

# ラピッドチャレンジ「課題レポート(深層学習 Day4)」

佐藤晴一

2021年6月19日

# 目次

<b>第 1 章</b>	<b>強化学習</b>	3
1.1	強化学習とは . . . . .	3
1.2	強化学習の応用例 . . . . .	4
1.3	探索と利用のトレードオフ . . . . .	4
1.4	強化学習のイメージ . . . . .	5
1.5	強化学習の差分 . . . . .	5
1.6	強化学習の歴史 . . . . .	5
1.7	価値関数 . . . . .	5
1.8	方策関数 . . . . .	6
1.9	方策勾配法 . . . . .	6
<b>第 2 章</b>	<b>Alpha Go</b>	8
<b>第 3 章</b>	<b>軽量化・高速化技術</b>	11
3.1	データ並列化 . . . . .	11
3.2	モデル並列化 . . . . .	12
3.3	GPU . . . . .	12
3.4	量子化 (Quantization) . . . . .	13
3.5	蒸留 (Distillation) . . . . .	13
3.6	プルーニング (Pruning) . . . . .	14
3.7	モデル軽量化まとめ . . . . .	15
<b>第 4 章</b>	<b>応用技術</b>	16
4.1	MobileNet . . . . .	16
4.2	DenseNet . . . . .	17
4.3	Layer 正規化/Instance 正規化 . . . . .	19
4.4	WaveNet . . . . .	19
<b>第 5 章</b>	<b>Transformer</b>	22
5.1	Seq2Seq . . . . .	22
5.2	Transformer . . . . .	23
<b>第 6 章</b>	<b>物体検知・セグメンテーション</b>	26



# 第1章

## 強化学習

強化学習 (Reinforcement Learning, RL) とは、システム自身が試行錯誤しながら、最適なシステム制御を実現する、機械学習手法のひとつ。強化学習という概念自体は、昨今の AI ブームよりかなり前から存在します。強化学習の原型は、機械の自律的制御を可能にする「最適制御」の研究として、1950 年代には既に存在しました。1990 年頃には「強化学習の父」とも呼ばれるカナダ・アルバータ大学のリチャード・サットン教授らを中心に、活発に研究されていた。

古くから存在した強化学習に、飛躍的な技術進展をもたらしたのが「深層強化学習」です。従来の強化学習にディープラーニングを応用した深層強化学習の登場は、強化学習をベースに駆動する AI が社会に実装される契機となった。

### 1.1 強化学習とは

長期的に報酬を最大化できるように環境のなかで行動を選択できるエージェントを作ることを目標とする機械学習の一分野である。つまり、行動の結果として与えられる利益 (報酬) をもとに、行動を決定する原理を改善していく仕組みのことである。

■強化学習と機械学習 機械学習 (Machine Learning, ML) とは、人工知能のプログラムが大量のデータを学習し、識別や予測を行うアルゴリズムを自動で構築する技術の全般を指します。その機械学習の中のひとつの領域が「強化学習」であり、その「強化学習」に「教師あり学習」、「教師なし学習」のふたつを加えた 3 つの領域が機械学習には存在する<sup>\*1</sup>。

---

<sup>\*1</sup> <https://mse238blog.stanford.edu/2017/07/choftun/the-building-blocks-of-machine-learning/>

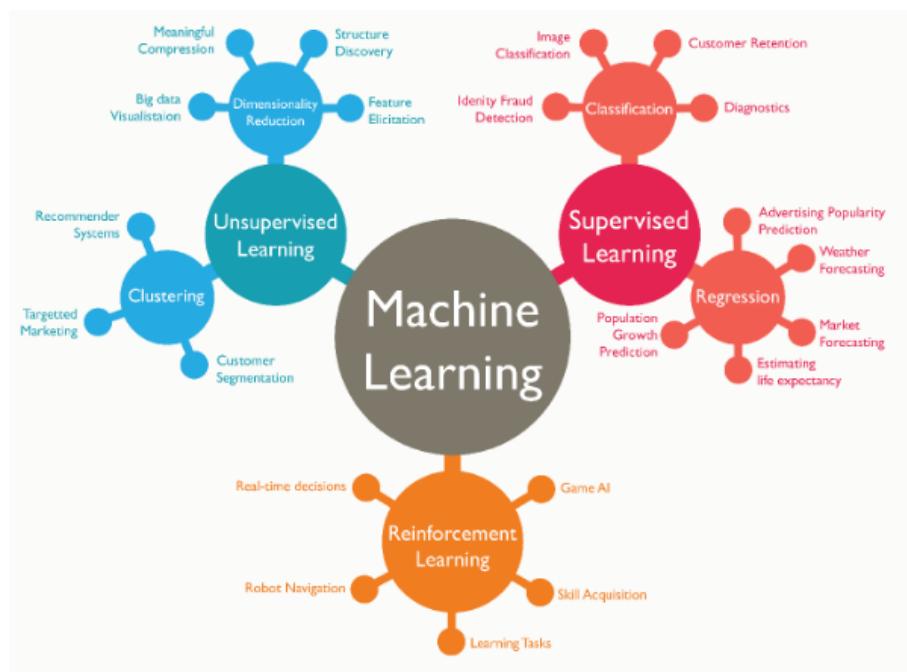


図 1.1 機械学習の全体像

## 1.2 強化学習の応用例

### ■マーケティングの場合

- 環境 Q 学習 … 会社の販売促進部
- エージェント … プロフィールと購入履歴に基づいて、キャンペーンメールを送る顧客を決めるソフトウェアである
- 行動 … 顧客ごとに送信、非送信のふたつの行動を選ぶことになる。
- 報酬 … キャンペーンのコストという負の報酬とキャンペーンで生み出されると推測される売上という正の報酬を受ける

## 1.3 探索と利用のトレードオフ

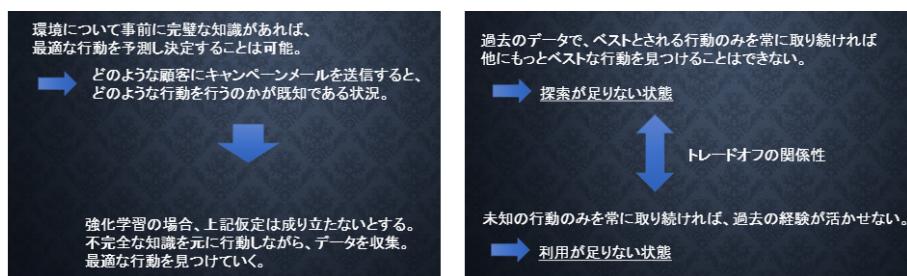


図 1.2 探索と利用のトレードオフ

## 1.4 強化学習のイメージ

方策には、方策関数。価値には、行動価値関数を与える。この両者を学習させることになる。

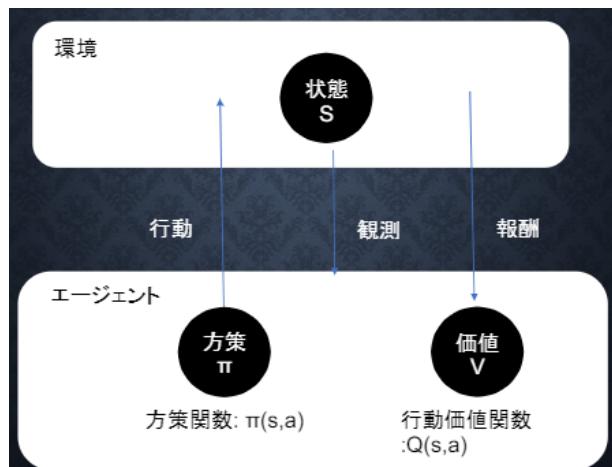


図 1.3 強化学習のイメージ

## 1.5 強化学習の差分

強化学習と通常の教師あり、教師なし学習との違いとは何か？これは、「目標」が違うと言いかれる。

1. 教師なし、あり学習 … データに含まれるパターンを見つけ出すおよびそのデータから予測することが目標
2. 強化学習 … 優れた方策を見つけることが目標（よりよく動く行動指針を探す ※ 特徴量を見つけるということではない）

## 1.6 強化学習の歴史

強化学習について・冬の時代があったが、計算速度の進展により大規模な状態をもつ場合の、強化学習を可能としつつある。具体的には、関数近似法と、Q学習を組み合わせる手法の登場である。

1. Q学習 … 行動価値関数を、行動する毎に更新することにより学習を進める方法
2. 関数近似法 … 価値関数や方策関数を関数近似する手法のこと

## 1.7 価値関数

エージェントの価値（目標設定）を表す関数としては、「状態価値関数」と「行動価値関数」の2種類がある。最近は、後者の行動価値関数がQ学習としてよく採用されて用いられている。

1. 状態価値関数 … ある環境の状態の価値に注目する場合 ※ エージェントの方策・価値には無関係
2. 行動価値関数 (Q 学習) … 状態と価値を組み合わせた価値に注目する場合 ※ エージェントの行動に依存

## 1.8 方策関数

方策ベースの強化学習手法において、ある状態でどのような行動を探るのかの確率を与える関数のこと。囲碁の Alpha Go でいうと、次の一手をどうするか考えるのが「方策関数」で、この方策を取り続けた場合、価値（勝率）がどうなるか予測する部分を「価値関数」という。

### ■方策関数

$$\pi(s, a|\theta)$$

ここで、 $\theta$  は方策関数のパラメータ。

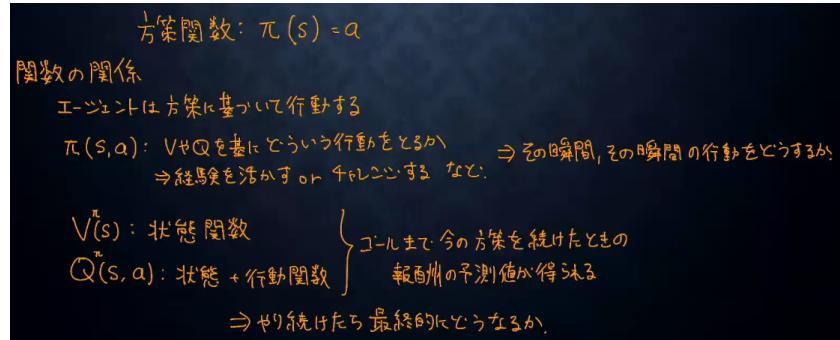


図 1.4 方策関数

## 1.9 方策勾配法

方策をモデル化して最適化する手法

$$\theta^{(t+1)} = \theta^{(t)} + \epsilon \nabla J(\theta)$$

ここで、 $J(\theta)$  は、「方策の良さ」を意味しており定義する必要がある（ニューラルネットワークでいうところの誤差関数に相当）。これまで誤差を「小さく」という考え方であったが、強化学習では期待値を「最大化」するように更新する。

このように定式化（関数）で表現できるようになるとニューラルネットワークの概念が適用できるようになる。つまり、方策関数を NN として学習させれば良い。今までのニューラルネットワークでは、重み  $w$  で表現していた部分が  $\theta$  で表記されている。

### ■定義方法 平均報酬・割引報酬和

**■方策勾配定理** 上記の定義に対応して、行動価値関数:  $Q(s, a)$  の定義を行い、方策勾配定理が成り立つ<sup>\*2</sup>。方策勾配定理とは、Q 値を用いて、累積報酬を増加させる方策の勾配を求めるための定理。つまり、この

<sup>\*2</sup> <http://rail.eecs.berkeley.edu/deeprlcourse-fa17/>

勾配を用いることで、累積報酬の向上を目指して方策を更新できます。方策勾配定理によって得られる勾配  $\nabla J(\theta)$  は次式で与えられる。この式のおかげで、これまで学習してきた Q 値を用いて累積報酬値を向上させることができます。

$$\nabla_{\theta}J(\theta) = \mathbb{E}_{\pi_{\theta}}[(\nabla_{\theta}\log\pi_{\theta}(a|s)Q^{\pi}(s,a))]$$

- $s$ : 状態
- $a$ : 行動
- $Q$ : Q 値
- $\theta$ : 方策  $\pi$  のパラメータ

図 1.5 方策勾配定理

せる方向の勾配を求めることができる。

## 第2章

# Alpha Go

AlphaGo は、Google DeepMind によって開発されたコンピュータ囲碁プログラムである。2015年10月に、人間のプロ囲碁棋士を互先（ハンディキャップなし）で破った初のコンピュータ囲碁プログラムとなった。

コンピュータが人間に打ち勝つことが最も難しいと考えられてきた分野である囲碁において、人工知能が勝利を収めたことは世界に衝撃をもたらした。AlphaGo の登場は単なる一競技の勝敗を越え、人工知能の有用性を広く知らしめるものとなり、世界的 AI ブームを呼び起こすきっかけともなった。

1. AlphaGo (Lee) … David Silver Mastering the game of Go with deep neural networks and tree search nature 27 January 2016 <https://www.nature.com/articles/nature16961>
2. 強化学習 … AlphaGo Zero David Silver Mastering the game of Go without human knowledge nature 18 October 2017 <https://www.nature.com/articles/nature24270Section2>

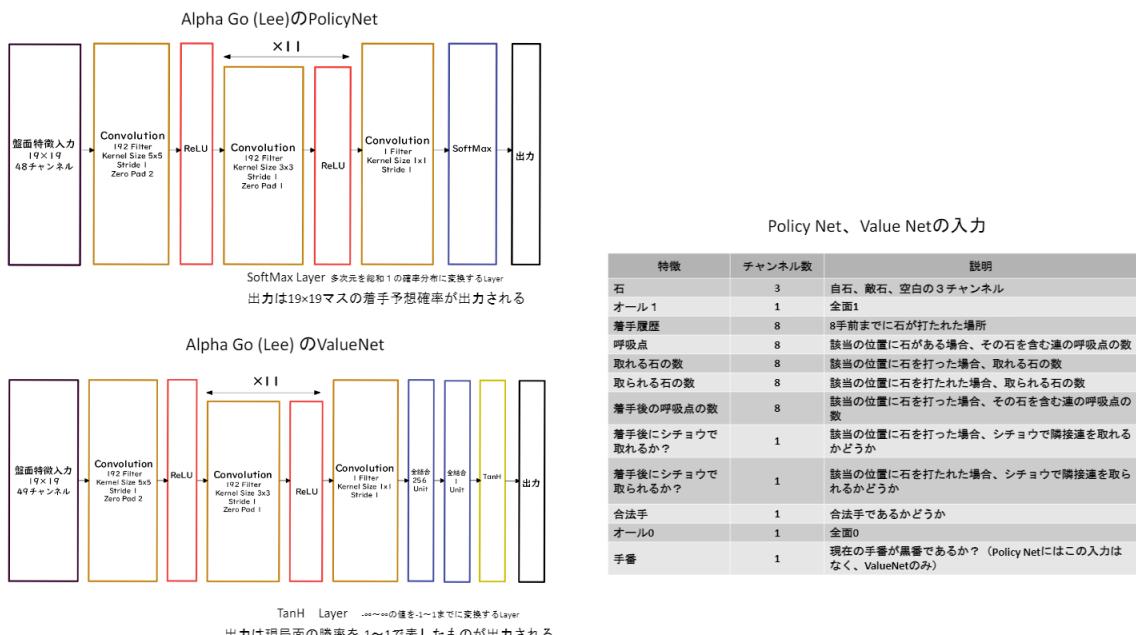


図 2.1 AlphaGo における方策関数と行動価値関数

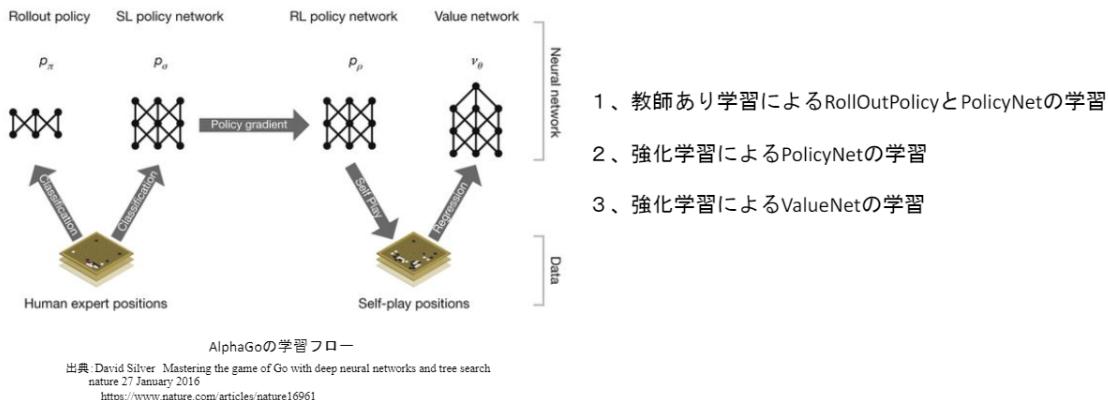


図 2.2 AlphaGo における学習ステップ

■AlphaGo(Lee) における学習ステップ（約精度 57%）をしてから強化学習の phase に移っている点がポイントはある。

#### ■AlphaGo(Lee) と AlphaGoZero の違い

1. 教師あり学習を一切行わず、強化学習のみで作成
2. 強化入力からヒューリスティックな要素を排除し、石の配置のみにした
3. PolicyNet と ValueNet を 1 つのネットワークに統合した
4. Residual Net（後述）を導入した
5. モンテカルロ木探索から RollOut シミュレーションをなくした

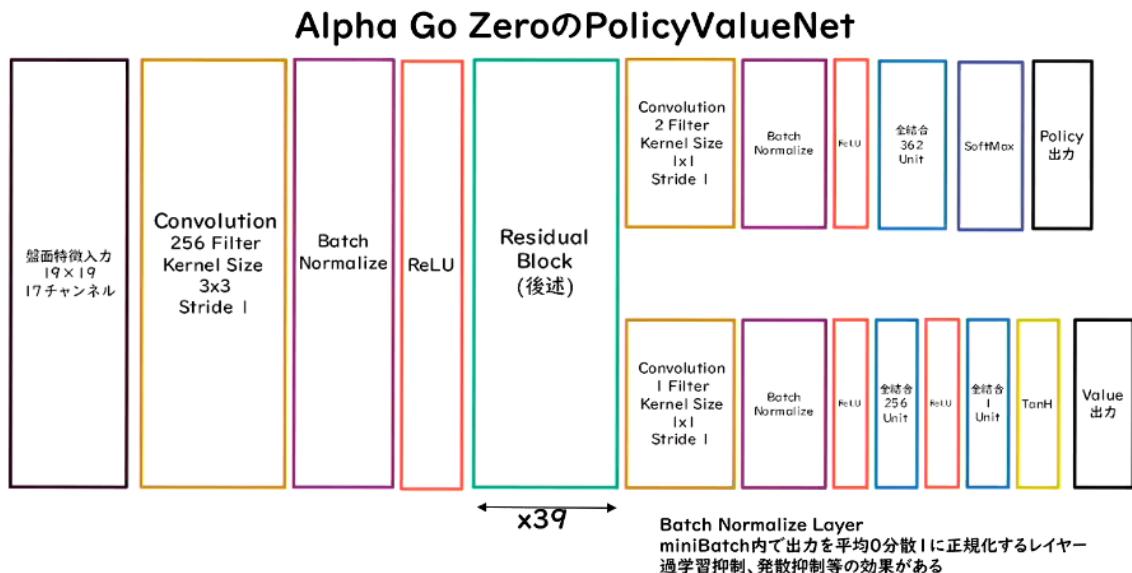


図 2.3 AlphaGo (Zero)

## ■AlphaGo(Zero)

■Residual Network ネットワークにショートカット構造を追加して、勾配の爆発、消失を抑える効果を狙ったもの Residual Network を使うことにより、100 層を超えるネットワークでの安定した学習が可能となった。基本構造は Convolution → BatchNorm → ReLU → Convolution → BatchNorm → Add → ReLU の Block を 1 単位にして積み重ねる形となる。さらに、Residual Network を使うことにより層数の違う Network のアンサンブル効果が得られているという説もある。

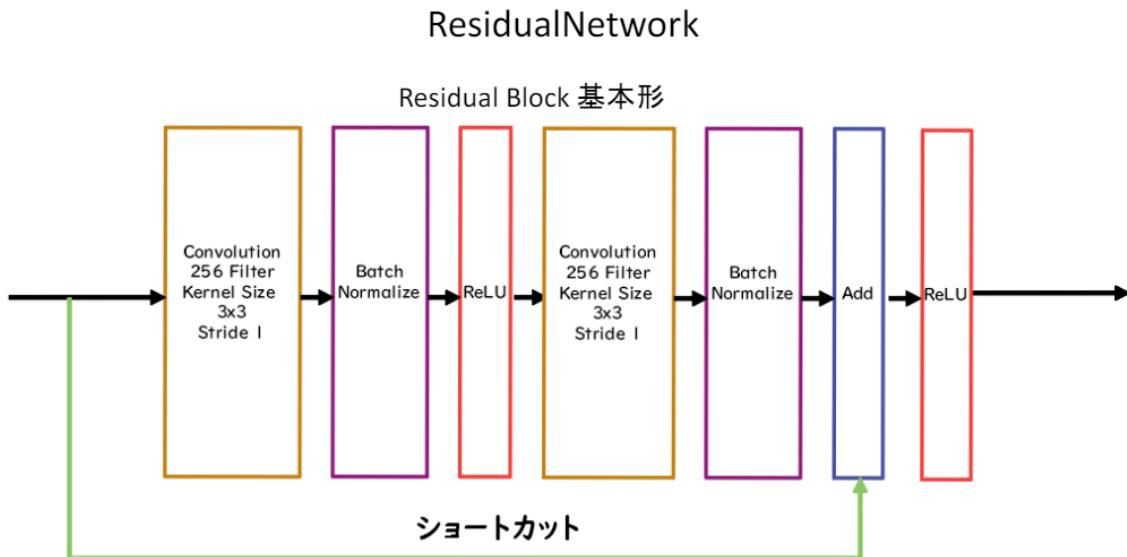


図 2.4 Residual Network

## ■AlphaGo(Zero) の学習ステップ

### 1. 自己対局による教師データの作成

現状のネットワークでモンテカルロ木探索を用いて自己対局を行う。まず 30 手までランダムで打ち、そこから探索を行い勝敗を決定する。自己対局中の各局面での着手選択確率分布と勝敗を記録する。教師データの形は（局面、着手選択確率分布、勝敗）が 1 セットとなる。

### 2. 学習

自己対局で作成した教師データを使い学習を行う。Network の Policy 部分の教師に着手選択確率分布を用い、Value 部分の教師に勝敗を用いる。損失関数は Policy 部分は CrossEntropy、Value 部分は平均二乗誤差。特徴入力からヒューリスティックな要素を排除し、石の配置のみにした

### 3. ネットワークの更新

学習後、現状のネットワークと学習後のネットワークとで対局テストを行い、学習後のネットワークの勝率が高かった場合、学習後のネットワークを現状のネットワークとする。

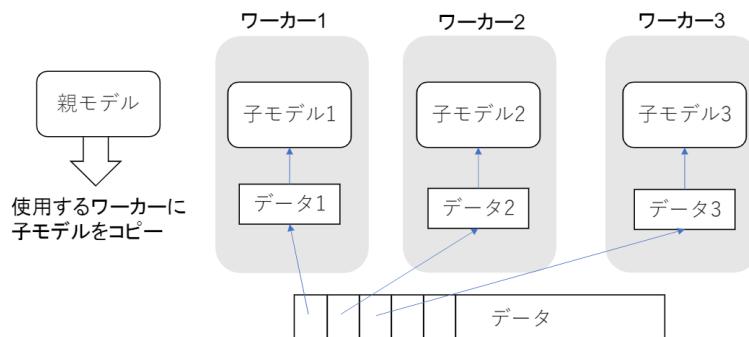
## 第3章

# 軽量化・高速化技術

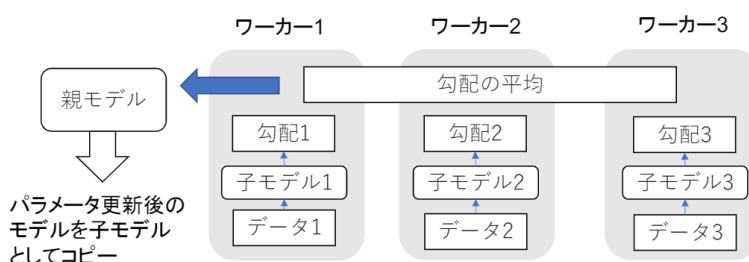
分散深層学習とは・深層学習は多くのデータを使用したり、パラメータ調整のために多くの時間を使用したりするため、高速な計算が求められる。複数の計算資源（ワーカー）を使用し、並列的にニューラルネットを構成することで、効率の良い学習を行いたい。データ並列化、モデル並列化、GPUによる高速技術は不可欠である。

### 3.1 データ並列化

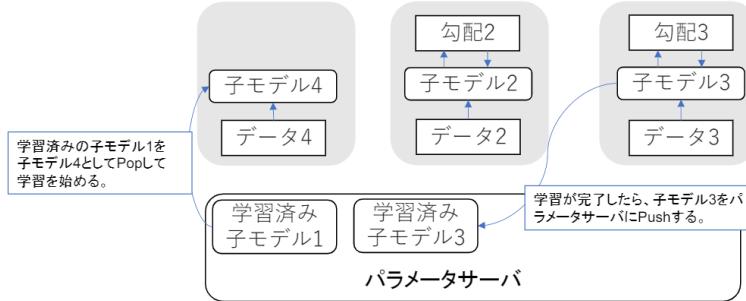
親モデルを各ワーカーに子モデルとしてコピー。次いで、データを分割し、各ワーカーごとに計算させる。このデータ並列化は、各モデルのパラメータの合わせ方で「同期型」か「非同期型」か決まる。



■データ並列化：同期型 同期型のパラメータ更新の流れ。各ワーカーが計算が終わるのを待ち、全ワーカーの勾配が出たところで勾配の平均を計算し、親モデルのパラメータを更新する。



**■データ並列化：非同期型** 非同期型のパラメータ更新の流れ。各ワーカーはお互いの計算を待たず、各子モデルごとに更新を行う。学習が終わった子モデルはパラメータサーバに Push される。新たに学習を始める時は、パラメータサーバから Pop したモデルに対して学習していく。



### ■同期型と非同期型の比較

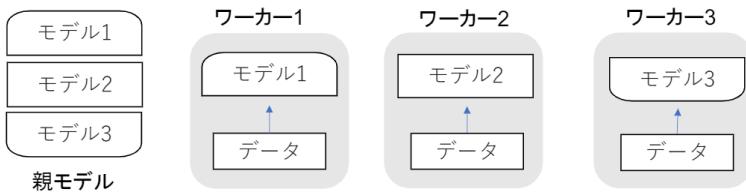
1. 処理のスピードは、お互いのワーカーの計算を待たない非同期型の方が早い。
2. 非同期型は最新のモデルのパラメータを利用できないので、学習が不安定になりやすい。-; Stale Gradient Problem。
3. 現在は同期型の方が精度が多いことが多いので、主流となっている。

## 3.2 モデル並列化

親モデルを各ワーカーに分割し、それぞれのモデルを学習させる。全てのデータで学習が終わった後で、一つのモデルに復元する。モデルが大きい時はモデル並列化を、データが大きい時はデータ並列化をすると良い。

一般に、データ並列化の際は複数の PC を用いて、モデル並列化の際は、1 台の PC でその中で複数の GPU を用いて計算を行う。その理由は、各モデルで計算した出力を用いて誤差関数を計算する際のデータの送受のやりとりに時間がかかるのを防ぐため（1 台の PC で扱っている）。

当たり前であるが、「モデルのパラメータ数が多いほど、スピードアップの効率も向上する。」



## 3.3 GPU

### ■GPU による高速化

1. GPGPU (General-purpose on GPU)

元々の使用目的であるグラフィック以外の用途で使用される GPU の総称

2. CPU
  - (a) 高性能なコアが少数
  - (b) 複雑で連続的な処理得意
3. GPU
  - (a) 比較的低性能なコアが多数
  - (b) 簡単な並列処理得意
  - (c) ニューラルネットの学習は単純な行列演算が多いので、高速化可能

### ■GPGPU 開発環境

1. CUDA
  - (a) GPU 上で並列コンピューティングを行うためのプラットフォーム
  - (b) NVIDIA 社が開発している GPU のみで使用可能。
  - (c) Deep Learning 用に提供されているので、使いやすい
2. OpenCL
  - (a) オープンな並列コンピューティングのプラットフォーム
  - (b) NVIDIA 社以外の会社 (Intel, AMD, ARM など) の GPU からでも使用可能。
  - (c) Deep Learning 用の計算に特化しているわけではない。

### サマリ

Deep Learning フレームワーク (Tensorflow, Pytorch) 内で実装されているので、使用する際は指定すれば良い。

## 3.4 量子化 (Quantization)

ネットワークが大きくなると大量のパラメータが必要になり学習や推論に多くのメモリと演算処理が必要となる。このため、通常のパラメータの 64 bit 浮動小数点を 32 bit など下位の精度に落とすことでメモリと演算処理の削減を行うことで軽量化が実現できる。

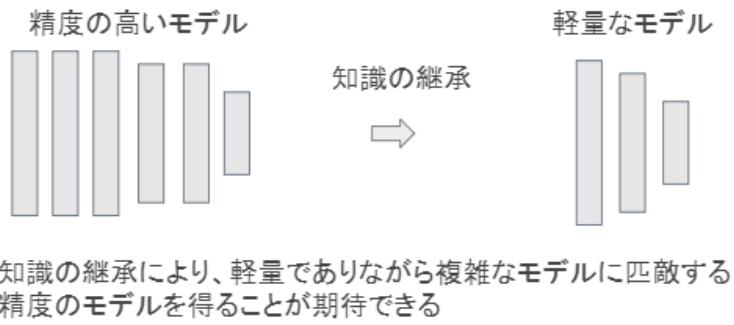
### ■量子化の利点と欠点

1. 利点 … 計算の高速化。省メモリ化
2. 欠点 … 欠点精度の低下 ※ただし、実際のところは、倍精度を单精度にしてもほぼ精度は変わらない。

## 3.5 蒸留 (Distillation)

精度の高いモデルはニューロンの規模が「大きなモデル」となっているため、推論に多くのメモリと演算処理が必要となる。蒸留は、規模の大きなモデルの知識を使って「軽量なモデルの作成」を行う手法である。

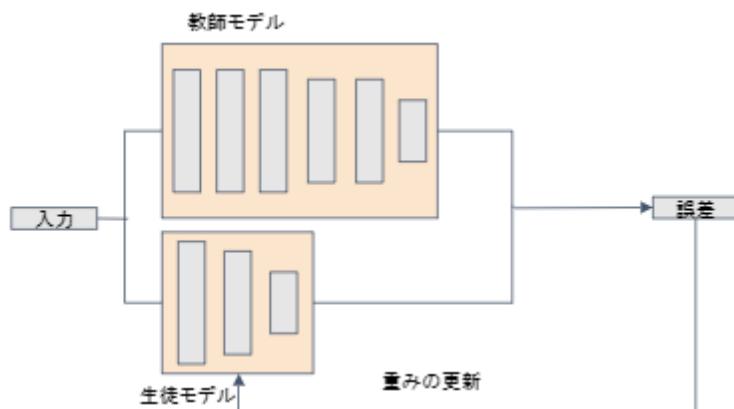
### ■モデルの簡約化 学習済みの精度の高いモデルの知識を軽量なモデルへ継承させる。



■教師モデルと生徒モデル 蒸留は教師モデルと生徒モデルの2つで構成される<sup>\*1</sup>。

1. 教師モデル … 予測精度の高い、複雑なモデルやアンサンブルされたモデル
2. 生徒モデル … 教師モデルをもとに作られる軽量なモデル

■生徒モデルの学習要領 教師モデルの重みを固定し生徒モデルの重みを更新していく。誤差は教師モデルと生徒モデルのそれぞれの誤差を使い重みを更新していく。



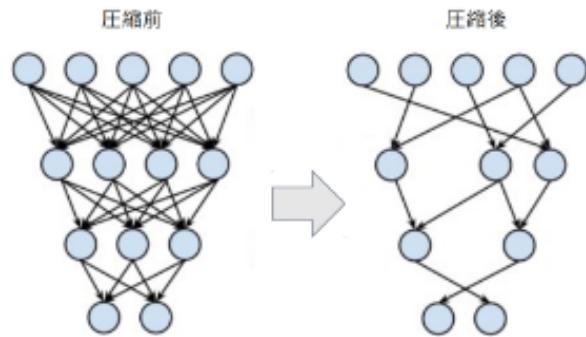
## 3.6 プルーニング (Pruning)

ネットワークが大きくなると大量のパラメータになるが、全てのニューロンの計算が精度に寄与しているわけではない。そこで、モデルの精度に寄与が少ないニューロンを削減することでモデルの軽量化、高速化が見込まれる。

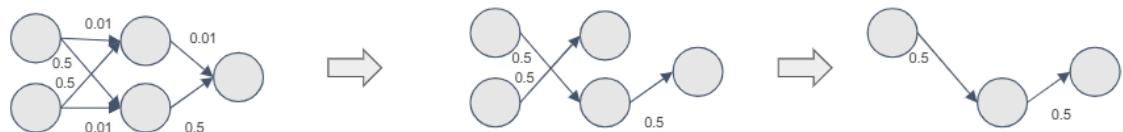
■計算の高速化 寄与の少ないニューロンの削減を行いモデルの圧縮を行うことで計算の高速化が実現できる。

---

<sup>\*1</sup> Geoffrey Hinton, Oriol Vinyals, Jeff Dean (2015) 「Distilling the Knowledge in a Neural Network」, <https://arxiv.org/abs/1503.02531>.



**■ニューロンの削減 (一例)** ニューロンの削減の手法は重みが閾値以下の場合ニューロンを削減し、再学習を行う。下記の例は重みが 0.1 以下のニューロンを削減した。



### 3.7 モデル軽量化まとめ

1. 量子化 … 重みの精度を下げることにより計算の高速化と省メモリ化を行う技術
2. 蒸留 … 複雑で精度の良い教師モデルから軽量な生徒モデルを効率よく学習を行う技術
3. プルーニング … 寄与の少ないニューロンをモデルから削減し高速化と省メモリ化を行う技術

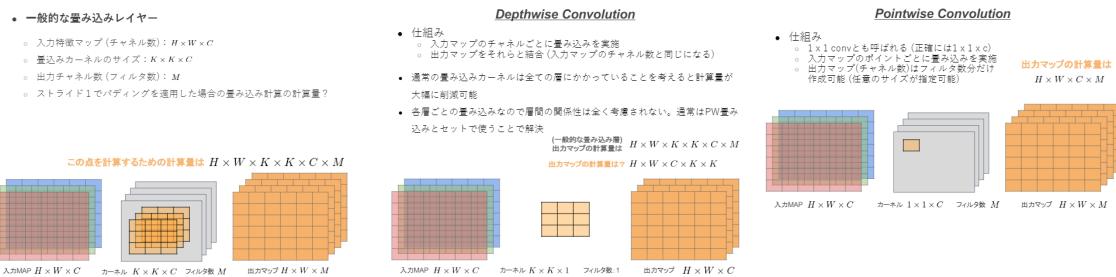
## 第4章

# 応用技術

### 4.1 MobileNet

1. 論文タイトル「MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications」
2. 提案手法
  - (a) ディープラーニングモデルは精度は良いが、その分ネットワークが深くなり計算量が増える。
  - (b) 計算量が増えると、多くの計算リソースが必要で、お金がかかってしまう。
  - (c) ディープラーニングモデルの軽量化・高速化・高精度化を実現（その名の通りモバイルなネットワーク）
  - (d) <https://qiita.com/HiromuMasuda0228/items/7dd0b764804d2aa199e4>

MobileNet は、画像認識分野における軽量化モデルの一例である。2017 年以降の進展はない成熟した分野（既に実用化）。現在の研究の主軸はいかにして軽量化するかという点に移っている。一般的な畳み込みレイヤーは計算量が多い。MobileNets は、Depthwise Convolution と Pointwise Convolution の組み合わせで軽量化を実現したものである。

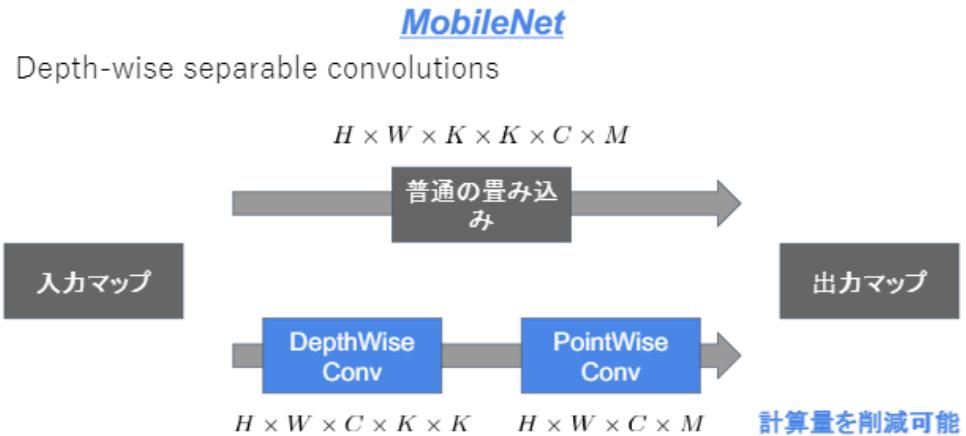


#### ■MobileNet のアーキテクチャ

1. Depthwise Separable Convolution という手法を用いて計算量を削減している。通常の畳み込みが空間方向とチャネル方向の計算を同時に使うのに対して、Depthwise Separable Convolution ではそれらを Depthwise Convolution と Pointwise Convolution と呼ばれる演算によって個別に行う。
2. Depthwise Convolution はチャネル毎に空間方向へ畳み込む。すなわち、チャネル毎に  $DK \times DK \times 1$  のサイズのフィルターをそれぞれ用いて計算を行うため、その計算量は  $(H \times W \times C \times K \times K)$  と

なる。

3. 次に Depthwise Convolution の出力を Pointwise Convolution によってチャネル方向に畳み込む。すなわち、出力チャネル毎に  $1 \times 1 \times M$  サイズのフィルターをそれぞれ用いて計算を行うため、その計算量は  $(H \times W \times C \times M)$  となる。



## 4.2 DenseNet

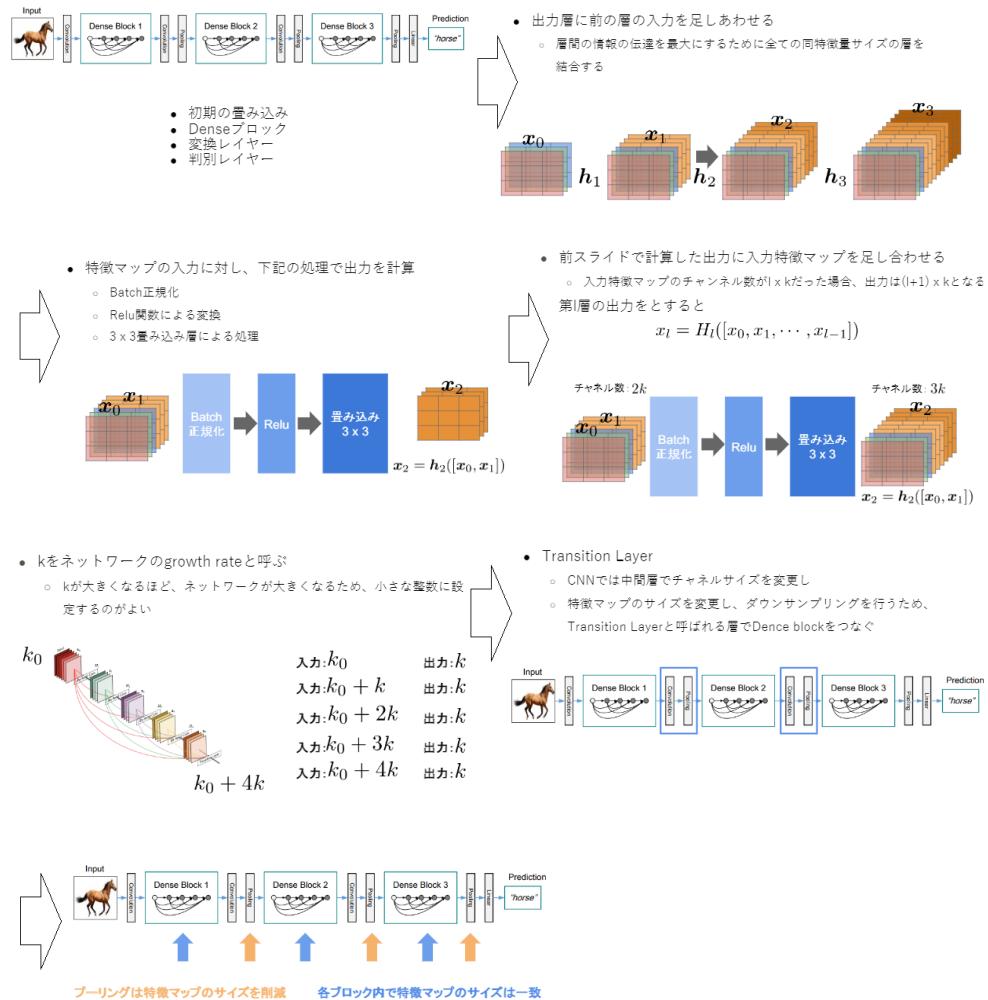
1. 論文タイトル「Densely Connected Convolutional Networks. G. Huang et., al. 2016」

<https://arxiv.org/pdf/1608.06993.pdf>

<https://www.slideshare.net/harmonylab/densely-connected-convolutional-networks>

2. 概要

Dense Convolutional Network (以下、DenseNet) は、畳込みニューラルネットワーク (以下、CNN) アーキテクチャの一種である。ニューラルネットワークでは層が深くなるにつれて、学習が難しくなるという問題があったが、Residual Network (以下、ResNet) などの CNN アーキテクチャでは前方の層から後方の層へアイデンティティ接続を介してパスを作ることで問題を対処した。DenseBlock と呼ばれるモジュールを用いた、DenseNet もそのようなアーキテクチャの一つである。

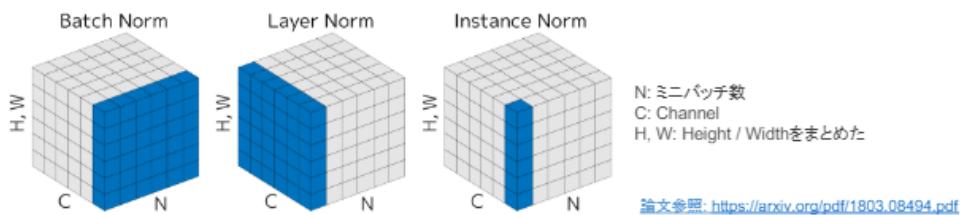


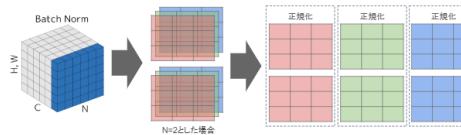
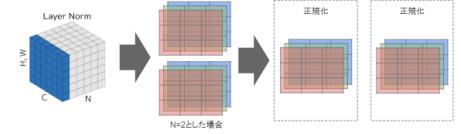
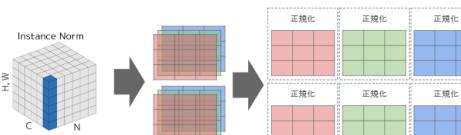
## ■サマリ

1. Batch Normalization
  - (a) ミニバッチに含まれる sample の同一チャネルが同一分布に従うよう正規化
  - (b) レイヤー間を流れるデータの分布を、ミニバッチ単位で平均が 0・分散が 1 になるように正規化
  - (c) Batch Normalization はニューラルネットワークにおいて学習時間の短縮や初期値への依存低減、過学習の抑制など効果がある。
  - (d) 問題点: Batch Size が小さい条件下では、学習が収束しないことがあり、代わりに Layer Normalization などの正規化手法が使われることが多い。
2. Layer Normalization… それぞれの sample の全ての pixels が同一分布に従うよう正規化
3. instance Normalization… さらに channel も同一分布に従うよう正規化

## 4.3 Layer 正規化/Instance 正規化

1. DenseNet と ResNet の違い
  - (a) DenseBlock では前方の各層からの出力全てが後方の層への入力として用いられる。
  - (b) ResidualBlock では前 1 層の入力のみ後方の層へ入力
2. DenseNet 内で使用される DenseBlock と呼ばれるモジュールでは成長率 (Growth Rate) と呼ばれるハイパーコンフィグレーションパラメータが存在する。この時、DenseBlock 内の各ブロック毎に  $k$  個ずつ特徴マップのチャネル数が増加していく時、 $k$  を成長率と呼ぶ。



<b>Batch Norm.</b> <ul style="list-style-type: none"> <li><math>H \times W \times C</math> の sample が <math>N</math> 個あった場合に、<math>N</math> 個の同一チャネルが正規化の対象                     <ul style="list-style-type: none"> <li>RGB の 3 チャネルの sample が <math>N</math> 個の場合は、それぞれのチャネルの平均と分散を求めて正規化を実施 (図の青い部分に対応)。チャネルごとに正規化された特徴マップを出力。</li> </ul> </li> <li>ミニバッチのサイズを大きく取れない場合には、効率が悪くなってしまう。</li> </ul> <p></p> <p>論文参照: <a href="https://arxiv.org/pdf/1803.08494.pdf">https://arxiv.org/pdf/1803.08494.pdf</a></p>	<b>Layer Norm.</b> <ul style="list-style-type: none"> <li>Layer Norm                     <ul style="list-style-type: none"> <li><math>N</math> 個の sample のうち一つに注目。<math>H \times W \times C</math> の全ての pixel が正規化の対象。                             <ul style="list-style-type: none"> <li>RGB の 3 チャネルの sample が <math>N</math> 個の場合は、ある sample を取り出し、全てのチャネルの平均と分散を求めて正規化を実施 (図の青い部分に対応)。特徴マップごとに正規化された特徴マップを出力</li> <li>ミニバッチの数に依存しないので、上記の問題を解消できていると考えられる。</li> </ul> </li> </ul> </li> </ul> <p></p> <p>論文参照: <a href="https://arxiv.org/pdf/1803.08494.pdf">https://arxiv.org/pdf/1803.08494.pdf</a></p>
<b>Instance Norm.</b> <ul style="list-style-type: none"> <li>Instance Norm                     <ul style="list-style-type: none"> <li>各 sample の各チャネルをごとに正規化                             <ul style="list-style-type: none"> <li>Batch Normalization のバッチサイズが 1 の場合と等価</li> </ul> </li> <li>コントラストの正規化に寄与・画像のスタイル転送やテクスチャ合成タスクなどで利用                             <ul style="list-style-type: none"> <li><a href="https://blog.cesnola.com/posts/1493/">https://blog.cesnola.com/posts/1493/</a></li> <li><a href="https://gengano.com/2019/06/16/post-573/">https://gengano.com/2019/06/16/post-573/</a></li> <li><a href="https://blog.albert2005.co.jp/2018/09/05/group_normalization/">https://blog.albert2005.co.jp/2018/09/05/group_normalization/</a></li> </ul> </li> </ul> </li> </ul> <p></p> <p>論文参照: <a href="https://arxiv.org/pdf/1803.08494.pdf">https://arxiv.org/pdf/1803.08494.pdf</a></p>	

## 4.4 WaveNet

囲碁で世界トップの実力を持つプロ棋士に勝利した AI 「AlphaGo」を作った、Google 傘下の DeepMind が開発した。波形接続 TTS ではなくパラメトリック TTS を利用している。2016 年にモデルが発表され、2017 年に実用化段階にいたった。最初のリリース時点 (2016 年) では WaveNet は非常に計算コストが高く、0.02

秒の音声を生成するのに 1 秒を要していた。(たった 1 秒の音声を生成するのにほぼ 1 分近い時間がかかっていたことを意味する。) そのため、実用には適していなかった。その後、新しく改良された WaveNet (2017 年) は、実時間に比べて 20 倍の速さで音声を生成するようになった。(2 秒の音声を 0.1 秒で生成することを意味する。) さらに 8 ビットではなく 16 ビットで、1 秒あたり 2 万 4000 回という高いレートでサンプリングを行なうことも可能になった。コンピュータによる音声合成・音声認識に対して大きな衝撃を与えた技術といわれる<sup>\*1</sup>。

**■wavenet の革新性** WaveNet は従来型よりも自然な発音に大きく近づいた。これは単純化や近似といった、大きな劣化を生むプロセスがないため、WaveNet の特筆すべき点とされる。反対にボーカロイドなどがいわゆる「機械っぽい」音声から抜け出せないのは、従来の数理モデルを利用しているためである。数理モデルは実際の現象をかなり単純化しているので、そこで大きな劣化が生まれているのが 1 つの要因だとされる。

WaveNet の誕生が可能になった土台には、インフラ的な側面とアルゴリズム的な側面があるとされる。インフラ的な側面としては「データ量と計算能力」の圧倒的な向上があり、アルゴリズム的な側面として「音声を『点』の時系列として捉え、ディープラーニングに入力すること」が発展したことがある。

WaveNet は「音声を点として考える」方式を取っている。音声は本来アナログ信号だが、コンピュータで扱う際には「サンプリング」および「量子化」(= 波形を分割して点にすること) という作業をする。WaveNet とは、量子化された個々の波形を一度に 1 サンプルずつ、毎秒 16,000 サンプルで作成し、個々のサウンド間のシームレスな遷移を可能にするディープ・ジェネレーティブ・モデルである。

#### ※量子化について

例えば CD 音質の「44.1kHz / 16bit」などのこと。これはある音声波形から 1 秒間に 4 万 4100 回、波形の「一瞬」を切り取り (標本化)、その波形の値を 2 の 16 乗の種類、すなわち、6 万 5536 種類の値に置き換え (量子化)、1 秒間の空気振動を 4 万 4100 個の 6 万 5536 種類からなる点というデジタルデータで表している。音声を取り扱う際には、16kHz のサンプリング周波数が比較的よく利用されている。音声の特徴をうまく活用すれば、8bit の量子化でも比較的高い音質を維持できる。

WaveNet では、この「点」を畳み込みニューラルネットワーク (CNN) で処理している。ただし、音声信号特有の難しさとして、時系列の依存関係が非常に長い点がある。この長い依存関係を自己回帰モデルで扱うのは困難な課題であった。WaveNet は dilated convolution という仕組みを使って受容野が指數関数的に広くなるように CNN を構築し、この問題を解決した。音声を、1 秒間だけでも 1 万 6000 個の点になる巨大な系列データとして捉え、音声合成の際にはこの点を直接 CNN から生成することで音声波形を作っている。これにより、従来の数理モデルで必要だった過程や近似などの研究者による調整作業がなくなった。※ dilated convolution のきっかけになったのが、DeepMind 社が同時期に発表した PixelRNN と PixelCNN モデルである。同モデルは複雑な自然画像を一度に 1 ピクセルずつではなく、一度に 1 色チャンネル (one colour-channel) ずつ生成することが可能であることを示した。これに触発されて、2 次元の PixelNet を 1 次元の WaveNet に適応させたことが問題解決へと導いた。

---

<sup>\*1</sup> <https://deepsquare.jp/2020/04/wavenet/>

- Wavenet

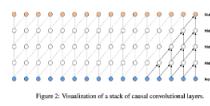
- Aaron van den Oord et. al., 2016らにより提案
  - AlphaGoのプログラムを開発しており、2014年にgoogleに買収される
  - 生の音声波形を生成する深層学習モデル
  - Pixel CNN(高解像度の画像を精密に生成できる手法)を音声に応用したもの

- 関連記事

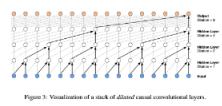
- <https://gigaom.net/news/2017/09/05/wavenet-launches-a-google-assistant/>
- <https://qz.com/1144469/ai-items/cod0fe4735a12071c2/>
- [https://www.slideshare.net/NU\\_TODALAB/wavenet-86493372](https://www.slideshare.net/NU_TODALAB/wavenet-86493372)

- Wavenetのメインアイディア

- 時系列データに対して畳み込み(Dilated convolution)を適用する
- Dilated convolution
  - 層が深くなるにつれて畳み込むリンクを離す
  - 受容野を簡単に増やすことができるという利点がある
  - 図(右)では、dilated = 1,2,4,8となっている



既存の畳み込み



畳み込み (Dilated)

<https://arxiv.org/pdf/1609.03490.pdf>

- 深層学習を用いて結合確率を学習する際に、効率的に学習ができるアーキテクチャを提案したことが WaveNet の大きな貢献の1つである。
- 提案された新しい Convolution 型アーキテクチャは (a) と呼ばれ、結合確率を効率的に学習できるようになっている。
- Dilated causal convolution
- Depthwise separable convolution
- Pointwise convolution
- Deconvolution

- (a) を用いた際の大きな利点は、単純な Convolution layer と比べて (い) ことである。

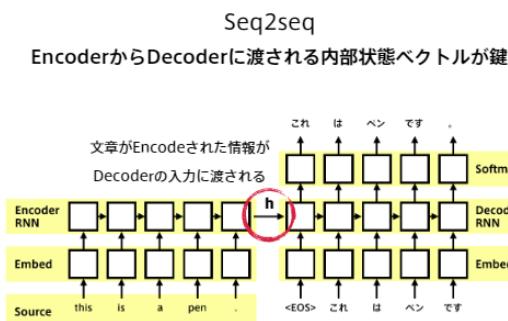
- パラメータ数に対する受容野が広い
- 受容野あたりのパラメータ数が多い
- 学習時に並列計算が行える
- 推論時に並列計算が行える

# 第 5 章

## Transformer

### 5.1 Seq2Seq

系列 (Sequence) を入力して、系列を出力するもの。



■RNN の理解 系列データを読み込むために再帰的に動作する NN（出力がまた次の入力に用いられる）

■言語モデルとは 言語モデルは単語の並びに確率を与える。単語の並びに対して尤度（それがどれだけ起こりえるか）、すなわち文章として自然かを確率で評価すること。この狙いは、時刻  $t-1$  までの情報で、時刻  $t$  のじ後確率を求めることが目標。これによって同時確率が計算できるようにすること。

■サマリ

1. RNN は系列情報を内部状態に変換することができる。
2. 文章の各単語が現れる際の同時確率は、事後確率で分解できる。したがって、事後確率を求めることが RNN の目標になる。
3. 言語モデルを再現するように RNN の重みが学習されれば、ある時点の次の単語を予測することができる。したがって、先頭単語を与えれば文章を生成することも可能

■Theacher Forcing この手法を用いる背景として、decoder の出力が初めの層で生起した場合は、どんどんエラーが増大しシステムとしての出力が不安定になる恐れがあるため。このため、正解ラベルを decoder の入力に用いる手法が用いられる。一方で正解ラベルに特化して decoder として学習しすぎると validation データに対して Self-Attention が肝入力を全て同じにして学習的に注意箇所を決めていくクロスエントロピー

が増大するため、実際の実装としては、「scheduled sampling として teacher forcing を使うか decoder の output を次の入力に使うか確率的に採用」することで対処している。

## 5.2 Transformer

BERT の構造が transgormer である。Transformer は、RNN を用いていない。Attention 機構のみで構築されている。トランسفォーマー（Transformer）は、2017 年に発表された深層学習モデルであり、主に自然言語処理（NLP）の分野で使用される。自然言語などの時系列データを扱って翻訳やテキスト要約などのタスクを行うべく設計されているのは回帰型ニューラルネットワーク（RNN）と同様だが、Transformer の場合、時系列データを逐次処理する必要がないという特徴がある。たとえば、入力データが自然言語の文である場合、文頭から文末までの順に処理する必要がない。このため、Transformer では回帰型ニューラルネットワークよりもはるかに多くの並列化が可能になり、トレーニング時間が短縮される。

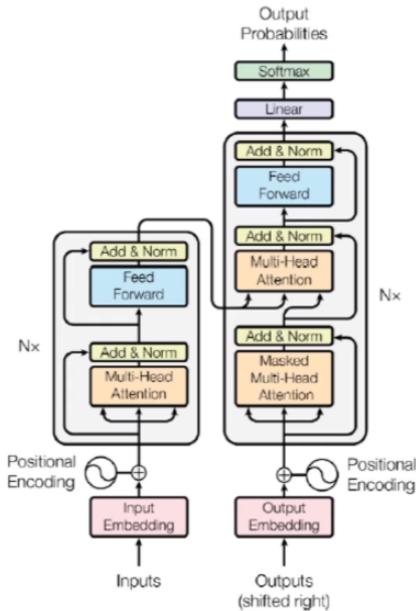
導入以来、Transformer モデルは自然言語処理の多くの問題に取り組む上で広く選択されており、Long Short-term Memory（LSTM）などの古い回帰型ニューラルネットワークモデルに取って代わった。Transformer モデルはトレーニング中の並列化を容易にするため、より大きなデータセットでのトレーニングを可能にした。このことが、BERT（Bidirectional Encoder Representations from Transformers）や GPT（Generative Pre-trained Transformers）などの事前トレーニング済みシステムの開発につながった。これらは、巨大な一般言語データセットでトレーニングされており、特定の言語タスクにファインチューニングできる。

■Attention 翻訳先の各単語を選択する際に、翻訳元の文中の各単語の隠れ状態を利用するもの。  
従来の NN 機械翻訳では、元の文の内容をひとつのベクトルで表現しているため、文長が長くなると表現力が足りなくなる。

# Transformer (Vaswani et al., 2017)

## Attention is all you need

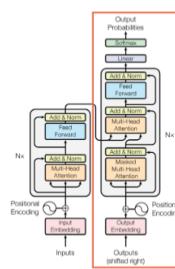
- 2017年6月に登場
- RNNを使わない
  - 必要なのはAttentionだけ
  - 当時のSOTAをはるかに少ない計算量で実現
  - 英仏(3600万文)の学習を8GPUで3.5日で完了



1. Attention 注意機構には二種類ある (ソース・ターゲット注意機構 / 自己注意機構)
2. Transformer-Encoder 自己注意機構により文脈を考慮して各単語をエンコード
3. Self-Attention が肝 入力を全て同じにして学習的に注意箇所を決めていく
4. Position-Wise Feed-Forward Networks 位置情報を保持したまま順伝播させる  
各 Attention 層の出力を決定 「2層の全結合 NN」、「線形変換→ReLU→線形変換」
5. Scaled dot product attention 全単語に関する Attention をまとめて計算する
6. Multi-Head attention 重みパラメタの異なる 8 個のヘッドを使用  
8 個の Scaled Dot-Product Attention の出力を Concat。 それぞれのヘッドが異なる種類の情報を収集

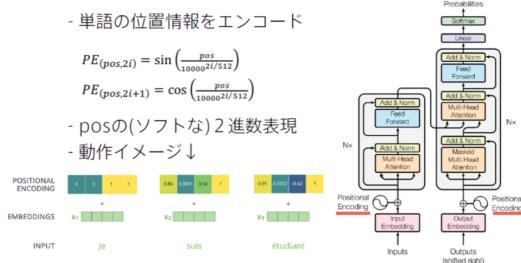
Decoder

- Encoderと同じく6層
- 各層で二種類の注意機構
- 注意機構の仕組みはEncoderとほぼ同じ
- 自己注意機構
  - 生成単語列の情報を収集
    - 直下の層の出力へのアテンション
    - 未来の情報を見ないようにマスク
- Encoder-Decoder attention
  - 入力文の情報を収集
  - Encoderの出力へのアテンション

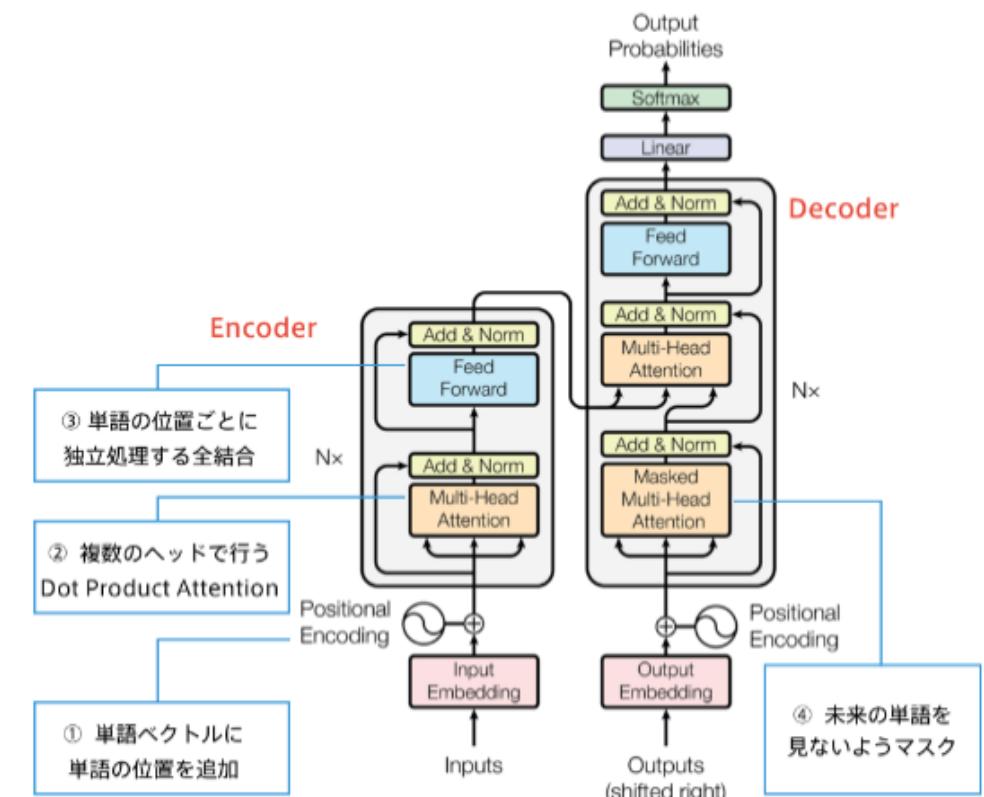


## Position Encoding

RNNを用いないで単語列の語順情報を追加する必要がある



### ■サマリ

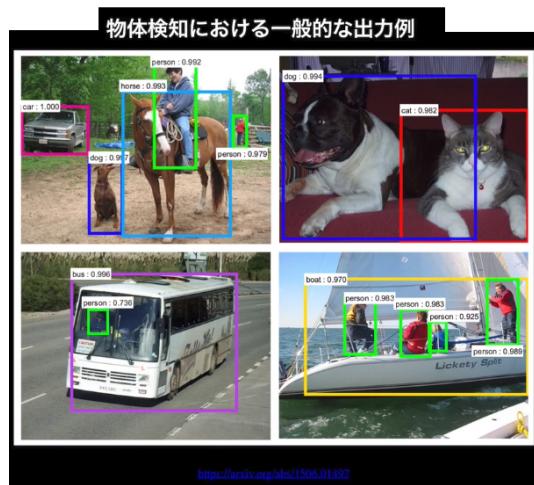
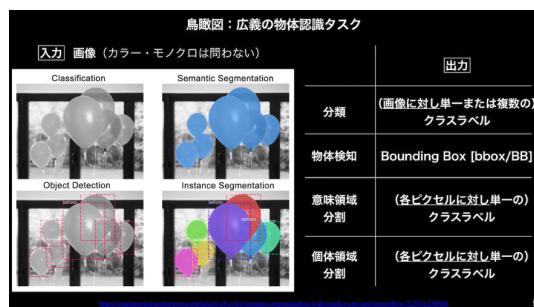


## 第6章

# 物体検知・セグメンテーション

■物体検出における出力 次の3種類の要素が含まれる。

1. 物体の位置 (bounding box, BB)
2. 予測ラベル
3. confidence (信頼度)



■代表的なデータセット クラス数、Trainデータ数/Validation数、Box/画像等の違いがあるが学習の用途に応じてどのデータセットを用いるか選択することが重要。この中でも特にBox/画像の値が重要であり、この値が大きくなると「部分的な重なり等も見られる日常生活のコンテキストに近く」なり、一方この値が小さ

くなると「アイコン的な映りとなり、日常感とはかけ離れやすい」といったことが生じる。

従って、目的に応じた Box/画像の選択をすべきであり、クラス数が大きいことは必ずしも嬉しいとは一概に言えない。

**表 2** 物体認識用の一般的なデータセット。PASCAL VOC, ImageNet, MS COCO, Open Images の画像例を図 9 に示す。(訳注: インターネット上で画像収集されることが多いため、画像によって解像度が異なることが多い。「画像サイズ」列に記載されているのは、およそその平均解像度または典型的な解像度と推測される。)

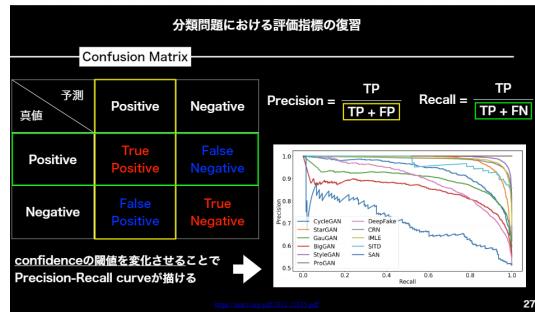
データセット名	総画像数	カテゴリ数	画像あたりの物体数	画像サイズ	開始年	注目点
PASCAL VOC (2012) [69]	11,540	20	2.4	470×380	2005	日常生活で一般的な 20 カテゴリのみカバー。多数の訓練画像。実世界のアプリケーションに近い。クラス内変動が非常に大きい。シーンコンテキスト内の物体。1 枚の画像に複数の物体。多数の難しいサンプルを含む。カテゴリあたりの画像数は 303~4087。
ImageNet [234]	1400万+	21,841	1.5	500×400	2009	多数の物体カテゴリ。画像あたりの物体のインスタンス数とカテゴリ数が多い。PASCAL VOC よりチャレンジング。ILSVRC チャレンジの根幹。画像は物体が中心。
MS COCO [166]	32.8万+	91	7.3	640×480	2014	実世界のシナリオに更に近い。各画像にはより多くの物体インスタンスとより豊富な物体アノテーション情報が含まれる。ImageNet データセットでは使用できない物体セグメンテーションのアノテーションデータを含む。
Places [319]	1000万+	434	—	256×256	2014	シーン認識用の最大のラベル付きデータセット。Places365 Standard, Places365 Challenge, Places 205, Places88 の 4 つのサブセットをベンチマークとして使用。
Open Images [143]	900万+	6,000+	8.3	多様	2017	画像レベルのラベル、物体の bbox, visual relationship (物体関係、視覚的関係) のアノテーションがされている。Open Images V5 は大規模な、物体検出、物体インスタンスセグメンテーション、visual relationship detection (物体関係検出、視覚的関係検出) をサポートしている。



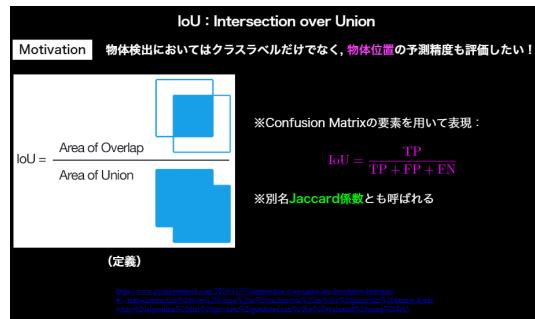
**図 9** PASCAL VOC, ILSVRC, MS COCO, Open Images の物体アノテーション付きの画像の例。データセットの概要については表 2 を参照されたい。

**■評価指標** 物体検出システムの性能を測る指標としては、大きく 2 つの観点から挙げることができる。1 つが処理速度であり、もう 1 つが精度である。特に処理速度を測る指標としてフレームパー毎秒 (FPS)、精度を測る指標として適合率と再現率がある。以上の指標は物体検出に限らず用いられる指標であるが、物体検出に特有の数値として IoU(Intersection over Union) がある。これはある推定結果と対応する正解がどの程度重なっているかを表す数値であり、完全に一致しているときには 1、全く重なる部分がないときには 0 となる。

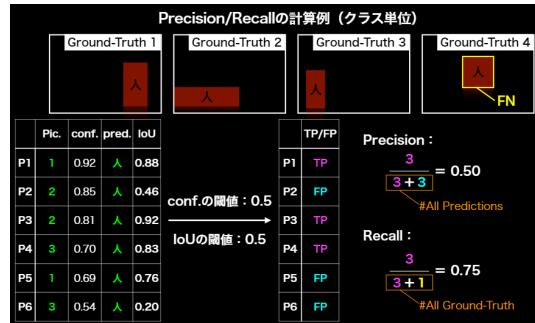
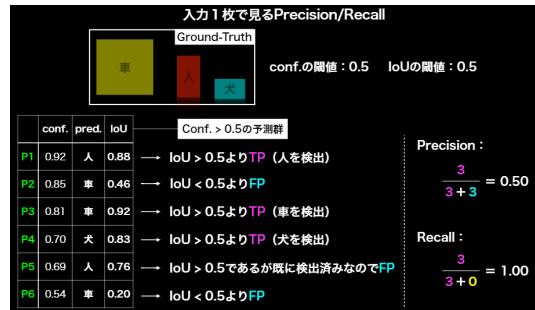
実際の検出システムでは完全に正解と一致する結果を得ることは困難であるため、実運用評価の際にはこの IoU が一定値以上の結果を正解とみなす精度を測ることになる。また、適合率と再現率の他に、これらを組み合わせた平均適合率 (Average Precision, AP) も用いられることが多い。推論時には推論した結果とともにどの程度の確からしさでその検出結果を得たかという指標も返されるが、この確からしさも用いて計算される指標である。適合率と再現率は一般にトレードオフの関係にあるため、双方の要素を取り込んだ平均適合率が使われる。



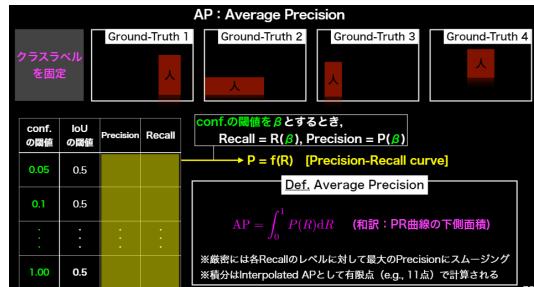
## ■IoU(Intersection of Union)



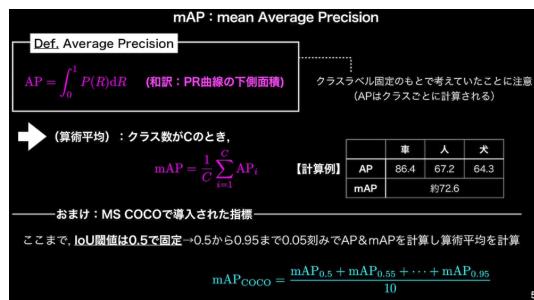
## ■Precision/Recall の一例



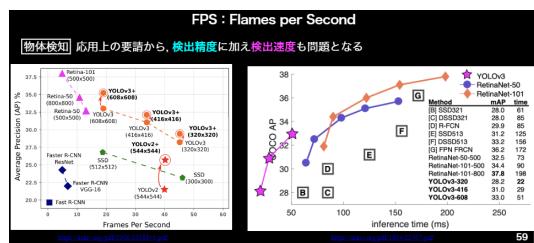
## ■AP



mAP

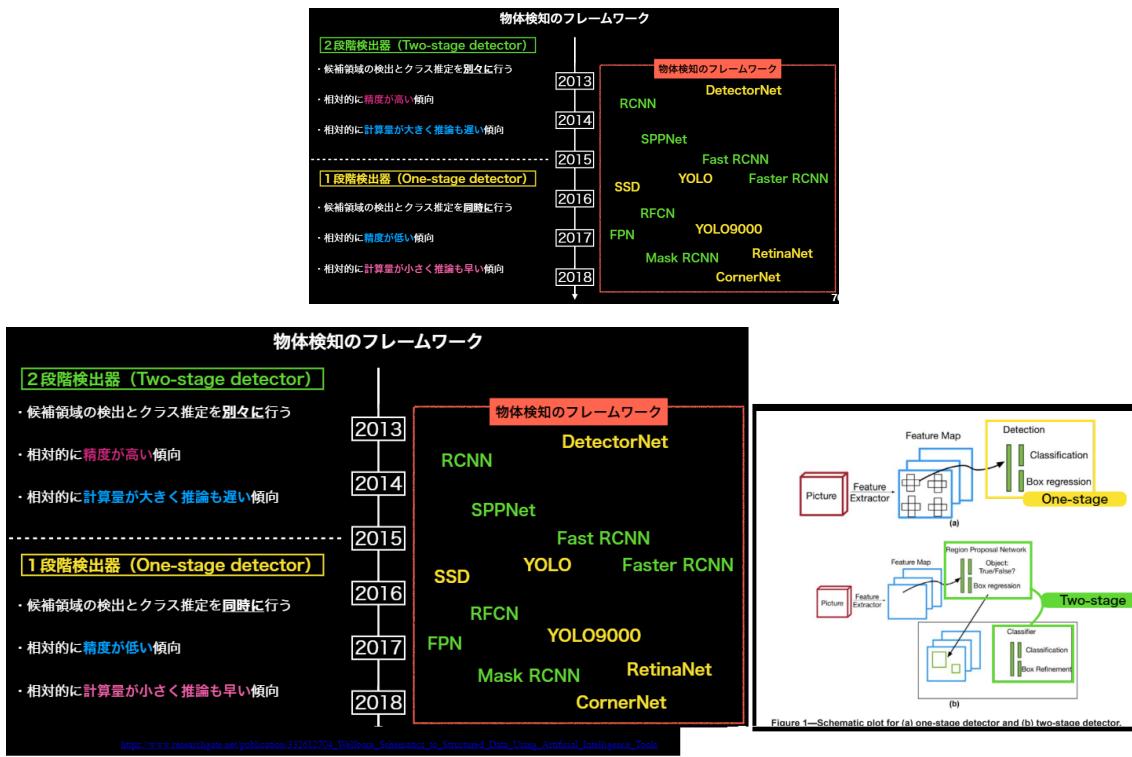


■ FPS

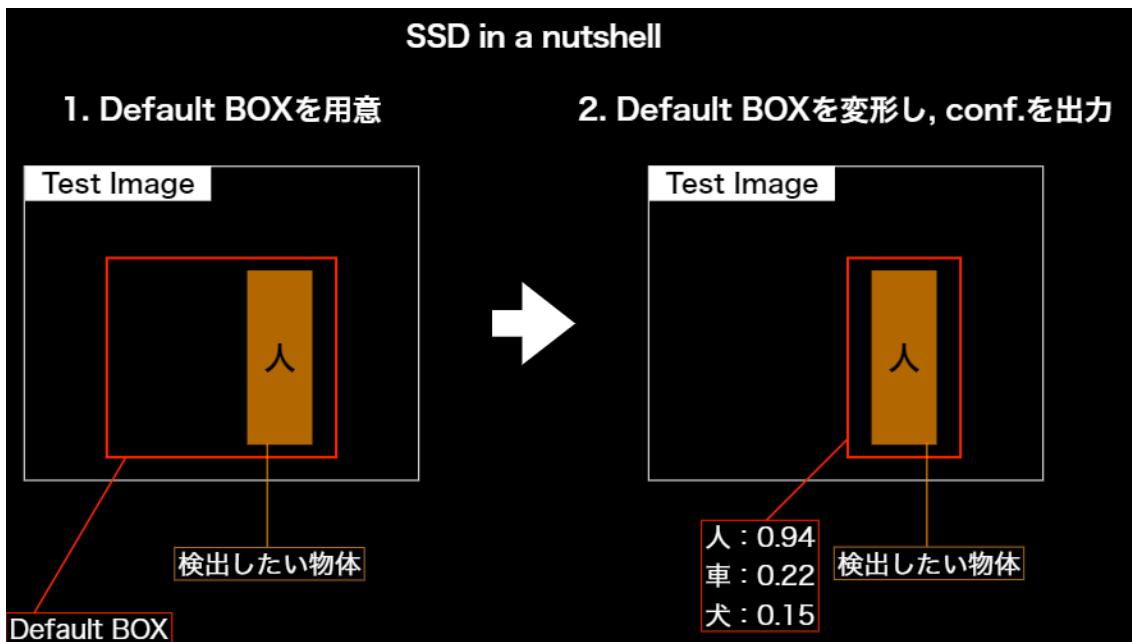


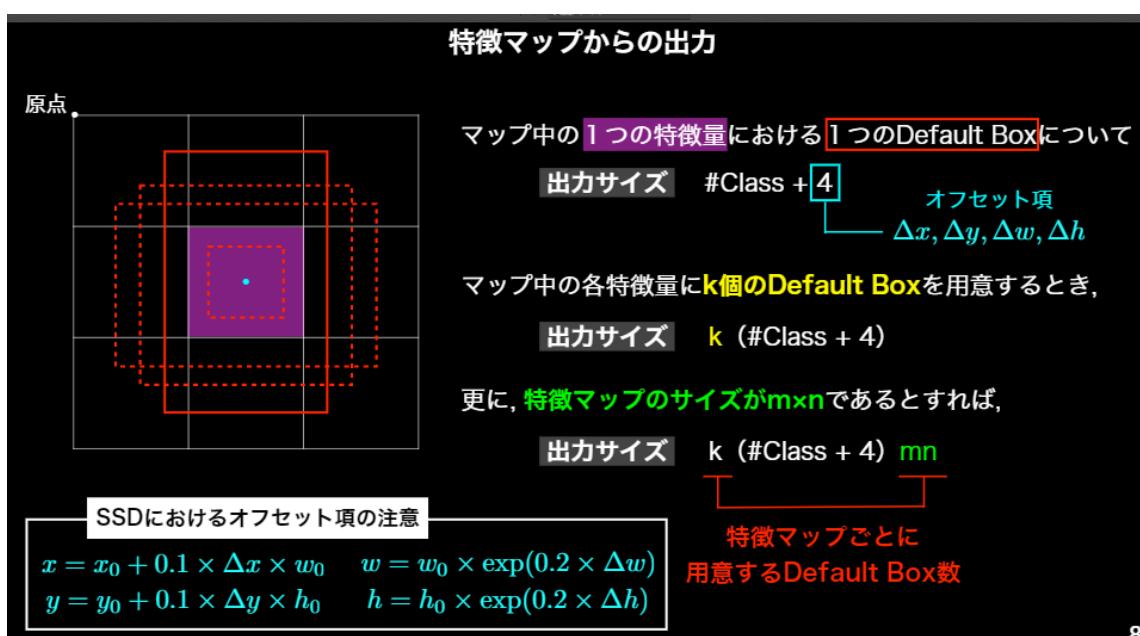
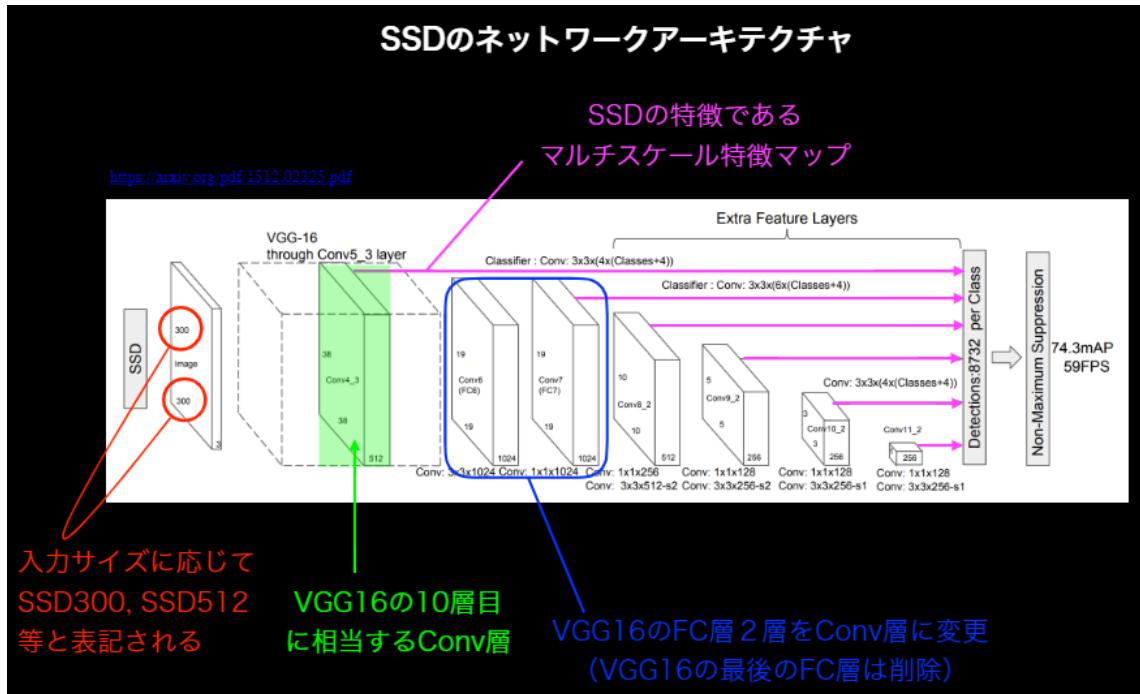
### ■物体検知のフレームワーク



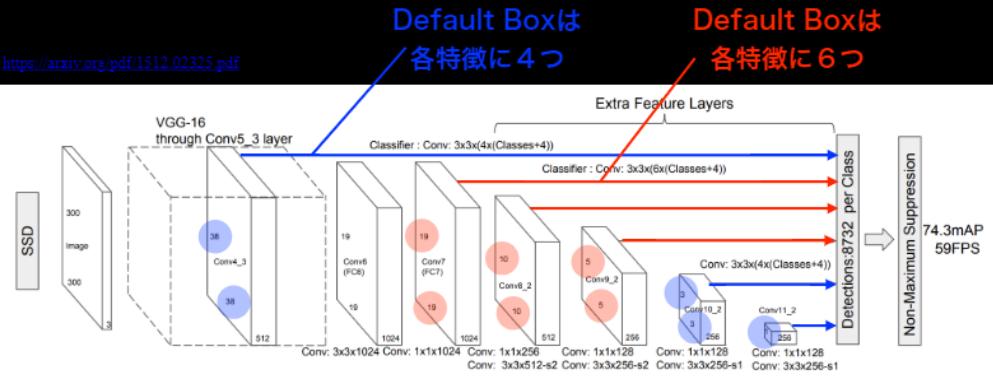


■SSD(Single Shot Detector) 1段階検出器。2016年に発表された手法。YOLOが抱える課題に対応して改良された1ステージ系の手法。YOLOでは物体位置の推定にネットワークの最終層で得られる特徴量しか用いなかったのに対し、SSDでは入力に近い層の特徴量も用いたことが特徴で、より小さいサイズの物体の検出にも対応できるようになった。また、複数のアスペクト比を持つ矩形内で畳み込みを行うことで、異なるスケール・アスペクト比を持つ物体の検出にも頑健になった。



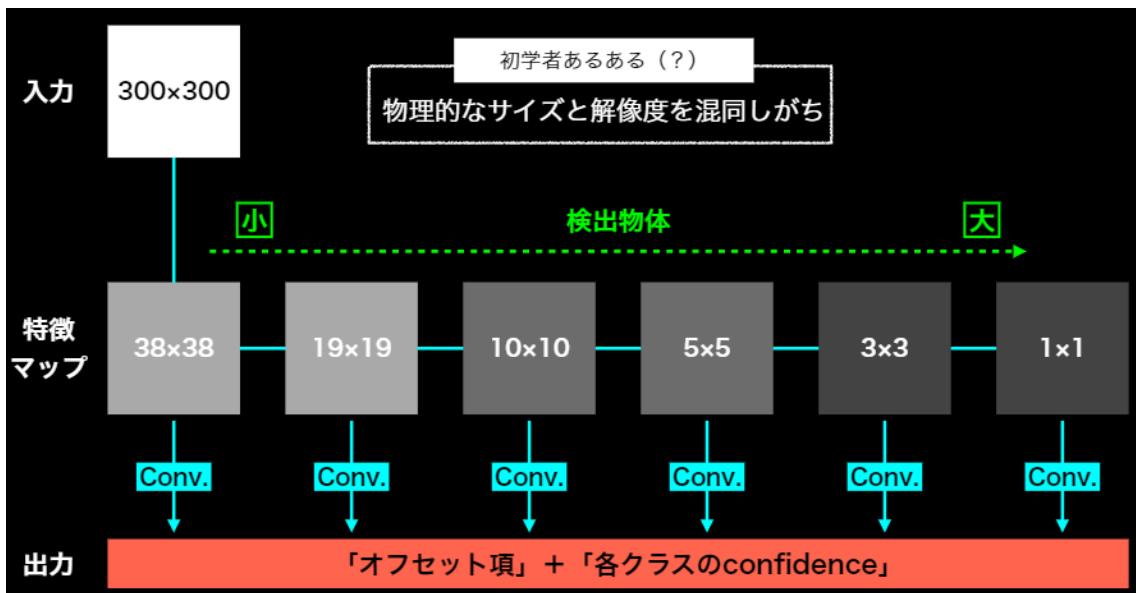


## SSDのデフォルトボックス数



VOCデータセットではクラス数20に背景クラスが考慮され, #Class = 21となることに注意して

$$\begin{aligned} & 4 \times (21+4) \times 38 \times 38 + 4 \times (21+4) \times 3 \times 3 + 4 \times (21+4) \times 1 \times 1 \\ & + 6 \times (21+4) \times 19 \times 19 + 6 \times (21+4) \times 10 \times 10 + 6 \times (21+4) \times 5 \times 5 \\ & = 8,732 \times (21+4) \end{aligned}$$



■semantic segmentation セマンティック セグメンテーション (Semantic Segmentation) は、画像内の全画素にラベルやカテゴリを関連付けるディープラーニング (Deep Learning) のアルゴリズムです。特徴的なカテゴリを形成する画素の集まりを認識するために使用される。たとえば、自動運転車は車両、歩行者、交通標識、歩道、その他の道路の特徴を識別する必要がある。セマンティック セグメンテーションは、自動運転、医療用画像処理、工業用検査など幅広い用途で使用されている。

一方で、semantic segmentation を実装する際に、入力データの情報（解像度）が畳み込み（プーリング）を繰り返すことで解像度が低下する課題が生起する。ここで 1 つの疑念が生じる semantic segmentation する際に up-sampling が必要となる。そもそもなぜプーリングが必要であったであろうか？

画像を正しく認識するためには、受容野にある程度の大きさが必要であった（小さな kernel をスライド）。具体的に受容野を広げる方策としては、「深い畳み込み層」の設置、「プーリング&ストライド」などを用いていた。この際、前者の深い畳み込み層では、メモリ、計算所要が大きくなるため、後者のプーリング（ストライド）が受容野拡大として用いることが必要となるのである。※ 逆畳み込みではなく、プーリングで失われた情報は再現ができないことに注意されたい

セマンティック セグメンテーションの簡単な例では、画像を 2 つのクラスに分けます。たとえば、図 1 の浜辺に佇む人物の画像では、画像内の画素は人物と背景という 2 つのクラスに分割されています。



図 1: 画像とラベル付き画素

# 参考文献

- [1] 奥村晴彦, 黒木裕介『 $\text{\LaTeX} 2_{\mathcal{E}}$  美文書作成入門 第7版』(技術評論社, 2017)
- [2] 加藤公一, 『機械学習のエッセンス』(SBクリエイティブ, 2018)
- [3] 大重美幸, 『Python 3入門ノート』(ソーテック, 2018)
- [4] 大閑真之, 『機械学習入門』(オーム, 2018)
- [5] Andreas C. Müller et al., 『Python ではじめる機械学習』(オライリージャパン, 2018)
- [6] 加藤公一(監修), 『機械学習図鑑』(翔泳社, 2019)
- [7] 岡谷貴之, 『深層学習』(講談社, 2015)
- [8] 竹内一郎, 『サポートベクトルマシン』、講談社, 2015)
- [9] 斎藤康毅, 『ゼロから作る Deep Learning』(オライリージャパン, 2019)
- [10] Guido van Rossum, 『Python チュートリアル(第4版)』(オライリージャパン, 2021)
- [11] 中田亨, 『多様性工学』(日科技連, 2021)