1. **強化学習**

**強化学習 (Reinforcement Learning) とは、機械学習の一種であり、コンピュータエージェントが動的環境と、繰り返し試行錯誤のやりとりを重ねることによってタスクを実行できるようになる手法である。この学習手法により、エージェントは、タスクの報酬を最大化する一連の意思決定を行うことがでる。人間が介入したり、タスクを達成するために明示的にプログラムしたりする必要が無い。（MathWorksより）**

1. **AlphaGo**

**囲碁で世界チャンピオンに勝利した最初のコンピュータプログラムである AlphaGo は、何百万ものゲームのプレイによる学習が、Q 学習というアルゴリズムを使いながら数日間休むことなく行われ、その結果、数千年分の人間の知識が蓄積された。比較的単純なアプリケーションであっても、学習には数分から数時間、場合によっては数日かかることがある。また、設計上必要な判断項目のリストがあるために、問題を適切に設定することが困難な場合がある。この場合、適切に進めるために反復が複数回必要になる場合がある。例えば、ニューラルネットワークの適切なアーキテクチャの選択、ハイパーパラメータの調整、報酬のシェイピングなどがある。（MathWorksより）**

1. **軽量化・高速化技術**
2. **データ並列化**

**ミニバッチを複数の計算機に分散する方法で、同期型と非同期型がある。非同期型は処理速度が速い。しかし、最新のパラメータが利用できないため、学習は不安定になりやすい。同期型の方が精度は良いため、主流となっている。データ並列では、複数あるGPUのそれぞれに、ニューラルネットワークの全体を複製する入力となるミニバッチは、事例の単位で分割され、異なるGPUで並列に処理される。データ並列は、実装が容易で実行効率も高いため、多くの深層学習フレームワークで実装されている。ただしデータでは、ニューラルネットワーク全体を複製するため、極めてパラメータ数が多い場合、1枚のGPUに全てのパラメータが収まらず、バッチサイズをできる限り小さくしても、学習ができないという問題がある。（自然言語処理の学会記事：自動並列化深層学習ミドルウェアRaNNCより）**

1. **モデル並列化**

**一つのモデルを複数の計算機に分散する方法である。モデル並列では、ニューラルネットワークを分割し、複数のGPUに配置する。各GPUは、ニューラルネットワークのパラメータの一部分のみを持つため、パラメータの多いニューラルネットワークでも学習できる。GPU1枚にパラメータが収まらない場合には、モデル並列の利用が必須となるが、どのようにニューラルネットワークを分割するかによって、学習の実行効率が異なる。（自然言語処理の学会記事：自動並列化深層学習ミドルウェアRaNNCより）**

1. **量子化（Quantization）**

**量子化とは、重みなどのパラメータをより小さいビットで表現することで、モデルの軽量化を図る手法である。使用するビットを制限することでネットワークの構造を変えずにメモリ使用量を削減できる。TensorFlowやPyTorchなどのディープラーニングのフレームワークでは、32ビット浮動小数点精度（float型）を使用することが主流である。（Laboroより）**

1. **量子化の利点と欠点**
2. **利点**

* **モデルサイズの大きさによりメモリ転送がボトルネックになっている場合のGPU推論の高速化**
* **モデルサイズ削減による、保存コストの削減**
* **通信コストの削減による、高頻繁なエッジデバイスへのモデルの配信**

1. **欠点**

* **使用するビット数を小さくするほど精度が低下する可能性が高い**

1. **速度の実験結果**

**32ビットと6ビットのモデルで画像から検出を行ったところ、6ビットの方が早かった。**

1. **蒸留（Distillation）**

**複雑で巨大な学習済みモデル(教師モデル)の関数を近似し、コンパクトなモデル(生徒モデル)を学習(知識を継承)するための手法である。（Qiitaより）**

**蒸留では、教師モデルの学習などで使用する正解ラベルをhard target、教師モデルの出力（スコア分布）をsoft targetと呼ぶ。生徒モデルを教師モデルに近づけるように学習するときに、hard targetとsoft targetの両方を用いる場合やsoft targetのみを用いる場合がある。soft targetを用いることが蒸留の大きな特徴である。これにより、hard targetに含まれない情報を含めて学習できるというメリットがある。**

****

図 1　 Distillationのイメージ

**「**図 1　 Distillationのイメージ**」の例において、教師モデルの出力スコアが自動車：0.4、犬：0.0であることから「この画像は、車に似ている部分があり、犬に似ている部分はない」と捉えることができる。hard targetからは「この画像は、電車である」という情報しか含まれていない。つまり、soft targetから「電車とは、車輪があって、窓があって、…、」のような一般的な概念を学習できることが期待できる。このように、学習時の情報量を増やすという観点からモデルを軽量化することが、プルーニングや量子化との違いである。他にも、蒸留はモデル圧縮以外の用途にも応用できることが知られている。その一つにデータ蒸留（Data Distillation）と呼ばれる手法がある。データ蒸留は、正解ラベルのないデータに対する教師モデルの予測結果を正解ラベルとして扱い、生徒モデルの訓練を行う手法である。この手法により、従来では利用できなかった正解ラベルのないデータを有効活用でき、かつ、精度向上に有効であることが示されている。（Laboroより）**

1. **プルーニング（Pruning：枝刈り）**

**ディプニューラルネットのモデルは、各ノードが密に結合している。そのノード間の重みが小さい箇所の接続を削除する、または影響の小さいノードを削除することでパラメータ数を削減する手法である。ノードや重みを削除することでパラメータ数を減少させる。記憶する必要があるパラメータが減少することによって、計算する回数が削減され、メモリ使用量が少なくなる。その結果、モデルの軽量化と処理の高速化が期待できる。どの接続を削除するのかについては様々な手法が提案されているが、「重みの絶対値が小さいものを削除する」手法が一般的で、性能も良いとされている。また、プルーニングによって80%〜90%程度の接続を削除できる場合もある。なお、一般的にはプルーニングだけでは精度が落ちてしまうため、プルーニング後に再学習を行うことで元の精度を保つことが行われる。（Laboroより）**

1. **応用技術**
2. **MobileNet**

**スマホなどの小型端末にも乗せられる高性能CNNを作りたいというモチベーションから生まれた軽量かつ(ある程度)高性能なCNNである。MobileNetにはv1,v2,v3がある。（Qiitaより）**

1. **Depthwise Separable Convolution**

**Depthwise Separable Convolutionとは、通常のConvolutionをDepthwise Conv.と Pointwise Conv.の2つに分けることで、パラメータ数を削減したものである。**

1. **通常のConvolution**

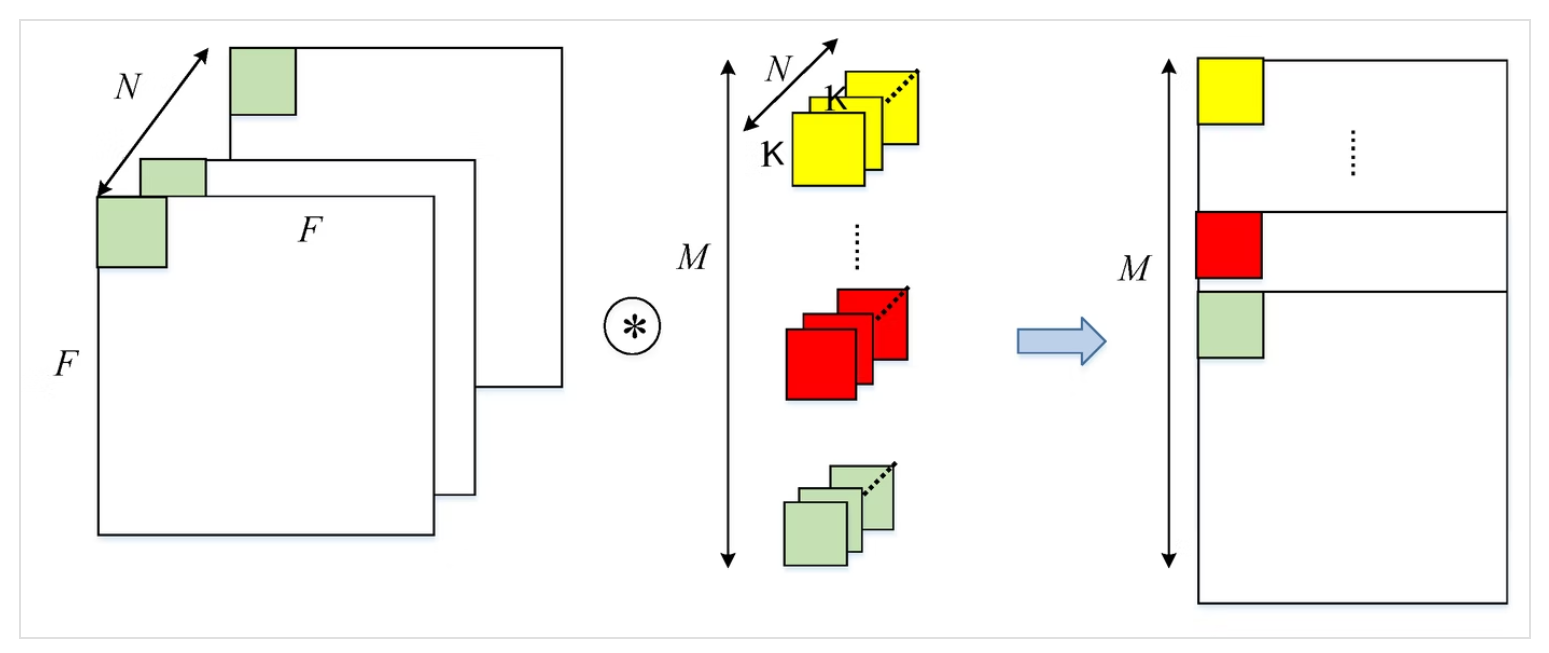
****

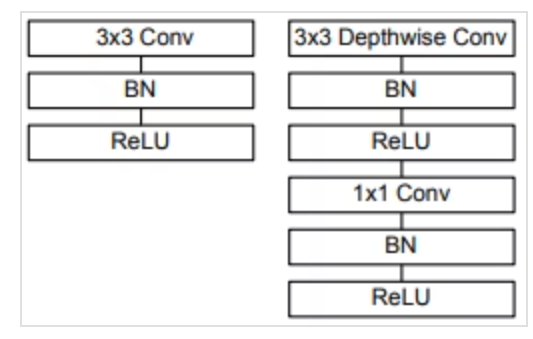
図 　通常Convolutionのイメージ

**チャネル数のデータに個のフィルタを畳み込み演算することで、 個の特徴量データができあがる。**

1. **Depthwise Separable Convolution**

****

**通常の畳み込みが空間方向とチャネル方向の畳み込みを同時に行うのに対して、Depthwise(空間方向) を行なったのちにPointwise(チャネル方向) を行なうようにしている。ポイントは空間方向とチャネル方向の畳み込みを同時に行うのではなく、順に行うという点である。**

****

**左は通常ののConvolutionで右は のConvolutionを近似した のDepthwise Separable Convolutionとなる。**

1. **Depthwise Separable Convolution**

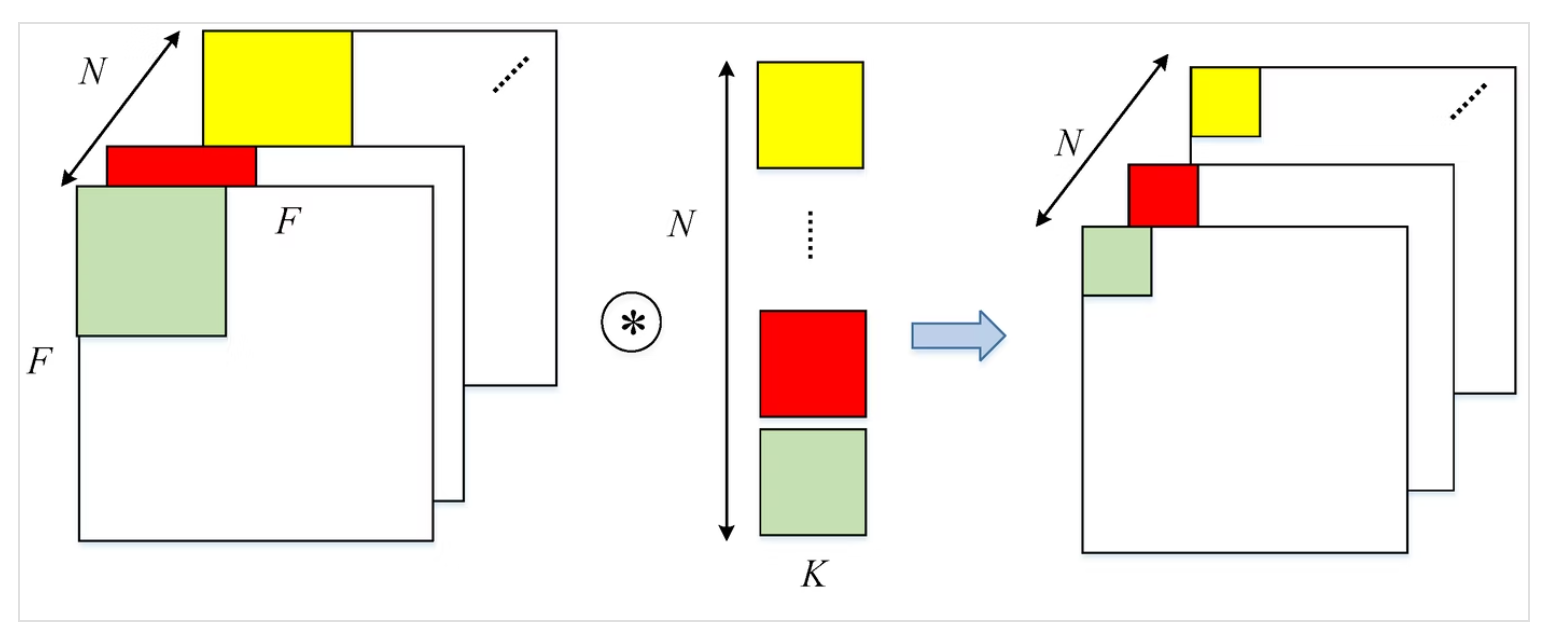
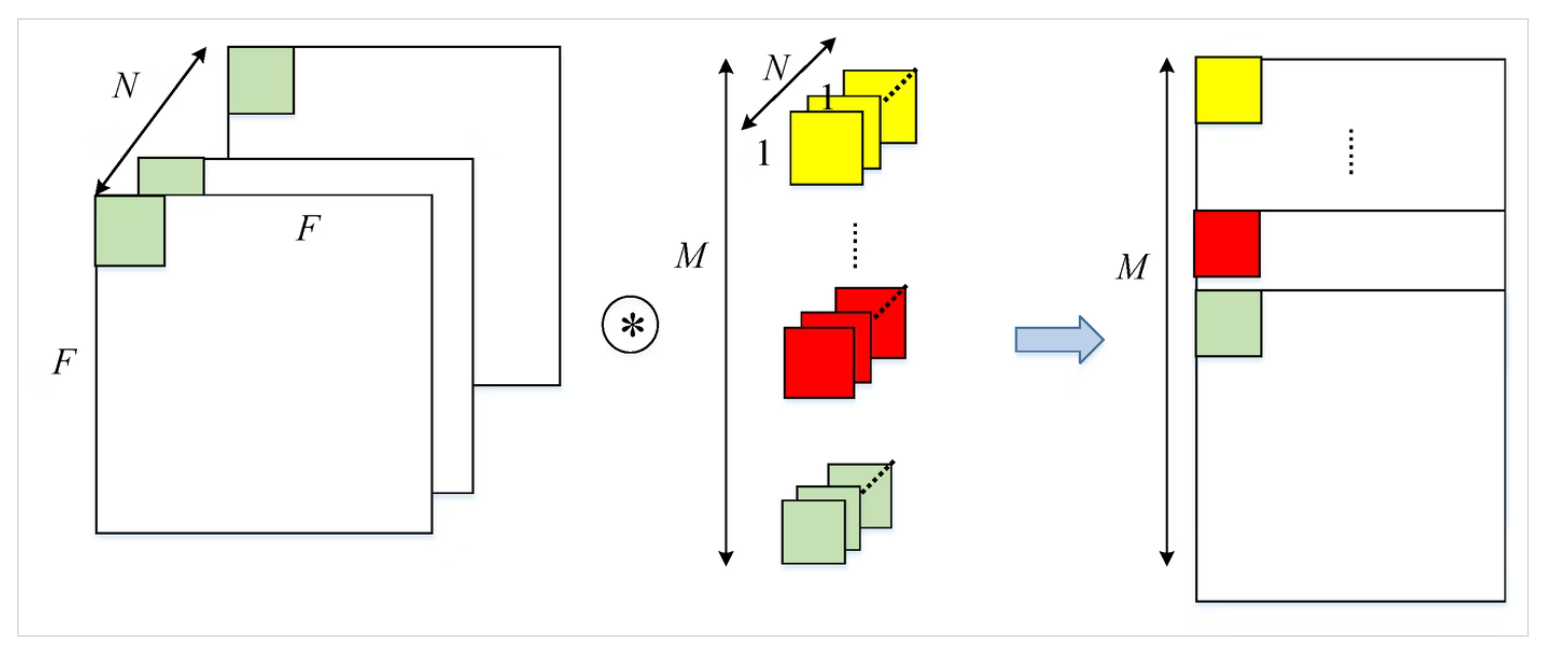
****

図 　Depthwise Separable Convolutionのイメージ

**1チャネルに1つのフィルタが対応しており、各チャネルごとに対応したフィルタで畳み込みする。畳み込み処理はチャネルごとに独立しており、入力と出力のチャネル数は変わらない。パラメータの数は、となる。**

1. **Pointwise Convolution（１×1 Convolution）**

****

**別名Convolutionと言われるように、単純にWindow Sizeが の畳み込みである。の畳み込みということは単純にピクセルごとのチャネル方向の圧縮だと考えると、Window Sizeがであること以外は通常の畳み込みと同じなので、出力のチャネル数はフィルタの数と等しくなる。パラメータの数は、となる。**

1. **どのくらいパラメータの数が小さくなるか？**

**通常のConvolutionは、であるのに対して、Depthwise Separable Convolutionは、となる。よって、Depthwise Separable Convolutionは、だけパラメータの数が減っている。 はフィルタの数、はフィルタの大きさでどちらも1より大きいため、かなり小さくなっていることがわかる。**

1. **DenseNet**

**レイヤー間の情報の伝達を最大化するためにすべての特徴量サイズが同じレイヤーを結合させている。逆伝播を保つため、ある層より前の層の出力を入力とする。レイヤー間が密に結合していることから、DenseNetと呼ぶ。（DeepSquareより）**

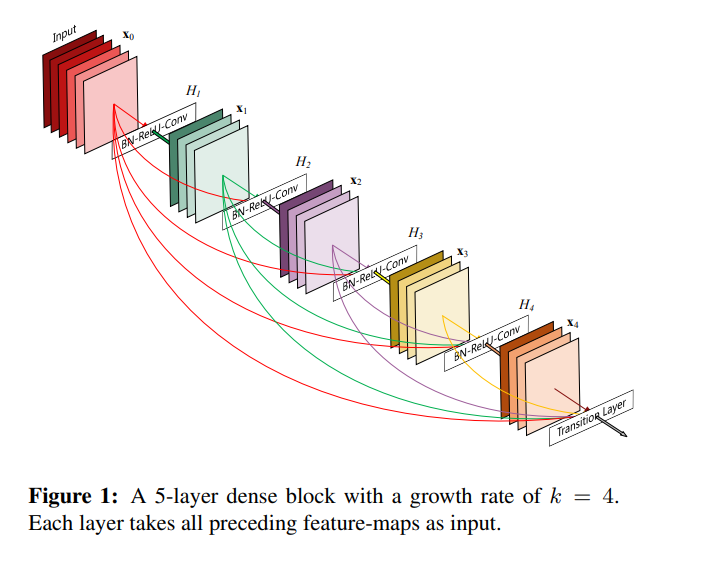


図 DenseNetの構造（https://arxiv.org/pdf/1608.06993.pdf より）

1. **Batch Norm（バッチ正規化：Batch Normalization）**

**Batch Normalizationは勾配消失・爆発を防ぐための手法であり、これまでは下記の手法で対応してきた。**

* **活性化関数を変更する。（ReLUなど）**
* **ネットワークの重みの初期値を事前学習する。**
* **学習係数を下げる。**
* **ネットワークの自由度を制約する。（Dropoutなど）**

**しかし、これまでとは違い、ネットワークの学習プロセスを全体的に安定化させて学習速度を高めることに成功した手法である。（DeepAgeより）**

**Batch Normalizationには、下記の特徴がある。（楽しみながら理解するAI・機械学習入門より）**

* **インプットとなる特徴量だけを正規化するのではなく、レイヤごとにインプットを正規化する。**
* **その際にミニバッチごとの統計量(平均・分散)を使って正規化する。**
* **以下の効果が見込まれる。**

1. **学習が安定する。**
2. **パラメータのスケールや初期値の影響が小さくなる。それにより、高い学習率を設定することが可能になり、学習スピードが速くなる。**
3. **ドロップアウトの必要性を減らすことができる。**
4. **Layer Norm（Layer Normalization）**

**Layer Normalizationでは、1つのサンプルにおける各レイヤーの隠れ層の値の平均・分散で正規化する。**

1. **WaveNet**

**PixelCNN3をベースにした音声波形を生成するためのディープニューラルネットワークの一つである。（Qiitaより）**

**人間に伝える内容を音声化する「音声生成」の技術である。（AIZINEより）**

**WaveNetが登場して従来の音声生成技術を用いるよりも、より人間に近い自然な音声生成を実現できるようになった。**

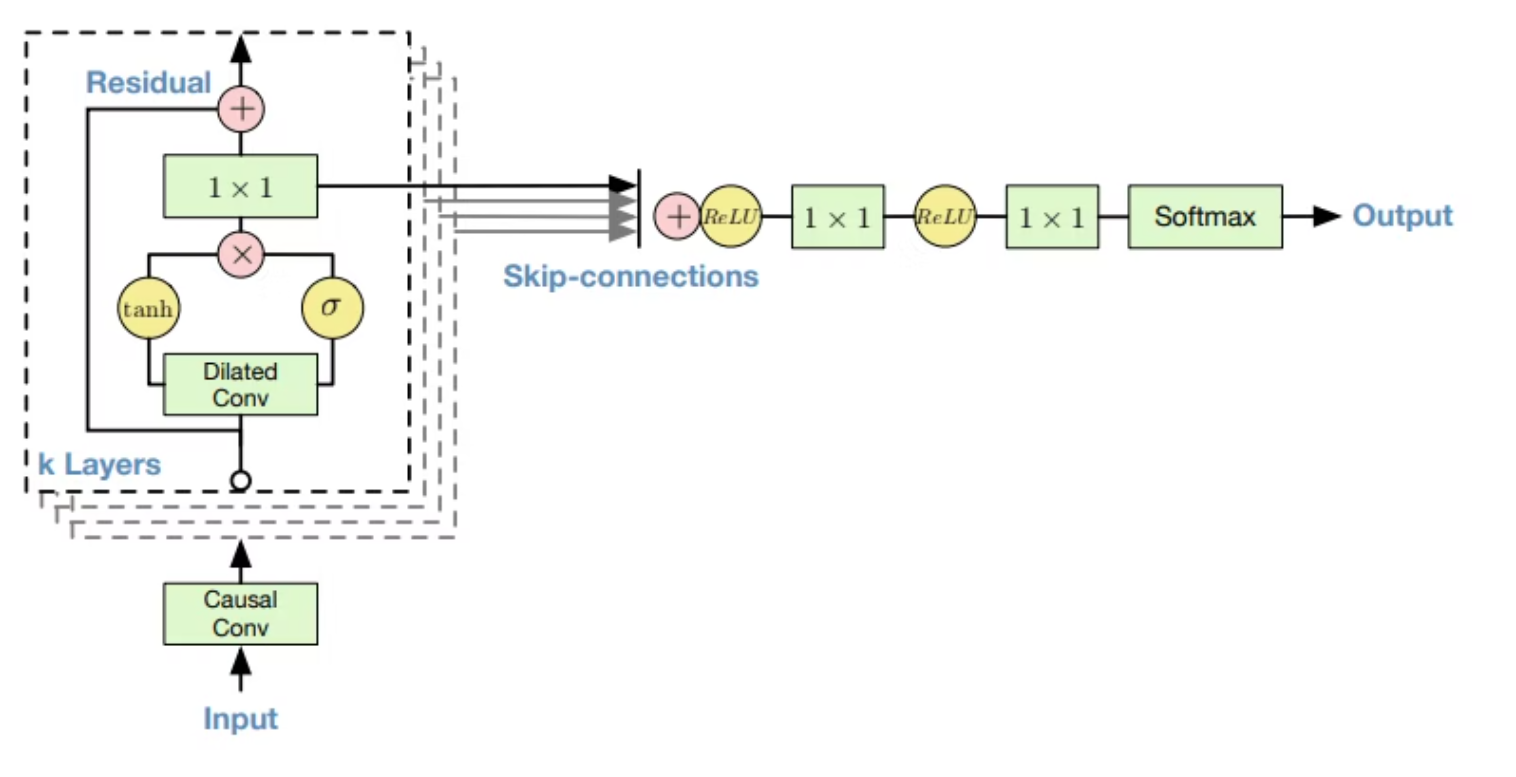
****

図 　WaveNetの全体像（Qiitaより）

1. **ResNet（Residual Network）**

**ResNetは、Microsoft Research(現Facebook AI Research)のKaiming He氏が2015年に考案したニューラルネットワークのモデルである。CNNにおいて層を深くすることは重要な役割を果たす。層を重ねるごとに、より高度で複雑な特徴を抽出していると考えられているからだ。ResNetのアイデアはシンプルで、「ある層で求める最適な出力を学習するのではなく、層の入力を参照した残差関数を学習する」 ことで最適化しやすくしている。ResNetでは、残差ブロックとShortcut Connectionを導入することで実現している。**

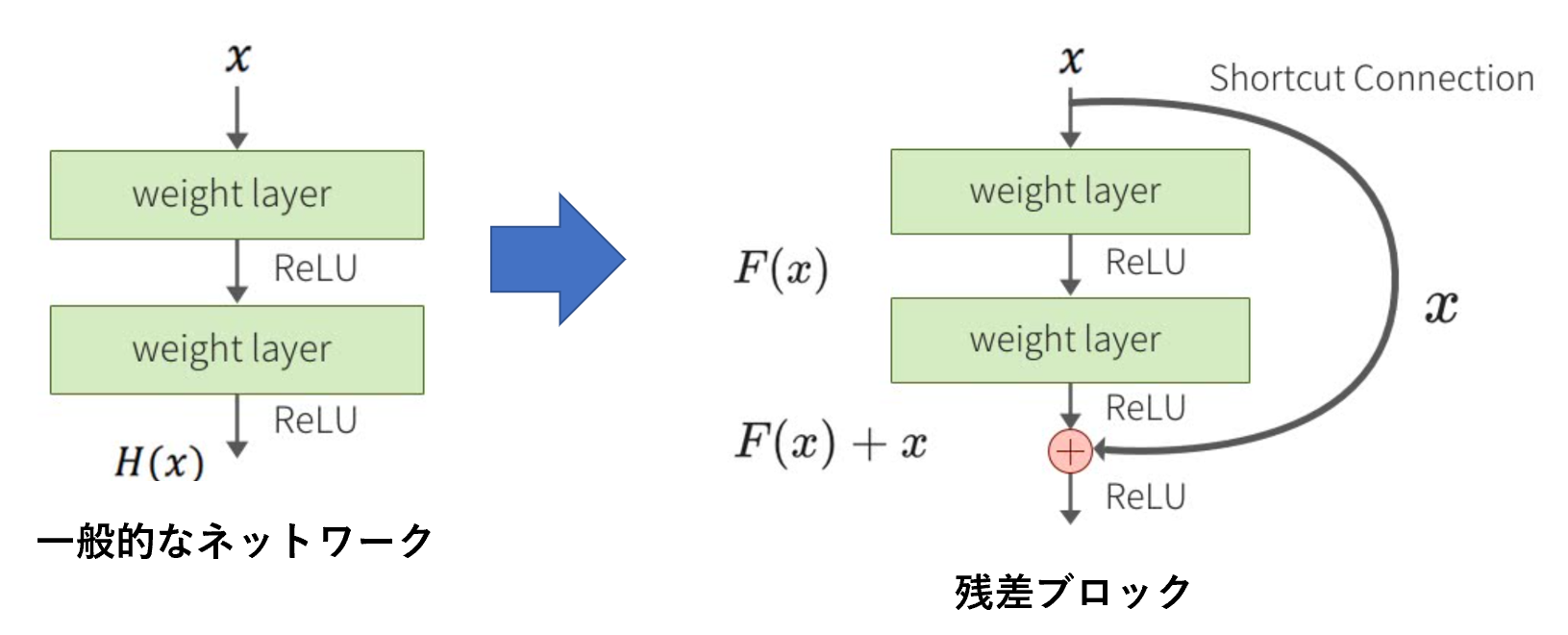
****

図 　残差ブロックのイメージ（DeepAgeより）

1. **EfficientNet**

**EfficientNetは、ConvNetsにおけるネットワークの深さや広さ、解像度などがモデルの性能にどう影響を及ぼすかを調べ、Compound Coefficient(複合係数) というものを導入することで性能を上げた。EfficientNetでは下記の特徴がある。**

* **ICML2019の論文で、新たなモデルスケーリングの「法則」が提案された。**
* **幅、深さ、解像度などを何倍増やすかは、複合係数（Compound Coefficient）を導入することで最適化**
* **EfficientNetではCompound Coefficientに基づいて、深さ・広さ・解像度を最適化したことにより、「小さなモデル」かつ高い精度を達成**
* **モデルが小さい（パラメータ数が少ない）→　効率化（小型化と動作の高速化）**

****

図 　Google AI Blog

1. **物体検知とSS解説（物体検知の基礎とセマンティックセグメンテーション）**
2. **物体検知**

**広義の物体認識タスクで出力されるものとして、下記のものがあげられる。タスクの難易度は下になるほど難しくなる。**

* **分類：画像に対し単一または複数のクラスラベル**
* **物体検知：Bounding Box [bbox/BB]**
* **意味領域分割：各ピクセルに対し単一のクラスラベル**
* **個体領域分割：各ピクセルに対し単一のクラスラベル**

****

図 　代表的なデータセット

1. **セマンティックセグメンテーション**

**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、物体領域を画素単位で切り出し、各画素にクラスを割り当てる手法である。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。例えば、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。（MathWorksより）**

1. **Mask R-CNN**

**Mask R-CNNはICCV 2017で発表された論文 Mask R-CNNで提案された、一般物体検出(Generic Object Detection)とInstance Segmentationを同時に行うマルチタスクの手法である。Mask R-CNN は、検出されたオブジェクトに対してピクセルレベルのセグメンテーションを実行する一般的な深層学習インスタンス セグメンテーション手法である。Mask R-CNN アルゴリズムは、複数のクラスと重複するオブジェクトに対応できる。**

1. **物体検出**

* **画像中の物体らしき領域とその領域が表すクラスを検出することを指す。**
* **物体らしき領域は、画像を特定の領域に区切り、それらをしらみつぶしに評価することで得られる。 物体らしき領域は大量に検出されることになるが、その中から「物体らしさ」が閾値以上の領域のみ、また、領域同士の重なりが閾値以上の領域群から「それらしさ」が最も高い領域のみ、といった条件で絞り込むことで、精度の高い結果が得られる。**

1. **クラス識別**

* **クラスは学習で利用したデータセットに含まれるクラスそれぞれについて、「物体らしさ」を指す確率で得られる。 この確率は上記の「物体らしき領域」毎に得られるが、その領域毎に最も確率が高いクラスのみを採用する。例えば、クラスが1〜3まであり、物体らしき領域Rが得られたとして、そのRに関連するクラス群の確率が [0.1, 0.9, 0.1] と得られたとすると、その領域が示すものはクラス2であると判断する。**

1. **セグメンテーション**

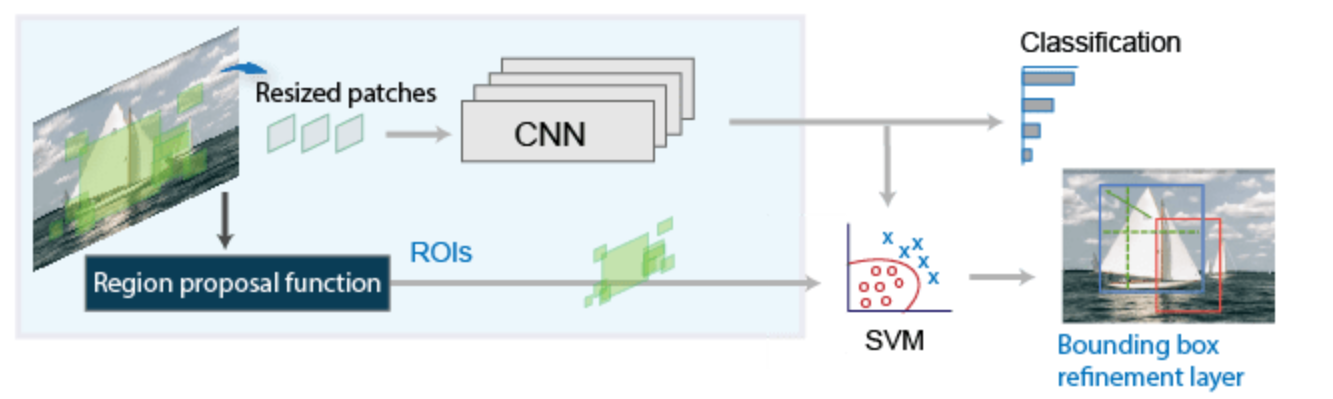
* **画像中のピクセル毎にクラスを検出することを指す。**
* **画像全体をセグメンテーションする場合、結果は以下のような感じで表現される。**
* **Mask R-CNNでは物体検出結果として得られた領域についてのみセグメンテーションする。これは、全ピクセルについてクラス検出よりもだいぶ効率的のように思える。**
* **しっかりと学習できたMask R-CNNのモデルであれば、以下のようなセグメントが得られる。**

1. **セマンティックセグメンテーション**

* **物体領域を画素単位で切り出し、各画素にクラスを割り当てる手法**
* **工業検査や医療画像解析など、精密な領域分割に応用される。**
* **重要な学習データセットに、VOC2012とMSCOCOがある。**
* **主要手法は、完全畳み込みネットワーク（FNC; Fully Convolutional Network）**
* **全ての層が畳み込み層、全結合層を有しない。**
* **画素ごとにラベル付した教師データを与えて学習する→ 出力ノードが多数**
* **未知画像も画素単位でカテゴリを予測する。**
* **入力画像のサイズは可変で良い。**

1. **R-CNN（Regional CNN）**

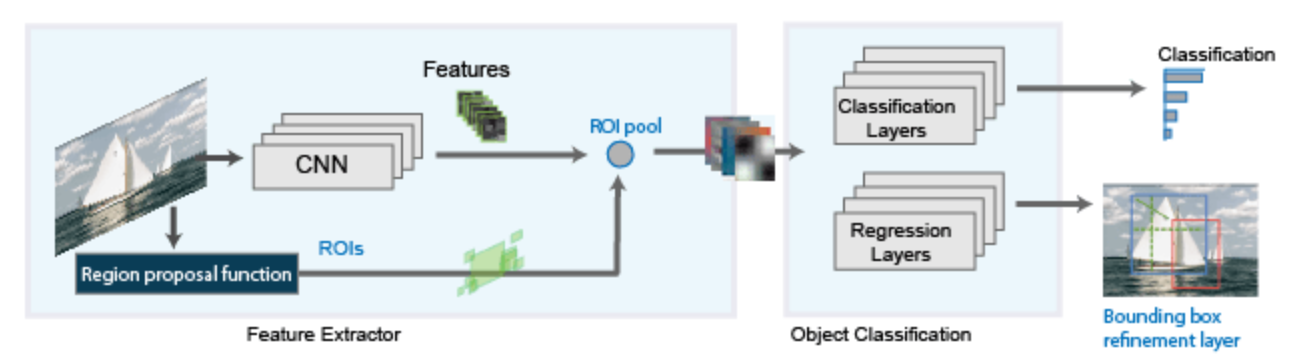
**「オブジェクトの検出」は、イメージ内のオブジェクトを検出して分類するプロセスである。深層学習アプローチの1つであるR-CNN(Regions with Convolutional Neural Networks) では、四角形の領域提案を畳み込みニューラルネットワークの特徴と組み合わせる。R-CNNは2段階検出アルゴリズムである。第1段階では、オブジェクトを含んでいる可能性のあるイメージ内の領域のサブセットを特定する。第2段階では、各領域に含まれるオブジェクトを分類する。R-CNN 検出器はまず、Edge Boxesなどのアルゴリズムを使用して領域提案を生成する。提案領域はイメージからトリミングされ、サイズ変更される。その後、トリミングされてサイズ変更された領域は、CNN によって分類される。最後に、CNN 特徴量を使用して学習したサポートベクターマシン(SVM) によって、領域提案境界ボックスが調整される。（MathWorksより）**

****

1. **高速R-CNN（Fast R-CNN）**

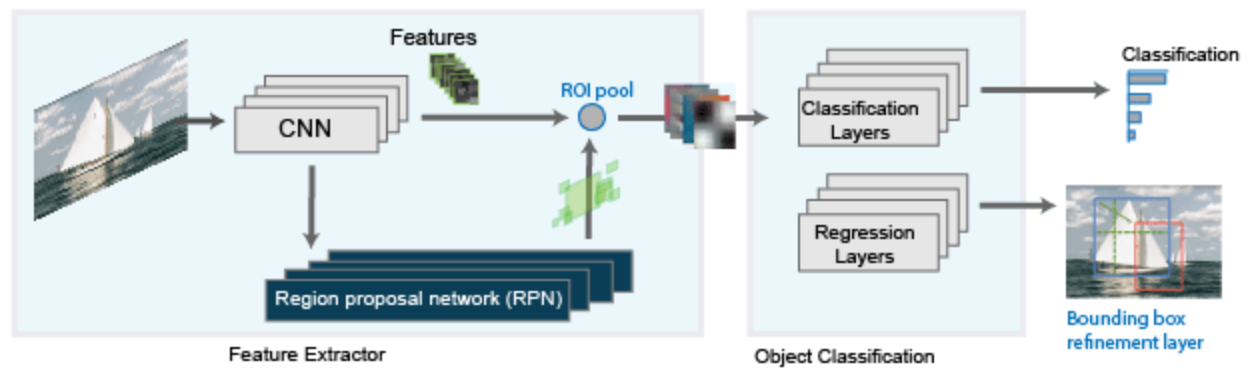
* **Fast R-CNN**

**R-CNN検出器と同様に、Fast R-CNN検出器もEdge Boxesなどのアルゴリズムを使用して領域提案を生成する。領域提案をトリミングしてサイズ変更する R-CNN検出器とは異なり、Fast R-CNN検出器ではイメージ全体を処理する。R-CNN検出器は各領域を分類しなければならないが、Fast R-CNN は各領域提案に対応するCNN特徴量をプーリングする。Fast R-CNN検出器ではオーバーラップする領域の計算を共有するので、Fast R-CNNはR-CNNよりも効率的である。**

****

* **Faster R-CNN**

**Faster R-CNN検出器では、Edge Boxesなどの外部アルゴリズムを使用する代わりに、領域提案ネットワーク(RPN)を追加してネットワーク内で領域提案を直接生成する。RPN はアンカーボックスによるオブジェクトの検出を使用する。ネットワーク内で領域提案を生成する方が高速であり、データに合わせてより適切に調整される。**

****

1. **インスタンスセグメンテーション**

**インスタンスセグメンテーションは、オブジェクトの検出されたインスタンスごとにセグメンテーションマップを生成する拡張タイプのオブジェクト検出である。インスタンスセグメンテーションは、オブジェクトのクラスに関係なく、個々のオブジェクトを個別のエンティティとして扱う。対照的に、セマンティックセグメンテーションは、同じクラスのすべてのオブジェクトを単一のエンティティに属するものと見なす。（MathWorksより）**

1. **RoI Pooling**

**RoI Poolingでは、ある程度畳み込み処理を行ったfeature mapから、region proposalにあたる部分領域をうまく「固定サイズのfeature map」として抽出する。**

1. **FCOS（Fully Convolutional One-Stage Object Detection）**

**アンカーボックス法の精度を超えたアンカーフリーの物体検出である。物体検出モデルには、RetinaNet、YOLOv2、SSDなどのアンカーボックスを使って物体の位置を測定するモデルがある。**

* **アンカーボックス**

**「アンカー ボックス」は、特定の高さと幅の事前定義された境界ボックスのセットである。これらのボックスは、検出する特定のオブジェクトクラスのスケールおよび縦横比を取得するために定義され、通常、学習データセットに含まれるオブジェクトサイズに基づいて選択される。（MathWorksより）**

* **アンカーボックスの利点**

**アンカー ボックスを使用する場合、すべてのオブジェクトの予測を一度に評価できる。アンカー ボックスにより、すべての考えられる位置で個別の予測を計算するスライディング ウィンドウを使用してイメージをスキャンする必要がなくなる。スライディング ウィンドウを使用する検出器の例には、集約チャネル特徴 (ACF) または勾配ヒストグラム(HOG)機能に基づく検出器がある。アンカー ボックスを使用するオブジェクト検出器は、イメージ全体を一度に処理でき、リアルタイムのオブジェクト検出システムを可能にする。（MathWorksより）**

* **アンカーボックスの欠点**
* **ハイパーパラメータ（アンカーボックスのサイズ、アスペクト比、数）の設定が難しい。**
* **アンカーボックスのサイズやアスペクト比が固定されている。**

**形の変化が大きいものや小型の物体に対応できない。**

* **ポジティブサンプルとネガティブサンプルのバランスが崩れる。**
* **ポジティブサンプル：検出したい物体が含まれている領域**
* **ネガティブサンプル：検出したい物体が含まれていな領域**
* **FPN（Feature Pyramid Network）**

**Feature Pyramid Networkは複数の特徴マップを生成する方法で、この方法は高レベルの特徴マップをアップサンプリングして、低レベルの特徴マップに加える事により、低レベル特徴マップの大範囲特徴が強化される。さらに、検出する物体のサイズは特徴マップごとに分業をして、高レベルの特徴マップは大き目の物体検出を担当し、低レベルの特徴マップは小さ目の物体検出を担当している。（Published in LSC PSDより）**

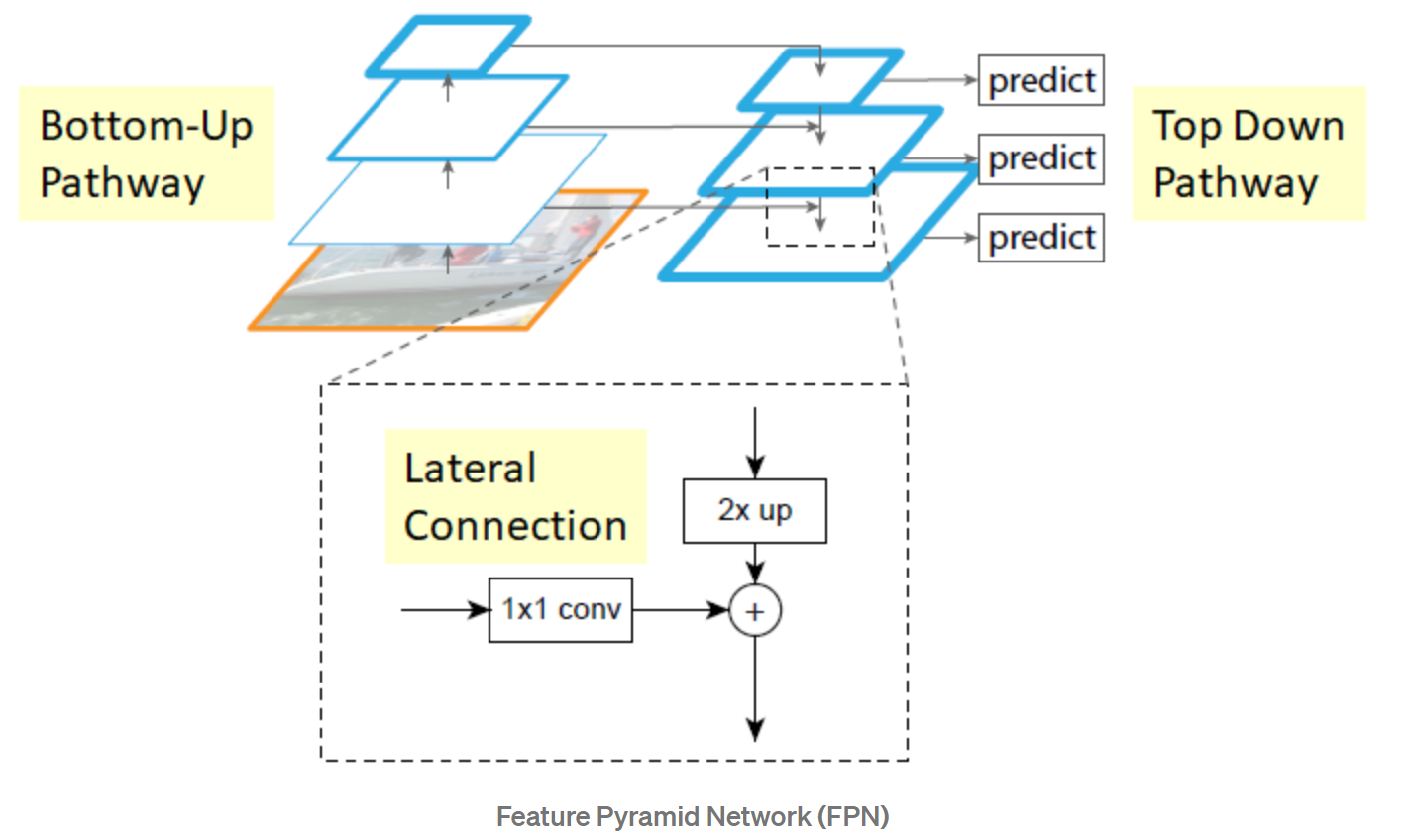


図 Published in Towards Data Science より

1. **TransFomer**

**トランスフォーマー（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。例えば、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。その導入以来、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。**

* **Seq2Seq**

**Seq2Seq(sequence to sequence)は、以下で説明するEncoderとDecoderを備えたEncoder-Decoderモデルを使って、系列データを別の系列データに変換するモデルのことを指す。seq2seqで翻訳をしたり、対話モデルを作ったりすることが可能になる。**

* **Encoder-Decoderモデル**

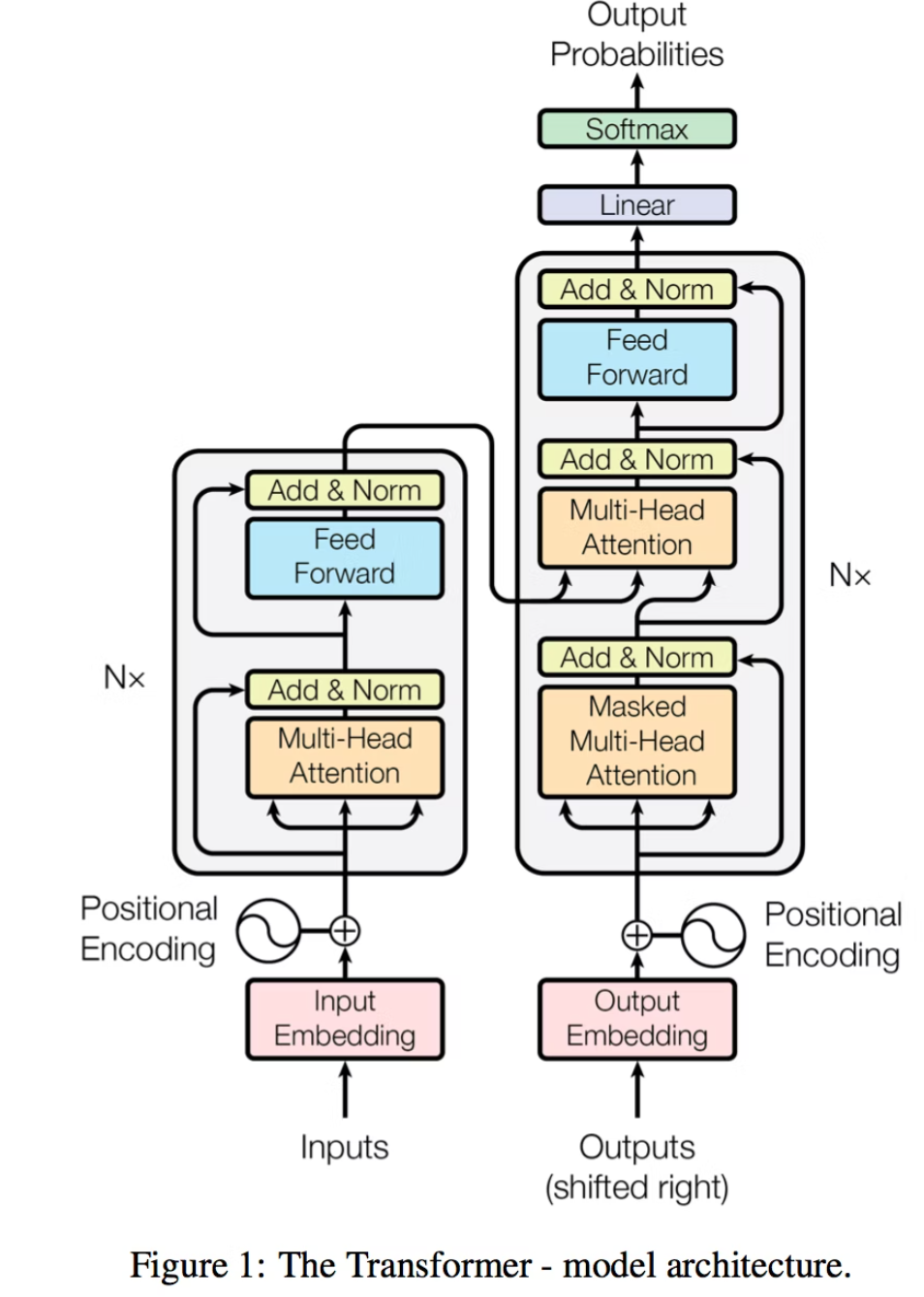
****

図 Transformerの全体像(Qiitaより)

* **エンコーダ**

**層で構成されていて、6層とも同じ構造である。各層はMulti-Head Attention層とPosition-wise全結合層の2つのサブ層で構成されている。それぞれのサブ層の後には残差結合とLayer Normalizationがある。**

* **デコーダ**

**同じく 層で構成されており、6層とも同じ構造である。ただし、各層はエンコーダの2つのサブ層の間にエンコーダの出力を受け取るMulti-Head Attention層を追加した形になっている。あとは、デコーダのサブ層の1つ目はMasked Multi-Head Attentionになっており、これはデコーダが現時刻で予測する単語およびそれ以降の単語たちをカンニングできないようにしている。**

* **Encoder RNN**

**翻訳元の文を読み込み、実数値ベクトルに変換**

* **Decoder RNN**

**実数値ベクトルから、翻訳先の言語の文を生成**

* **Attention**

**Attentionとは、文中のある単語の意味を理解する時に、文中の単語のどれに注目すれば良いかを表すスコアのことである。例えば英語でitが出て来たら、その単語だけでは翻訳できない。itを含む文章中のどの単語にどれだけ注目すべきか？というスコアを表してくれるのがAttentionである。（Qiitaより）**

* **Position-wise全結合層**
* **文字の埋め込みとソフトマックス**

**入力の単語は事前訓練済みの単語分散表現を使ってベクトルに変換する。また、デコーダの最終出力は、ニューラルネットワークに入力されソフトマックスで一番高い確率を示した単語を出力するが、そのニューラルネットワークに使う重みは最初の単語の分散表現を使うときの重みと同じものを使用する。**

* **位置エンコーディング**

**単語の位置関係を捉えられる再帰(recurrence)や畳み込み(convolution)を使っていないため、このままだとTransformerにとって単語の順番は関係なくなってしまっている。そこで登場するのが、位置エンコードという考え方である。原理は単純で、最初にこのモデルに単語の分散表現を入力するときに単語位置に一意の値を各分散表現に加算する。単語に一意な値を出力するような関数を使ってあげることで実現できる。**

1. **BERT（Bidirectional Encoder Representations from Transformers）（Ledge.aiより）**

**LINE Clova、Amazon Echoなどのスマートスピーカーやウェブ上のカスタマーサービスに見られるチャットボットはどれも自然言語処理というAI技術が用いられている。この自然言語処理においては、2018年10月にGoogleがBERTという手法を発表し、「AIが人間を超えた」と言わしめるほどのブレイクスルーをもたらした。**

* **BERTとは**

**BERTとは、Bidirectional Encoder Representations from Transformers の略で、「Transformerによる双方向のエンコード表現」と訳され、2018年10月にGoogleのJacob Devlinらの論文で発表された自然言語処理モデルである。翻訳、文書分類、質問応答など自然言語処理の仕事の分野のことを「（自然言語処理）タスク」と言うが、BERTは、多様なタスクにおいて当時の最高スコアを叩き出した。BERTの特徴として「文脈を読むことが可能になった」ことを挙げられる。BERTにはTransformerというアーキテクチャ（構造）が組み込まれており、文章を双方向（文頭と文末）から学習することによって「文脈を読むこと」が実現した。**

* **自然言語処理とは**

**自然言語処理（Natural Language Processing）とは、人間の言語（自然言語）を機械で処理することである。具体的には、言葉や文章といったコミュニケーションで使う「話し言葉」から、論文のような「書き言葉」までの自然言語を対象とし、それらの言葉が持つ意味を解析する処理を指す。また、自然言語には意味が一意に定まらない曖昧性がある。自然言語処理は、その曖昧性を含め、自然言語で書かれたテキストデータを処理するための技術である。自然言語処理によって、翻訳、感情分析（文章の意見がポジティブかネガティブか分析）、文書分類（文章のカテゴライズ）などが可能になった。**

* **BERT導入の背景**

**BERT導入の背景として、「多様化する検索クエリへの対応」が挙げられる。検索エンジン以外でも、声でコンピュータやデバイスを操作するVUI（Voice User Interface)の普及もBERT導入の背景として考えられる。「天気情報を調べてもらう」「アラームを設定してもらう」などの自然言語の命令は長文で複雑化しやすい傾向にあった。こうした中で文脈を読めるモデルの需要が高まっていた。また、一般に、タスクに応用するためには、構造ごと修正する必要があったが、BERTはその必要がなく容易に別のタスクに応用できることや、学習データの不足を克服した点も注目されている。**

* **BERTの仕組み**

**自然言語処理は、単語を高次元のベクトルに置き換える分散表現という技術を用いて入力される。単語データの並びのことを「シーケンス」と言い、これはいわゆる文章にあたるが、BERTは入力されたシーケンスから別のシーケンスを予測する。BERTは事前学習モデルであり、入力されたラベルが付与されていない、つまり名前がついていない分散表現をTransformerが処理することによって学習する。実際には、TransformerがMasked Language ModelとNext Sentence Predictionという2つの手法を同時進行で行うことで学習する。**

* **Masked Language Model**

**従来の自然言語処理モデルでは、文章を単一方向からでしか処理できなかった。そのため、目的の単語の前の文章データから予測する必要があった。しかし、先述の通りBERTは双方向のTransformerによって学習するため、従来の手法に比べ精度が向上した。それを実現しているのがMasked Language Modelである。**

* **Next Sentence Prediction**

**Masked Language Modelにおいて単語に関しての学習はできるが、文単位の学習はできない。そこで、2つの入力文に対して「その2文が隣り合っているか」を当てるよう学習する。これにより、2つの文の関係性を学習できる。Next Sentence PredictionによってBERTは文章も考慮した、より広範的な自然言語処理モデルとして機能できる。**

* **導入方法**

**BERTによってタスクを解くためには、既存のタスク処理モデルに接続する。一定のタスクのために学習されたアーキテクチャを別のタスクに活かす転移学習という学習手法がある。BERTは既存のタスク処理モデルに転移学習し導入する。**

* **BERTの特徴**
* **文脈理解**

**BERT以前にはELMo、OpenAI GPTといった言語処理モデルが存在した。ELMoは浅い双方向モデルであり、OpenAI GPTは未来の単語しか予測することができない単一方向モデルだったので、文脈読むことができなかった。BERT導入以前のGoogle検索では「to」のような文と文の関係を結ぶ言葉を処理できなかった。Googleが紹介している一例では、「2019 brazil traveler to usa need a visa（2019年 アメリカへのブラジル旅行者はビザが必要）」と検索したとする。この場合、ユーザーはブラジルからの旅行者はアメリカに行く際にビザが必要かどうかを知りたいはずである。ところが、BERT導入前は「to」を処理できないので「ブラジルへのアメリカ人旅行者」と解釈し、ニーズに合致しない検索結果を表示されていた。BERT導入後は、「アメリカへのブラジル人旅行者」と解釈ができるようになり、アメリカ大使館がブラジル人旅行者向けに公開しているビザ情報のページを上位表示するようになった。**

* **汎用性**

**BERTの特徴として汎用性の高さが挙げられる。従来のタスク処理モデルは特定のタスクにのみ対応している。しかし、BERTはモデルの構造を修正せずとも、様々なタスクに応用できる。既存のタスク処理モデルの前に接続（転移学習）するだけで、自然言語処理の精度を向上させられる。**

* **データ不足の克服**

**BERTは従来のモデルとは違い、ラベルが付与されていないデータセットを処理できる。現状、自然言語処理タスクのためにラベルが付与されたデータセットが少なく、手に入れることが困難である。またラベルを付与するにも時間と労力がかかる。一方、インターネットが普及した現代では、ラベルが付与されていないデータは大量に存在し、容易に獲得できる。そのため、BERTはこのようにデータ不足を克服した点で評価されている。**

* **性能の数値評価（ベンチマーク）**

**自然言語処理モデルの性能はGLUE（The General Language Understanding Evaluation）という指標で評価できる。GLUEはあらゆる自然言語処理タスクにおけるスコアを算出するベンチマークである。**

* **BERTの課題**

**大きなブレイクスルーをもたらしたBERTですが、いくつかの課題がある。その1つに計算する際のパラメータが多く、層が厚い「巨大なモデル」である点があげられる。巨大であれば、学習に時間がかかる、大きなメモリ量が必要になるなどのデメリットが生まれる。一般に配布されているBERTモデルでは、16個のTPU（機械学習に特化したGoogleの自社開発プロセッサ）を使って学習に4日かかると言われている。また、精度を上げるためにパラメータを増やしても一定のところで精度が逆に落ちてしまうという構造の複雑さがあった。それらの課題を克服するべく、BERTの軽量版ALBERT（A Lite　BERT）が発表された。ALBERTでは、層ごとにパラメータを共有する、入力したベクトルの処理方法を変えることで精度を落とさずにパラメータの数を減らすことができた。パラメータを減らすことによってよりシンプルで現実的なモデルになった。さらに、先述のNext Sentence Predictionは文脈解読のための手法だが、実際に文脈解読の精度を向上しているかという点で疑問視されていた。ALBERTではSentence Order Predictionという別の手法を用いて文脈解読の精度を向上させた。**

* **BERTの現状と展望**

**Googleは2019年10月25日、公式ブログにおいてBERTを検索エンジンに導入したと発表した。発表時は米Googleの英語検索のみ導入されていたが、2019年12月10日、日本語を含む70以上の言語に展開された。また、2019年11月25日に株式会社日立ソリューションズは独自技術の文書分類アプリ「活文 知的情報マイニング」にBERTを導入したと発表し、現在世界的に導入が進んだ。今後、BERTの文脈理解によってスマートスピーカーやチャットボットの強化が進められると考えられる。BERTには先述のALBERTだけではなく、XLNetや中国語向けに改良されたERNIEなどの進化系がいくつか開発されている。現在BERTは研究が盛んに行われているホットな分野である。人間の能力を超えたと言われることもあるが、まだ人間のように振舞うことはできでいない。**

1. **GPT（Generative Pretrained Transformer）（AI Academy Mediaより）**

* **GPT-3とは**

**GPT-3（****Generative Pretrained Transformer）はOpenAIが開発している1750億個のパラメータを使用した『文章生成言語モデル』である。**

* **GPTの歴史**

**GPT-3はGPTの3番目のバージョンで、これまでGPT→GPT-2→GPT-3とバージョンアップしている。2020年8月8日現在バージョンは3である。OpenAIが作ったGPTの初代バージョンである、GPTは2018/6/11日に1.1億パラメータを使用しており、バージョンが上がるごとにパラメータ数が増えている。（GPT-2（2019/2/14）では15億パラメータ、GPT-3（2020/5/28）1,750億パラメータ）**

* **GPT-3でできること**
* **ReactJSのコードを自動生成（自然言語からソースオードを自動生成）**
* **デザインの生成**
* **ToDOアプリのコード（React）を自動生成**
* **コードから仕様を英語で説明するツール**
* **データセットと必要な出力を記述するとKerasのモデルのコードを生成**
* **文章の自動生成（提案書、稟議書、マニュアル、仕様書など）**
* **GPT-3の仕組み**

**GPT-3の構造はGoogleによる言語モデルである「Transformer」を基本としている。GPT-3の基本的なアーキテクチャ、データ、学習方法はGPT-2を踏襲しており、それぞれをスケールアップさせている。まずファインチューニングとfew, one, zero-shotを整理する。（Zennより）**

* **Fine-Tuning**

**広く利用されるアプローチ。対象タスクで高い性能を得やすいが、タスク固有のデータが必要、与えられたデータに過学習してしまうなどの懸念がある。GPT-3はタスクを未知とした場合の性能に関心があるため、本論文ではファインチューニングは検証しない。**

* **Few-Shot**

**モデルのパラメータは固定したまま、少量のデモンストレーションから予測を行う方式。タスク固有のデータが少量で済み、過学習の心配がない。一方でファインチューニングほどの性能は発揮できない場合が多い。**

* **One-Shot**

**Few-shotでデモンストレーションが一つだけ与えられるケース。人がタスクに取り組むシチュエーションに近いとして個別に取り上げている。**

* **Zero-Shot**

**タスクの指示のみが与えられる。最も汎用だが最も難しい設定。**

* **GPT-3のできないところや問題点（NTT Dataより）**

**このようにGPT-3ではこれまで以上に文章を生成する事が出来るようになっているが、「空はなぜ青いのか？」のような複雑な質問にはうまく答えられないようである。他にも、GPTは意味を理解せず、喜びや怒りなどの感情や、常識も備えていない。また、機械学習アルゴリズムの開発に置いてデータセットに含まれる偏見や差別は大きな問題である。（AI倫理分野の課題）**

* **文章生成と文章の関係性に関する課題**

**GPT-3で生成する文章は過去の情報を元に単語を並べ、文法的にそれらしい文章を作っていることにすぎないため、長い文章を生成すると、同じ意味の単語を繰り返し、結論が矛盾した文章を生成してしまうことがある。また、人間のような常識を持っていないため、社会通念上ふさわしくない文章を作り出すこともある。さらに、2文間の関係性を比較する能力に課題がある。具体的には、2文の中で使われている単語が同義であるかどうかを比較することや2文の中で一方が他方を暗に意味するかどうかを比較することが苦手である。これは、GPT-3が前の文章にある単語との関係性をパターン学習するというところに起因しており、後ろの単語から前の単語への関係性が問われる問題を解くことが難しいとされている。しかし、この2文間の関係性を理解する能力は自然言語処理分野では難しい課題のひとつと言われている。**

* **推論における課題**

**GPT-3は物理現象の推論問題に弱いのではないかという報告がある。有名な事例として「冷蔵庫にチーズを入れて溶けるか？」という質問に対して、GPT-3は正しく回答することができない。さらに、人間であれば「意味が分からない」と答えるようなナンセンスな質問に上手く答えられない。例えば「太陽の目はいくつありますか？」という質問に対して、「太陽の目は1つです」と答えてしまう。このように、人間が持つ常識や推論に基づく自然言語理解を実現しようとする試みは、それが実現可能であるか？ということも含めて大きな議論のひとつに挙げられる。また、推論や常識を求められるタスクに対して、テキストと知識ベースを融合させ、深層学習を行うことで人間並みの自然言語理解を獲得しようとするアプローチはトレンドのひとつになっている。**

* **膨大な運用コストの課題**

**GPT-3の運用に掛かるコストの問題がある。具体的には、GPT-3の土台とのなるインフラ整備コストや膨大なパラメータを使用した事前学習などの運用コストがあり、それらを踏まえると気軽に実運用に踏み切れないのではないかという点が指摘されている。この課題に対しては、今後の事前学習のアルゴリズムの発展により、より小さなコストでGPT-3が利用できるようになる可能性もある。**

1. **音声認識**

**音声認識はスマホやゲームコンソール、スマートウォッチなど、日常に浸透してきている。ホームオートメーションにも欠かせません。近年、コンタクトセンターなどのカスタマーサービスにおいて、AI技術を導入する動きが広がっている。その際、特に大きな効果を期待できるのが、高度な機械学習（深層学習）によって発達したAIの音声認識能力の活用である。**

* **機械学習機能を備えた音声認識サービスのメリット（コンタクトセンターの森より）**
* **音声アシスタント機能**

**音声認識技術は、すでにさまざまな形で活用されている。Appleの「Siri」やAndroidの「Googleアシスタント」などの音声アシスタント機能を活用することで、ユーザーは手で操作せずとも必要な情報を検索したり、音楽を再生したりと、さまざまな操作が行える。これはAmazonの「Alexa」など、スマートスピーカーも同様である。こうした音声アシスタント機能は単に操作の利便性を高めるだけに留まらず、従来とは違った新鮮なユーザーエクスペリエンスをユーザーに提供することで、サービスに付加価値を与えている。**

* **音声の文字起こし作業**

**録音した音声の文字起こし作業なども、音声認識技術の活用例として挙げられる。会議や打ち合わせ、お客様対応の録音データなどの文字起こしを人力で行うと、その入力作業には大変な労力と時間がかかってしまう。しかし、音声認識技術を使えば、こうした作業を自動化できるため、疲労による入力ミスを削減し、大幅な業務効率化も実現される。**

* **翻訳アプリ**

**さまざまな外国語をリアルタイムに翻訳する翻訳アプリなど、音声認識技術の活用可能性は多岐に渡る。音声認識技術をビジネスで活用することによって、業務効率化・サービスの質改善・顧客満足度の向上などのメリットが見込める。**

* **音声データ**

**音は空気の振動による波のことで、振幅（音の大きさ）と波長（音の高さ、周波数）の2つの要素から成り立っている。**

* **量子化**

**量子化とは、信号の大きさを離散的な値（デジタル値）で近似して表すことである。**

**大きさを表す際に用いるビットの数を量子化ビット数（ビット深度）と呼び、これが大きいほど振幅を細やかに表現できる。特に音声信号については、振幅を-1〜1の間の値に収まるように近似して表現する。またビット深度には、多くの場合16bit（65536段階）が用いられる。（SkillUp AIより）**

* **標本化**

**標本化（サンプリング）とは、連続信号（アナログ信号）を⼀定の間隔を空けて測定し、離散信号（デジタル信号）として記録することである。測定する周期の逆数をサンプリング周波数（Sf）と呼び、 Hz のとき1 秒間に 回波形のサンプルを取ることを表す。（SkillUp AIより）**

* **スペクトル解析**

**音声データの分析においては、音声波形そのものを分析するということは少なく、通常は周波数成分に分解した状態で分析を行う。フーリエ変換を用いて音声波形を周波数成分に分解したものをスペクトル（Spectrum）と呼び、スペクトルを解析することをスペクトル解析と呼ぶ。**

* **フーリエ変換**

**音声波形を周波数成分に分解する際にはフーリエ変換を用いる。コンピュータで扱われる音声信号は離散信号（デジタル信号）であるため、フーリエ変換の中でも離散フーリエ変換を用う。元の音声信号を、スペクトルをとおくと離散フーリエ変換は次のように定式化される。**

**は元の音声信号の番目の値、はサンプリング周期（サンプリング周波数の逆数）であり、は1サンプルあたりの時間を表す。一見すると複雑ですが、この式は波形に関する公式（①）とオイラーの公式（②）の知識があれば理解することができる。**

1. **周波数・角速度・角度・時間の関係式波形に関する公式**
2. **複素指数関数と三角関数の間の関係式（オイラーの公式）**

**この2式を踏まえると、最初のフーリエ変換の式は「ある角速度における周波数成分は、信号のサンプルに角速度のコサイン波・サイン波を掛け合わせたものを、全サンプル分足し合わせることにより求められる」ということを表している。**

* **スペクトルとスペクトログラム**

**フーリエ変換の結果は実部（コサイン波）と虚部（サイン波）の複素数となり、その絶対値を取るとスペクトルを見ることができる。スペクトルを時間方向に並べたものを、スペクトログラム（Spectrogram）またはソナグラム（Sonagram）と呼ぶ。スペクトルは2次元（周波数×振幅）、スペクトログラムは3次元（周波数×時間×振幅）のデータである。**

* **FFT（高速フーリエ変換）**

**スペクトルを算出するための数値計算ライブラリでは、高速フーリエ変換（FFT: Fast Fourier Transform）という手法が用いられている。また、音声波形を短い区間に切り分け、窓関数を掛けながら FFTを実行することを短時間フーリエ解析という。短時間フーリエ解析を行うとソナグラムが出力される。短時間フーリエ解析を行うための関数では、通常、n\_fft（フレーム長）・hop\_length（フレーム間隔）・win\_length（窓関数の長さ）の3つの引数を受け取る。n\_fft は切り出す区間の長さ、win\_length は切り出した波形に掛ける窓関数の長さを表す。基本的には同じ値を用いますが、win\_length を n\_fft より小さく設定すると、波形の特徴を強調することができる。また hop\_length は、波形を切り出す間隔を表す。これを小さくすると、出力されるソナグラムが時間方向に長くなる。n\_fftやwin\_lengthを大きくすると周波数分解能が細かくなるが、時間分解能は粗くなってしまう。逆に時間分解能が細かすぎると、低い音（波長の長い信号）を捉えることができない。低い音を解析するためには、その波形を少なくとも3波含むようにパラメータを設定する必要がある。**

* **スペクトルから派生した特徴量**

**音声認識・音声合成のモデルでは、入力データとして、スペクトルから取り出した情報がよく用いられる。**

* **ケプストラム**

**対数スペクトルを波形とみなしてフーリエ変換したものをケプストラム（Cepstrum）と呼ぶ。また、ケプストラムの横軸をケフレンシー（Quefrency）と呼ぶ。ケフレンシーの低い部分を取り出して逆フーリエ変換を行うと、スペクトル包絡（声道の特性を表す特徴量）を取り出すことができる。**

* **メルスペクトル**

**人間の聴覚には、高い音ほど音高（音の高さ）の変化に鈍感になるという特性がある。その知覚特性を考慮した尺度をメル尺度と呼ぶ。メル尺度における数値の差は、人間が感じる音高の差とほぼ同じになっている。また、スペクトルの周波数をメル尺度に変換したものをメルスペクトルと呼ぶ。**

* **MFCC**

**メルスペクトルからケプストラムを算出し、さらに対数をとったあとに、離散コサイン変換を行ったものを MFCC（メル周波数ケプストラム係数）と呼ぶ。MFCC の低い方から 12～18 次元ほどを取り出したものは、スペクトル包絡のような声道特性（話者の特徴）を表す。**

1. **CTC**

**CTC（Connectionist Temporal Classification）は文字認識や音声認識でよく使われる手法で、LSTMやRNNと組み合わせて使用される。文字認識や音声では一つの文字の横幅や、一つの音素の時間長さが可変である。そこで、デコーダ側で同じ文字が連続した場合に消し込むことでこの可変性の問題を解決する。**

* **音声認識の仕組み（Ledge.aiより）**

**音声認識の技術は下記の4つの過程を経て機械は認識する。**

* **音響分析**

**音響分析では、入力された音声データの音の強弱や周波数、音と音の間隔、時系列などさまざまな特徴量を抽出し、音響モデルで扱いやすい（コンピュータが認識しやすい）データに変換する。**

* **音響モデル**

**音響モデルでは、音響分析により抽出された特徴量が、どの記号（音素や単語）にどれほど近いのかを学習したパターンと照らし合わせ、整合率を計算する。例えば、「ありがとう」と音声を入力した場合、音声分析により抽出された特徴量を用いて「A-R-I-G-A-T-O-U」という音素になるように、音声を正しい文字にマッチングさせる。**

* **発音辞書**

**言語モデルでは、膨大な量のデータから単語のつながりを予測判定し、より正確な文章を組み立てる。あらかじめ蓄積したデータから使用する単語の出現率を算出し、単語を文章化する。言語モデルで主に利用されるモデルが「隠れマルコフモデル」である。これは、ある文字列に続く直後の文字の出現しやすさをパターン化し、それらの出現確率を定義する。**

* **言語モデル**

**発音辞書では、音声の最小単位の「音素」ごとにモデル化されている膨大なデータベースから、音の組み合わせをピックアップし、「単語」として認識させる。音の組み合わせを「単語化」し、音声モデルと言語モデルを結びつける役割をする。**

* **CTC（Connectionist Temporal Classification）**

**ほとんどの音声データベースには時間情報がなく、発話内容しか与えられていない。これは、音素の時間情報を同定する作業が非常に高コストだからである。このため、ニューラルネットを用いた音声認識は、これまでは学習時にHMM(隠れマルコフモデル)を用いた従来の音声認識を援用し、はじめに各音素の区間を音声認識によって推定する必要があった。これに対し、近年ニューラルネットだけで音声認識を可能とする枠組がいくつか提案されている。CTC (Connectionist Temporal Classification) は、そのようなアルゴリズムの1つである。CTCにおける重要な発明は次の2点である。**

* **ブランク(blank)と呼ばれるラベルの導入**

**CTCの学習では、通常の交差エントロピー損失ではなく、縮約すると正解音素列と一致するような全ての音素列の確率の和をネットワークの出力の「望ましさ」と考え、その対数にマイナスを付けた関数を損失関数とする。例えば、縮約すると /h a i/ という音素列と一致するような長さ5の音素列を全て挙げるとなる。ただし\_はどの音素でもない記号で、ブランクと呼ぶ。音響的特徴の確率分布モデル＋音素の時系列モデル＋言語モデルといった、複数のモジュールを組み合わせて構築していた従来の音声認識システムと異なり、CTCでは音響的特徴から音素／音節／単語を直接出力する音声認識システムを構築できる。この文書では、小規模なデータからでも構築できるよう、CTCを用いた音素認識の例を示すことを目的としている。**

* **前向き・後ろ向きアルゴリズム(forward-backward algorithm)を用いたDNNの学習**

1. **DCGAN**

**DCGAN（Deep Convolutional GAN)は、2015年にA.Radfordらによって発表された敵対的生成ネットワークの一種であり、生成ネットワーク（generator）と識別ネットワーク（discriminator）の2つのネットワークに畳み込みニューラルネットワーク(CNN)を用いたモデルのことである。DCGANにおける生成モデルは100次元の一様分布Zを入力とし、転置畳み込みによって徐々に画像空間へ投影していく仕組みである。また識別モデルは同じくプーリング層を使わずに畳み込みによってダウンサンプリングしていき、活性化関数にはReLUの代わりに漏洩ReLU（Leaky ReLU）を使用する。プーリング層や全結合層を使わずにCNNによって学習を進めることで、通常のGANよりも鮮明な画像の生成が可能になった。さらにDCGANによる生成は、その入力のベクトル空間的な性質が極めて良いことが明らかになり、値の近い入力同士なら似たような画像を生成し、二つの入力の間の値によって生成される画像は二つの画像の意味的な中間となる。これにより、入力を適切に調整することでより高次な特徴レベルの画像をコントロールすることができる。（ウィキペディアより）**

* **GAN（Qiitaより）**

**GANは、ニューラルネットワークのモデルにトレーニングデータの分布を学習させるためのフレームワークで、学習した分布から新しいデータを生成できる。GANは、2014年にイアン・グッドフェロー氏によって考え出され、論文 Generative Adversarial Nets で最初に発表された。GANは、生成器(generator)と識別器(discriminator)の2つのモデルで構成されている。生成器は、トレーニング画像を元に「偽の」画像を生成する。一方、識別器は、画像が実際のトレーニング画像であるか、生成器からの偽の画像であるかを判別する。生成器は識別器が判断できないような良い偽物を生成することで学習し、識別器は実際の画像と偽の画像を正しく分類するように学習する。この学習のゴールは、生成器がトレーニングデータのように見える完全な偽物を生成しているときであり、識別器は、生成器の出力が本物か偽物かを常に50％の信頼度で推測する。を画像を表すデータとする。は、が本物の画像である確率（スカラー値）を出力する識別器です。ここでは、画像を扱っているため、への入力はCHW（色、縦、横）で、このチュートリアルでは の画像である。)のスカラー値は、が本物の画像の場合は大きい値になり、生成器が作製した画像の場合は小さい値になる。は通常の2クラス分類器と考えることもできる。次に生成器を見ていく。を標準正規分布からサンプリングされた潜在空間ベクトルとする。 )は、潜在ベクトルをデータ空間にマッピングするジェネレーター関数を表す。 の目標は、トレーニングデータの取得元の分布を推定して、推定された分布から偽のサンプルを生成できるようにすることである。**

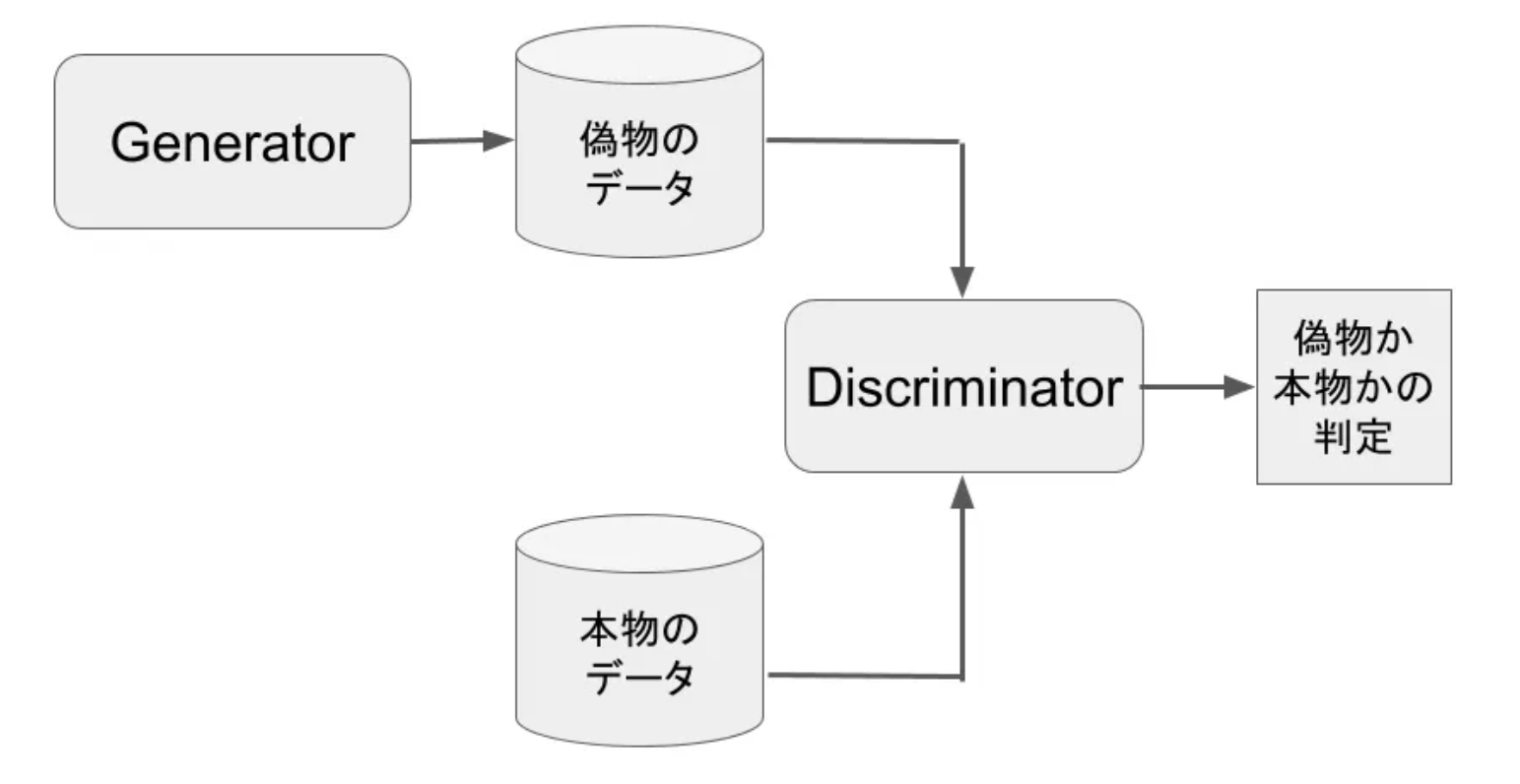
****

図 　GANの構造

* **DCGAN（Qiitaより）**

**DCGAN(Deep Convolutional GAN) は、上記のGANを拡張したものだが、識別器と生成器でそれぞれ畳み込み層と畳み込み転置層を使用する点が異なる。これは、Radford氏らによって、論文「Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks」で最初に説明された。識別器は、strided　convolution　層（畳み込み層の一種）、Batch Normalization 層、および LeakyReLU activation(活性化関数) で構成されている。入力はの画像で、出力は入力データ（画像）が実在の写真である確率（スカラー値)である。生成器は、convolutional-transpose 層、Batch Normalization 層、および ReLU activation(活性化関数) で構成される。入力は、標準正規分布から抽出された潜在ベクトル、出力はからなるRGB画像である。論文では、オプティマイザーのセットアップ方法、損失関数の計算方法、モデルの重みの初期化方法に関するヒントもいくつか示している。**

1. **Conditional GAN**

**CGAN (Conditional GAN)は2014年にarXivで公開された論文 Conditional Generative Adversarial Netsで提案された生成手法で、生成する画像を明示的に書き分けるために、訓練時に教師データのカテゴリ（ラベル）情報を用いてやろう、というのがconditional GANである。**

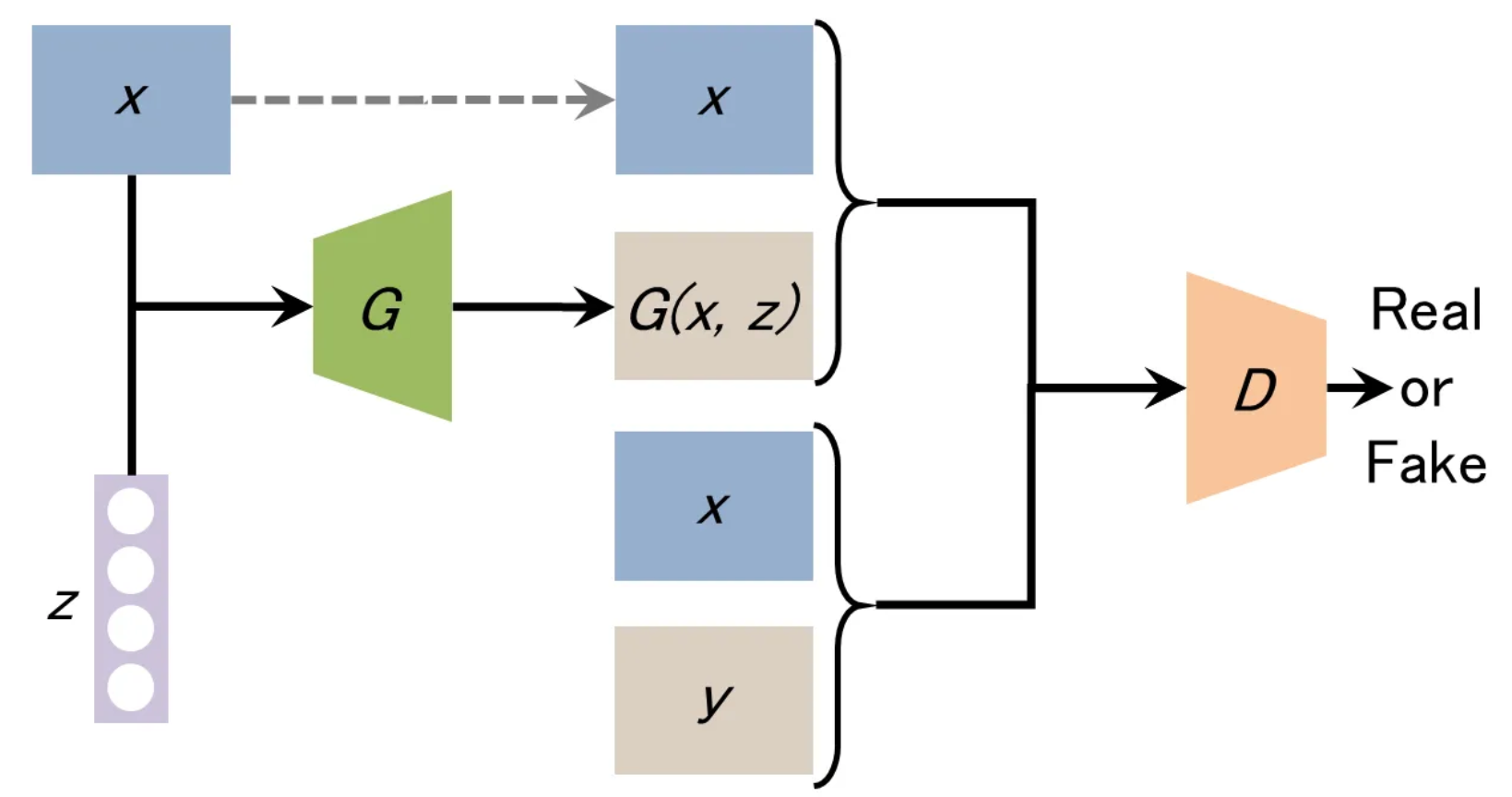
1. **Pix2Pix（Crystal-Method Co.,Ltd.より）**

**条件画像と画像のペアから画像同士の関係を学習することで、1枚の画像から学習したペアの画像関係を補完した画像を生成することができる。またpix2pix は汎用性の高い画像生成アルゴリズムであり，問題ごとにネットワークを設計する必要がないというのも特徴である。**

* **Pix2Pixの特徴**

**航空写真から地図の作成，白黒画像からカラー画像の生成，昼の風景から夜の風景の生成など多種多様な画像生成に用いられる。主に地理情報分野における活用が期待されており、地図から衛星画像へ、衛星画像から地図への変換においても実施事例として紹介されている。土地利用や地すべりなどの予測に用いられた事例もあり、今後土地の有効活用や災害などの予測などで活躍することが期待される。**

* **ネットワーク構造**

****

1. **A3C（Asynchronous Advantage Actor-Critic）（Qiitaより）**

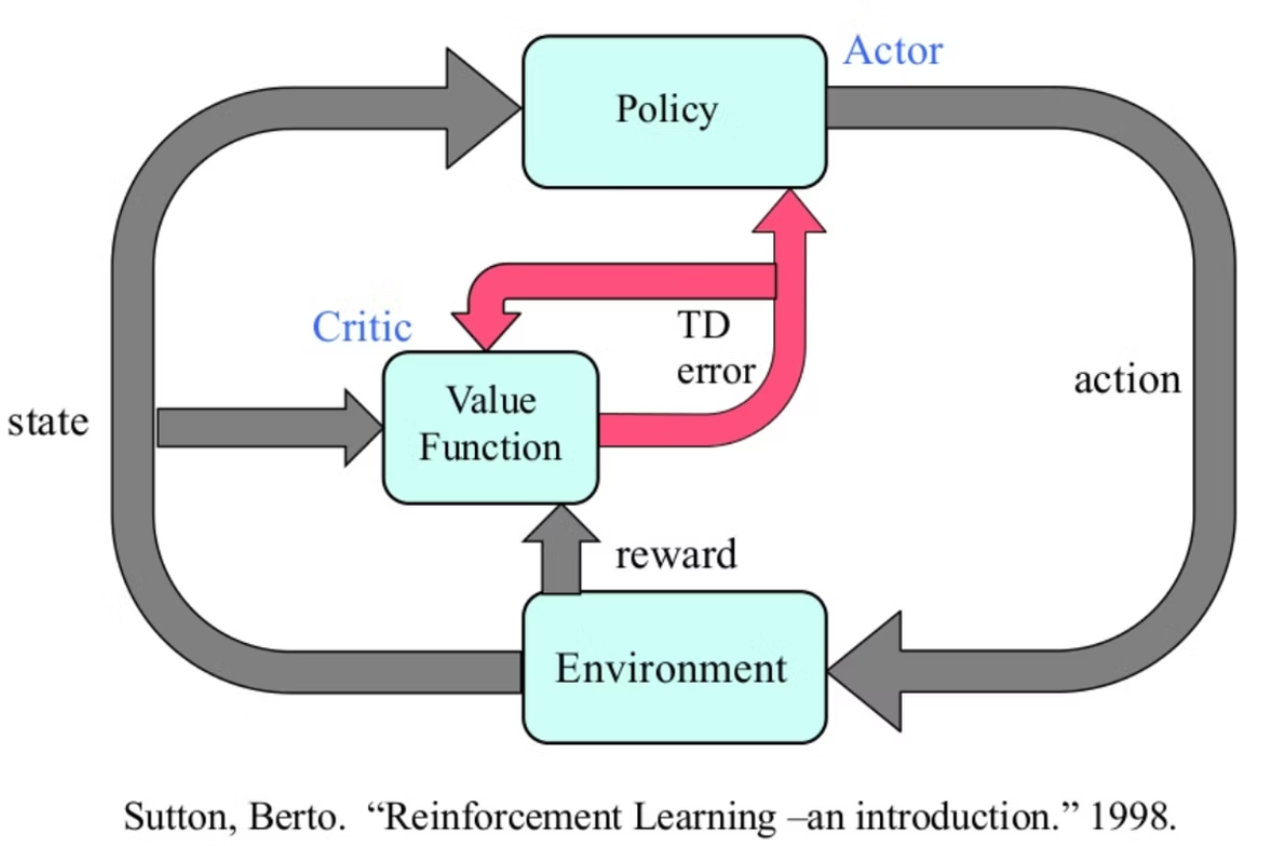
**A3CはDQNの発展と、並列化の流れが合体したような手法である。強化学習の世界では、DQNの次の世代の手法として注目を浴びた革新的なアルゴリズムになる。強化学習の学習法の一つで、DeepMindのVolodymyr Mnih(ムニ)のチームが提案した。複数のエージェントが同一の環境で非同期に学習する、といった特徴を持つ。**

* **A3Cのアルゴリズム**
* **Advantage**

**通常のQ学習、DQNではQ関数の更新を、が、 に近づくように、Q関数を学習してきた。ここで、は時刻で得た報酬、は状態での行動をした結果の状態である。つまり、は時間割引率である。で再帰的にQ関数を学んでいく。CartPoleの場合は、もしくは倒れたときが終端（終了）となるので、終端の場合、次の状態がないため、とQ関数が再帰的でなく確定する。このように、終端の状態からQ関数がどんどん確からしくなっていく。Advantageは、このQ関数の更新を「1ステップ先でなく、2ステップ以上先まで動かして、更新しよう」という考え方である。**

* **Actor-Critic**

**Q学習、DQNなどは、状態において、行動を行った場合に、「その先得られるであろう報酬の合計を時間割引した総報酬R」（割引総報酬）を出力するのQ関数を使用して強化学習を実施していた。このようなQ関数を用いた強化学習は、Value-Basedと呼ばれる。一方で、「Actor-Critic」はPolicy-Basedと呼ばれる別の枠組みと、value-Basedの組み合わせとなる。Policy-BasedはQ関数を求めず、状態から直接行動を決める手法である。アルファ碁ゼロも、一つのネットワークにPolicy-Basedと、value-Basedを組み合わせたActor-Criticになっている。Actor-Criticの場合には、ネットワークが行動を出力するActor部分と、状態の割引総報酬を出力するCritic部分に分かれている。**

****

* **Asynchronous**

**Asynchronousは、非同期的でマルチエージェントな分散学習になる。これはマルチスレッドで複数の学習環境を用意し、各環境のAgentがそれぞれ勝手に経験を積み重ねる。各スレッドは各自自分の、Advantage-Actor-Criticのネットワークを持っている。さらに、全スレッドで共有したAdvantage-Actor-Criticのネットワーク（Parameter Server）が存在する。各Agentは勝手に動いて溜めた経験から、よりたくさん報酬が得られるように、Advantage-Actor-Criticで、ネットワークの重みを更新させる方向（gradient）を求める。gradientを求めるタイミングは、一定ステップTmaxが経過するか、終端に達したときである。ここでそのAgentは自分のネットワークの重みを更新するのではなく、全スレッドで共有したParameterServerにネットワークの重みを更新させる方向（gradient）を渡す。そして、共有ネットワーク（Prameter Server）でネットワークの重みをgradientの方向に更新する。gradientを渡したスレッドは、更新されたPrameterServerの重みをコピーし、シミュレーションを継続する。これらを各スレッドが非同期的に勝手なタイミングで実施する。以下の流れとなる。**

1. **スレッドはParameter Serverからネットワークの重みをコピーする。**
2. **スレッドのAgentは自分のネットワークにを入力して、を得る。**
3. **を実行し、とを得る。**
4. **をスレッドのメモリに格納する。**
5. **②〜④を繰り返す。(各スレッドでTmaxステップ経過もしくは、終端に達するまで)**
6. **経験が十分に溜まったら、自分スレッドのメモリの内容を利用して、ネットワークの重みを更新させる方向gradを求める。**
7. **gradをParameter Serverに渡す。**
8. **Parameter Serverはgradの方向にParameter Serverのネットワークを更新する。**
9. **①へ戻る**
10. **Metric-leaning(距離学習)**

**距離学習ではデータ間のmetric、すなわち「データ間の距離」を学習する。データ間の距離を適切に測ることができれば、距離が近いデータ同士をまとめてクラスタリング1ができたり、他のデータ要素から距離が遠いデータを異常と判定することで異常検知したりと様々な応用が可能となる。ディープラーニング技術を利用した距離学習の手法が数多く提案されている。このような手法は、特に深層距離学習(deep metric learning)と呼ばれている。**

* **Siamese network（シャムネットワーク)（OPTim TECH BL.OGより）**

**Contrastive lossは、2006年に次元削減(Dimensionality Reduction)の目的で提案され、一緒に使われる代表的なネットワーク構造はSiamese networkである。2枚の画像ペアを入力して、共有重み(shared weights)のネットワークで特徴量を抽出し、その距離(D)によってネットワークを学習させる。intra-classのペア(positive)を入力したときは距離を最小化するように学習し、inter-classのペア(negative)を入力したときは、距離を最大化するようネットワークを学習させる。損失関数は、下記のとおりである。**

**とは、入力画像のペアである。はペアのラベルであり、positiveペアの場合は0でラベルされ、negativeペアの場合は1でラベルされる。マージンは、正の定数であるハイパーパラメータである。上記の式によると、positiveペアを入力する場合は、になるよう学習させ、negativeペアを入力する場合はになるよう学習させる。ただし、クラス内またはクラス間の距離はクラスごとに異なり、大きく変動する可能性があるため、すべてのnegativeペアに一定のマージンを適用することは不適切だと思われる。**

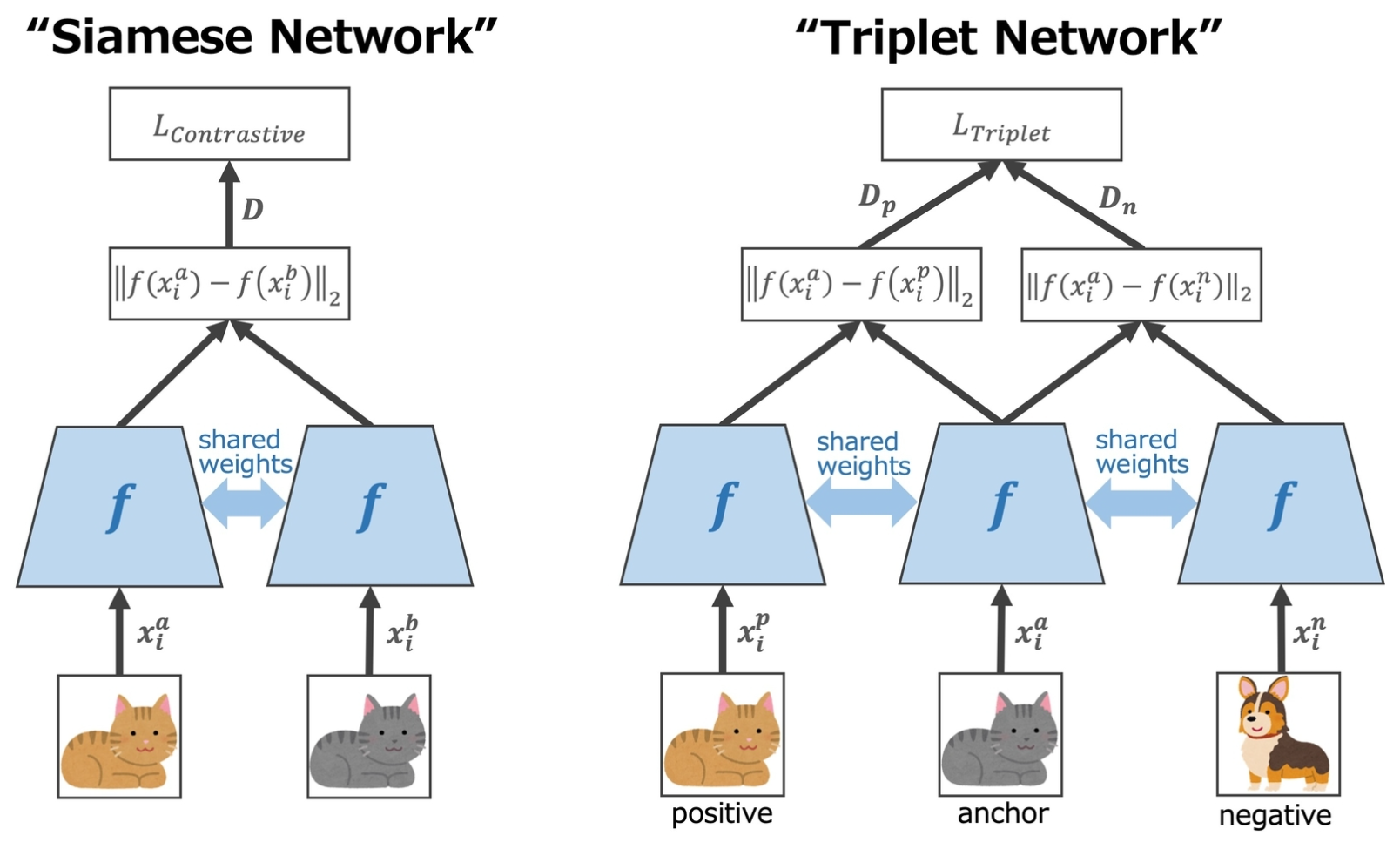
****

図 対象的アプローチのネットワーク構造

* **Triplet network（トリプレットネットワーク）**

**Triplet lossでは、相対距離にマージンが適用されたため、Contrastive lossにおけるマージンの課題に対処することができる。Contrastive lossのように絶対距離にマージンをかけるのではなく、3枚組の入力画像でpositiveペアとnegativeペアの相対距離を算出し、それにマージンをかける。つまり、Contrastive lossではnegativeペアをマージンの距離よりも離れるように学習させるが、Triplet lossではnegativeサンプルとの距離をpositiveサンプルとの距離よりもマージンの距離以上遠くなるよう学習させることである。同時で、intra-classのサンプルを引き寄せてinter-classのサンプルを離れようとすることができる。現在でも広く使われている距離学習手法で、先行研究では、Triplet lossの損失関数としていくつかの微妙な違いのある式が使用されていたが、その中によく引用されたのはWang et al.(2014)5である。その損失関数は下記の式で表される。**

**、、はサンプル組の基準となるanchor画像、positive画像とnegative画像である。この論文ではHinge lossと呼ばれたが、最近の研究では同様な式をTriplet lossと呼ばれることが多い。FaceNetの論文は、上記の式をもとにして最小化される損失関数をTriplet lossとして提案した。下記の式では、最小損失をゼロに設定している。**

**なお、上記の2つの論文では、サンプル間の距離を比較するために正規化された特徴量を用いている。Triplet lossと一緒に使われている代表的なネットワーク構造はTriplet networkである。**

1. **MAML(メタ学習：Model-Agnostic Meta-Learning)**

**ICML2017に採択された論文で，回帰問題や分類問題，そして強化学習といった様々なタスクに対応できる一般的なモデルとなっている。少ないパラメータ更新で未知のタスクを学習できるModel-Agnosticなアルゴリズムを持っているのが特徴である。**

* **Inner loop（AI-SCHOLARより）**

**あるモデルを (は学習パラメータ)と仮定し，学習用の新タスクをとする。尚、は学習率である。するとInner Loopはタスクごとに下式の通り，タスクごとの勾配に対してパラメータ更新を行い、を導出する。これは1回の更新以外にも任意回数の更新でも可能である。**

* **Outer loop（AI-SCHOLARより）**

**その後Outer loopでは，Inner Loopにて任意回数の更新を行ったモデルに対して，各タスクの勾配をまとめてOuter loopの更新を行う。βはOuter loopの学習率とすると，下式のように定義される。**

**損失に関してはInner loopにてタスクごとに更新した損失をとっているが，更新自体はではなく、Inner loop更新前のを使う。これがMAMLの基本的な考え方となっている。**

1. **グラフ畳み込み（GCN：Graph Convolutional Network）**

**GCNsとは、グラフデータに対して畳み込み （Convolution）を行う手法である。グラフデータには、ノード自体の性質を表すノード特徴量と、エッジによるノード同士の隣接関係を示すデータが含まれているとする。（Albertより）**

* **畳み込み**

**グラフデータにおける畳み込みは、グラフ内の1ノード（以下ターゲットノード）が持っている特徴量に、隣接関係にあるノードの特徴量に重みをかけたものを加えていく作業である。この作業によって、ターゲットノードの特徴量にはターゲットノード自体の特徴量だけでなく、どのような隣接関係を持ち、周囲ノードにはどのような特徴量を持つものがあるのかといった情報が含まれることになる。（Albertより）**

* **GCN**
* **Spatial GCN**

**Spatialは空間的という意味で、グラフにおける局所性と解像度を定義して、グラフに拡張することで、ノイズが目立たなくなり、形がはっきりするといった特徴がある。**

* **Spectral GCN**

**スペクトル（Spectrum）とは、光や信号などの波を成分に分解し、成分ごとの大小(強度)を見やすく配列したものである。手順は下記となる。（logmi Techより）**

1. **フーリエ変換は波形を周波数の成分に分解する。グラフフーリエ変換は、グラフ上で定義されたノードとの急峻さを定義し、急峻さごとに成分分解する。**
2. **グラフラプラシアンの固有値計算で、「急峻さ」を成分で分解することが可能となる。**
3. **2つの関数のたたみ込み合成後の関数のフーリエ変換は、合成前の個々の関数のフーリエ変換の積に等しいという定理から要素積で畳み込み畳み込みを行う。**
4. **要素積に対して逆フーリエ変換を行うことで特徴が明確になる。**
5. **Grad-CAM, LIME, SHAP**

* **CAM（Class Activation Map）（Qiitaより）**

**CAMは，2016のCVPRにて発表されたCNNの判断根拠可視化の手法である。この手法はResNetのようにGlobal Average　Poolingを使用するモデルに適応できる手法として提案された。具体的な計算方法は下記のようになる。まず下記の図のようにGlobal Average Pooling -> 全結合層 -> 予測のようなモデルを考える。クラスのsoftmax前のスコアをとし，はクラスと接続する重みであり，と接続する重みであり，は全結合層の添え字である(厳密には番目の特徴マップのGlobal Average Pooling後のスカラー値とクラスのユニットをつなぐ重み)。は全結合層の添え字である(厳密には番目の特徴マップのGlobal Average Pooling後のスカラー値とクラスのユニットをつなぐ重み)。は，番目の特徴マップの座標に対応する出力である。そうすると，は実は，下記のように表すことができる。はGlobal Average Poolingの計算である。**

**つまり，が番目の特徴マップの重みづけることによって番目のスコアが出力される。このことを利用して，の部分を用いれば特徴マップのに関しての重要さを表すヒートマップができる。**

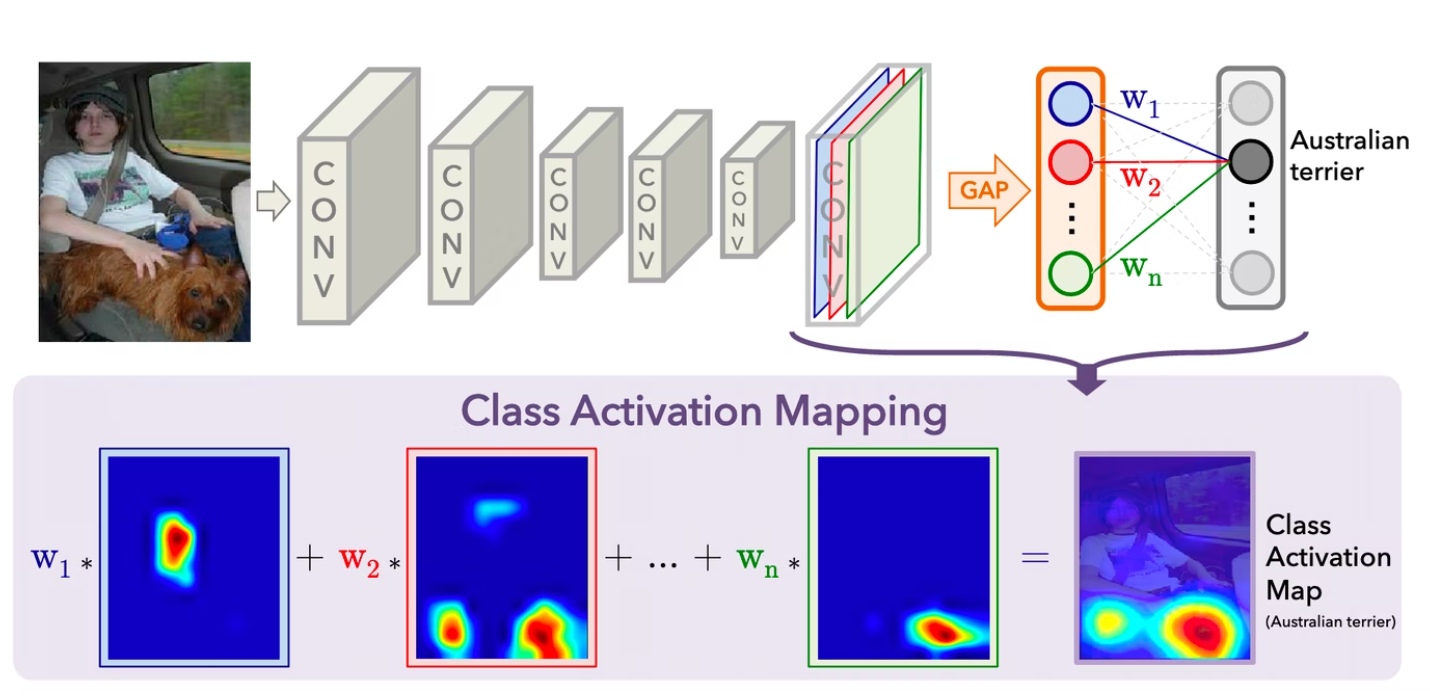
****

図 CAMのモデルのイメージ

* **Grad-CAM**

**CAMは特徴マップと最終層の重みの積で計算されることでヒートマップを作成していたが、Grad-CAMはこの各特徴マップの重み付けの部分のところとGlobal Average Poolingの部分を逆伝搬時の勾配で代用できるというものでした。Grad-CAMは有名な手法なので、Global Average Pooling -> 全結合層の部分を特徴マップへの勾配の各特徴マップ内の平均で代用できる、という手法である。これにより，様々なネットワークでCAMと同様のことができるようになり、幅広くCVの分野では使われている。基本的には下記のように変更があっただけである。**

****

図 Grand-CAMのモデルのイメージ

* **LIME（Local Interpretable Model-agnostic Explanations）**

**LIMEはあるデータに対するAIモデルの予測について、寄与した特徴量とそのスコアを可視化する技術だ。様々なAIモデルに対して適用が可能であり、構造データだけでなく画像やテキストデータも説明できる。（日経XTECHより）**

**LIMEでは、一つの予測結果があったときに、その結果に対してのみ局所的に近似させた単純な分類器を作って、その単純な分類器から予測に効いた特徴量を選ぶということをやっている。大局的に近似させようとするのではなく、局所的に近似させることで近似の誤差を許容範囲内に収められている。ではこの単純な分類器をどのように作るのか、ということだが、それを説明しているのが次の式である。説明したい事象に対してそれを説明する分類器は次の目的関数によって求まる。（Qiitaより）**

* **局所近似する分類器を求める目的関数**
* **：** **解釈可能なモデルの集合**
* **：のうちの一つのモデル（例えば、線形モデルなど）**
* **：説明したい分類器**
* **：データとの距離**
* **：データの周辺での結果がどれだけ違っているか（は損失関数）**
* **：説明用分類器の複雑さ**

**つまり事象の周辺で、できるだけ単純な分類器を作ろうとしている。**

* **：の周辺のデータの集合**
* **：非ゼロの要素を一部だけ含むサンプリングにより生成された2値のスパースな点。で定義される。**

**損失関数はこのようになっていて、説明したい分類器とLIMEで近似する分類器との距離を元データ との類似度で重み付けしたものになっている。**

**はこのような感じで、との距離関数にマイナスをかけたものをカーネル指数で割ったものである。は元データとサンプリングしたデータとの距離が近ければ近いほど小さくなる類似度関数となっているので、そのぶん損失関数も小さくなりやすくなる、というわけで損失関数の重み付けに使われている。**

**最後にですが、これはLIMEで作り出した単純な分類器が利用する特徴量の数が高々個であることを示す関数である。そもそもLIMEで利用する特徴量個をどのように選ぶのか、という話はここまでの式の中には全く出ていなかったが、ではこの個をどのようにして選択するのかというと、筆者らがK-Lassoと呼んでいるLassoで正則化パスを使用して利用する特徴を個選択し、最小二乗法を介して重みを学習する方法を用いることで、選択していく。最終的には目的関数をRidge回帰で解いて、欲しい分類器を獲得し、その分類器の偏回帰係数から特徴量の寄与度を表示する。**

* **SHAP（SHapley Additive exPlanations）**

**SHAP「シャプ」はモデルの予測結果に対する各変数（特徴量）の寄与を求めるための手法である。**

* **XAI (Explainable AI)**

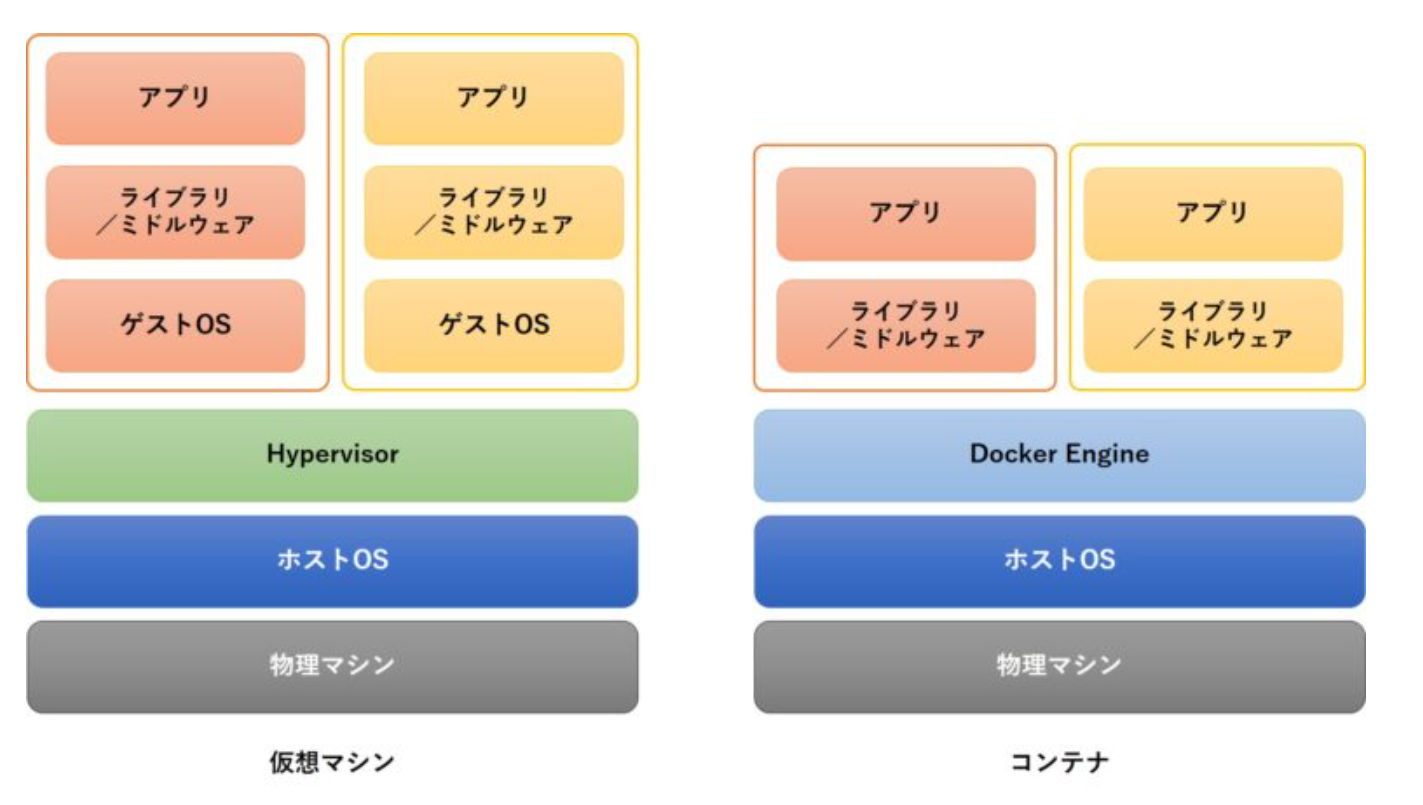
**説明可能な AIのことである。機械学習はブラックボックスであったが、解釈性・説明性が強く求められるようになり、それを実現するためのテクニックも世の中に多く出てきている。（例：SHAP、LIME、DeepLIFT、AIX360 (IBM)、Activation Atlases）**

* **Shapley値**

**協力ゲーム理論において複数プレイヤーの協力によって得られた利得を各プレイヤーに公正に分配するための手段の一つ。**

1. **Docker**

**Dockerは、インフラ関係やDevOps界隈で注目されている技術の一つで、Docker社が開発している、コンテナ型の仮想環境を作成、配布、実行するためのプラットフォームである。Dockerは、Linuxのコンテナ技術を使ったもので、よく仮想マシンと比較される。VirtualBoxなどの仮想マシンでは、ホストマシン上でハイパーバイザを利用しゲストOSを動かし、その上でミドルウェアなどを動かす。それに対し、コンテナはホストマシンのカーネルを利用し、プロセスやユーザーなどを隔離することで、あたかも別のマシンが動いているかのように動かすことができる。そのため、軽量で高速に起動、停止などが可能である。（さくらのナレッジ）**

****

**また、Dockerはミドルウェアのインストールや各種環境設定をコード化して管理する。これはInfrastracture as Code(IaC)と呼ばれていて、ソフトウェア開発で実施されてきた構成管理や自動化などを、サーバーの各種設定やミドルウェアのインストールなどにも適用したものである。これらにより、以下のような利点が生まれる。**

* **コード化されたファイルを共有することで、どこでも誰でも同じ環境が作れる。**
* **作成した環境を配布しやすい。**
* **スクラップ＆ビルドが容易にできる。**

**例えば、開発環境(Windows上)では動いていたけどLinuxで動かなかった、といったケースも、開発工程からDockerを活用していくことで防ぎやすくなる。そして、開発工程の中で使っていた環境をそのまま本番環境に持っていくことも可能なため、環境差分が少なく、環境による問題を減らすことができる。作成したDockerイメージを他の人にも渡して使ってもらうことで、各自の環境のバージョンずれ防止や、開発環境準備の短縮化にもつながる。開発環境の準備で、説明不足や手順ミスで、丸一日かかったとなってしまうのは時間の無駄となる。開発環境からDockerを使うことで、新規メンバ参画時のコスト削減や、環境依存問題を減らせることが可能となる。また、クラスタ構成を構築する場合も、Dockerイメージがあれば、それを起動する名前（コンテナ名）などを変えるだけで、複数の環境（コンテナ）を起動できるので、一から手順に沿って環境を作る作業もなくなり、クラスタ構成を構築するのも容易になる。これは、コンテナを管理するオーケストレーションツールを使うことで、より運用が楽になる。最近では、オーケストレーションツールとしてKubernetes（https://kubernetes.io/）が注目を浴びていて、Dockerによりネイティブでサポートされることになり、話題にもなった。さらに、コード化したもの(Dockerfile)を構成管理し、それをCIツール（JenkinsやCircleCIなど）で、ビルド（Dockerイメージの作成）、デプロイ（コンテナの起動）、環境のテスト（Serverspecなど）、画面試験（Seleniumなど）を毎日実行すれば、環境面の確認も含めて安心して開発を進められる。このサイクルを回せると、いざ環境を構築するとなった際に「アプリを動かすのに何のパッケージが必要なの？」と慌てることも減らせて、開発・リリースサイクルも安定してくる。**