# aa-8. 人工知能とコンピュータビジョン

（人工知能）

金子邦彦

# 「人工知能」第８回の内容

Database Lab.

**画像は数値の集まり。その数値からディープラーニングが特徴を学び、画像を理解する**

## ① 画像の数値表現

**画像＝格子状の画素**

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | | | 128 | 0 |
| 0 | 0 | 128 | 255 | 0 |
| 0 | | | 255 | 0 |
| 0 | 0 | 0 | 0 | 0 |

濃淡：0〜255

R 0〜255

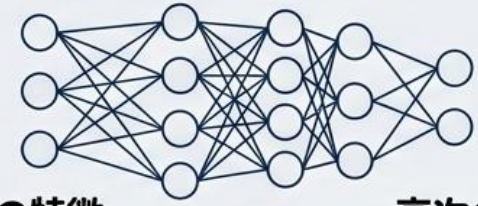
G 0〜255

B 0〜255

カラー＝RGBの3数値

## ② ディープラーニング

**多層で特徴を抽出**



低次の特徴
エッジ・色

高次の特徴
物体

## ③ CNNの構造

**CNN（畳み込みニューラルネットワーク）**

畳み込み層 → プーリング層 → 全結合層

局所的な特徴を抽出　情報を圧縮・縮小　判断・出力

局所的特徴を効率的に捉える

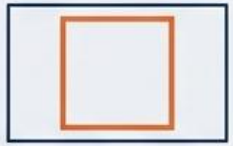
## ④ 画像理解の3種類と応用

**画像理解の3つの方法**

**分類**


画像全体が何か

**物体検出**



位置を四角で囲む

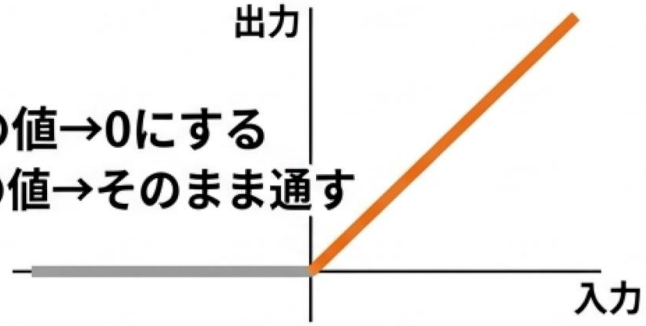
**セグメンテーション**

画素ごとに領域分け

応用：顔処理・自動運転 など

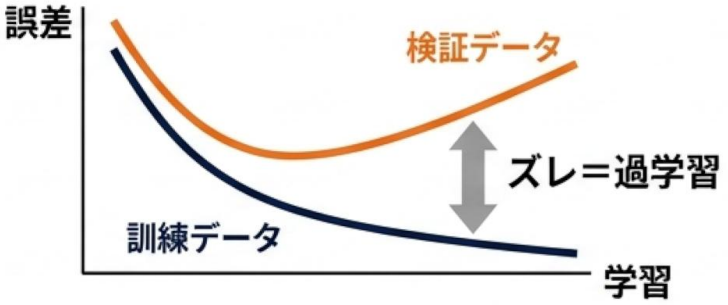
# ディープラーニングによるコンピュータビジョンを学ぶときに出てくる4つの基本用語

## 活性化関数(ReLU)




非線形性を生み複雑なパターンを表現／勾配消失も起きにくい

## 過学習




訓練データは誤差小、検証データは誤差大の状態／
畳み込み層・プーリングは過学習を抑える工夫

## 勾配と勾配消失


勾配＝損失を小さくする重みの更新方向／
層が深いと入力層側で勾配が消え浅い層が学習できない

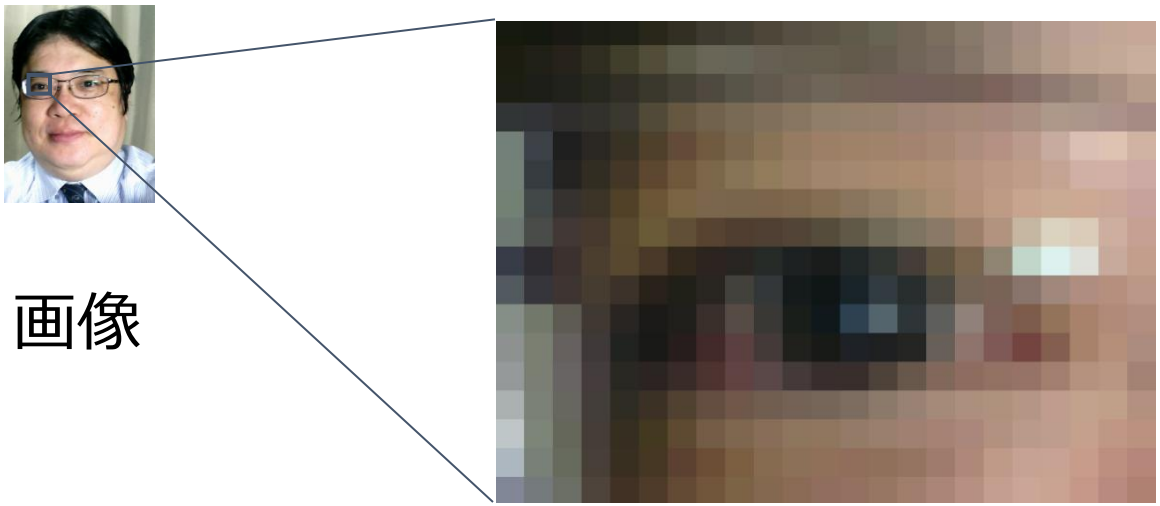
## 特徴マップ


カーネルを画像上で動かし、各位置での反応の強さを並べたもの

# 8-1. 画像、ディープラーニング

# デジタル画像の構造：画像と画素

- **画像**は**格子状に並んだ画素**（画像を構成する最小の要素）から構成される.
- 画像を拡大すると，個々の画素が確認できる.



画像

それぞれの格子が画素

# 画像の種類

カラー画像

**輝度（明るさの度合い）**と**色**の情報を持つ画像

濃淡画像

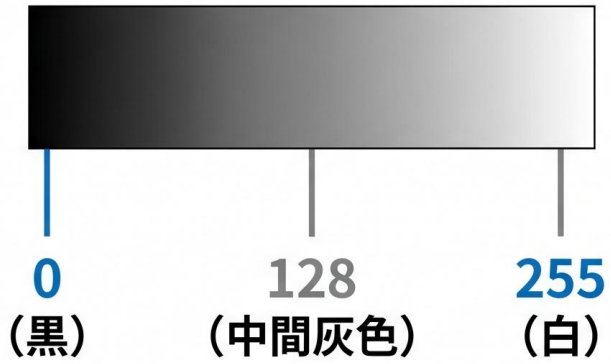
**輝度**のみの情報を持つ画像

# 濃淡画像の基礎

## 1. 画像の輝度を数値化する（4段階の例）

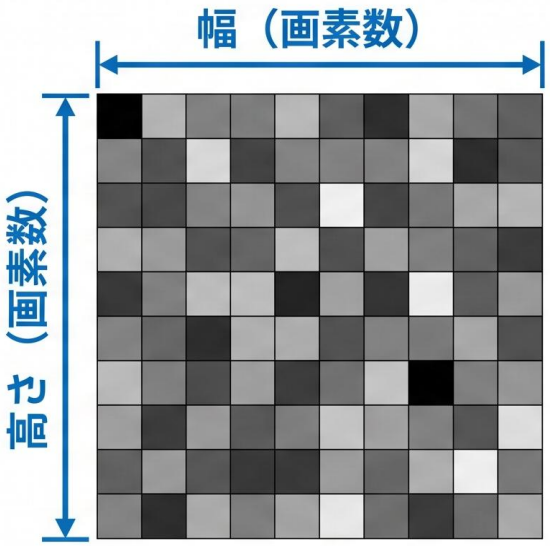
| 濃淡見本 | 対応する数値（コード） |
|---|---|
| ■ | 0 |
| ■ | 1 |
| ■ | 2 |
| □ | 3 |

## 2. 一般的な256階調（0～255）の表現




デジタル画像では通常、濃淡を0～255の数値で表す。

## 3. 画像のサイズ（画素数）




画像は『画素（ピクセル）』の集まりであり、サイズは幅と高さの画素数で表される。

# カラー画像の成分

カラー画像の成分の表し方は2種類ある．

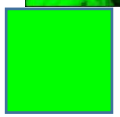
- **R（赤）成分，G（緑）成分，B（青）成分**で考える場合

**各成分**は，画素ごとに**1つの数値**を持つ．すべて合わせて，画素ごとに3つの数値となる．





R（赤）成分


G（緑）成分


B（青）成分

- **輝度成分，色成分**で考える場合

輝度成分は明るさの情報，色成分は色の情報を表す．

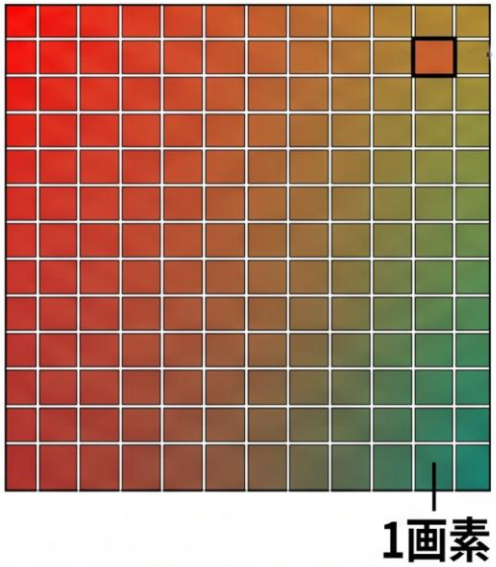



輝度成分

色成分

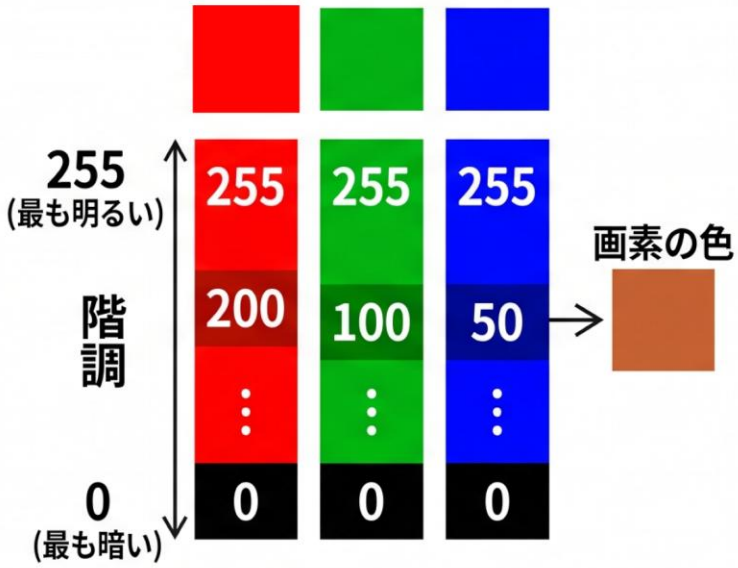
# カラー画像の基礎

## 画像全体の構成




画像は格子状に並んだ
画素（ピクセル）の集まり

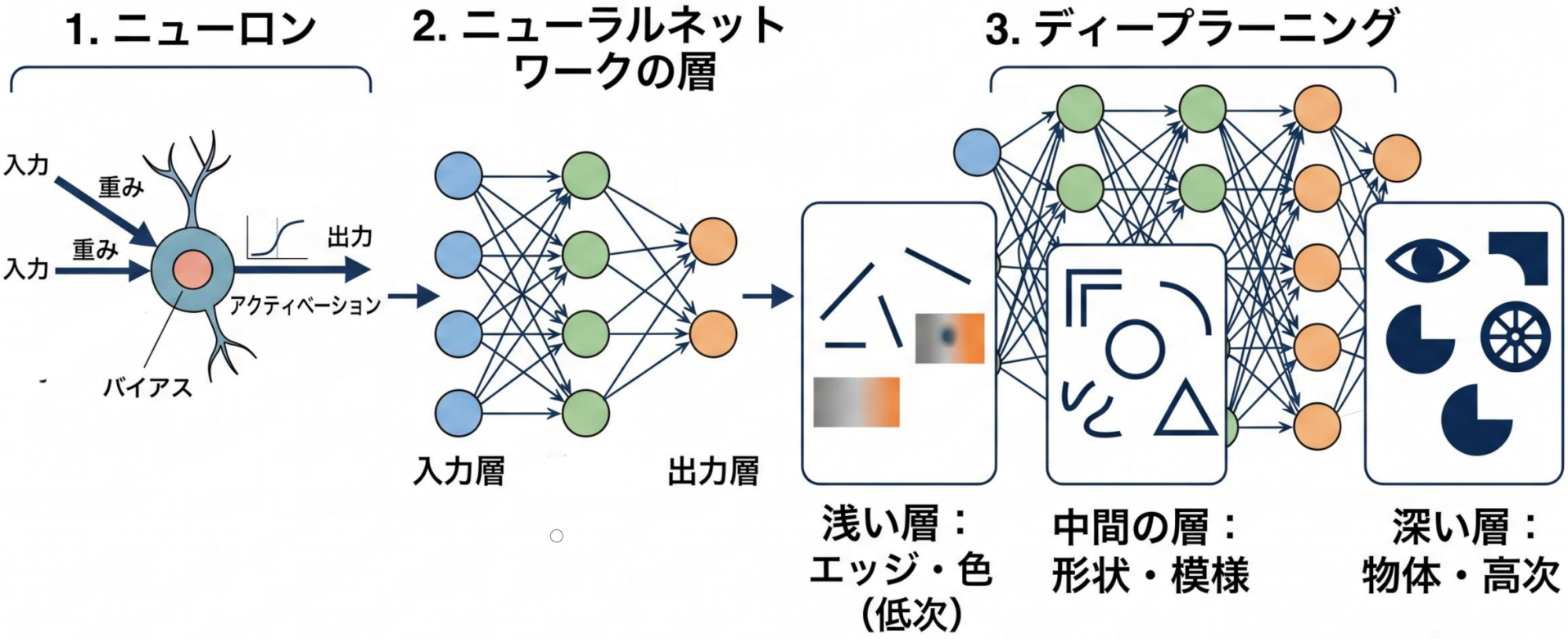
## 画素の表現
（R, G, B）




光の三原色（R, G, B）で表す
各成分 0～255 の整数（256 階調）

(0, 0, 0) 黒
(255, 255, 255) 白
(200, 100, 50)
(255, 0, 0) 赤
(0, 255, 0) 緑
(0, 0, 255) 青

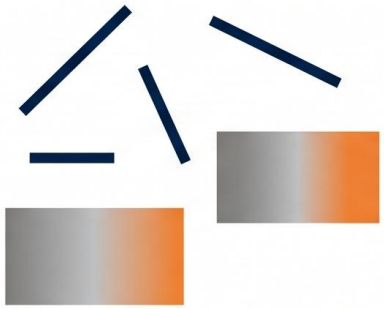
# ニューラルネットワークとディープラーニング



# ディープラーニング

## ニューラルネットワークは、層が深くなるほど単純なパターンを組み合わせて複雑なパターンをとらえる

### 浅い層



エッジ・色の変化など単純で局所的なパターン（低次の特徴）

組み合わせる

### 中間の層

エッジを組み合わせた形状・模様のパターン

さらに組み合わせる

### 深い層

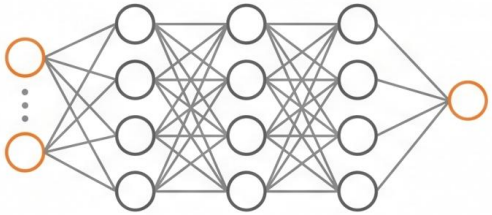
形状を組み合わせた物体の部分など複雑なパターン（高次の特徴）

単純・局所的 ——層が深くなる——→ 複雑・全体的

# ディープラーニングの特色

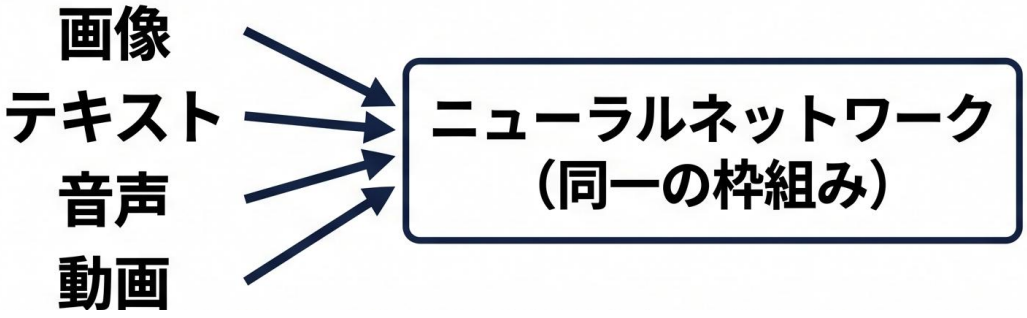
## 1「複雑なパターンを抽出する能力」

入力 → 低次の特徴 → 中間の特徴 → 高次の特徴



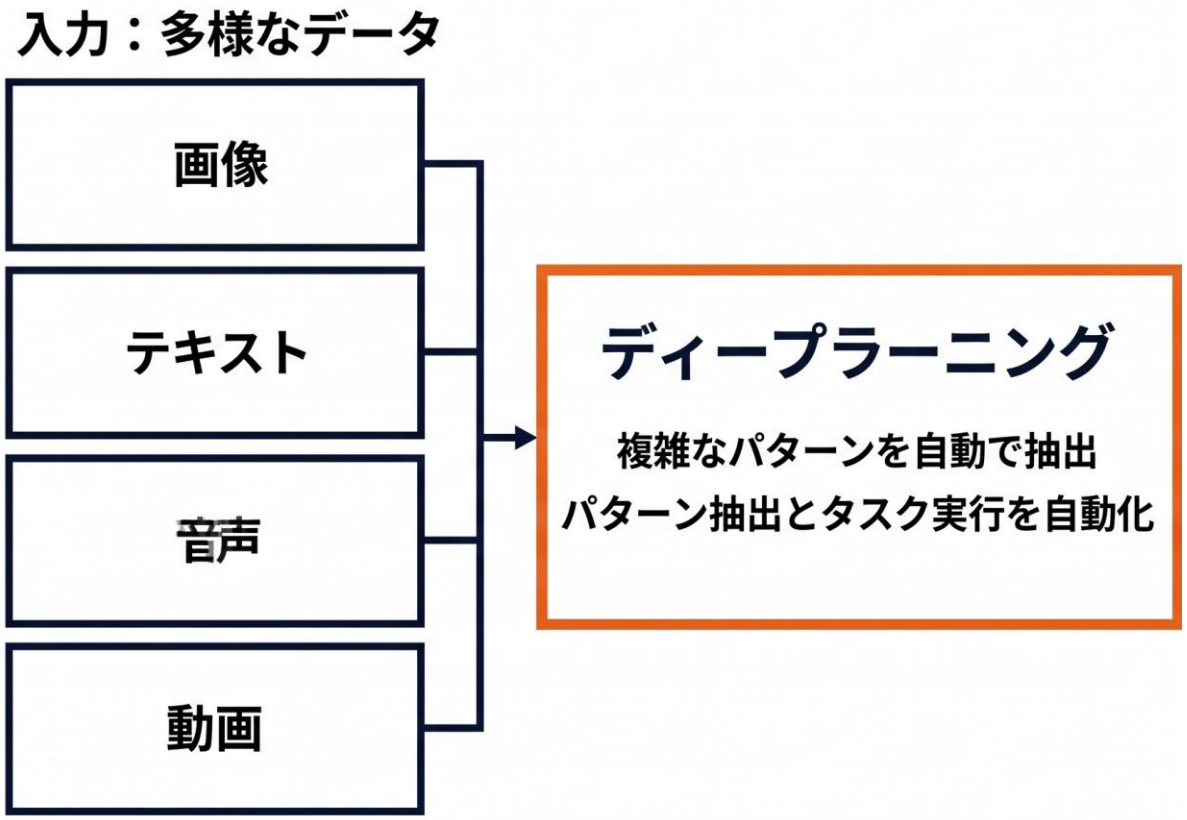
多層構造により低次から高次へ階層的に特徴を獲得する

## 2「多様なデータ形式への対応」




| | | |
|---|---|---|
| 画像 | → | CNN |
| 系列データ（テキスト・音声） | → | RNN / LSTM |
| 全般（近年の主流） | → | Transformer |

# ディープラーニングの応用分野




## 応用分野

### 画像認識
物体検出、顔認証、医用画像診断、自動運転の認識系。

### 自然言語処理
機械翻訳、文章生成、要約、対話システム（大規模言語モデル）。

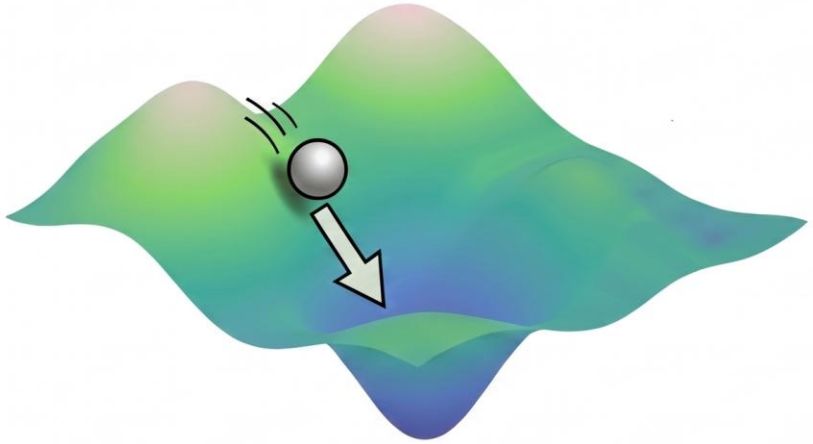
### 音声認識
音声入力、文字起こし、音声合成（TTS）。

### 横断的応用
マルチモーダルモデル（画像＋テキストなど複数データを統合）。

# ディープラーニングの問題：勾配消失問題

ニューラルネットワークでは、**層を深く**すると、学習時に、**誤差を伝える勾配が浅い層**に届きにくくなる『**勾配消失問題**』があった

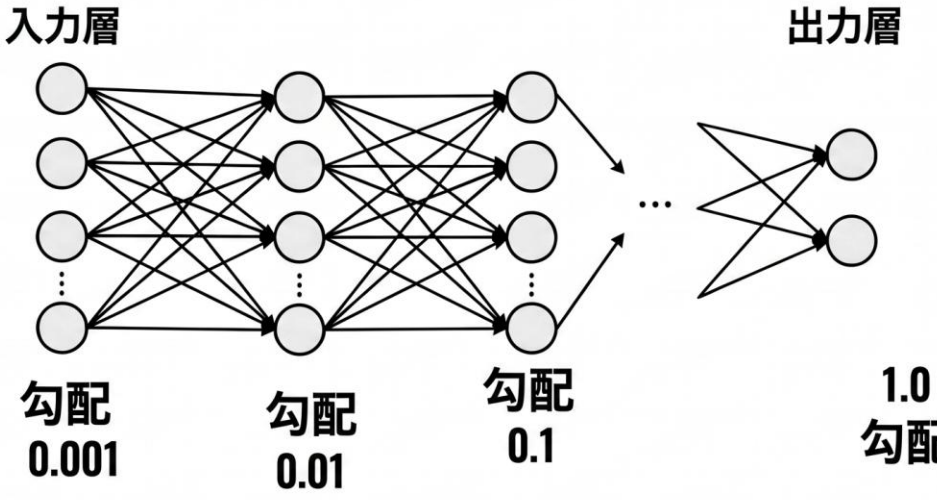
## 勾配



学習の方向
を表す

損失関数を最小化する方向へ重みを更新

## 勾配消失問題

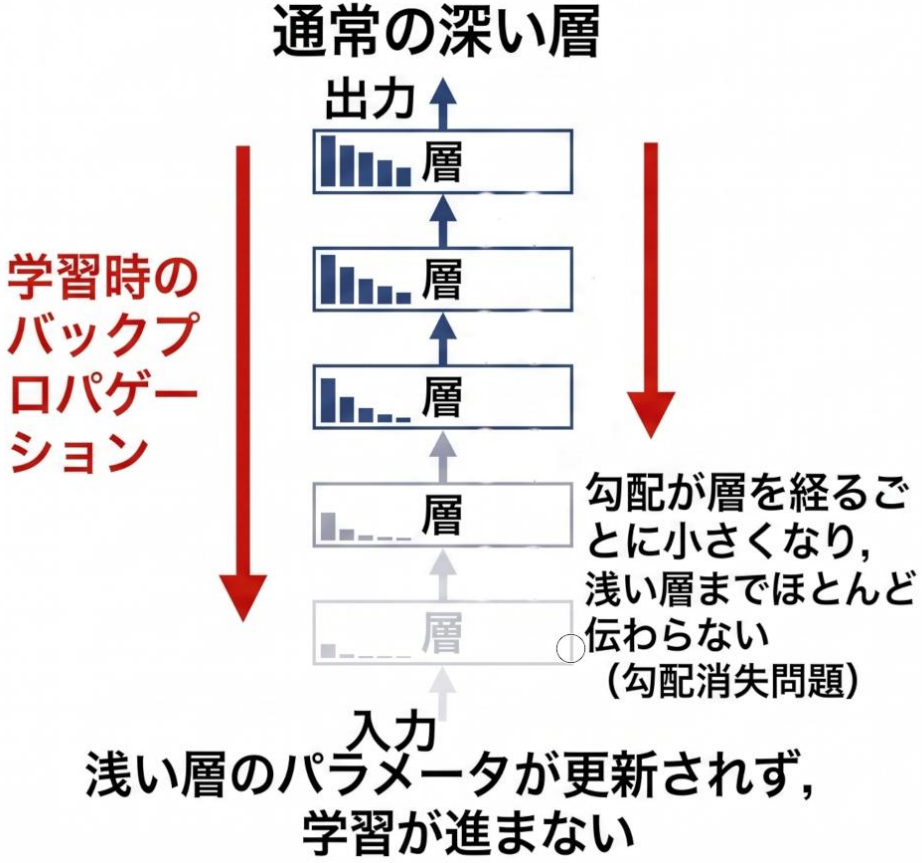


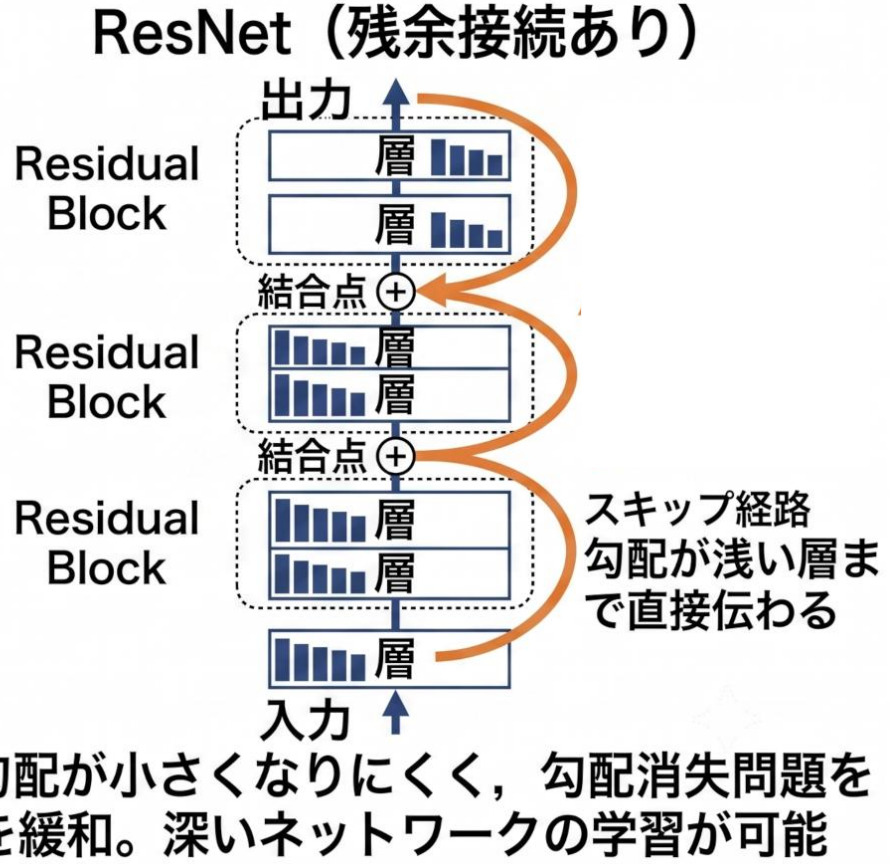

勾配が消失
(ほぼゼロ)

結果：浅い層の重みが更新されなくなる

# ディープラーニングを可能にする技術

**ResNet：層をスキップする経路（残余接続）を加え，勾配消失を緩和して深いネットワークの学習を可能にする**



# 8-2. コンピュータビジョン

# コンピュータビジョン

**コンピュータビジョン ＝ コンピュータが『視覚』を持つこと**

**入力**

**実世界の様子**
**（カメラ映像・画像）**

コンピュータが理解

**理解**

- 何があるか（認識）
- 何が起きているか（状況把握）
- 何が起きそうか（予測）

理解を活用

**活用**

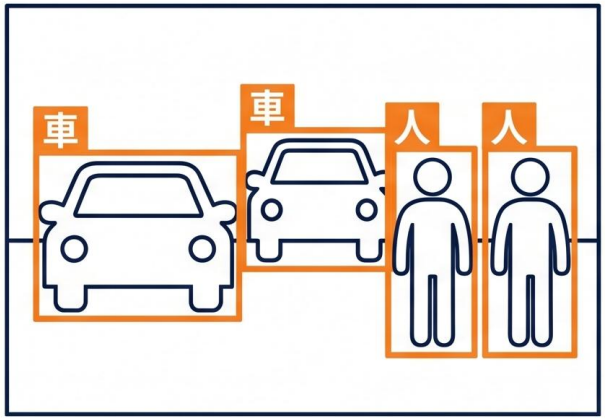
現実世界での判断・行動に役立てる

例：自動運転、検査の自動化

# コンピュータビジョンの主要タスク

種類、画素、個体の特定の違い

## 物体検出




物体の位置と大きさを四角形で特定する

## セマンティック・セグメンテーション


画素ごとに種類（クラス）を割り当てる。

## インスタンス・セグメンテーション


種類に加えて、同じ種類でも一つ一つ（個体）を区別する

# セマンティックセグメンテーションの例

# インスタンスセグメンテーションの例

# 8-3. 畳み込みニューラルネットワーク（CNN）

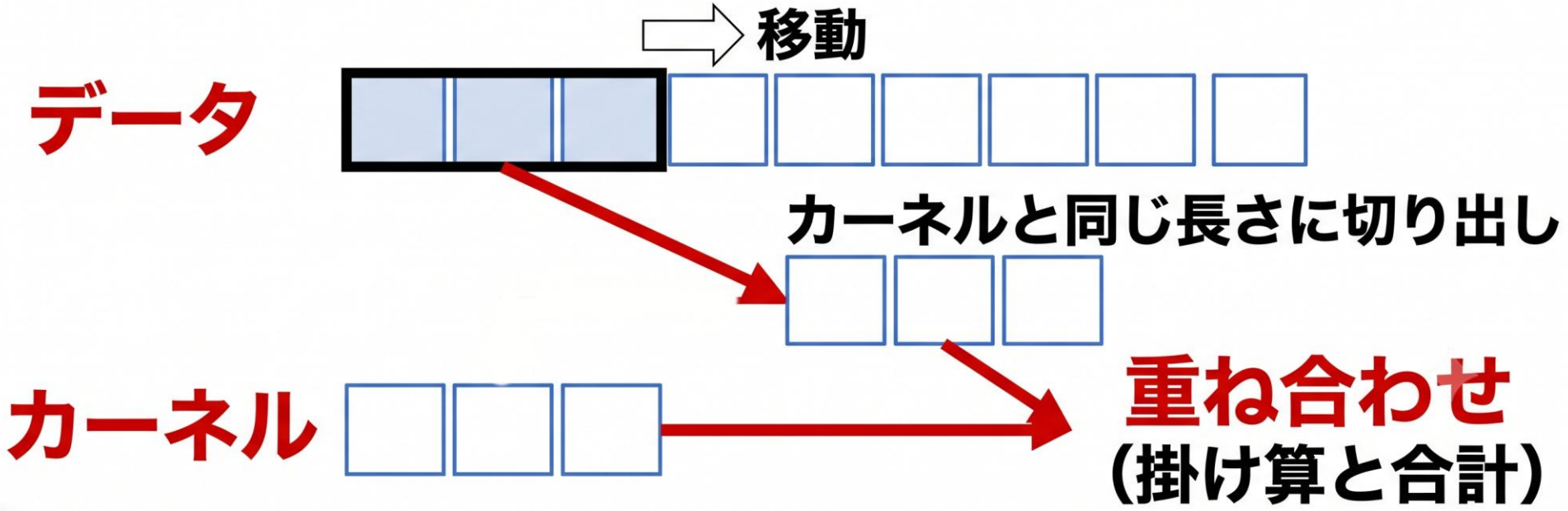
# 畳み込みニューラルネットワーク（CNN）

**CNNは画像の理解・分析に特化したディープラーニングで、3種類の層を組み合わせて構築する**


これら3種類の層（畳み込み層・プーリング層・全結合層）を組み合わせることで、さまざまなCNNを構築できる

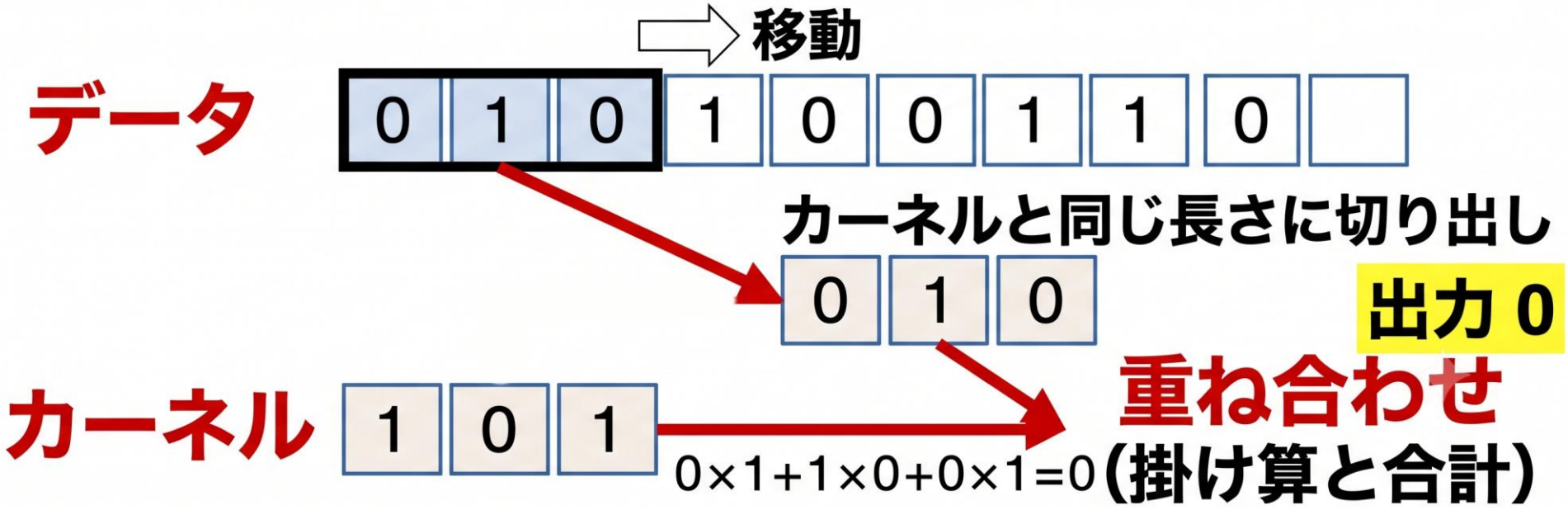
# 畳み込みの仕組み　数字の並びの例

① データ上を移動しながら処理
② データをカーネルと同じ長さに切り出し
③ カーネルとの重ね合わせを実行（掛け算と合計）



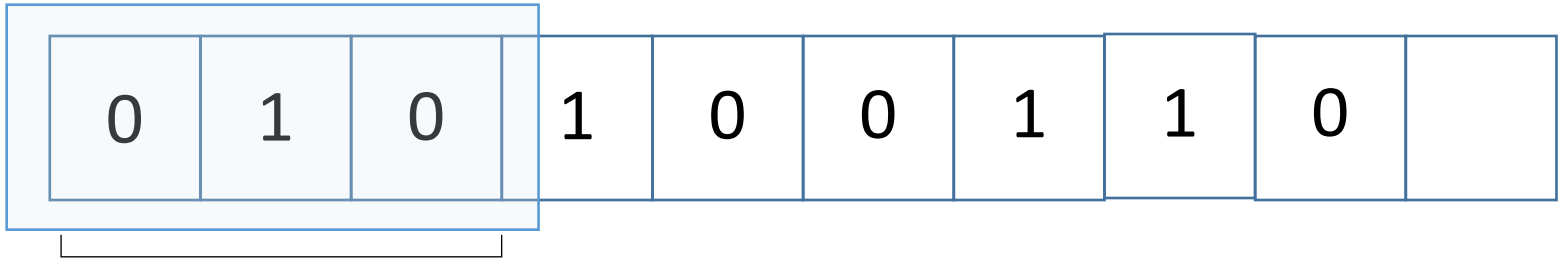
# 畳み込みの仕組み　数字の並びの例

① データ上を移動しながら処理
② データをカーネルと同じ長さに切り出し
③ カーネルとの重ね合わせを実行（掛け算と合計）

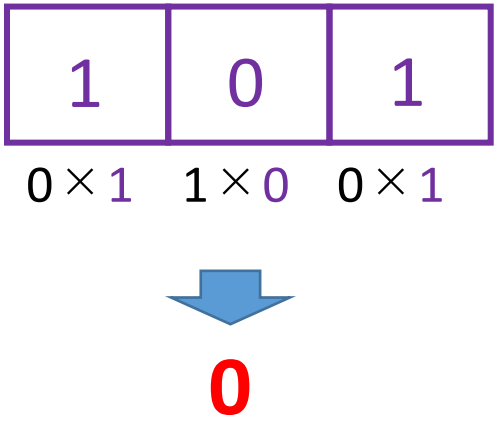


# 畳み込みの仕組み　数字の並びの例

**データ**

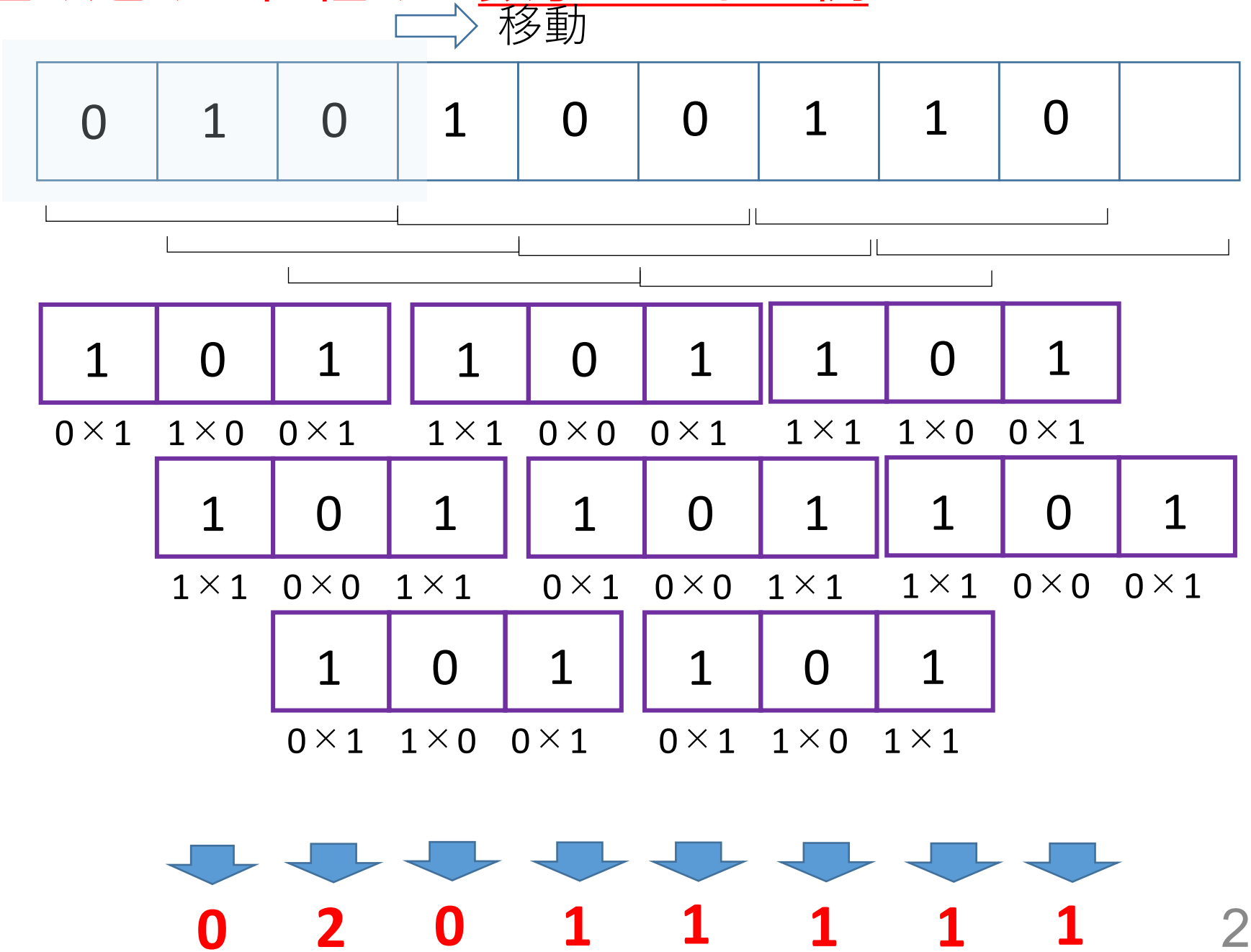



**カーネル**



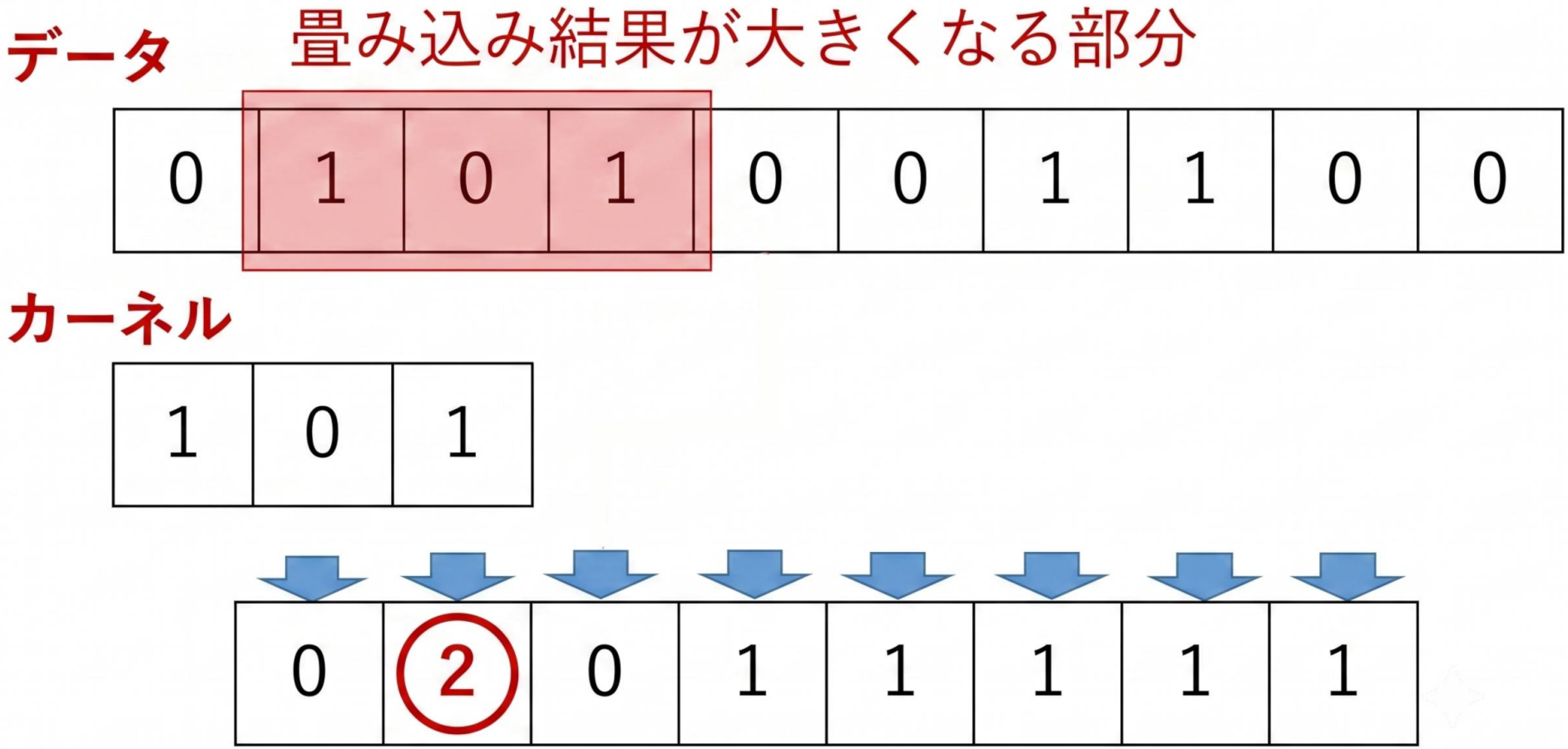

重ね合わせの結果：　0×1 + 1×0 + 0×1 = 0

# 畳み込みの仕組み　数字の並びの例



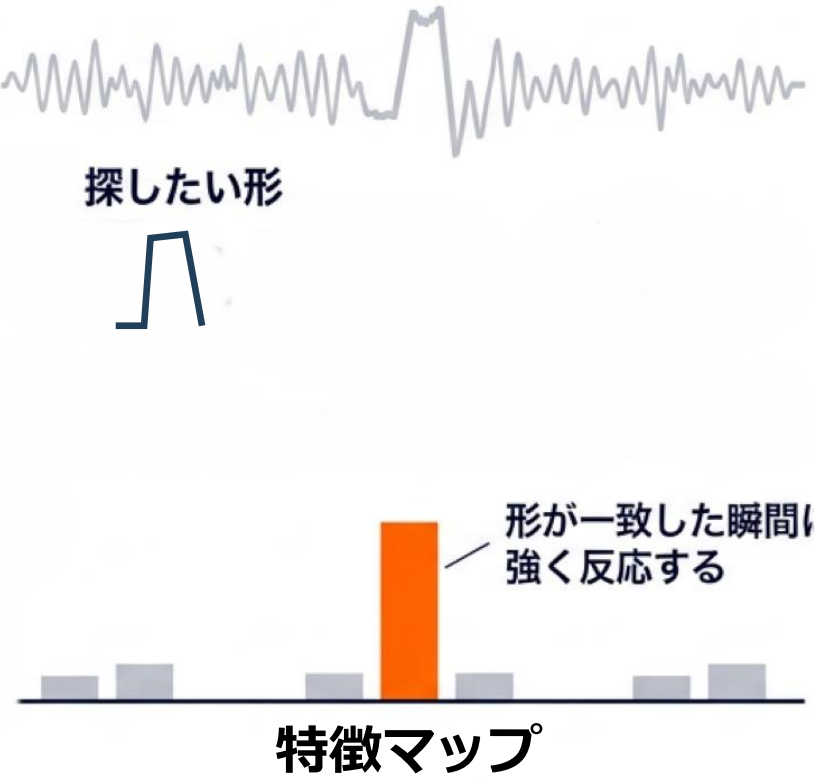
# 畳み込みの仕組み　数字の並びの例

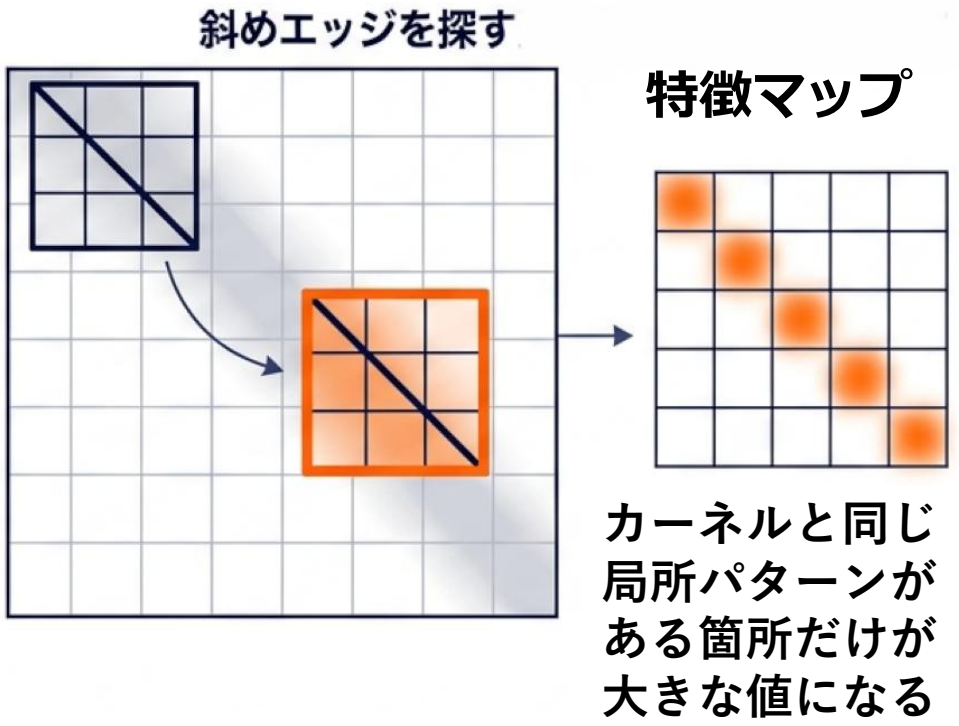
畳み込みは、「特定の局所的特徴に強く反応する」と考えることもできる



# 畳み込みと特徴マップ

畳み込みは、特定の局所的特徴に強く反応する

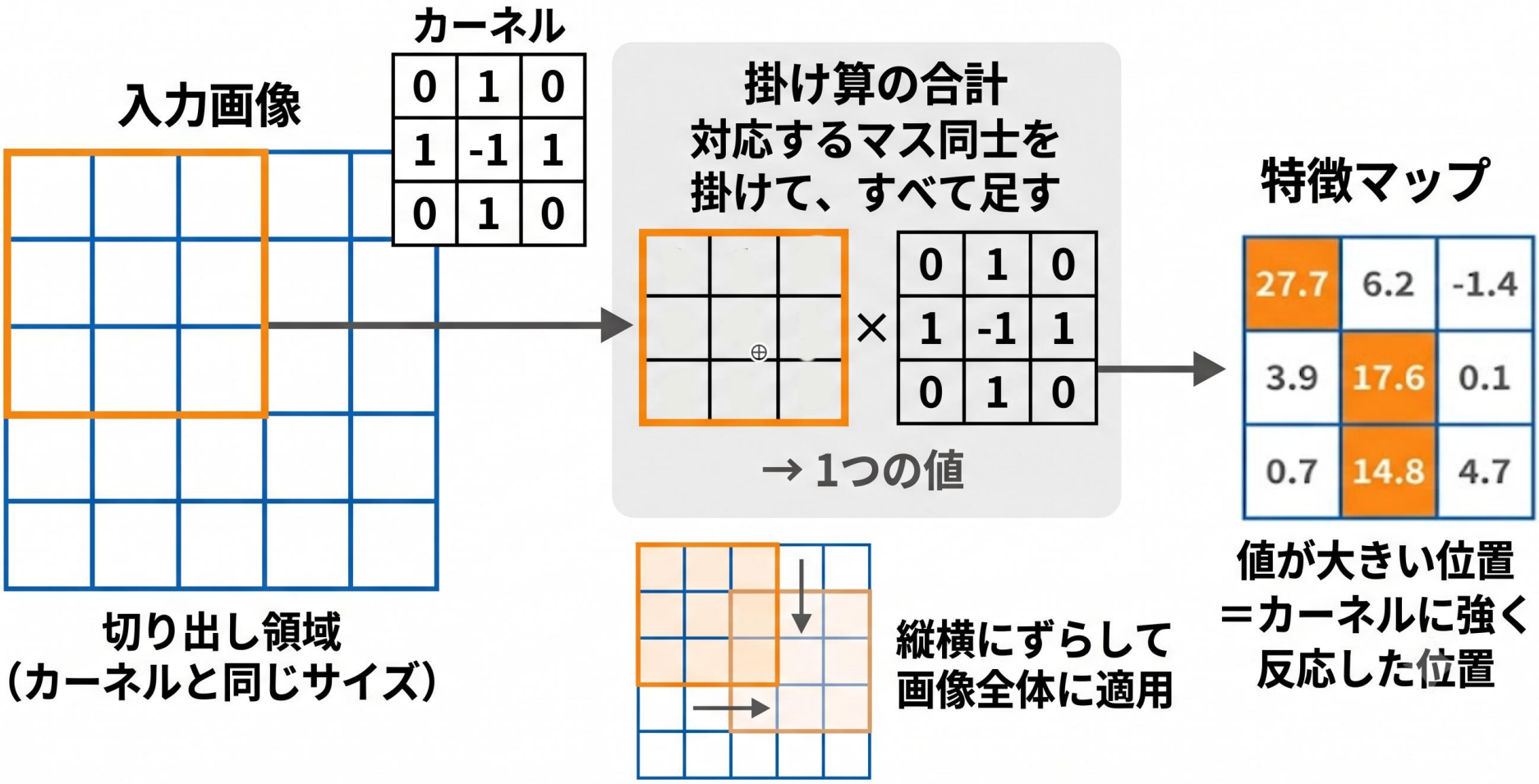



**原理は同じ：畳み込みは局所パターンの一致を検出している**

# 画像の畳み込みと特徴マップ

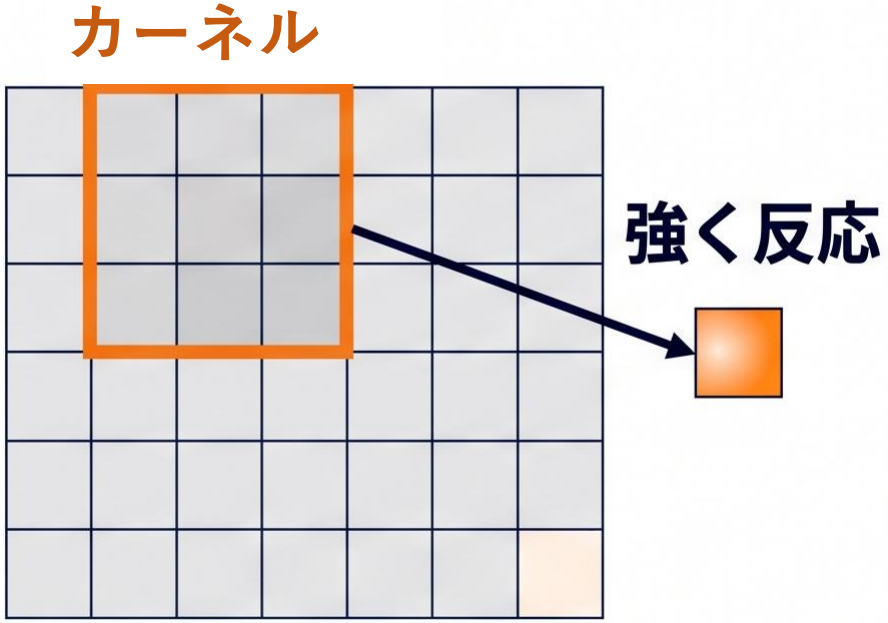
カーネルを画像上でずらしながら重ね合わせ、各位置での反応の強さを特徴マップにまとめる処理



# 画像の畳み込みの特性

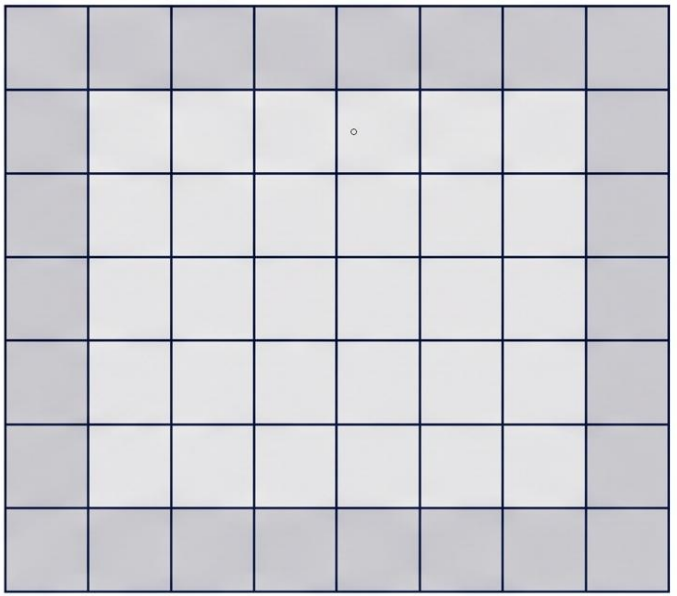
畳み込みは特定の局所的特徴に強く反応する。

**畳み込み＝局所的特徴への反応**




**特定のパターンに一致する領域で出力が大きくなる**

**パディング＝周囲に値を補う**



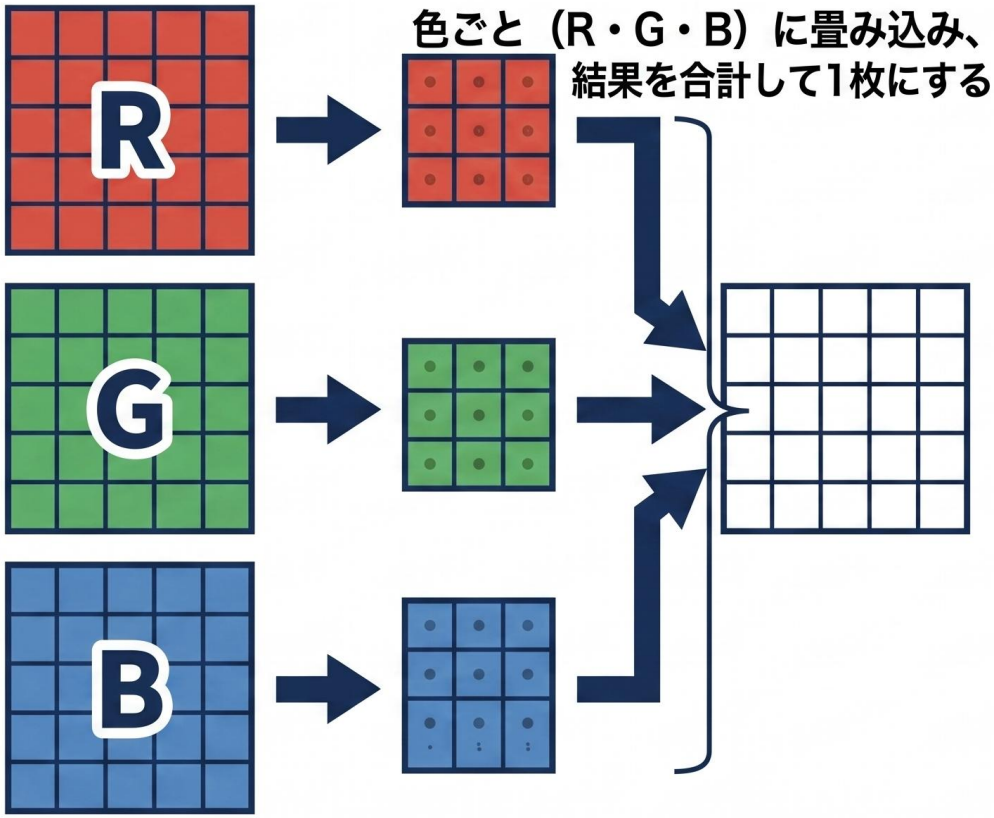
**画像の端を処理できる。**
**特徴マップのサイズを調整できる**

**パディングなし→出力が縮む**
**パディングあり→サイズを保てる**

# 画像の畳み込みの応用例

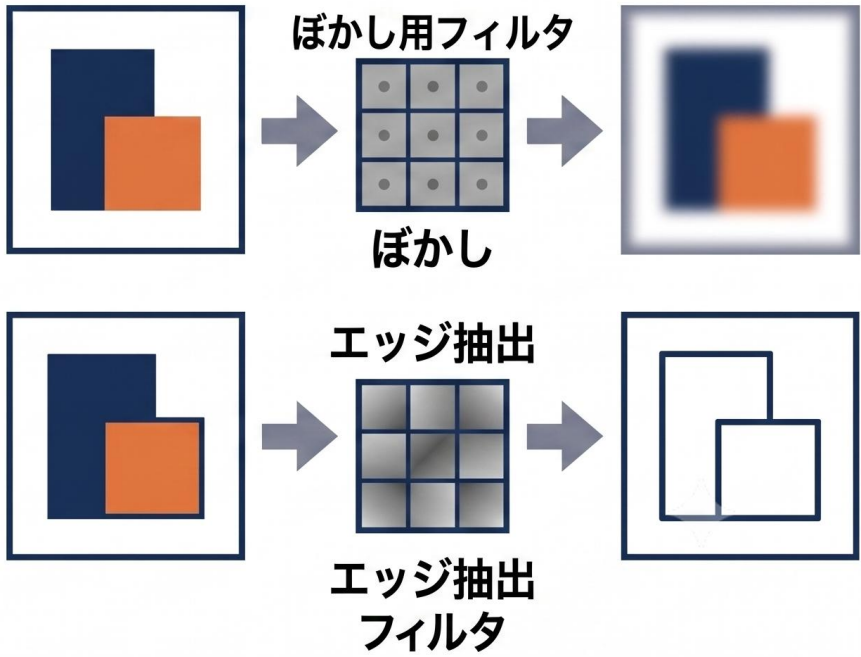
畳み込みは画像の各画素に小さなフィルタを重ね、周囲を計算して新しい画像を作る処理である

## カラー画像の畳み込み




## 応用：人工知能以外でも使える

カーネルの数字を変えるだけで、ぼかし、
エッジ抽出など多様な処理ができる



# 画像の畳み込みを行う Python プログラムの例

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |

元画像（５×５マス）

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 0 | 0 | 1 |

カーネル
（３×３マス）

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |

切り出し（３×３マス）

**カーネルと同じサイズ**
**で切り出す**

**切り出した部分とカーネルの**
**掛け算の合計**

**0×1　1×0　1×1**
**0×1　1×1　1×1**
**0×0　1×0　1×1**

---

**合計: 4**（これが畳み込み結果）

畳み込み

# 画像の畳み込みを行う Python プログラムの例

```
x = [[0, 1, 1, 0, 1],
     [0, 1, 1, 0, 1],
     [0, 1, 1, 0, 1],
     [0, 1, 1, 0, 1],
     [0, 1, 1, 0, 1]]
k = [[1, 0, 1],
     [1, 1, 1],
     [0, 0, 1]]
y = [[0, 0, 0],
     [0, 0, 0],
     [0, 0, 0]]

def conv(i, j):
    return x[i + 0][j + 0] * k[0][0] + x[i + 0][j + 1] * k[0][1] + x[i + 0][j + 2] * k[0][2] ¥
    + x[i + 1][j + 0] * k[1][0] + x[i + 1][j + 1] * k[1][1] + x[i + 1][j + 2] * k[1][2] ¥
    + x[i + 2][j + 0] * k[2][0] + x[i + 2][j + 1] * k[2][1] + x[i + 2][j + 2] * k[2][2]

y[0][0] = conv(0, 0)
y[0][1] = conv(0, 1)
y[0][2] = conv(0, 2)
y[1][0] = conv(1, 0)
y[1][1] = conv(1, 1)
y[1][2] = conv(1, 2)
y[2][0] = conv(2, 0)
y[2][1] = conv(2, 1)
y[2][2] = conv(2, 2)

print(y)
```

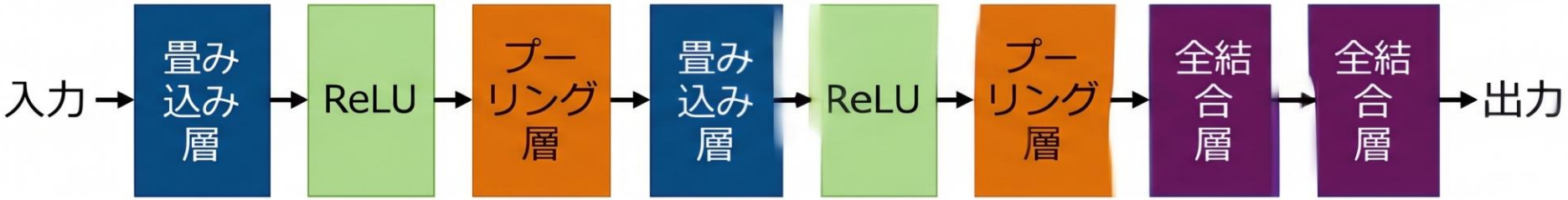
```
[[4, 3, 5], [4, 3, 5], [4, 3, 5]]
```

https://colab.research.google.com/drive/1pcdD-I5-2VbLizKb3egJIQWXn8tBfOuK?usp=sharing

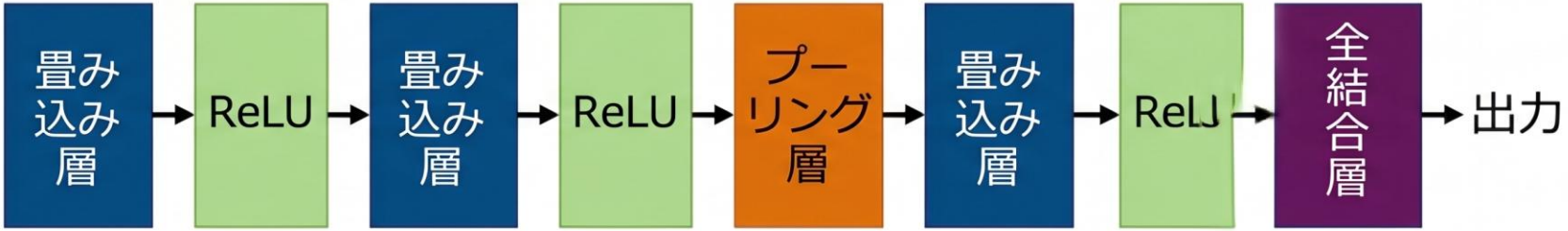
# 畳み込みニューラルネットワーク（CNN）

**畳み込み層、プーリング層、全結合層**を組み合わせることで，さまざまなCNNを構築できる。

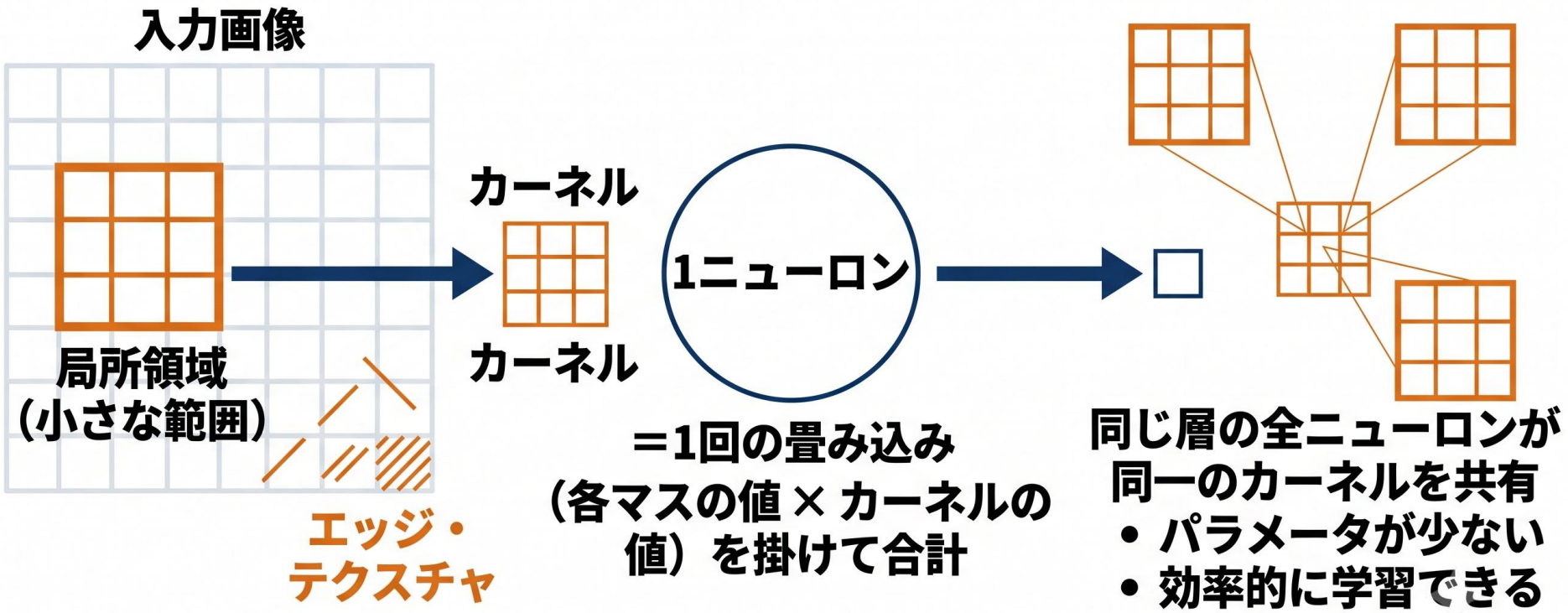
**例 1: 古典的な組み合わせの繰り返し**




**例 2: 畳み込みを連続させる構成（VGGスタイル）**



# 畳み込み層は画像の局所的な特徴をとらえる




1ニューロンが1つの局所領域の畳み込みを担当し、層全体でカーネルを共有する

# プーリング層


**プーリング層：特徴マップを縮小し，過学習を防ぐ**

4×4 の特徴マップ

| | | | |
|---|---|---|---|
| 1 | 3 | 5 | 6 |
| 2 | 8 | 7 | 2 |
| 4 | 2 | 9 | 3 |
| 1 | 0 | 1 | 6 |

2×2 の
最大値をとる

2×2 の出力

| | |
|---|---|
| 8 | 7 |
| 4 | 9 |

**サイズ縮小**
特徴マップが小さくなる

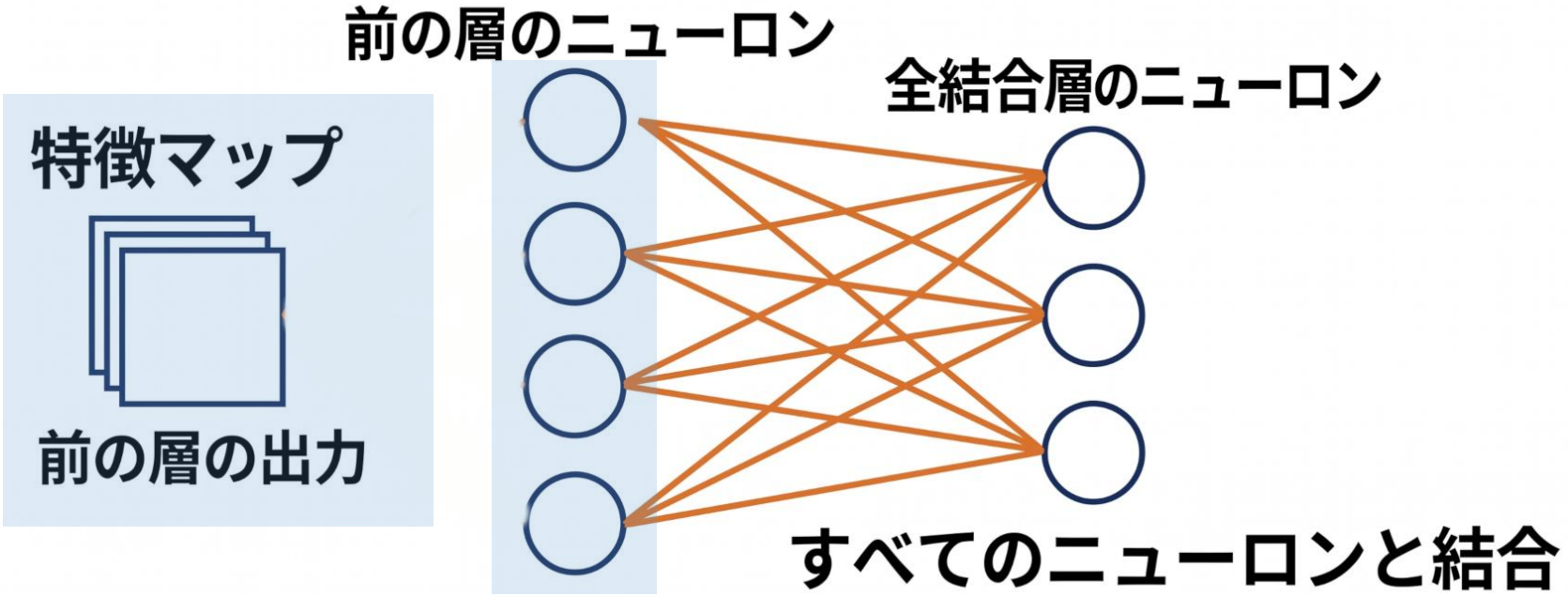
**パラメータ減少**
後続の層で調整する値が減る

**位置のずれに強い**
物体が多少ずれても
同じ結果になりやすい

**過学習を抑える**

# 全結合層

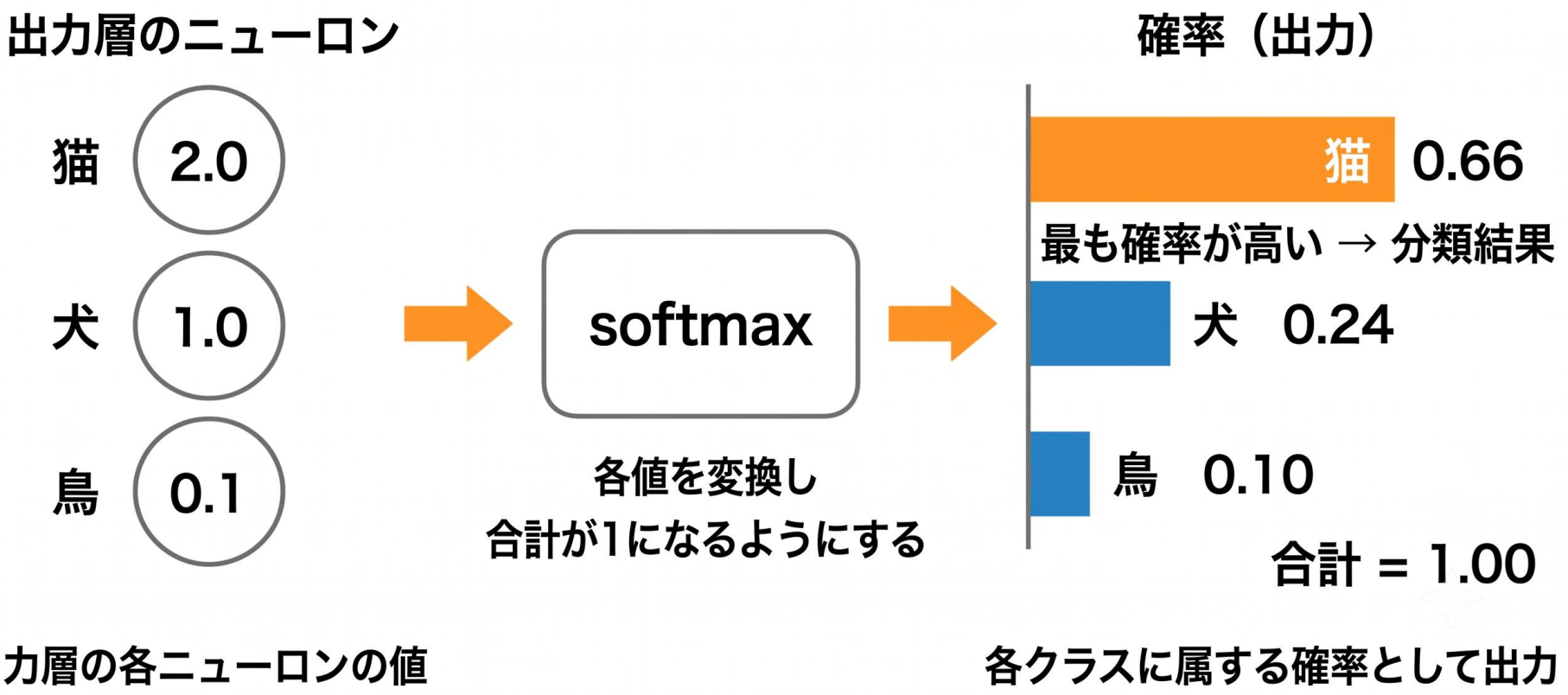
**全結合層：前の層のすべてのニューロンとつながり，特徴を統合して画像を処理する**




- 全結合層は特徴マップをもとに画像全体を処理する
- CNNだけでなく，通常のニューラルネットワークにも用いられる

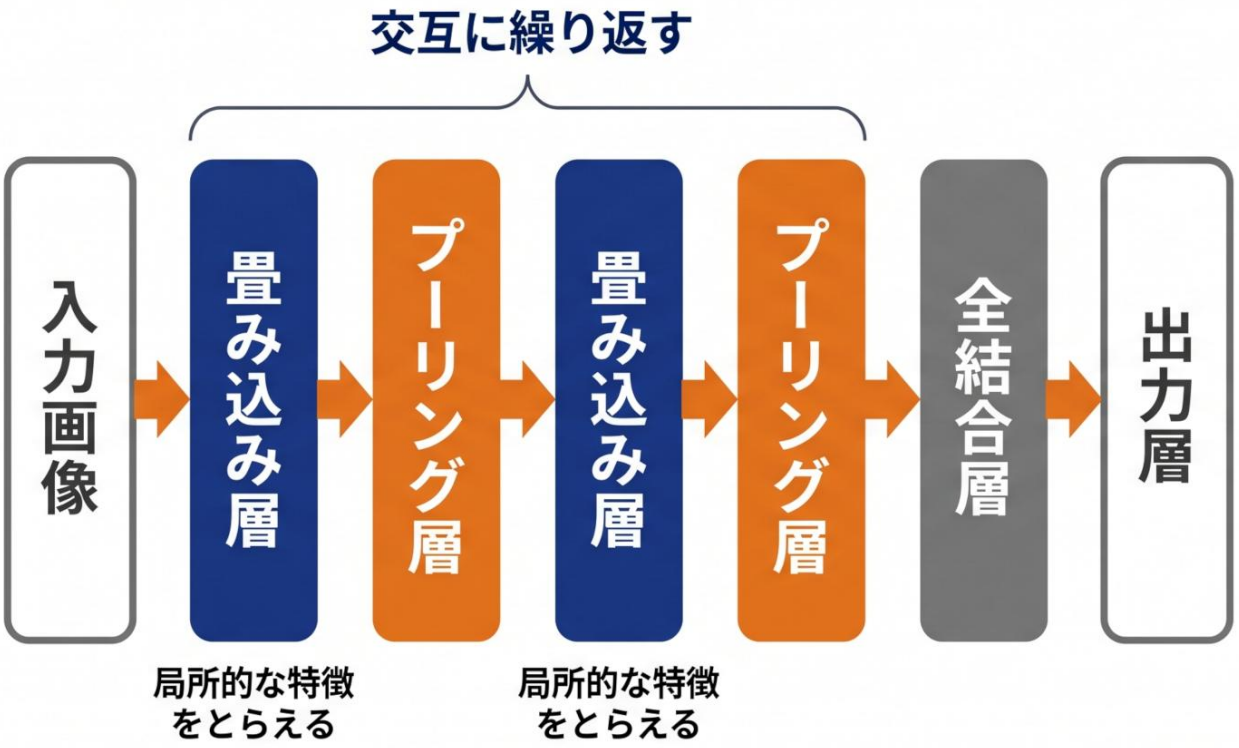
# 出力層とソフトマックス (softmax)

softmax は出力層の各値を合計1の確率に変換し、最も確率の高いクラスを選ぶ



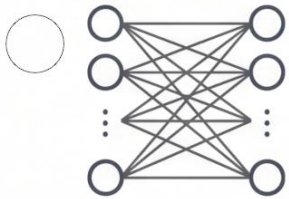
# 畳み込みニューラルネットワーク（CNN）の利点

**CNN は畳み込み層とプーリング層を繰り返し、画像の特徴を自動で抽出する**



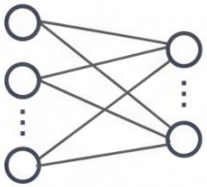

一般的なニュラルネットワーク

全結合層だけの場合



結合が多い → 過学習しやすい

CNN（畳み込み層）



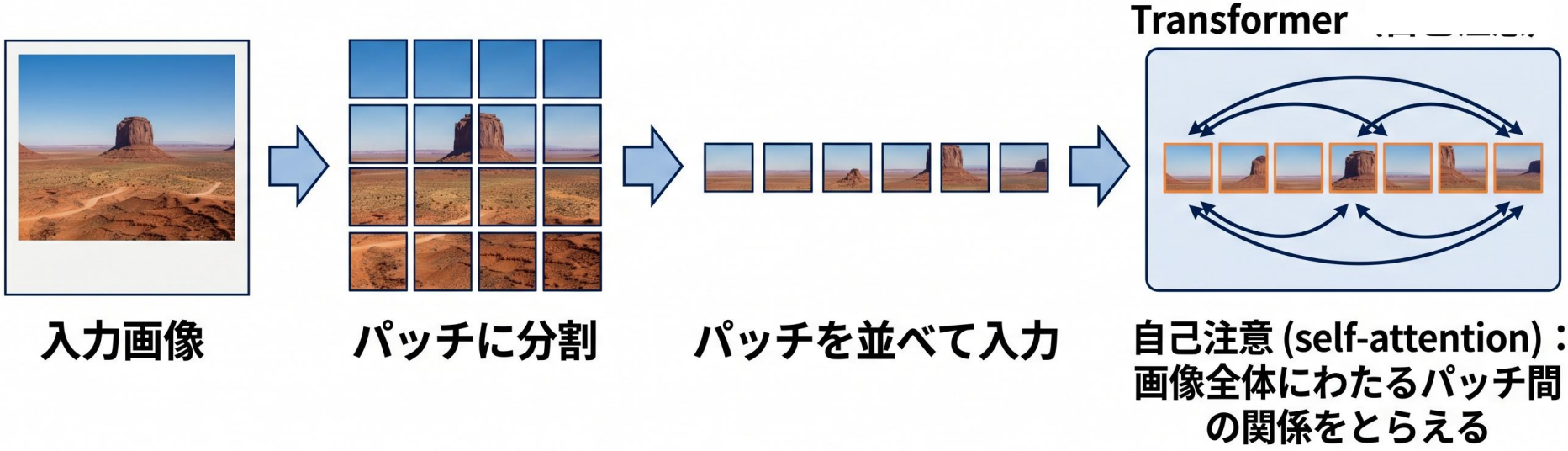
結合を局所に限定 → 結合数を大幅削減
カーネルを共有 → 学習が効率化

- 結合数の削減で過学習を緩和

# 8-4. Vision Transformer

# Vision Transformer (ViT)

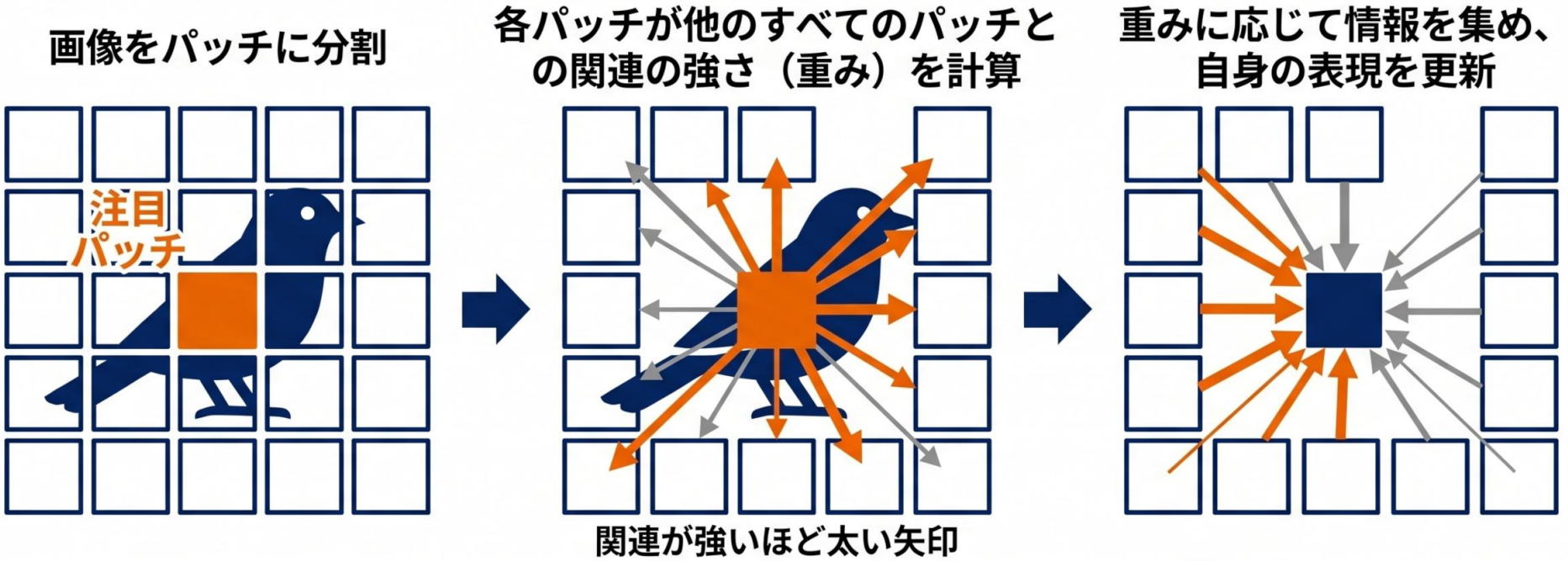
## ViTは，画像を小さなパッチに分割し，Transformerで処理する画像認識モデル




Transformerは自然言語処理で使われる，自己注意を中心とするモデルである。
ViTはこれを画像に適用している。

# Vision Transformer (ViT) の仕組み

**Vision Transformer (ViT) は、「自己注意」という仕組みにより、画面上で離れたパッチ同士が、関連の強さに応じて情報をやり取りできる**




**CNN**

近くの領域を段階的に処理
（長距離の関係は苦手）

**ViT（自己注意）**

離れたパッチも
直接やり取り

# ViT と CNN の使い分け


**ViTとCNNは対立せず、データ量や用途に応じて使い分ける**

## ViT (Vision Transformer)

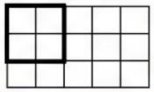
高い性能を出すには
大量の学習データと
計算資源が必要

画像向けの前提を構造として持たない

## CNN

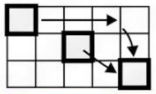
画像に適した前提を
構造としてあらかじめ持つ

前提1：近くの画素どうしが関係する



カーネルが局所領域だけを見る（畳み込み）

前提2：位置がずれても同じ特徴をとらえる



同一のカーネルを共有してずらして適用

**データが少ない／局所的な特徴を扱う場合はCNNが有利**
**ViTとCNNは用途とデータ量で使い分ける**

# 8-5. 画像理解の3種類

# 画像分類

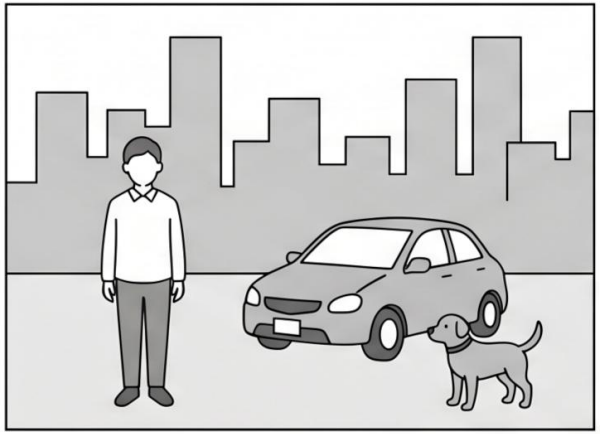
**画像分類：画像全体を見て『何が写っているか』を1つのクラスラベルで答える**


**ポイント：出力は『どこに写っているか』ではなく、『何が写っているか』というラベル**
**画像全体に対して1つの答えを返す（物体検出のように位置は示さない）**

# 物体検出

**物体検出：画像の中の「何が」「どこに」「どの大きさで」あるかを特定する**

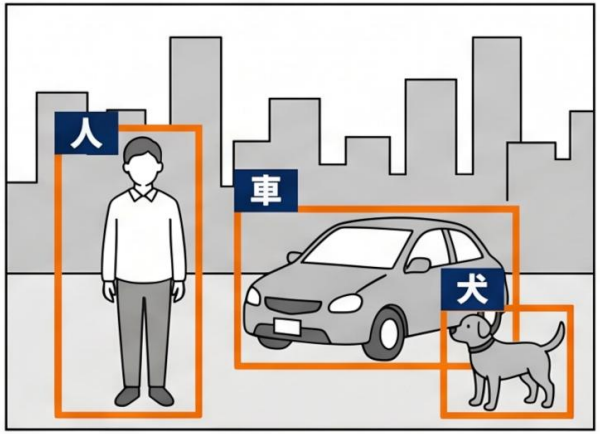
**入力**



**物体検出**

位置・大きさ・種類を
同時に特定

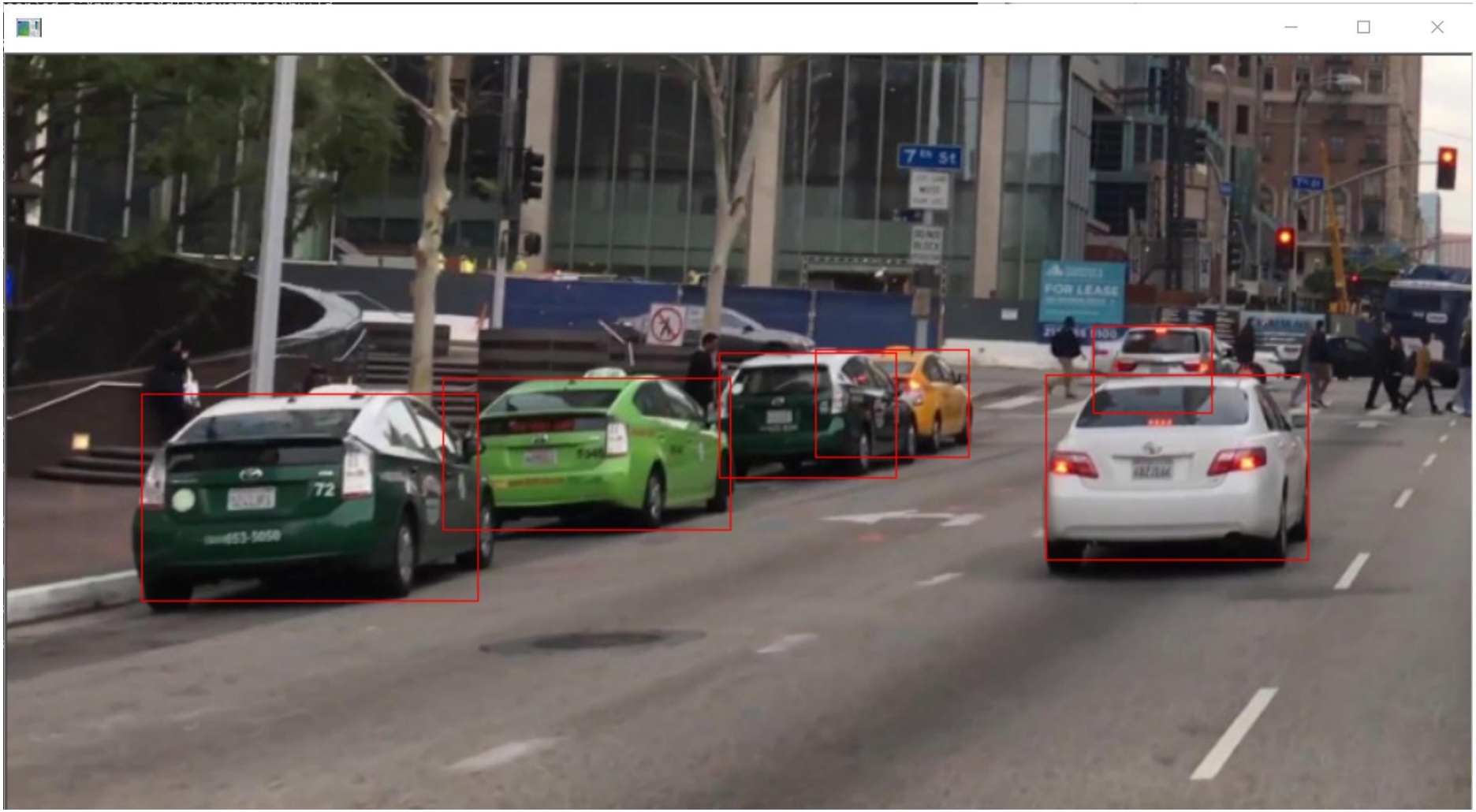
**出力**



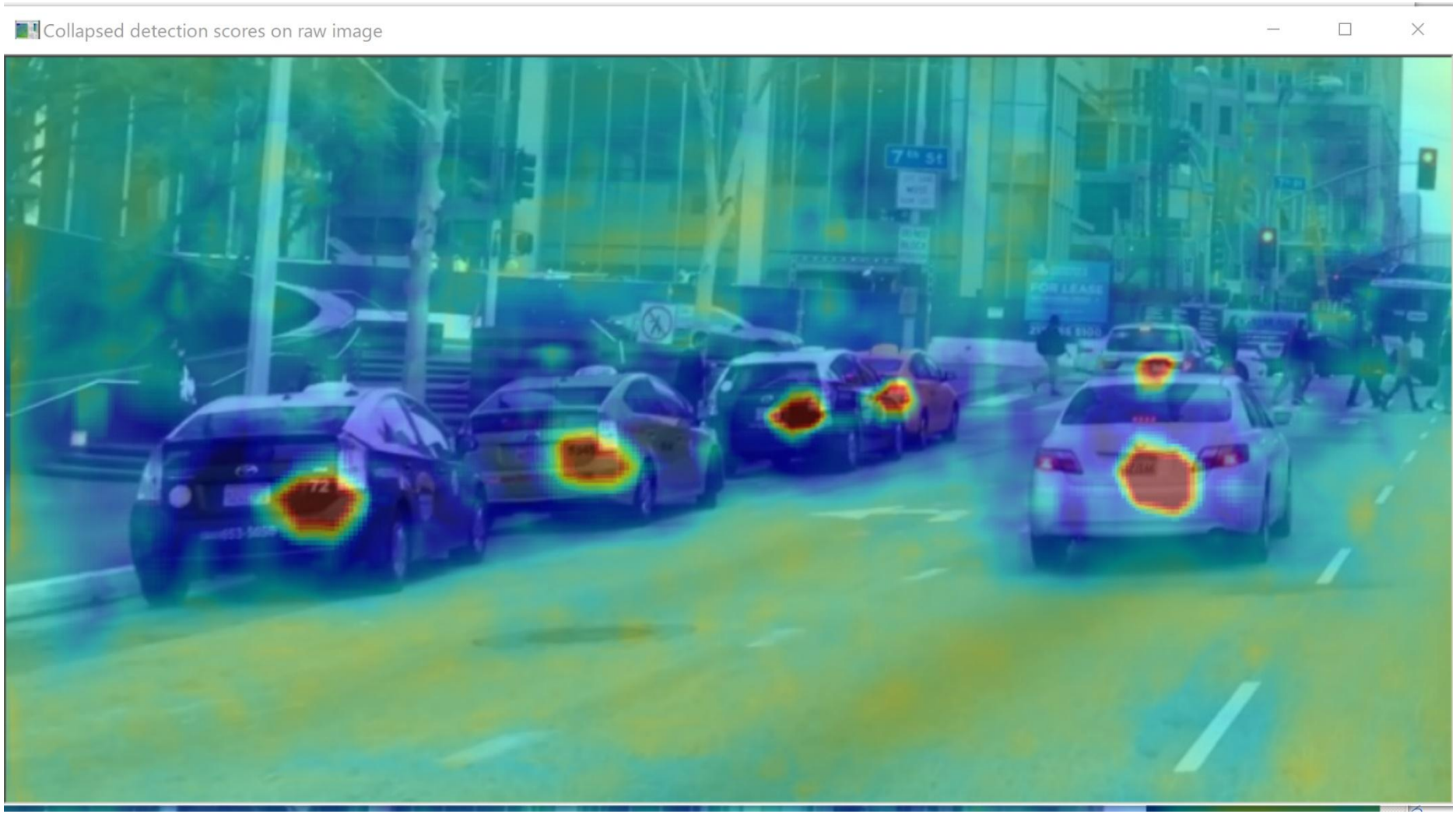

**バウンディングボックス＝物体を囲む最小の四角形**


**出力されるのは「物体を囲む四角形（位置と大きさ）」と「ラベル（種類）」**
**四角形は物体を囲む最小の長方形で、位置と大きさを表す**

# 物体検出



# 物体検出における特徴マップ



# 物体検出モデル YOLO のしくみ

**"物体の位置" と "種類" を一括で予測する高速な物体検出モデル**

## 画像分類（既知）

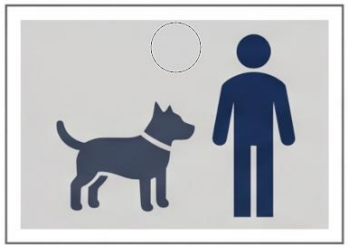

画像全体に1つの種類を予測

## 物体検出（YOLO）

＋位置


位置（枠）＋種類 を同時に予測

## 入力画像



## 単一のネットワーク（CNN基盤）

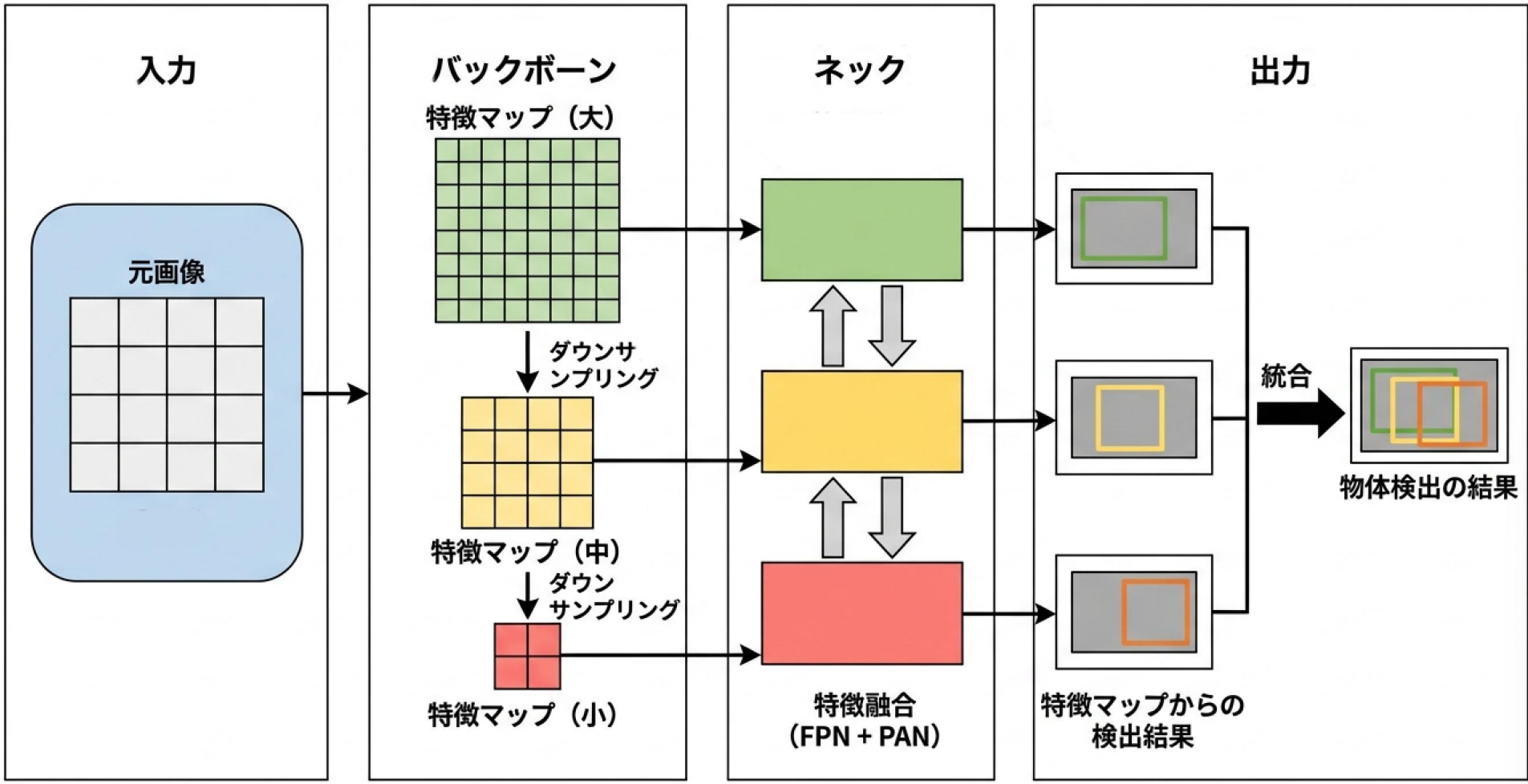
A **バックボーン**
特徴抽出

B **ネック**
特徴マップの
統合・改善

## 出力


**1段階方式**：領域の候補を別に求めず，**1つのネットワーク**で一括推論 → 高速

# バックボーンとネック



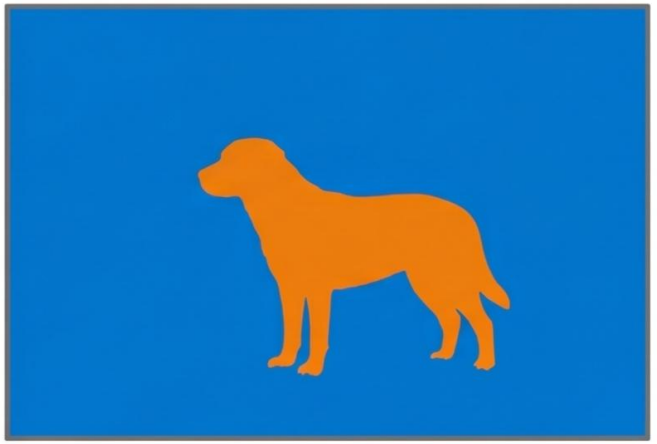

**FPN+PAN**（複数サイズの特徴マップを融合する仕組み）
**ダウンサンプリング**（特徴マップを縮小する処理）

# セグメンテーション

**セグメンテーション：画像を意味のある領域に分割し、物体の形を詳細に解析する**

入力画像

領域に分割



物体の形を領域として抽出

**位置が分かる**

物体が画像のどこにあるかを正確に把握できる

**形・大きさを数値化**

物体の形や大きさを数値として測る基礎になる

# セグメンテーションの例

元画像


セグメンテーションの結果

# セグメンテーションの種類

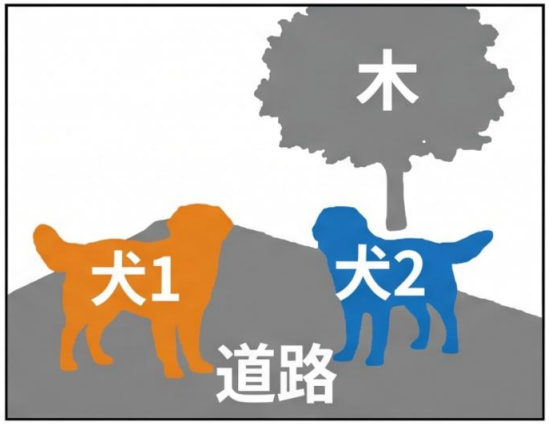
画素ごとにラベルを付け、物体の正確な形を抽出する技術

入力画像

セマンティック・
セグメンテーション


同じクラスの物体
を区別しない

インスタンス・
セグメンテーション



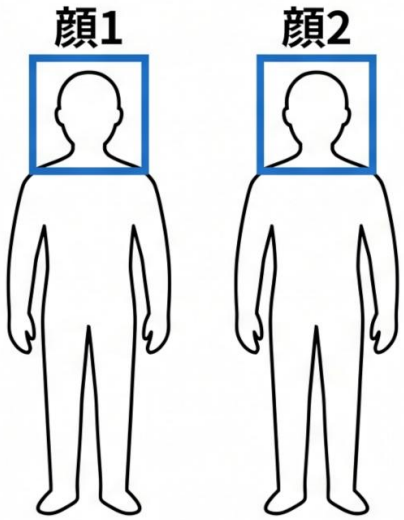

同じクラスでも
個々の物体を区別する

どちらも画素ごとにクラスを割り当て、物体の形状を画素単位で正確に抽出する

# 顔情報処理：顔検出と顔ランドマーク検出

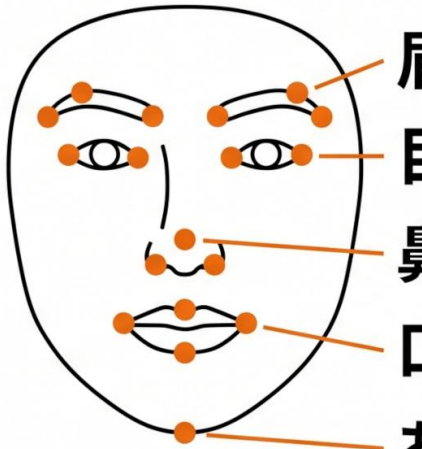
## 顔情報処理は『顔検出』→『顔ランドマーク検出』の順に進む

### 1. 顔検出



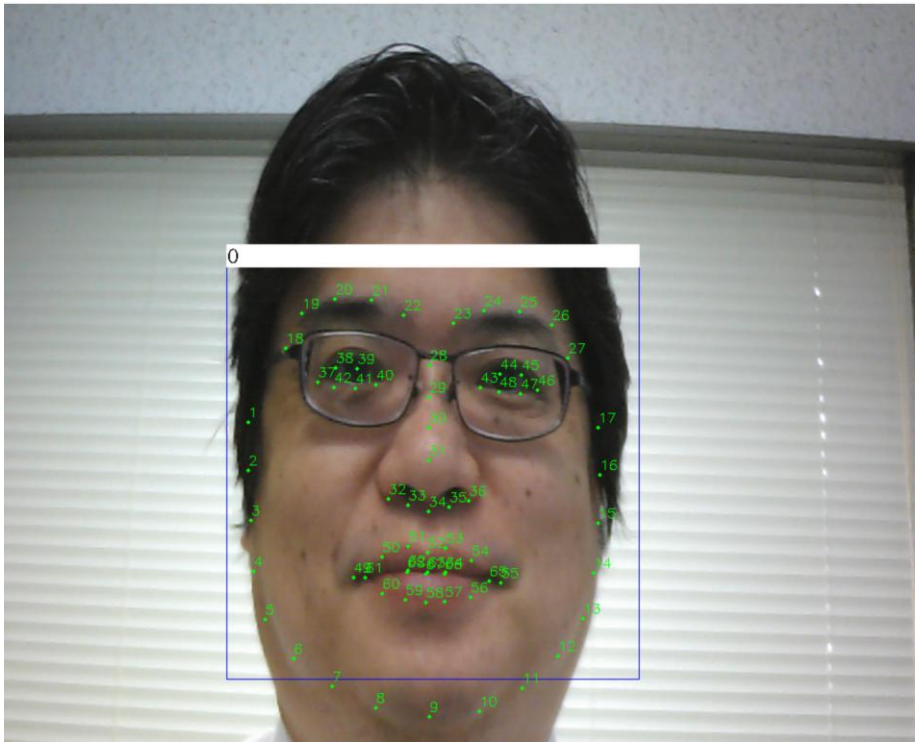

- 画像の中から顔を見つける
- 顔の数を数える
- 顔情報処理の最初の処理

### 2. 顔ランドマーク検出

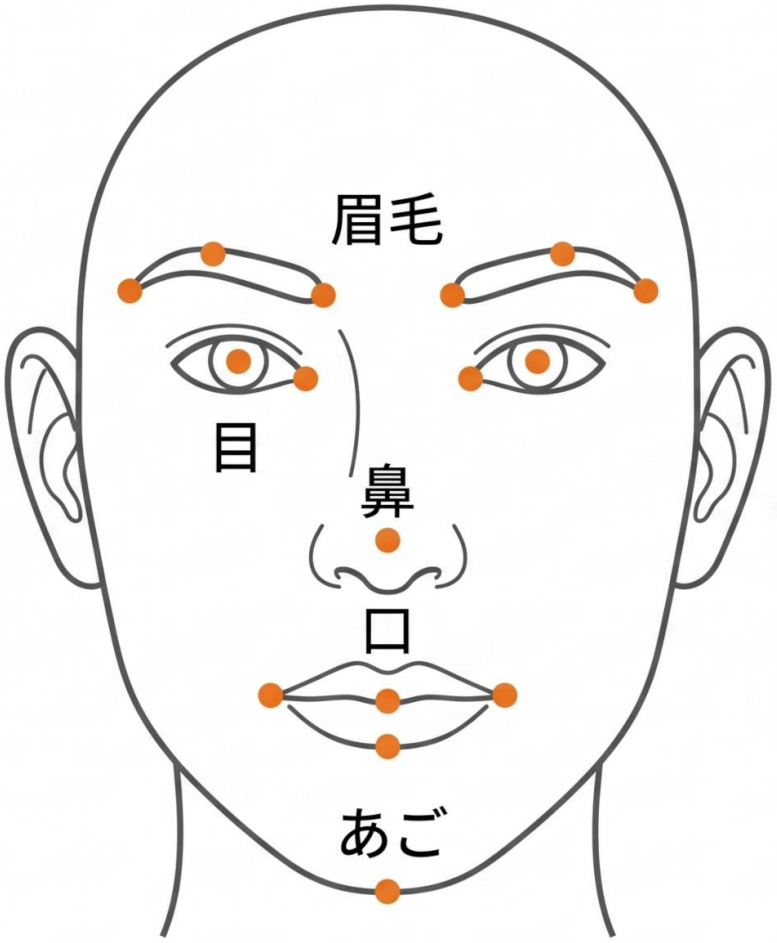



- 目，鼻，口，眉，あごなどの位置を特定
- 用途：位置合わせ/表情分析/人物特定

# 顔ランドマーク



# 顔ランドマークの応用

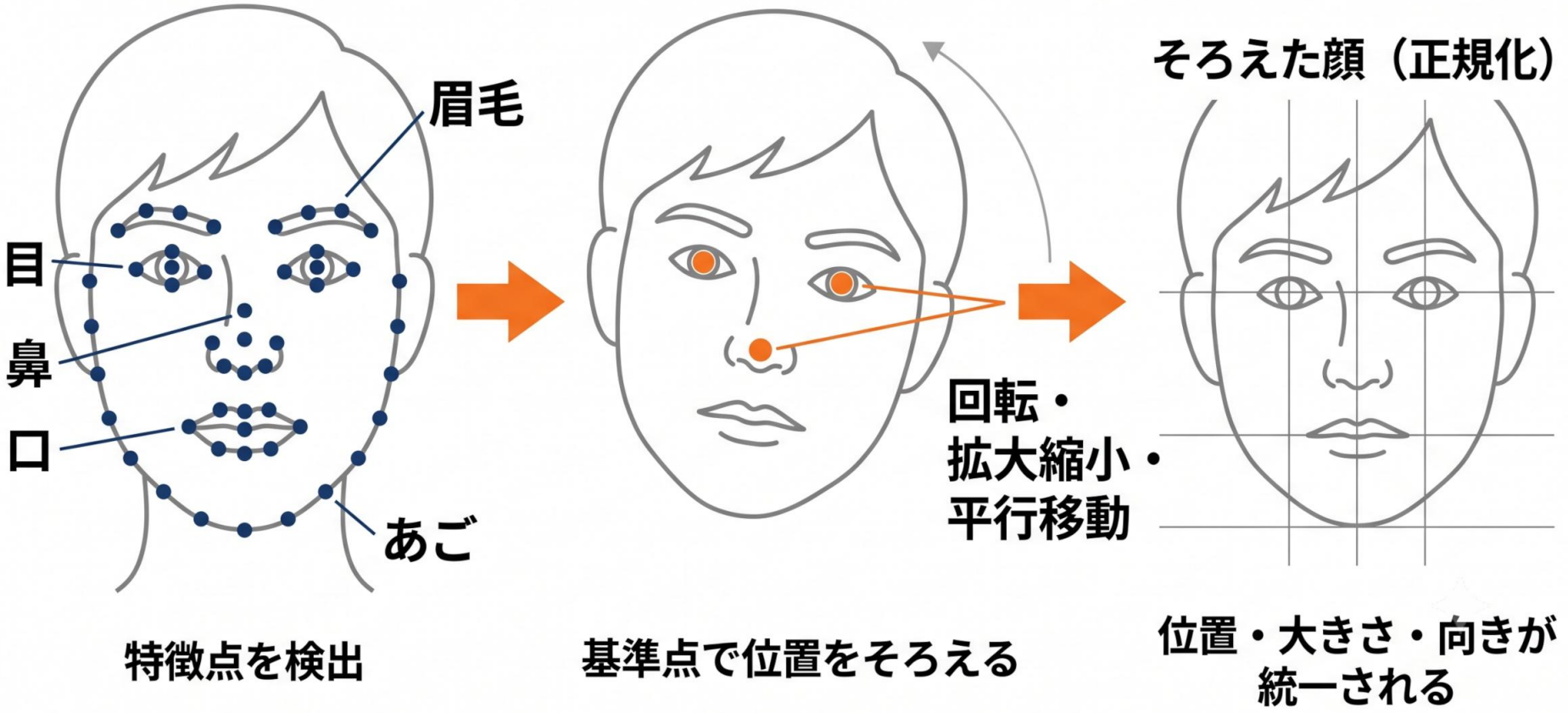



顔ランドマーク

- 位置合わせ
- 年齢・性別・表情の推定
- 人物特定
- 顔の3次元再構成

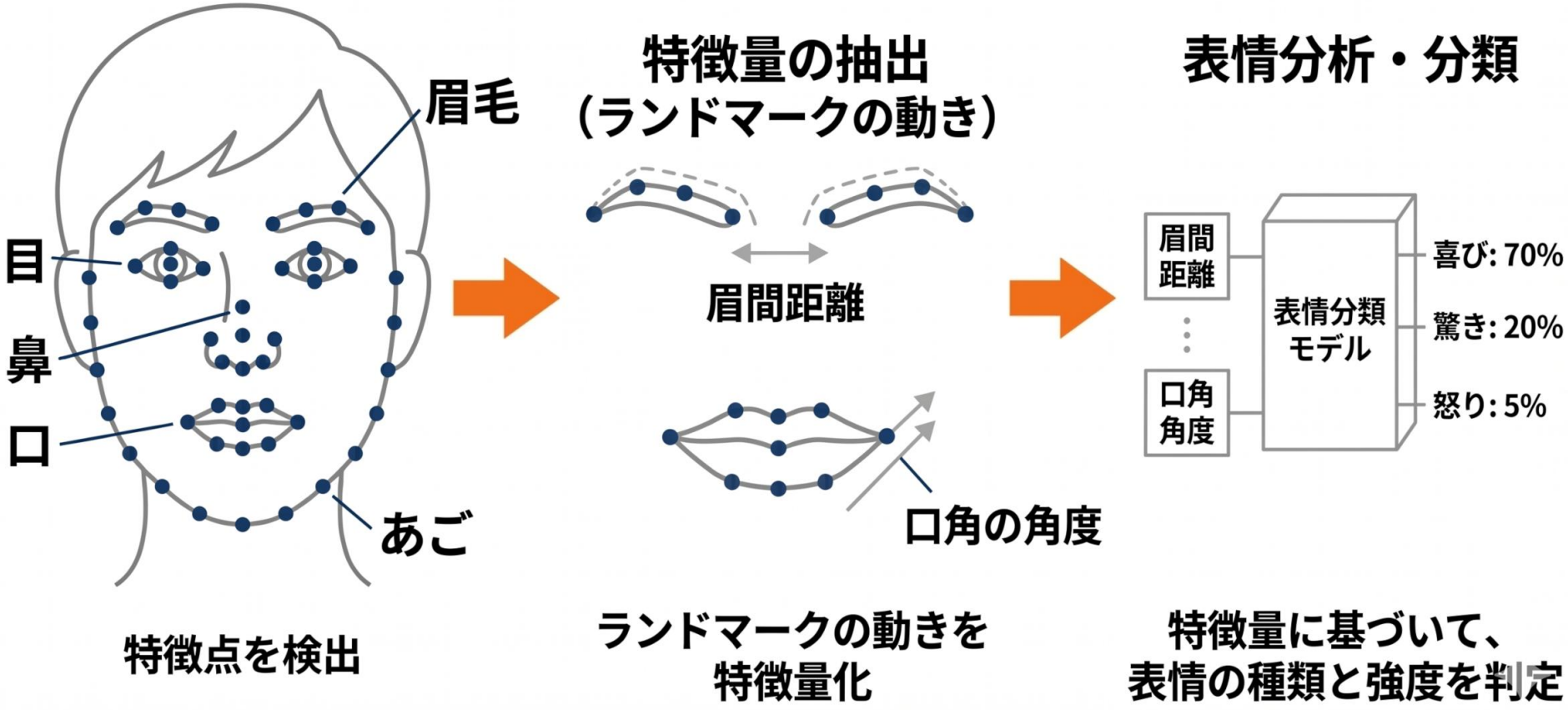
# 顔の位置合わせ（正規化）

顔ランドマーク（目・鼻・口・眉毛・あご）を基準点として、顔を標準の位置・大きさ・向きにそろえる

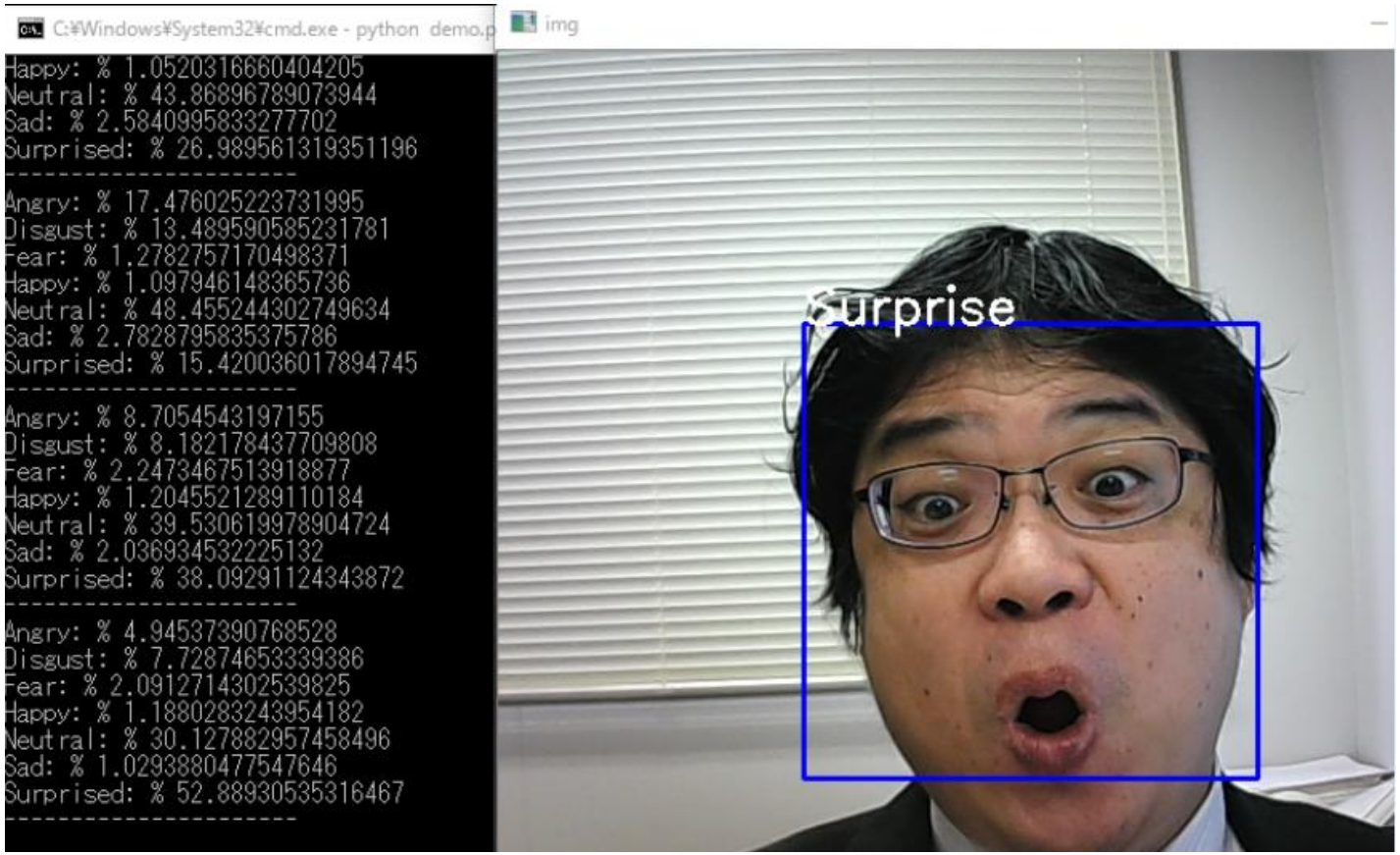


# 表情の推定

検出された顔ランドマークの相対的な位置関係から、表情や感情を分析する

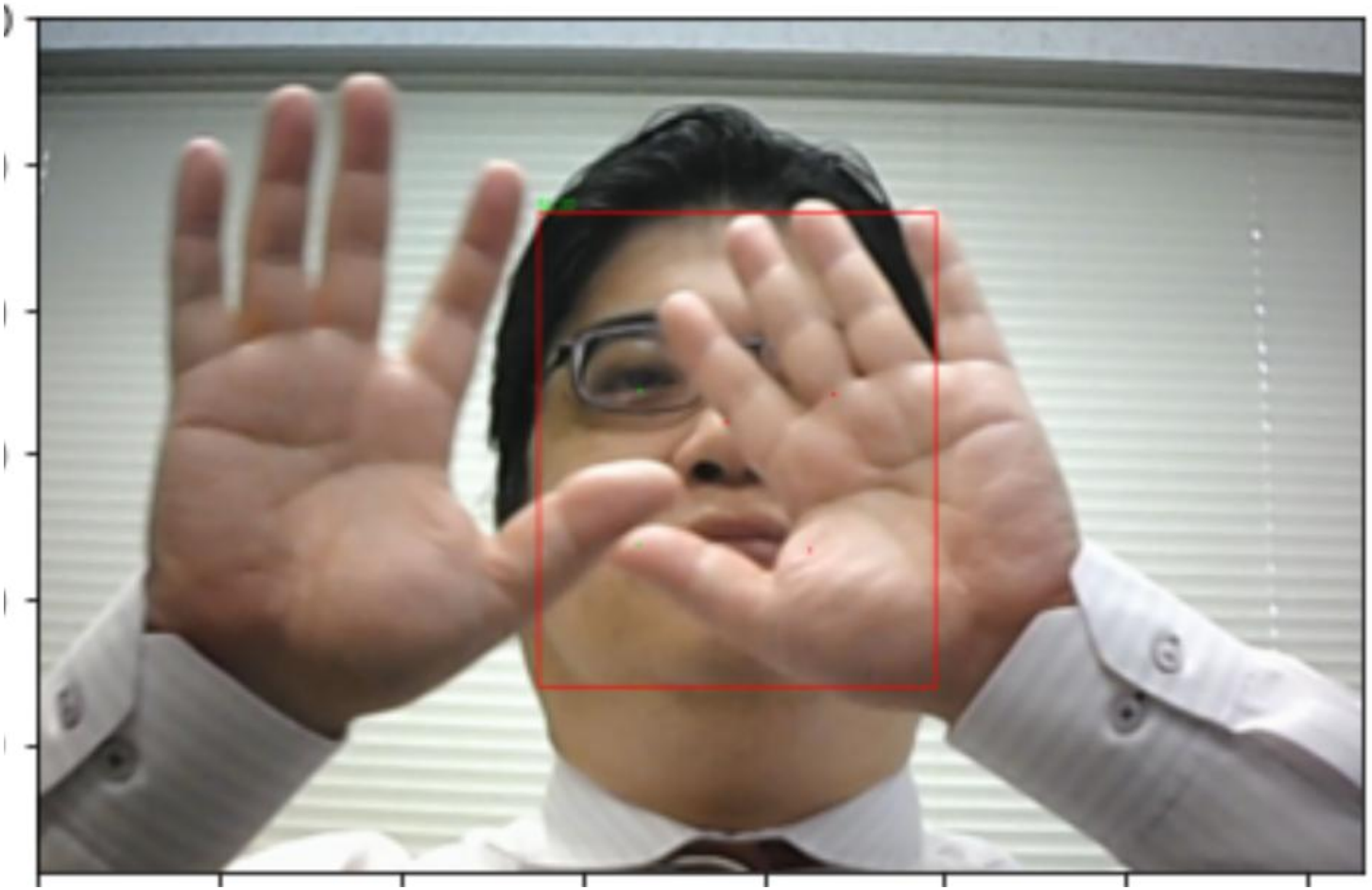


# 表情の推定




- ここでは，**７種の表情** Angry, Disgust, Fear, Happy, Neutral, Sad, Surprised の**それぞれの確率**を推定
- https://github.com/ezgiakcora/Facial-Expression-Keras で公開されている成果物

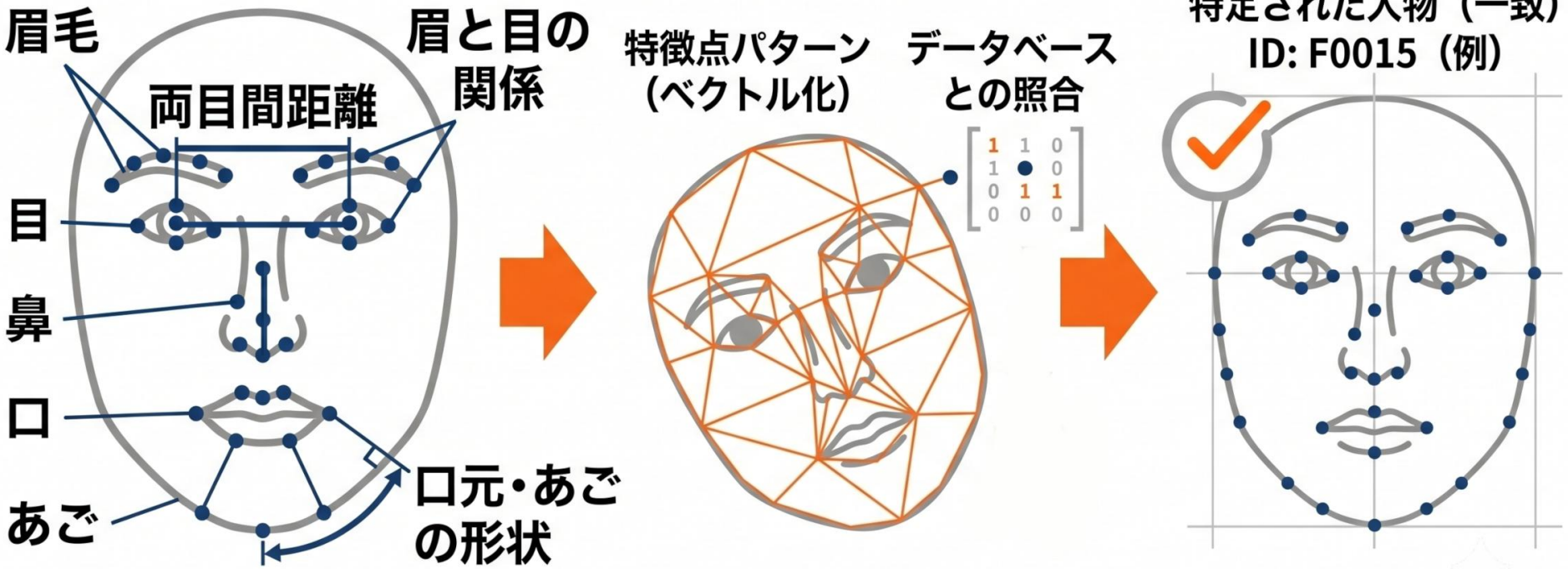
# 年齢の推定，性別の推定



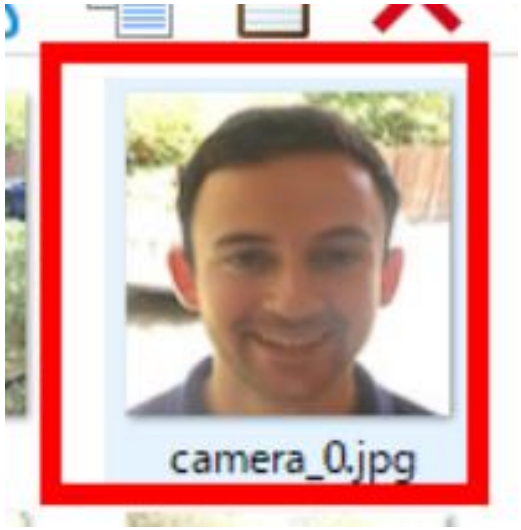
‘gender’: 1 (男性), ‘age’: 30

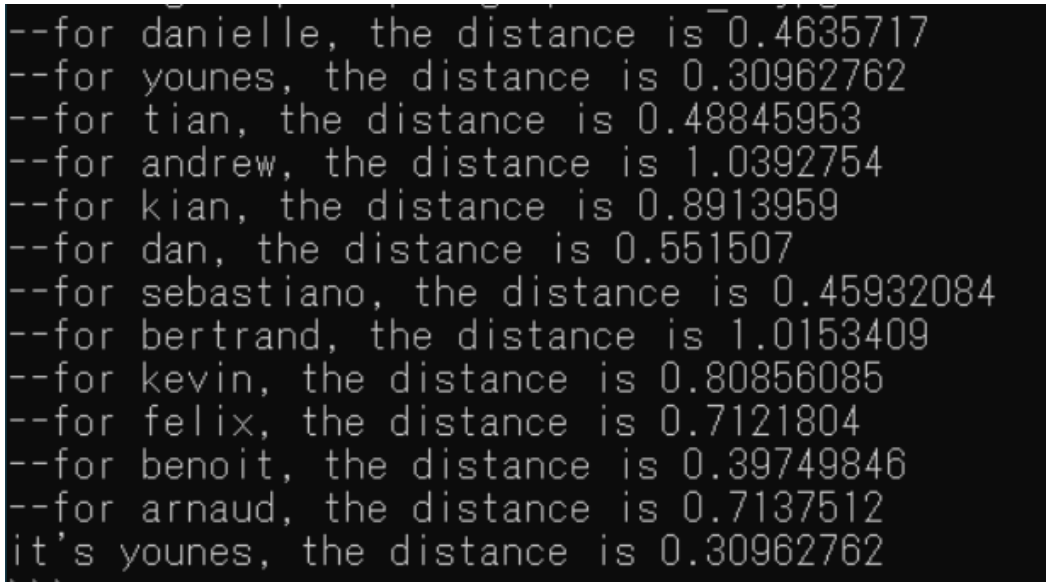
# 顔ランドマークと人物特定

顔ランドマーク（特徴点）の幾何学的配置を分析し、人物特定（マッチング）を行うプロセス



# 人物特定

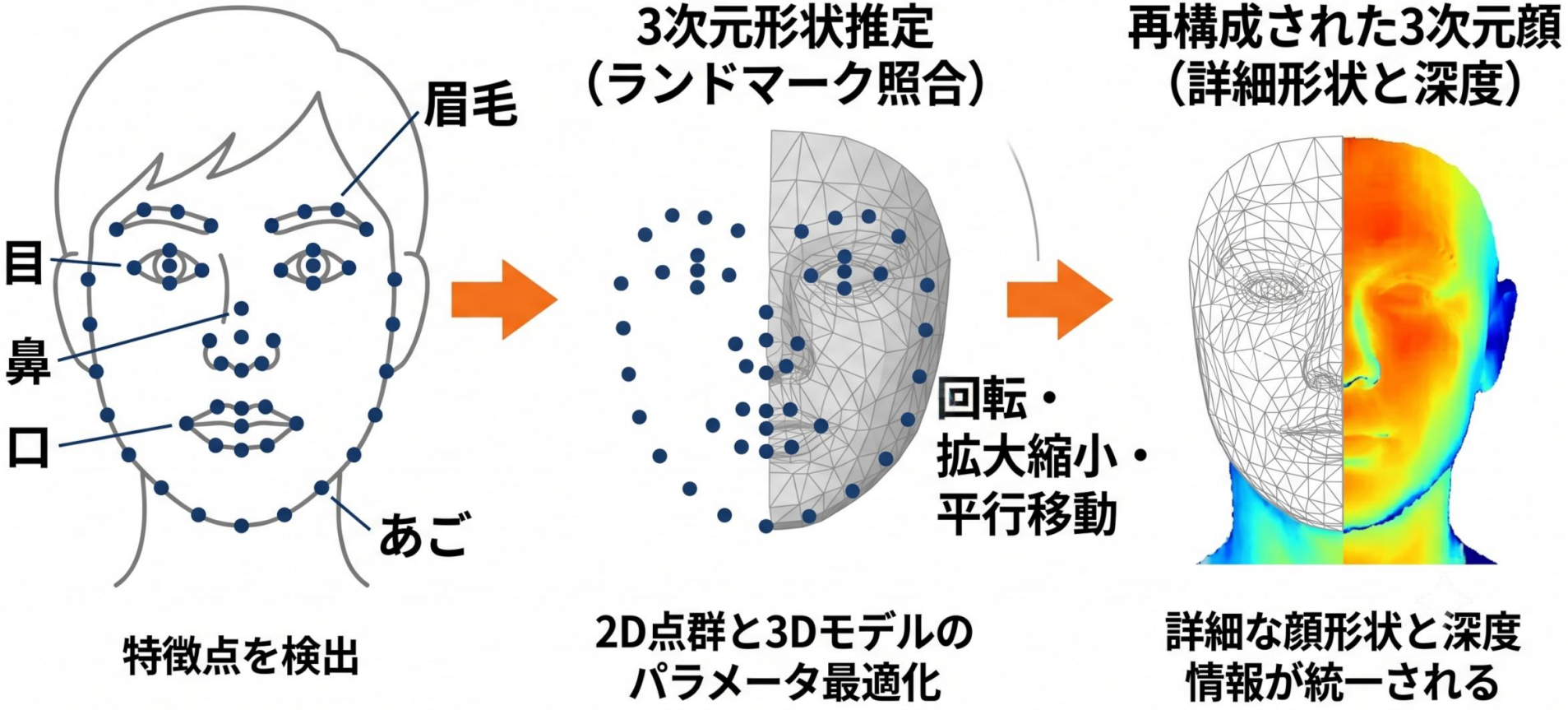

camera_0.jpg


--for danielle, the distance is 0.4635717
--for younes, the distance is 0.30962762
--for tian, the distance is 0.48845953
--for andrew, the distance is 1.0392754
--for kian, the distance is 0.8913959
--for dan, the distance is 0.551507
--for sebastiano, the distance is 0.45932084
--for bertrand, the distance is 1.0153409
--for kevin, the distance is 0.80856085
--for felix, the distance is 0.7121804
--for benoit, the distance is 0.39749846
--for arnaud, the distance is 0.7137512
it's younes, the distance is 0.30962762

younes

# 顔ランドマークと顔の3次元再構成

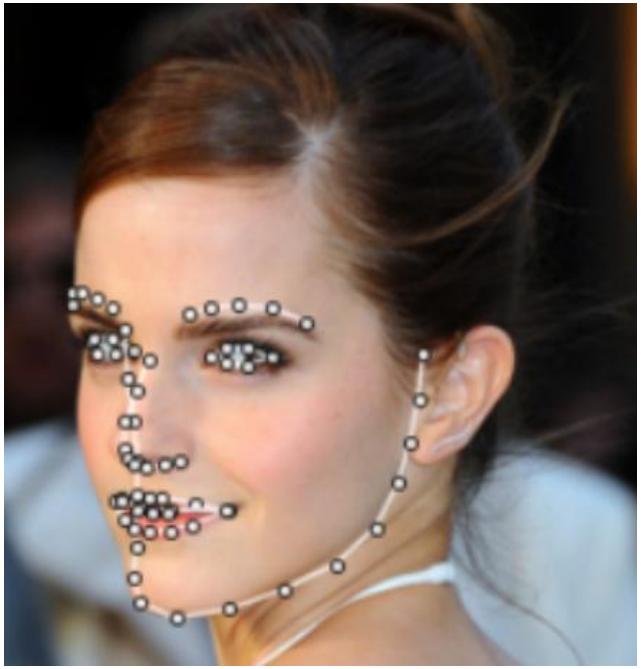
顔ランドマーク（目・鼻・口・眉毛・あご）を基準点として、2次元画像から顔の3次元形状を再構成する



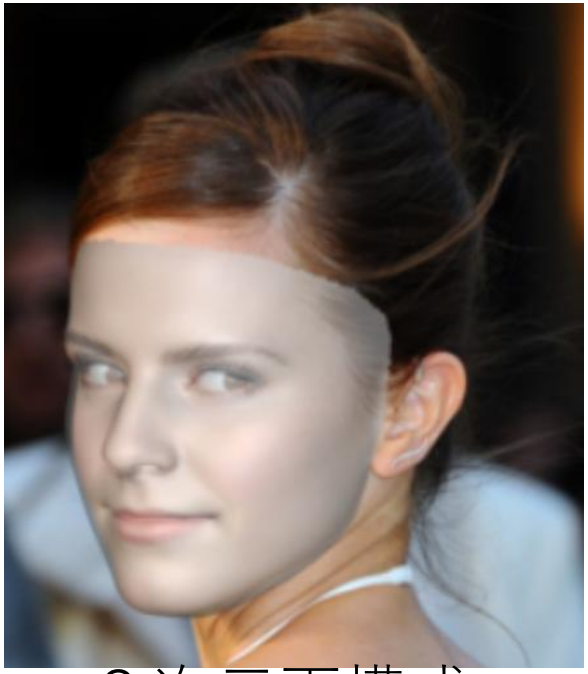
# 顔の３次元再構成

## 3DFFA 法（2022年発表）

元画像



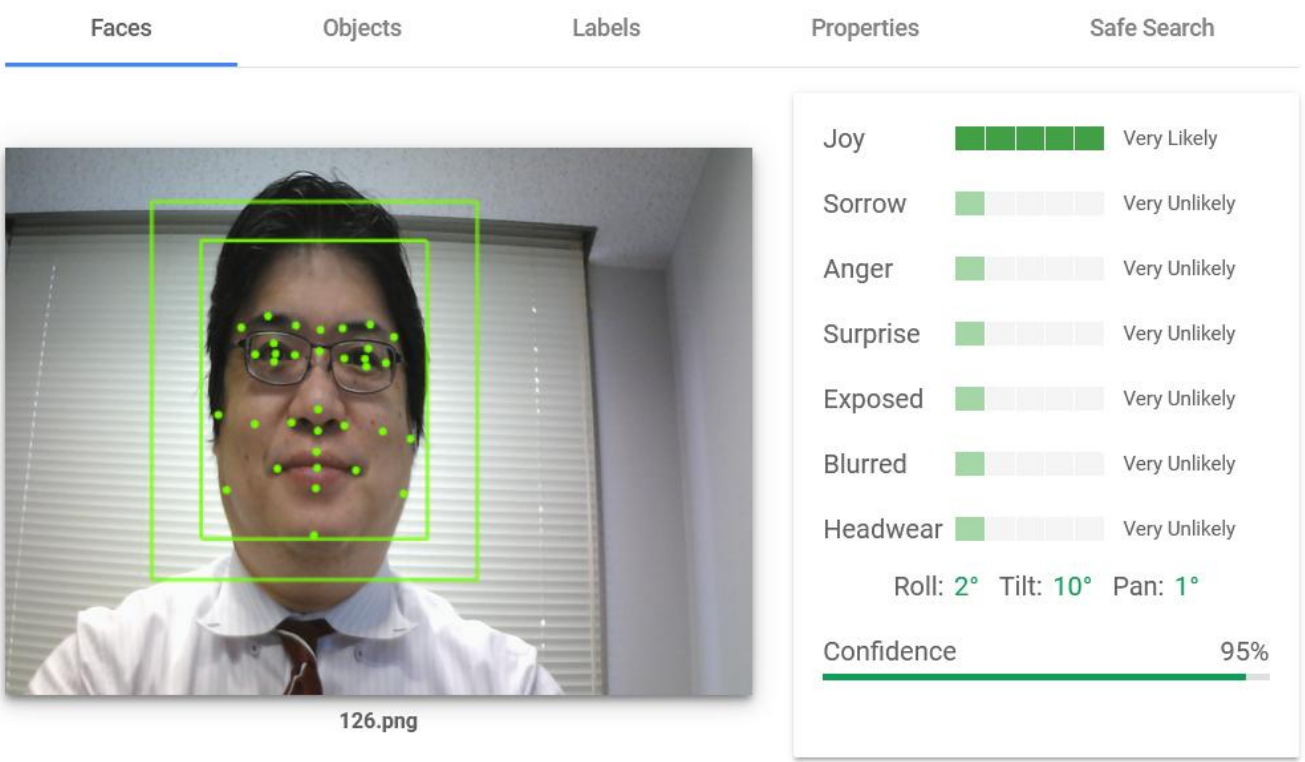
顔ランドマーク



３次元再構成

# キーポイントの抽出と，表情の読み取りを行うオンラインサービス

元画像




画像分類の結果

URL：https://cloud.google.com/vision/docs/drag-and-drop

# 顔情報処理の社会的論点（プライバシー・人権）

**顔情報処理は便利だが、いまはプライバシーと人権の尊重が前提になっている**

## 便益・安全

**SNSの顔自動タグ付け**
写真の中の顔を自動で特定

**職場での行動把握**
適切な監視で安全が高まる

**便利さより重視**

## プライバシー・人権

**タグ付け機能を廃止**
2021年のニュース

**尊重が前提**
監視はプライバシー・人権の尊重が条件

**便利さや安全だけでなく、プライバシーと人権の尊重が前提となる**

# 演習

# 演習１．CNN Explainer

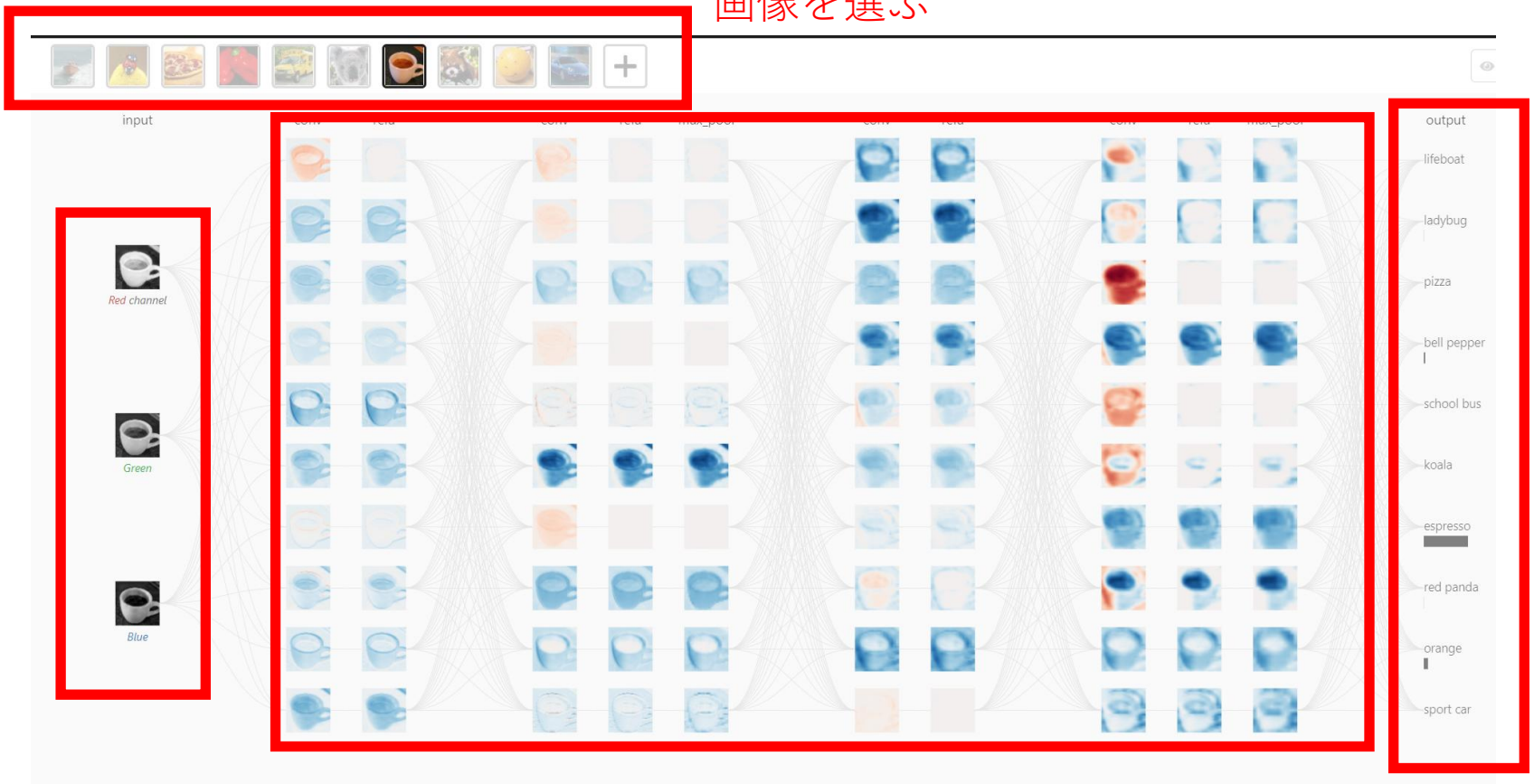
- **CNN Explainer**　ジョージア工科大学 Polo Club
- 畳み込み層などの仕組みをビジュアルに学ぶことができるサイト

**Webブラウザで次の URL を開く**

https://poloclub.github.io/cnn-explainer/

# CNN Explainerの基本操作

## 画像分類を行うニューラルネットワーク




元画像の
赤青緑の成分

ニューラルネットワークの可視化

画像の分類結果.
ここでは espresso

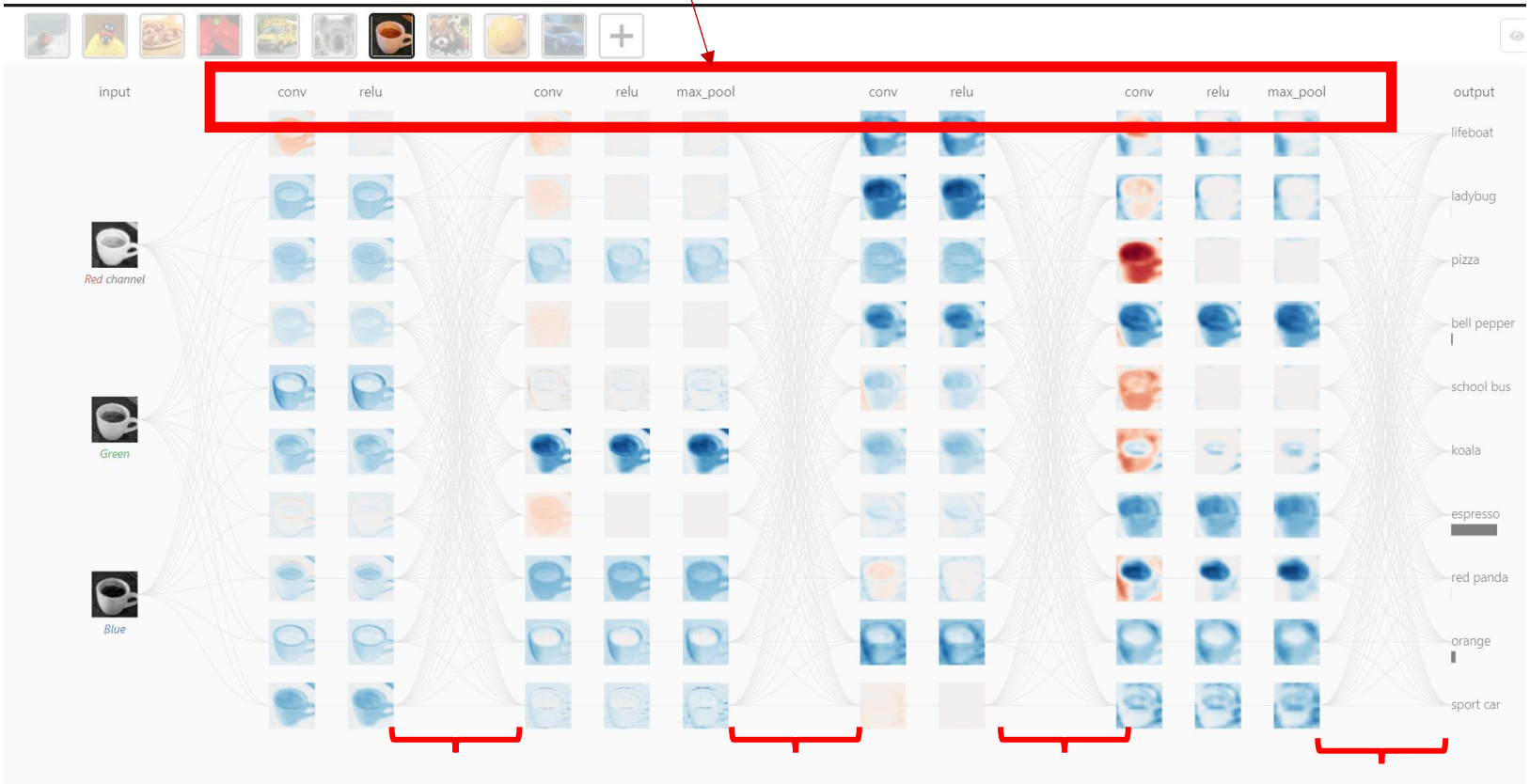
# ネットワーク構成の確認

**全結合層，畳み込み層，プーリング層**から構成される

conv　relu　　conv relu max_pool　conv relu　conv relu max_pool

畳み込み層　畳み込み層　プーリング層　畳み込み層 畳み込み層 プーリング層

**conv は畳み込み層で，max_pool はプーリング層**



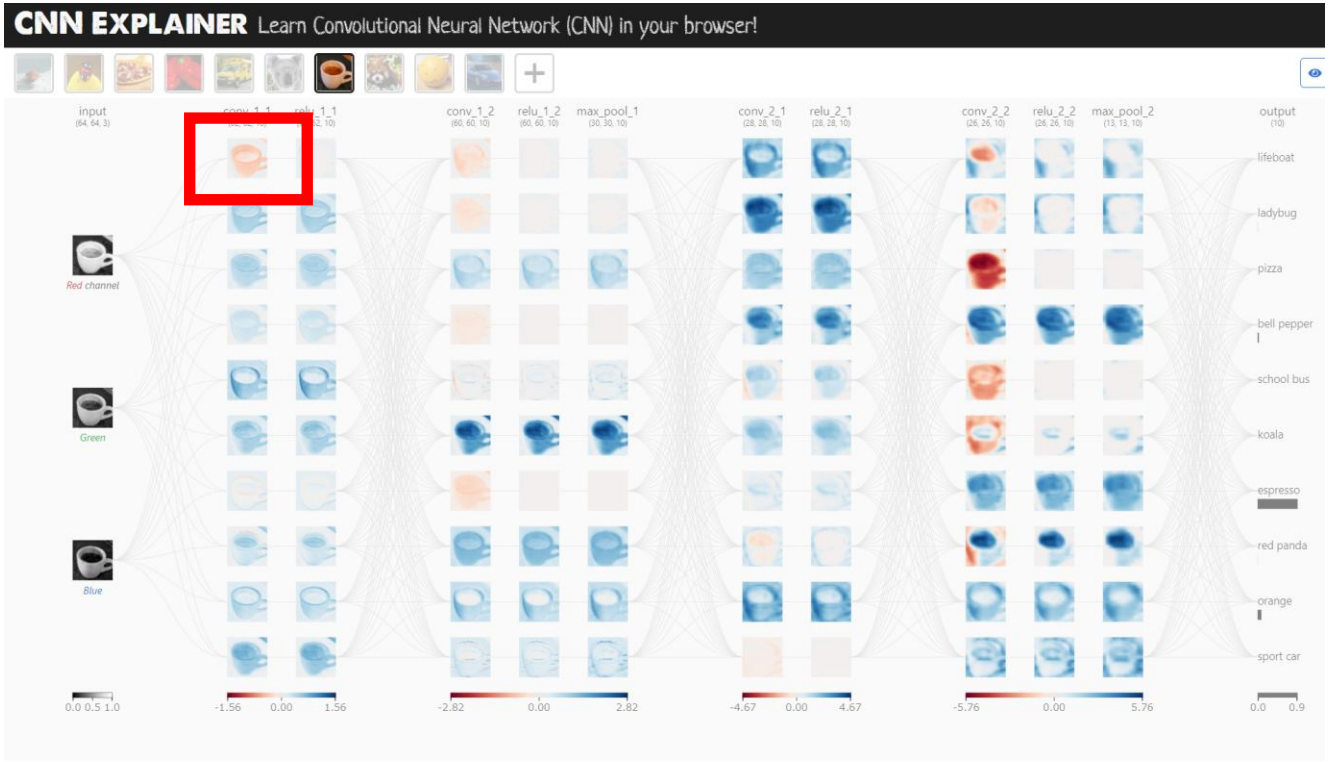

全結合層　全結合層　全結合層　全結合層

# 畳み込み処理の可視化

**左上の画像をクリック**

**→　畳み込みの様子をアニメーションで確認できる**

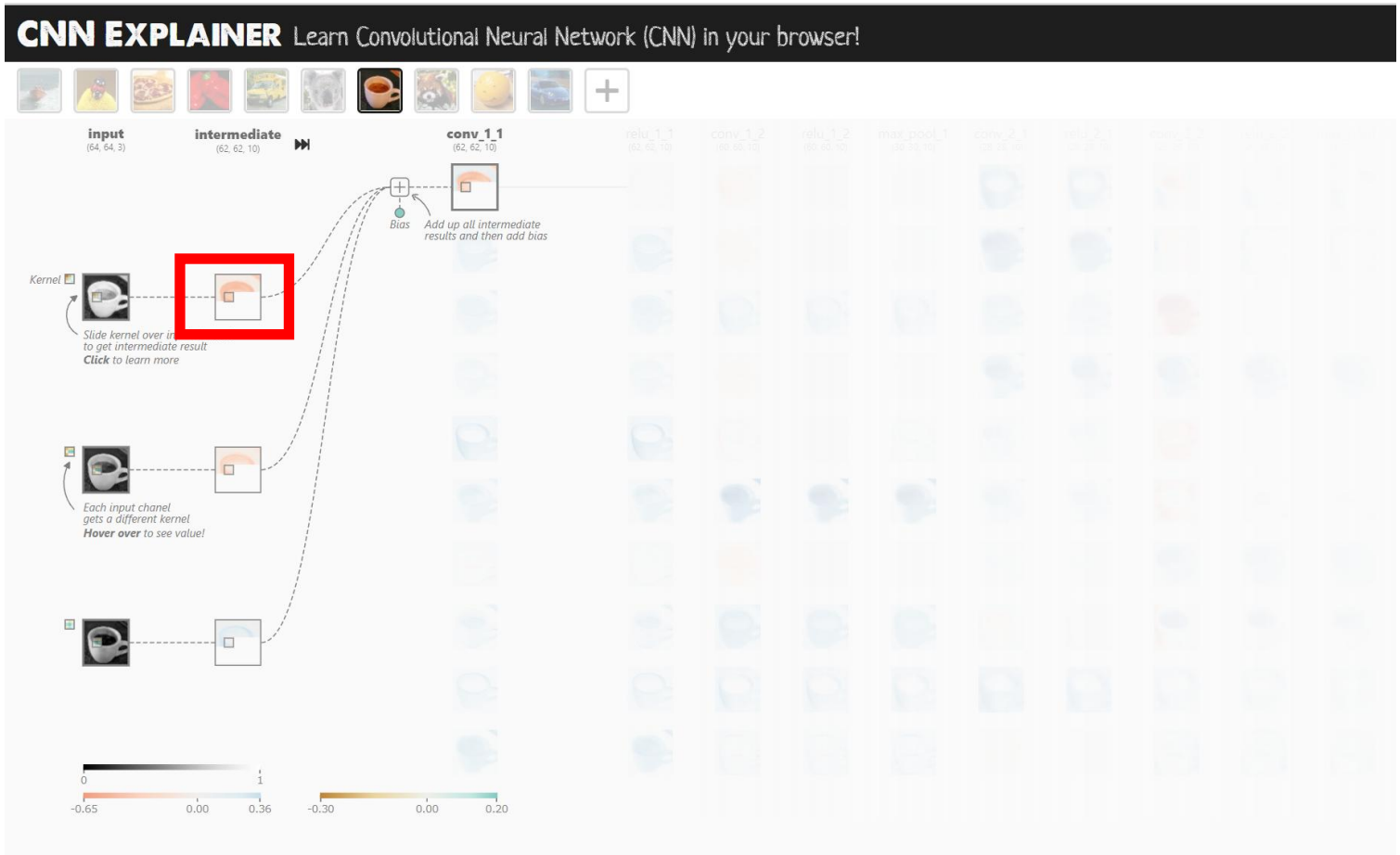
（この画像は，**各層での処理結果**である）



# 詳細な畳み込み過程

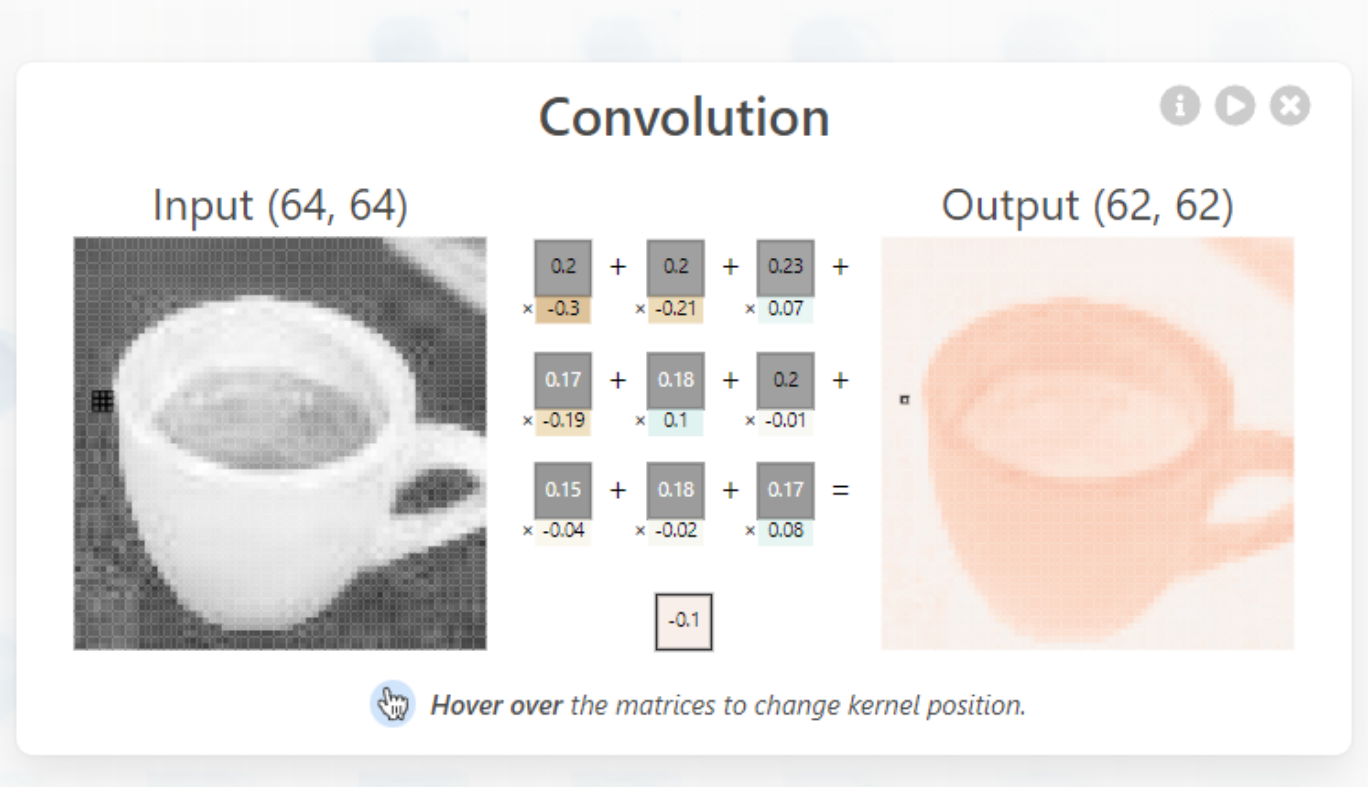
**出てきた画像をクリック**

**→　畳み込みの詳細をアニメーションで確認できる**



# アニメーションによる学習

畳み込みの様子がアニメーションで表示される




その他の層についてもビジュアルに表示できる（いろいろ試すことは，各自の自主的な自習とする）

# 演習２．Animated AI

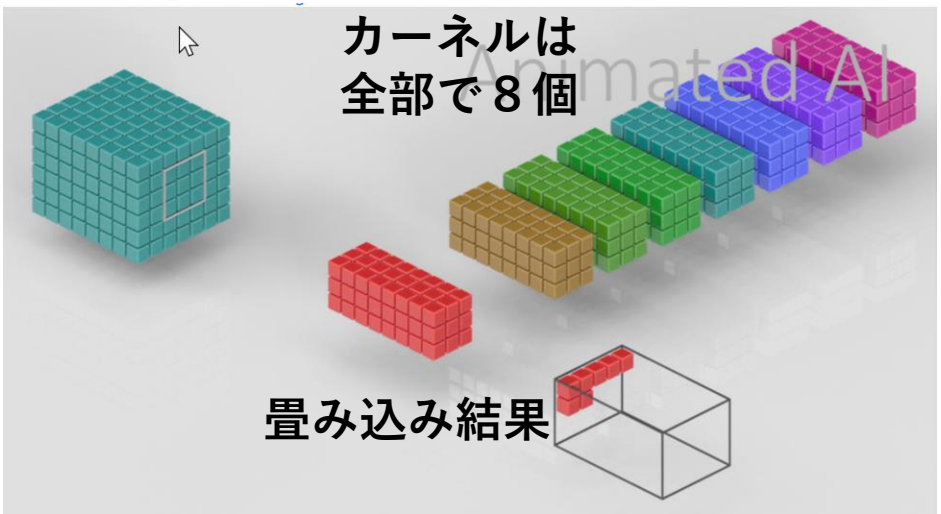
①「Animated AI」のサイトにアクセスする

https://animatedai.github.io/

②このサイトでは、さまざまな**ニューラルネットワーク**の**アルゴリズム**が解説されている

③トップの「**The Basic Algorithm**」は、**畳み込み**をアニメーションで示している。

- 元データとカーネルは３次元である
- **畳み込み**の仕組み上，畳み込み層の**同じ層のニューロン**はすべて，**同一のカーネルを共有**



# 演習3
# . YOLO による物体検出

① Google Colaboratory のページを開く

https://colab.research.google.com/drive/18IPPkY96Oc6jkYD2su4cFgWcoYAskLo_?usp=sharing

② プログラムや説明や実行結果が掲載されていることを確認。

各自でよく読む。

③「**5. 物体検出と YOLO**」を確認。

5. **物体検出と YOLO**

【概要】

使用モデル：YOLOv5（COCO データセットで学習済み）
特徴：YOLO（You Only Look Once）は、畳み込みニューラルネットワークを基盤としたリアルタイム物体検出手法であり、高速かつ高精度な特徴があります。