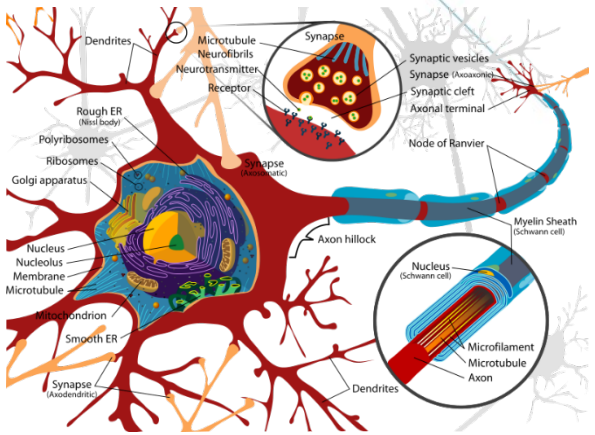


脳の中の神経構造に着想を得て、
その構造を模したアルゴリズム

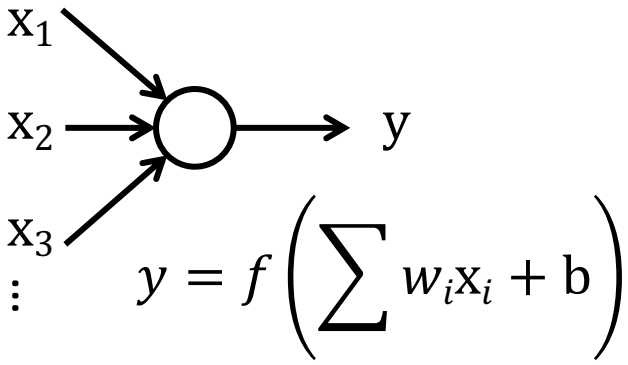
Deep Learningとは

脳の学習機能をコンピュータでシミュレーションするニューラルネットワークを用いた技術である。
近年の技術的な発展の背景には、計算機能力の向上がある。

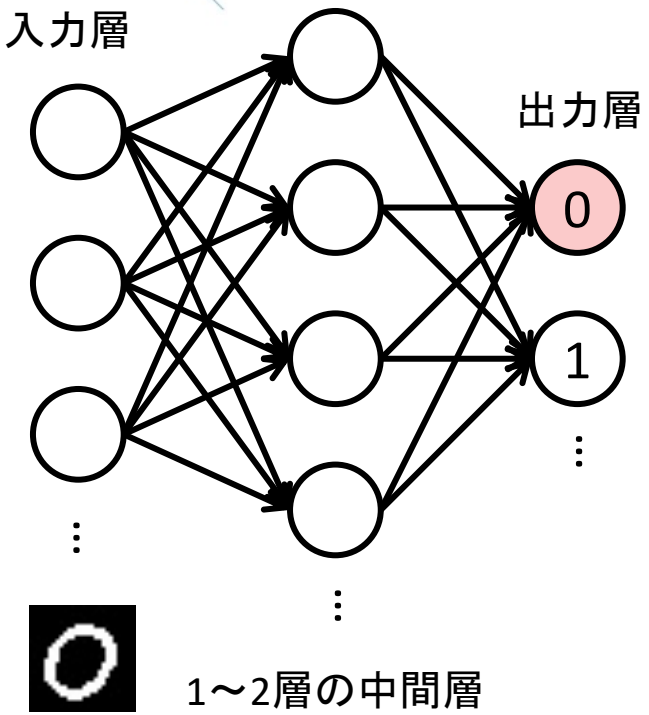
神経細胞



人工ニューロン

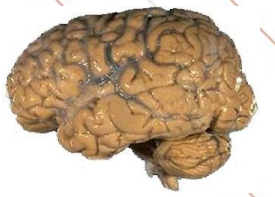


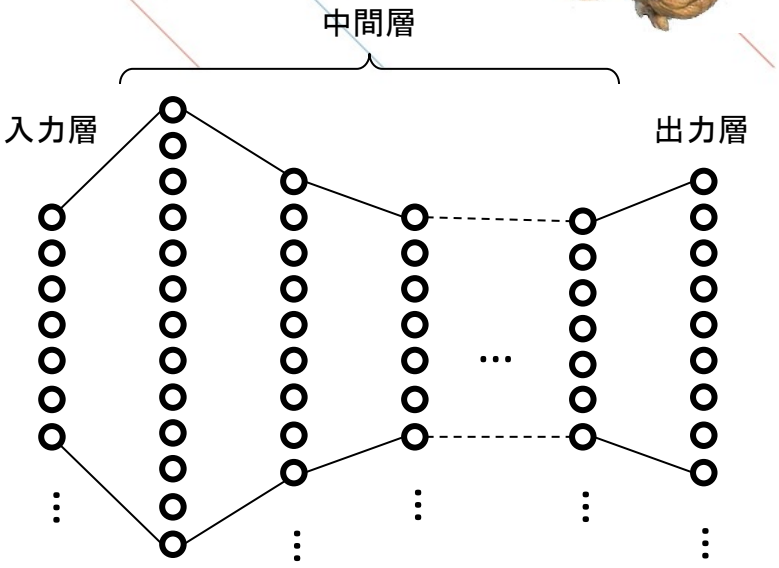
ニューラルネットワーク
（1960～1990頃）



1～2層の中間層

Deep Learning
(2006～)



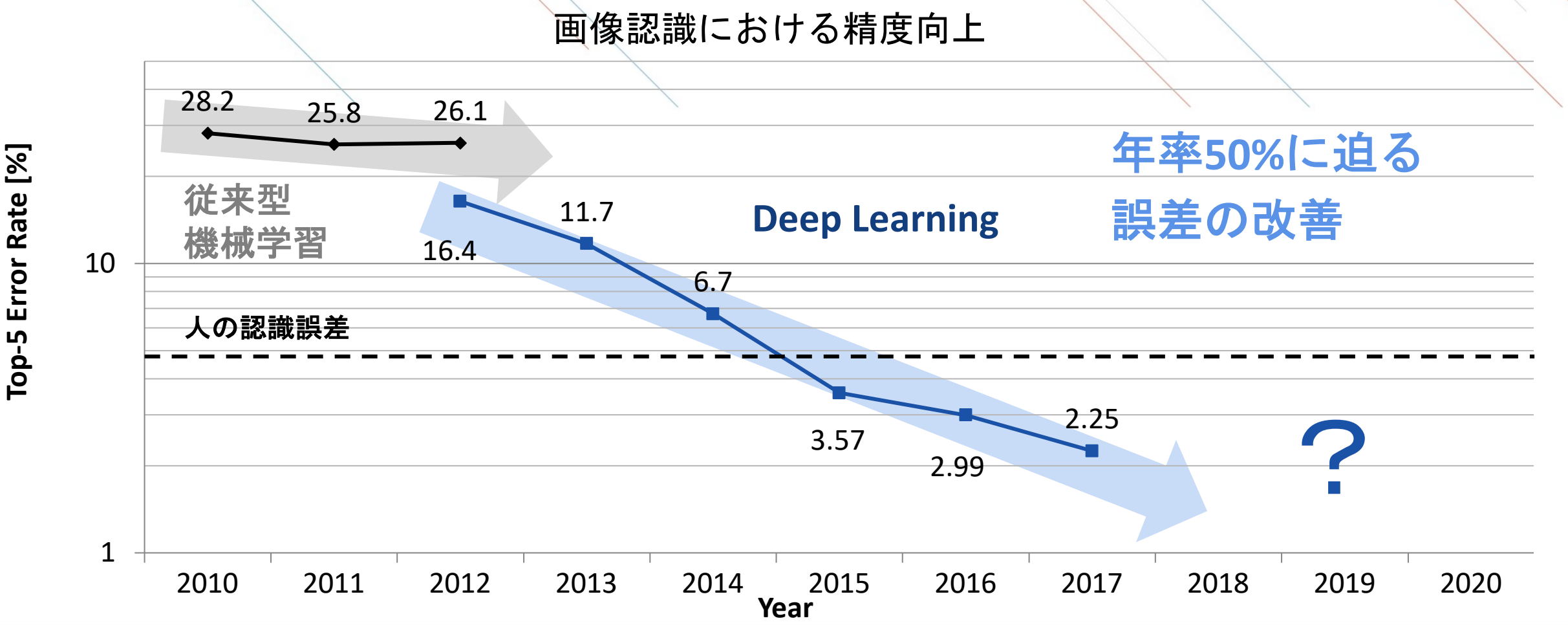


3層～1000以上の中間層
数十M規模のニューロン数

大規模なニューラルネットワークの学習が可能になり、大幅に性能向上

圧倒的な認識性能を示すDeep Learning

画像認識の分野では、Deep Learningにより認識精度がはるかに向上した。
Deep Learningの精度は人の認識精度をも上回っている。



圧倒的な認識性能を示すDeep Learning

画像以外での分野でもDeep Learningの利用が広がっている。
いずれの分野でも、従来の性能限界を打ち破り、技術革新が進んでいる。

画像以外の分野でのDeep Learningの適応例

音声認識

- 2011年　音声認識にDeep Learningを適用し、音声認識誤差を30%前後改善
  スマートフォン等で音声認識が一般化する契機に
- 2016年10月　Microsoftは音声認識技術において人間並みの性能を実現したと発表
- 2019年5月　Googleがリアルタイムで文字起こしができる自動字幕プログラムを発表

囲碁

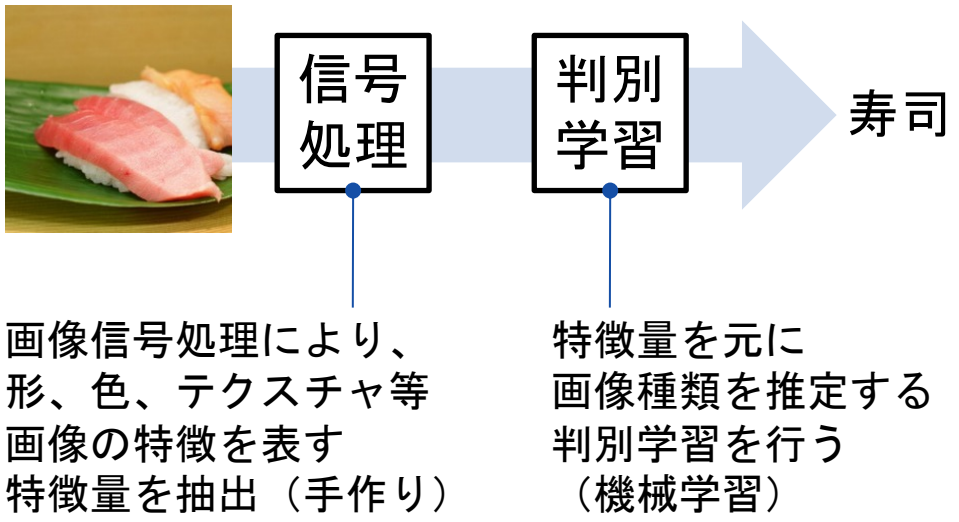
- 2015年10月　Google傘下のDeep Mindが開発したDeep Learningによる囲碁プログラム
  Alpha Goがプロ棋士に勝利
- 2016年3月　世界最強棋士の一人である李セドル九段に勝利
- 2018年5月　Facebook開発のELF OpenGoが世界ランク上位30位のプロ棋士に14試合全勝

Deep Learningにより認識機全体を自動的に獲得

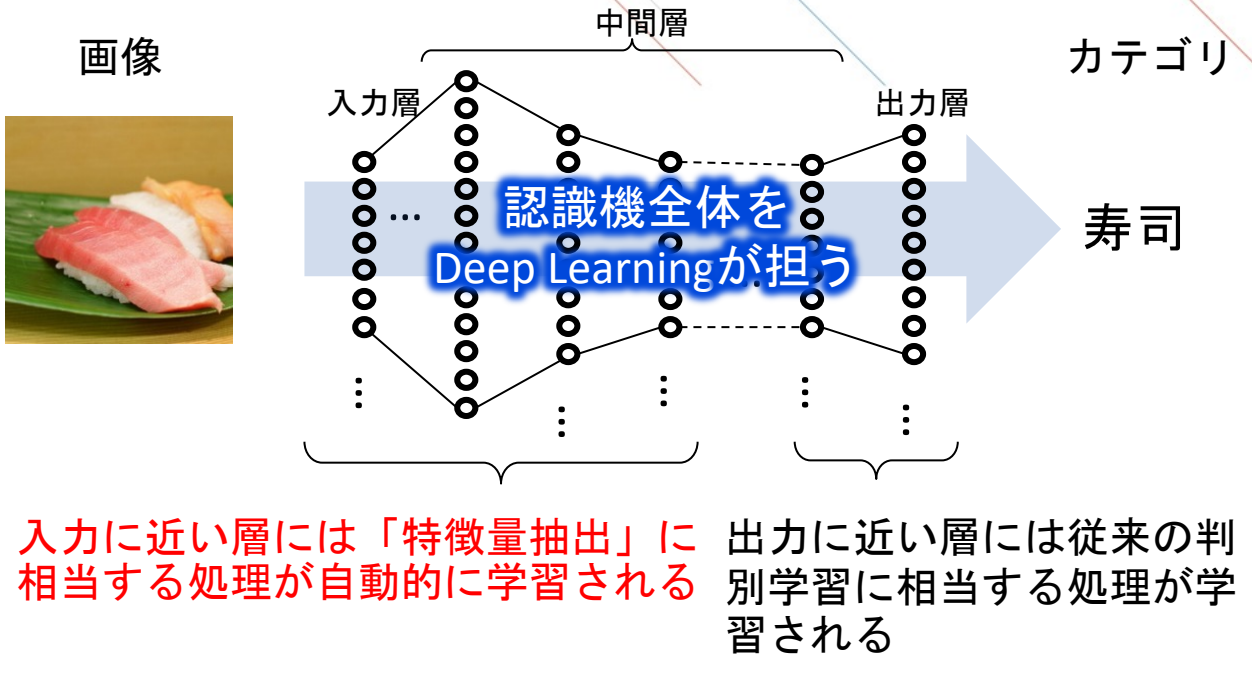
従来型の機械学習は、特徴量抽出のノウハウが性能を左右していた。

Deep Learningでは、特徴量抽出は自動的に学習されるため、データの量と質が性能を決める。

従来型機械学習による画像認識



Deep Learningによる画像認識



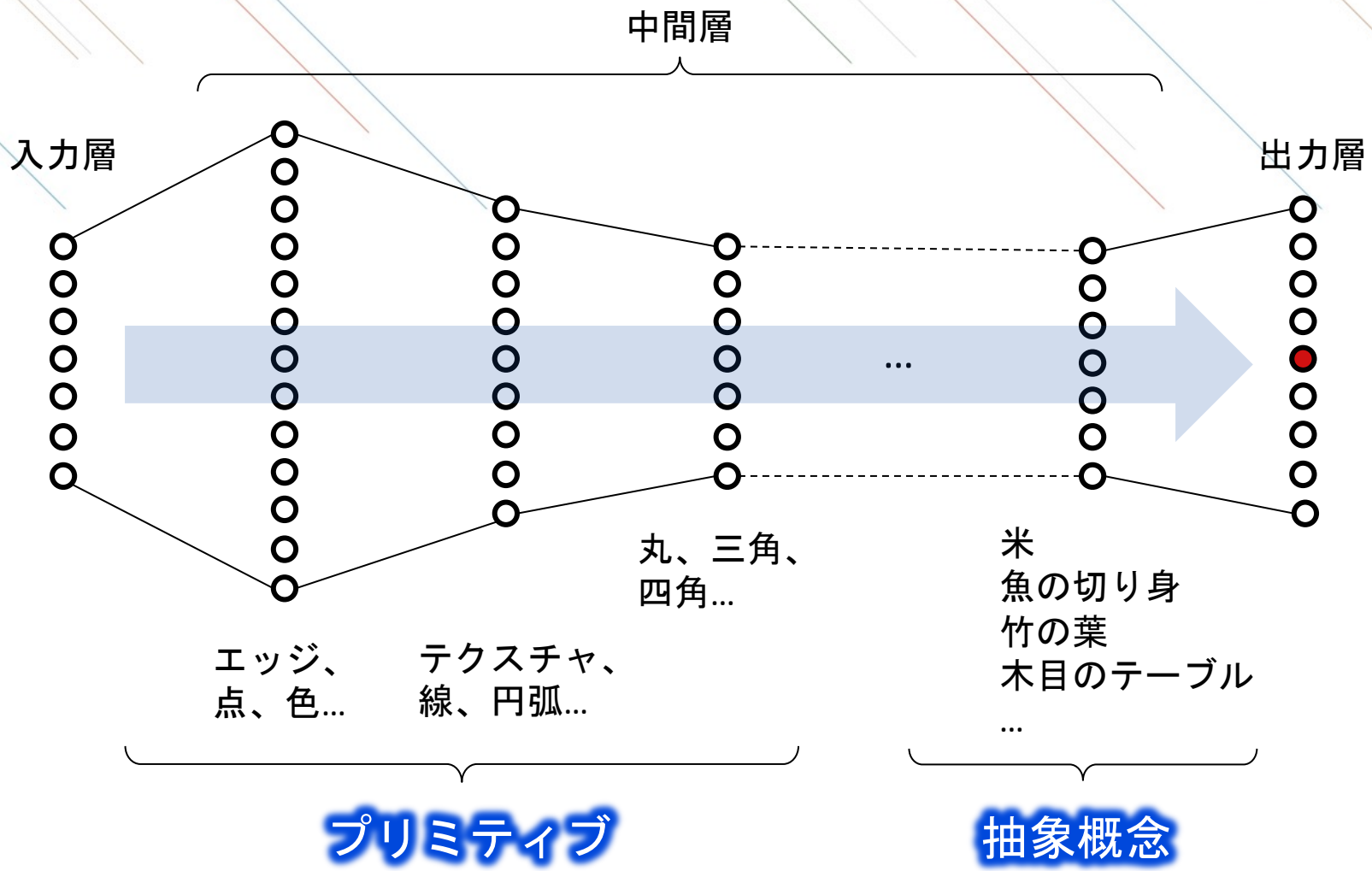
- 特徴量を抽出する信号処理の設計に多大な労力と専門知識を要していた
- 最適化に限界があった

- 所望の入出力関係を与えるだけで、認識機を獲得可能に
- データに合わせた特徴量の最適化が可能になり精度が向上

学習されたニューラルネットワークの分析

学習の結果、人間の脳に似た機能が獲得されることが実験的に確認されている。

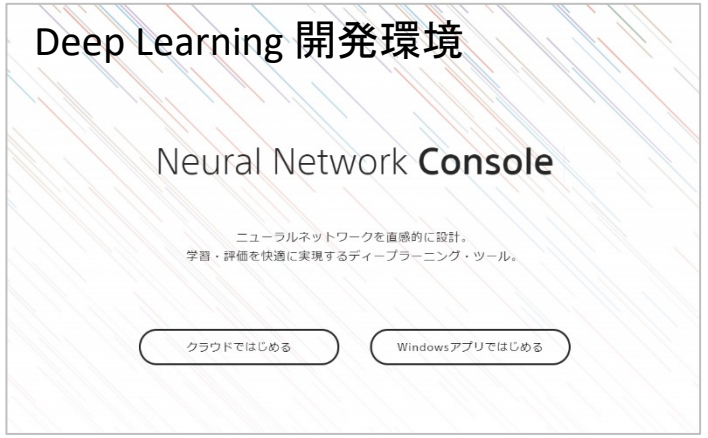
入力データ



SONYにおけるAIへの取り組み

ソニー×AI

AIとロボティクスの領域で進化させてきた要素技術とDeep Learningなどの最新技術を融合し、
パートナー企業や研究機関とともに、より広域な領域でのAIの創出に挑戦している。



ソニーのDeep Learningに対する取り組み

AIやDeep Learningが注目される以前から、Deep Learningを含めた機械学習の研究開発を行ってきた。
近年では、積極的に開発技術のオープンソース化などAI時代を見据えた取り組みを行っている。

2000年以前～ 機械学習の研究開発

2010年～ Deep Learningの研究開発

2010年～ Deep Learning開発者向けソフトウェアの開発

2011年～
初代コアライブラリ

2013年～
第二世代コアライブラリ

2016年～ 第3世代コアライブラリ

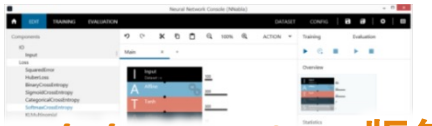
Neural Network **Libraries**

17/6/27 オープンソースとして公開

Deep Learningを用いた認識技術等の
開発者が用いるソフトウェア群

技術開発効率を圧倒的に向上

2015年～ GUIツール



Neural Network
Console

17/8/17 Windows版無償公開
18/5/9 クラウド版正式サービス開始

Neural Network Libraries / Consoleとは

研究開発における課題を解決し、Deep Learningの研究開発を効率化するソフトウェア

Neural Network Libraries

- Deep Learning研究開発者向けオープンソースフレームワーク（他社製既存Deep Learning FWに相当）
- コーディングを通じて利用→**高い自由度**
- 最先端の研究や製品への実装にも柔軟に対応

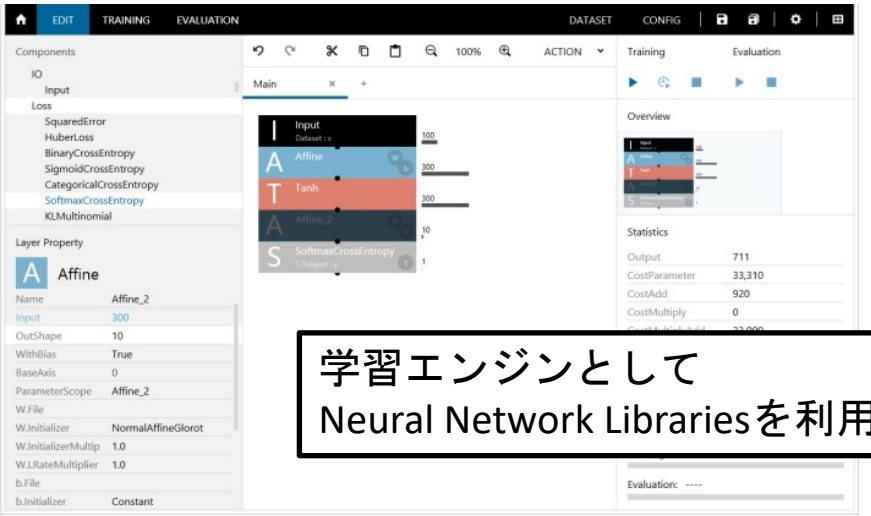
```
import nnabla as nn
import nnabla.functions as F
import nnabla.parametric_functions as PF


x = nn.Variable(100)
t = nn.Variable(10)
h = F.tanh(PF.affine(x, 300, name='affine1'))
y = PF.affine(h, 10, name='affine2')
loss = F.mean(F.softmax_cross_entropy(y, t))
```

主なターゲット

- **じっくりと研究・開発に取り組まれる方**
- **プログラミング可能な研究、開発者**

Neural Network Console

- 研究や、商用レベルの技術開発に対応したDeep Learningツール
- 様々なサポート機能→**高い開発効率**
- GUIによるビジュアルな操作→**敷居が低い**



学習エンジンとして
Neural Network Librariesを利用

主なターゲット

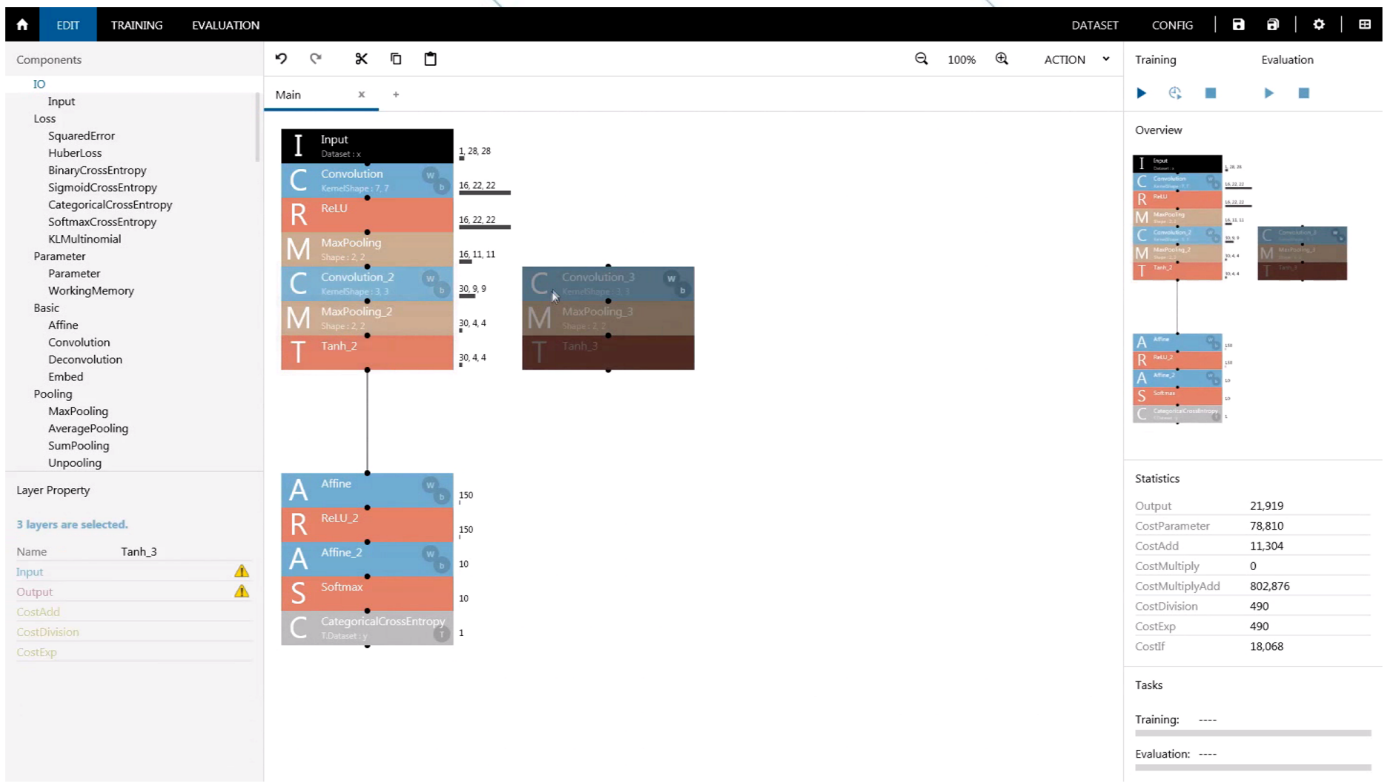
- **特に開発効率を重視される方**
- **はじめてDeep Learningに触れる方**

Demo Movie

https://www.youtube.com/watch?v=1AsLmVniy0k

特長1. ドラッグ＆ドロップによるニューラルネットワーク構造の編集

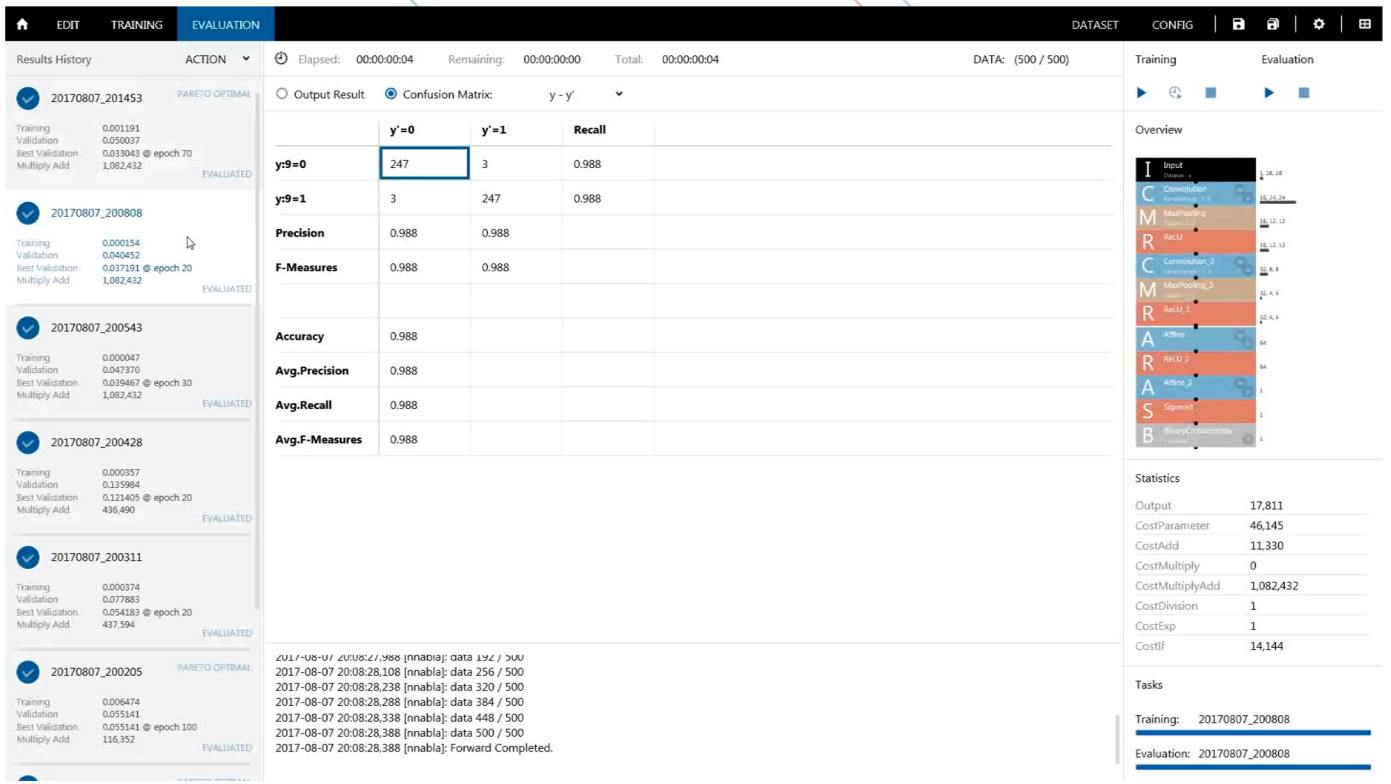
コーディングレスでのニューラルネットワーク設計を実現（コーディングスキル不要）
複雑なニューラルネットワークもすばやく構築可能（作業時間の短縮）
ニューラルネットワークの構造を視覚的に確認しながら、 短期間でDeep Learningを習得可能



- 関数ブロックを用いたVisual Programmingでニューラルネットワークを設計
- 後段のニューロン数など、ネットワークの設計により変更するパラメータは自動計算
- ネットワーク設計にエラーがある場合はその場で提示
- 画像認識だけでなく、AutoEncoder、RNN、GAN、半教師学習などの設計にも対応

特長2. 試行錯誤結果の管理機能

手動で複数のネットワーク構造を管理する必要がなく、試行錯誤に集中できる
どのようなネットワークで、どの程度の精度が得られたのかの分析が容易



↑学習履歴　↑精度評価結果　　ネットワーク構造のプレビュー↑

- すべての学習の試行を自動的に記録
- 記録した学習結果は、一覧して過去の結果と比較可能
- ネットワーク構造に変更がある場合、差分となる箇所をビジュアルに提示
- 分類問題の場合、Confusion Matrixを表示
- 過去学習したネットワーク構造に遡ることも可能

参考資料

特長3. 構造自動探索機能

ネットワーク構造のチューニングにおける最後の追い込み作業を大幅に効率化
フットプリントも同時最適化するため、HWリソースの限られた組み込み用途にも有効

- ネットワーク構造の変更→評価を自動で繰り返すことにより、優れたネットワーク構造を自動探索する機能
- 精度とフットプリントの同時最適化が可能
- ユーザは、最適化の結果の得られる複数の解の中から、所望の演算量と精度を持つネットワーク構造を選択できる

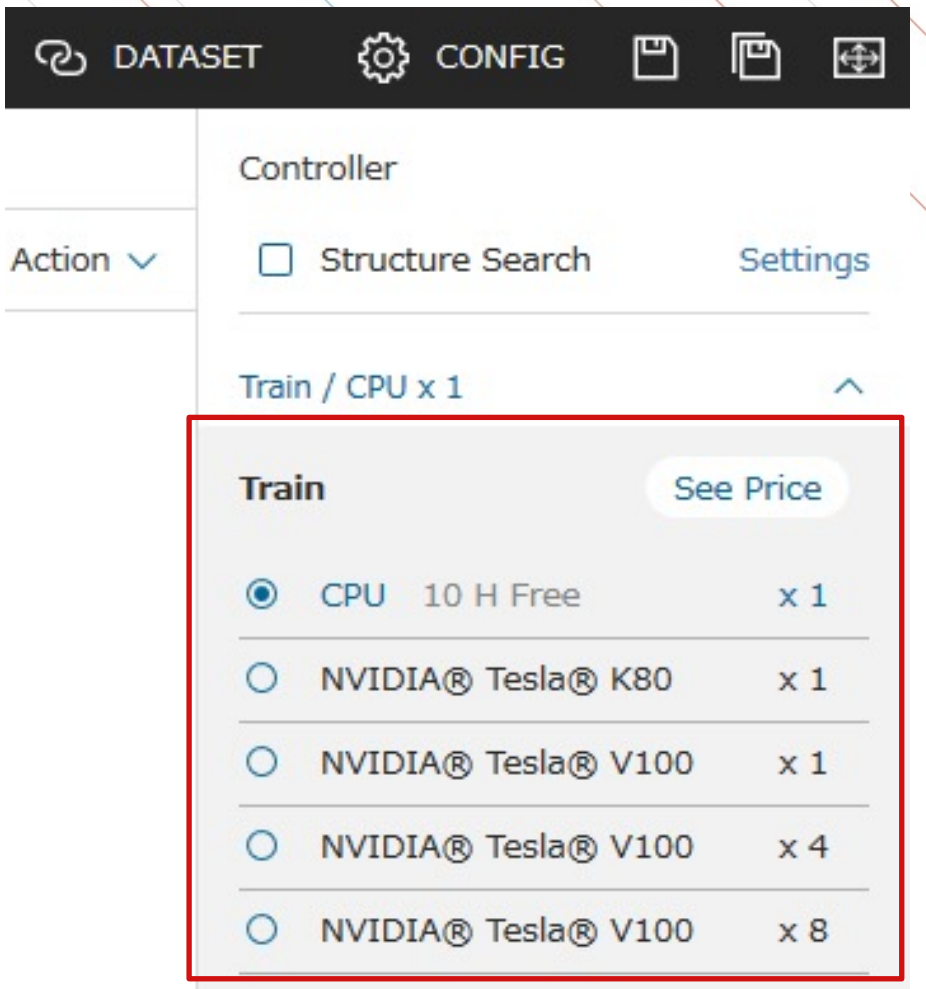
※ グラフの縦軸は誤差、横軸は演算量（log）、各点はそれぞれ異なるニューラルネットワークの構造を示す
※ 動画は最適化済み結果を早送りしたもの

参考資料

特長4. 豊富なGPUリソースを利用可能（クラウド版）

最先端研究者と同等の環境をGUI環境から利用可能

- ニューラルネットワークの学習には膨大な演算が必要
  - 必要な演算量は主に扱うデータの量とニューラルネットワークの構造に依存
- マルチGPUを用いると、学習完了までの時間を大幅に短縮できる
  - 同じ開発期間でより多くの試行錯誤を行うことが可能に
- 環境のセットアップ、メンテナンス作業不要で豊富なGPUリソースを利用可能
  - 開発者はDeep Learningの開発作業に集中できる

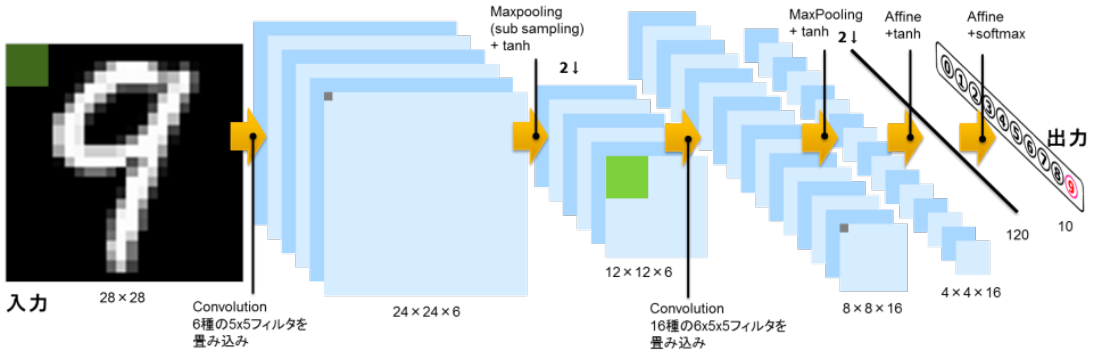


Neural Network **Libraries**

nnabla.org

C++で実装されたコンパクト、高速かつ移植性の高いコアと、利便性に優れたPython APIからなるライブラリ

研究者・開発者向けオープンソース（Apache2.0 License）プログラミングライブラリ




```python
# 自動微分用の変数コンテナ
from nnabla import Variable
# ニューラルネットワークの関数ブロック
import nnabla.functions as F
# パラメタ付きの関数ブロック
import nnabla.parametric_functions as PF

c1 = PF.convolution(image, 16, (5, 5), name='conv1')
c1 = F.relu(F.max_pooling(c1, (2, 2)), inplace=True)
c2 = PF.convolution(c1, 16, (5, 5), name='conv2')
c2 = F.relu(F.max_pooling(c2, (2, 2)), inplace=True)
c3 = F.relu(PF.affine(c2, 50, name='fc3'), inplace=True)
c4 = PF.affine(c3, 10, name='fc4')
```

設計・学習・推論（実行）ロジックをプログラミング（Python/C++）で柔軟記述

Neural Network Libraries/Consoleまとめ

優れたDeep Learningの開発環境を提供し、需要の急拡大するAI技術の普及・発展に貢献

Neural Network Libraries

https://nnabla.org/

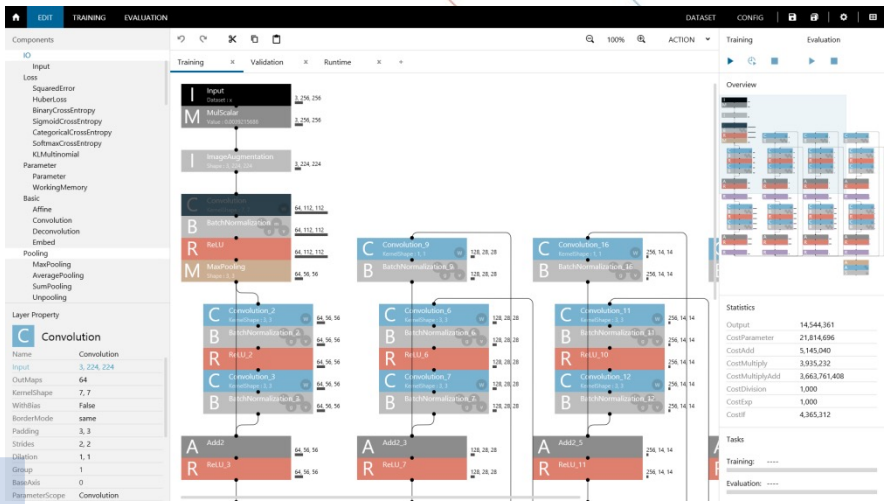
様々な特長を兼ね備えた最新世代のDeep Learningフレームワーク

```
import nnabla as nn
import nnabla.functions as F
import nnabla.parametric_functions as PF

x = nn.Variable(100)
t = nn.Variable(10)
h = F.tanh(PF.affine(x, 300, name='affine1'))
y = PF.affine(h, 10, name='affine2')
loss = F.mean(F.softmax_cross_entropy(y, t))
```

Neural Network Console

https://dl.sony.com/

商用クオリティのDeep Learning応用技術開発を実現する統合開発環境



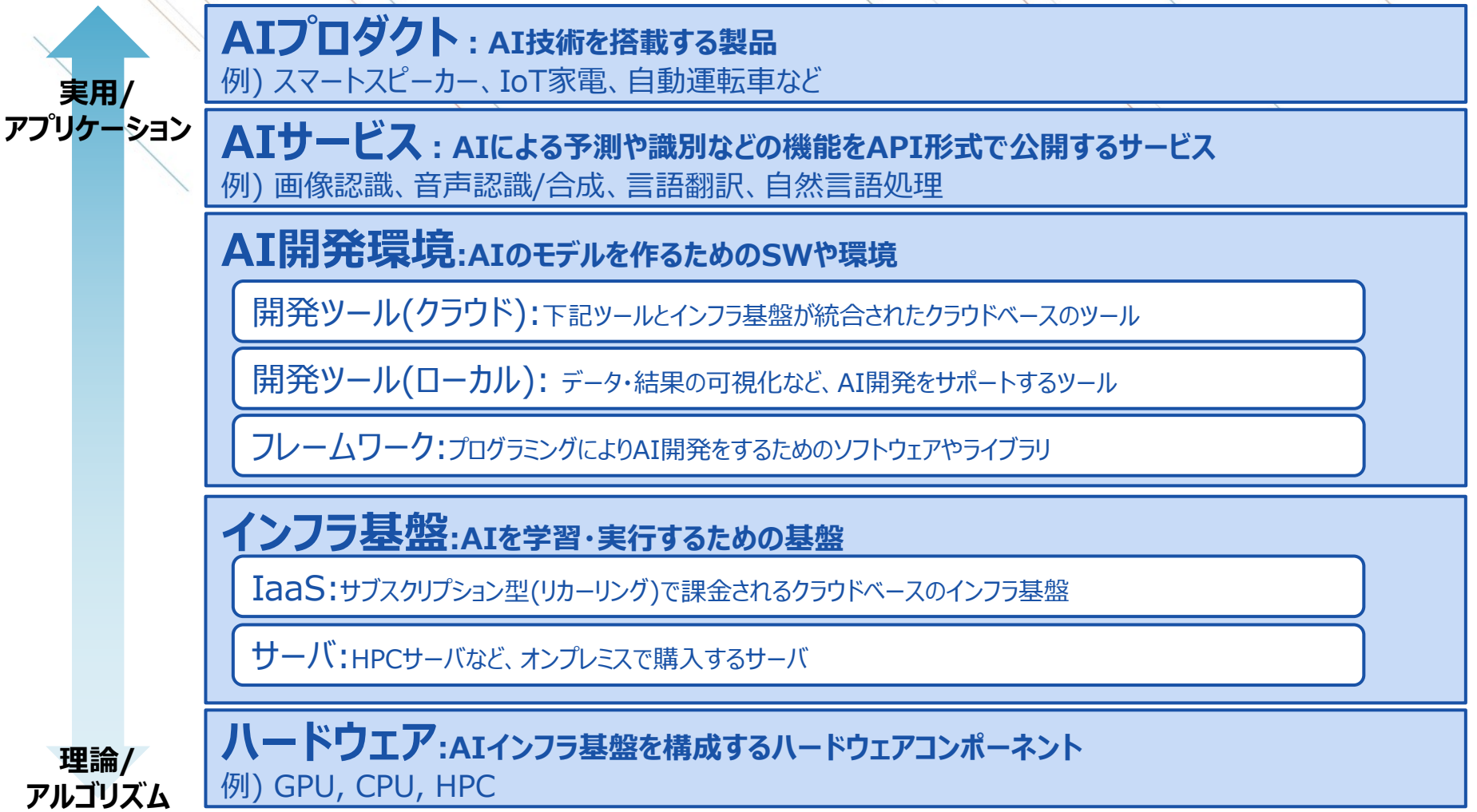
実現

- Deep Learning応用技術者の迅速な育成
- 効率的なDeep Learning応用技術の研究開発～実用化

ディープラーニングを用いたAI開発のポイント

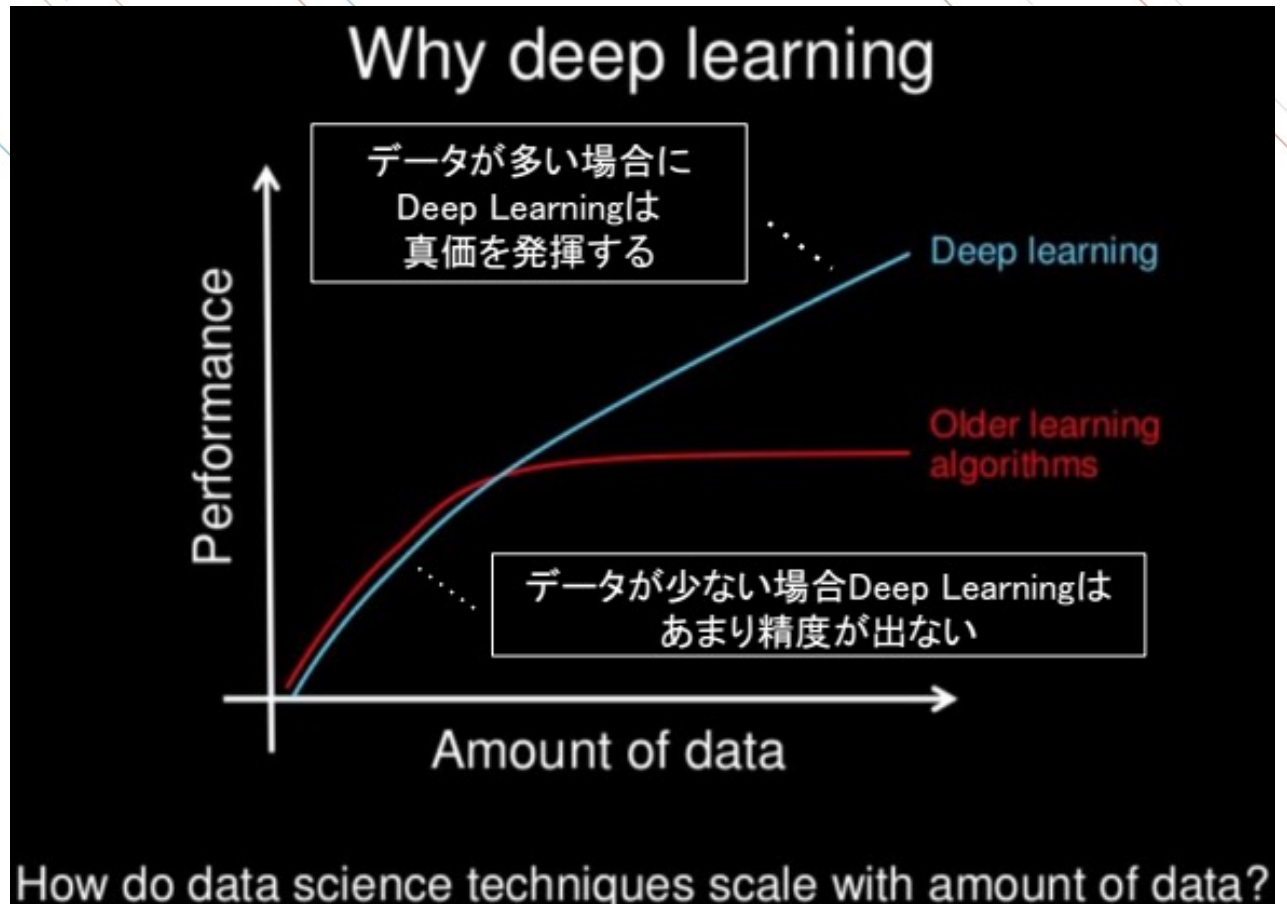
AI開発の構成レイヤー

開発したいサービスや開発スキルに応じて、既存のサービスなどと連携して開発を行う。



データ量の重要性

Deep Learningで高い精度を得るにはデータ量が重要である。
データ量が少ない場合には、既存の機械学習に比べて精度が劣ることがある。



出典: https://www.slideshare.net/ExtractConf/andrew-ng-chief-scientist-at-baidu

ラベル付け作業

データ収集に加えて、Deep Learningによって予測する値を事前に準備する必要がある。
ラベル付け作業には時間がかかるので、開発スケジュールを考える場合には注意が必要である。

画像に対するラベル付けの例

分類

→ “4”

→ “9”

回帰（定量化）

→ 0.9

→ 0.6

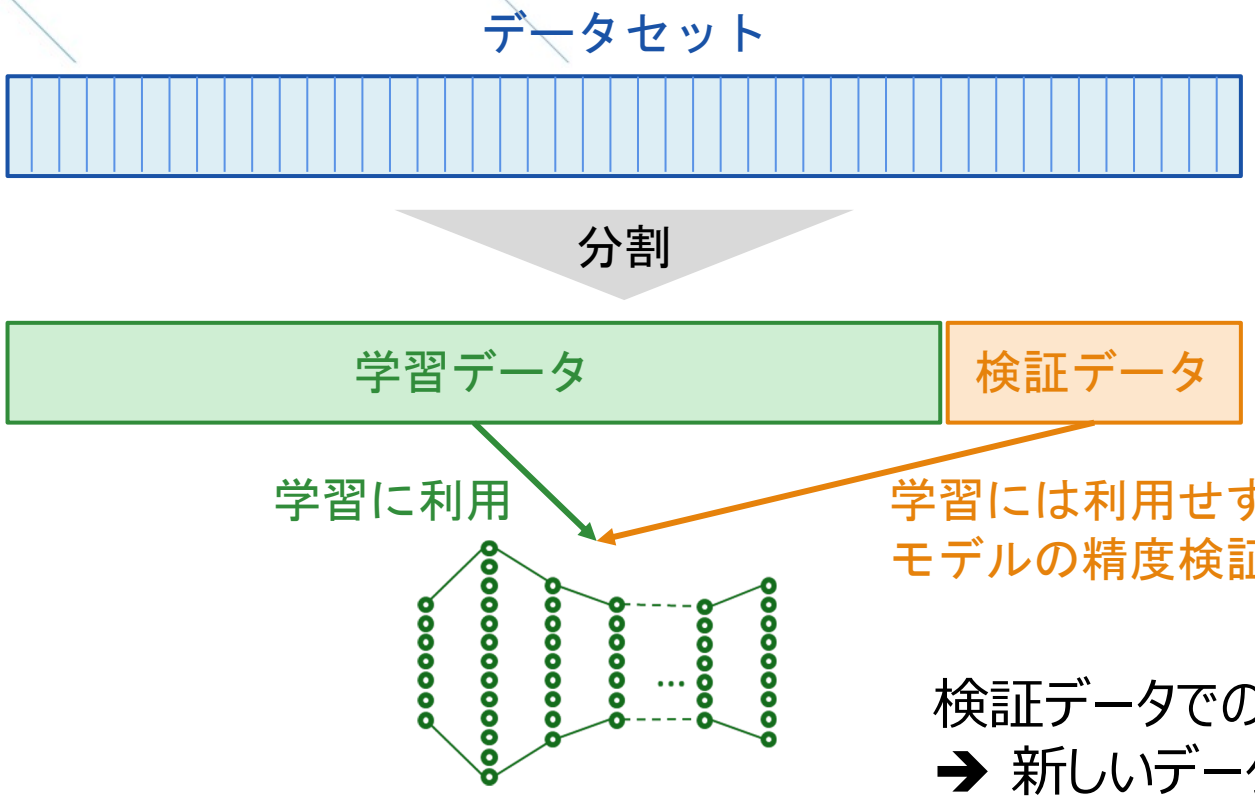
ディテクション

→

データセットの分割

モデルの精度を検証するために、データを学習に使用するデータ（学習データ、Training Data）と、使用しないデータ（検証データ、Validation Data）に分割する。

分割割合は7:3や8:2が一般的である。



検証データでの精度が高い
➔ 新しいデータに対しても高精度を期待できる

モデルの評価方法の検討

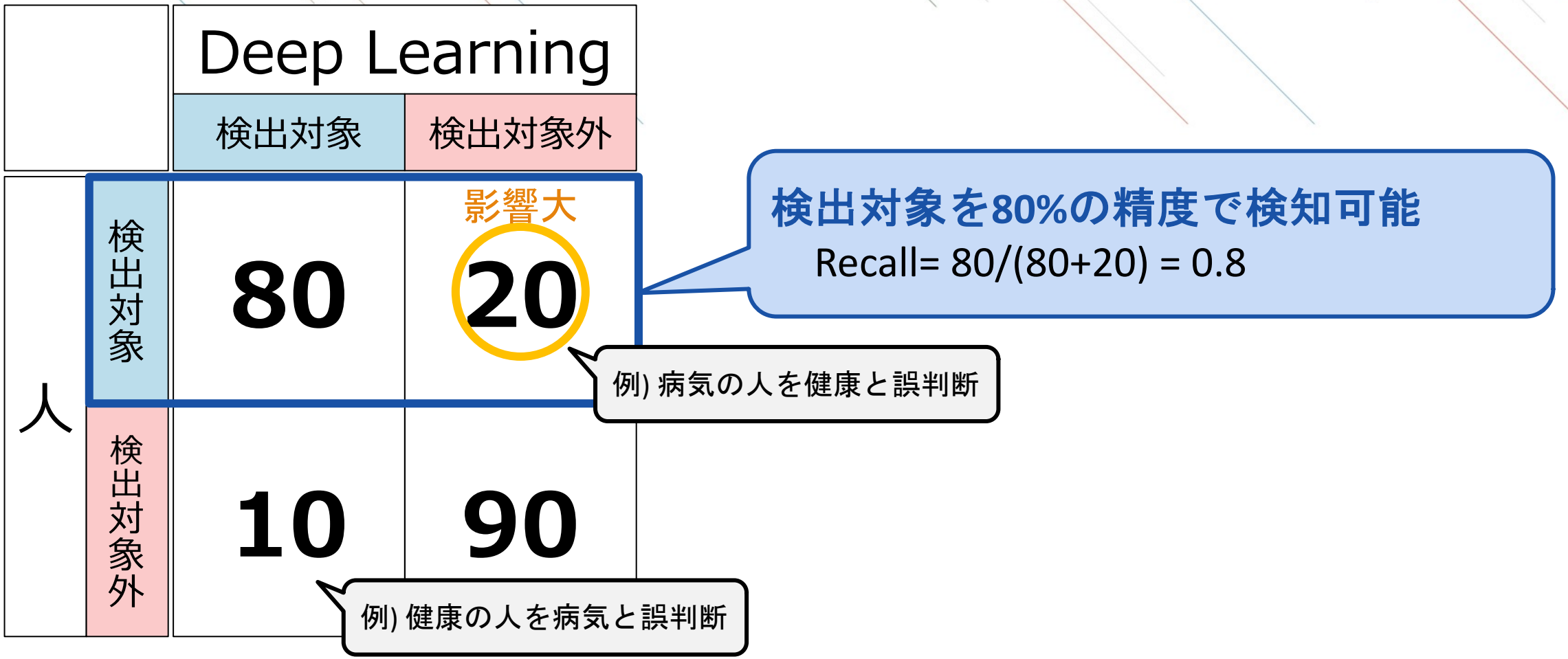
モデルの評価指標は複数存在し、その中で利用シーンに応じて適切な指数を選択する必要がある。
モデル開発には統計学などの分析技術だけでなく、ドメイン知識も必要になることが多い。

分類問題向けの評価指数の例

指標	説明	利用シーン例
Accuracy (正解率)	予測結果全体に対して答えが一致した割合 不均衡データでの利用には注意が必要	正例と負例の数が均衡している場合
Recall (再現率)	答えが正の中で、予測が正とされた割合	病気の発見など、正例を逃してはならない場合
Precision (適合率)	予測を正と判断した中で、答えも正のものである割合	迷惑メールフィルタなど、間違って正例としてしまい重要な情報の破棄が許されない場合
F-measure (F値)	PrecisionとRecallの調和平均 PrecisionとRecallは互いにトレードオフの関係にある　調和平均：率や比に用いる平均	分析要件から適合率、再現率のどちらか一方が重要だと決まらない場合

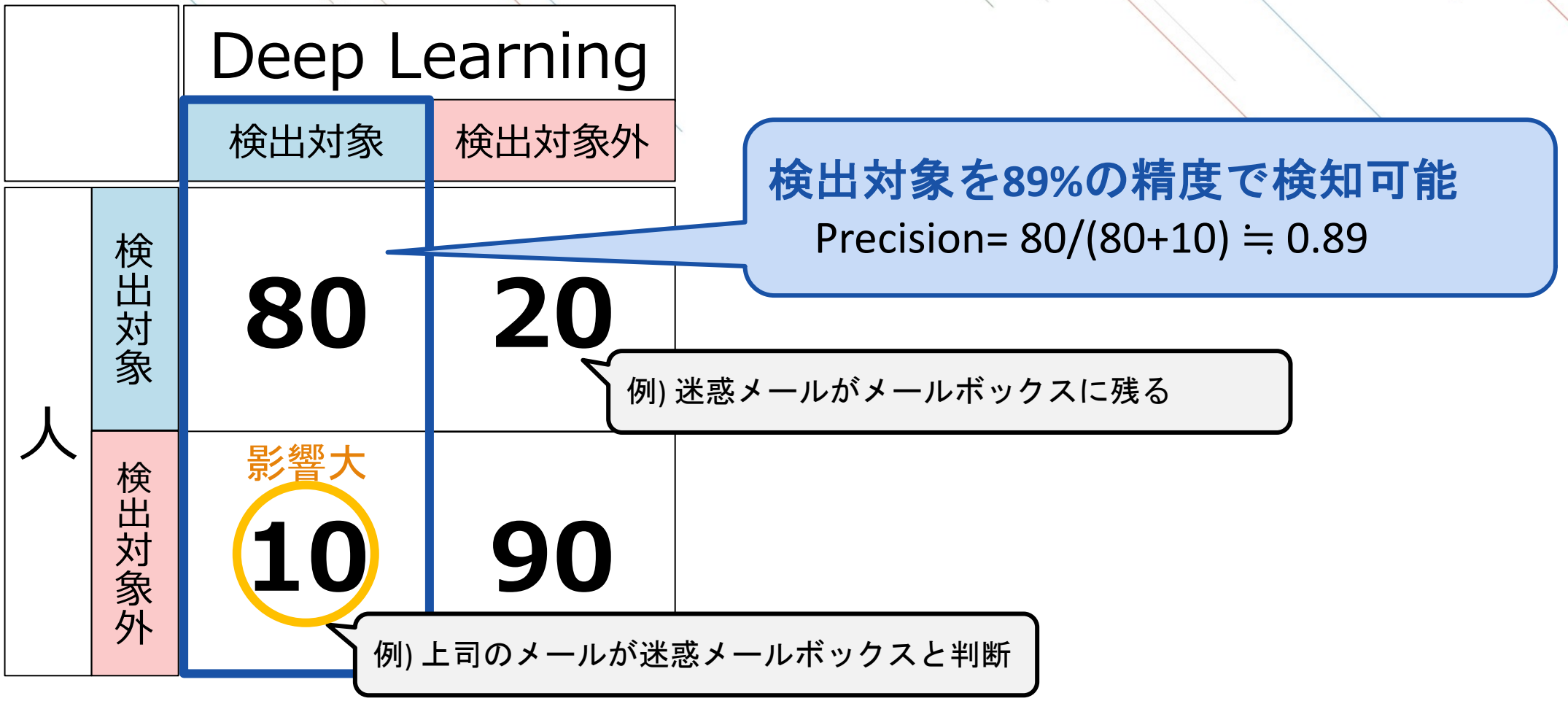
モデルの評価方法の検討: Recall（再現率）

答えが正の中で、予測が正とされた割合。病気の発見など、見落としの少なさの評価に用いる。



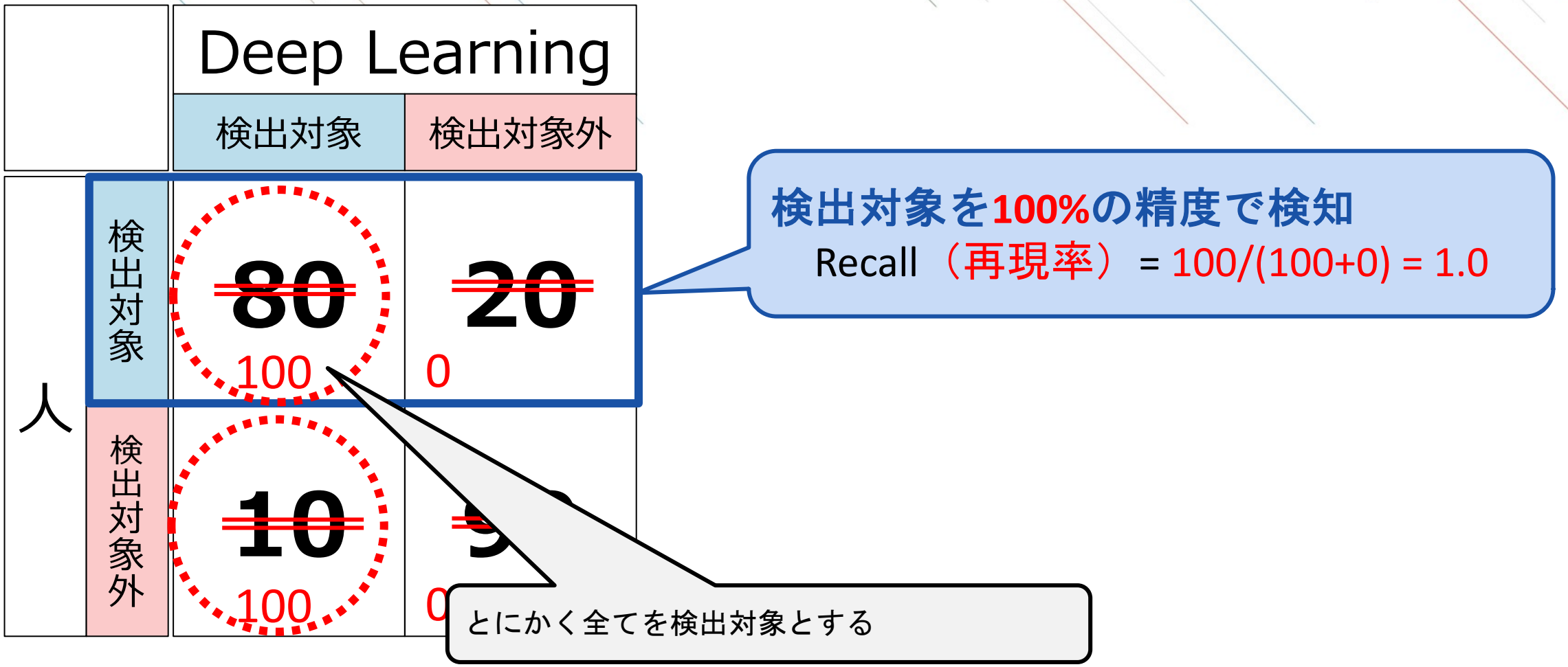
モデルの評価方法の検討: Precision（適合率）

予測を正と判断した中で、答えも正のものである割合。迷惑メールフィルタなど、誤検出の少なさの評価に用いる。



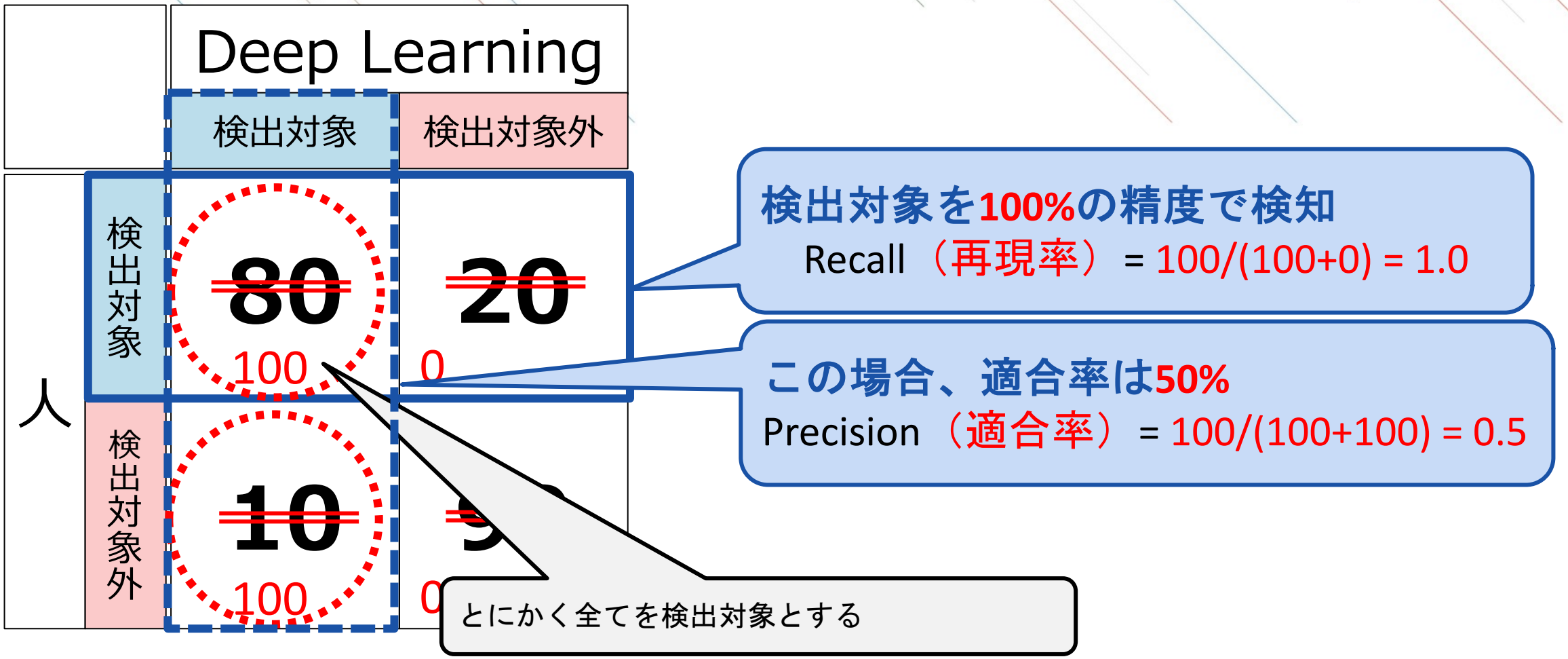
モデルの評価方法の検討: Recall（再現率）　— 再現率を高めるように設定 —

答えが正の中で、予測が正とされた割合。病気の発見など、見落としの少なさの評価に用いる。



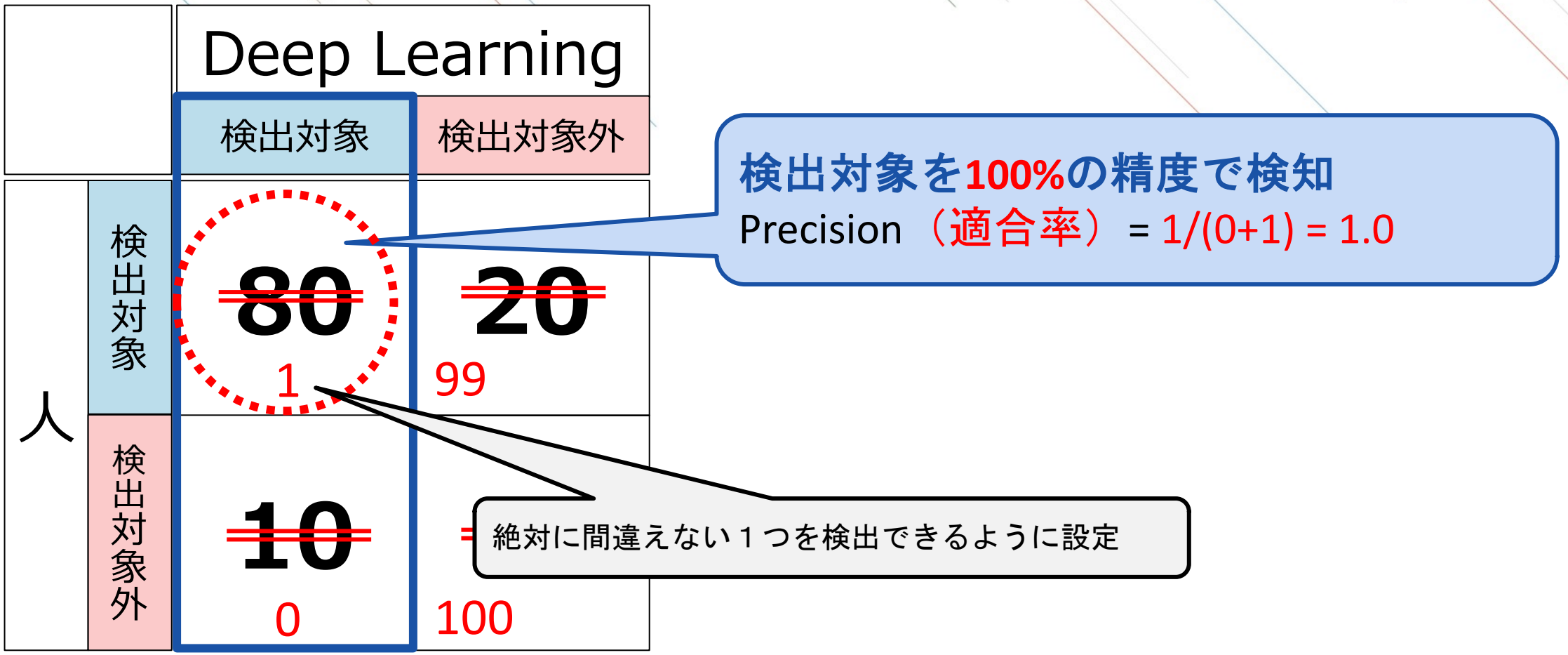
モデルの評価方法の検討: Recall（再現率）　— 再現率を高めるように設定 —

答えが正の中で、予測が正とされた割合。病気の発見など、見落としの少なさの評価に用いる。



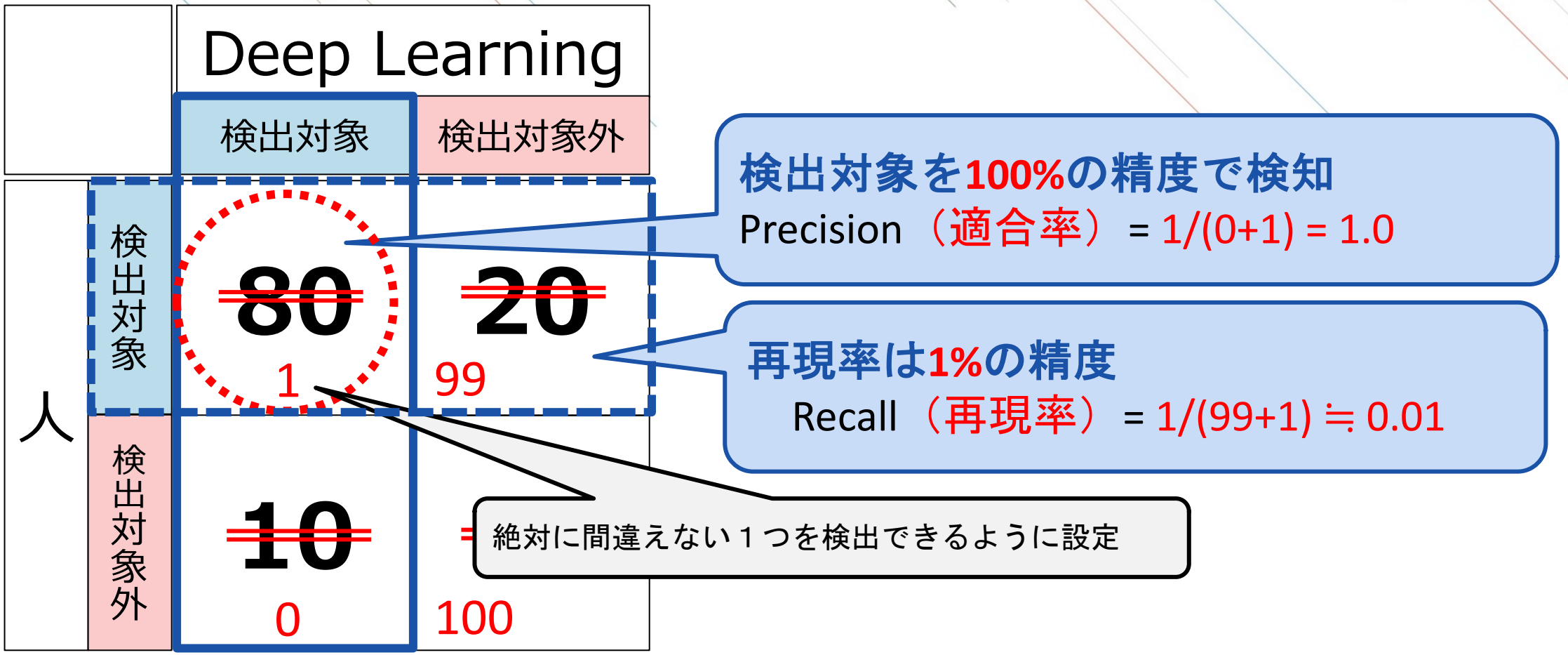
モデルの評価方法の検討: Precision（適合率）― 適合率を高めるように設定 ―

予測を正と判断した中で、答えも正のものである割合。迷惑メールフィルタなど、誤検出の少なさの評価に用いる。



モデルの評価方法の検討: Precision（適合率）― 適合率を高めるように設定 ―

予測を正と判断した中で、答えも正のものである割合。迷惑メールフィルタなど、誤検出の少なさの評価に用いる。



モデルの評価方法の検討: F-measure (F値)

Precision（適合率）とRecall（再現率）の調和平均

※調和平均：逆数の算術平均の逆数

		Deep Learning	
		検出対象	検出対象外
人	検出対象	80	20
	検出対象外	10	90

検出対象を84%の精度で検知

$$\text{F-measure} = \frac{2}{\frac{1}{0.8}+\frac{1}{0.89}} \fallingdotseq 0.84$$

無理に再現率を高める設定

$$\text{F-measure} = \frac{2}{\frac{1}{1}+\frac{1}{0.5}} \fallingdotseq 0.67$$

無理に適合率を高める設定

$$\text{F-measure} = \frac{2}{\frac{1}{1}+\frac{1}{0.01}} \fallingdotseq 0.02$$

十分な精度が出ない場合の要因分析

モデル精度に効いてくる要因は、「データ」と「モデル」がある。
どちらが根本的な原因かを判断するのは難しい。

モデル精度が悪い場合に想定される原因

データ

- データ量が不十分である
- データの質が悪い

モデル

- モデルのネットワーク構造が悪い
- チューニングが不十分である

一般的にどちらが原因であるかを判断するのは難しい

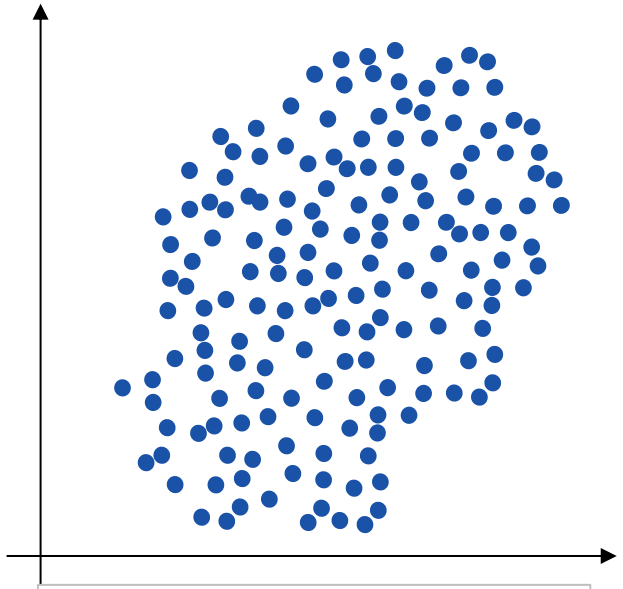
※認識器、識別器、モデル：機械学習で「犬」「ねこ」などのように判別をするしくみ

データセットに問題がある場合

モデル作成時のデータセットに偏りがあると精度が出ないことが多い。
モデル作成前のデータの事前分析が重要になってくる。

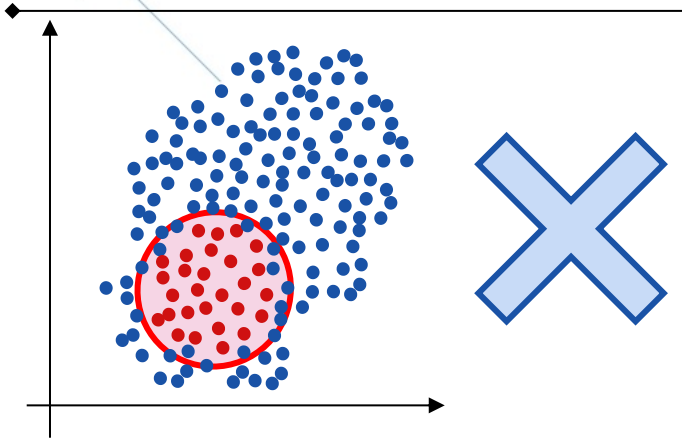
真のデータ分布

概念図

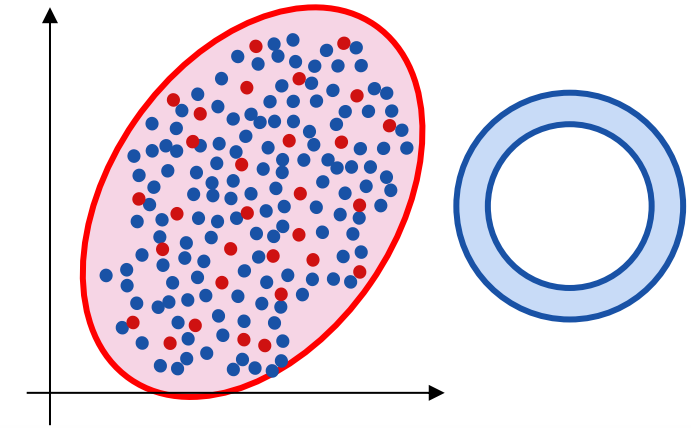


未来過去含め、現実で起こり得る全ての事象の分布

モデル作成時のデータ分布



データが網羅できていないため、精度が出ない
（一般に外挿は精度が低い）

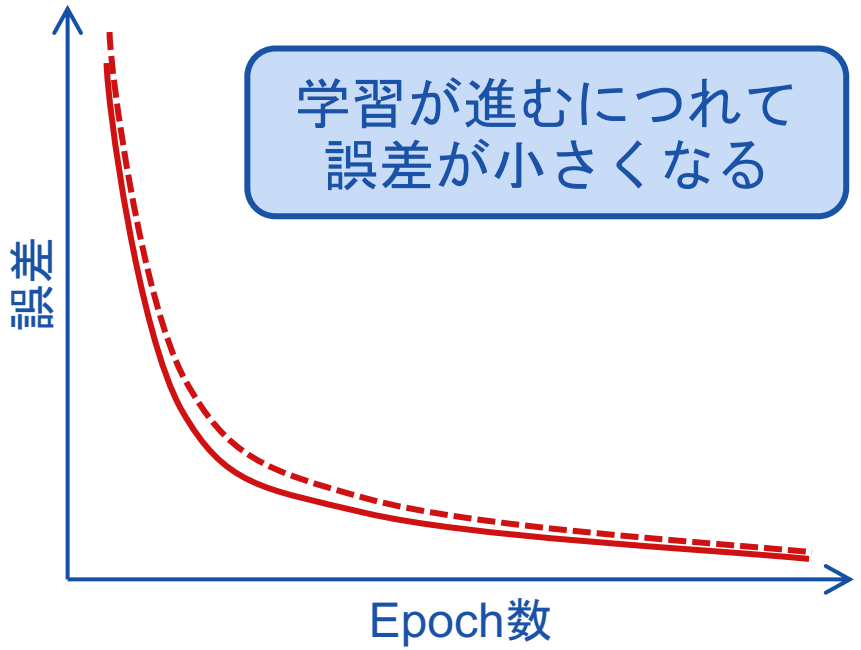


データ量は少ないが、分布が空間を網羅しているため、よい精度が期待できる

モデルに問題がある場合（過学習）

学習の過程でモデルが学習データに特化し過ぎる過学習に注意する必要がある。
学習曲線を確認し、モデルが過学習でないかどうかは常に確認しなければいけない。

よいモデルの学習曲線



悪いモデルの学習曲線（過学習）