# 付録
# Mittendorf et al.(2024) で使用された アルゴリズム

流体地球物理学教育研究分野

本間 友子

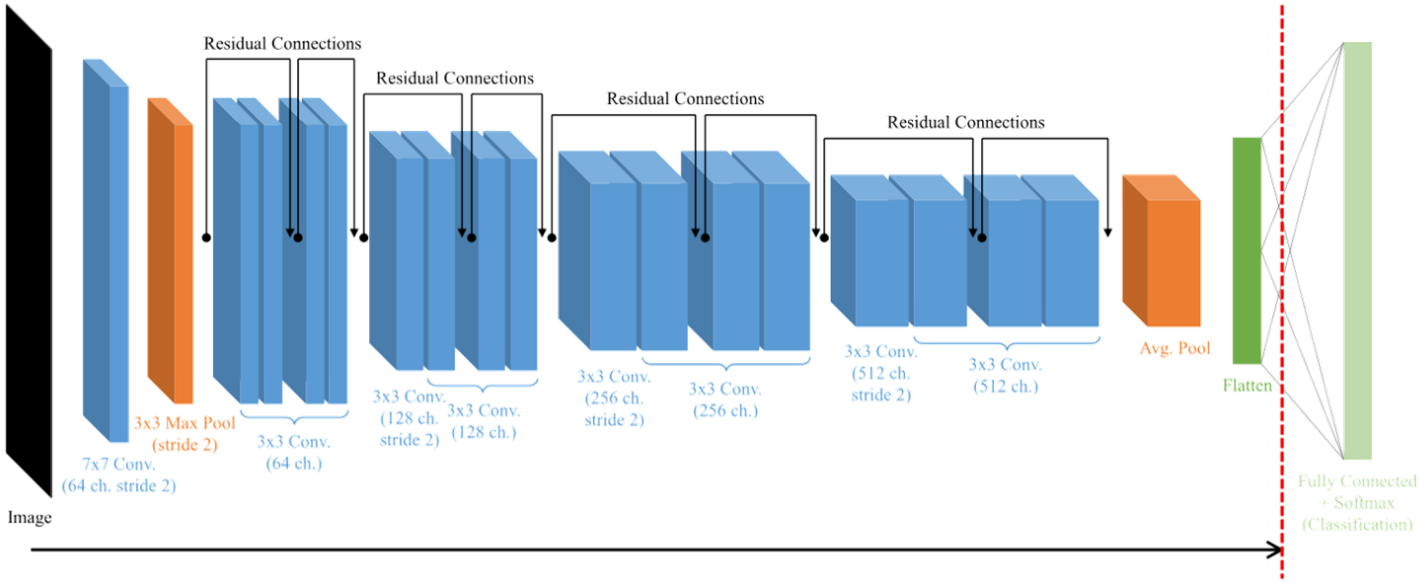
# はじめに

- 以下のアルゴリズム, モデルを紹介する
  - ResNet-18
  - SimCLR
  - HAC
- いずれも Mittendorf et al.(2024) のクラスタリングモデルの開発に採択されたものである
- 内容は, あくまで本間が Web ページから学習中の内容であり, 不正確な点も含む (かもしれない)

# 付録A ResNet18 の紹介

# 付録A-1 ResNet18 の概要

- 基礎モデルとして使用した画像認識用畳み込みニューラルネットワーク (CNN)『ResNet』のうち, 畳み込み層が 18 層で構成されているもの
- Residual Module というモジュールが搭載されていることが特徴

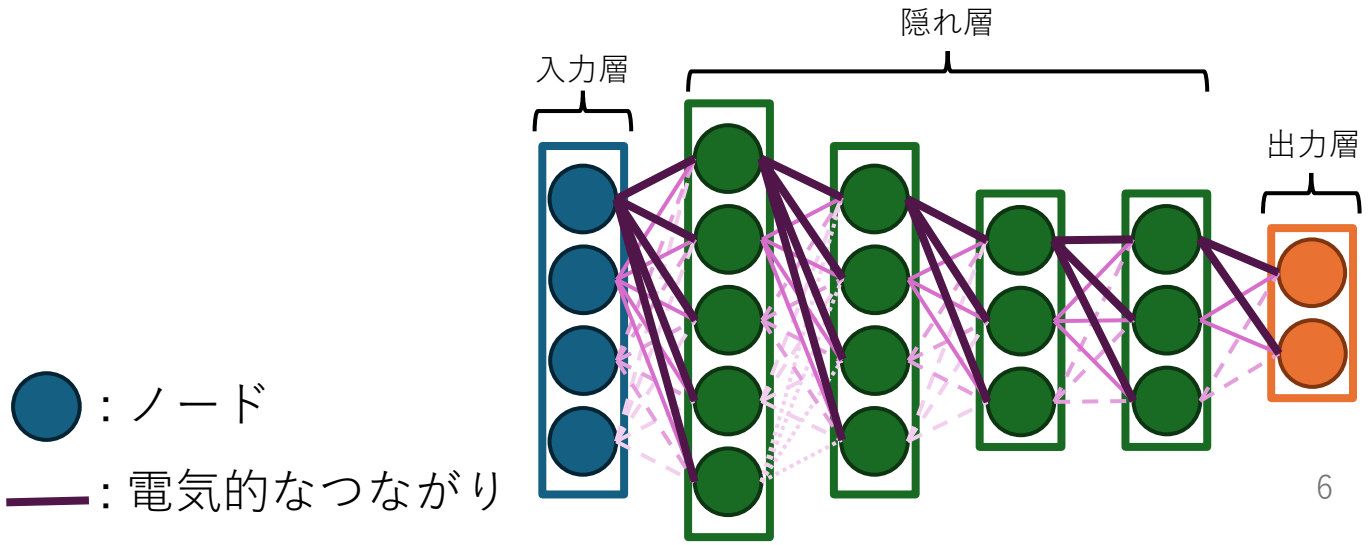


# 付録A-1-1 CNN とは

- Convectional Neural Network (畳み込みニューラルネットワーク)
- 特に画像認識に有効なニューラルネットワーク (NN)

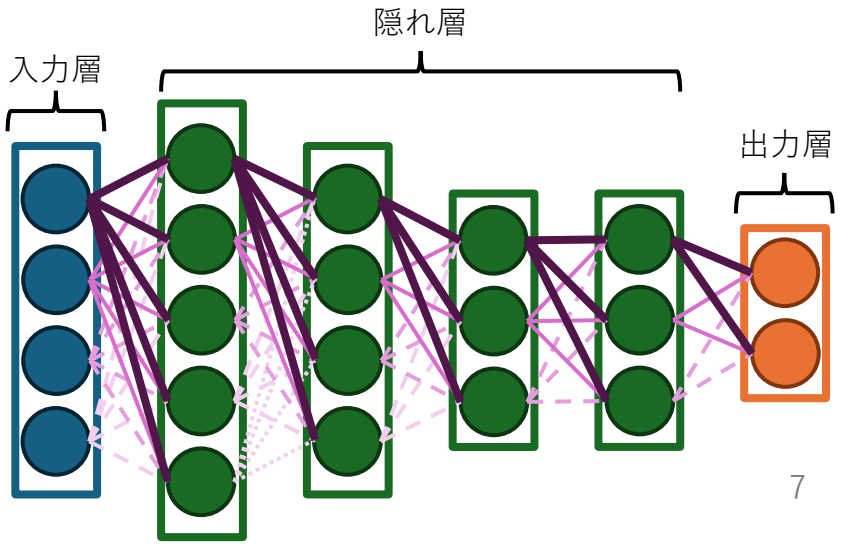
# 付録A-1-2 NN とは

- ニューラルネットワークとは
- 機械学習の一つで, 複数の層 (入力層, 隠れ層 × n, 出力層)によって構成される
- 各層に『ノード』を多数配置し, それらを電気的に相互接続させている



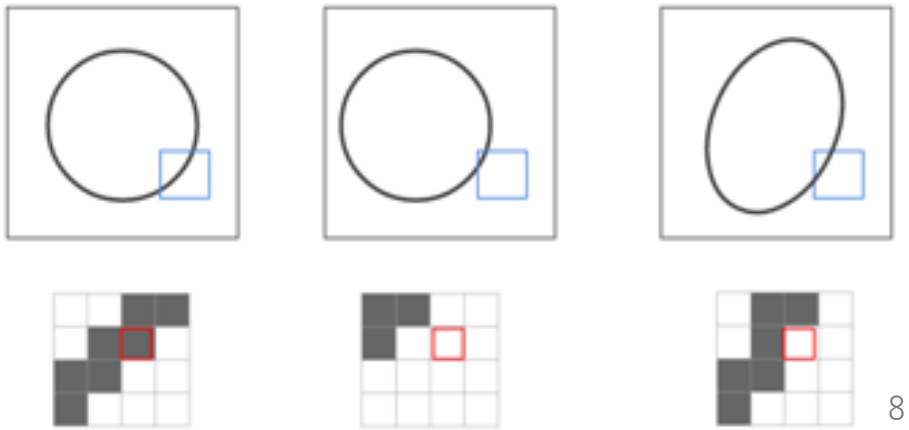
# 付録A-1-2 NN とは

- まず解析対象のデータ (各ピクセルの色情報, ベクトル情報) が入力層の各ノードに割り当てられる
- 次に, 入力層のノードのデータを, 隠れ層のすべてのノードに渡す
- 受け取ったデータに対しそれぞれ処理を施す (重みづけ (係数をかける), 活性化関数 (非線形変形))
- 隠れ層のノードのデータを, 次の層のすべてのノードに渡す
- これを繰り返し, 最終的に出力層へデータが渡される



# 付録A-1-1 CNN とは

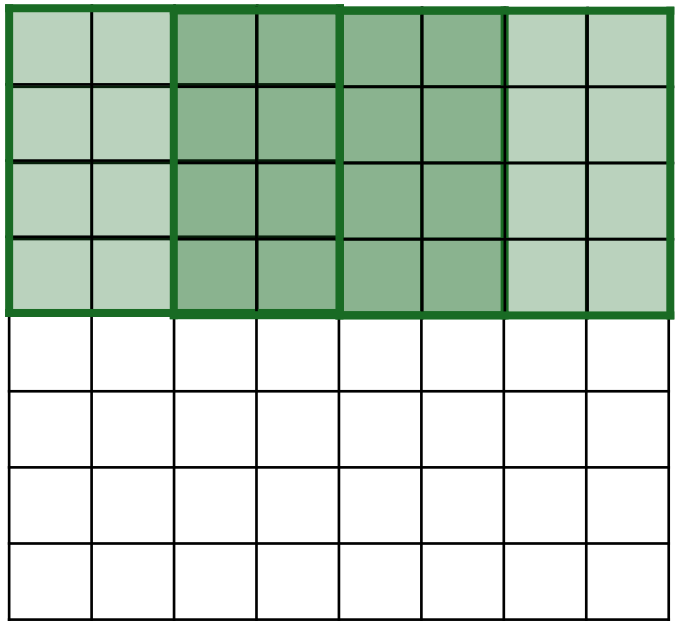
- Convectional Neural Network (畳み込みニューラルネットワーク)
- 特に画像認識に有効なニューラルネットワーク (NN)
- ノードに渡されるデータがピクセル毎の情報の場合, わずかにずれがある場合でも違う画像として認識されてしまう
- そこで, 各ノードには複数のピクセルから得られる『領域ごとの特徴量』を渡す
  - 領域ごとの特徴量を得る過程を『畳み込み』と呼ぶ

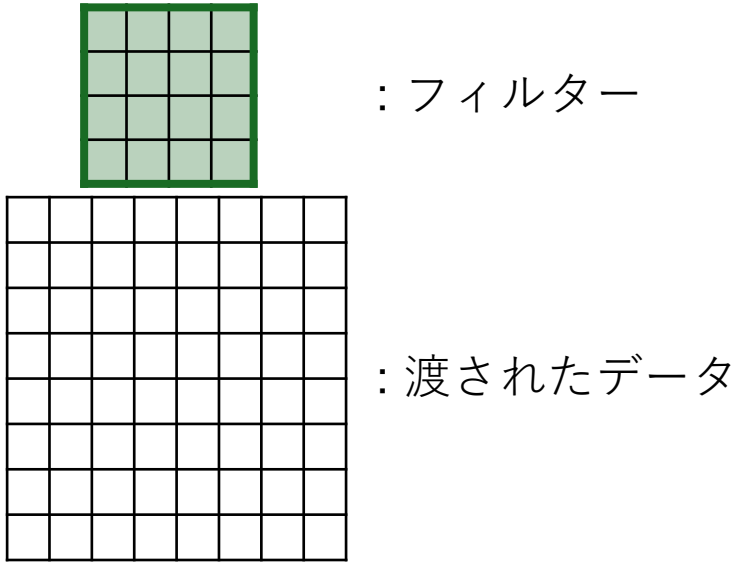


Convolutional Neural Networkとは何なのか #Python – Qiita より (2026/07/26 閲覧)

# 付録A-1-3 畳み込みの手順

- 渡されたデータに対し『フィルタ』を掛け, そこを領域とする
- フィルタをスライドさせ, 全体をカバーするまで繰り返す





# 付録A-1-3 畳み込みの手順

- フィルターが持つ係数と, 領域内の各データをかけ合わせる
- その後, すべての積を足し合わせることで『その領域の特徴量』が計算できる
  - この手順が畳み込みである
  - 例：4×4 のデータから 1×1 の特徴量を得られる

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | |
| 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | | |
| 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | | | |
| 0.4 | 0.5 | 0.6 | 0.7 | | | | |
| 0.5 | 0.6 | 0.7 | | | | | |
| 0.6 | 0.7 | | | | | | |
| 0.7 | | | | | | | |
| | | | | | | | |

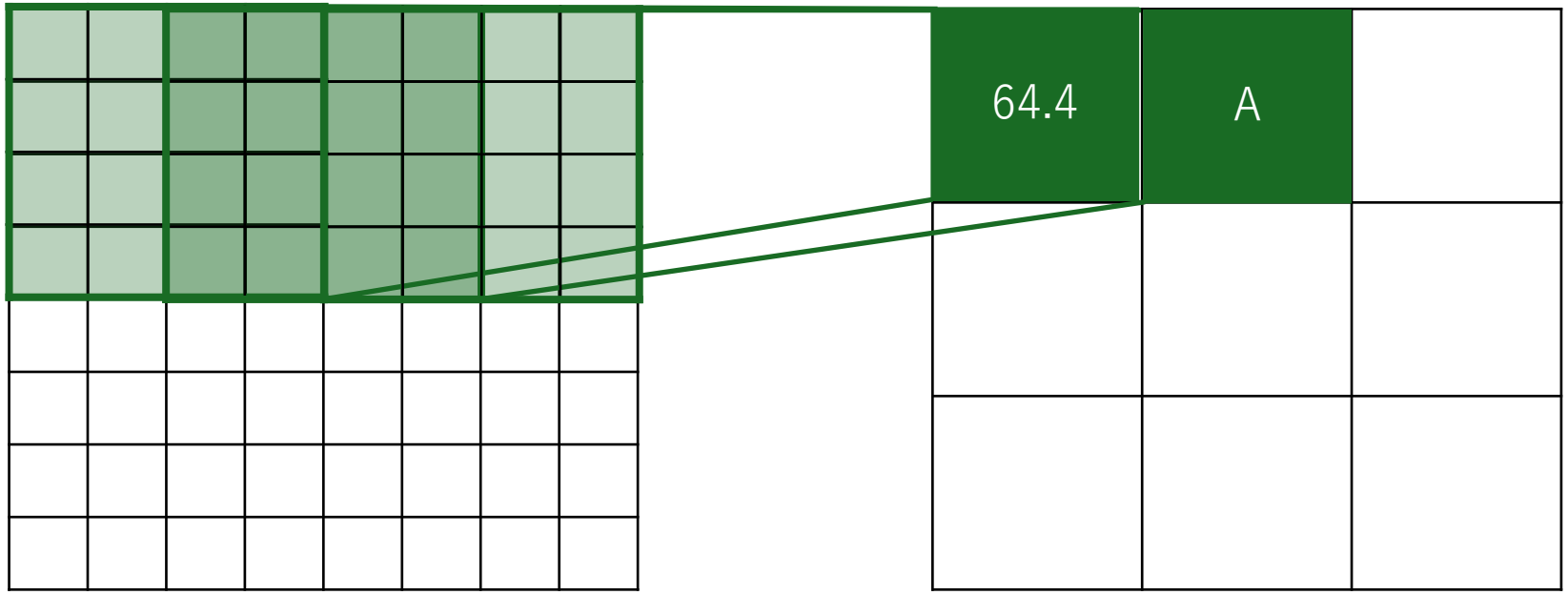
×

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 |

$= 0.1 + 0.4 + 0.9 + \ldots + 11.2 =$ 64.4

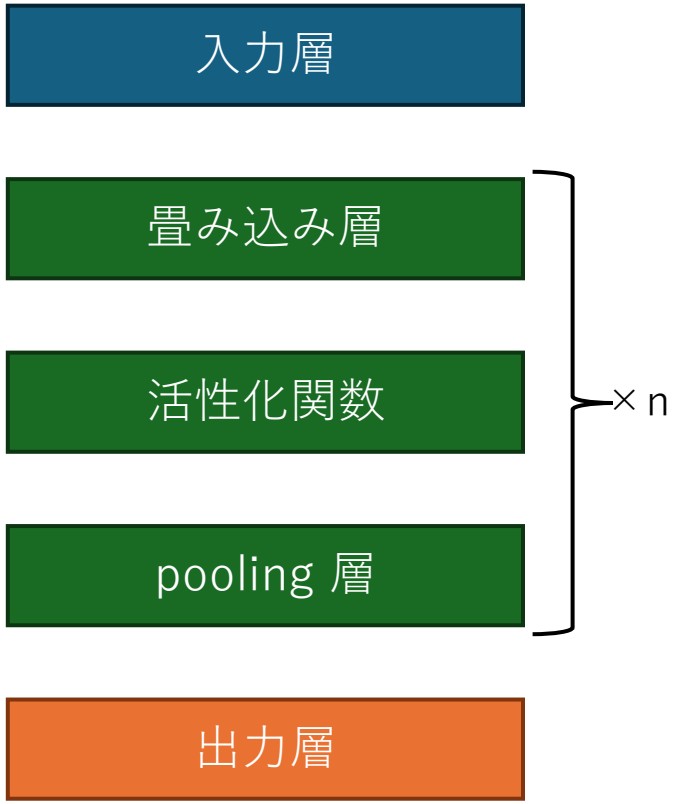
# 付録A-1-3 畳み込みの手順

- 生成した領域すべてに対し, 畳み込みを行う
- その結果, (見方を変えると) 元の画像よりデータ量が少ない新しい画像を生成される
- 次の層に渡す



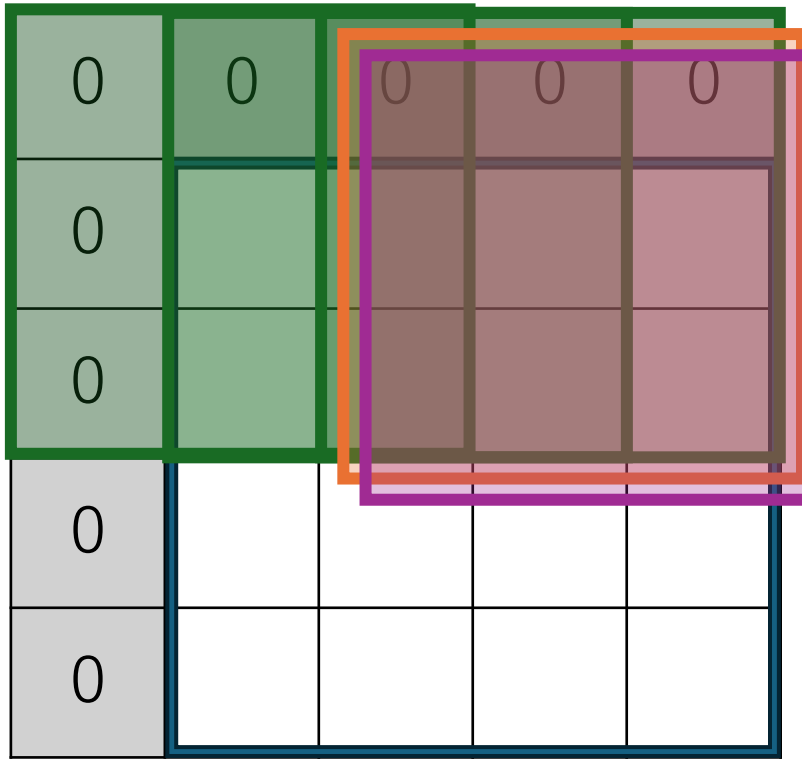
# 付録A-1-1 CNN とは

- CNN における層の構造を模式的にあらわすと右図の通りになる
  - 畳み込み層：前の層で出力される画像を畳み込む層
  - 活性化関数：前の層から渡された情報を，非線形に変形する層
  - pooling 層：扱いやすくするため，画像を圧縮（サイズを小さく）する層
  - 活性化関数は畳み込み層の一部として見られることもある
- 機械学習においては，出力結果から畳み込み層で使用されるフィルターの係数を更新していく



# 付録A-1-4 使用するパラメータ

- フィルタの数：一つの隠れ層に存在するフィルタの種類
- フィルタの大きさ：畳み込む領域の大きさ
- フィルタの移動幅：何データポイントずつフィルタを移動させるかの幅
- パディング：画像の端の領域を，どれぐらい0で埋めるか
  - 本来の画像の端の部分について，畳み込まれる回数が少なくなってしうまうことを避けるために行う



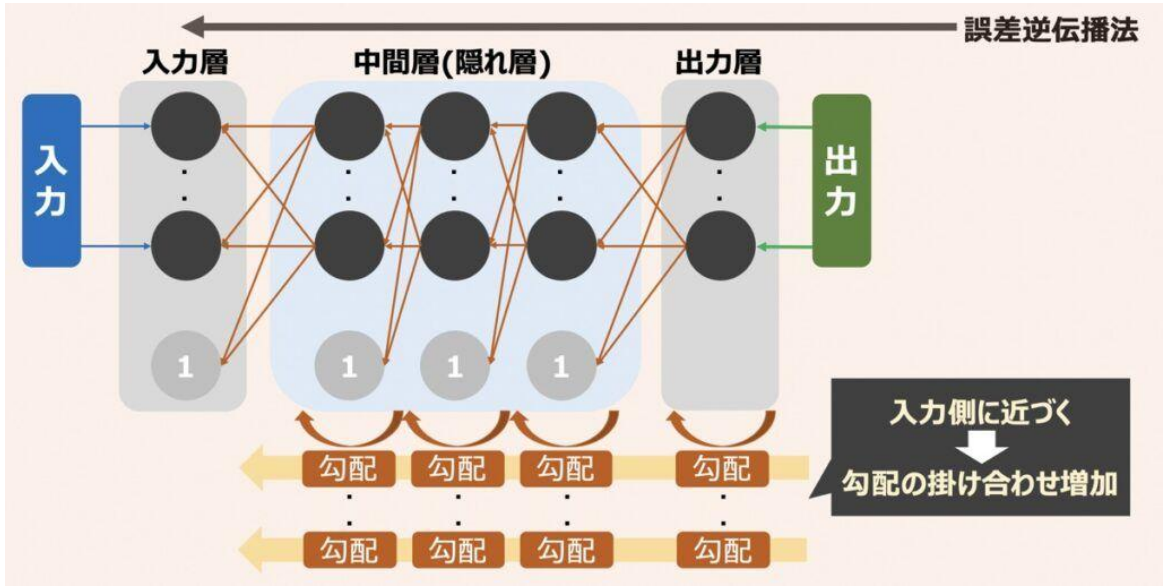
# 付録A-2 ResNet18 の開発背景

- ResNet 登場以前 (~2015 年)
- CNN において, 畳み込み層を増やすほど, 画像内のより潜在的な特徴を捉えることができるようになると考えられていた
- しかし, ある程度深くなると (タスクに対する) 精度が向上しない段階が生じてしまう(勾配消失問題, 劣化問題)
  - 劣化問題については説明しない
- そのため, 一定以上の層数から重ねることができなかった

# 付録A-2-1 勾配消失問題

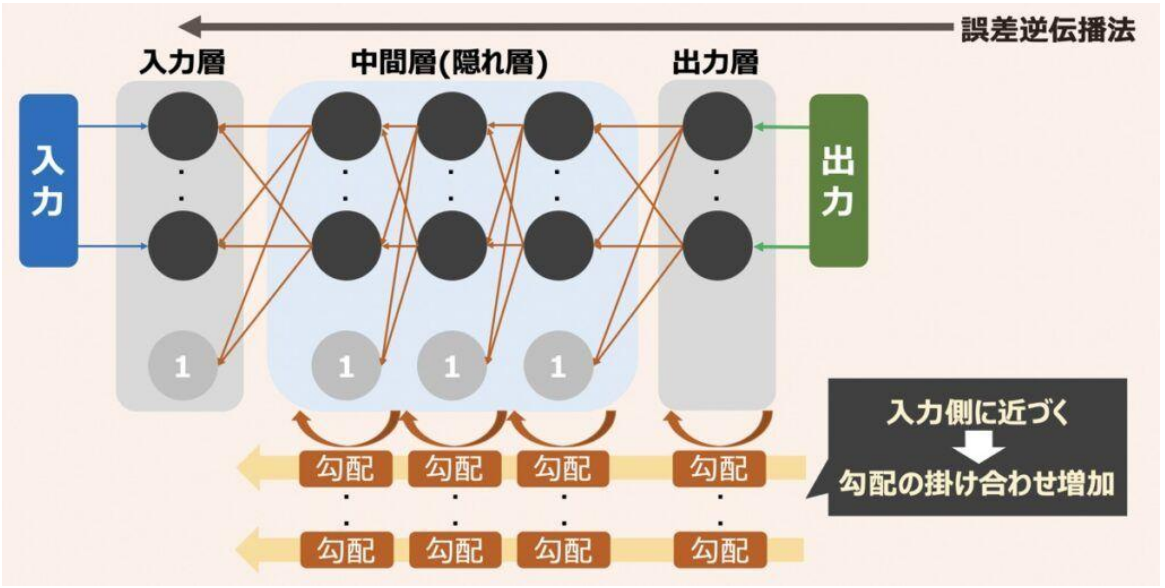
- 機械学習時に係数を更新するとき，『損失関数』とその『勾配』を考える
  - 損失関数：学習モデルの目的に応じた関数のこと
  - 目的の状態と出力間の差を関数として表し，これを最小にしていく

  - 勾配：損失関数の入力値に対する勾配
  - 係数の更新する方向性を与える



# 付録A-2-1 勾配消失問題

- 機械学習時に係数を更新するとき，『損失関数』とその『勾配』を考える
  - 勾配は, 各層における出力, 入力を変数とした連鎖率の形で表される
    - 詳しくは A-3-2
  - それぞれの層で伝搬されている勾配を掛け合わせながら, 入力層までの勾配が計算される
  - ある程度学習が進み, それぞれの勾配が小さくなると, 入力層付近の勾配がゼロに近づき, 学習がうまくできなくなる

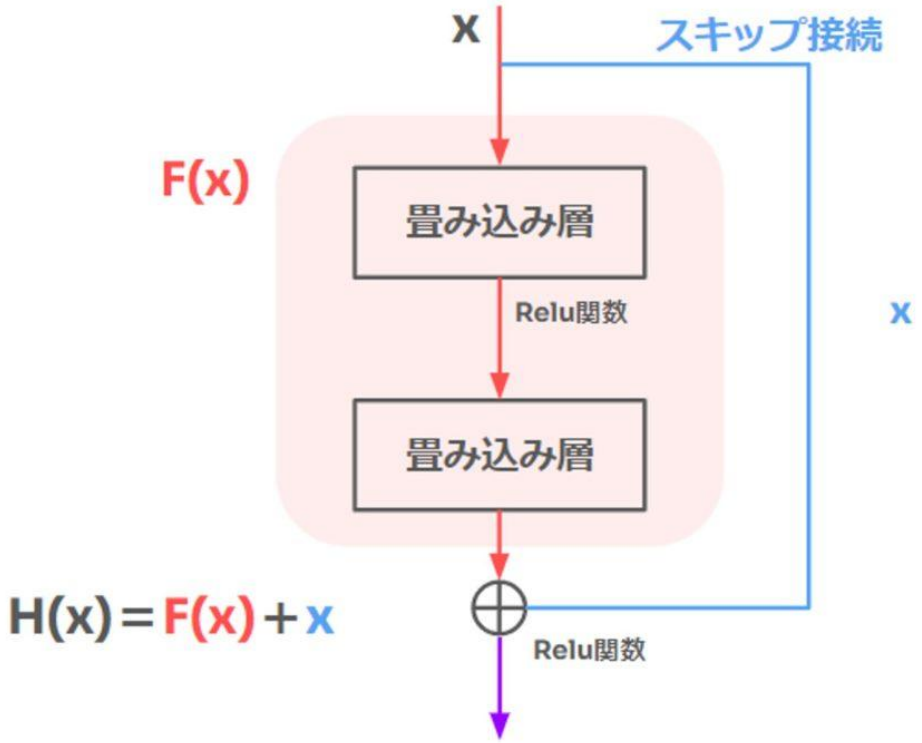


# 付録A-2 ResNet18 の開発背景

- ResNet 登場(2015 年)
- 問題を解決するため, 畳み込み層 (と活性化関数) の出力結果をそのまま学習するのではなく,
- 出力結果に, 入力する値を足し合わせた関数を学習するモデルを開発した
- (層数としてはそれ以前の 22 層モデルから 6 倍の 152 層モデルを実現した)
- 恒等写像 (モデルを通した出力が入力値と等しくなる) を学習したときに十分な精度を提供できた

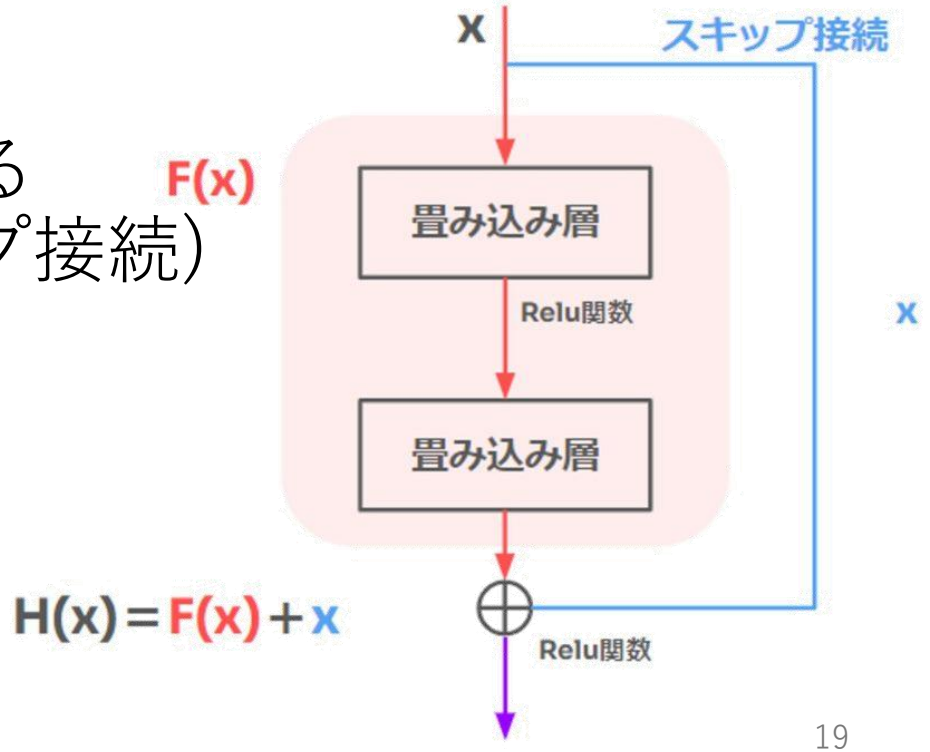
# 付録A-3 ResNet18 の構造, 仕組み

- Residual Network
  - ResNet における一番の特徴
  - 出力結果に, 入力する値を足し合わせる
  - x : 入力値
  - F(x) : ResBlock
  - H(x) : ResNet で学習する関数
  - (ReLU 関数 : 活性化関数の一つ)



# 付録A-3-1 Residual Network

- 通常のネットワークにおいての出力は F(x) である
- F(x) = x となるように学習するとき, 関数 F のパラメータを調整する
  - どうやらここに問題が生じる
- そのため出力を H(x) = F(x) + x となるように Residual Connection (スキップ接続) を追加する
- H(x) = x となるとき,
  F(x) = (H(x) - x) = 0, すなわち
  w = 0 になるように学習する
  - どうやらこの負荷が少ないらしい



# 付録A-3-2 Residual Network

- 勾配消失問題に対してなぜ有効なのかを数式で追っていく
- 『損失関数』を $L$, 入力量を $x$, モデルの出力結果を $H(x)$ とするとき, 以下の式のように書くことにする

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial H} \cdot \frac{\partial H}{\partial x}$$

  - (左辺) = 入力値に対する損失関数の勾配
  - (右辺第一項) = Residual Network の出力結果に対する損失関数の勾配 = Residual Network の出力時点まで伝搬されている勾配
  - (右辺第二項) = 入力値に対する Residual Network の出力結果の勾配 = Residual Network の出力時点から入力時点まで伝搬されている勾配

# 付録A-3-2 Residual Network

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial H} \cdot \frac{\partial H}{\partial x}$$

- $H(x)$ に $F(x) + x$ を代入すると, 右辺第二項は

$$\frac{\partial H}{\partial x} = \frac{\partial (F(x) + x)}{\partial x} = \frac{\partial F(x)}{\partial x} + 1$$

と変形できる

# 付録A-3-2 Residual Network

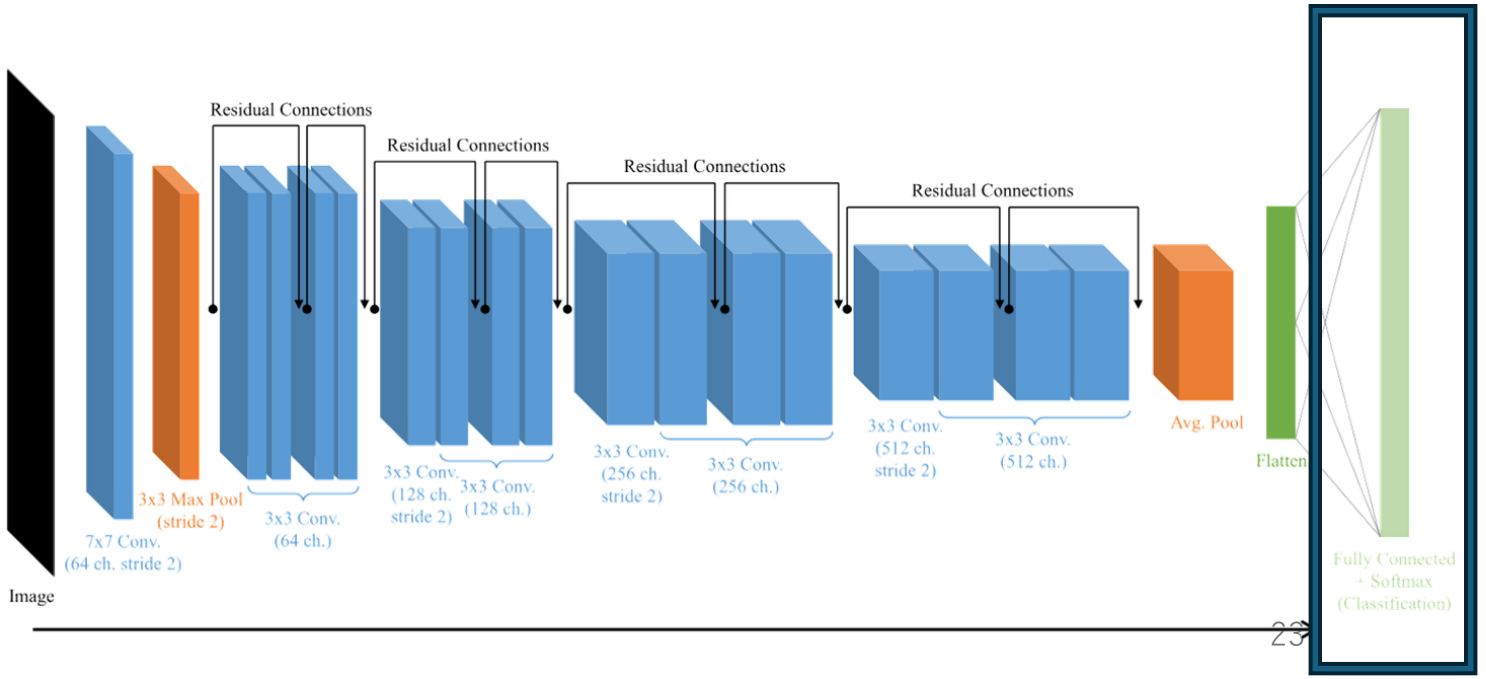
- したがって損失関数の勾配は

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial H} \cdot \left( \frac{\partial F(x)}{\partial x} + 1 \right) = \frac{\partial L}{\partial H} \cdot \frac{\partial F(x)}{\partial x} + \frac{\partial L}{\partial H}$$

  と変形できる
- 勾配消失問題において, 最右辺第一項が小さくなってしまい, 入力付近の係数の学習ができないことが問題であった
- しかし Residual Network を用いると,『 Residual Network の出力時点まで伝搬されている勾配』がそのまま足されている
  - そのため, 勾配消失問題に対し有効である (らしい)

# 付録A-3-3 事前学習

- ImageNet-1Kという, 様々な画像が含まれたデータセットを用いて (画像分別用に) 事前学習されている
- そのため, 最後の層はImageNet-1K の画像クラスに対応した確率分布レイヤーである
  - このモデルを流用するときは, 最後の層を取り除いたものを使用する



# 付録A 参考資料


- 【図解解説】ResNetとはなにか？わかりやすく解説 #AI – Qiita (2026/07/26 閲覧)
- Resnetを解説，実装する！ #Python – Qiita (2026/07/26 閲覧)
- Pytorch – ResNet の仕組みと実装について解説 | pystyle (2026/07/26 閲覧)
- ResNet18の構造をやんわりと理解する #初心者 – Qiita (2026/07/26 閲覧)
- 【実践事例】ResNet × PyTorchでロゴを含む画像を見分ける | 現場で使える製造業データ分析入門 (2026/07/26 閲覧)

# 付録A 参考資料

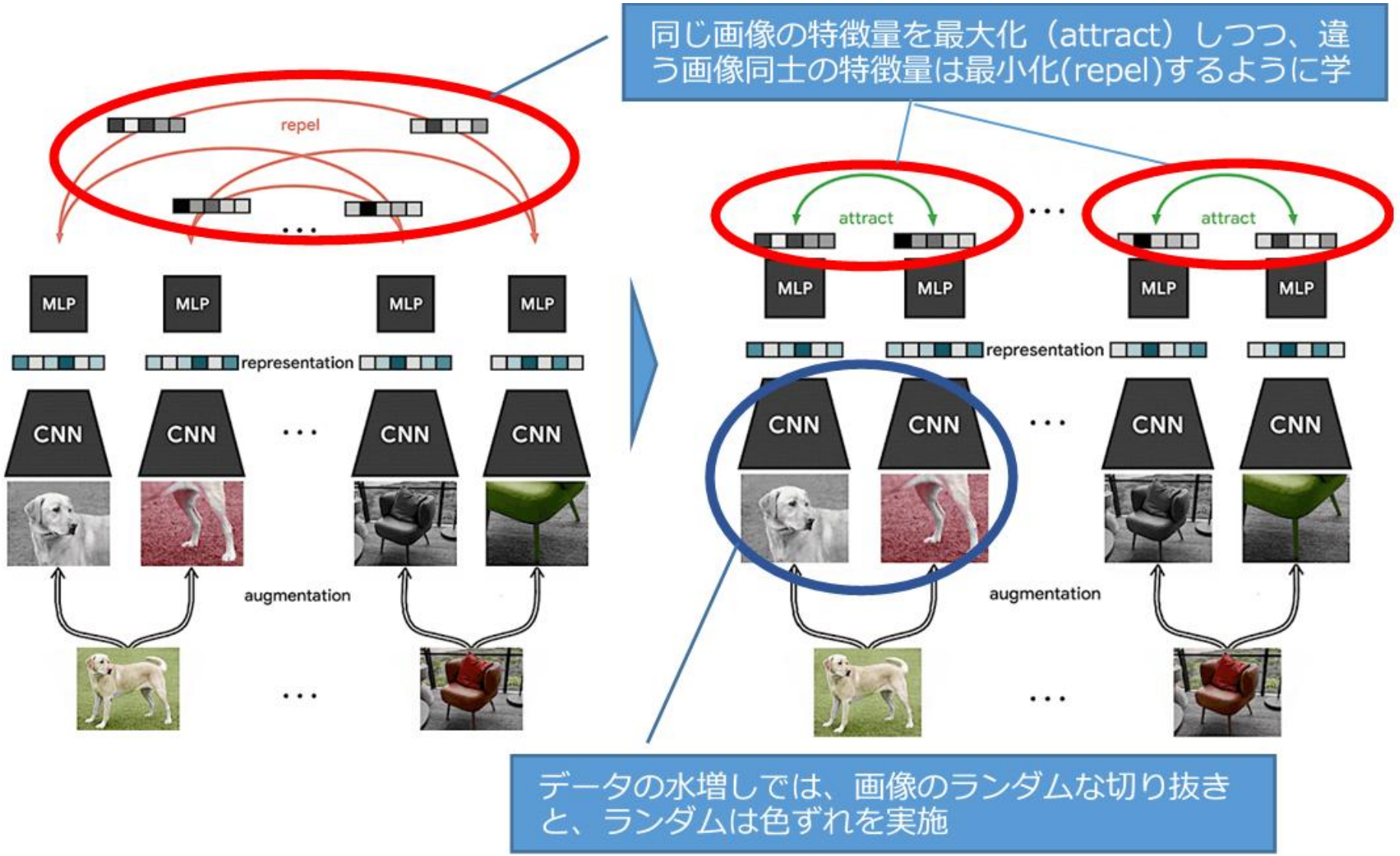

- ニューラルネットワークとは？仕組み・種類・学習方法をわかりやすく解説 | JAPAN AI ラボ (2026/07/26 閲覧)
- Convolutional Neural Networkとは何なのか #Python – Qiita (2026/07/26 閲覧)
- 具体例で覚える畳み込み計算（Conv2D、DepthwiseConv2D、SeparableConv2D、Conv2DTranspose） #Python – Qiita (2026/07/26 閲覧)
- 誤差関数（損失関数）とは？AI・機械学習で重要な役割をわかりやすく解説 - IT用語辞書 (2026/07/26 閲覧)
- 【深層学習】勾配消失問題とは？ニューラルネットワーク学習時の対処方法 | ディープラーニング入門 | DXCEL WAVE (2026/07/26 閲覧)

# 付録B SimCLR の紹介

# 付録B-1 SimCLR の概要

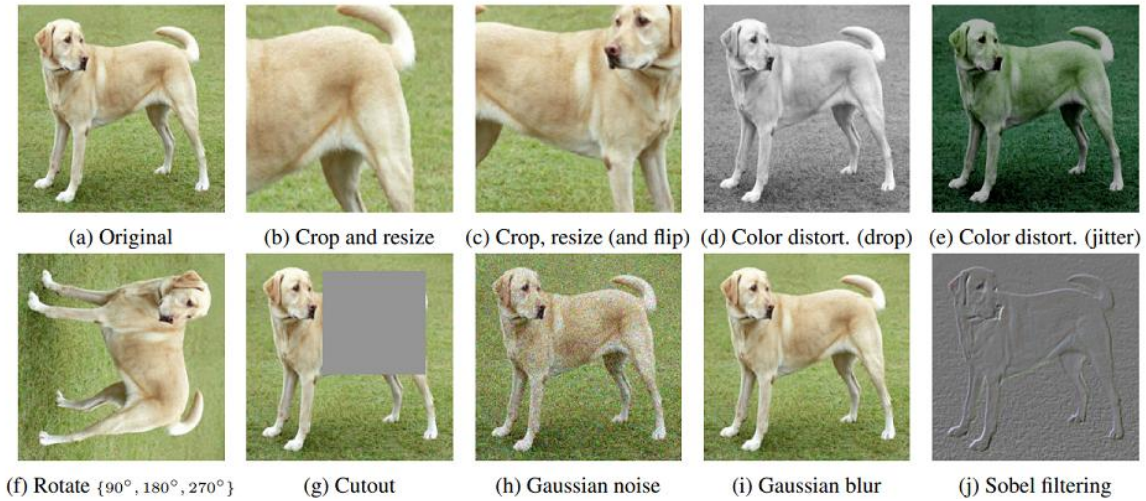
- (論文中で採用された学習手法)
- 機械学習における学習方法の一つ
- 同じ画像から得られる特徴量 (特徴ベクトル) の一致度を上げ, 違う画像から得られる特徴量の一致度を下げるように学習を進める
- 特徴ベクトルが近しいものは, ベクトル空間上で近くに配置されるようにすることが目的
- 画像のクラスタリングなどで用いられる

# 付録B-1 SimCLR の概要

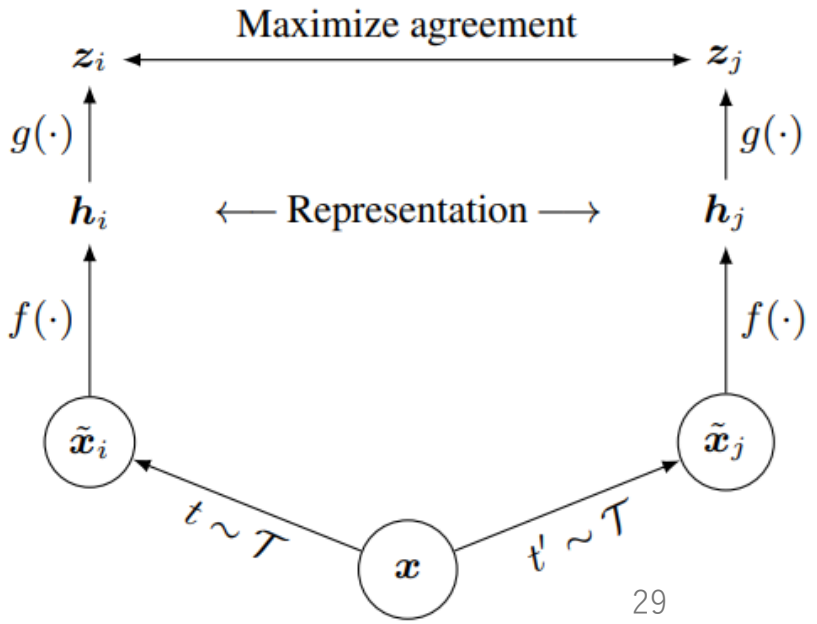


# 付録B-2 SimCLR の仕組み

1. $\mathbf{x}$ という入力から 2 つの増幅例 $\tilde{\mathbf{x}}_i$ と $\tilde{\mathbf{x}}_j$ を生成する




- 増幅は上の 10 パターンから選択する

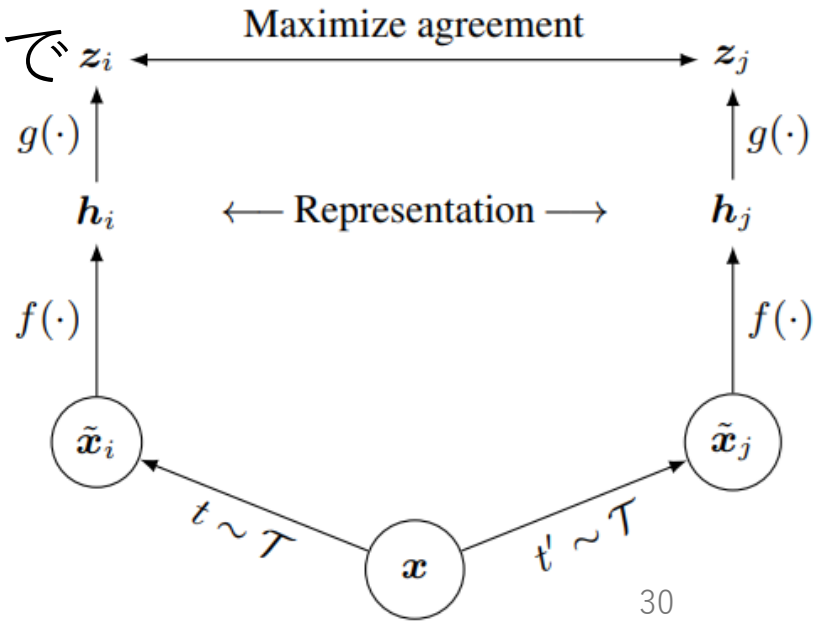


# 付録B-2 SimCLR の仕組み

2. $\tilde{\mathrm{x}}_{\mathrm{i}}$ と $\tilde{\mathrm{x}}_{\mathrm{j}}$ をそれぞれエンコード（基礎モデル $\mathrm{f}(\mathrm{x})$, 今回なら ResNet-18 に通すこと）し, 特徴ベクトル $h_i, h_j$ を得る
   - $h_i = \mathrm{f}(\tilde{\mathrm{x}}_{\mathrm{i}}), h_j = \mathrm{f}(\tilde{\mathrm{x}}_{\mathrm{j}})$

3. 投影ヘッドにより, 『コントラスト損失』で最適化するための空間へ投影する
   - 投影ヘッド：一層の隠れ層を持ったモデル
   - 出力結果を $z_i, z_j$, 投影ヘッドを $\mathrm{g}(\mathrm{h})$ とすると

     $z_i = g(h_i), z_j = g(h_j)$



# 付録B-2 SimCLR の仕組み

4. コントラスト損失を計算する
   - N 枚の元画像に対し, 2 回ずつ増幅を与えたときを考える
   - 増幅されたデータは 2N 枚になる
   - $\widetilde{\mathbf{x}_{2n-1}} = \mathrm{Aug}(\mathbf{x}_n)$, $\widetilde{\mathbf{x}_{2n}} = \mathrm{Aug}'(\mathbf{x}_n)$ とし, それぞれをエンコード, 投影ヘッドでの投影を行う
   - 増幅されたデータの集合 $\{\widetilde{\mathbf{x}_k}\}$ において, $\widetilde{\mathbf{x}_i}$ に対し同じ画像から増幅された $\widetilde{\mathbf{x}_j}$ を『正例』と呼ぶ
   - $\widetilde{\mathbf{x}_i}$ と任意のデータ $\widetilde{\mathbf{x}_k}$ の類似度を, $\mathbf{z}$ を用いて記述すると

$$sim(z_i, z_k) = \frac{{z_i}^{\mathrm{T}} z_k}{\|z_i\|\|z_k\|}$$

   とする

# 付録B-2 SimCLR の仕組み

4. コントラスト損失を計算する
   - ここにパラメータ $\tau$ を導入し

     $$s_{i,k} = \frac{sim(z_i, z_k)}{\tau}$$

     という関数を考える
     - $\tau$ は類似度分布の鋭さを表す (=標準偏差)
   - k の候補は$K_i = \{1, 2, \ldots, 2\mathrm{N}\}/\{\mathrm{i}\}$ の集合であり, $k \in K_i$ が正例である確率は

     $$p(k|i) = \frac{\exp(s_{i,k})}{\sum_{m \in K_i} \exp(s_{i,m})}$$

     とする
   - さらにクロスエントロピー損失

# 付録B-2 SimCLR の仕組み

4. コントラスト損失を計算する
   - 正例の場合の確率を求め, クロスエントロピー損失を求めると

$$
\begin{aligned}
0 < l_{i|j} &= -\log p(j|i) \\
&= -\log \frac{\exp(\frac{sim(z_i, z_j)}{\tau})}{\Sigma_{m=1}^{2N} \mathbb{N}_{[m \neq i]} \exp(\frac{sim(z_i, z_m)}{\tau})}
\end{aligned}
$$

と記せる. ただし

$$
\mathbb{N} = \begin{cases} 1 & (m \neq i), \\ 0 & (m = i) \end{cases}
$$

である

# 付録B-2 SimCLR の仕組み

5. 損失関数を最小にする
   - $l_{i,j}$ と $l_{j,i}$ の損失の合計が損失関数 L となる

$$L = \frac{1}{2N}\Sigma_{n=1}^{N}(l_{2n-1,2n} + l_{2n,2n-1})$$
$$= \frac{1}{2N}\Sigma_{i=1}^{2N}(l_{i,pos(i)})\ (pos(i) : i \text{ 番目の画像に対する正例})$$

   - これが最小になるように学習していく
     - sim(i|j) が大きくなると p~1, $l_{i,j} \sim 0$
     - L を下げるためにsim(i|j) を上げていく必要がある

# 付録B 参考資料


- SimCLR（A Simple Framework for Contrastive Learning of Visual Representations） #Python – Qiita (2026/07/26 閲覧)
- 対照学習(Contrastive Learning)と代表手法 SimCLR まとめ - kaeken(嘉永島健司)ブログ (2026/07/26 閲覧)
- SimCLR 論文解説 & PyTorch 実装（Contrastive Learning / 自己教師あり学習） (2026/07/26 閲覧)
- 【論文】自己教師あり学習「SimCLR」を理解、実装する #Python – Qiita (2026/07/26 閲覧)
- 対照学習（Contrastive Learning）の理論と損失関数の導出 | 機械学習と情報技術 (2026/07/26 閲覧)

# 付録C HAC の紹介

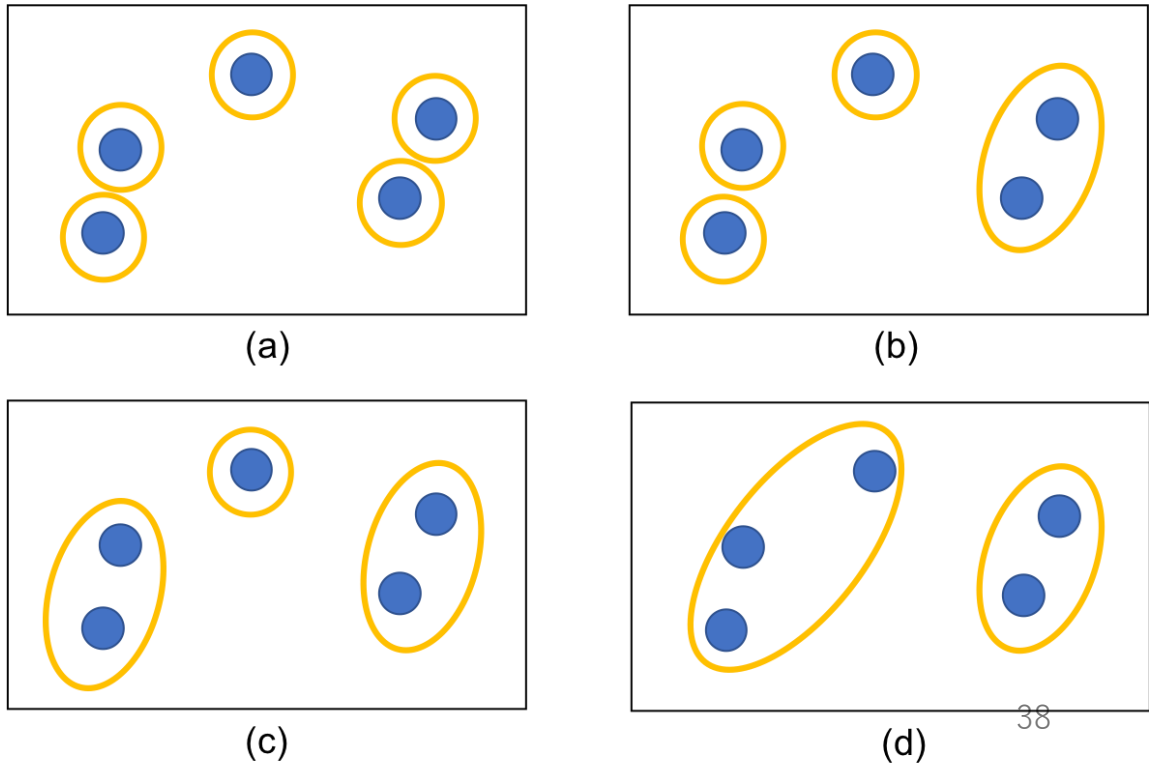
精度比較の結果採用されたクラスタリング手法

# 付録C-1 HAC の概要

- hierarchical clustering (階層的クラスタリング)
- 階層構造的にクラスタリングを行っていく
- 二つの方法に大別できる
  - 凝集型
    - これについて紹介する
    - ボトムアップ (下 (各データ) から上 (数個のクラスタ) ) で動作する
  - 分割型
    - すべてのデータを 1 つのクラスタとみて, それを区切っていく
    - トップダウン (上 (大きなクラスタ) から下 (小さなクラスタ)) で動作する

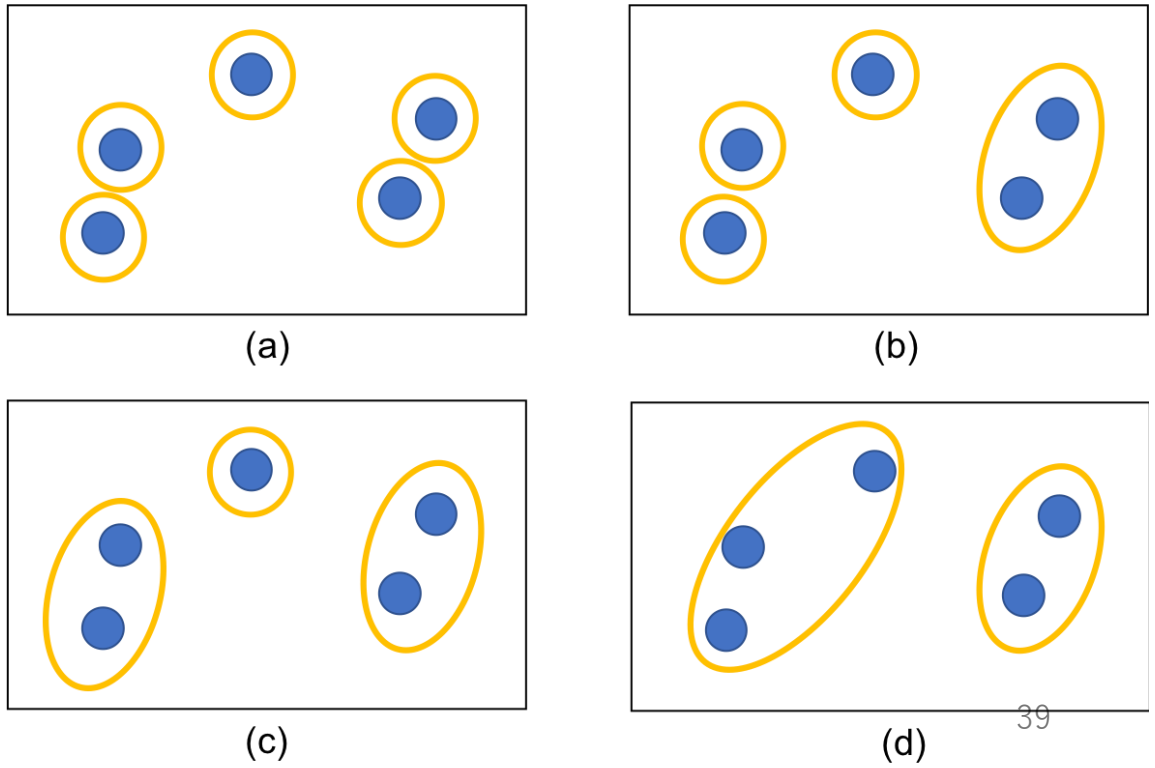
# 付録C-2 凝集型クラスタリング

- 簡単に言うと『単純に似ているもの同士をくっつけていく』クラスタリング手法
- それぞれのデータを要素数１のクラスタに格納
- 最も距離が近いもの同士を同じクラスタとして結合する
  - 距離の測り方には複数種類あるが割愛



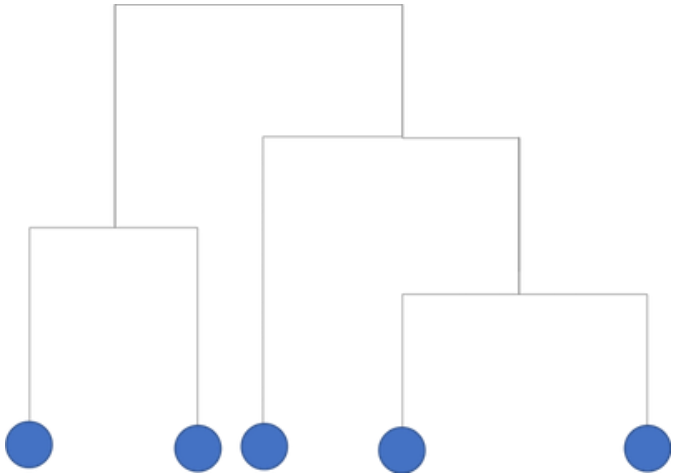
# 付録C-2 凝集型クラスタリング

- 簡単に言うと『単純に似ているもの同士をくっつけていく』クラスタリング手法
- それぞれのデータを要素数１のクラスタに格納
- 最も距離が近いもの同士を同じクラスタとして結合する
  - 距離の測り方には複数種類あるが割愛



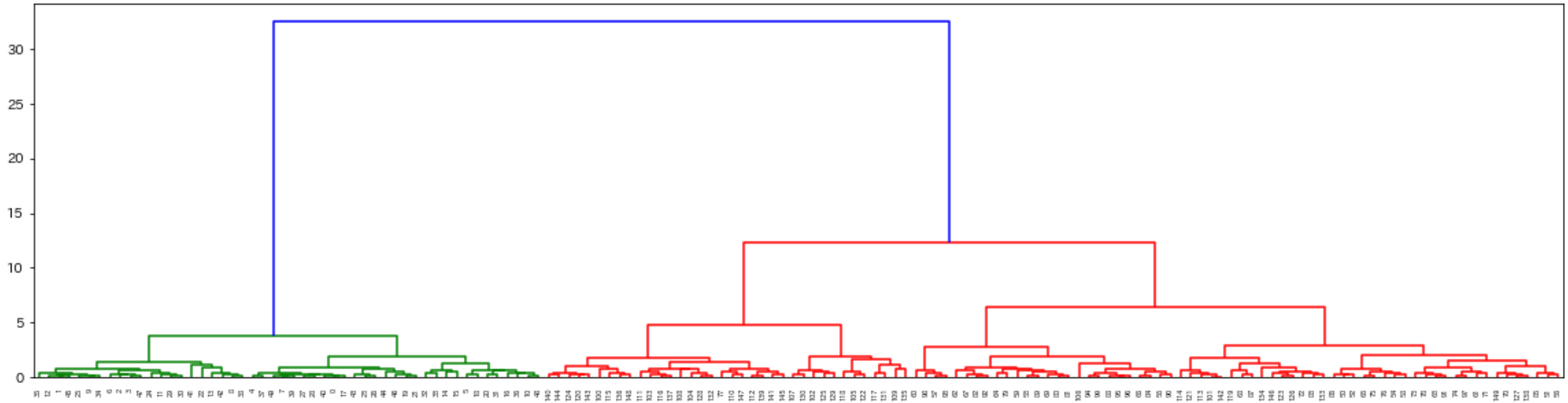
# 付録C-2 凝集型クラスタリング

- クラスタリングのプロセスは樹形図を用いて表すことができる



  - 下から順に結合し, 線が合流しているところでクラスタの結合が生じる
  - 高さが低いほど先に結合が発生している
    - なお, データの距離は類似度が高いほど近くなる
  - 実際に使用するときは, ある一定の高さに相当するステップで終了させる

# 付録C-2 凝集型クラスタリング




アヤメの品種を HAC でクラスタリングしたときの階層構造
凝集型(階層型)クラスタリングを理解する #Python – Qiita
(2026/7/26 閲覧)

# 付録C 参考資料


- 階層的クラスタリングとは？ メリットや実装方法を丁寧に解説 - DS Media by Tech Teacher (2026/07/26 閲覧)
- Scikit-learnで学ぶ凝集型クラスタリング(AgglomerativeClustering)の使い方と実践コード - コードの道しるべ (2026/07/26 閲覧)
- 凝集型(階層型)クラスタリングを理解する #Python – Qiita (2026/07/26 閲覧)
- 11. 階層的クラスタリング — 機械学習帳 (2026/07/26 閲覧)