HOKKAIDO UNIVERSITY

講義「人工知能」
第6回 ニューラルネットワーク
ニューラルネットワークの拡張

北海道大学大学院情報科学研究院
情報理工学部門 複合情報工学分野
調和系工学研究室 准教授 山下倫央
http://harmo-lab.jp
tomohisa@ist.hokudai.ac.jp
2024年4月25日(木)

ニューラルネットワークの拡張

- 画像処理における拡張
  - 画像認識タスクの分類
  - 画像認識モデル
    - 畳み込みニューラルネットワーク
  - 画像データの作成
    - アノテーションの付与
  - 画像データの増強
    - Data Augmentation (データの水増し手法)の導入

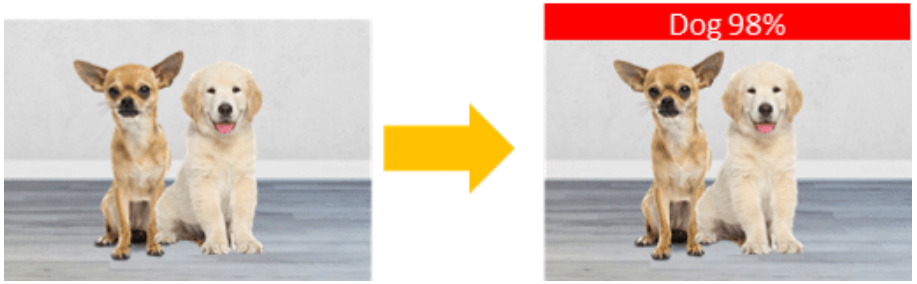
画像認識(Image Recognition)の分類

- 画像分類：Image Classification
- 画像分類・物体位置特定：Image Classification・Localization
- 物体検出：Object Detection
- セグメンテーション：Segmentation
  - セマンティック・セグメンテーション：Semantic Segmentation
  - インスタント・セグメンテーション：Instant Segmentation
  - パノプティック・セグメンテーション：Panoptic Segmentation

- 参照：画像分類・物体検出・セグメンテーションの比較
  - https://data-analysis-stats.jp/%E6%B7%B1%E5%B1%9E%E5%AD%A6%E7%BF%92/%E7%94%BB%E5%83%8F%E5%88%86%E9%A1%9E%E3%83%BB%E7%89%A9%E4%BD%93%E6%A4%9C%E5%87%BA%E3%83%BB%E3%82%BB%E3%82%B0%E3%83%A1%E3%83%B3%E3%83%86%E3%83%BC%E3%82%B7%E3%83%A7%E3%83%B3%E3%81%AE%E6%AF%94%E8%BC%83/

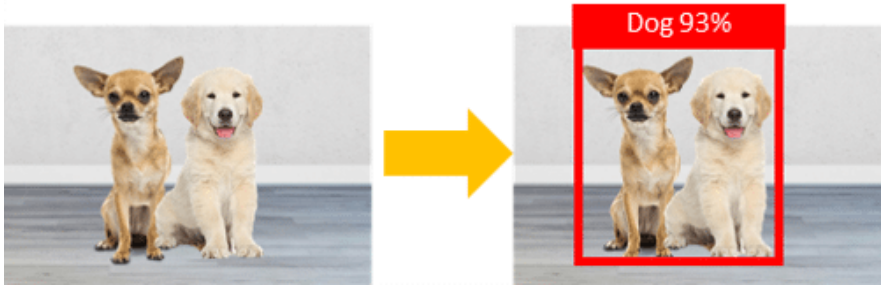
画像認識(Image Recognition)の分類

- 画像分類：Image Classification



出力：クラスと信頼度

- 画像分類・物体位置特定：Image Classification・Localization

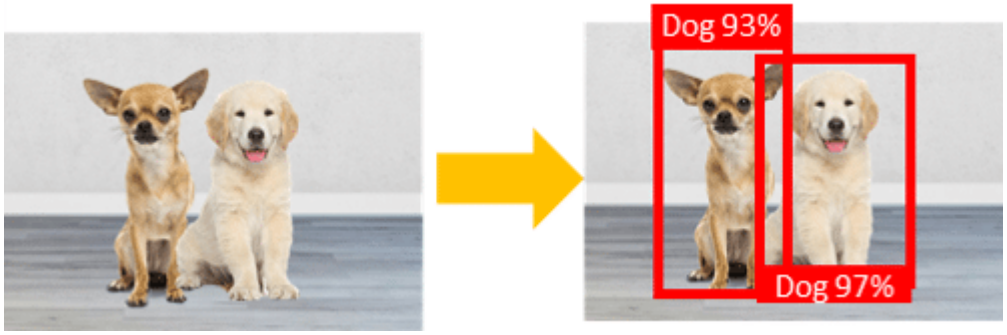


出力：クラスとその物体を含む矩形（バウンディングボックス

– 参照：画像分類・物体検出・セグメンテーションの比較
  - https://data-analysis-stats.jp/%E6%B7%B1%E5%B1%9E%E5%AD%A6%E7%BF%92/%E7%94%BB%E5%83%8F%E5%88%86%E9%A1%9E%E3%83%BB%E7%89%A9%E4%BD%93%E6%A4%9C%E5%87%BA%E3%83%BB%E3%82%BB%E3%82%B0%E3%83%A1%E3%83%B3%E3%83%86%E3%83%BC%E3%82%B7%E3%83%A7%E3%83%B3%E3%81%AE%E6%AF%94%E8%BC%83/

画像認識(Image Recognition)の分類

- 物体検出：Object Detection

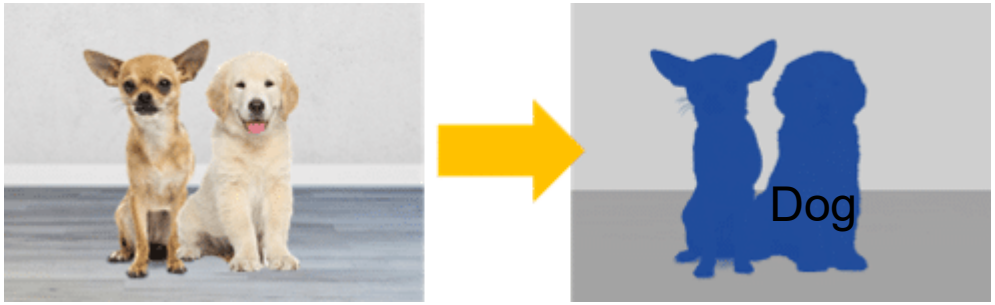


出力：クラス、クラスの信頼度、バウンディングボックス

– 参照：画像分類・物体検出・セグメンテーションの比較
  - https://data-analysis-stats.jp/%E6%B7%B1%E5%B1%9E%E5%AD%A6%E7%BF%92/%E7%94%BB%E5%83%8F%E5%88%86%E9%A1%9E%E3%83%BB%E7%89%A9%E4%BD%93%E6%A4%9C%E5%87%BA%E3%83%BB%E3%82%BB%E3%82%B0%E3%83%A1%E3%83%B3%E3%83%86%E3%83%BC%E3%82%B7%E3%83%A7%E3%83%B3%E3%81%AE%E6%AF%94%E8%BC%83/

画像認識(Image Recognition)の分類

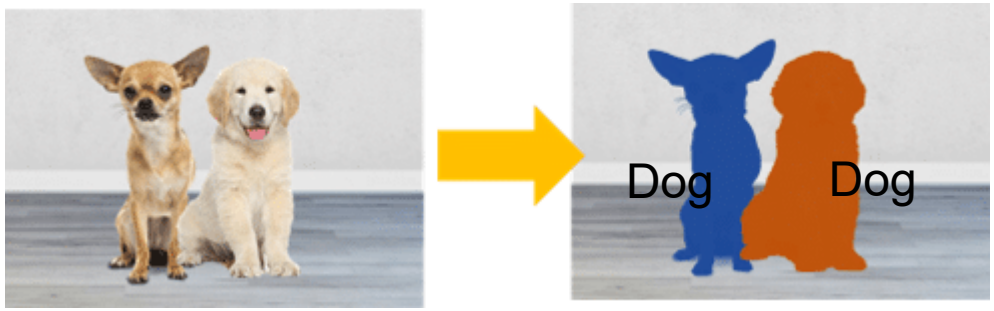
- セグメンテーション：Segmentation
  - セマンティック・セグメンテーション：Semantic Segmentation
    - 画像の各ピクセルを物体クラスに分類
    - 物体ごとの認識・カウントができない



- 参照：画像分類・物体検出・セグメンテーションの比較
  - https://data-analysis-stats.jp/%E6%B7%B1%E5%B1%9E%E5%AD%A6%E7%BF%92/%E7%94%BB%E5%83%8F%E5%88%86%E9%A1%9E%E3%83%BB%E7%89%A9%E4%BD%93%E6%A4%9C%E5%87%BA%E3%83%BB%E3%82%BB%E3%82%B0%E3%83%A1%E3%83%B3%E3%83%86%E3%83%BC%E3%82%B7%E3%83%A7%E3%83%B3%E3%81%AE%E6%AF%94%E8%BC%83/

画像認識(Image Recognition)の分類

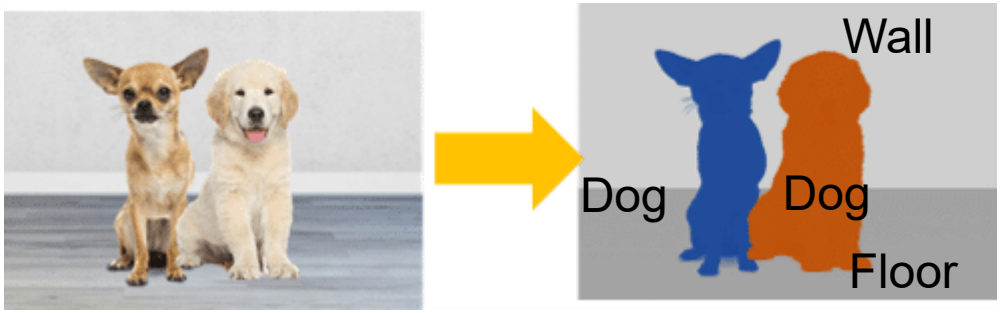
- セグメンテーション：Segmentation
  - インスタント・セグメンテーション：Instant Segmentation
    - 画像の各ピクセルを物体クラス・インスタンスに分類
    - 画像全てのピクセルに対してラベルを振るわけではない



- 参照：画像分類・物体検出・セグメンテーションの比較
  - https://data-analysis-stats.jp/%E6%B7%B1%E5%B1%9E%E5%AD%A6%E7%BF%92/%E7%94%BB%E5%83%8F%E5%88%86%E9%A1%9E%E3%83%BB%E7%89%A9%E4%BD%93%E6%A4%9C%E5%87%BA%E3%83%BB%E3%82%BB%E3%82%B0%E3%83%A1%E3%83%B3%E3%83%86%E3%83%BC%E3%82%B7%E3%83%A7%E3%83%B3%E3%81%AE%E6%AF%94%E8%BC%83/

画像認識(Image Recognition)の分類

- セグメンテーション：Segmentation
  - パノプティック・セグメンテーション：Panoptic Segmentation
    - 全てのピクセルにラベルが振る
    - 数えられる物体は、個別で認識



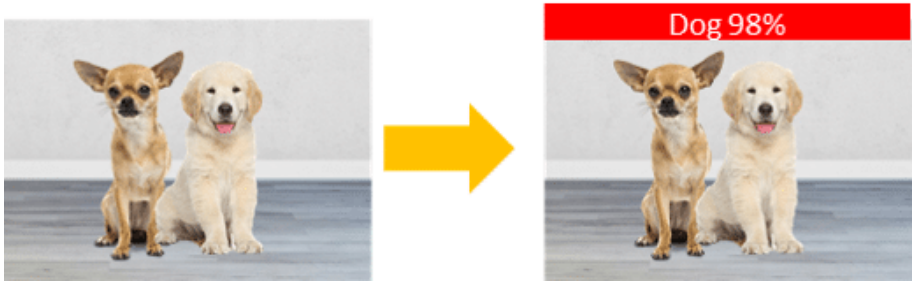
- 参照：画像分類・物体検出・セグメンテーションの比較
  - https://data-analysis-stats.jp/%E6%B7%B1%E5%B1%9E%E5%AD%A6%E7%BF%92/%E7%94%BB%E5%83%8F%E5%88%86%E9%A1%9E%E3%83%BB%E7%89%A9%E4%BD%93%E6%A4%9C%E5%87%BA%E3%83%BB%E3%82%BB%E3%82%B0%E3%83%A1%E3%83%B3%E3%83%86%E3%83%BC%E3%82%B7%E3%83%A7%E3%83%B3%E3%81%AE%E6%AF%94%E8%BC%83/

ニューラルネットワークの拡張

- 画像処理における拡張
  - 画像認識タスクの分類
  - 画像認識モデル
    - 畳み込みニューラルネットワーク
  - 画像データの作成
    - アノテーションの付与
  - 画像データの増強
    - Data Augmentation (データの水増し手法)の導入

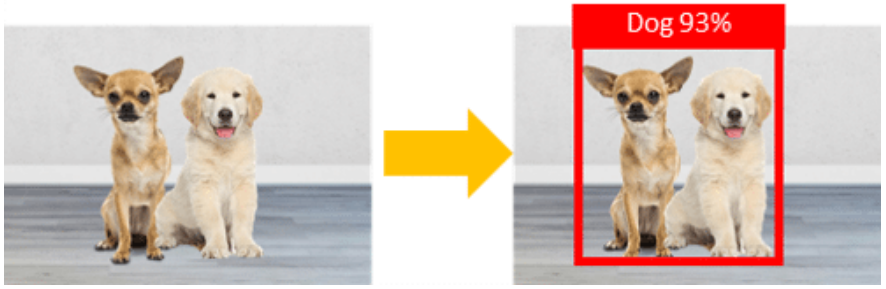
画像認識(Image Recognition)の分類

- 画像分類：Image Classification



出力：クラスと信頼度

- 画像分類・物体位置特定：Image Classification・Localization



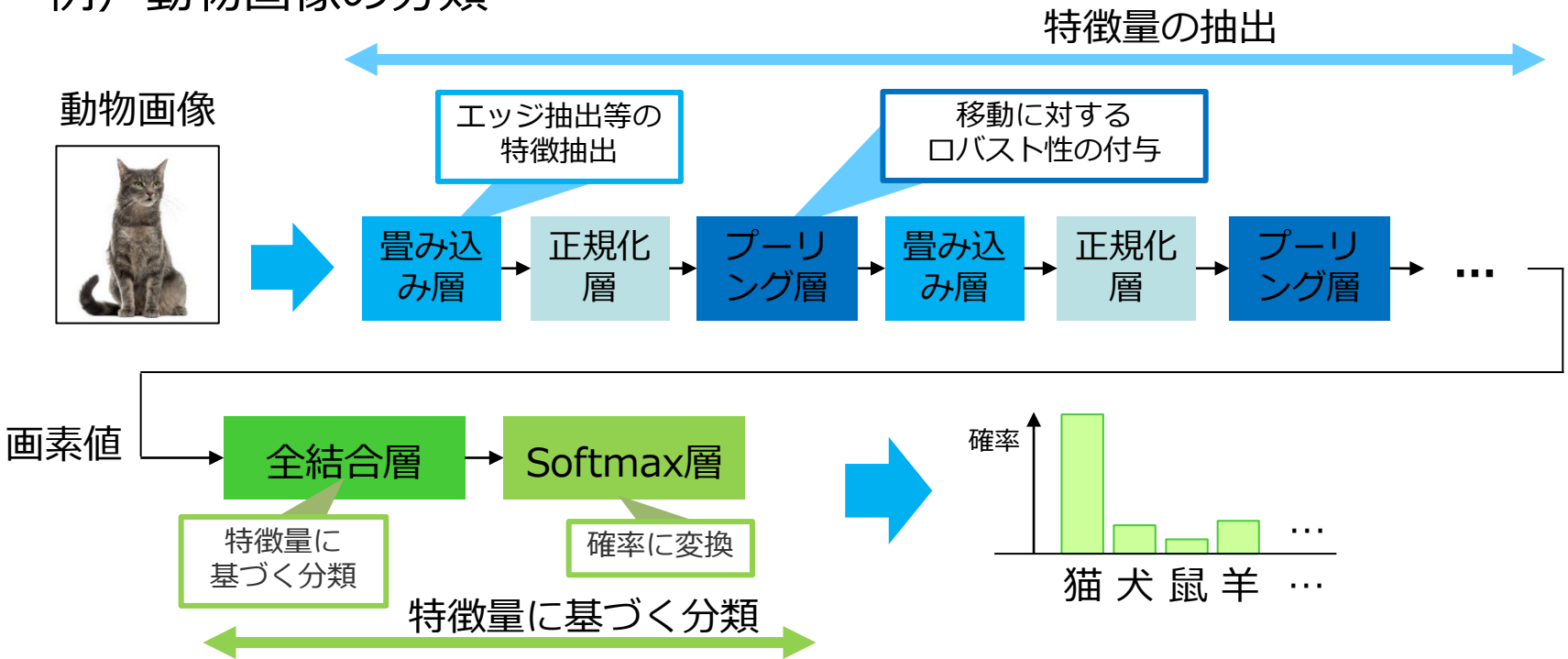
出力：クラスとその物体を含む矩形（バウンディングボックス

– 参照：画像分類・物体検出・セグメンテーションの比較
  - https://data-analysis-stats.jp/%E6%B7%B1%E5%B1%9E%E5%AD%A6%E7%BF%92/%E7%94%BB%E5%83%8F%E5%88%86%E9%A1%9E%E3%83%BB%E7%89%A9%E4%BD%93%E6%A4%9C%E5%87%BA%E3%83%BB%E3%82%BB%E3%82%B0%E3%83%A1%E3%83%B3%E3%83%86%E3%83%BC%E3%82%B7%E3%83%A7%E3%83%B3%E3%81%AE%E6%AF%94%E8%BC%83/

畳み込みニューラルネットワーク

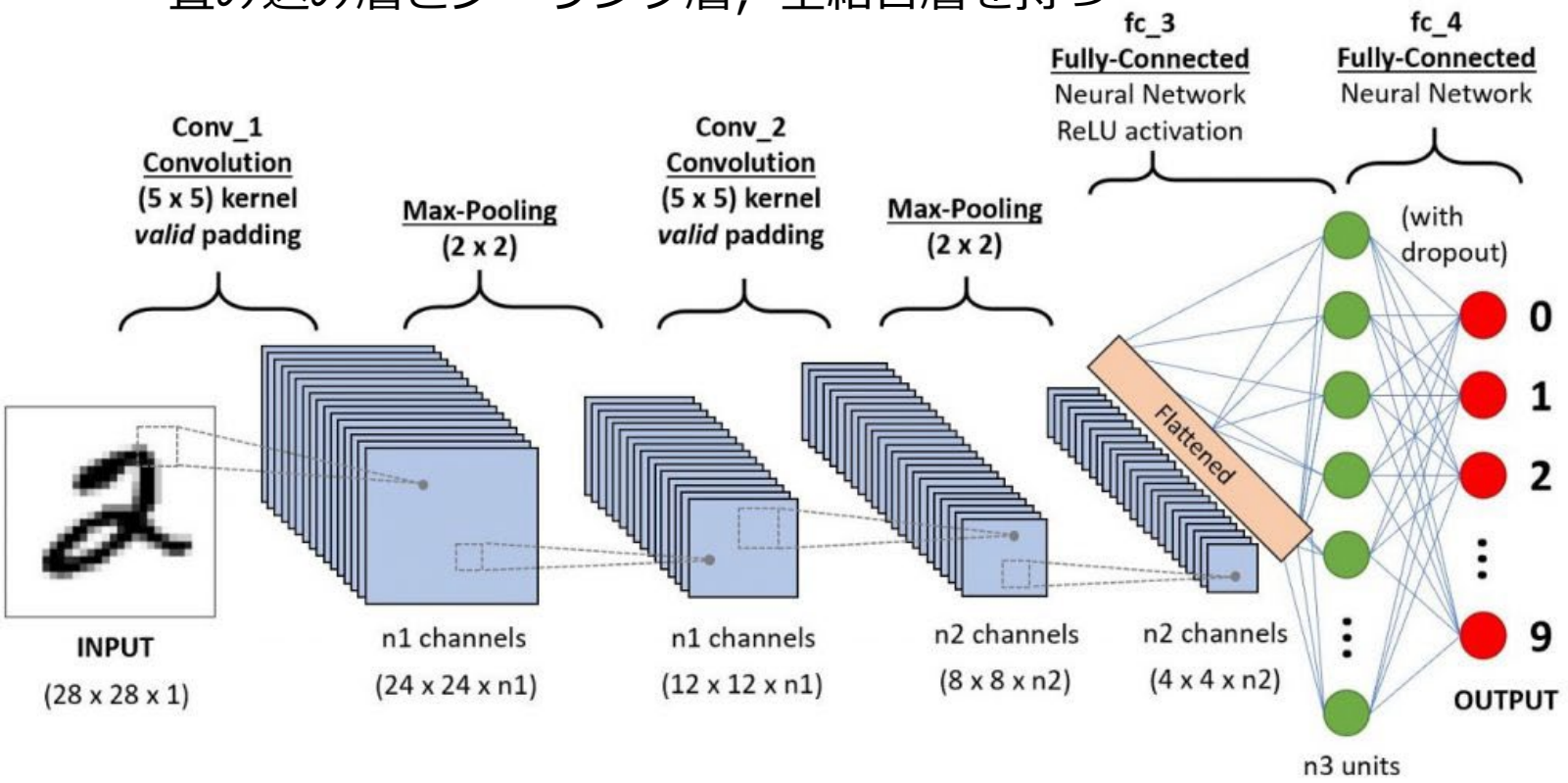
- 畳み込みニューラルネットワーク
  (**C**onvolutional **N**eural **N**etwork: CNN)
  - 動物の脳の視覚野の構造に基づくモデル
  - 順伝播型ネットワーク

例）動物画像の分類



CNNの概要

- CNN(畳み込みニューラルネットワーク)
  (**C**onvolutional **N**eural **N**etwork: CNN)
  - 動物の脳の視覚野の構造に基づくモデル
  - 入力を画像として画像分類を行うニューラルネットワーク
  - 畳み込み層とプーリング層，全結合層を持つ



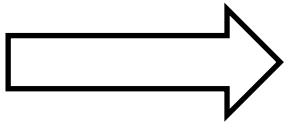
出典：A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science

CNNが画像処理に用いられる理由

- 一般的なニューラルネットワークは情報量の多い画像の処理に向かない

画像データ：ピクセル値の行列

0	3	4
9	2	1
8	5	7



行列をベクトルに変換

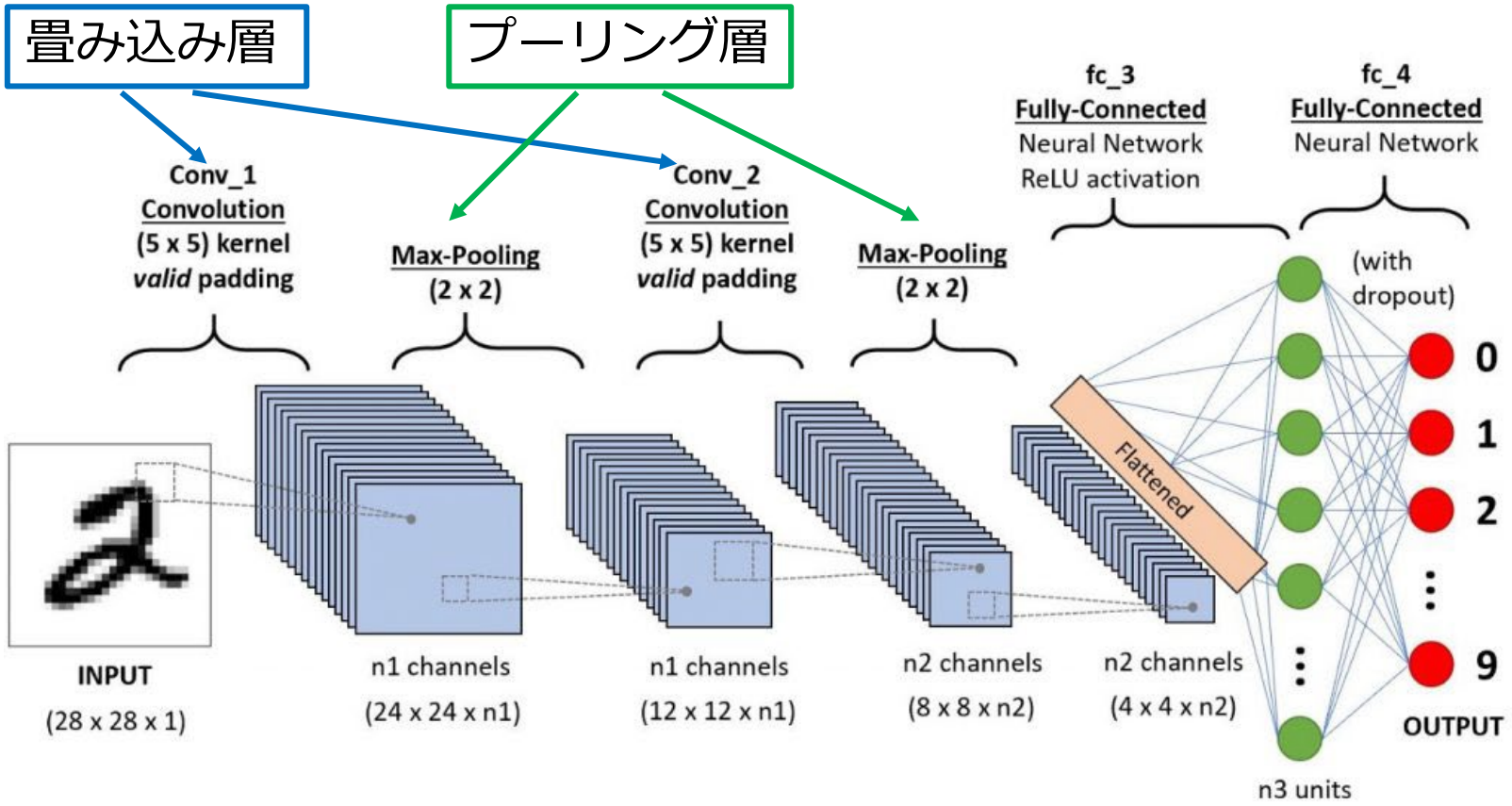
0
3
4
9
2
1
8
5
7

画素の依存関係情報を保持できない

- CNNは画像データを2次元データで処理
  - 画像内の空間的，時間的な依存関係の情報を抽出可能

CNNの各層の処理

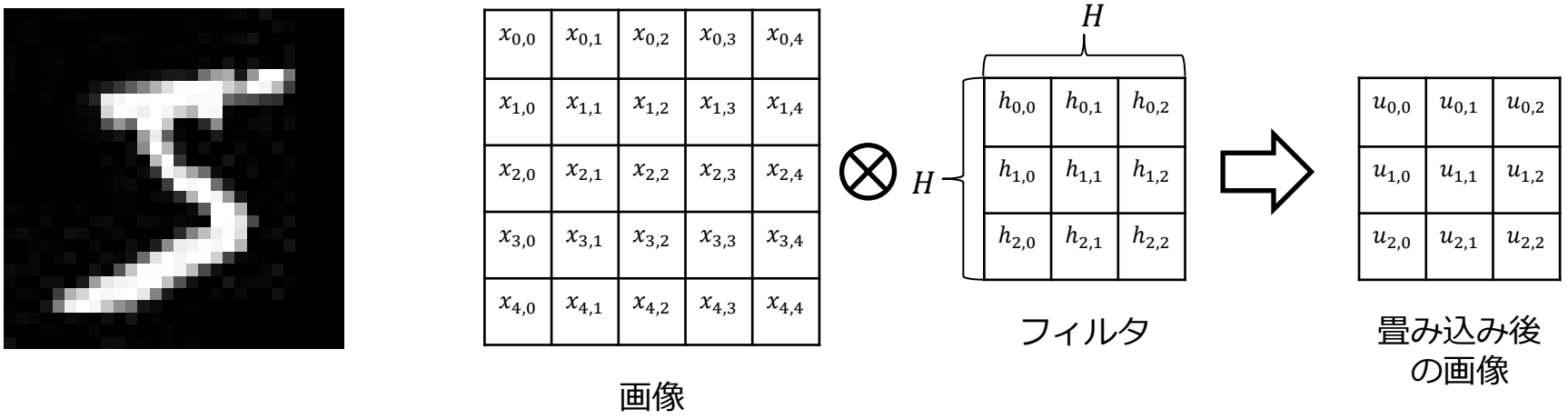
- CNN(畳み込みニューラルネットワーク)
  - 入力を画像として画像分類を行うニューラルネットワーク
  - 畳み込み層とプーリング層，全結合層を持つ



出典：A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science

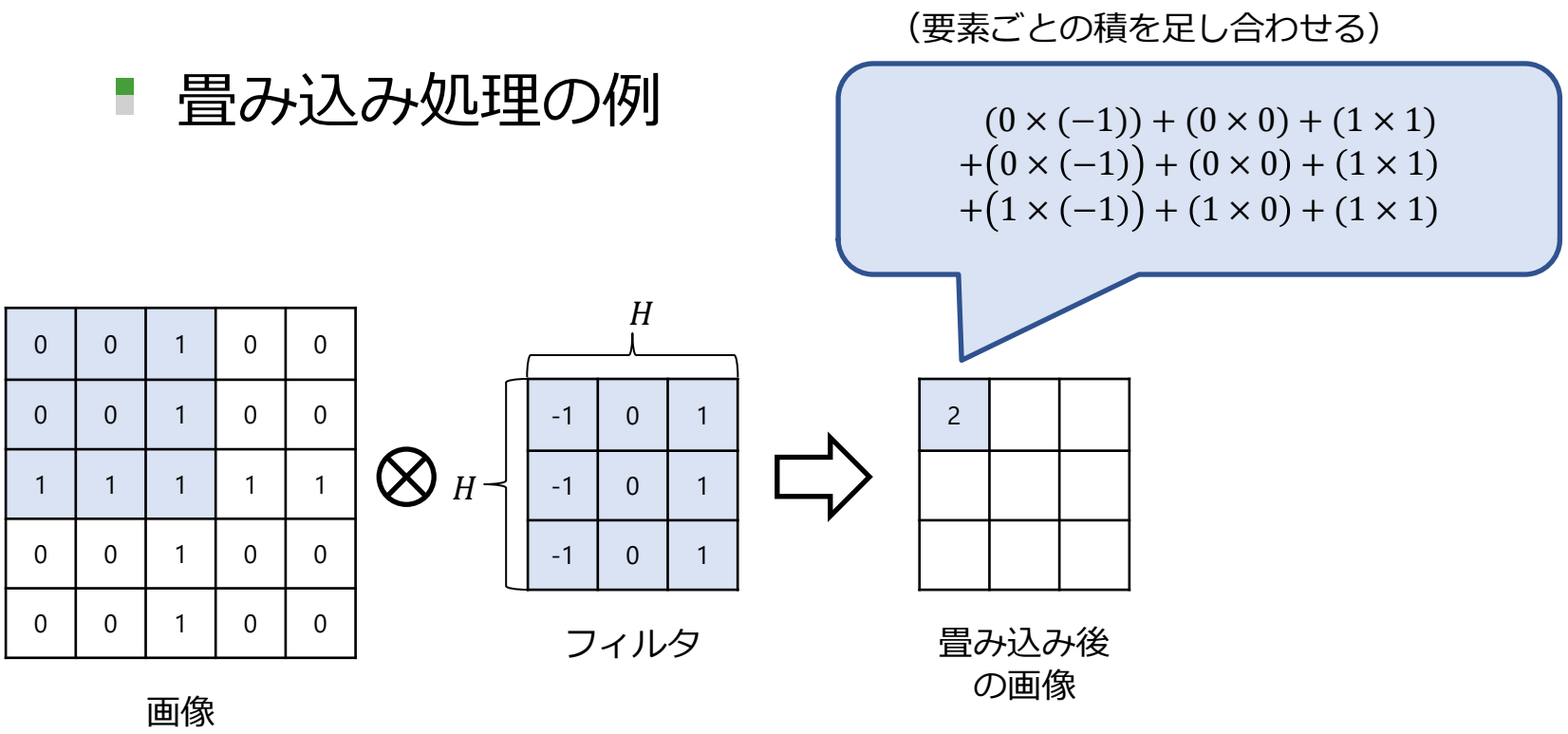
畳み込み層

- 畳み込み層で行っている処理
  - $u_{i,j} = \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} x_{i+p,j+q}\, h_{p,q}$
    - H:フィルタサイズ



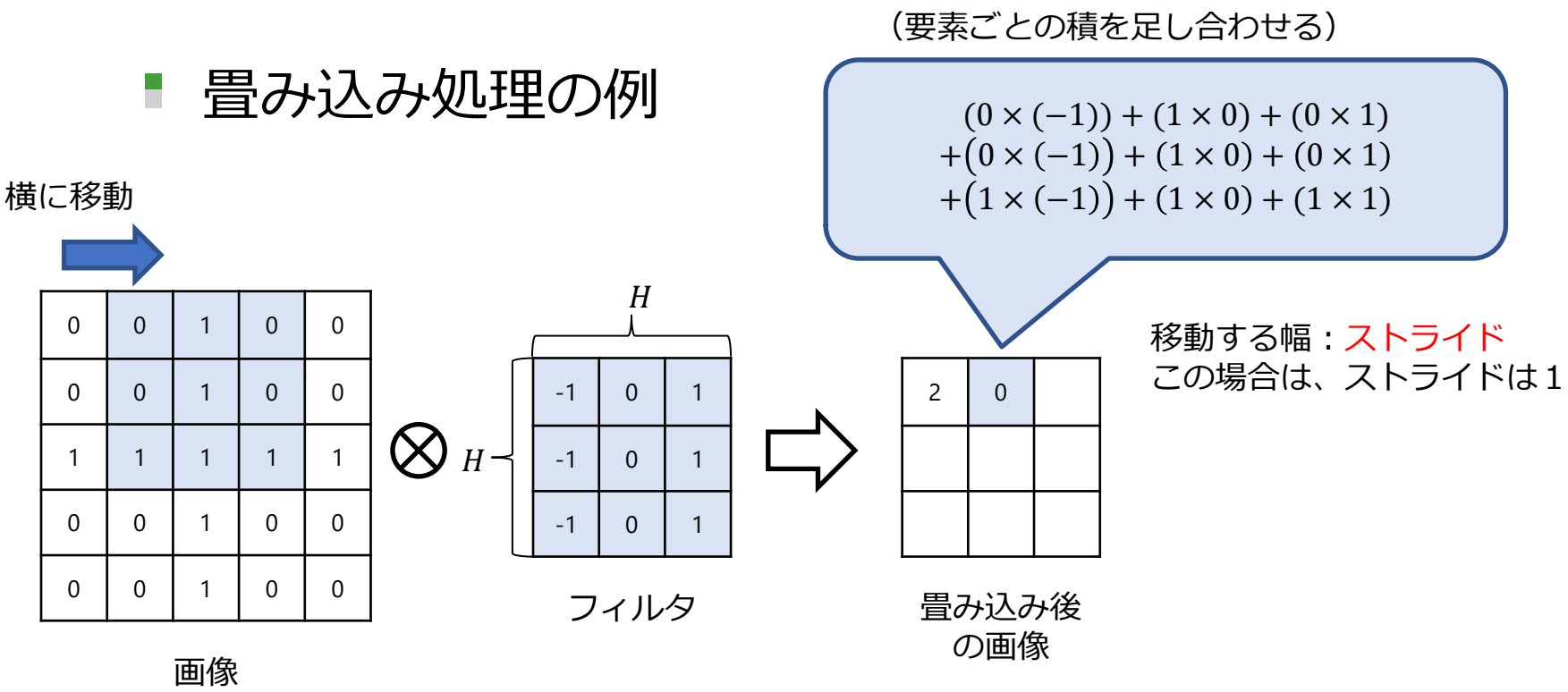
畳み込み層

- 畳み込み層で行っている処理
  - $u_{i,j} = \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} x_{i+p,j+q}\, h_{p,q}$
    - H:フィルタサイズ
  - 畳み込み処理の例



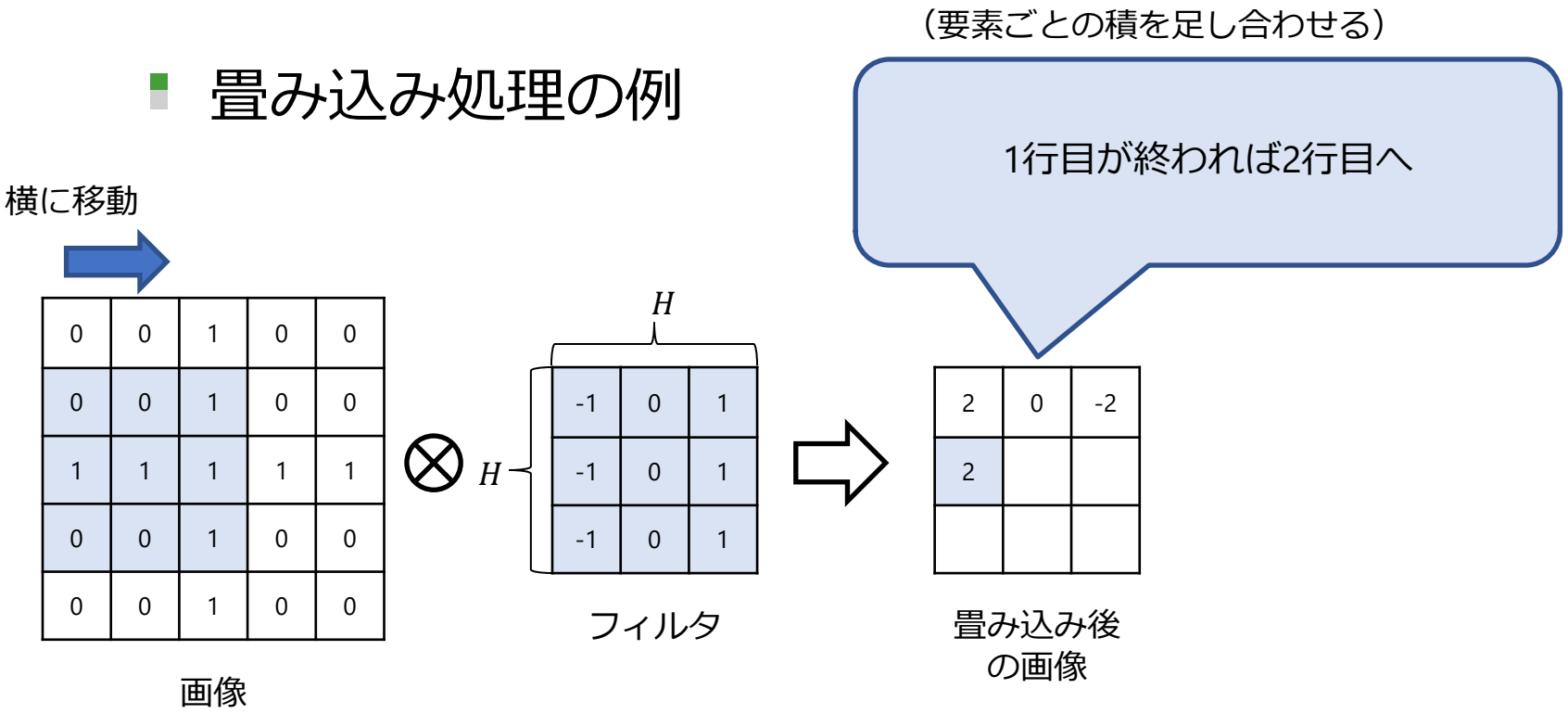
畳み込み層

- 畳み込み層で行っている処理
  - $u_{i,j} = \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} x_{i+p,j+q}\, h_{p,q}$
    - H:フィルタサイズ
  - 畳み込み処理の例



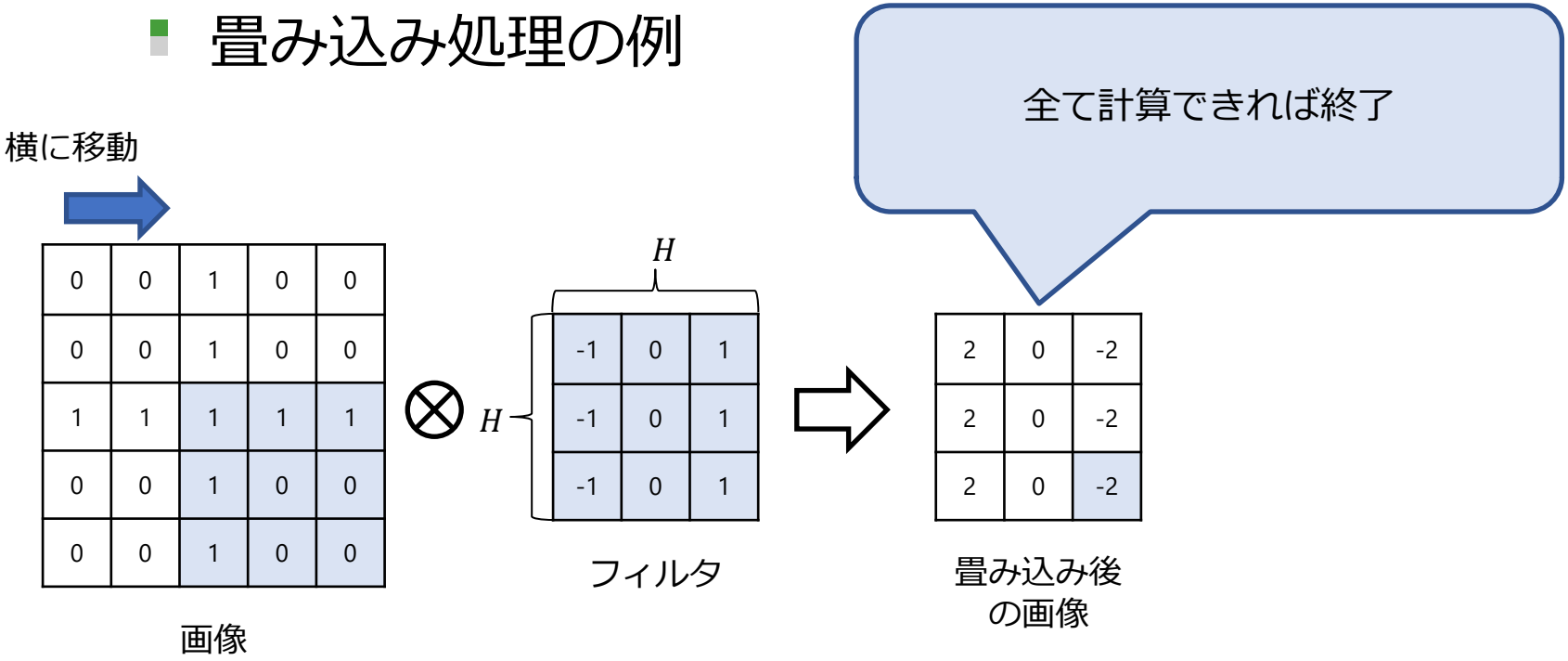
畳み込み層

- 畳み込み層で行っている処理
  - $u_{i,j} = \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} x_{i+p,j+q}\, h_{p,q}$
    - H:フィルタサイズ
  - 畳み込み処理の例



畳み込み層

- 畳み込み層で行っている処理
  - $u_{i,j} = \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} x_{i+p,j+q}\, h_{p,q}$
    - H:フィルタサイズ
  - 畳み込み処理の例



畳み込み層

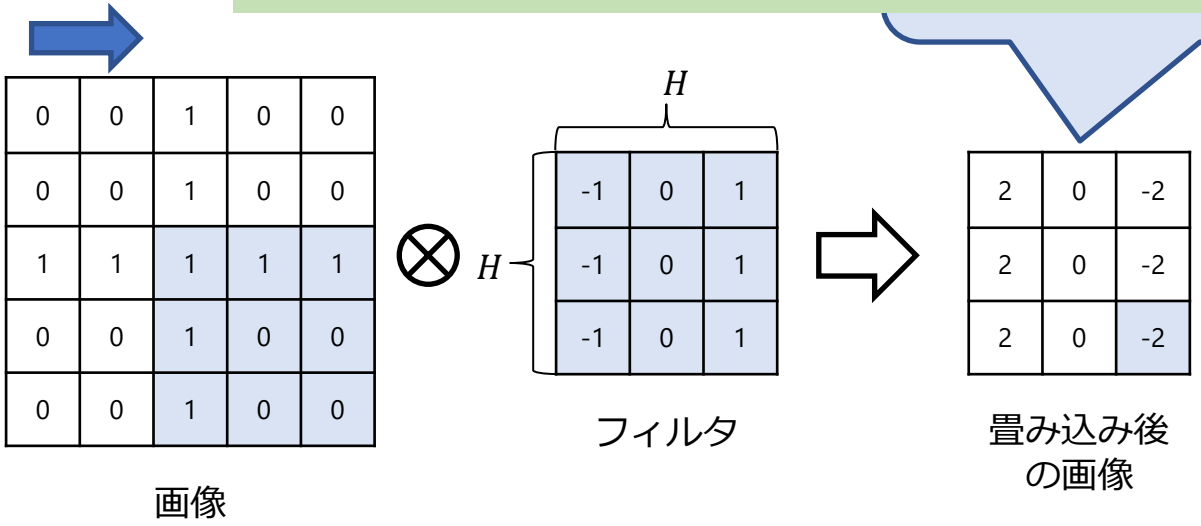
ポイント

- 画像全体に対してフィルタを移動させて適用

↓

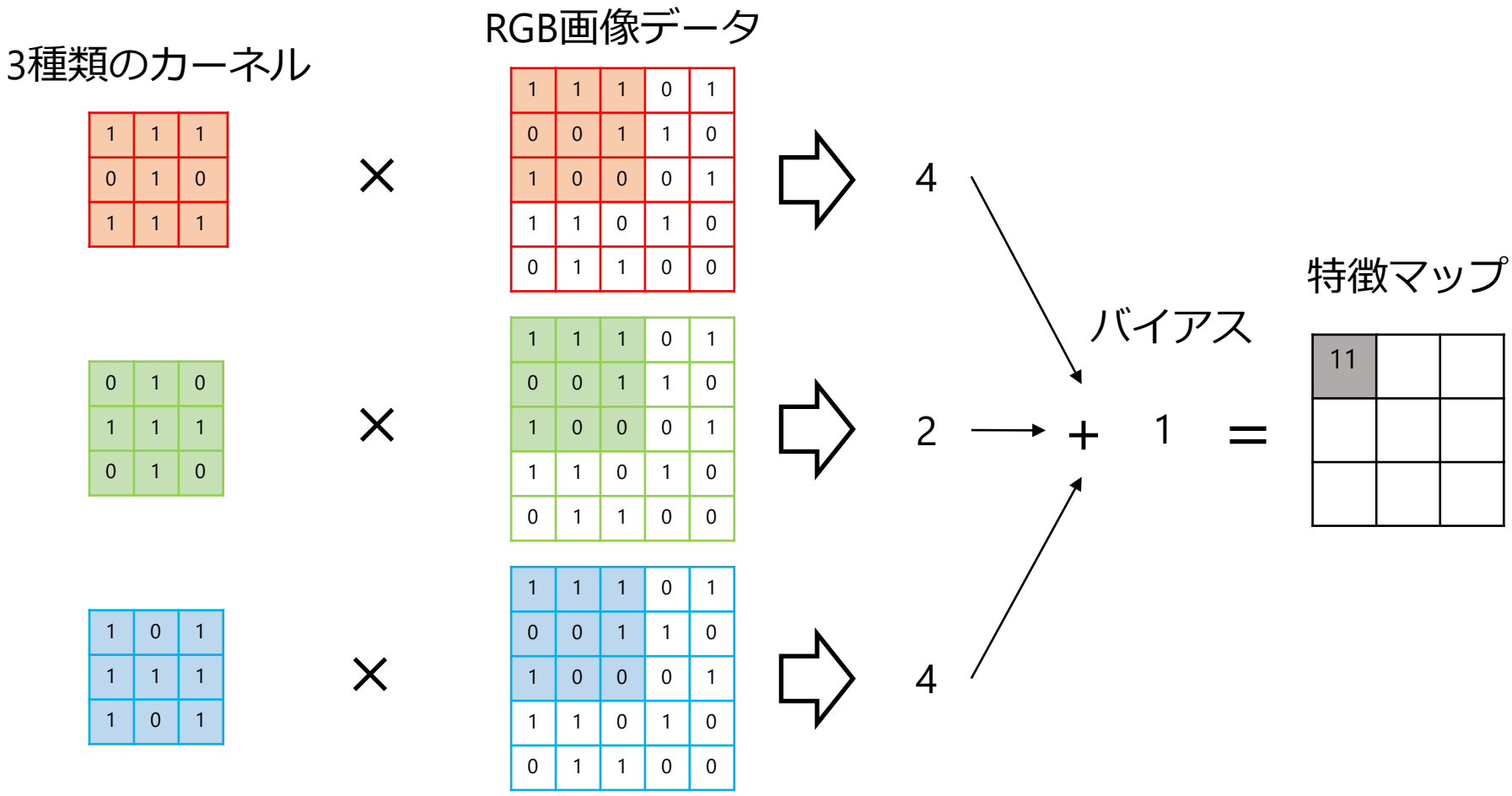
特徴が画像内のどこにあっても抽出可能（位置不変性）

横に移動



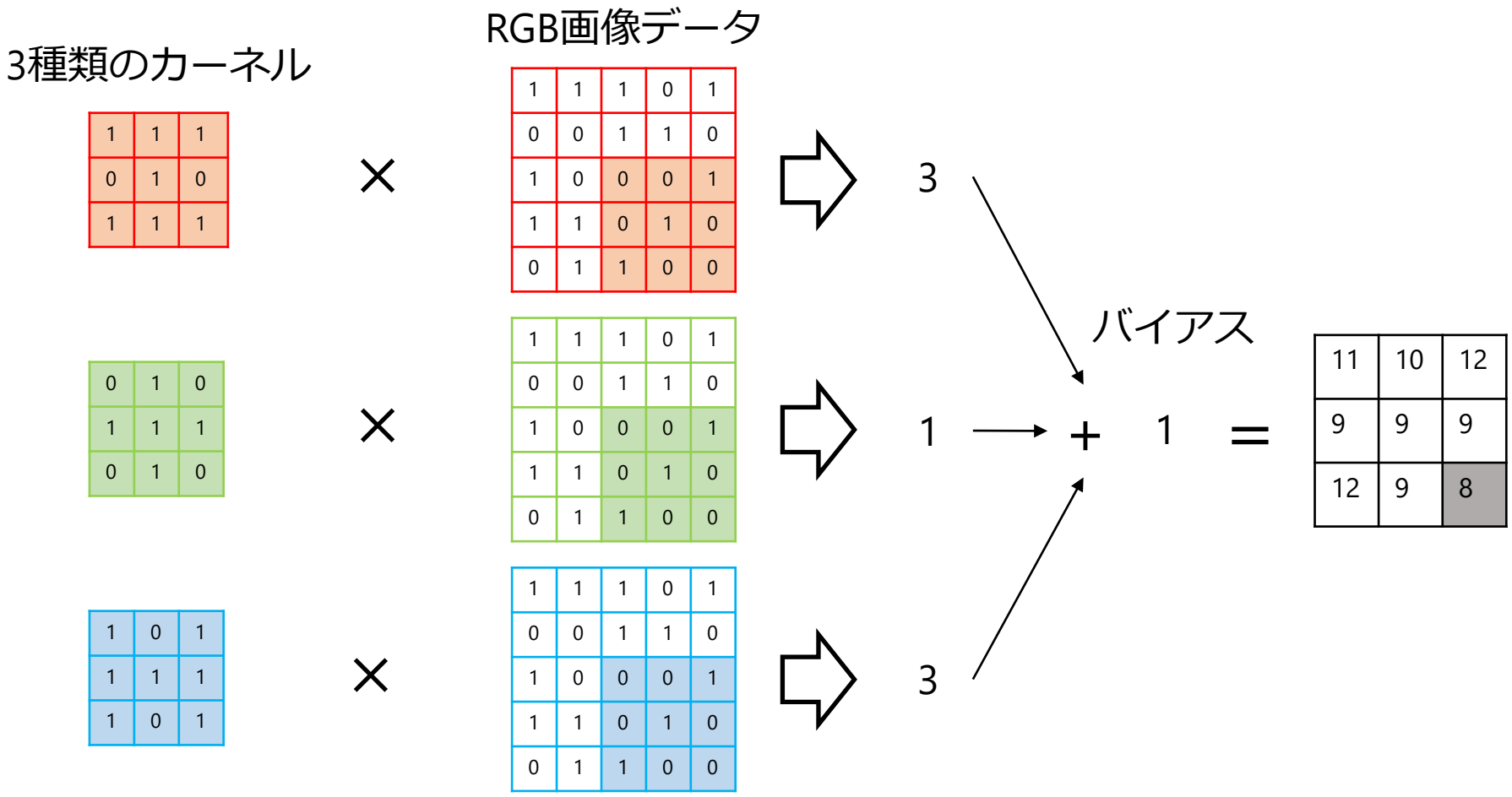
畳み込み層

- 入力が複数の色情報を持つ画像の場合
  - 色情報と同じチャンネル数のカーネルによる畳み込み結果とバイアスを足し合わせる



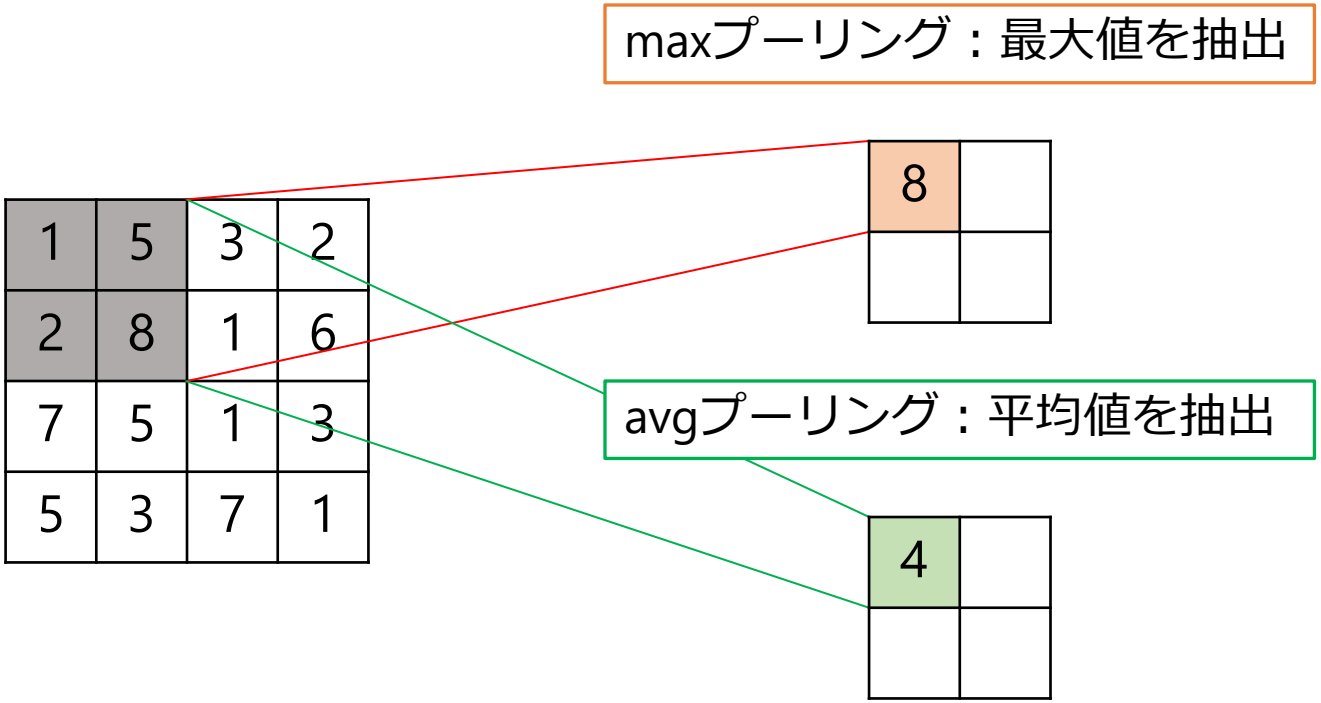
畳み込み層

- 入力が複数の色情報を持つ画像の場合
  - 色情報と同じチャンネル数のカーネルによる畳み込み結果とバイアスを足し合わせる



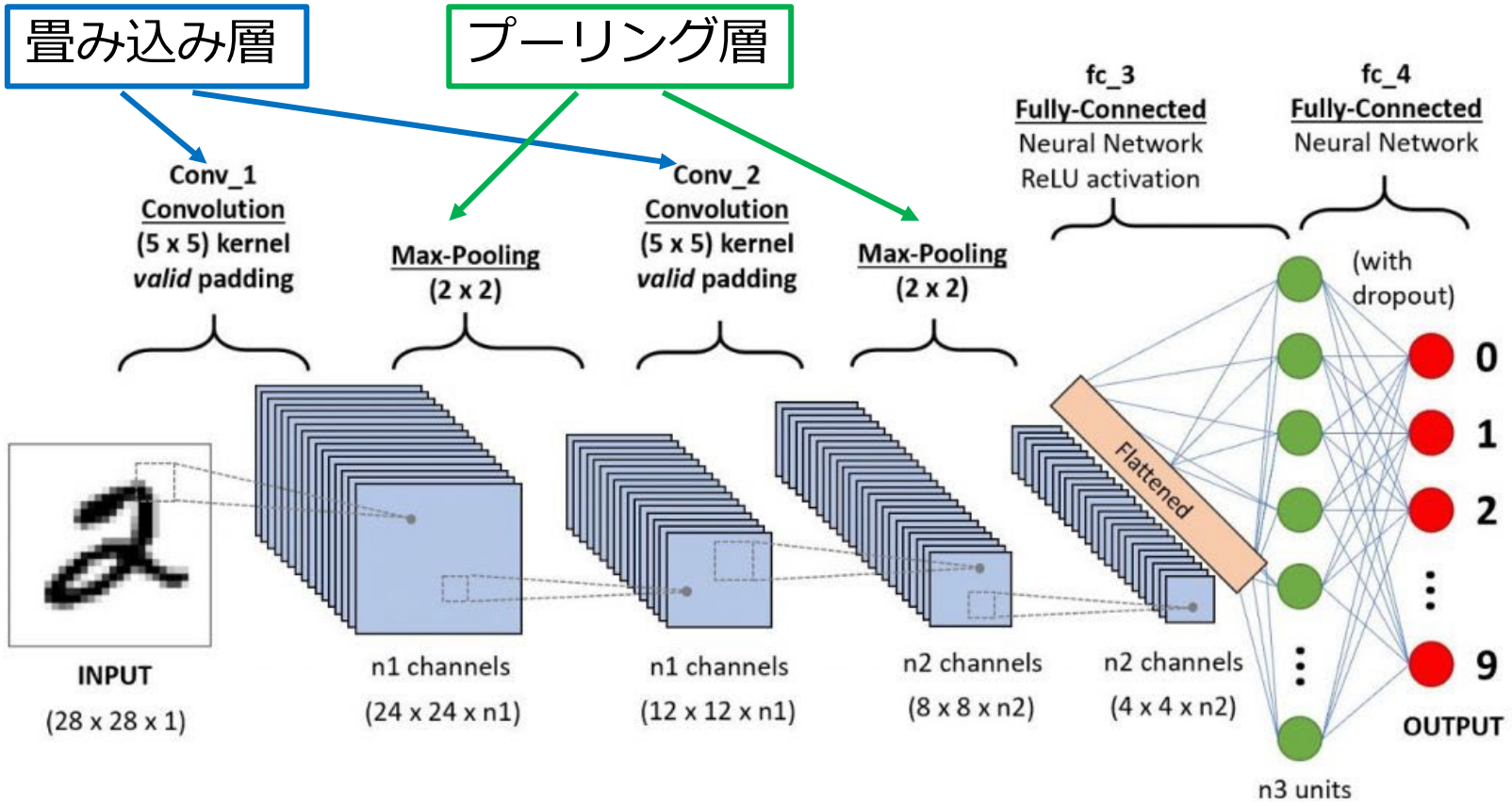
プーリング層

- 特徴マップを縮小し新たな特徴マップを生成する層
  - データの次元削減によって計算コストを下げる
  - 画像の位置移動に対する普遍性を得る
  - ノイズを抑える



CNNの各層の処理

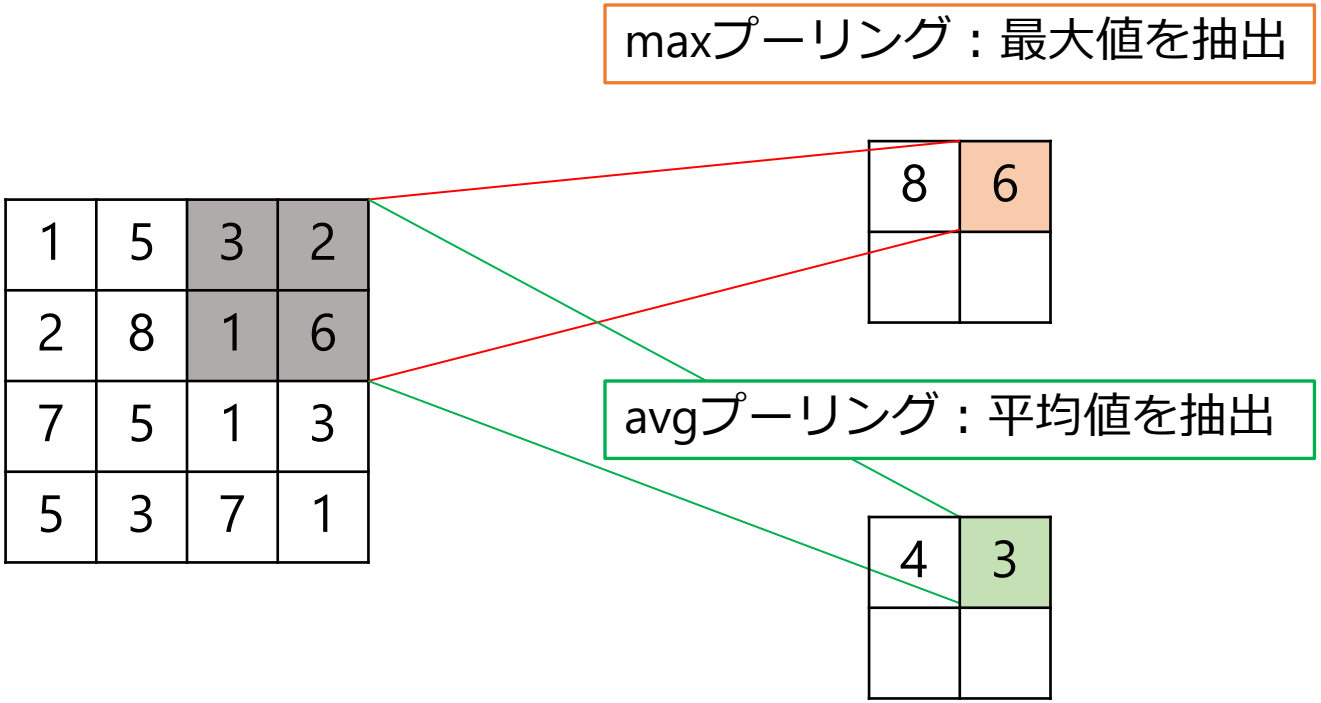
- CNN(畳み込みニューラルネットワーク)
  - 入力を画像として画像分類を行うニューラルネットワーク
  - 畳み込み層とプーリング層，全結合層を持つ



出典：A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science

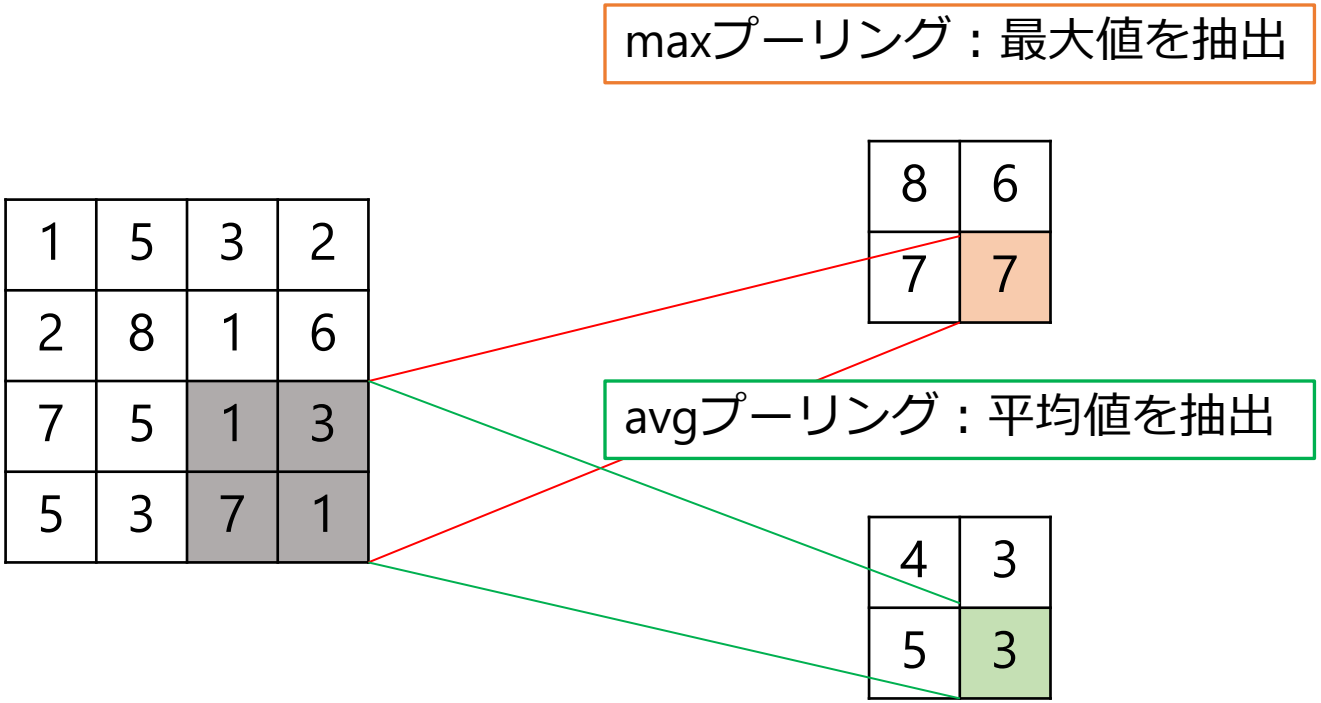
プーリング層

- 特徴マップを縮小し新たな特徴マップを生成する層
  - データの次元削減によって計算コストを下げる
  - 画像の位置移動に対する普遍性を得る
  - ノイズを抑える



プーリング層

- 特徴マップを縮小し新たな特徴マップを生成する層
  - データの次元削減によって計算コストを下げる
  - 画像の位置移動に対する普遍性を得る
  - ノイズを抑える

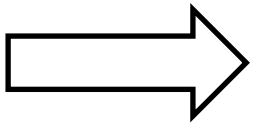


パディング

- 特徴マップの周りをピクセルで囲うテクニック
  - 畳み込み層やプーリング層で用いられる
- パディングの利点
  - 画像のサイズを保持
  - 特徴マップの端のピクセルの特徴量を中央のピクセルと同様に利用可能

- 例）ゼロパディング
  - 代表的なパディング手法の1つ

0	3	4
9	2	1
8	5	7

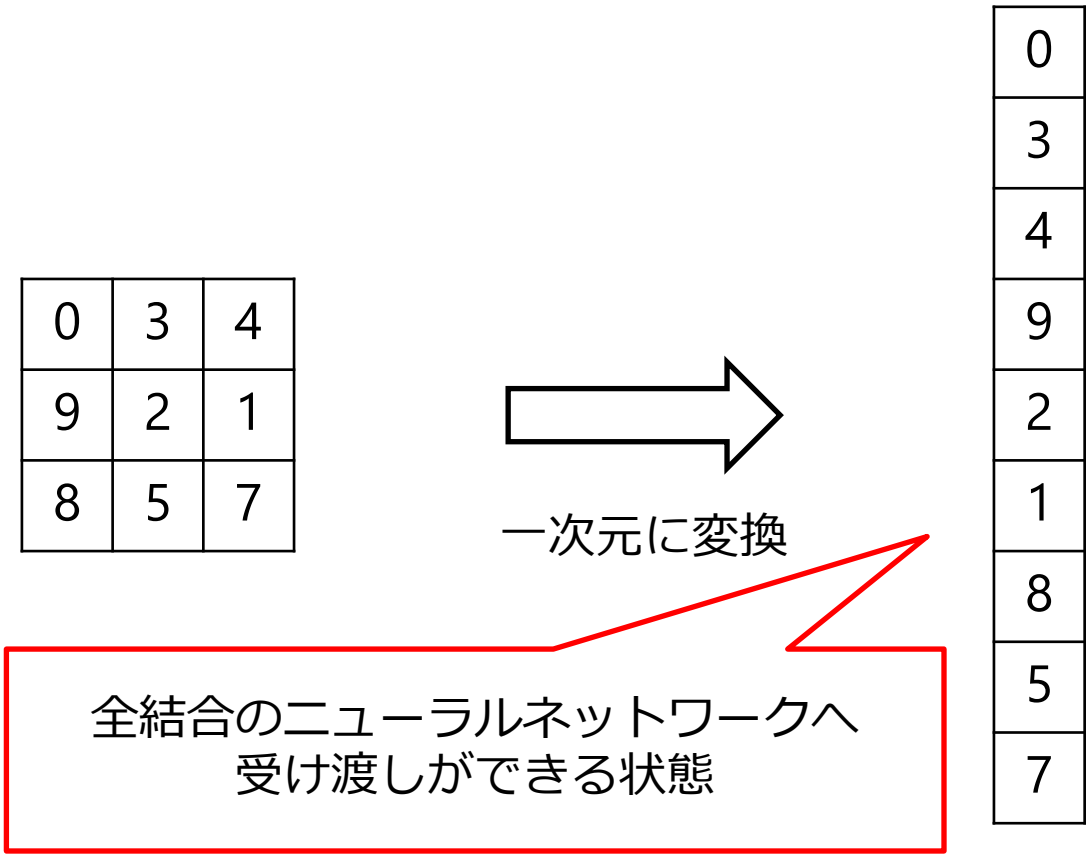


0で周りを囲う

0	0	0	0	0
0	0	3	4	0
0	9	2	1	0
0	8	5	7	0
0	0	0	0	0

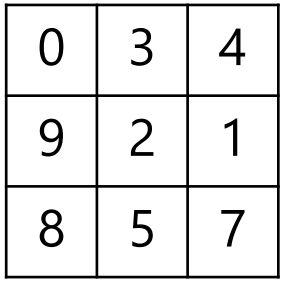
全結合層

- 特徴マップを列ベクトルに変換し分類する層
  - ニューラルネットワークへ受け渡すために一次元に変換
  - 活性化関数を用いて分類する
    - Softmax関数やSigmoid関数など

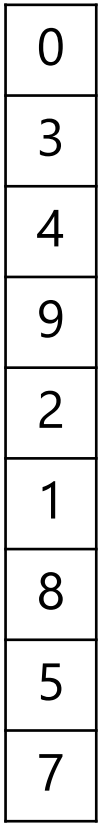


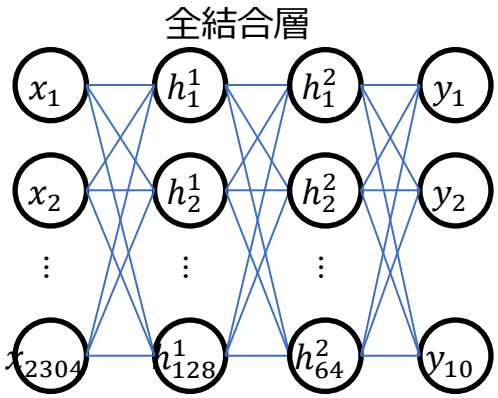
全結合層

- 特徴マップを列ベクトルに変換し分類する層
  - ニューラルネットワークへ受け渡すために一次元に変換
  - 活性化関数を用いて分類する
    - Softmax関数やSigmoid関数など



1次元に変換

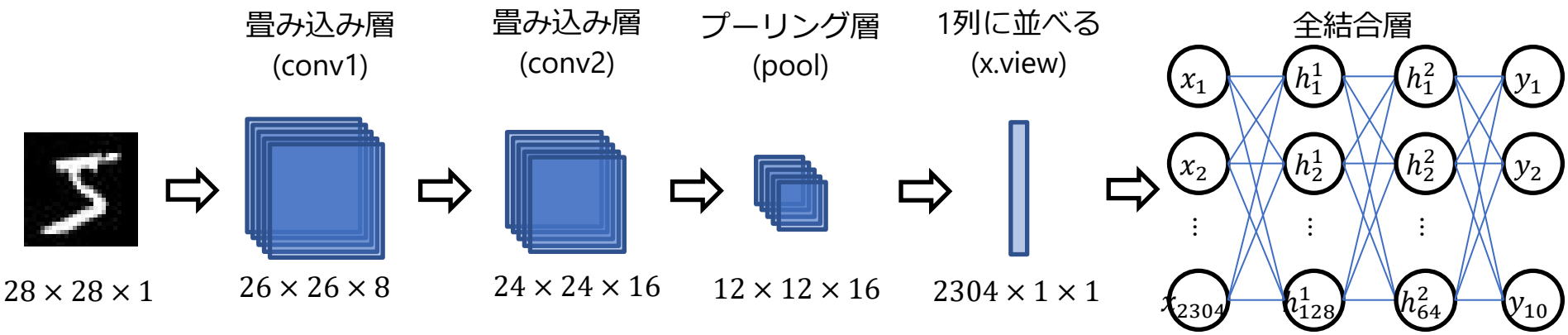




全結合のニューラルネットワークへ
受け渡しができる状態

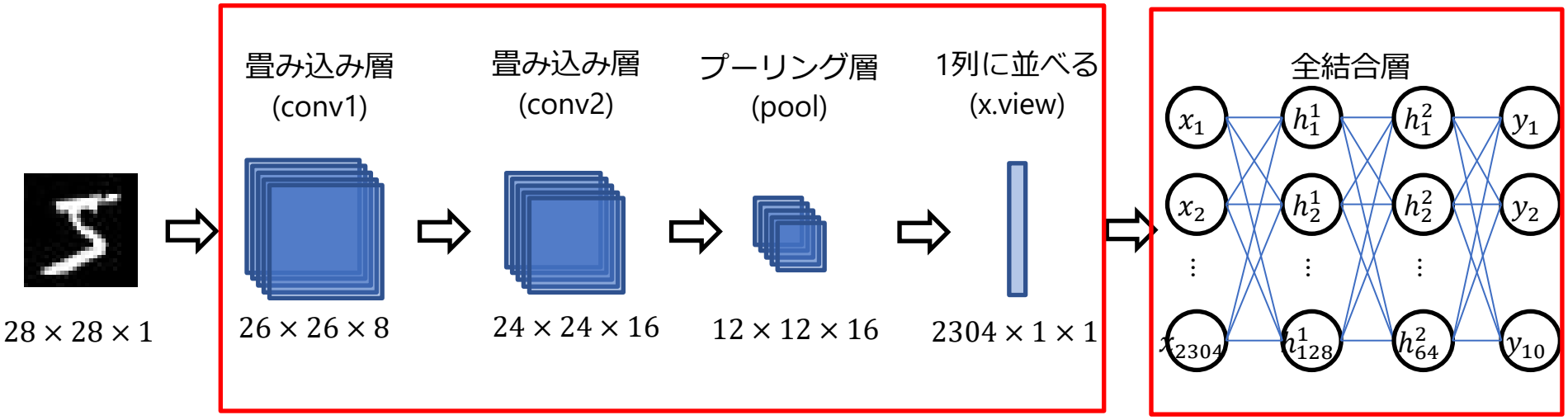
畳み込みニューラルネットワーク

- 画像サイズに合わせて作成



畳み込みニューラルネットワーク

- 画像サイズに合わせて作成

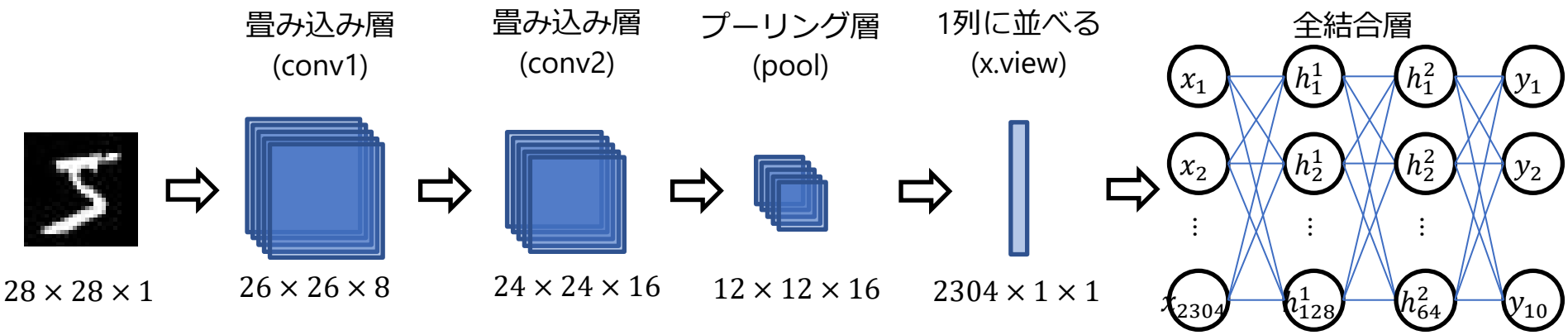


画像の特徴を抽出

どの特徴が分類に有効か学習

畳み込みニューラルネットワーク

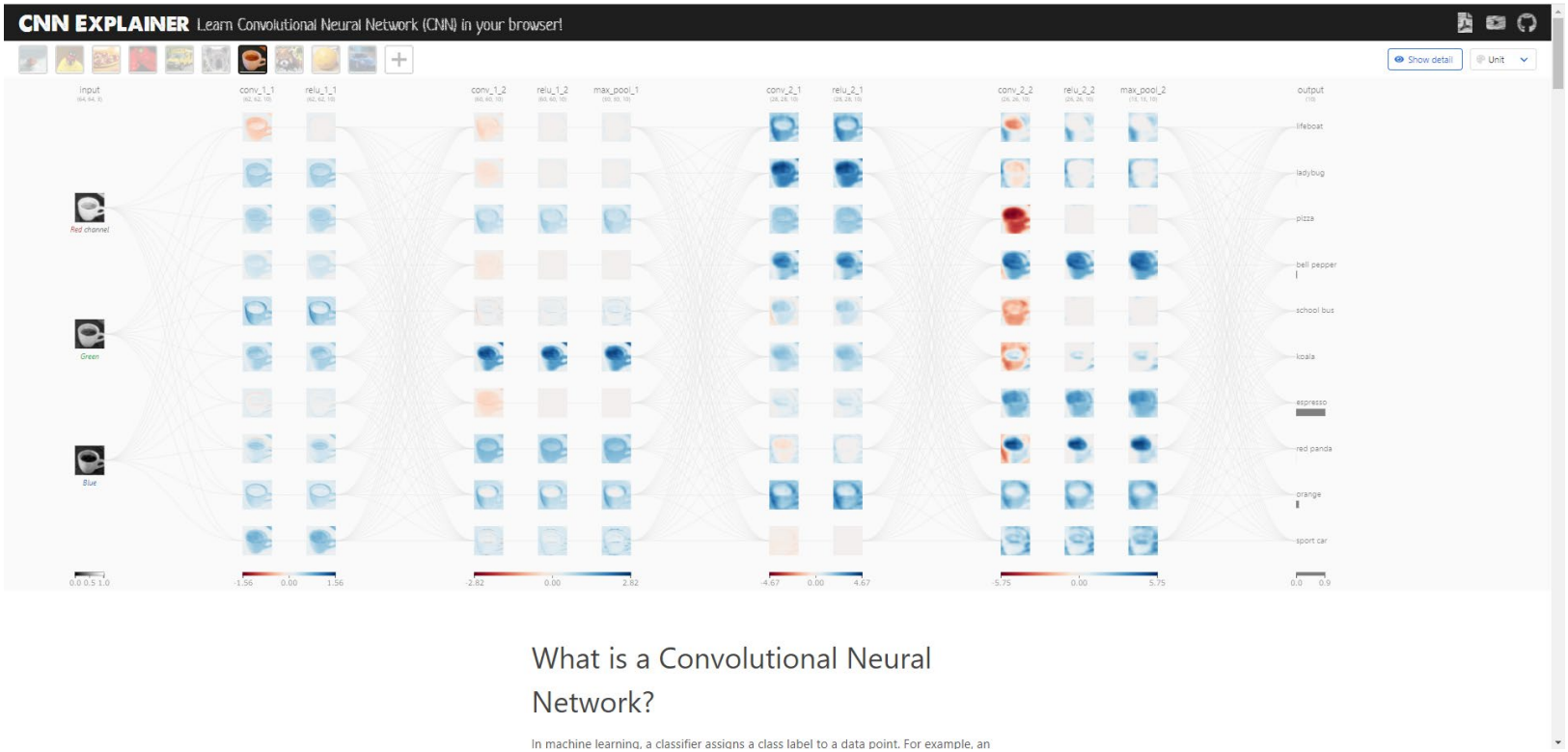
- 画像サイズに合わせて作成



- 実際にCNNを使う際は学習済みの CNN を使うことが多い
  - 既存のCNNのサイズに合わせて画像をリサイズ
    - 画像サイズがフリーなモデルもある

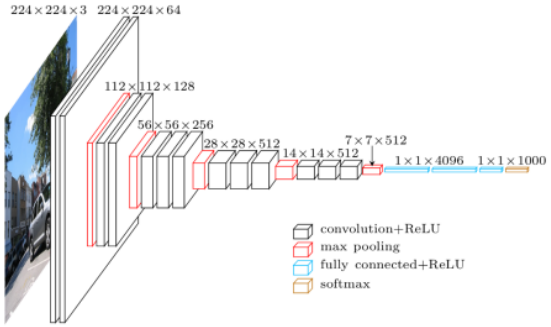
畳み込みニューラルネットワークの可視化

- CNN Explainer
  - 畳み込みニューラルネットワーク可視化ツール
    - https://poloclub.github.io/cnn-explainer/



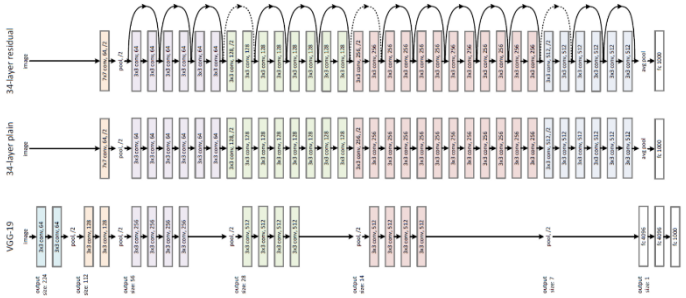
CNNの代表的なアーキテクチャ

- VGGNet
  - Oxford大学VGGチームが考案したアーキテクチャ
  - 16層，もしくは19層で構成させる
  - 小さいフィルタで連続して畳み込むことで特徴量をより良く抽出



出典：https://qiita.com/mine820/items/1e49bca6d215ce88594a

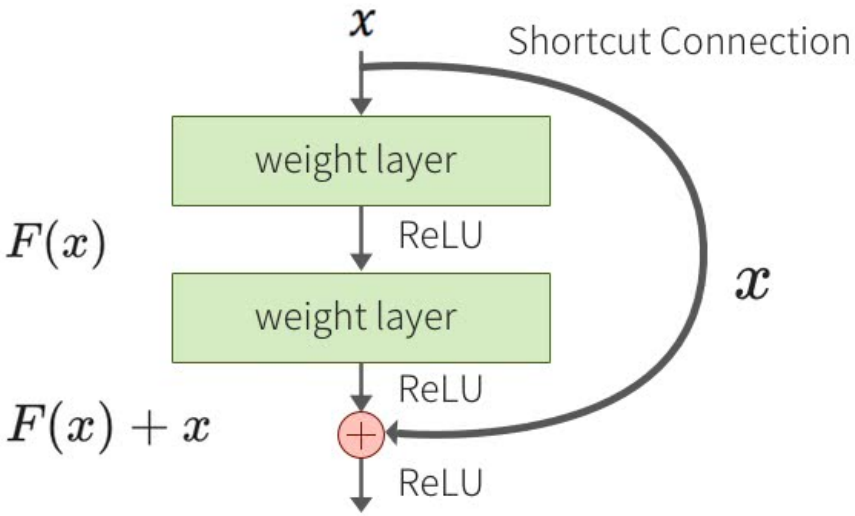
- ResNet
  - Facebook AI Researchの Kaiming He氏がに考案したアーキテクチャ
  - 152層という非常に深い層で構成される
  - Shortcut Connectionを利用した残差ブロックとBatch Normalizationを実装



出典：https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8

ResNetのアイデア

- Shortcut Connection
  - 畳み込みの出力結果に入力を足し合わせる
  - 層が深くなることで学習が進まなくなる問題を解決



出典：https://deepage.net/deep_learning/2016/11/30/resnet.html

- Batch Normalization
  - 全結合や畳み込みの後にノードの値をミニバッチ単位で正規化
  - ミニバッチ学習の際に入力の分布が偏り学習が進まなくなる問題を解決

ニューラルネットワークの拡張

- 画像処理における拡張
  - 画像認識タスクの分類
  - 画像認識モデル
    - 畳み込みニューラルネットワーク
  - 画像データの作成
    - アノテーションの付与
  - 画像データの増強
    - Data Augmentation (データの水増し手法)の導入

アノテーション

- データにタグを付ける作業
  - 対象データ：テキスト、音声、画像等
  - 正確にタグ付けされていないデータ
  - → 意図通りに学習できない

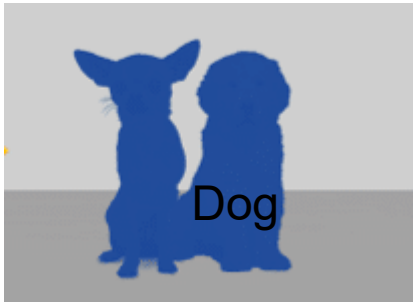
分類

画像にクラス名のアノテーションを付与(複数可：マルチラベル)

物体検出

画像中の対象物ににバウンディングボックスとクラス名のアノテーションを付与

セグメンテーション



画像の各ピクセルにクラス名のアノテーションを付与

ディープラーニングを使ったファッション画像の理解

共同研究先： TSIホールディングス、INSIGHT LAB株式会社

目的

洋服の画像から「かわいい」「甘い」といった「感性」を分析

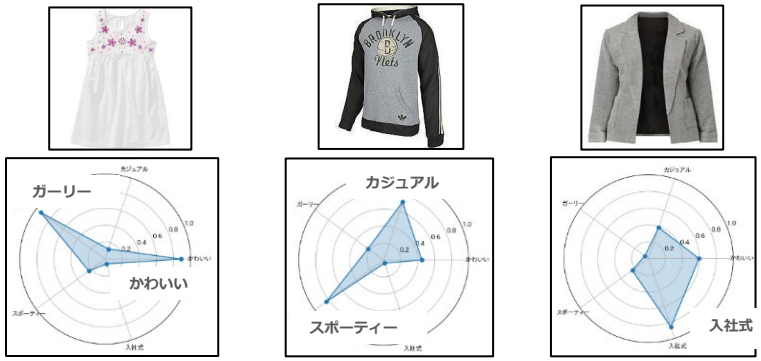
概要

2万枚の服飾画像に対し主観的・機能的な特徴をタグ付け

148種類のファッション用語
（ガーリー，かわいい，とろみ，ふんわり，スポーティー，
カジュアル，パーティー，女子会，着回ししやすい… ）

畳み込みネットワークによりタグを学習

洋服の画像を読み込み
消費者がその洋服を見た時のイメージを数値化



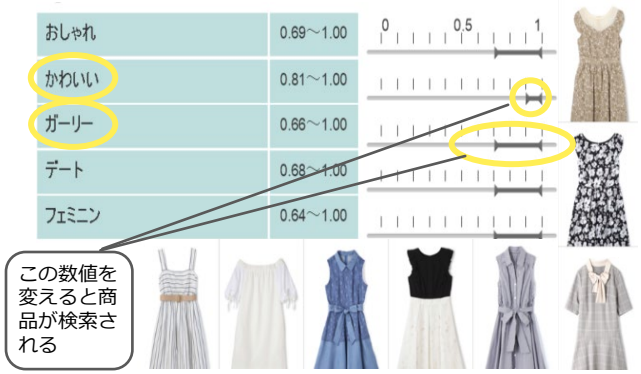
ディープラーニングによる認識結果

印象的な特徴：カジュアル　甘い　こっくり　休日のお出かけに　リラックスできる　ガーリー　とろみ　秋らしい　きれいめ

形状的な特徴：無地　ドロップショルダー　ファー　フリル

応用例

ファッションに物差しを作成することにより，ネット通販サイトのユーザーが「かわいい度は0.9以上」「ガーリー度は0.7以上」とグラフ上で数値を指定すると該当する商品を絞り込み表示できる



harmo-lab.jp 調和系工学研究室

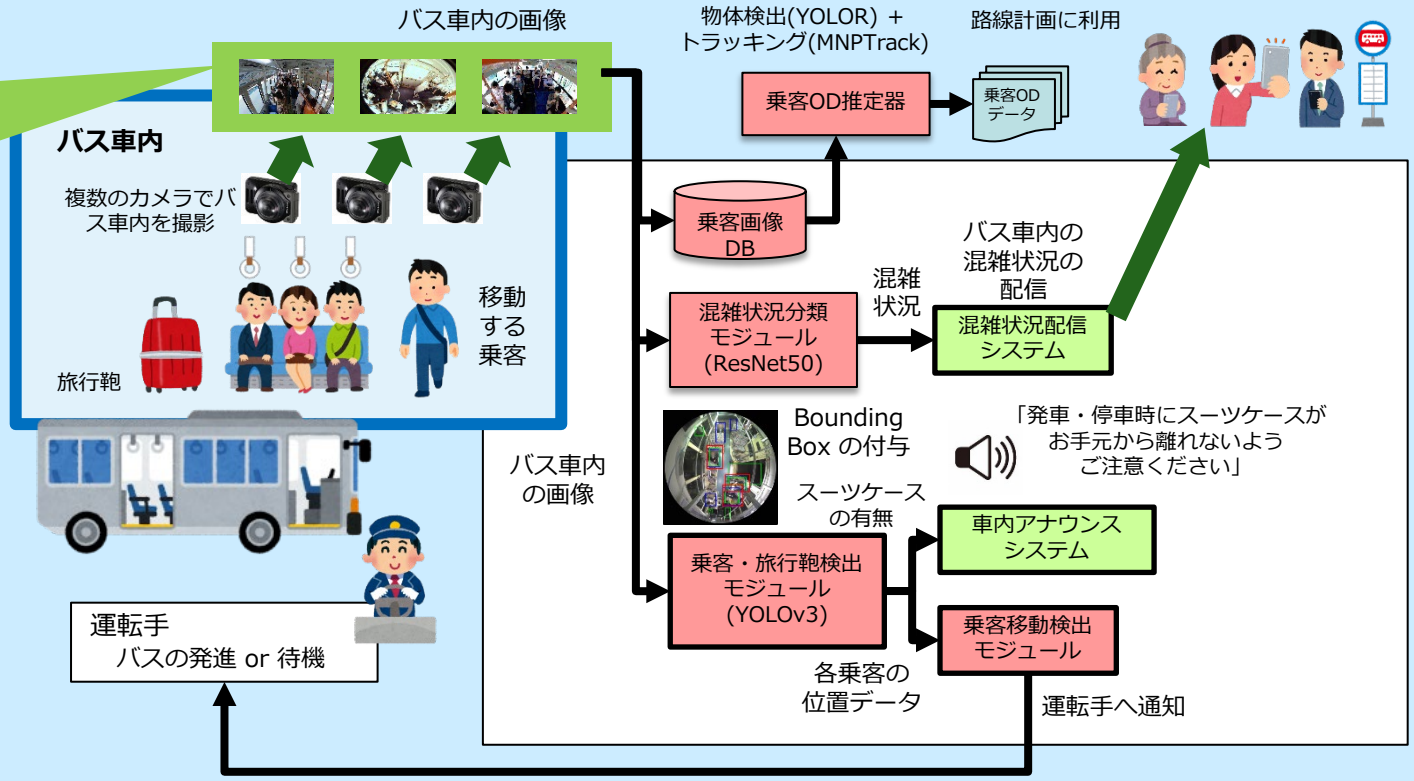
深層学習を用いたバス車内モニタリングシステムによる車内状況分析

共同研究先：
株式会社シーズ・ラボ
ノーステック財団の助成

目的
バス車内モニタリングシステムによる
公共交通機関としての
路線バスの利便性の向上

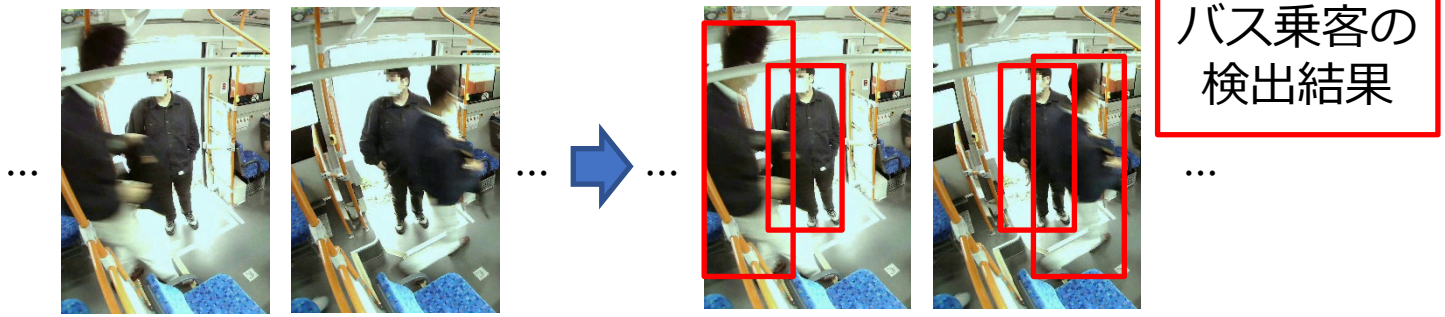
- 運行円滑化の支援
  - 路線バス車内画像から乗客数や混雑状況を推定
- 乗客の安全性の向上
  - バス車内の状況にリアルタイムに対応
    - 乗客の不意な車内移動
    - 乗客のどこにも掴まらずに立つ

バス車内モニタリングシステムの概要



1. 人物検出器によるバス乗客の検出

- 動画のフレームごとに人物検出器を適用



- 使用する人物検出器
  - 高い精度で推論可能な YOLOR [8]
  - MSCOCO [9] データセットを学習
  - 確信度 30% 以上で "Person" クラスと推定された結果をバス乗客とする
  - 体の一部のみ映る乗客やバスに乗り込まない人物を除外する
    - 人物検出器の推定結果から面積 30,000 ピクセル以下の矩形を除外

[8] Chien-Yao Wang and I-Hau Yeh and Hong-Yuan Mark Liao "You Only Learn One Representation: Unified Network for Multiple Tasks"(2021), arXiv
[9] Lin et al. "Microsoft COCO: Common Objects in Context"(2015), arXiv

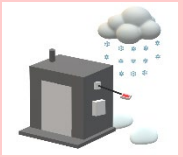
セマンティックセグメンテーションによる積雪状態の認識とロードヒーティングの制御

共同研究先：北海道ガス株式会社（共同で特許「融雪制御装置，ニュートラルネットワークの学習方法，融雪制御方法及び融雪制御用プログラム」出願中），ティ・アイ・エル株式会社（北大発認定ベンチャー企業）

・ロードヒーティングとは

- 駐車場などの地下に熱源を設置
- 熱により地表に積もる雪を融かす
-雪かきの省力化，転倒事故の防止

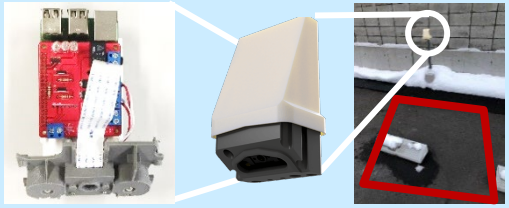
従来の降雪センサによるロードヒーティング制御



降雪センサ：屋外に設置され，表面の水分を検出

制御対象の路面積雪状態を考慮しないため無駄が多い

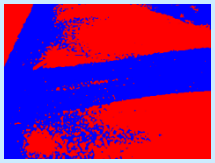
・画像認識によるロードヒーティング制御の開発



コントローラ（カメラ・マイコンを内蔵した融雪制御器）

撮影された路面画像

Google DeepLab v3+ をベースにしたモデルの利用（セマンティックセグメンテーションの適用）

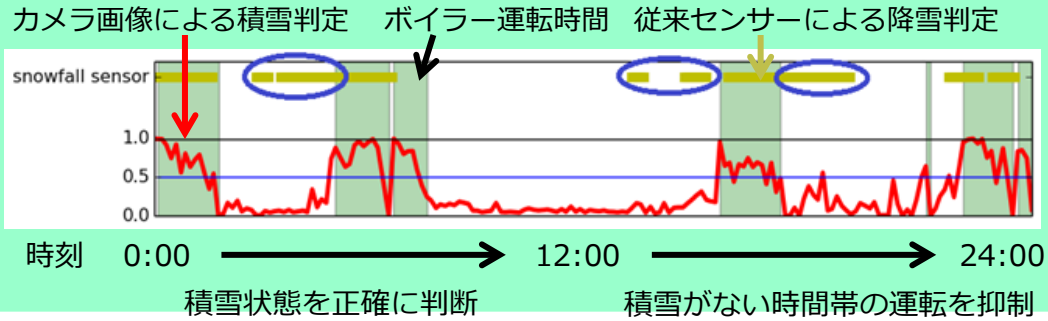


画像中の積雪部分を検出

路面に積雪があると判断すればボイラーを稼働

・適切な路面状況の把握による運転時間の削減

運転時間削減の例



- 路面の積雪状態に応じてボイラーを制御
- 融け残りを防止しつつ運転時間を削減

札幌市内の複数地点で実証実験を実施
全体で 40.5%程度のボイラー運転時間の削減効果を確認

俯瞰画像の利用

- 俯瞰画像
  - 融雪対象である領域を高い位置から撮影した画像

より広い範囲を撮影し積雪状況を認識した方が、融雪対象領域内の積雪の偏りの影響が少なく効果的な制御となる

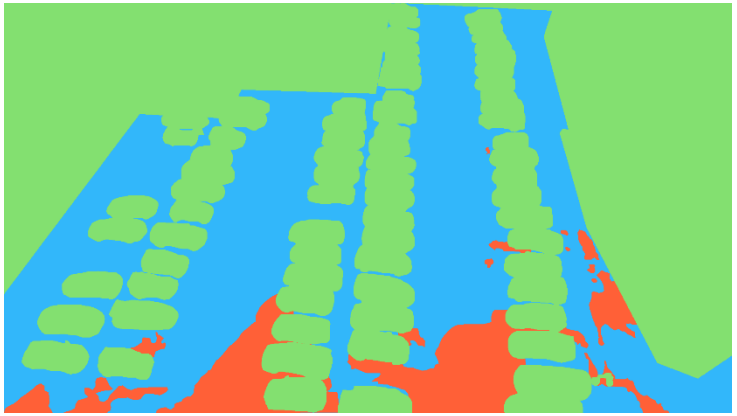
- 俯瞰画像を用いるうえでの課題
  - 路面画像の認識難易度が高い
    - 認識の障害になる物体（車、建物など）の多さ

認識の障害となるものが映り込んだ画像例

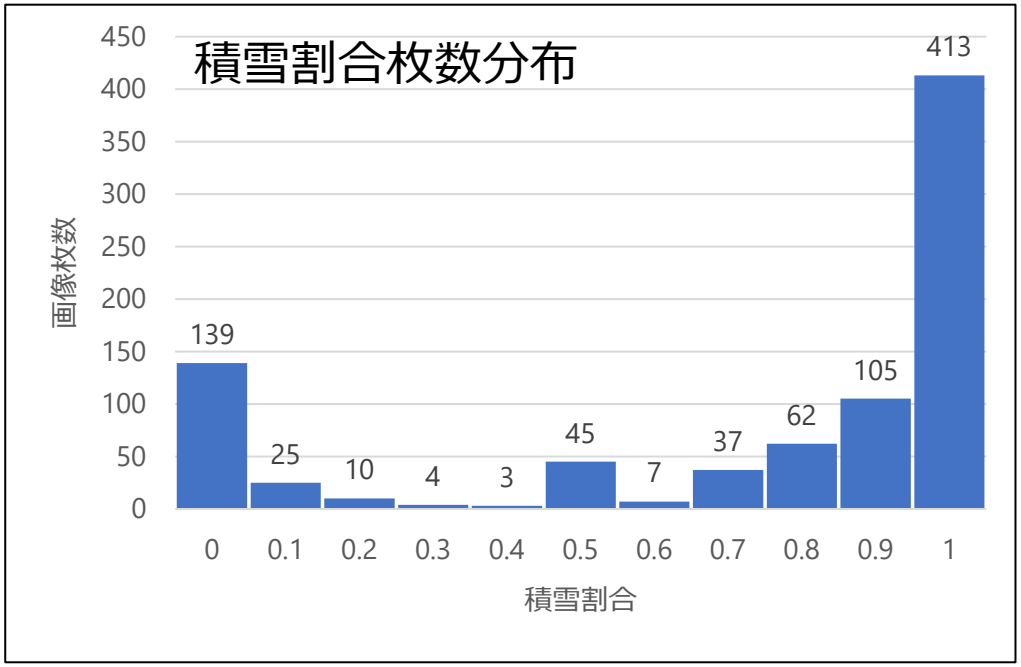
データセット作成

- 撮影場所
  - 北海道大学 大学院情報科学研究院棟 駐車場
    - 同棟 9階から路面を撮影
    - 撮影期間：2020年11月20日～ 2021年1月25日
- ラベリング方法
  - 積雪・非積雪・障害物の3分類
  - 合計850件のラベル付きデータを作成
    - アノテーションツール CVAT を利用

積雪
非積雪
障害物



データセット詳細



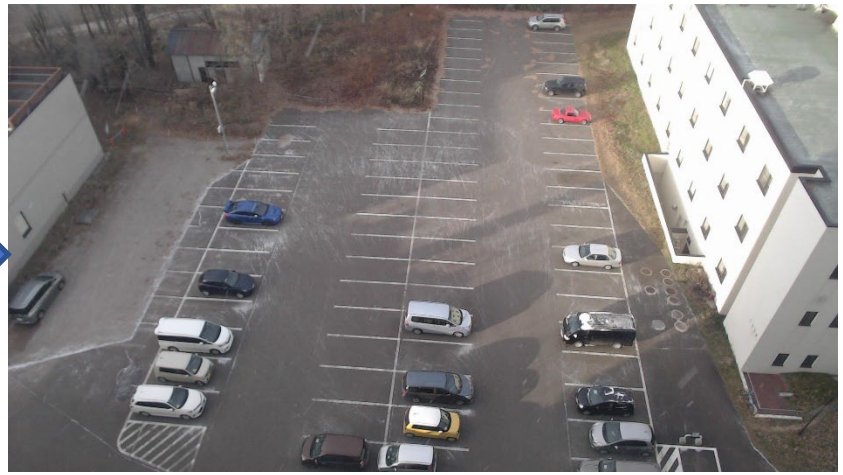
$$\text{積雪割合} = \frac{\text{積雪ラベルのピクセル数}}{\text{（積雪ラベル＋非積雪ラベル）のピクセル数}}$$

- 現状のデータセットは積雪割合分布が均等ではない
  - 積雪割合分布の両端を間違えない制御

撮影データの時間経過例

①2020年12月14日 午前8時10分 積雪割合0.0

②2020年12月14日 午前8時20分 積雪割合0.0



③2020年12月14日 午前8時30分 積雪割合1.0

④2020年12月14日 午前8時40分 積雪割合1.0

急激な積雪割合の変化があり、均等なデータ分布を得ることは難しい

ニューラルネットワークの拡張

- 画像処理における拡張
  - 画像認識タスクの分類
  - 画像認識モデル
    - 畳み込みニューラルネットワーク
  - 画像データの作成
    - アノテーションの付与
  - 画像データの増強
    - Data Augmentation (データの水増し手法)の導入

ニューラルネットワークの拡張

- 画像処理における拡張
  - 画像認識タスク
  - 画像認識モデル
    - 畳み込みニューラルネットワークの概要
  - 画像データ
    - アノテーションの付与
    - Data Augmentation (データの水増し手法)の導入

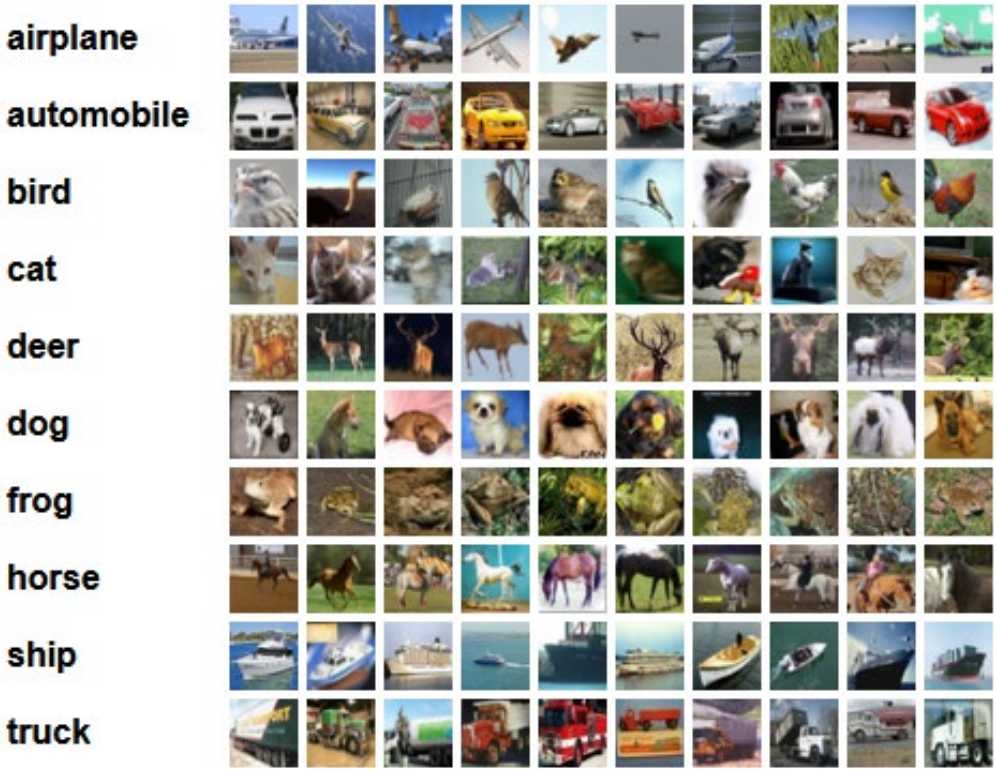
不均衡データの影響

- 不均衡な訓練データセット
  - 各クラスの画像枚数は揃えるのが望ましい
    - しかし、実際に収集してみると偏りがでることが多い
- データ数が偏った状態における学習
  - The CIFAR-10 dataset を使った実験で確認

- 1クラス(今回は車クラス)を減少
  - 各クラス4,000枚で学習
  - 車クラスだけ 1,000枚に減らして学習

サンプルデータセット

- The CIFAR-10 dataset
  - 主に画像認識を目的としたチュートリアル的なオープンデータセット
  - より複雑なCIFAR-100も提供
  - 画像の特徴
    - 画像サイズ：32x32ピクセル
    - 画像枚数：6万枚
    - カラー画像
    - 10クラス



- ラベル「**0**」： airplane（飛行機）
- ラベル「**1**」： automobile（自動車）
- ラベル「**2**」： bird（鳥）
- ラベル「**3**」： cat（猫）
- ラベル「**4**」： deer（鹿）
- ラベル「**5**」： dog（犬）
- ラベル「**6**」： frog（カエル）
- ラベル「**7**」： horse（馬）
- ラベル「**8**」： ship（船）
- ラベル「**9**」： truck（トラック）

CIFAR-10：
https://www.cs.toronto.edu/~kriz/cifar.html

学習結果の例

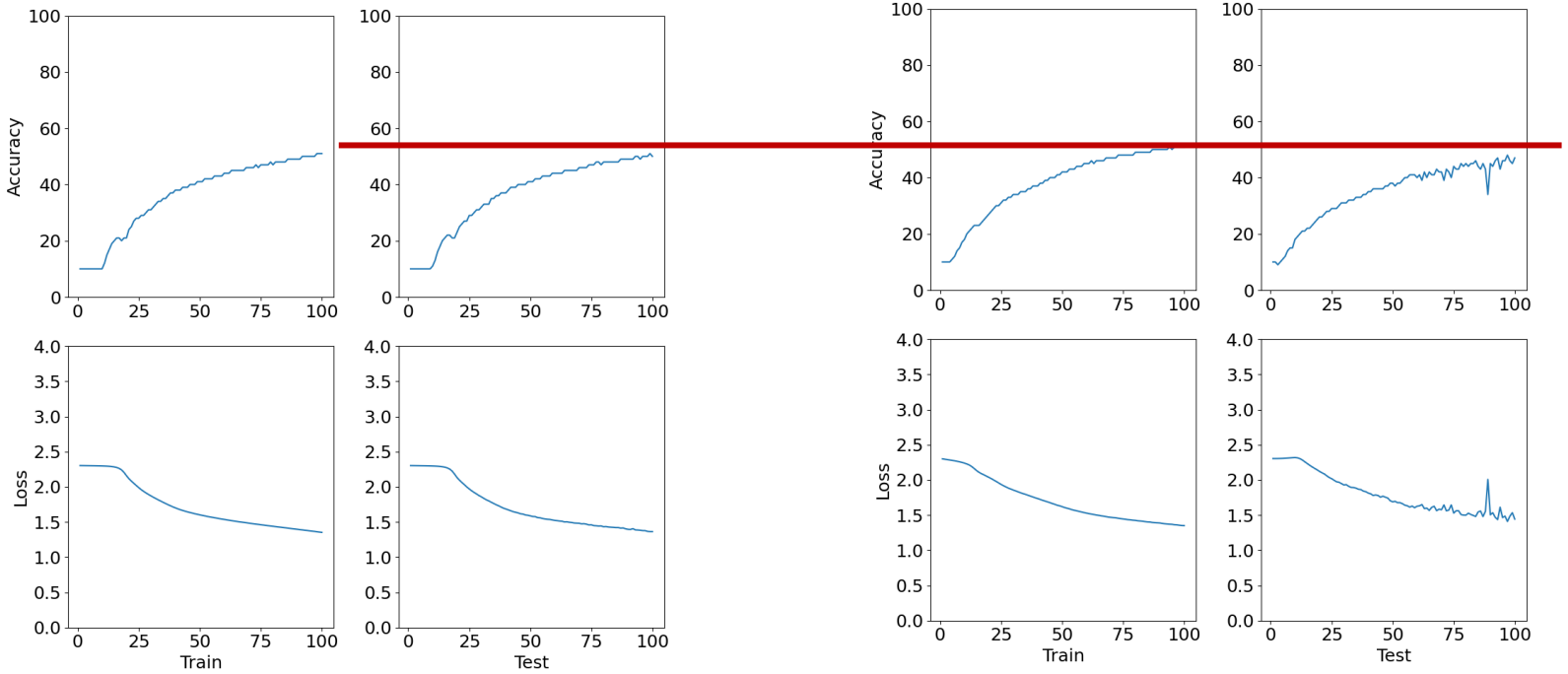
- 混同行列
  - 各クラス4000枚で学習：正解率51%

正解クラス

推定クラス

学習結果　不均衡データの影響

- 訓練データの Loss，正解率は変化なし
- 一方，テストデータに対する精度は低下



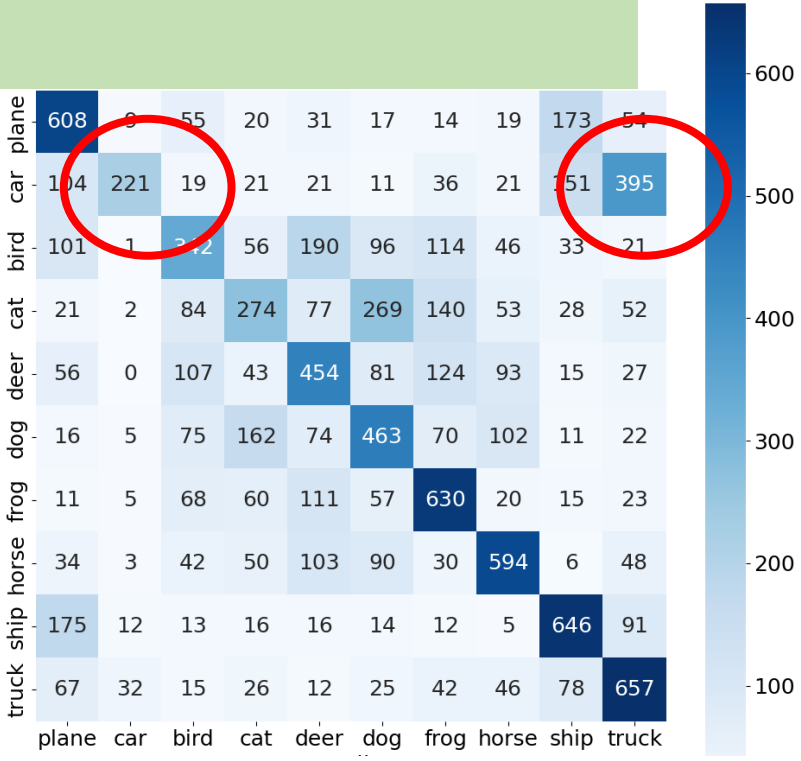
通常データで学習

不均衡データで学習

学習結果　不均衡データの影響

ポイント

- **精度は51%→48%に低下**
  - クラス“car”は”truck”と見た目が似てることから分類が上手くいっていないことがわかる

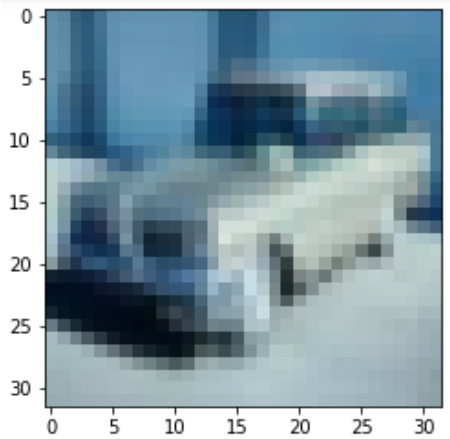


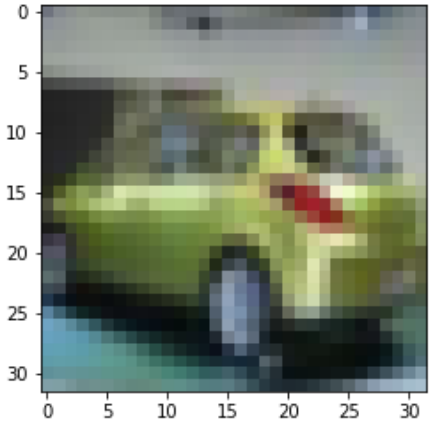
通常データで学習

不均衡データで学習

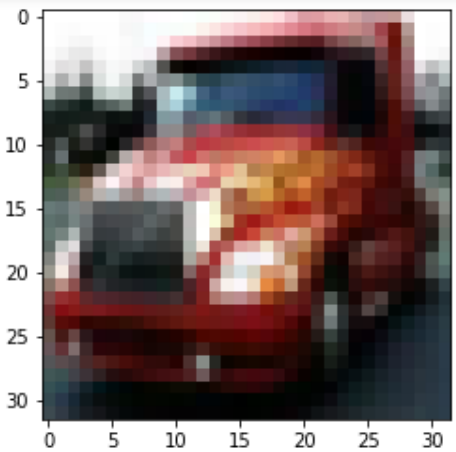
誤分類例

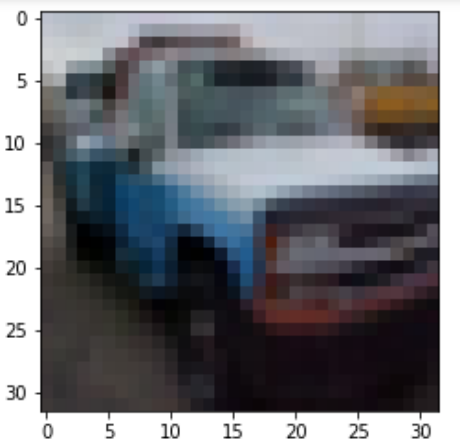
- ラベル：車・推定：トラック





- ラベル：トラック・推定：車





Data Augmentation の導入

- 機械学習において直面する一般的な課題：データ不足
  - 十分なデータを集めるのは非常に高コストで困難
    - プライバシーに関わるデータや発生頻度の低いデータは集めにくい
      - 例）医療診断画像

- Data Augmentation の導入
  - 1枚の画像に処理を加えて、データが水増しされているかのように、データ数を増加させる

  - データ数の増加
    - モデルの学習に必要なデータを確保
  - 過学習の防止

    ※1 過学習
      - 訓練データにモデルが適合しすぎて、テストデータに適合しなくなる
    - データの質を上げてデータを汎化させる
      - データ数を増やす ≠ データの質を上げる
        - データ数が多くても、偏りを持ったデータがある場合がある

Data Augmentation の例

- 例）左右反転
  - 反転させてもラベルが変わらないときに適用可能

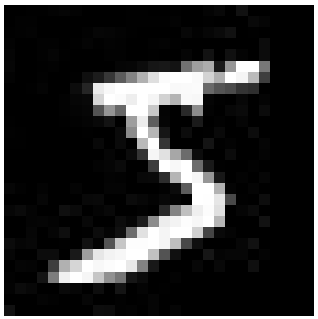
元画像

左右反転

反転画像

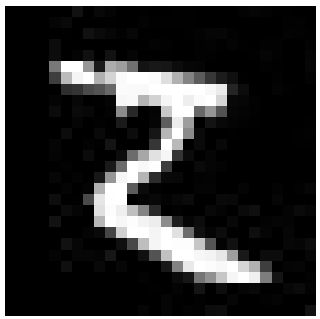
「猫」というラベルの意味には変化はない

  - 文字などの向きに意味がある軸に対する適用は不適当



mnistの”5”

左右反転



もう”5”ではない

Data Augmentation の例

- 猫は上下反転しても猫だが・・・

猫

上下反転

猫だが・・？

Data Augmentation の例

- 猫は上下反転しても猫だが・・・

猫

上下反転

猫だが・・？

ポイント

- テストデータにそのような猫が存在しうるかどうかが判断ポイント
  - 実際のタスクの場合は適用した際の状況から決まる
- いたずらにデータを複雑にしすぎるとニューラルネットが取り組んでいる問題の難易度が上がり，学習が難しくなる
- 今回の場合だと，写真にうつった猫が上下逆になっている状況は多くない→必要なさそう
  - もちろん，そのような状況が考えられるなら有効

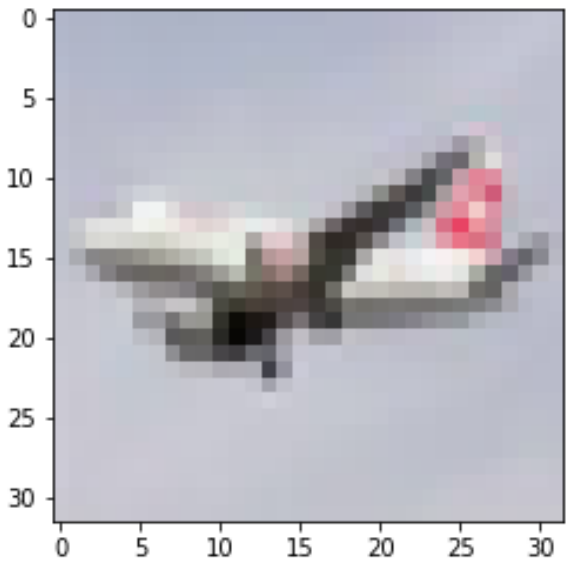
データオーグメンテーション手法1

- 各ピクセル位置の変更
  - Rotate
    - 画像の中心を原点に回転
  - ShearX,Y
    - X,Y軸でせん断
  - TranslateX,Y
    - X,Y軸で移動
  - Zoom
    - 拡大・縮小
  - verticalflip
    - 上下反転
  - horizontalflip
    - 左右反転
- 画像の変更
  - RandomCrop
    - トリミング
  - Cutout・Random Erasing
    - 部分マスクを付加

サンプルコード

- Rotate: 画像の中心を原点に回転

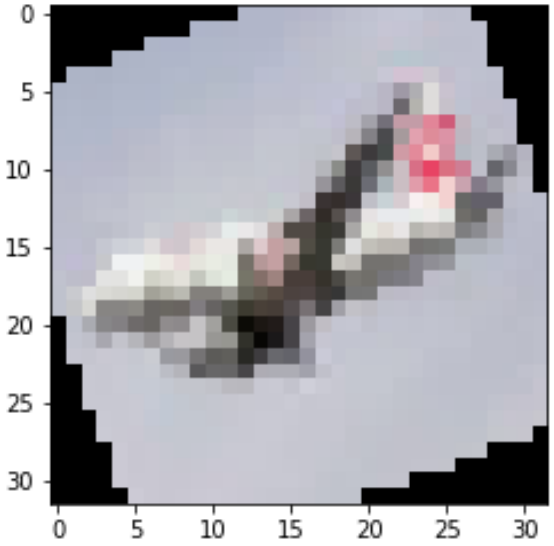
```
:
1  image = images[0]
2  imshow(image)
```



処理前

20度回転した場合

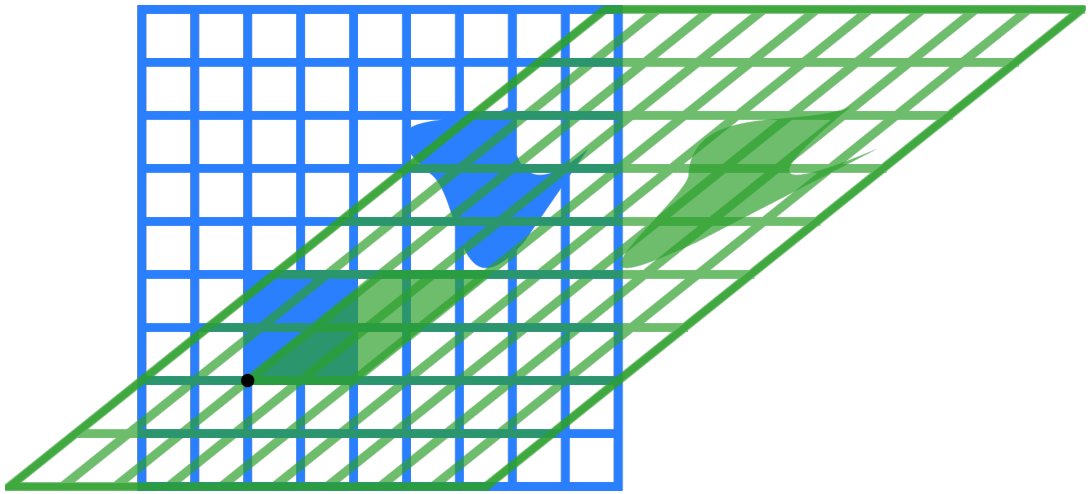
```
1  #Rotate
2  result = func.rotate(image, 20)
3  imshow(result)
```



処理後

せん断：Shear処理について

- 青(元画像)をshear処理することで緑(処理後)の形となる



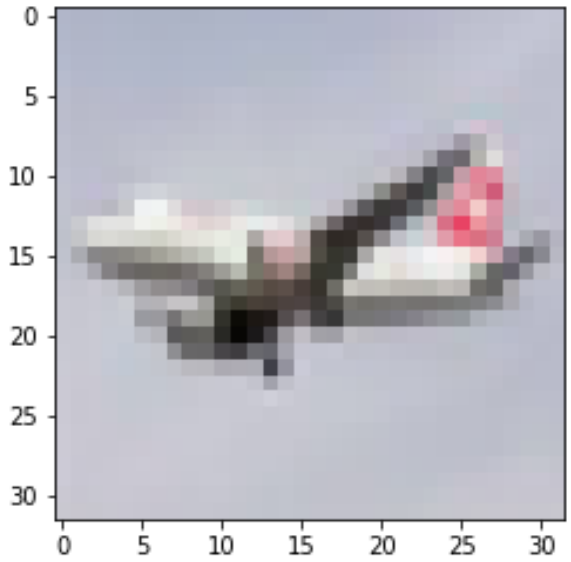
- Wikipedia*より

Wikipedia：
https://ja.wikipedia.org/wiki/%E3%81%9B%E3%82%93%E6%96%AD%E5%86%99%E5%83%8F

サンプルコード

- ShearX,Y：X,Y軸でせん断

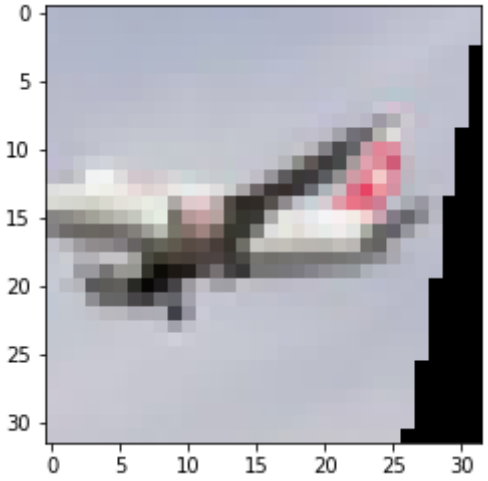
```
image = images[0]
imshow(image)
```



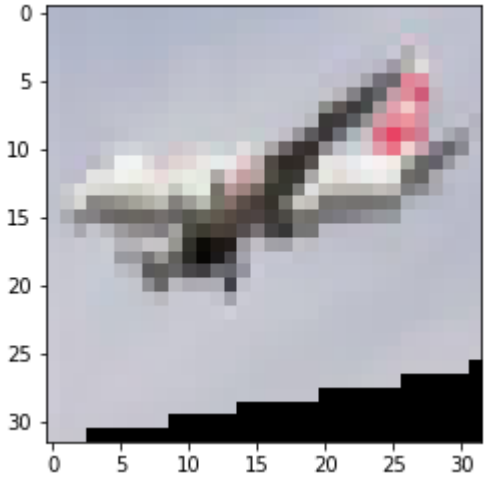
処理前

```
#ShearY
result = func.affine(
    image,
    angle=0.0,
    translate=[0, 0],
    scale=1.0,
    shear=[0, 10],
    center=[0, 0]
)
imshow(result)
```

X



Y



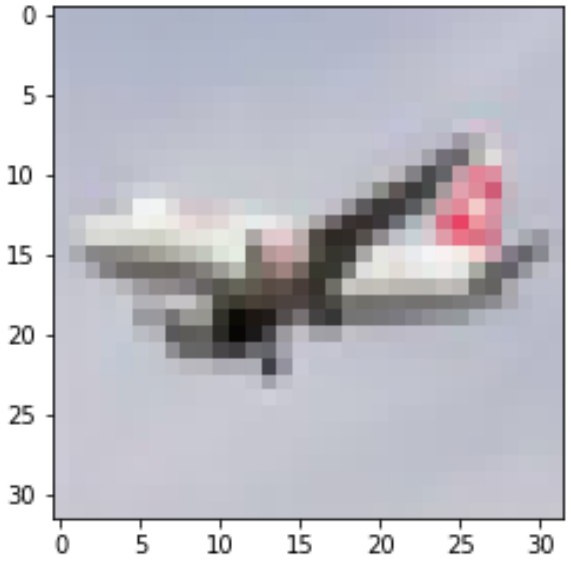
処理後

サンプルコード

- TranslateX,Y：X,Y軸で移動

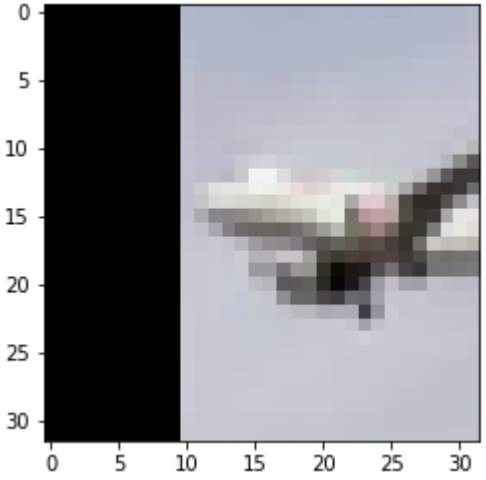
```
#TranslateY
result = func.affine(
    image,
    angle=0.0,
    translate=[0, 10],
    scale=1.0,
    shear=[0, 0],
    center=[0, 0]
)
imshow(result)
```

```
image = images[0]
imshow(image)
```

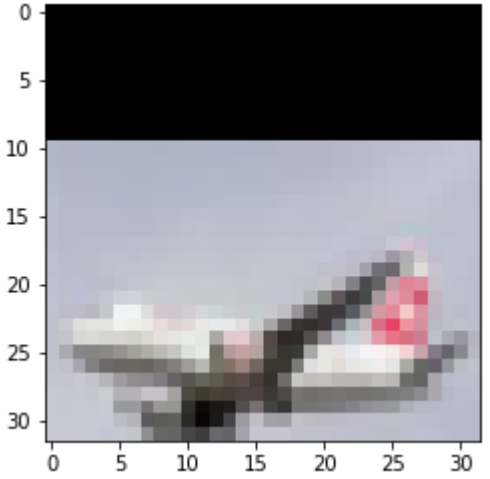


処理前

X



Y

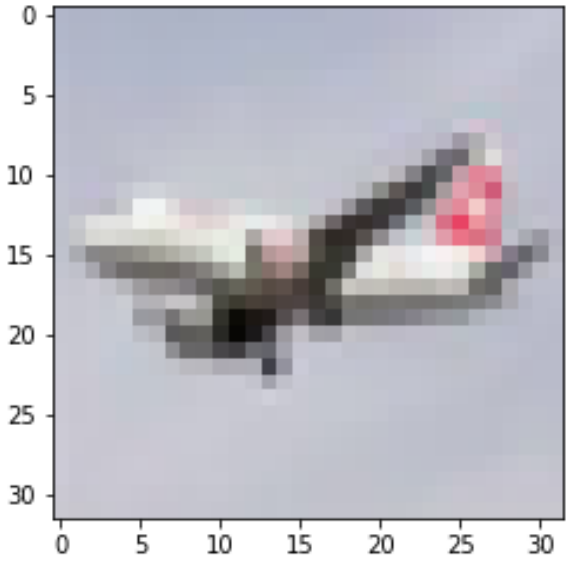


処理後

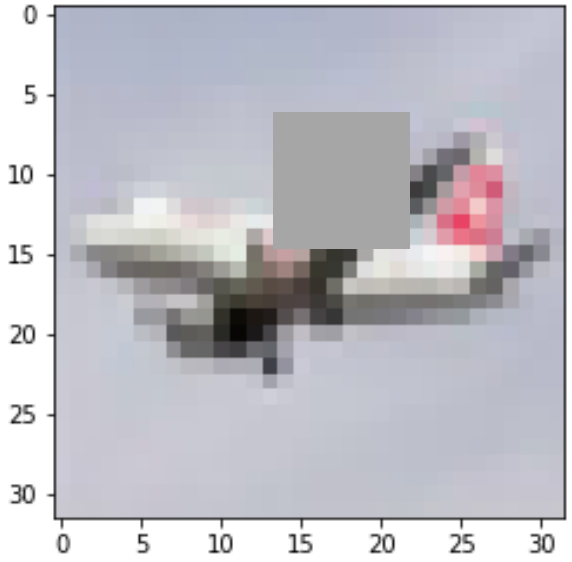
(サンプルコード)

- Cutout：画像中のランダムな位置を正方領域でマスクする
  - マスクはランダムなピクセル値、または平均的なピクセル値で構成される

```
image = images[0]
imshow(image)
```



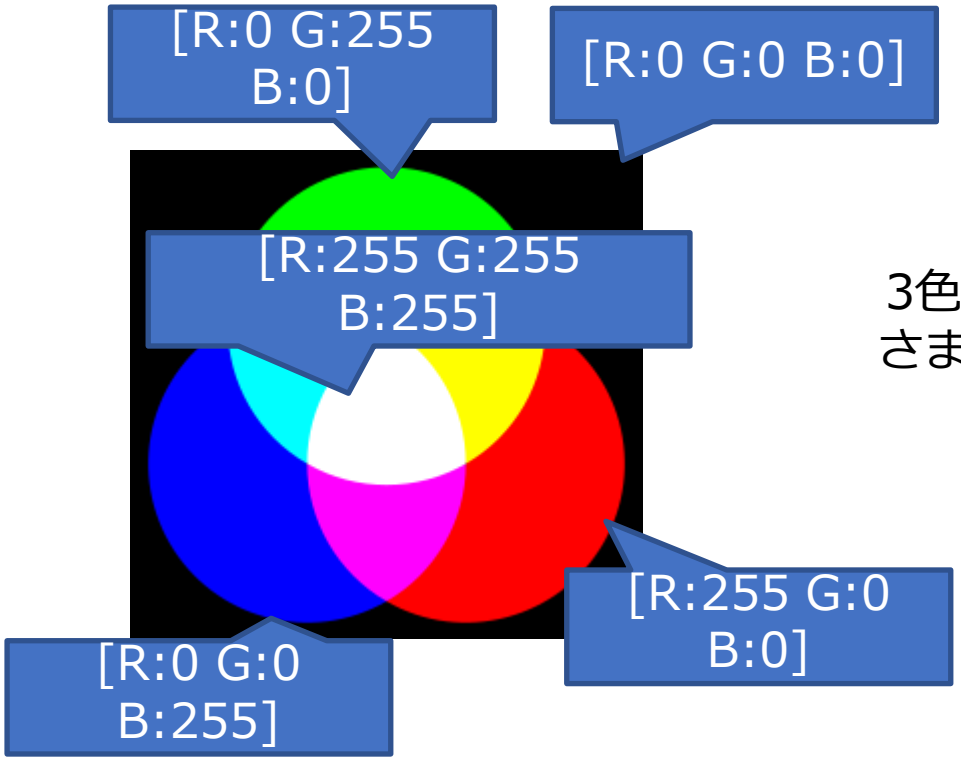
処理前



処理後

カラー画像の表示仕組み

- カラー画像の各ピクセル
  - RGBの3つの値で構成
    - 値の範囲は0～255の256段階



Wikipediaより

3色を重ね合わせて
さまざまな色を表現

Wikipedia RGB:https://ja.wikipedia.org/wiki/RGB

データオーグメンテーション手法2

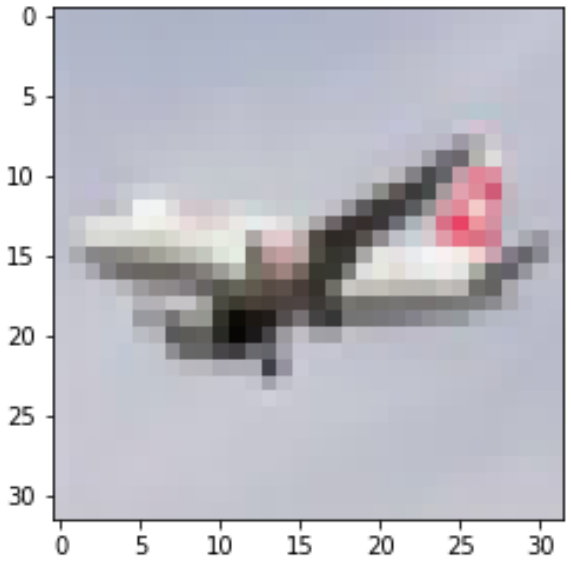
- 各ピクセル数値の変更
  - Invert
    - 255からRGBの値を引くことで、画像の色を反転
  - Autocontrast
    - 最も暗い画素を黒、明るい画素を白に再配置し、コントラストを最大化
  - Equalize
    - 出力に均一なグレースケール値の分布を作成
      - 入力に非線形マッピングを適用して画像のヒストグラムを均等化
  - Solarize
    - RGB/グレースケール画像を閾値以上の画素値をすべて反転
    - (Invert関数と同じ処理)
  - Color
    - 画像の彩度(各RGB値の差)を調整
  - Posterize
    - 各カラー チャネルのビット数を減らすことで色をはっきりとさせる
  - Sharpness
    - 輪郭を強調
  - Noise
    - ノイズを付加（ガウシアンノイズやインパルスノイズ）

サンプルコード

- Invert： 255からRGBの値を引くことで、画像の色を反転する。

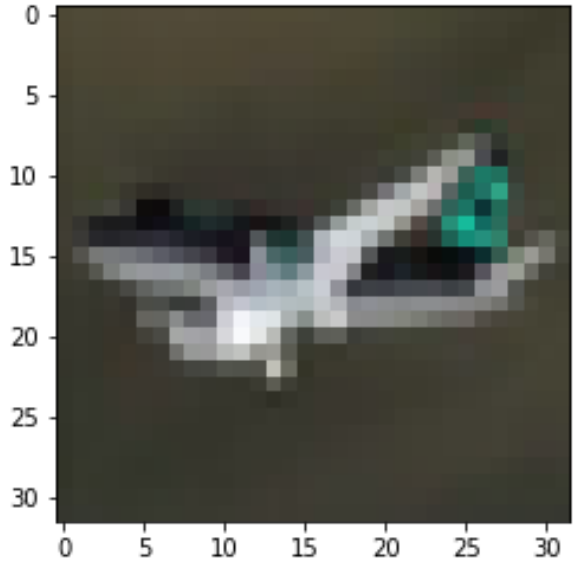
例)R=100 の場合、155となる

```
image = images[0]
imshow(image)
```



処理前

```
#Invert
result = func.invert(image)
imshow(result)
```

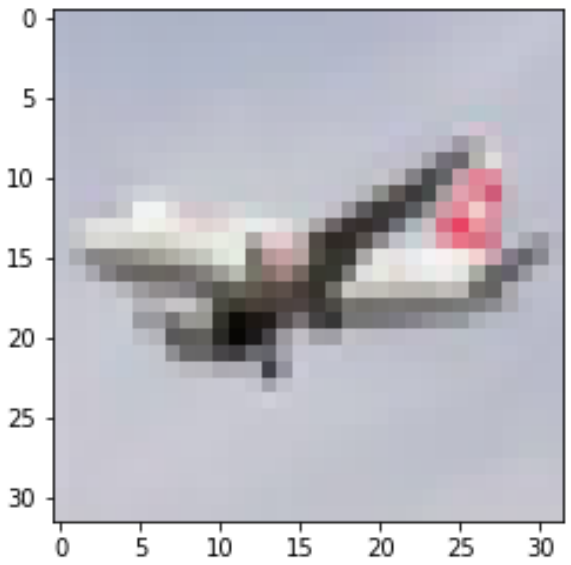


処理後

サンプルコード

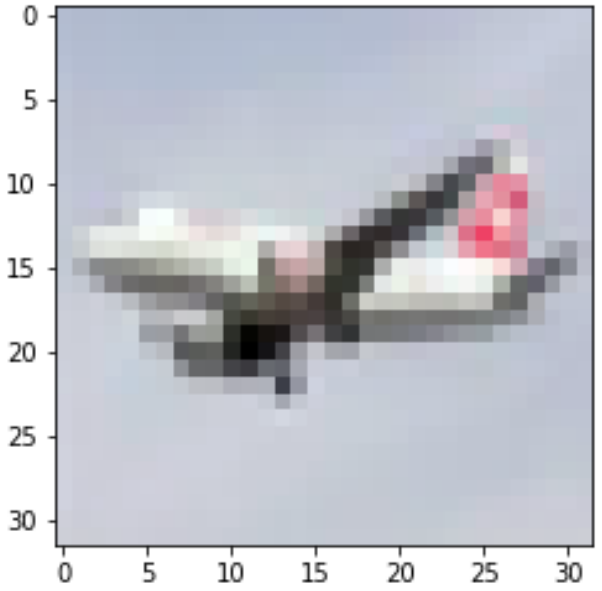
- Autocontrast：
  最も暗い画素を黒、明るい画素を白に再配置し、コントラストを最大化

```
image = images[0]
imshow(image)
```



処理前

```
#AutoContrast
result = func.autocontrast(image)
imshow(result)
```

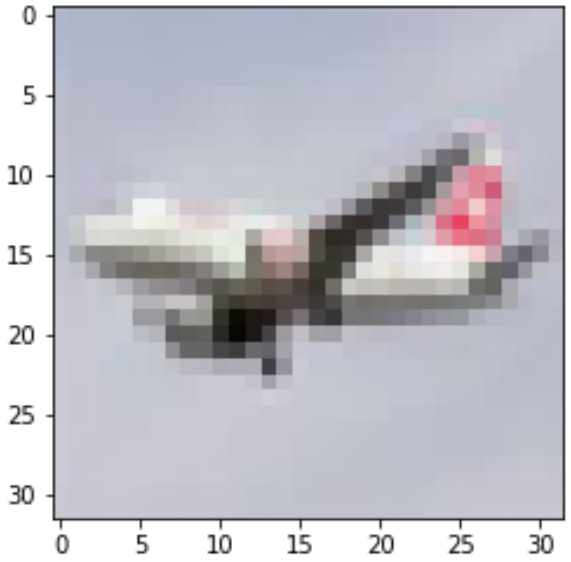


処理後

サンプルコード

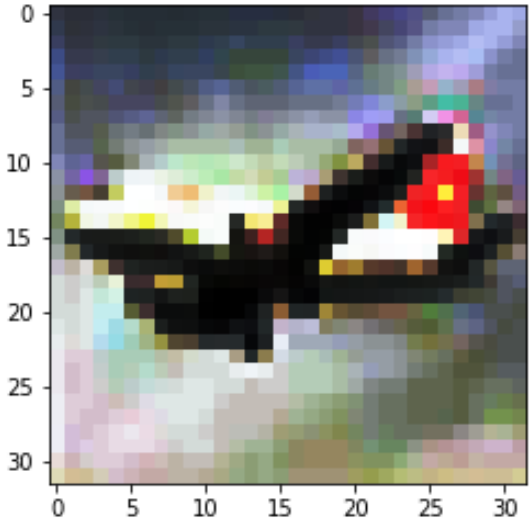
- Equalize：出力に均一なグレースケール値の分布を作成するために、入力に非線形マッピングを適用して画像のヒストグラムを均等化する

```
image = images[0]
imshow(image)
```



処理前

```
#Equalize
tmp = image*255
result = func.equalize(tmp.to(torch.uint8))
imshow(result)
```

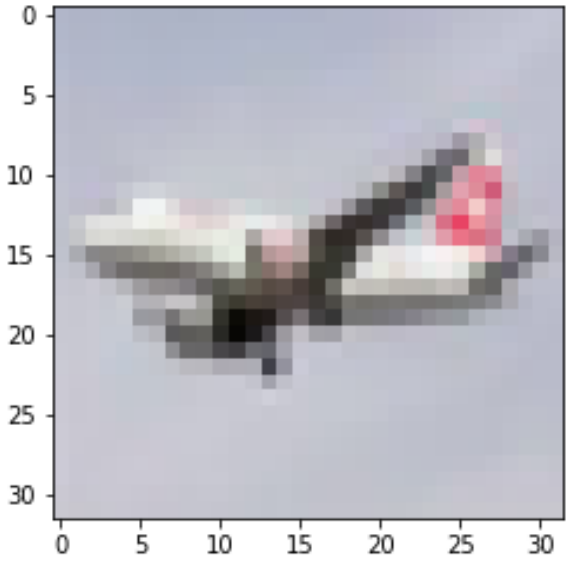


処理後

サンプルコード

- Solarize: RGB/グレースケール画像を、閾値以上の画素値をすべて反転(Invert関数と同じ処理)

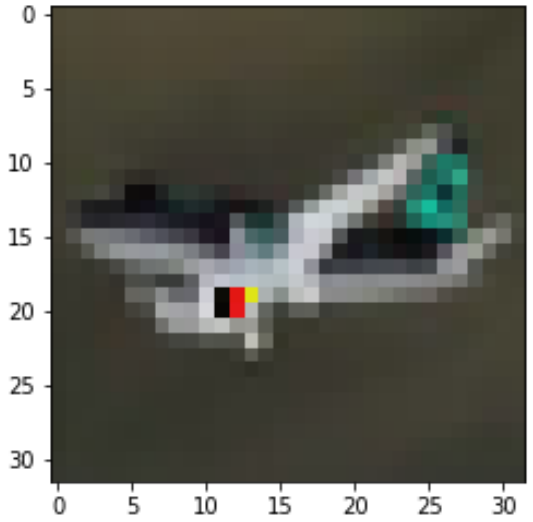
```
image = images[0]
imshow(image)
```



処理前

閾値は-1~1の間で決定

```
#Solarize
result = func.solarize(image, 0.1)
imshow(result)
```

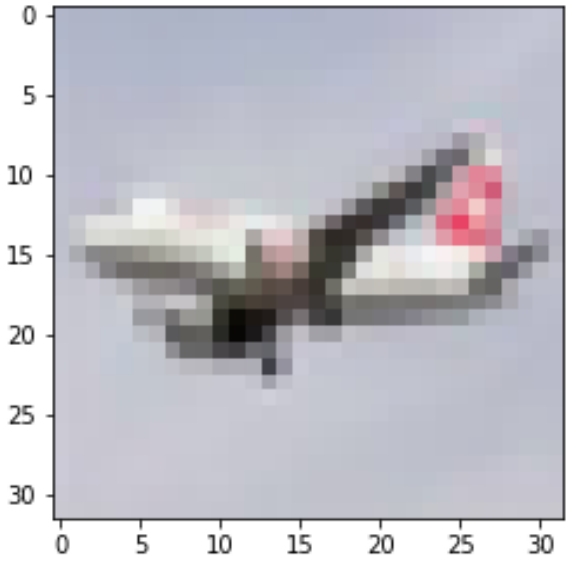


処理後

- Color:画像の彩度(各RGB値の差)を調整

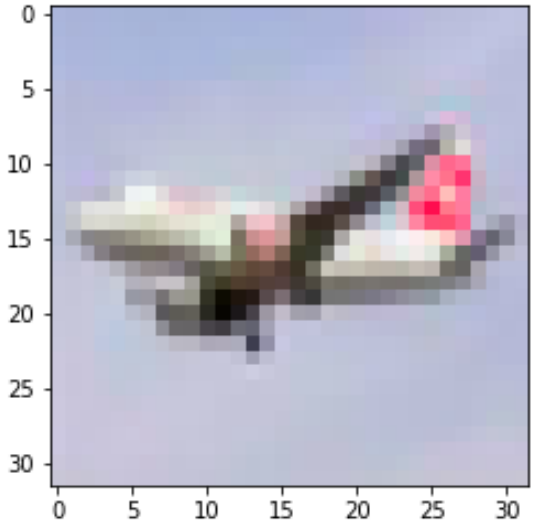
この値は0は差がないので白黒となる。1は元画像を表し、2は彩度が2倍となることを意味します。小数点を使うことで調整可能です

```
image = images[0]
imshow(image)
```



処理前

```
#Color
result = func.adjust_saturation(image, 2.0)
imshow(result)
```

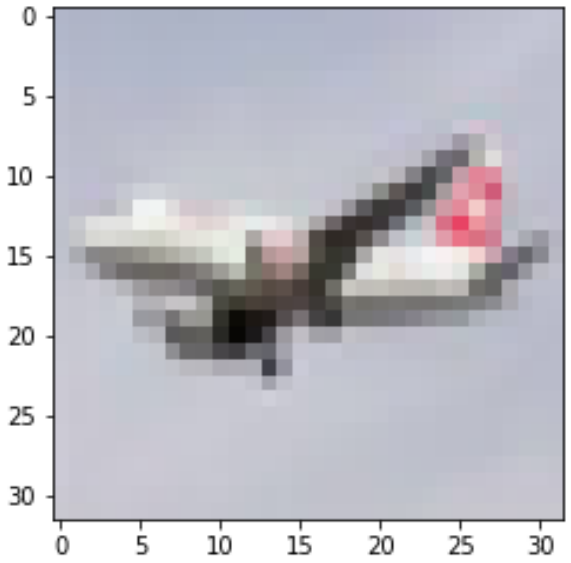


処理後

サンプルコード

- Posterize：各カラー チャネルのビット数を減らすことで色をはっきりとさせている

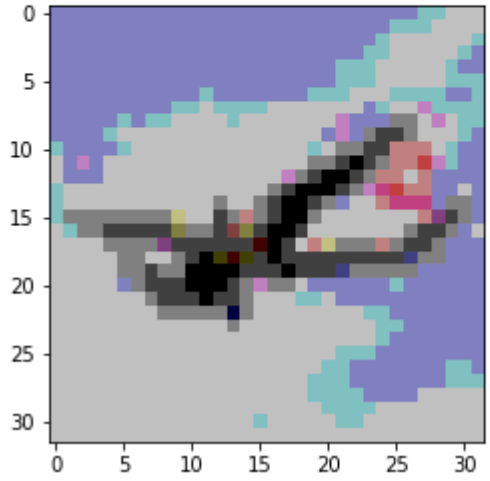
```
image = images[0]
imshow(image)
```



処理前

```
#Posterize
tmp = image*255

result = func.posterize(tmp.to(torch.uint8), 2)
imshow(result)
```

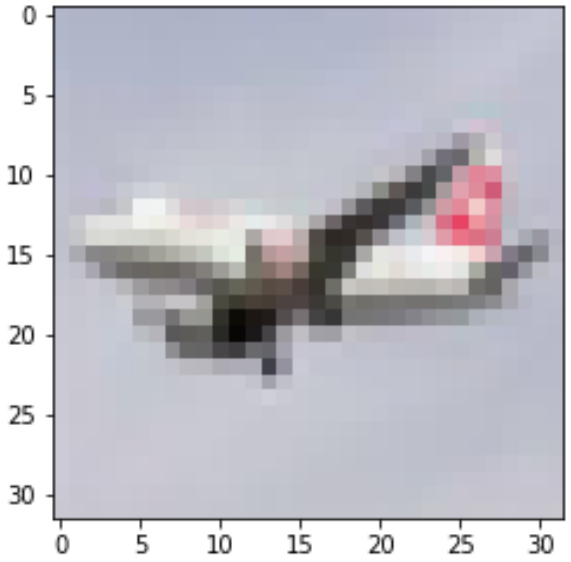


処理後

サンプルコード

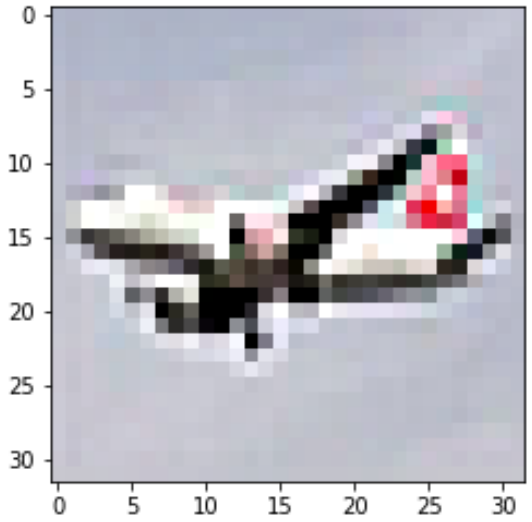
- Sharpness：輪郭を強調

```
image = images[0]
imshow(image)
```



処理前

```
# sharpness
result = func.adjust_sharpness(image, 4)
imshow(result)
```



処理後