

# 画像を扱うニューラルネットワーク

URL: <https://www.kkaneko.jp/cc/ni/index.html>

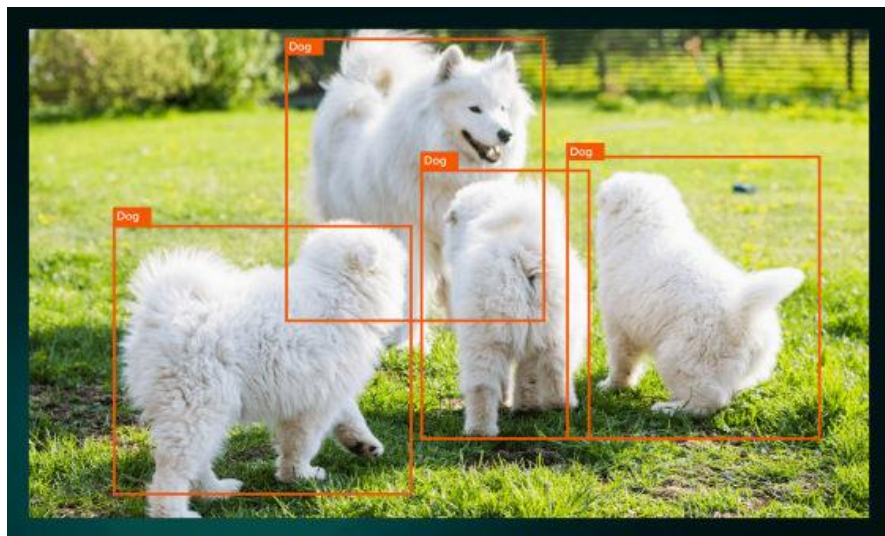
金子邦彦



# アウトライン

1. 画像を扱うニューラルネットワーク
2. 畳み込み
3. CNN の仕組み
4. CNN の詳細

# ニューラルネットワークでの画像処理の例



物体検出

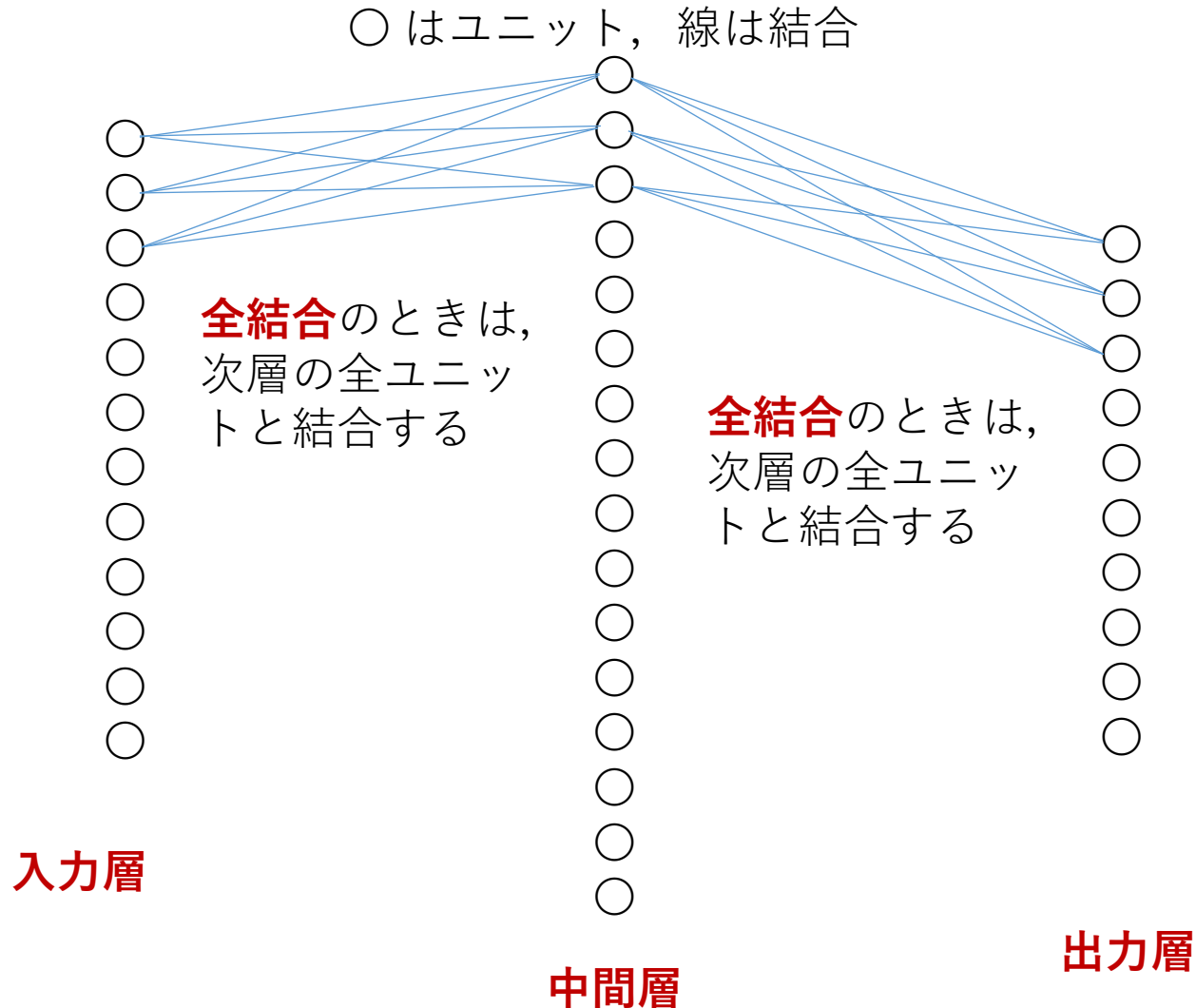


写真の分析

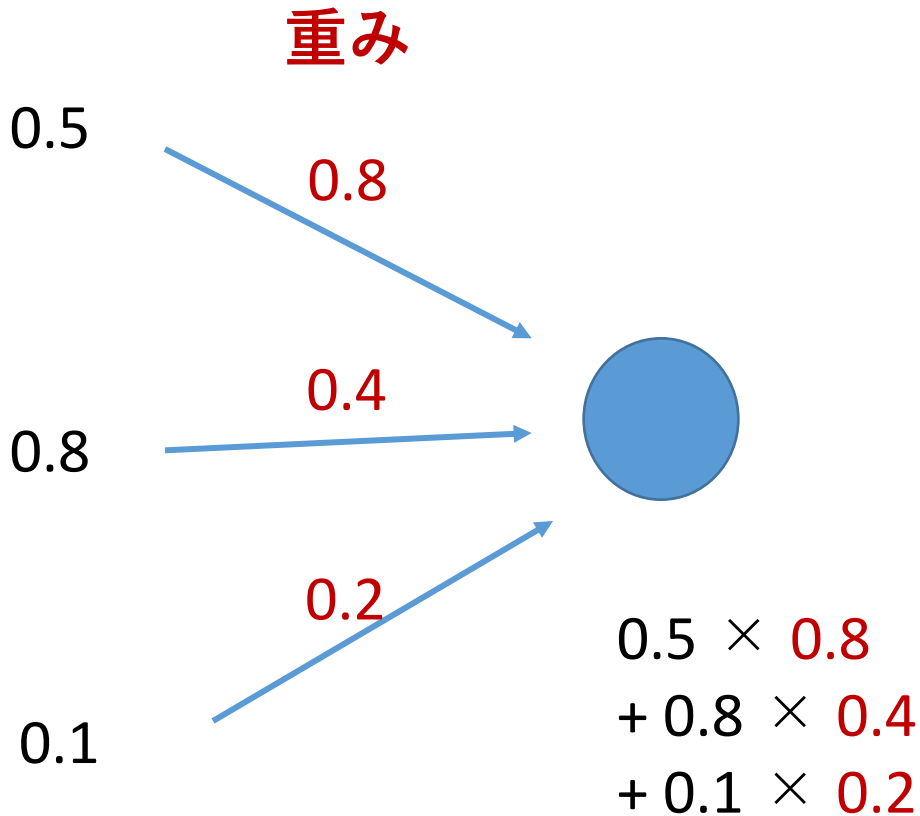
# ニューラルネットワークでの処理の仕組み



- 前の層から結果を受けとって、次の層へ結果を渡す
- 学習の時には、結合の重みが自動調整される



# ユニットの仕組み



前の層のユニットの出力に  
**重み**をかけたものの**総和**

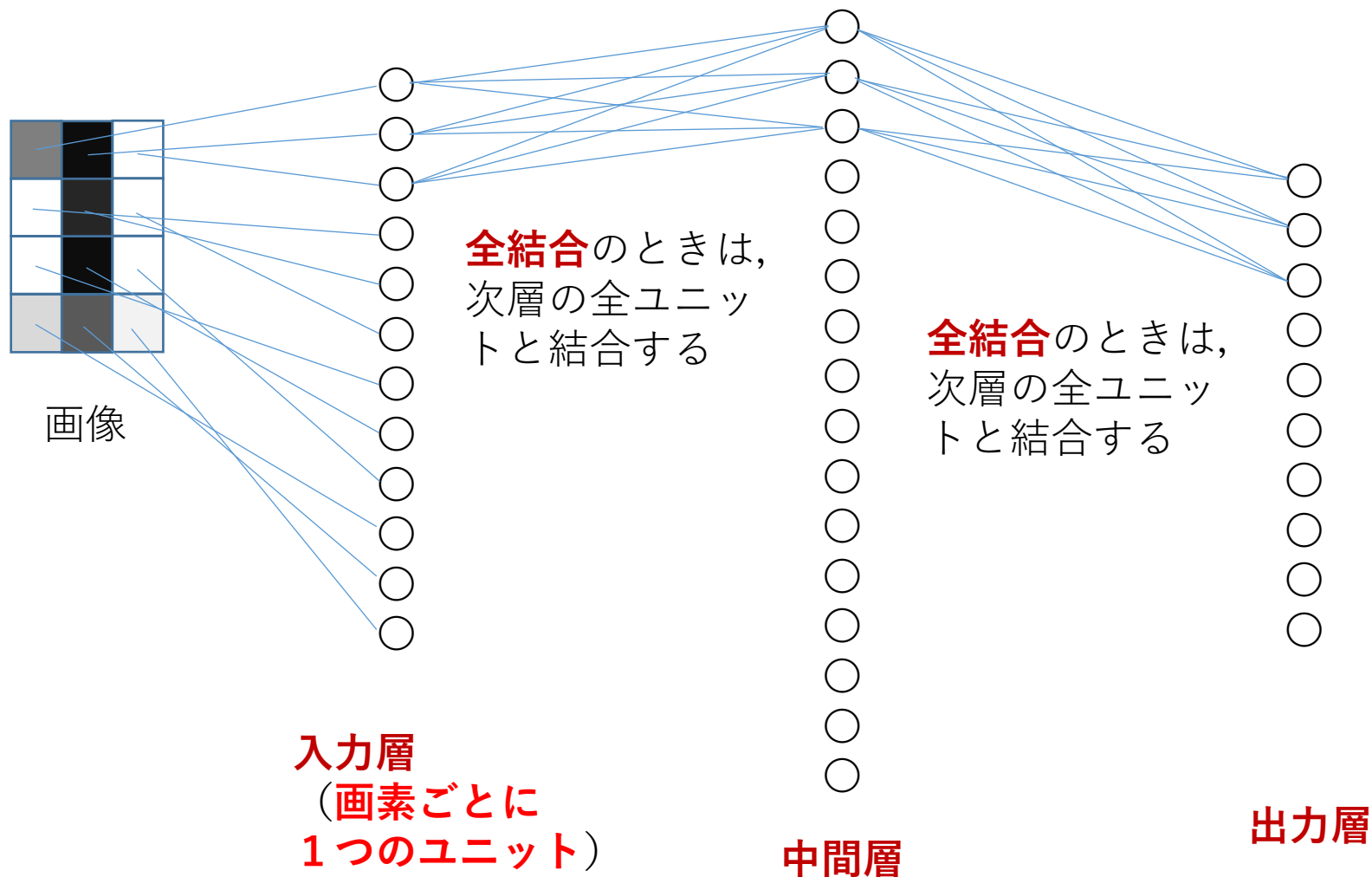


この値に応じて、このユニットは**活性化**する  
(値が大きいほど、**活性化**の度合いが高い)

# 画像とニューラルネットワーク

モノクロ画像（濃淡画像）をニューラルネットワークで扱うとき、  
入力層では、画素ごとに1つのユニットである

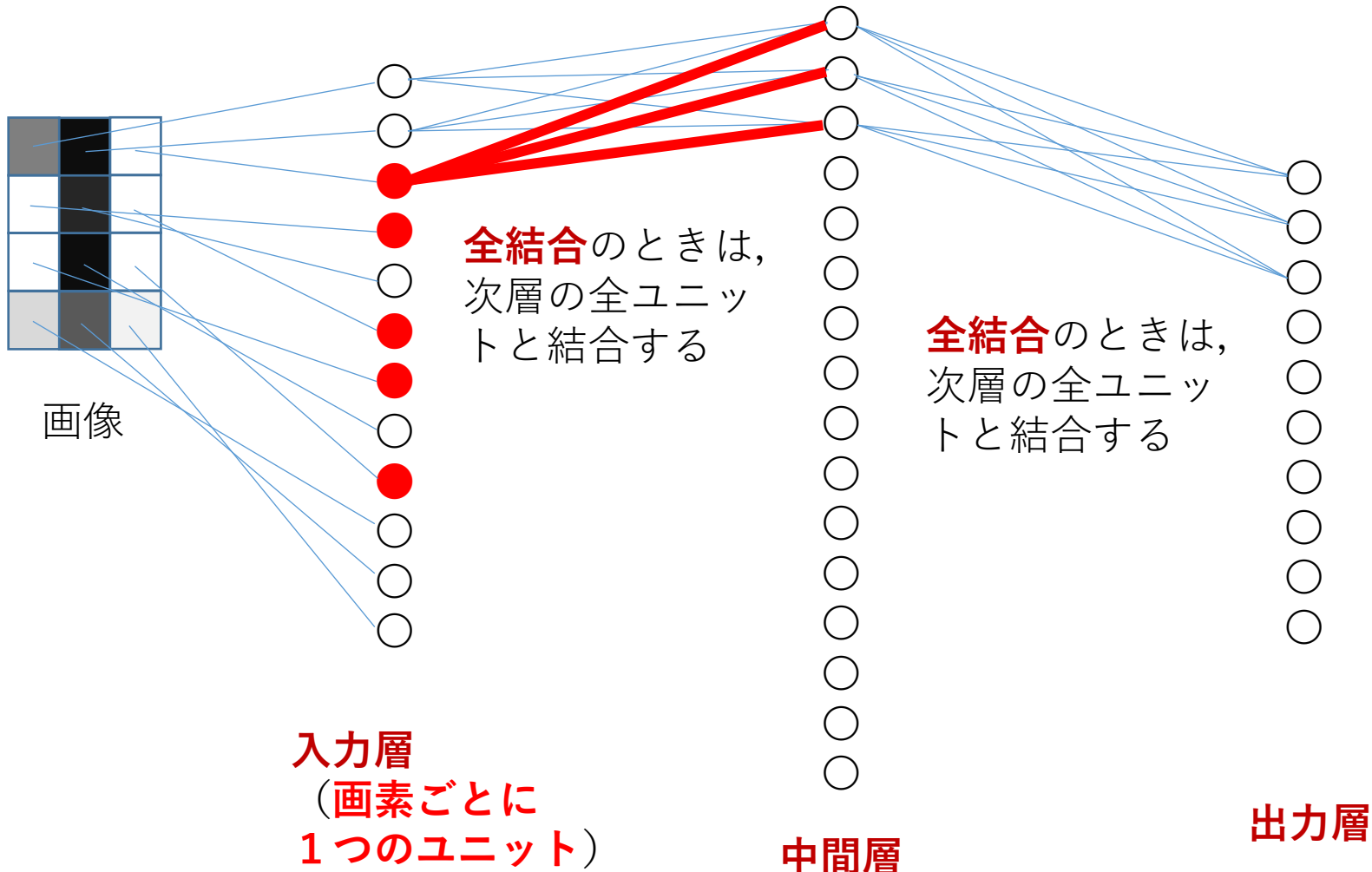
○はユニット，線は結合



# 伝搬

入力層では、画素が「明るい」ときに活性化。その情報は結合を通して、次の層に伝搬する

○はユニット，線は結合



# ニューラルネットワークの仕組み



- **学習**では、正しい結果が得られるように、**結合の重み**を**自動調整**する
- 入力層： 濃淡画像（モノクロ画像）を扱うときは、ユニット数が画素数に等しい（画素1つごとにユニット1つ）
- 中間層： 中間層の数，ユニット数などは自由に設定できる
- 出力層： 「**10種類に分類したい**」というときは、**出力層のユニット数は10**。



## 2 畳み込み

金子邦彦



- 画像を、**全結合のニューラルネットワーク**で扱おうときの問題
  - **入力層のユニット数は大**  
640 × 480 のモノクロ画像：画素数は 307,200
  - **結合の数也大**
  - **学習（結合の重みの調整）のために必要となる画像也大**
- **畳み込み**を利用して、**近くの画素の情報を1つにまとめることにより**、**ユニット数と結合数の削減**を行う

# 畳み込み



## • カーネル（フィルタ）を使用

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

Filter / Kernel

1	0	1
1	1	1
0	0	1

元画像の中から、  
**カーネル**に似た  
部分を取り出すために、  
**畳み込み**を行う

カーネル（ $3 \times 3$  マス）

元画像（ $5 \times 5$  マス）

# 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)

畳み込みのために、  
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

$0 \times 1$	$1 \times 0$	$1 \times 1$
$0 \times 1$	$1 \times 1$	$1 \times 1$
$0 \times 0$	$1 \times 0$	$1 \times 1$

合計: 4 (これが畳み込み結果)

畳み込み結果

カーネルとの類似度  
を表している

# 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)

切り出し領域を横にずらす

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

# 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)

切り出し領域を縦横にずらす

画像全体について  
畳み込み



<b>4</b>	<b>3</b>	<b>5</b>
<b>4</b>	<b>3</b>	<b>5</b>
<b>4</b>	<b>3</b>	<b>5</b>

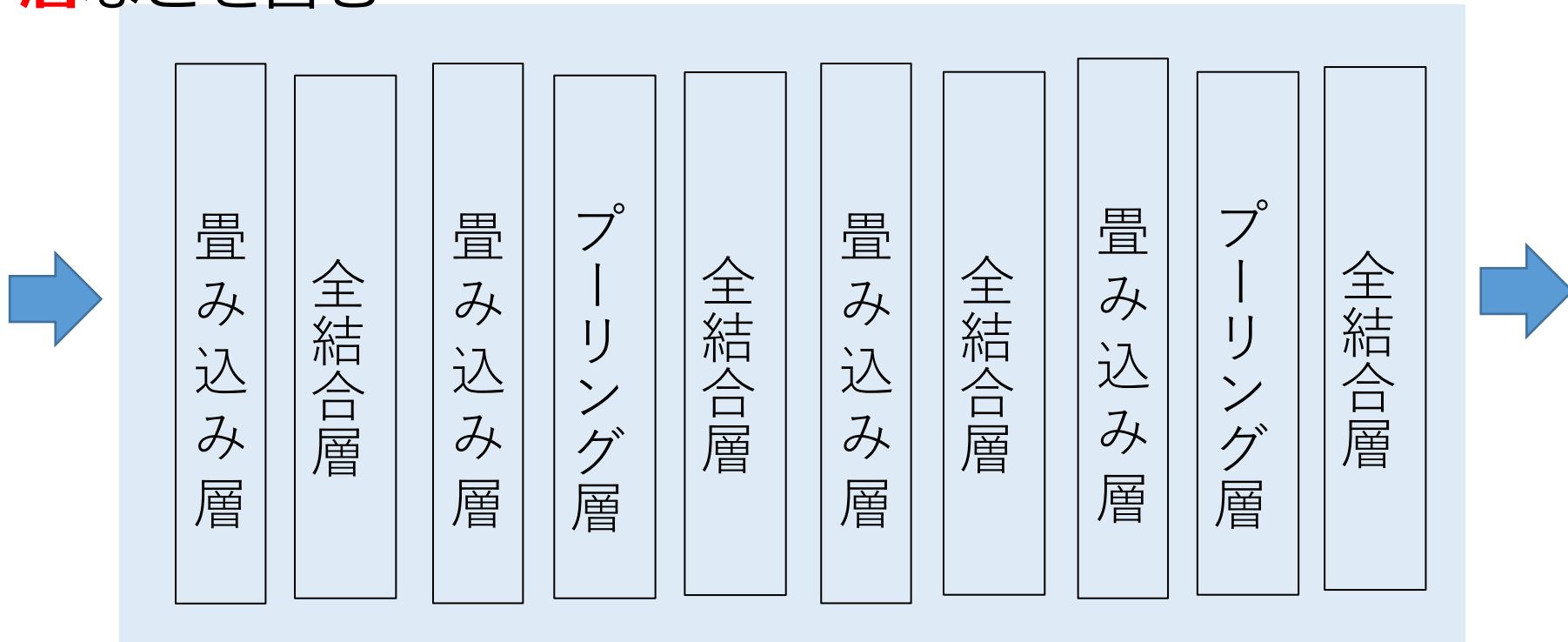
畳み込み結果

# 3. CNN の仕組み

金子邦彦



- CNN は、畳み込みニューラルネットワーク (Convolutional Neural Network) で、**ニューラルネットワークの種類**
- CNN は、**全結合層**の他に、**畳み込み**を行う**畳み込み層**などを含む



CNN の例



# 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

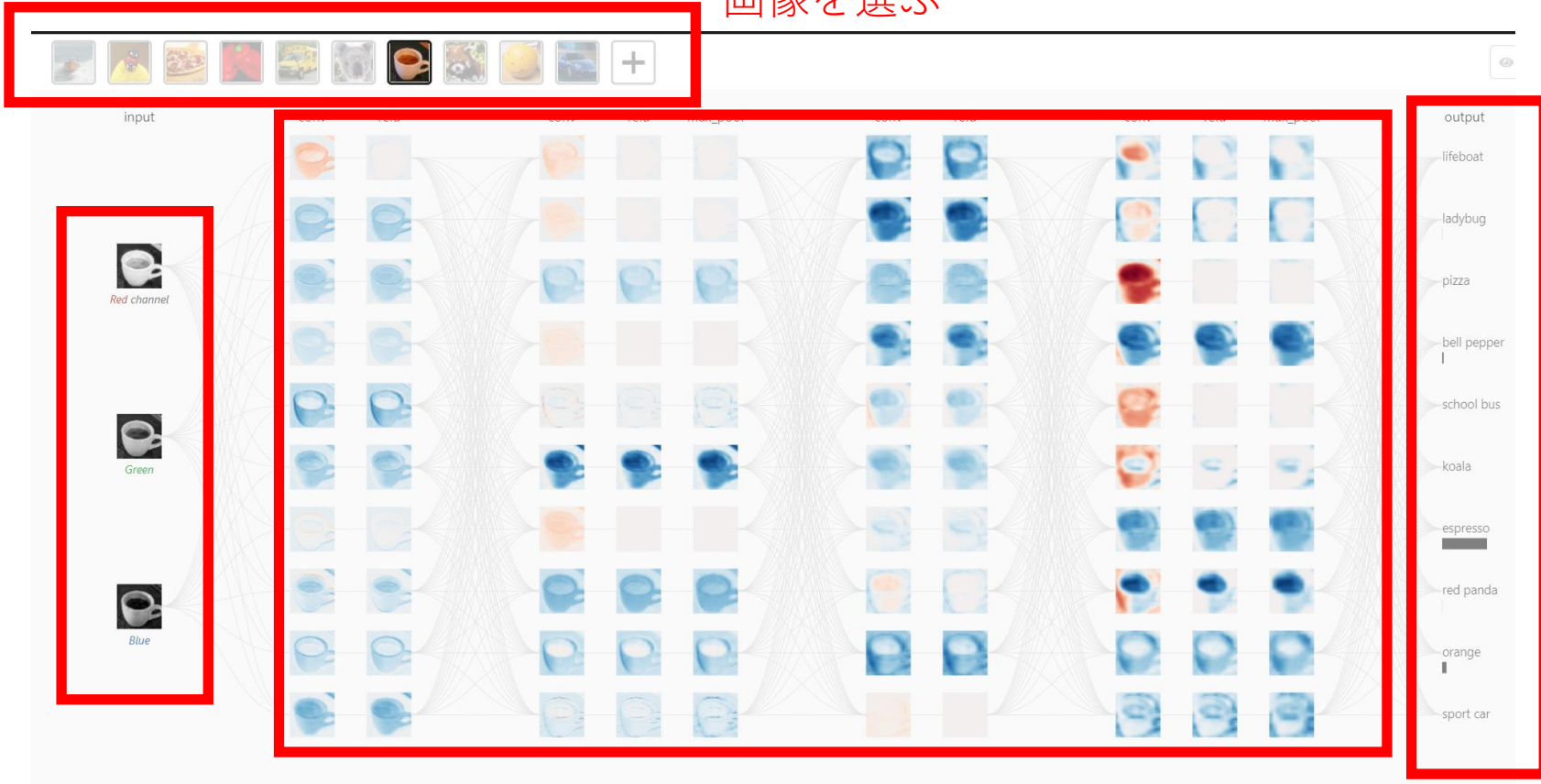
① Webブラウザで次の URL を開く

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

## ② 画面の確認

このニューラルネットワークは、画像を分類する

画像を選ぶ



The screenshot shows a web-based interface for a neural network. At the top, there is a horizontal bar with a row of image thumbnails and a plus sign, labeled "画像を選ぶ". Below this, the main area is divided into three sections:

- Input:** A vertical list of three image thumbnails labeled "Red channel", "Green", and "Blue".
- Neural Network:** A large diagram showing a multi-layered neural network with nodes and connections, processing the input images.
- Output:** A vertical list of classification categories: lifeboat, ladybug, pizza, bell pepper, school bus, koala, espresso, red panda, orange, and sport car. The "espresso" category is highlighted with a thick black bar.

元画像の  
赤青緑の成分

ニューラルネットワーク

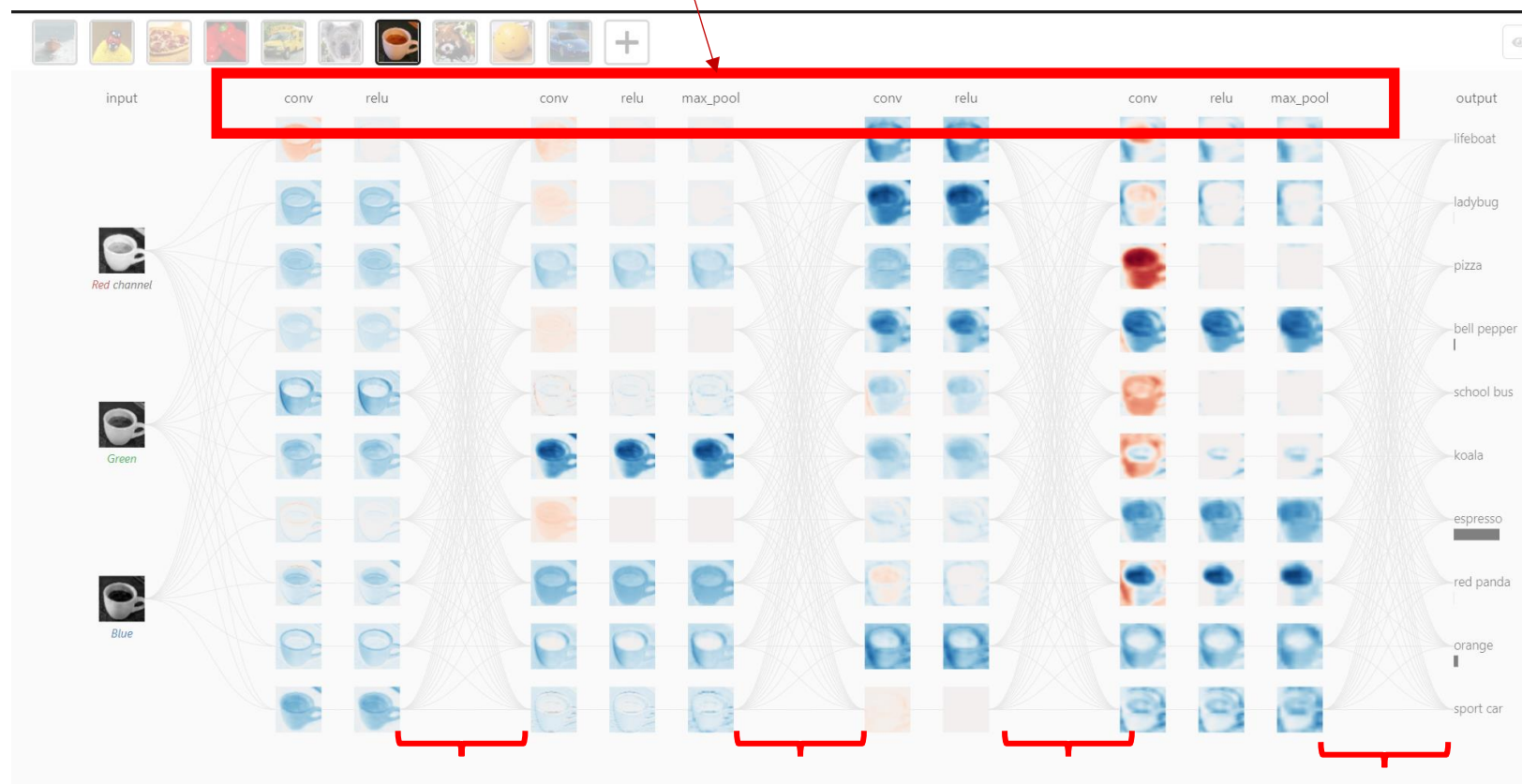
画像の分類結果。  
ここでは espresso

# ③ 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

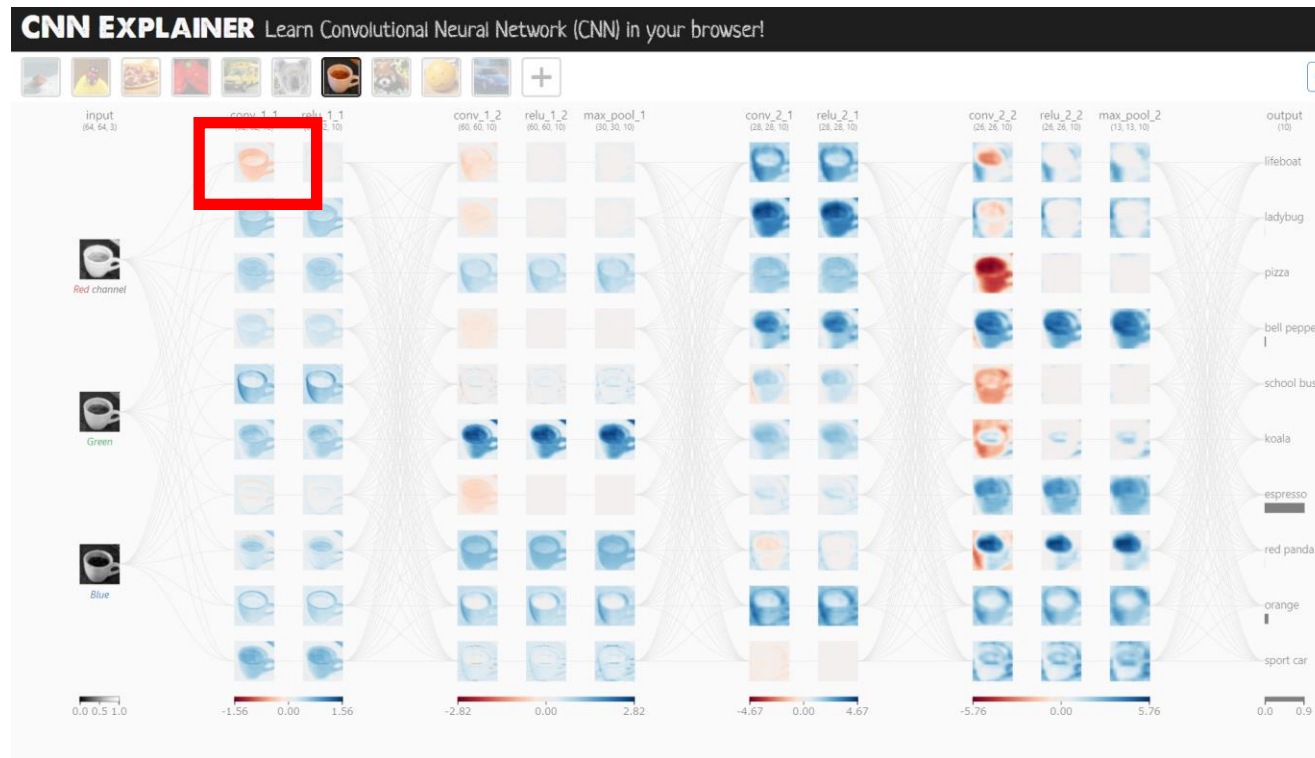
conv は畳み込み層で， max\_pool はプーリング層



全結合層      全結合層      全結合層      全結合層

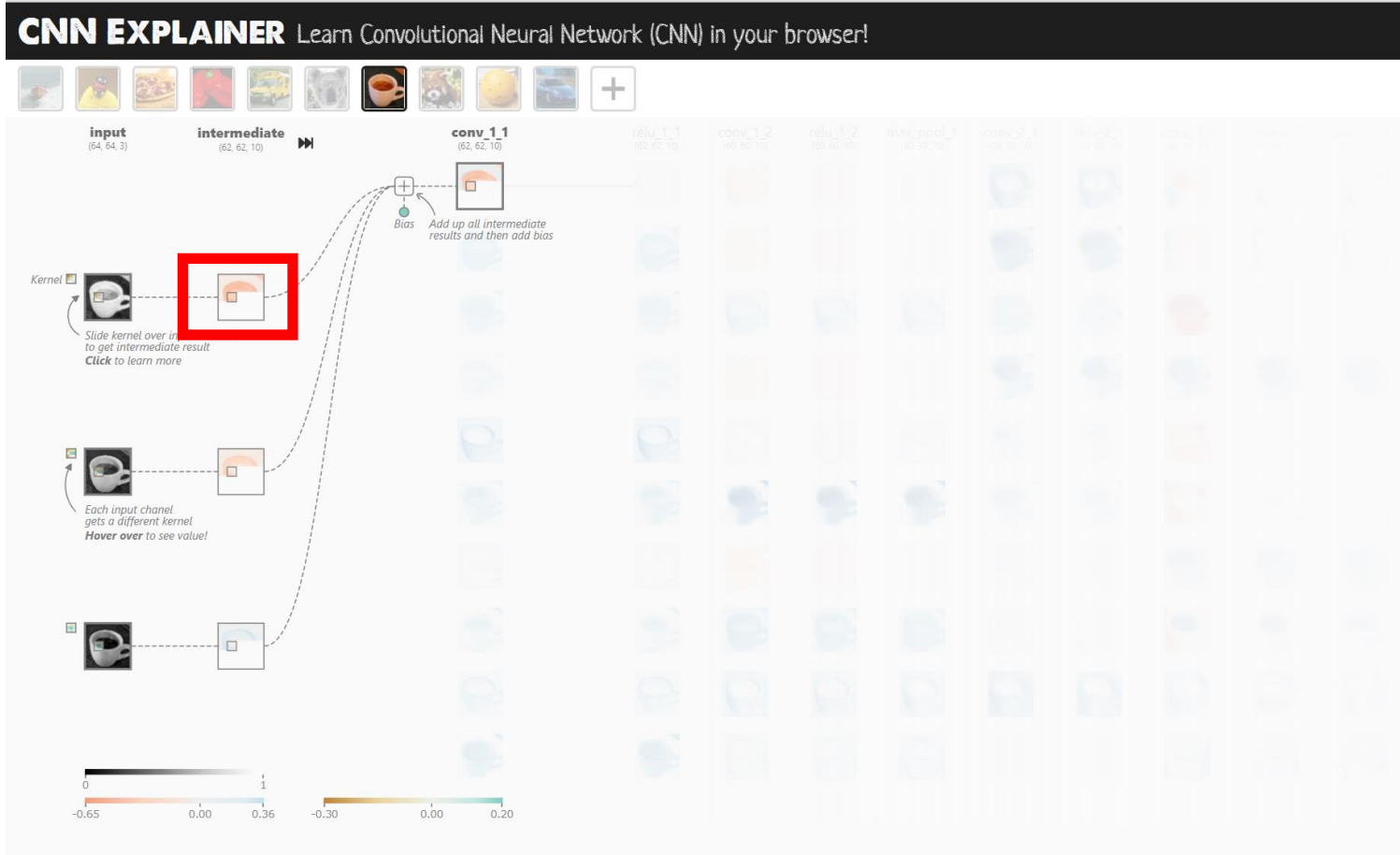
## ④ 左上の画像をクリック

(この画像は、各層での処理結果である。画像1個がユニット1つというわけではない)



## ⑤ 出てきた画像をクリック

**CNN EXPLAINER** Learn Convolutional Neural Network (CNN) in your browser!



The interface displays the following layers and their dimensions:

- input (64, 64, 3)
- intermediate (62, 62, 10)
- conv\_1.1 (62, 62, 10)
- relu\_1.1 (62, 62, 10)
- conv\_1.2 (64, 64, 10)
- relu\_1.2 (64, 64, 10)
- max\_pool\_1 (32, 32, 10)
- conv\_2.1 (32, 32, 10)
- relu\_2.1 (32, 32, 10)
- conv\_2.2 (32, 32, 10)
- relu\_2.2 (32, 32, 10)

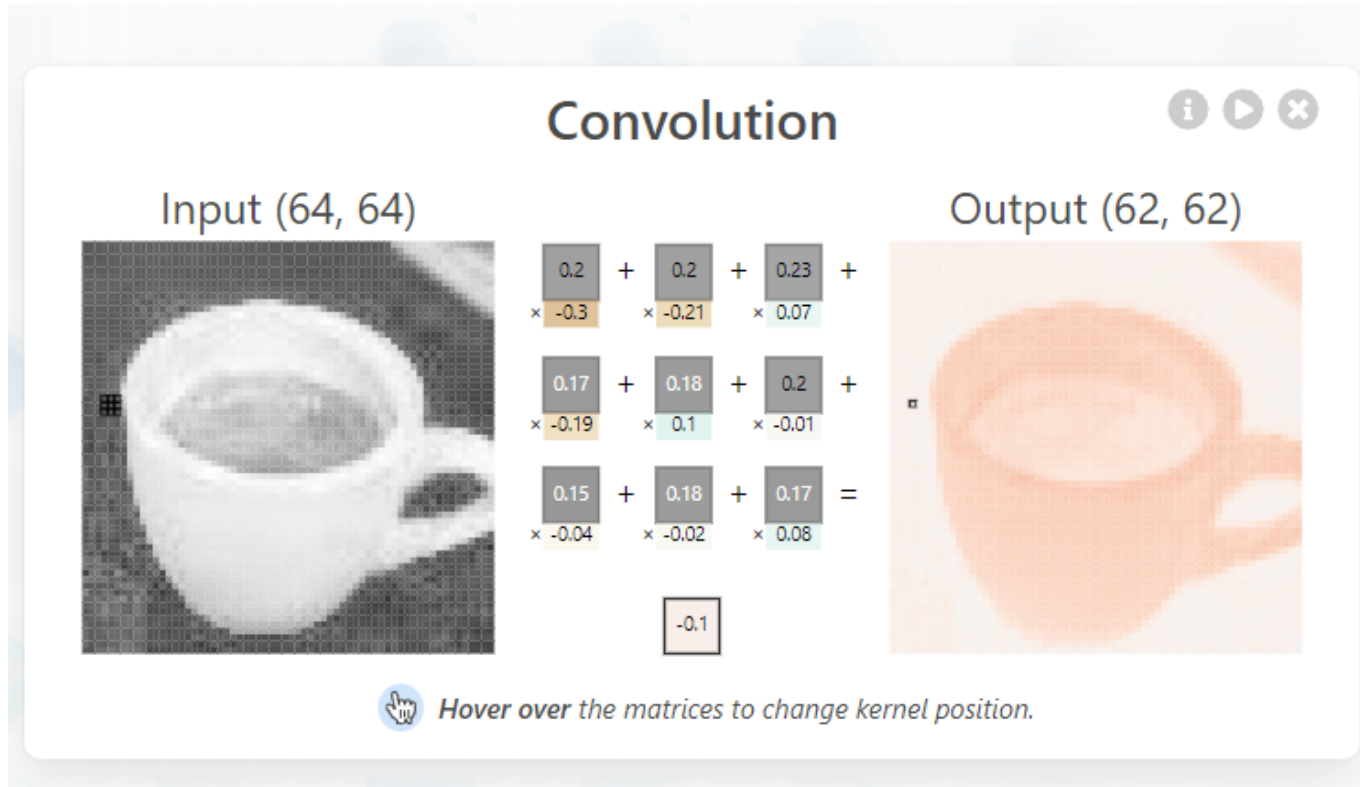
Annotations and instructions:

- Kernel:** Slide kernel over it to get intermediate result. Click to learn more.
- intermediate:** Each input channel gets a different kernel. Hover over to see value!
- conv\_1.1:** Add up all intermediate results and then add bias.

Color scales at the bottom:

- Scale 1: 0 to 1 (0.65 to 0.36)
- Scale 2: -0.30 to 0.20 (0.00 to 0.20)

## ⑥ 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)

# 4. CNN の詳細

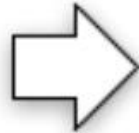
金子邦彦



# プーリング

- プーリングにより，画像サイズが小さくなる
- Max pooling（マックス・プーリング）は，**最大値**を利用するプーリング

4	8	2	1	1	3
7	1	1	2	2	6
2	2	3	8	1	9
2	2	6	2	8	4
3	2	1	2	1	8
1	1	1	3	6	1



8	2	6
2	8	9
3	3	8

- 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  $\frac{1}{4}$  になる



# CNN の作成プログラム



## • 第2回授業のプログラム

```
[ ] from tensorflow.keras import backend as K
    from tensorflow.keras.models import Model, Sequential
    import tensorflow.keras

# 畳み込みニューラルネットワークの作成
num_classes = 10
img_rows, img_cols = 28, 28
if K.image_data_format() == 'channels_first':
    input_shape = (1, img_rows, img_cols)
else:
    input_shape = (img_rows, img_cols, 1)

m = Sequential()
m.add(Conv2D(32, kernel_size=(3, 3),
            activation='relu',
            input_shape=input_shape))
m.add(Conv2D(64, (3, 3), activation='relu'))
m.add(MaxPooling2D(pool_size=(2, 2)))
m.add(Dropout(0.25))
m.add(Flatten())
m.add(Dense(128, activation='relu'))
m.add(Dropout(0.5))
m.add(Dense(num_classes, activation='softmax'))
m.compile(loss='categorical_crossentropy', optimizer='adam',
          metrics=['accuracy'])
m.summary()
```

① 1層目：畳み込み層

② 2層目：畳み込み層

③ 3層目：プーリング層

④ 4層目：全結合層

⑤ 5層目：全結合層

# 全体まとめ



- 画像の画素数が多い
- **全結合層**だけでは、画像をうまく扱えない
- **CNN**（**畳み込みニューラルネットワーク**）の**畳み込み層**、**プーリング層**を用いて、画像を扱う技術が考案されている
- **CNN**（**畳み込みニューラルネットワーク**）の作成は、プログラムで簡単にできる