aa-8. 人工知能とコンピュータビジョン

（人工知能）

URL: https://www.kkaneko.jp/ai/mi/index.html

金子邦彦

① **コンピュータビジョン**と**ディープニューラルネットワーク**の概要

② **畳み込み**の仕組みと、**畳み込みニューラルネットワーク**の応用

③ **物体検出**の紹介

④ **顔情報処理技術**（顔検出、顔ランドマーク、顔のコード化、3D再構成）の説明とプライバシーへの配慮の必要性

アウトライン

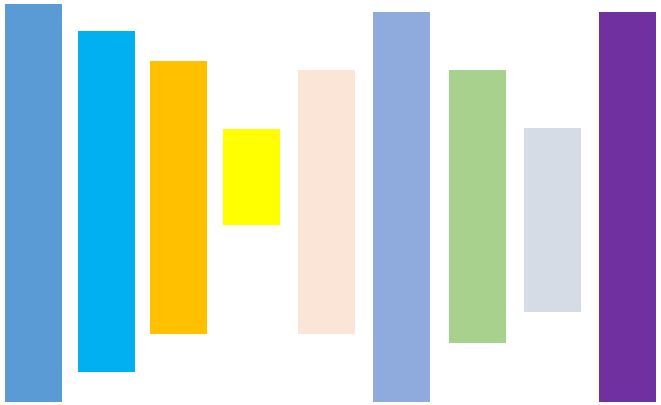
1. コンピュータビジョン
2. 画像を扱うための畳み込み
3. 畳み込み、プーリング、畳み込みニューラルネットワーク
4. 畳み込みニューラルネットワークの演習
5. 物体検出
6. 顔情報処理

ディープニューラルネットワーク

・**ディープニューラルネットワーク**は，

層が深い（**層の数が多い**）**ニューラルネットワーク**

層の数が少ない（浅い）



層の数が多い（深い）

ディープラーニングが広く利用されている理由

- **多様なデータに対応**

例：画像、テキスト、音声、動画など

- **応用の広さ**

例：画像認識、自然言語処理、音声認識など

- **パターン抽出能力**

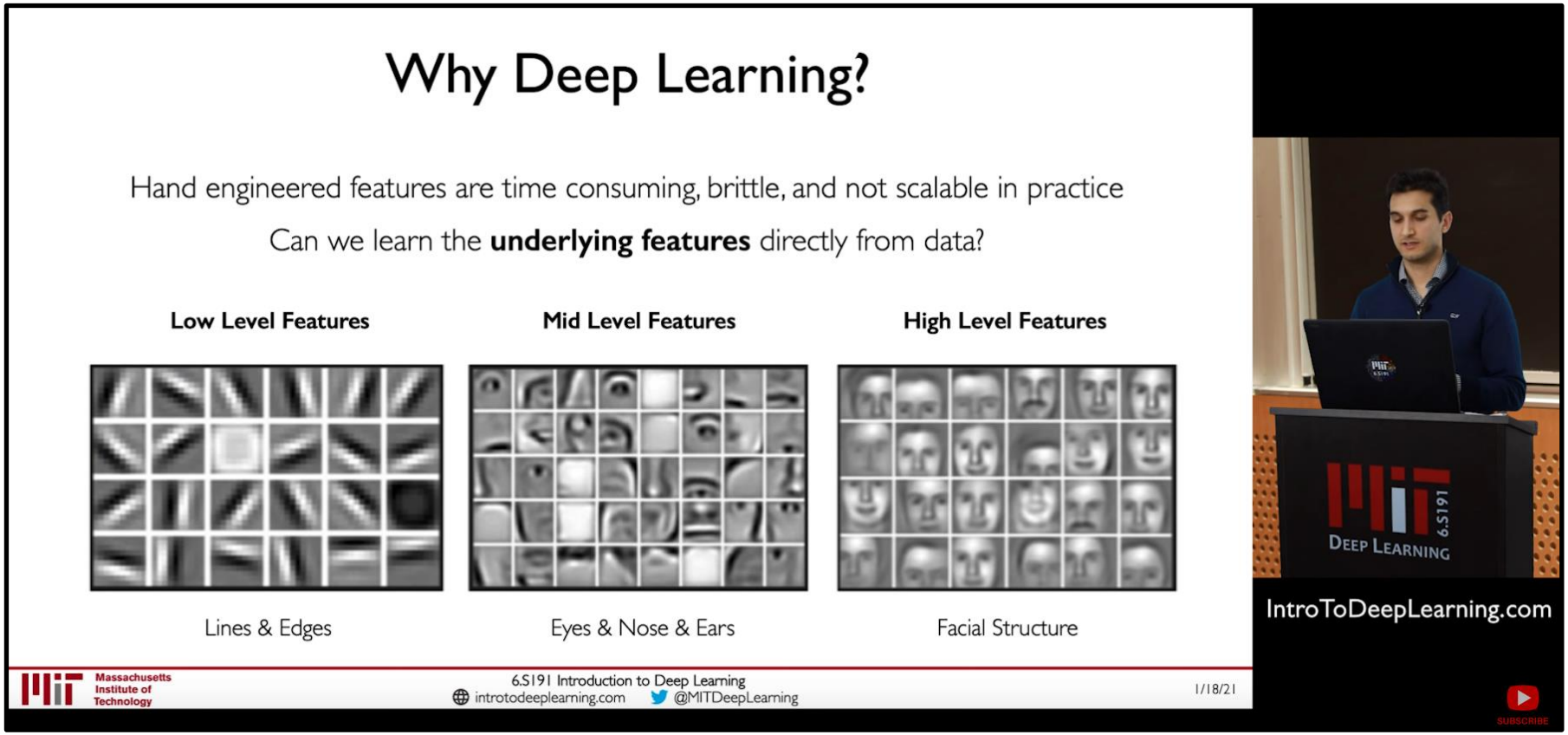
複雑なパターンを抽出する能力に優れる。

- **自動でのタスク実行**

パターン抽出ならびにタスク実行は自動で行われる。

ディープラーニングへの期待

さまざまなレベルのパターン



線や点のレベル　目，鼻，耳のレベル　顔の構造のレベル

MIT Introduction to Deep Learning | 6.S191,
https://www.youtube.com/watch?v=5tvmMX8r_OM
の「Why Deep Learning」のページ

8-1 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
- **実世界の様子**を理解し活用する（**何があり，何が発生し，何が起きそうかを予測する**など）

物体検出

インスタンス・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

コンピュータビジョンまとめ

- **コンピュータビジョン**：コンピュータが実世界の様子を理解し活用する
- **ディープニューラルネットワーク（DNN）：層が多いニューラルネットワーク**で、画像分類の精度向上に貢献。

8-2 画像を扱うための畳み込み

画像の畳み込みの応用例

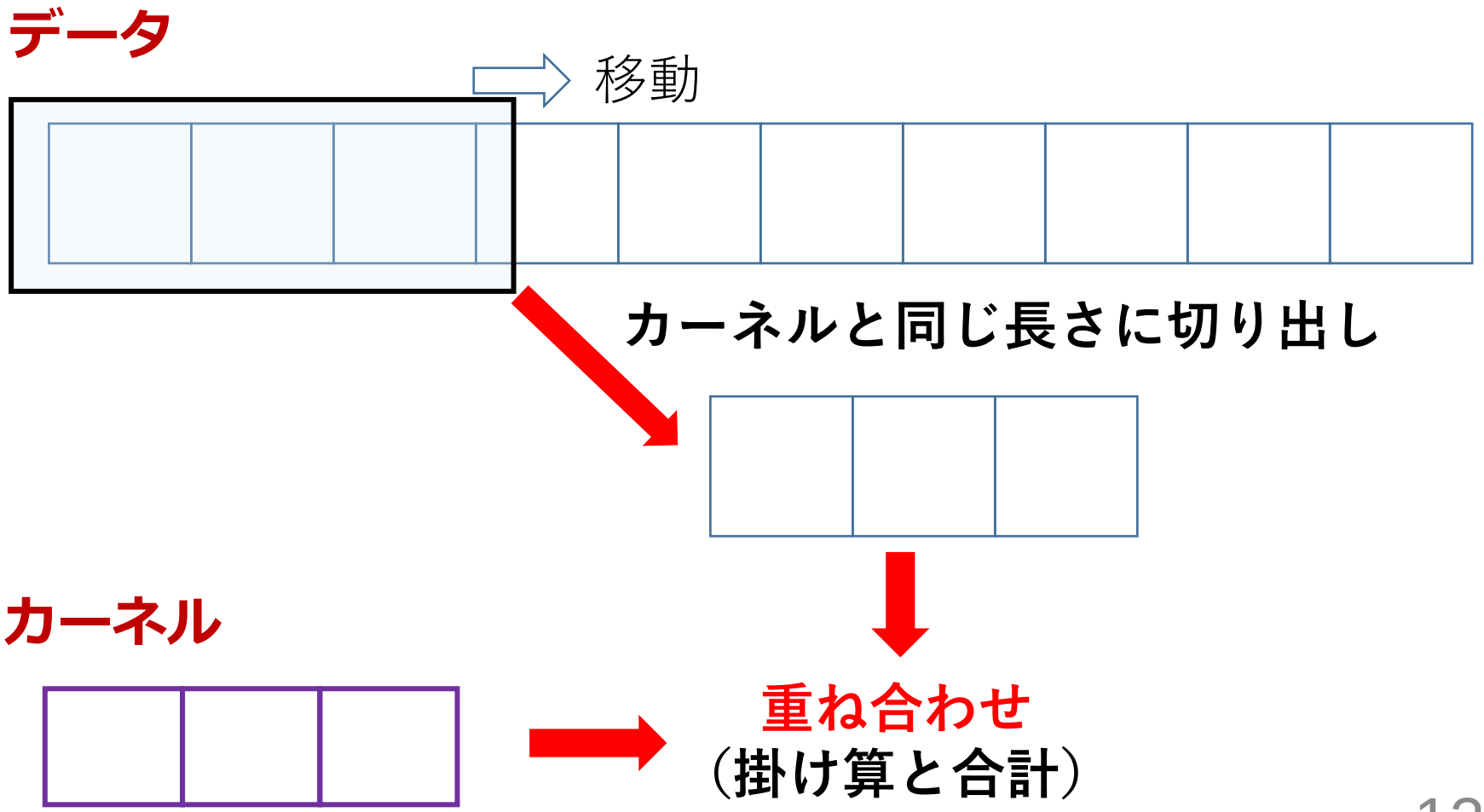
- 人工知能以外でも，ぼかし，エッジ抽出などさまざまな処理で，**畳み込み**を使用できる

畳み込みによる
ぼかし

畳み込みによる
エッジ抽出

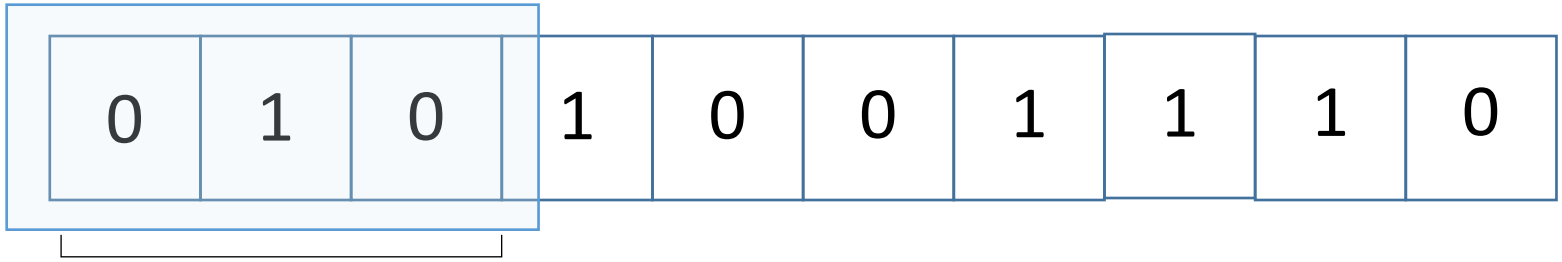
畳み込み

畳み込みは，あるデータを移動しながら，**カーネル**と重ね合わせる．重ね合わせの結果は１つの値になる．

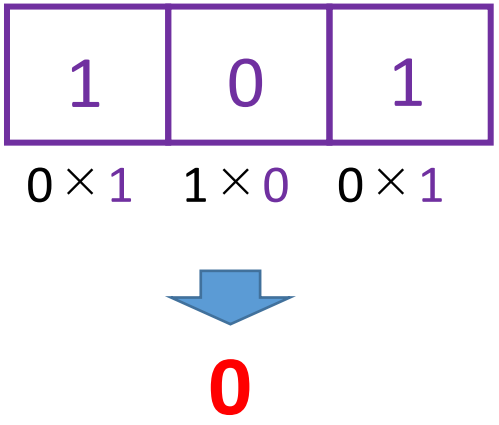


畳み込みの例

データ

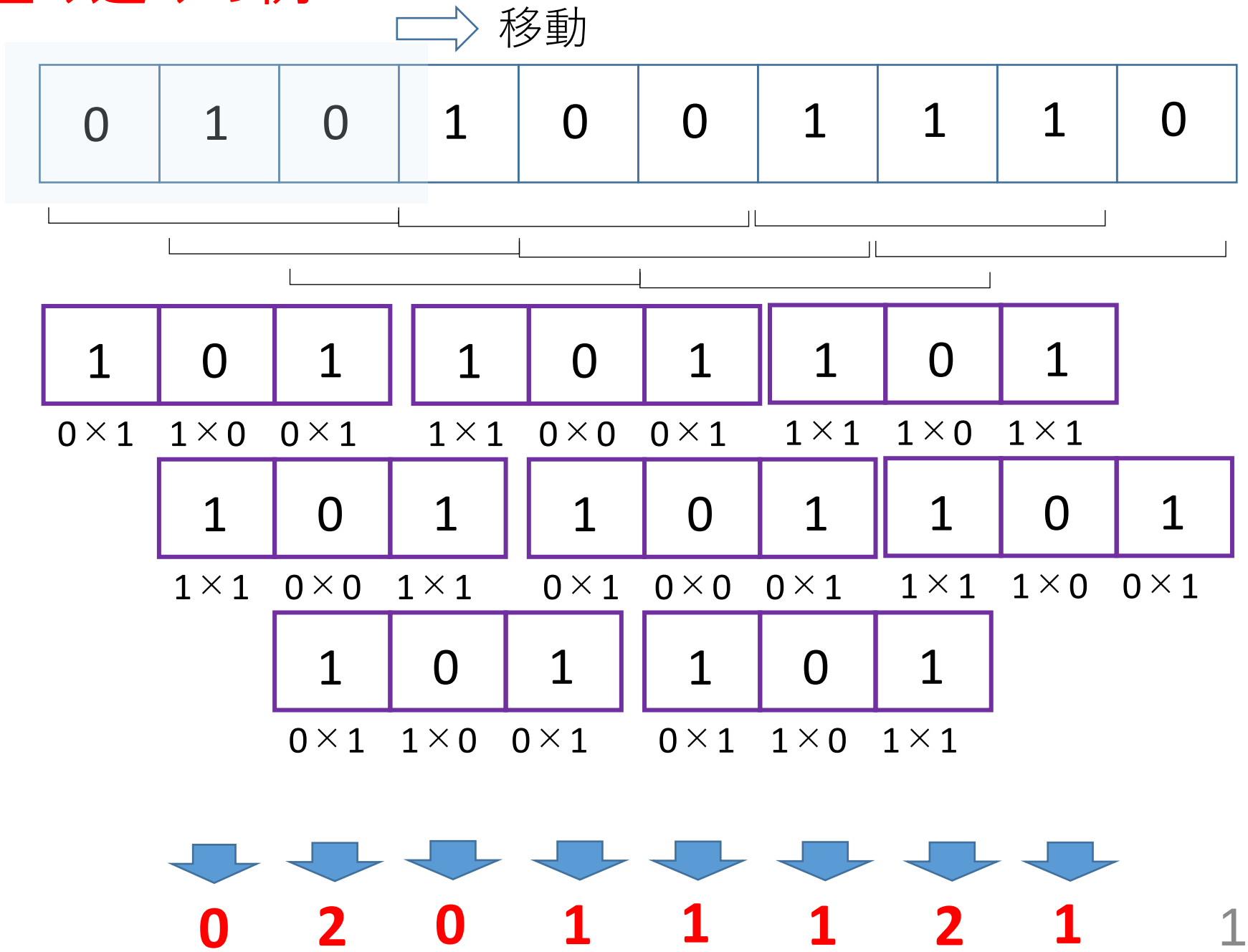


カーネル



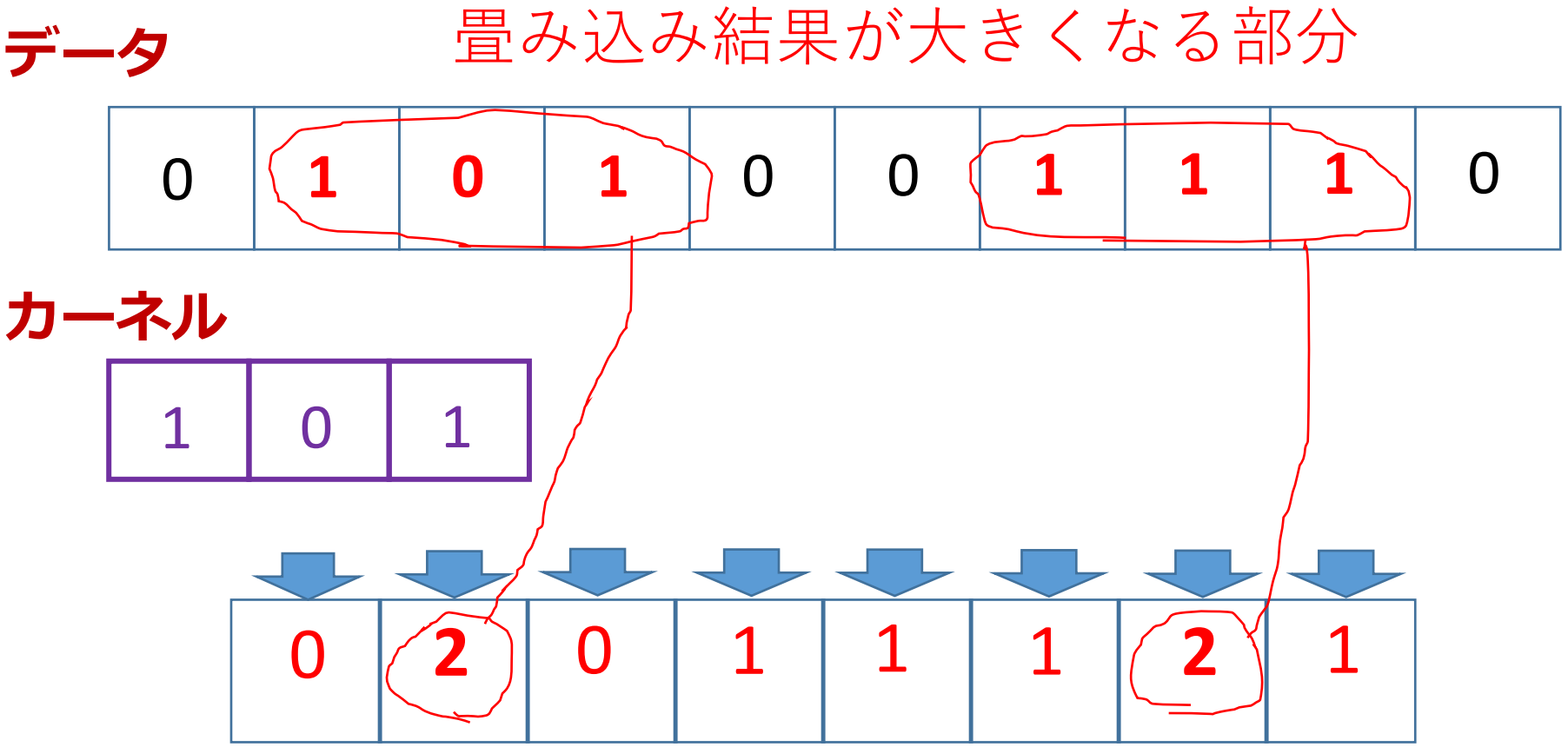
重ね合わせの結果：　0×1 + 1×0 + 0×1 = 0

畳み込みの例


移動
0 1 0 1 0 0 1 1 1 0
1 0 1
0×1 1×0 0×1
1 0 1
1×1 0×0 0×1
1 0 1
1×1 0×0 1×1
1 0 1
0×1 0×0 0×1
1 0 1
1×1 1×0 0×1
1 0 1
0×1 1×0 0×1
1 0 1
0×1 1×0 1×1
0 2 0 1 1 1 2 1

畳み込み

畳み込みは，「**特定のパターンに強く反応する**」働きがある
例：縦線を検出するための畳み込み＝縦線に強く反応



画像の畳み込み

Input

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）

Filter / Kernel

1	0	1
1	1	1
0	0	1

カーネル（3 × 3 マス）

出典: https://serokell.io/blog/introduction-to-convolutional-neural-networks

画像での畳み込み

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

画像での畳み込み

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）

0×1	1×0	1×1
0×1	1×1	1×1
0×0	1×0	1×1

合計: 4

1×1	1×0	0×1
1×1	1×1	0×1
1×0	1×0	0×1

合計: 3

4	3	

畳み込み結果

畳み込み結果

切り出し領域を横にずらす

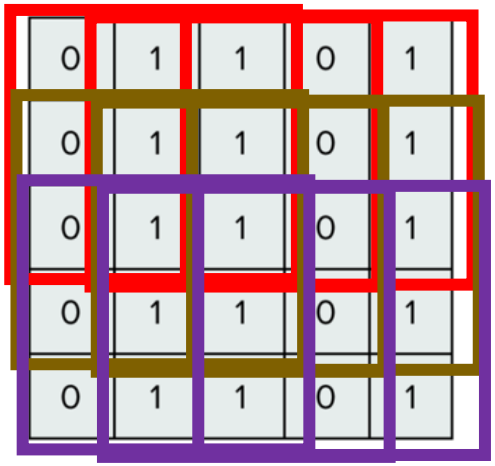
画像での畳み込み

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）



切り出し（３×３マス）

画像全体について
畳み込み

4	**3**	**5**
4	**3**	**5**
4	**3**	**5**

畳み込み結果

切り出し領域を縦横にずらす

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)
```

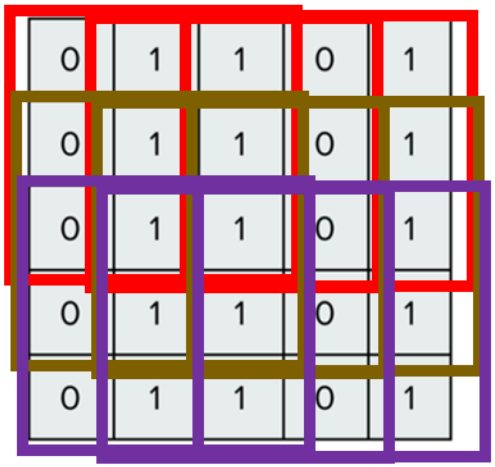
```
[[4, 3, 5], [4, 3, 5], [4, 3, 5]]
```

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

畳み込みまとめ

畳み込み:　データに対し移動しながらカーネルと重ね合わせ、その結果を１つの値にまとめる。

- 畳み込みは，「**特定のパターンに強く反応する**」働きがある
- コンピュータビジョンに欠かせない技術
- 人工知能以外でも使用可能。例えば、画像のぼかしやエッジ抽出などに活用。



切り出し（３×３マス）

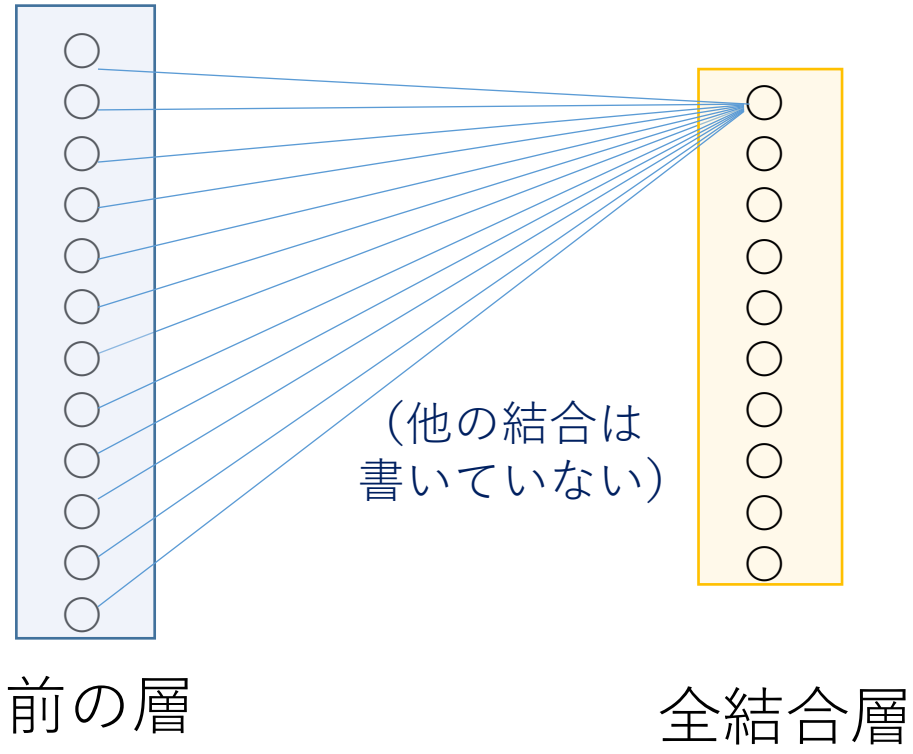
画像全体について
畳み込み

4	3	5
4	3	5
4	3	5

8-3. 畳み込み、プーリング、畳み込みニューラルネットワーク

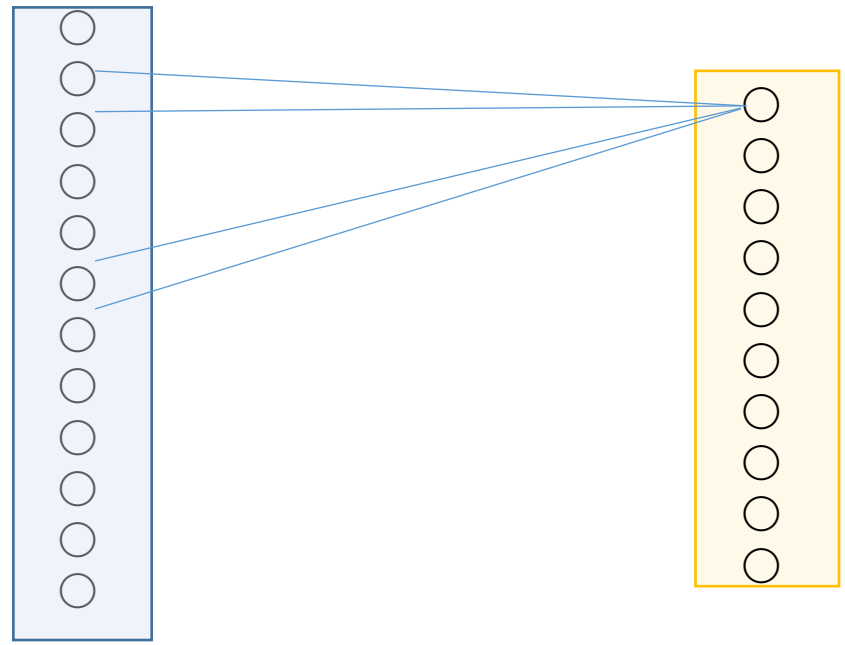
全結合層

- **全結合層**の**ユニット**は，**前の層**の**すべてのユニットと結合**している



畳み込み層

- **畳み込み層**は，**畳み込み**を行う　0×1 + 1×0 + 0×1 = 0
- **結合の重み**が，**畳み込み**の**カーネル**になる
- 前の層の，一部分のユニットとのみ結合

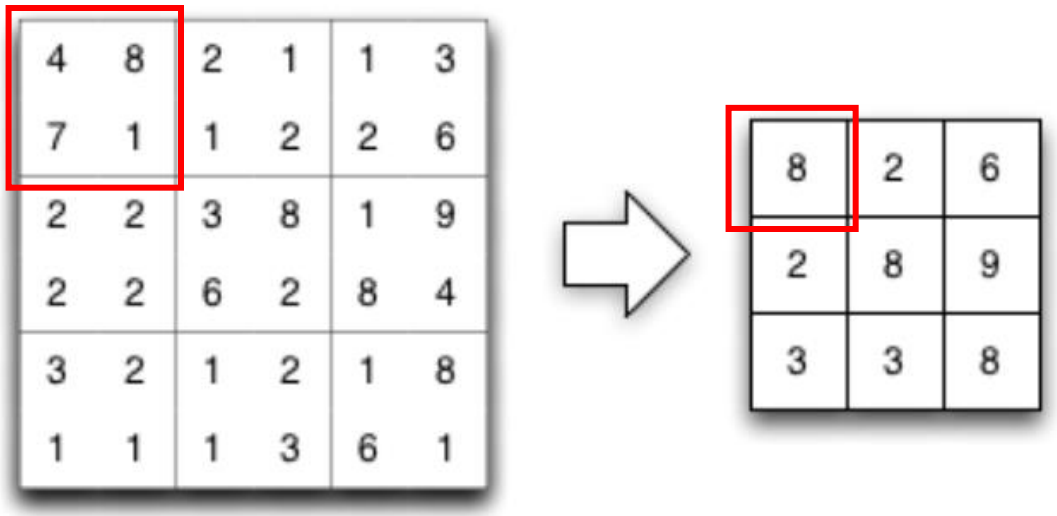


前の層　　畳み込み層

畳み込みのカーネルのサイズが 2×2 のとき：
前の層の 4つのユニットとのみ結合

プーリング

- **プーリングにより，画像サイズが小さくなる**
- **マックス・プーリング**は，プーリングの一種で，**最大値を利用するプーリング**

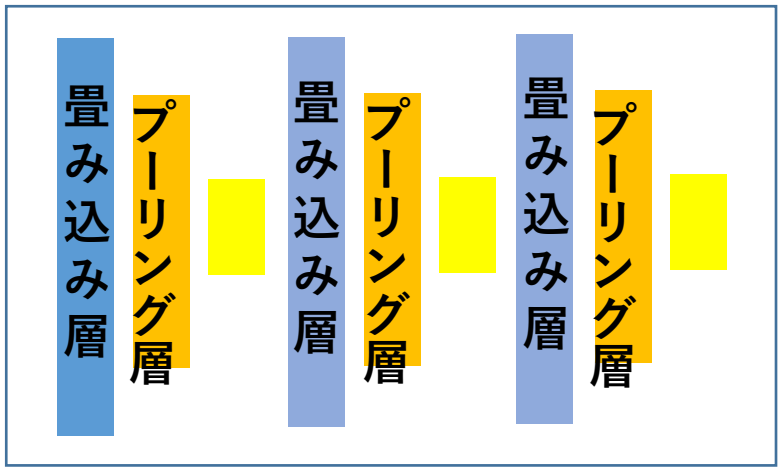


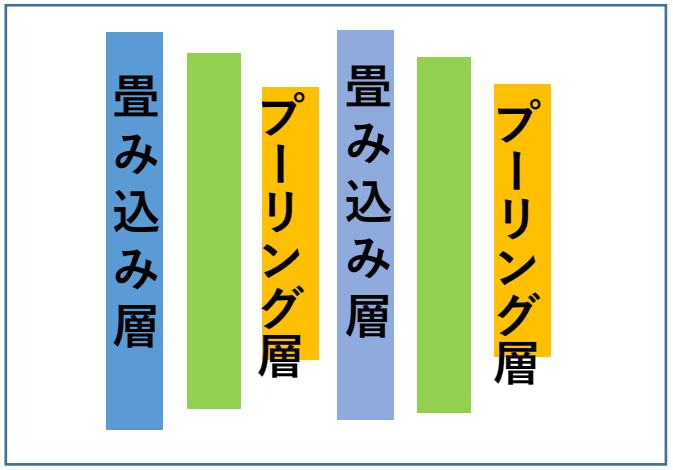
・4, 8, 7, 1 の最大値は 4
・「**4, 8, 7, 1**」**の 4マス**から，**最大値の 8 を選ぶ**.
すると，サイズは ¼ になる

出典: https://github.com/jeffheaton/t81_558_deep_learning/blob/084023876b6cf09c931b452584dbd44c56314a03/t81_558_class_06_2_cnn.ipynb

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は，**畳み込み層**と**プーリング層**を交互に繰り返した後，**最終的に全結合層などを経て出力層**に接続





さまざまなバリエーション

- **畳み込み層**　・・・　畳み込みによるパターンの識別
- **プーリング層**　・・・　画像の小移動に対して，出力が不変になる

畳み込みニューラルネットワークの特徴

全結合層のみの場合と比べて.

- ユニット間の結合を局所に限定
- 結合の数を，大幅に削減
- **結合**の数が減り，**過学習**の問題を緩和

畳み込みニューラルネットワーク（CNN）の用途

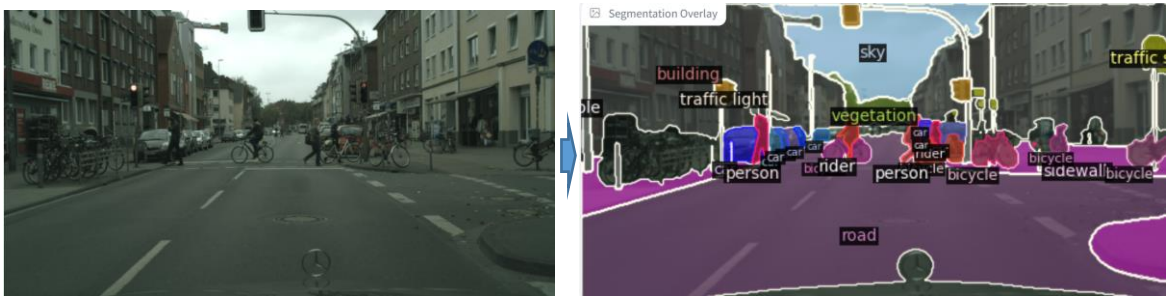
画像分類，物体検出，セグメンテーションなどで高い性能・機能を発揮する場合がある

画像分類

Score 0.9827020168304443, Label lab_coat
Score 0.0030872616916894913, Label syringe
Score 0.0024311079178005457, Label beaker
Score 0.0016609227750450373, Label stethoscope
Score 0.0003795088559854757, Label plate

物体検出

セグメンテーション



演習　畳み込みニューラルネットワークのデモ

【トピックス】

- **畳み込みニューラルネットワーク**
- **CNN Explainer**
- **畳み込み層**
- **プーリング層**

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/

① 画面の確認

このニューラルネットワークは，**画像分類**を行う

元画像の
赤青緑の成分

ニューラルネットワーク

画像の分類結果.
ここでは espresso

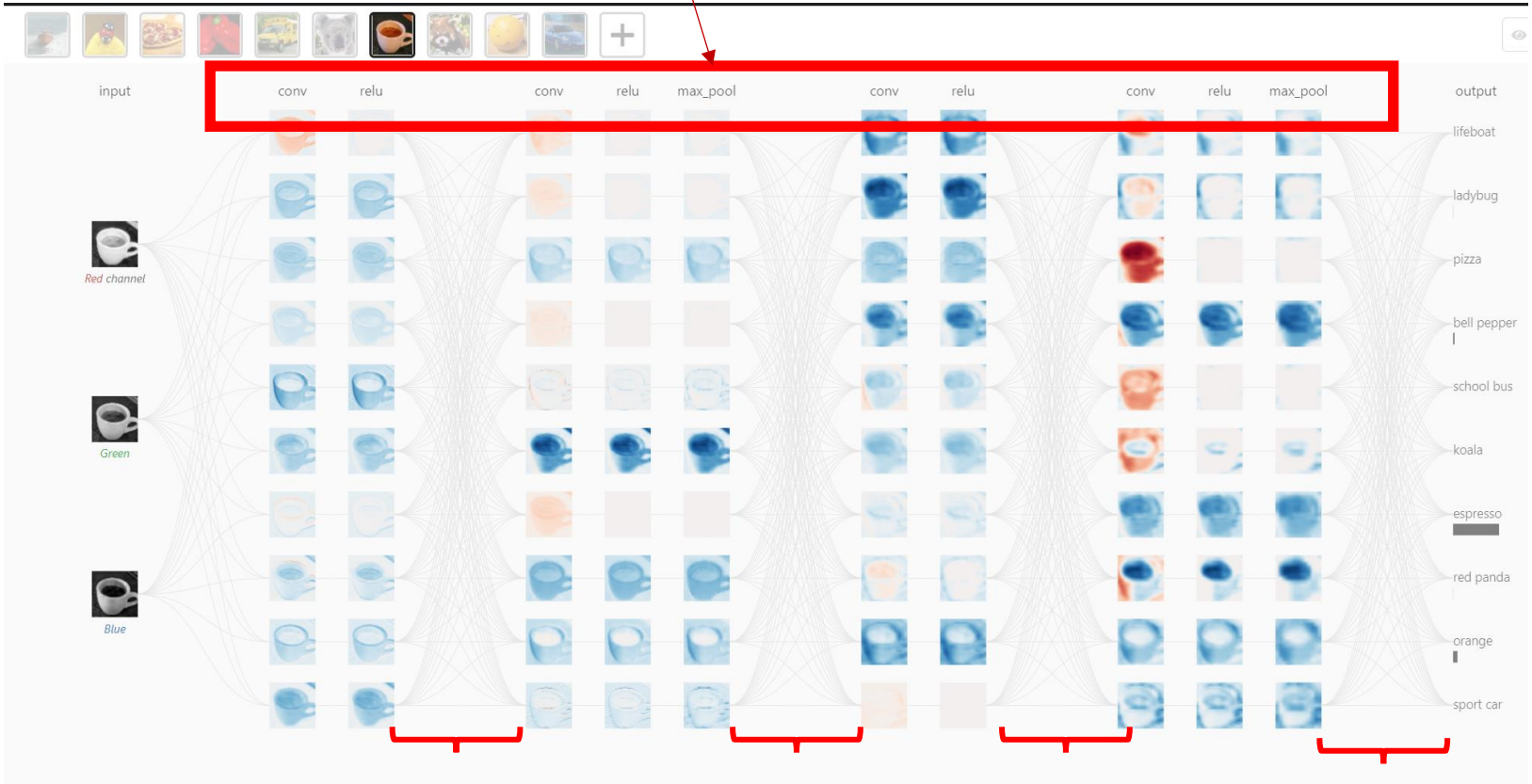
② ニューラルネットワークの構成

畳み込み層と**プーリング層**を含む

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

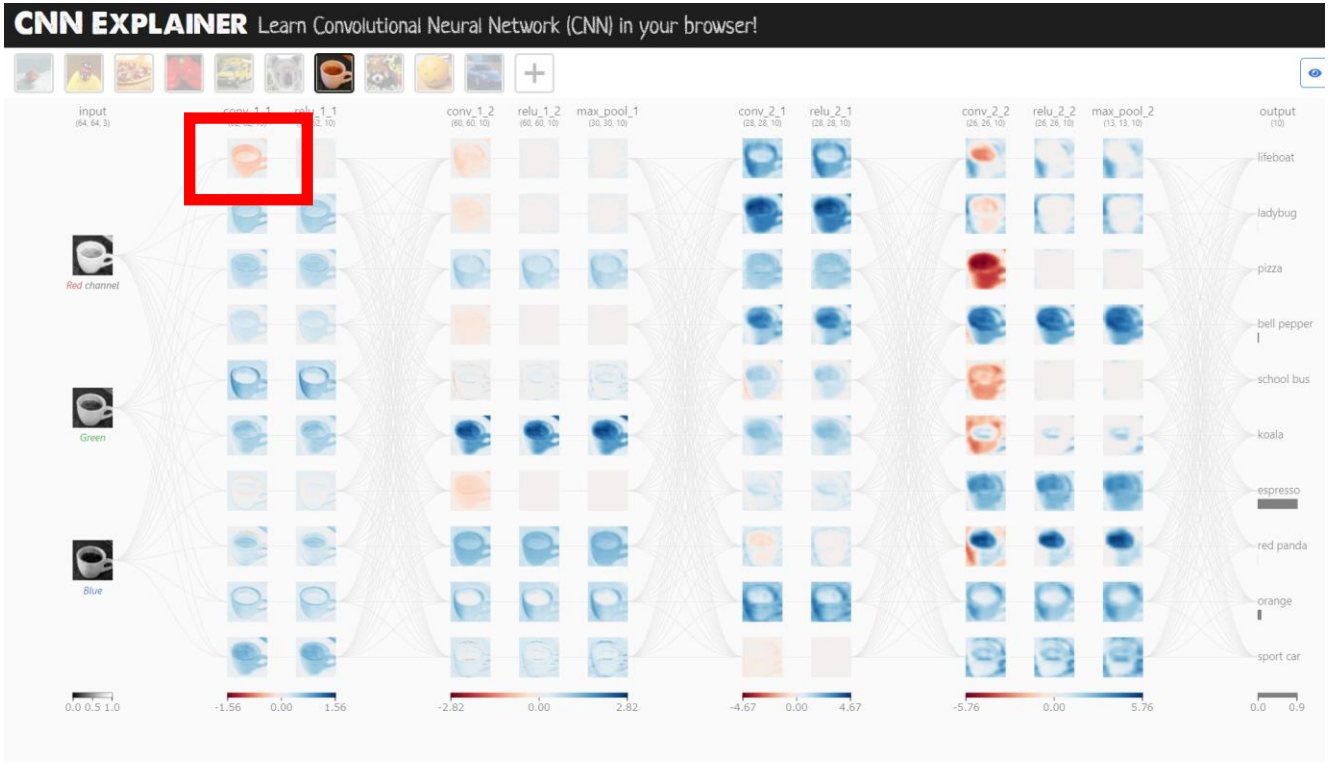


全結合層　全結合層　全結合層　全結合層

③ 左上の画像をクリック

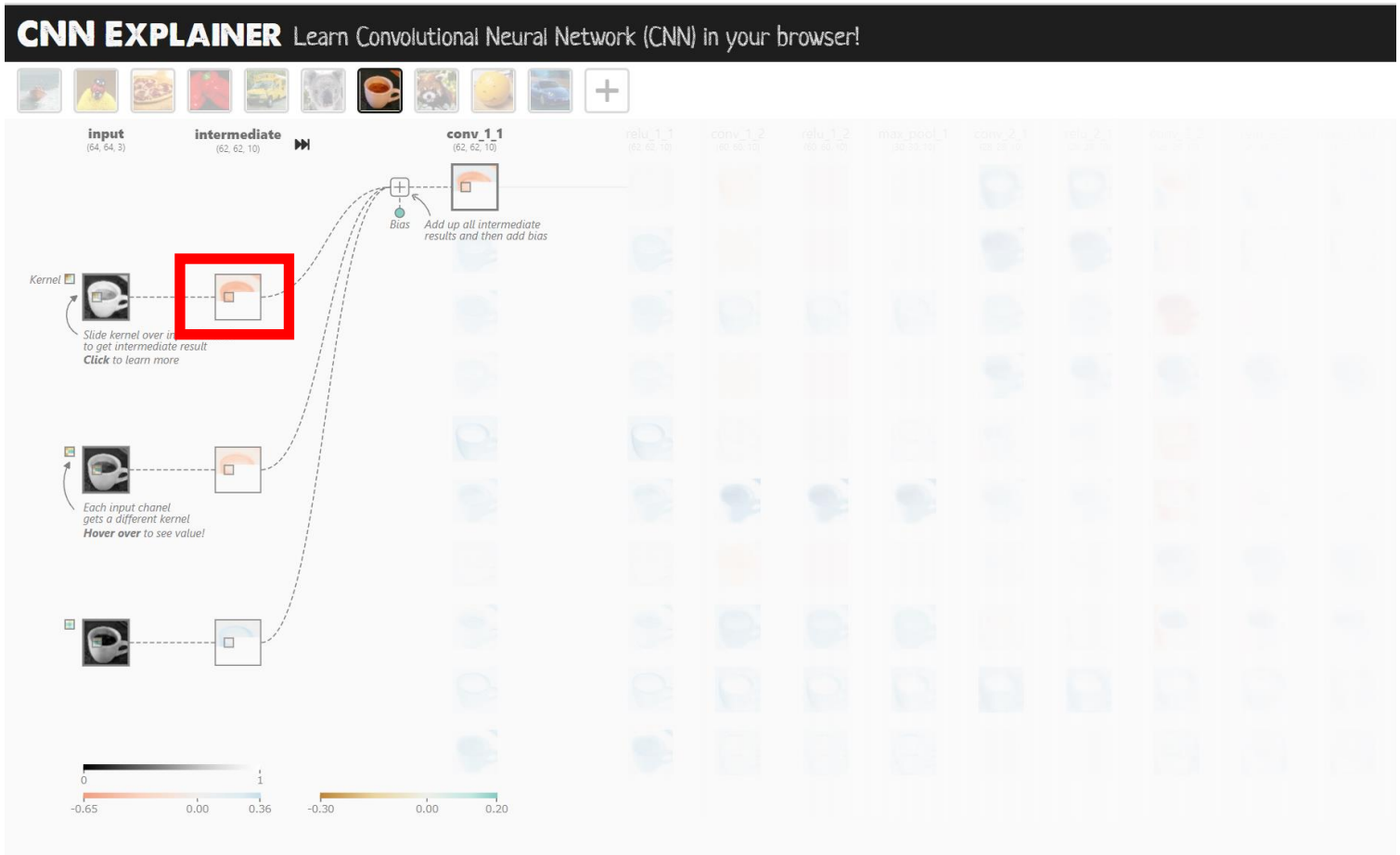
→　畳み込みの様子をアニメーションで確認できる

（この**画像**は，**各層での処理結果**である．画像１個がニューロン１つというわけではない）

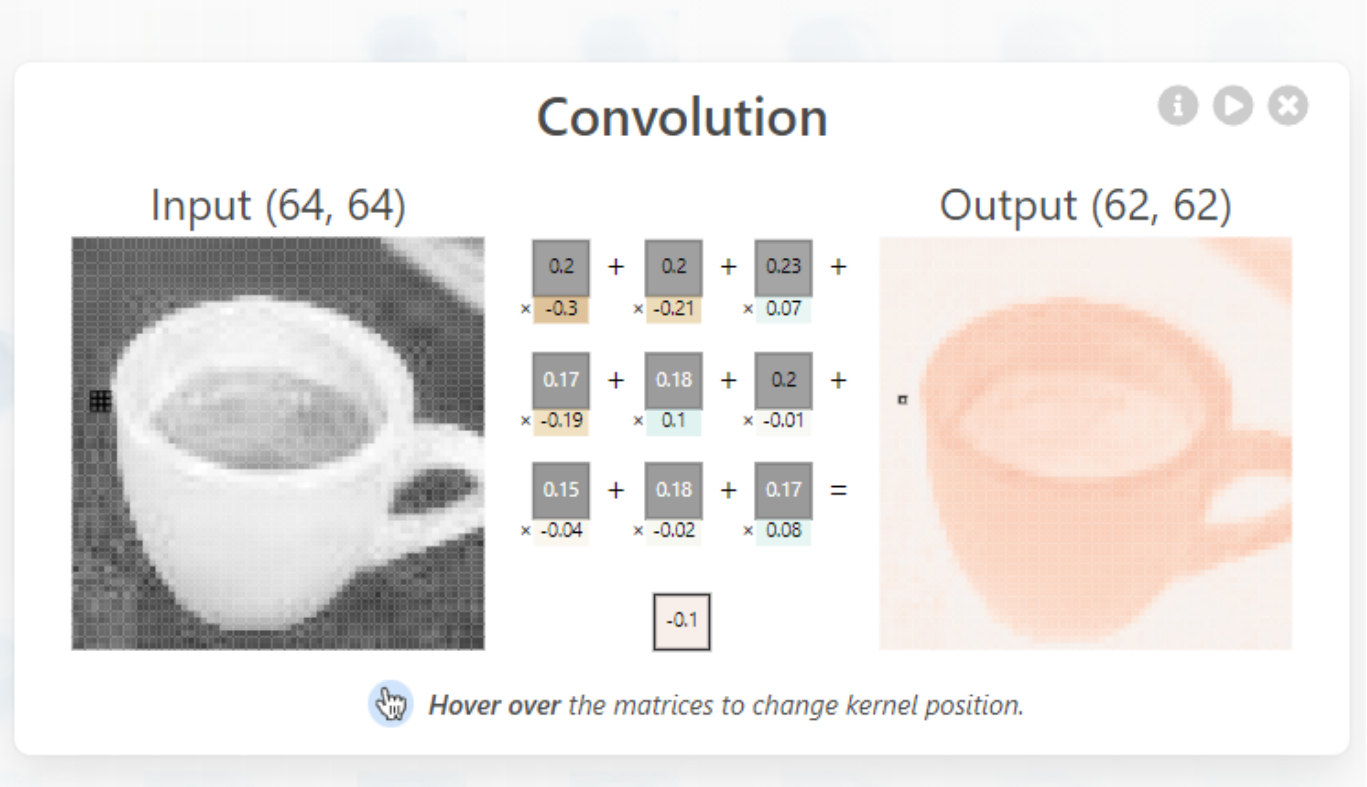


④ 出てきた画像をクリック

→　畳み込みの詳細をアニメーションで確認できる



⑤ 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）

畳み込みニューラルネットワークまとめ

- **全結合層**：前の層の**全てのユニットと結合**
- **畳み込み層**：前の層の一部のユニットと結合し、**畳み込みを行う**
- **プーリング**：**画像サイズを小さくする**。マックスプーリングは最大値を利用。
- **畳み込みニューラルネットワーク**：**畳み込み層**と**プーリング層**を含むニューラルネットワーク。
  - ユニット間の**結合を局所に限定**し、結合の数を大幅に削減して**過学習を緩和**する。
  - **画像分類、物体検出、セグメンテーション**などで高性能を発揮。

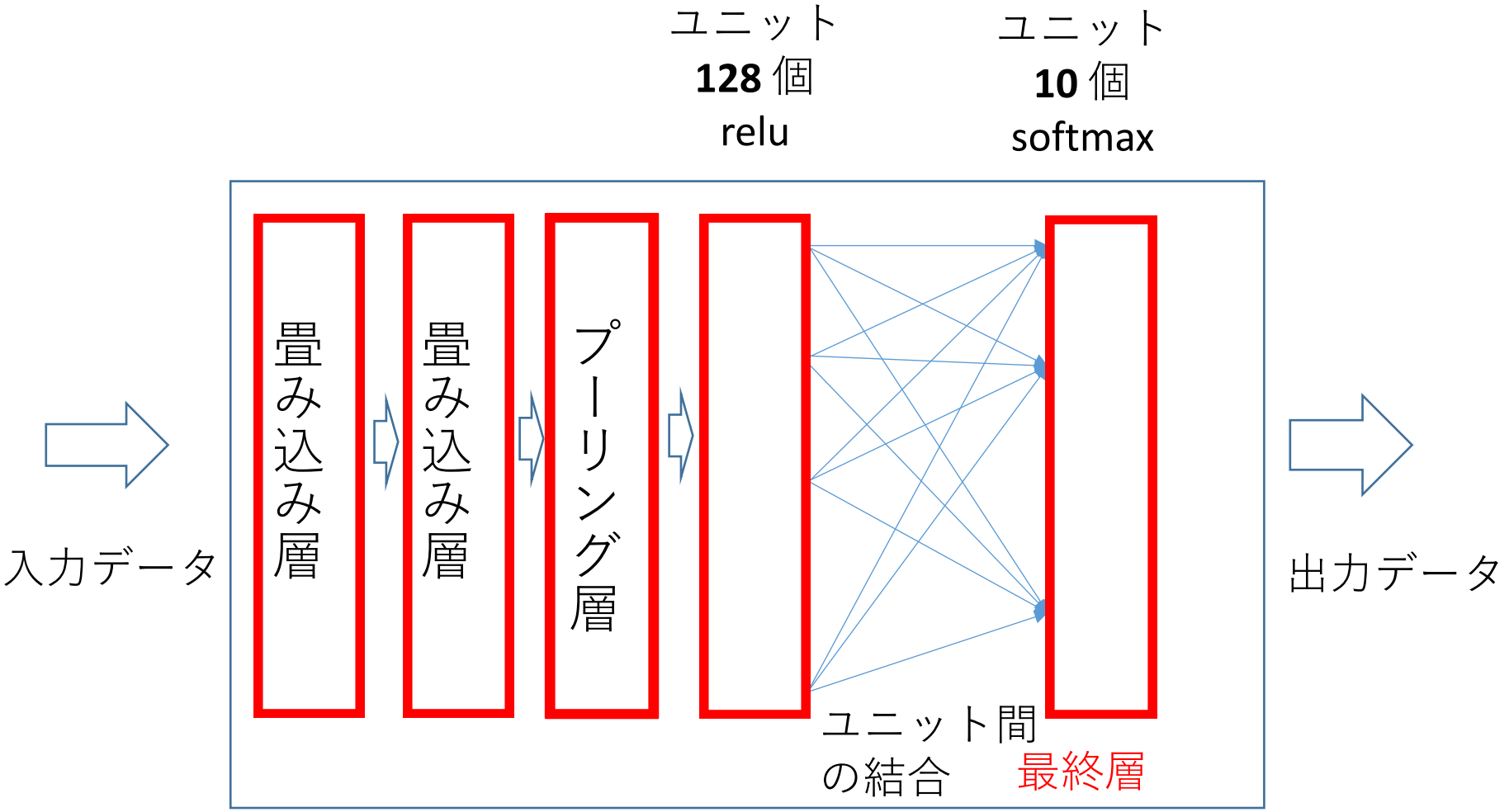
8-4. 畳み込みニューラルネットワークの演習

次のページを使用

- **畳み込みニューラルネットワーク**を用いた画像分類
- 学習と検証を行うプログラム
- 学習曲線をプロット

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

畳み込み層を含むニューラルネットワークの例



ニューラルネットワーク作成のプログラム例

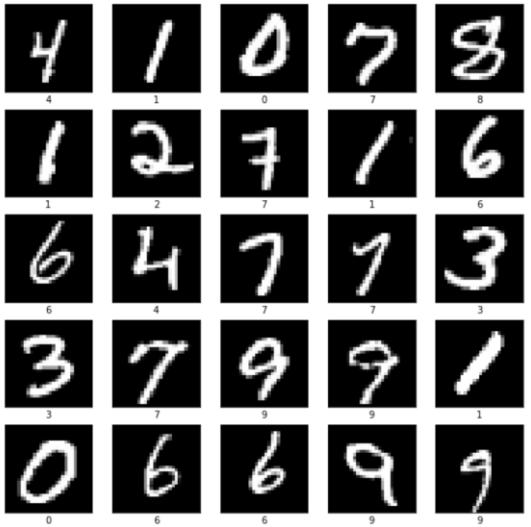
```
import tensorflow as tf
m = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3),          ←畳み込み層
                           activation='relu',
                           input_shape=(28, 28, 1)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),  ←畳み込み層
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),         ←プーリング層
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=128, activation='relu'),    ←全結合層
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(units=10, activation='softmax')←全結合層
])
```

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

学習のための準備

- **ニューラルネットワーク**の作成
- **訓練データ**　（学習用）

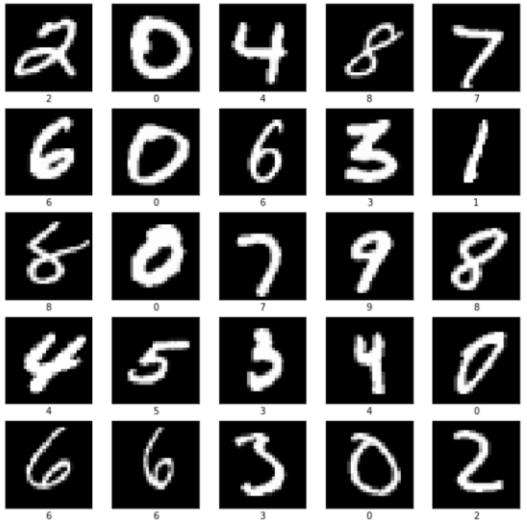
60000枚の画像と正解



抜粋

- **検証データ**　（検証用）

10000枚の画像と正解



抜粋

学習の繰り返しを行うプログラム例

学習の繰り返し回数は **10**

```
EPOCHS=10
history = m.fit(x=x_train,
          y=y_train,
          epochs=EPOCHS,
          validation_data=(x_test, y_test),
          callbacks=[tensorboard_callback],
          verbose=2)
```

訓練データの指定

検証データの指定

学習の繰り返しを行うプログラム例

学習の繰り返し回数は **10**

```
EPOCHS=10
history = m.fit(x=x_train,
          y=y_train,
          epochs=EPOCHS,
          validation_data=(x_test, y_test),
          callbacks=[tensorboard_callback],
          verbose=2)
```

訓練データの指定

検証データの指定

学習の繰り返しを行うプログラムと実行結果

同じ訓練データを用いた学習を１０回繰り返し.
そのとき，検証データで検証

```
EPOCHS=10
history = m.fit(x=x_train,
          y=y_train,
          epochs=EPOCHS,
          validation_data=(x_test, y_test),
          callbacks=[tensorboard_callback],
          verbose=2)
```

プログラム

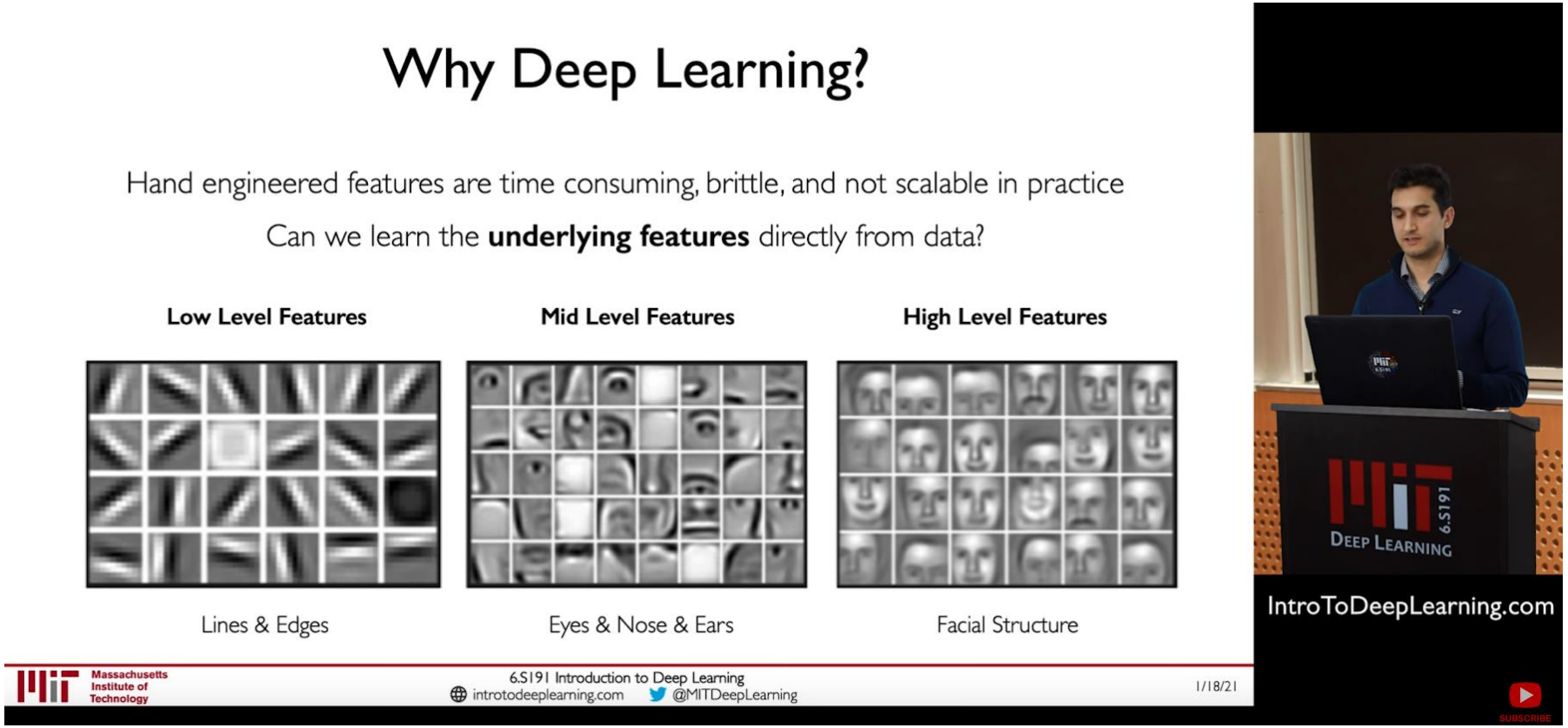
```
Epoch 1/10
1875/1875 - 147s - loss: 0.2847 - accuracy: 0.9110 - val_loss: 0.0606 - val_accuracy: 0.9804 - 147s/epoch - 79ms/step
Epoch 2/10
1875/1875 - 143s - loss: 0.1025 - accuracy: 0.9696 - val_loss: 0.0447 - val_accuracy: 0.9860 - 143s/epoch - 76ms/step
Epoch 3/10
1875/1875 - 143s - loss: 0.0770 - accuracy: 0.9771 - val_loss: 0.0392 - val_accuracy: 0.9871 - 143s/epoch - 76ms/step
Epoch 4/10
1875/1875 - 149s - loss: 0.0619 - accuracy: 0.9808 - val_loss: 0.0373 - val_accuracy: 0.9873 - 149s/epoch - 80ms/step
Epoch 5/10
1875/1875 - 150s - loss: 0.0546 - accuracy: 0.9836 - val_loss: 0.0354 - val_accuracy: 0.9892 - 150s/epoch - 80ms/step
Epoch 6/10
1875/1875 - 147s - loss: 0.0476 - accuracy: 0.9854 - val_loss: 0.0309 - val_accuracy: 0.9903 - 147s/epoch - 78ms/step
Epoch 7/10
1875/1875 - 151s - loss: 0.0430 - accuracy: 0.9864 - val_loss: 0.0329 - val_accuracy: 0.9894 - 151s/epoch - 80ms/step
Epoch 8/10
1875/1875 - 147s - loss: 0.0389 - accuracy: 0.9879 - val_loss: 0.0290 - val_accuracy: 0.9907 - 147s/epoch - 78ms/step
Epoch 9/10
1875/1875 - 148s - loss: 0.0361 - accuracy: 0.9881 - val_loss: 0.0286 - val_accuracy: 0.9904 - 148s/epoch - 79ms/step
Epoch 10/10
1875/1875 - 150s - loss: 0.0322 - accuracy: 0.9897 - val_loss: 0.0312 - val_accuracy: 0.9904 - 150s/epoch - 80ms/step
```

実行結果

画像分類の精度は 0.99

学習の**繰り返し**ごとに，
訓練データや検証データ
での**精度**や**損失**
の変化を確認

うまく設計されたニューラルネットワークは，「データからのパターンの抽出」を行っているという考え方も



ニューラルネットワークが扱うさまざまなレベルのパターン

MIT Introduction to Deep Learning | 6.S191,
https://www.youtube.com/watch?v=5tvmMX8r_OM
の「Why Deep Learning」のページ

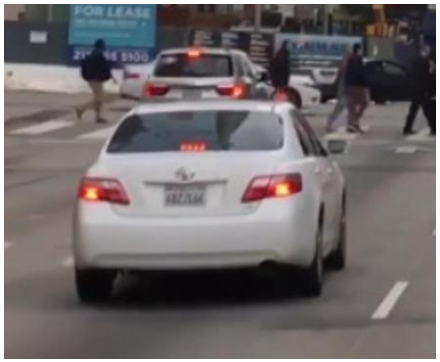
8-5 物体検出

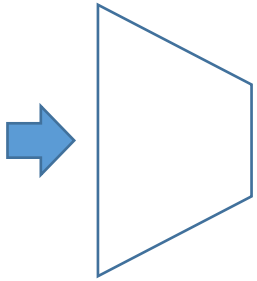
物体検出


umbrella|0.64
potted plant|0.41
potted plant|0.63
potted plant|0.50
potted plant|0.53
chair|0.53
chair|0.99
chair|0.98
chair|0.72
potted plant|0.58
dining table|0.35
dining table|0.40
chair|1.00
potted plant|0.66
potted plant|0.57
potted plant|0.36
dining table|0.45
chair|0.93
chair|0.93
chair|0.96
dining table|0.91
chair|0.88

物体検出の仕組み

場所の特定　　**画像分類**



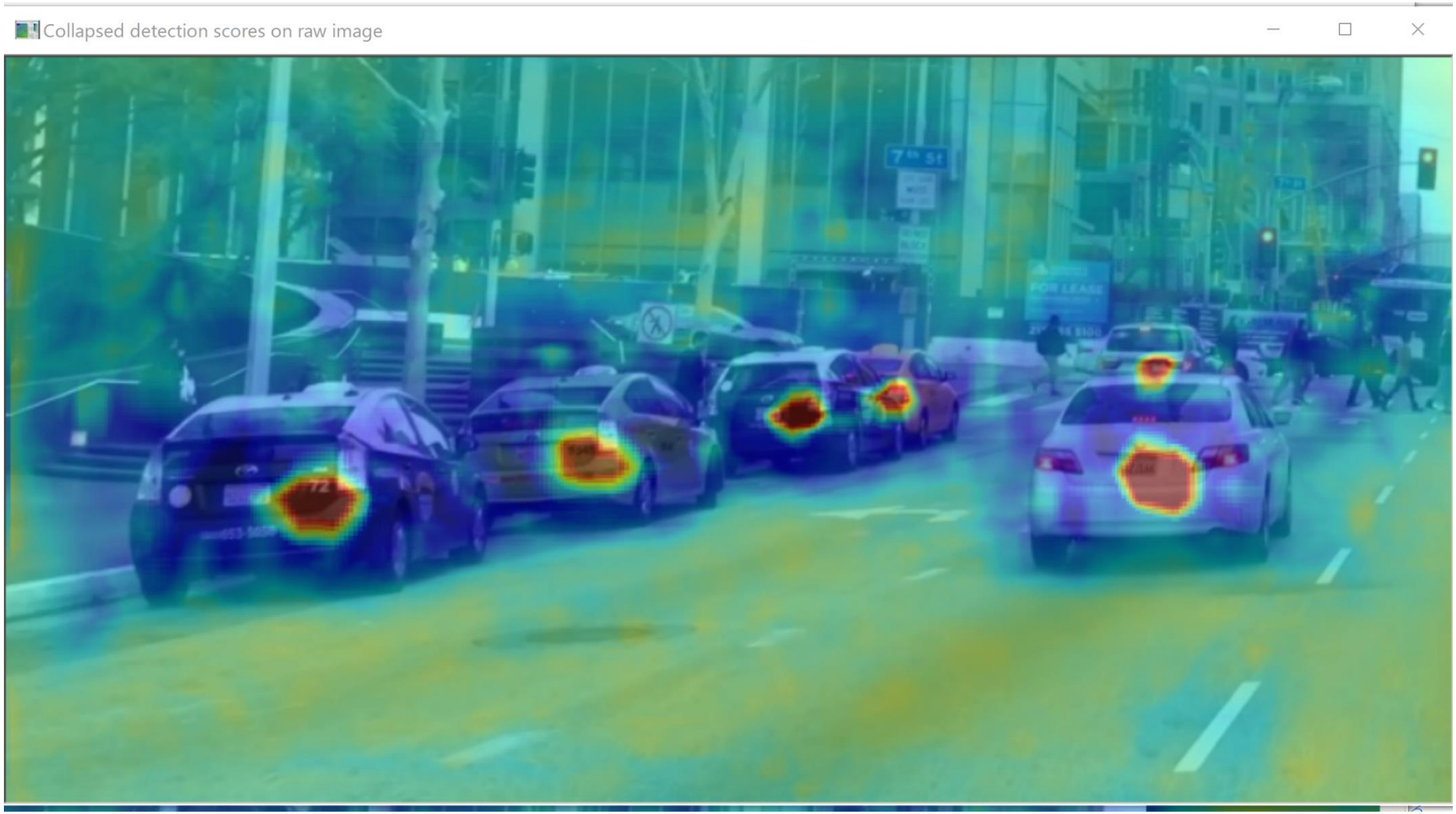


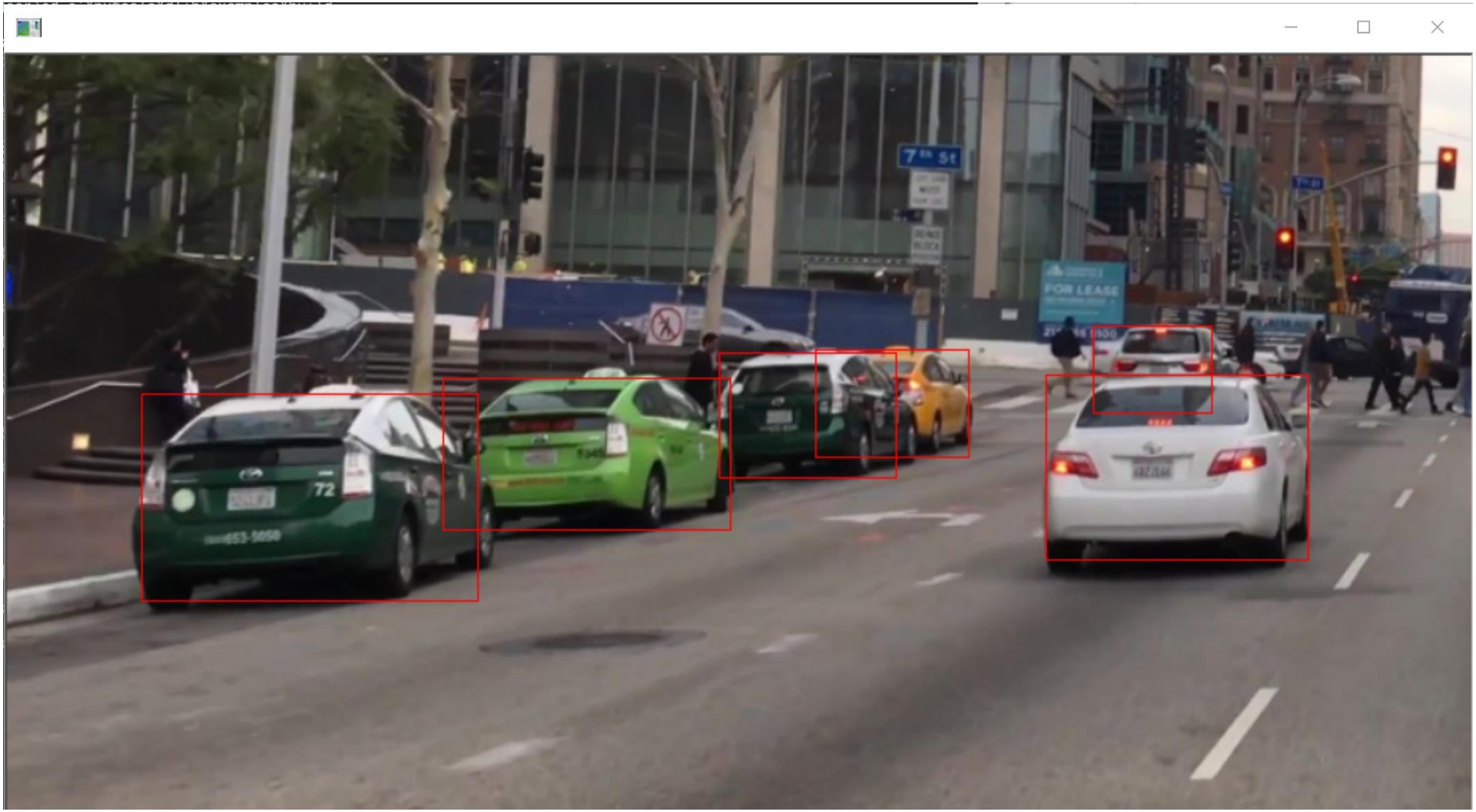
畳み込み，
プーリング，
全結合

taxi　　種類

画像全体の中で，
物体がありそうな
場所を特定
（**特徴マップ**）

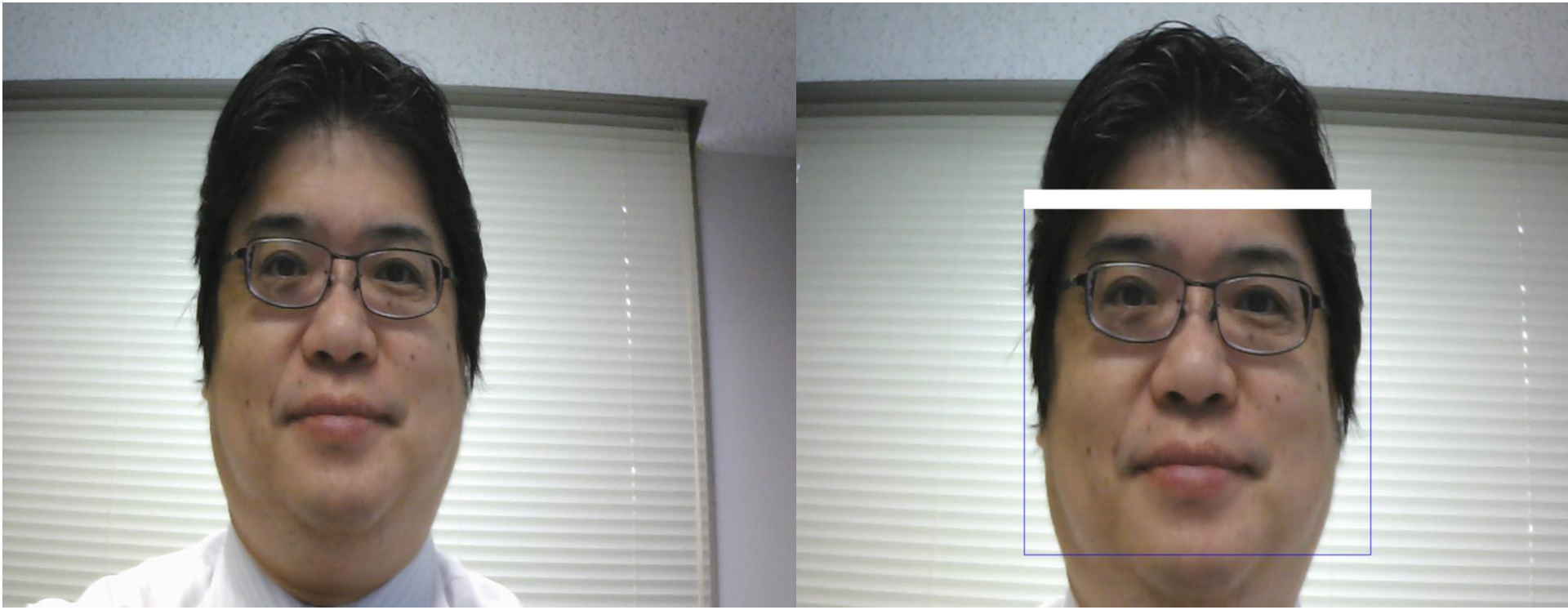
場所と大きさ
を表すバウンディングボックス


Collapsed detection scores on raw image


7TH ST
FOR LEASE

8-6 顔情報処理

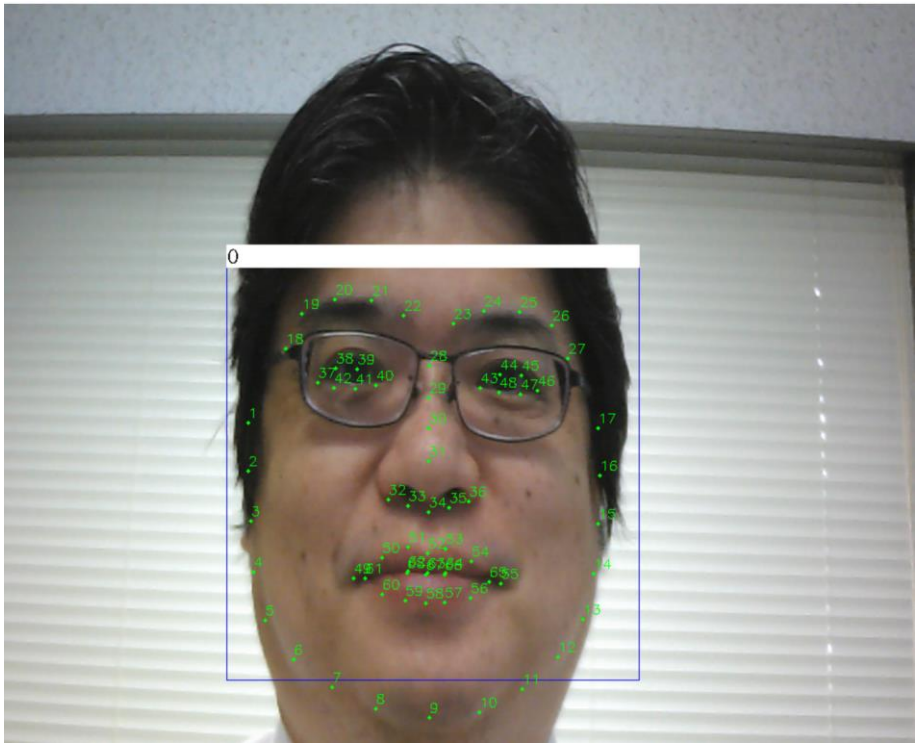
顔検出



- 用途：顔の数のカウント，顔情報処理の最初の処理

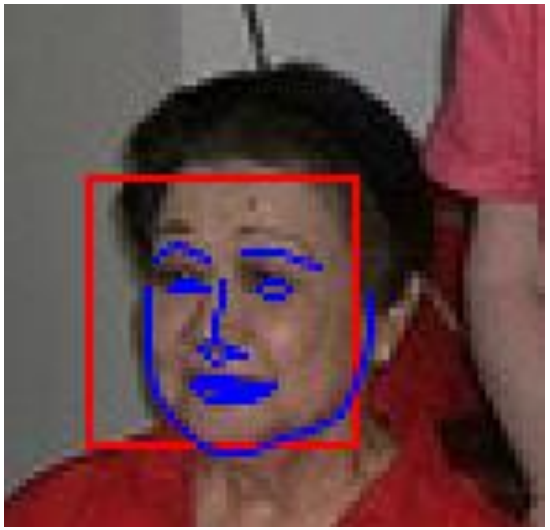
顔ランドマーク (facial landmark)）

- 目，鼻，口，眉毛，あごなどが**ランドマーク**になる
- 位置合わせ，表情分析，人物特定の手がかり

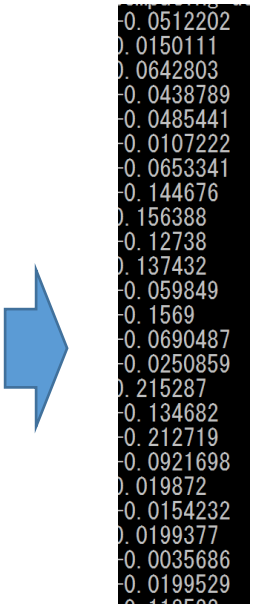


顔のコード化

- **顔のコード**は，複数の数値（ふつう１００以上）の組み合わせ
- **顔画像**から，**顔ランドマーク**を求め，顔のコードを得る
- **さまざまな用途：顔識別（本人の特定），顔認識，年齢の推定，性別の推定，表情の推定，顔の３次元再構成**など

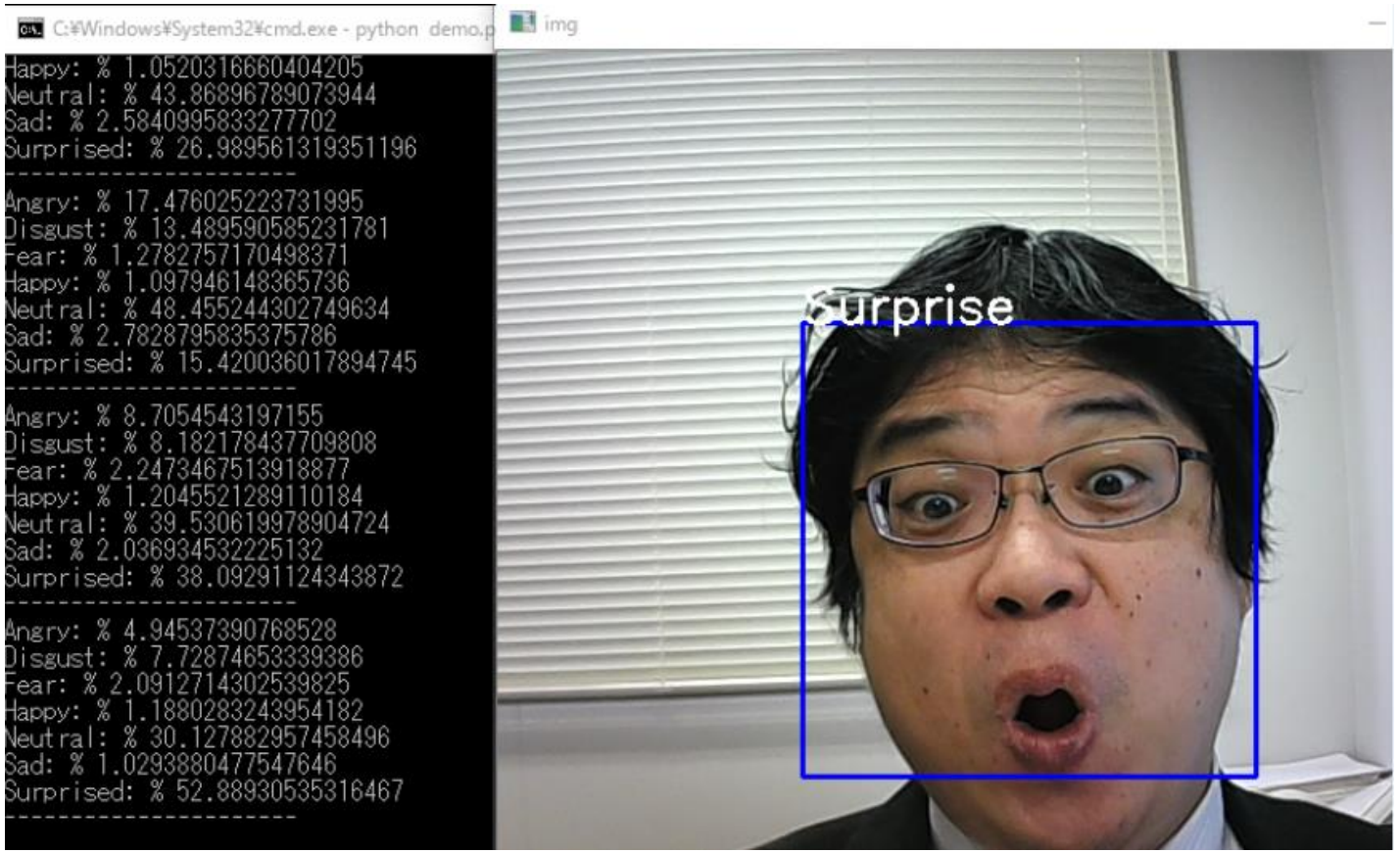


顔検出，
顔ランドマーク



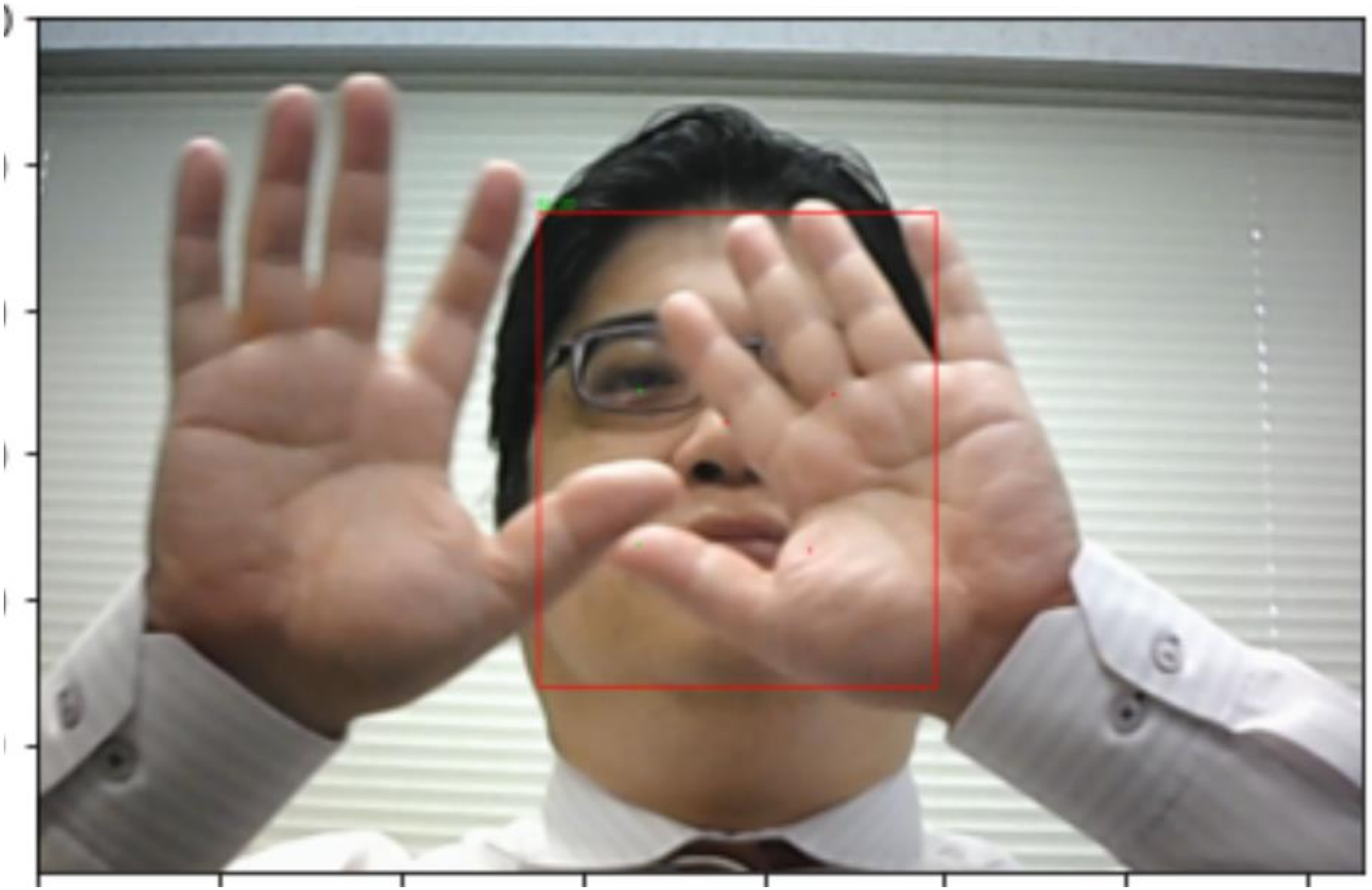
顔のコード

表情の推定



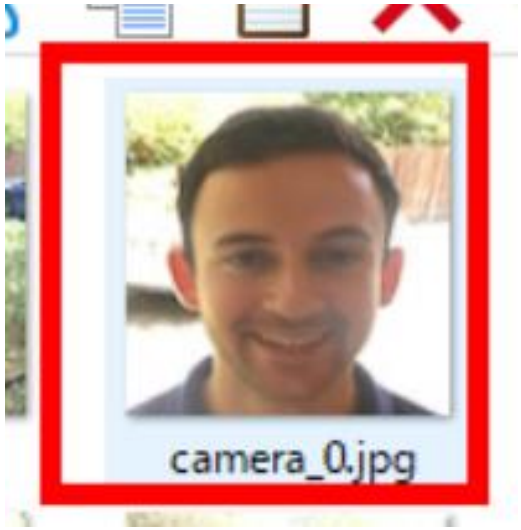
- ここでは，**７種の表情** Angry, Disgust, Fear, Happy, Neutral, Sad, Surprised の**それぞれの確率**を推定
- https://github.com/ezgiakcora/Facial-Expression-Keras で公開されている成果物

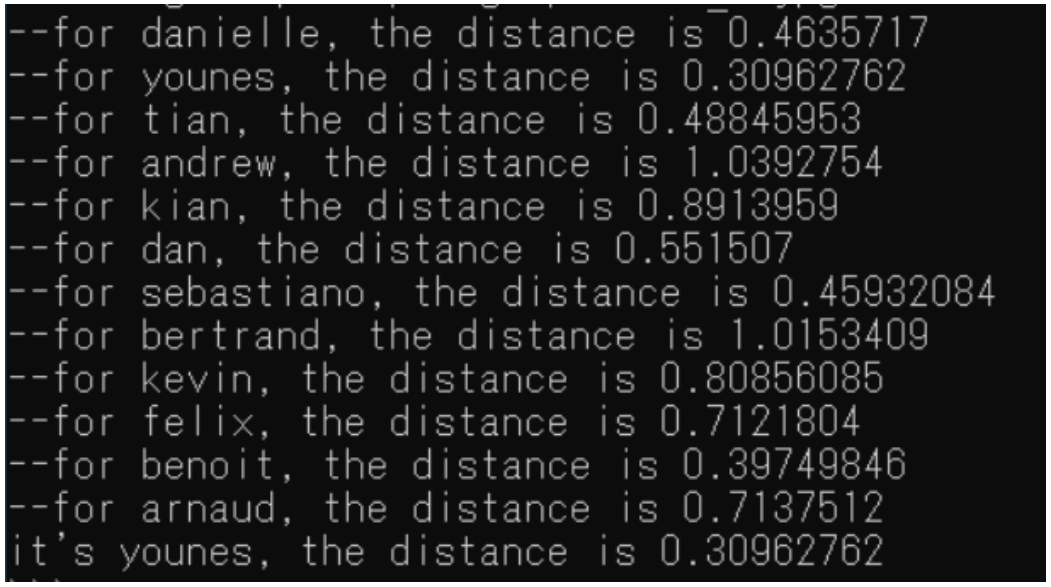
年齢の推定，性別の推定



'gender': 1, 'age': 30, '

人物特定の例





younes

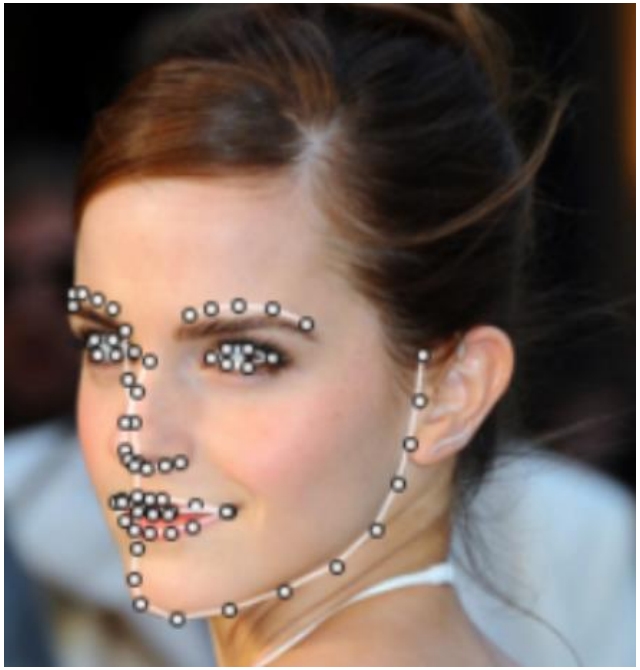
顔写真からの３次元再構成

3DFFA 法（2022年発表）

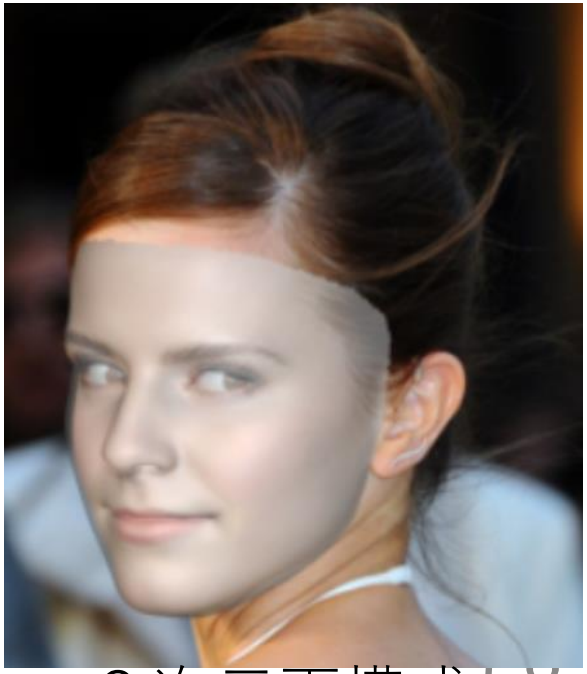
- まず入力画像から顔検出を行い、顔領域を特定
- 次に、その顔領域に対して顔ランドマークの検出を行う

  ランドマーク：目、鼻、口などの特徴的な点
- 検出されたランドマークを手がかりに、**あらかじめ用意された3次元の顔モデル**を**顔画像に合わせて変形**

元画像

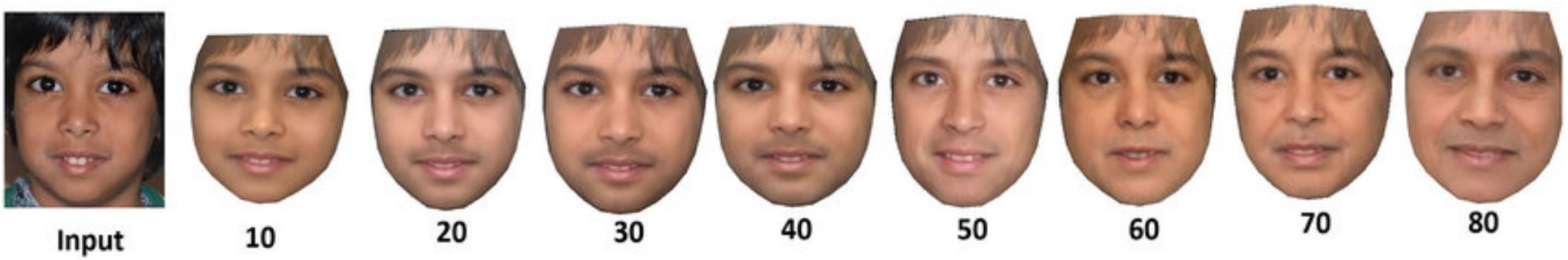


顔ランドマーク



３次元再構成

Face aging



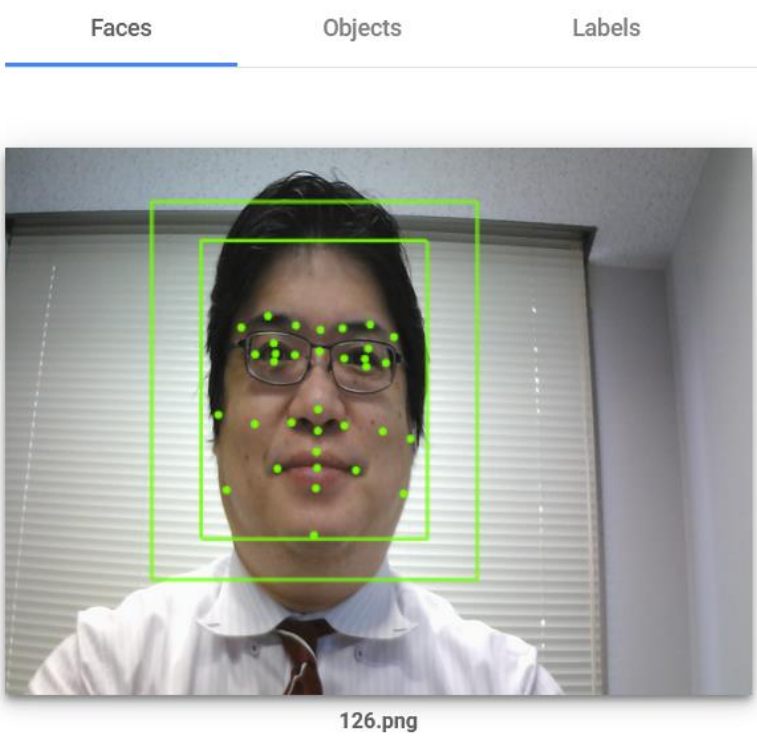
Given the frontal view of a single image of a child, an ageing algorithm can generate his faces with varied ages

元の顔画像から，さまざまな年齢の顔画像を生成

Ali Elmahmudi, H. Ugail,
A framework for facial age progression and regression using exemplar face templates, 2020.

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



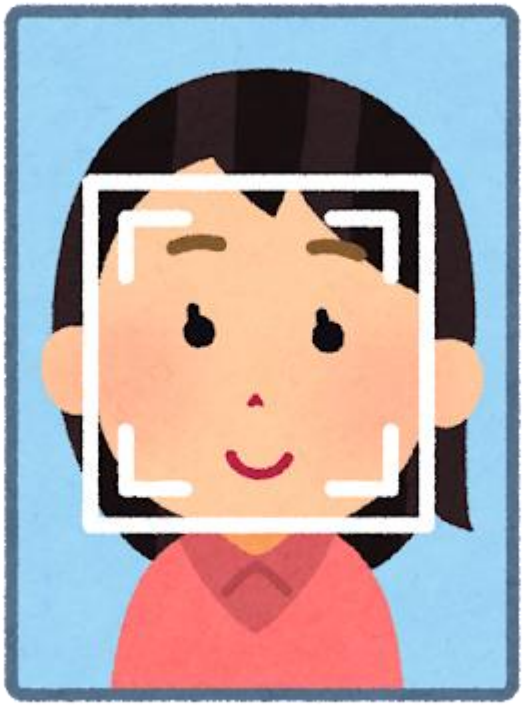
Joy	Very Likely
Sorrow	Very Unlikely
Anger	Very Unlikely
Surprise	Very Unlikely
Exposed	Very Unlikely
Blurred	Very Unlikely
Headwear	Very Unlikely

Roll: 2° Tilt: 10° Pan: 1°

Confidence 95%

画像分類の結果

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



- **顔識別（本人の特定）**を行い，鍵代わりに使用

過去，**写真の中の顔が誰であるか**を，自動で特定，**SNS のタグ付け**が行われてきた
この機能を**廃止**するというニュースも (2021年)

便利さより　⇒　**プライバシー**，**人権**が重要に

職場での行動把握

・**「顔」が本人の特定の手掛かり**

・適切な監視により，安全が高まるという考え方も（プライバシ，人権の尊重は前提）

群衆のカウント

- 群衆のカウント（画像内の人数を数える）
- 監視等に役立つ.

元画像

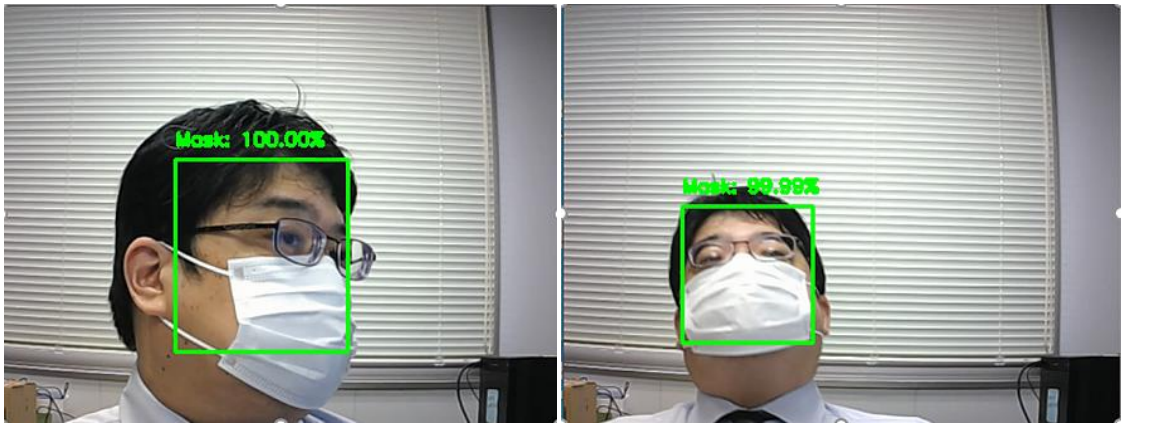
FIDTM 法による群衆のカウント

FIDTM 法（2021年発表）は，
それ以前の手法よりも，さまざまな大きさの顔を精度よく検出できるとされている

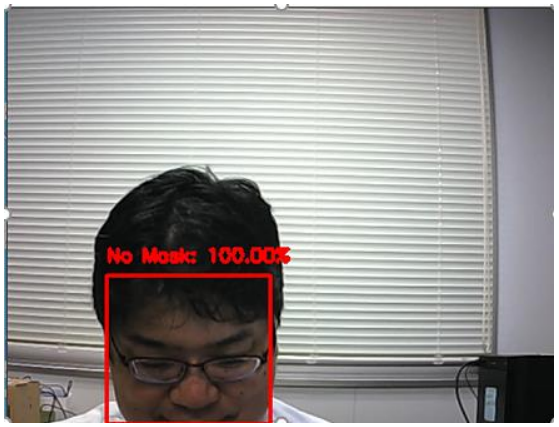
マスク有りの顔，マスク無しの顔検出

Chandrika Deb の顔マスク検出法では，次を同時に実行

- 顔検出
- マスク有りの顔とマスク無しの顔の画像分類



マスク有りの顔検出



マスク無しの顔検出

訓練データ：顔のデータセット（マスク有り: 2165 枚，マスク無し 1930 枚）

顔情報処理の用途

- **顔による本人確認**

- **集団行動解析，人流解析，人数推定**

- **その他**

  表情の推定，年齢判定，３次元再構成，
  顔のセグメンテーション，顔画像の生成　など

顔情報処理まとめ

顔情報処理は次の要素から成り立つ

- **顔検出**: 顔の数をカウントし、顔情報処理の初期ステップを行う。
- **顔ランドマーク**: 目、鼻、口などの特定のポイントを使用して、位置調整（アラインメント）や表情分析、人物特定を行う。
- **顔のコード化**: 顔画像からランドマークを抽出し、これをもとに顔を数値で表現。これは顔識別、顔認識、年齢・性別・表情の推定、顔の3次元再構成などに利用される。
- **3次元再構成**: 元の画像から3次元の顔を生成する。

これらの技術はさまざまな応用が可能ですが、プライバシーや人権の観点から慎重な使用が求められます。

全体まとめ

- **コンピュータビジョン**：コンピュータが実世界の情報を理解・活用する技術。
- **ディープニューラルネットワーク**：画像分類等の精度向上に貢献。
- **畳み込み**：データとカーネルを重ね合わせ、１つの値にまとめる技術。コンピュータビジョンで重要。
- **顔情報処理**：顔検出、顔ランドマーク、顔のコード化、3D再構成から成る。プライバシーや人権の観点から注意が必要。

今回の授業を学ぶ意義と満足感

① コンピュータビジョンとディープラーニングの学習継続による知的好奇心の充足と専門知識の習得

② AI エンジニアとしての成長

③ AI の応用事例の理解．社会課題の解決，アイデア創出力の向上

④ 技術の進展を踏まえ，技術の社会的影響を考察する考察力の強化，倫理観を備えた技術者としての成長