

目標

1. ともかく自分のPCで動くようになる
2. 自分のデータで処理できるようになる
3. 前処理と後処理（評価方法）を知る

目標

- 分類問題を深層学習で解決する
- 回帰問題を深層学習で解決する
- ファインチューニングを理解する
- 転移学習を理解する
- 分類の評価方法を理解する：2クラス分類／多クラス分類
- 領域分割の概念を理解する：U-Net, U-Net++

課題 1：画像の分類から領域分割

< DataLoaderの使い方 >

- ・ Numpyの配列からデータを読み込む
- ・ フォルダーに保存された画像ファイルをCSVに基づいて読み込む

< モデルの作り方 >

☆自分でCNNを作って画像の2分類を行いROCで評価する

- ・ 自分でCNNを作って画像の4分類を行い混同行列で評価する

< ファインチューニングと転移学習 >

- ・ 既存のモデルをファインチューニング 2分類問題 | 4分類問題
- ・ 既存のモデルで転移学習の結果を得る 2分類問題 | 4分類問題

< 流行りの方法・ViT (Vision Transformer) >

- ・ ともかく動かしてみる (評価なし) 4分類問題

< 画像の領域分割・U-Net, **U-Net++(Nested U-Net)**を使う >

- ・ 胸部X線画像の肺野／背景／人体／心臓領域の分割
- ・ CT画像における顎骨領域の抽出

分類問題のポイント

- 正解がカテゴリ（クラス）である
- 2分類か、3クラス以上の分類かで評価方法が異なる

良悪性分類：2分類

クラス分類：I, II, III, IV, V

2 クラスの画像分類



入



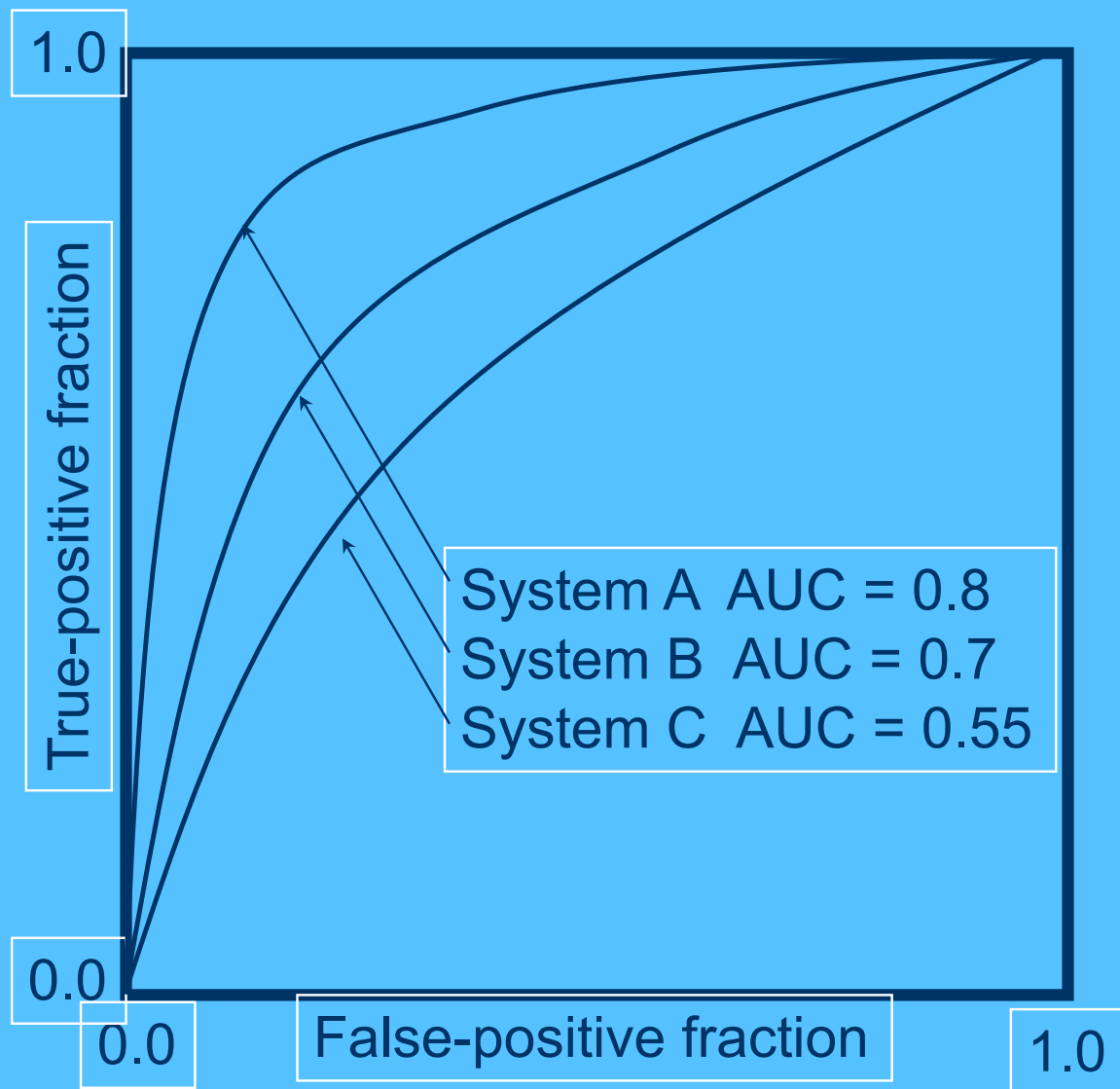
酒

寿司

2クラス分類の例

x $\longrightarrow$ y

良性悪性（2分類）



ROC曲線で評価

ROC曲線下面積
カットオフを決める
感度, 特異度

論文での取り扱い

ORIGINAL RESEARCH

A Deep Learning Approach for Assessment of Regional Wall Motion Abnormality From Echocardiographic Images

Kenya Kusunose, MD, PhD,^a Takashi Abe, MD, PhD,^b Akihiro Haga, PhD,^c Daiju Fukuda, MD, PhD,^a Hirotugu Yamada, MD, PhD,^a Masafumi Harada, MD, PhD,^b Masataka Sata, MD, PhD^a

ABSTRACT

OBJECTIVES This study investigated whether a deep convolutional neural network (DCNN) could provide improved detection of regional wall motion abnormalities (RWMAs) and differentiate among groups of coronary infarction territories from conventional 2-dimensional echocardiographic images compared with that of cardiologists, sonographers, and resident readers.

BACKGROUND An effective intervention for reduction of misreading of RWMAs is needed. The hypothesis was that a DCNN trained using echocardiographic images would provide improved detection of RWMAs in the clinical setting.

METHODS A total of 300 patients with a history of myocardial infarction were enrolled. From this cohort, 3 groups of 100 patients each had infarctions of the left anterior descending (LAD) artery, the left circumflex (LCX) branch, and the right coronary artery (RCA). A total of 100 age-matched control patients with normal wall motion were selected from a database. Each case contained cardiac ultrasonographs from short-axis views at end-diastolic, mid-systolic, and end-systolic phases. After the DCNN underwent 100 steps of training, diagnostic accuracies were calculated from the test set. Independently, 10 versions of the same model were trained, and ensemble predictions were performed using those versions.

 x

超音波画像 3 枚

 y

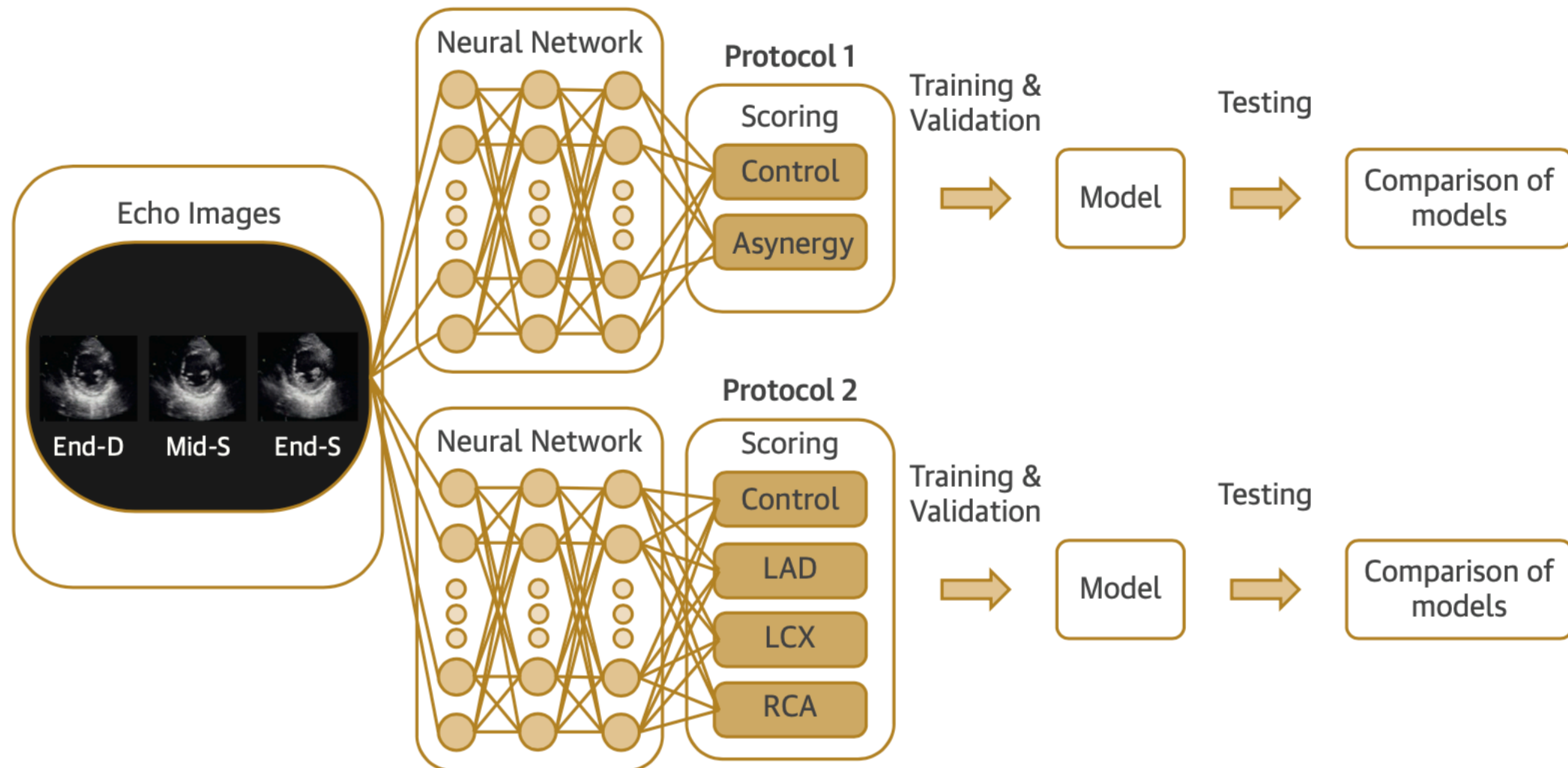
局所壁運動

異常の有無

(2 分類)

 $n = 300$

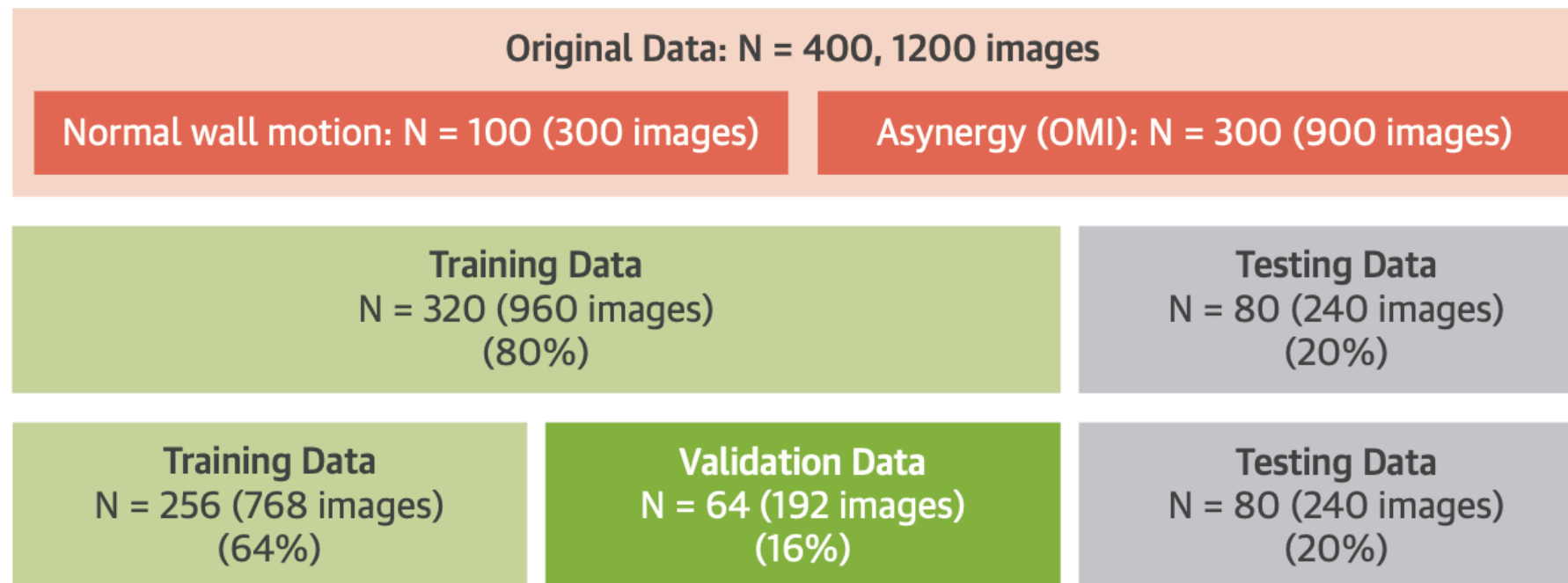
CENTRAL ILLUSTRATION Neural Networks for the Presence of RWMA and the Territory of RWMA



Kusunose, K. et al. J Am Coll Cardiol Img. 2020;13(2):374-81.

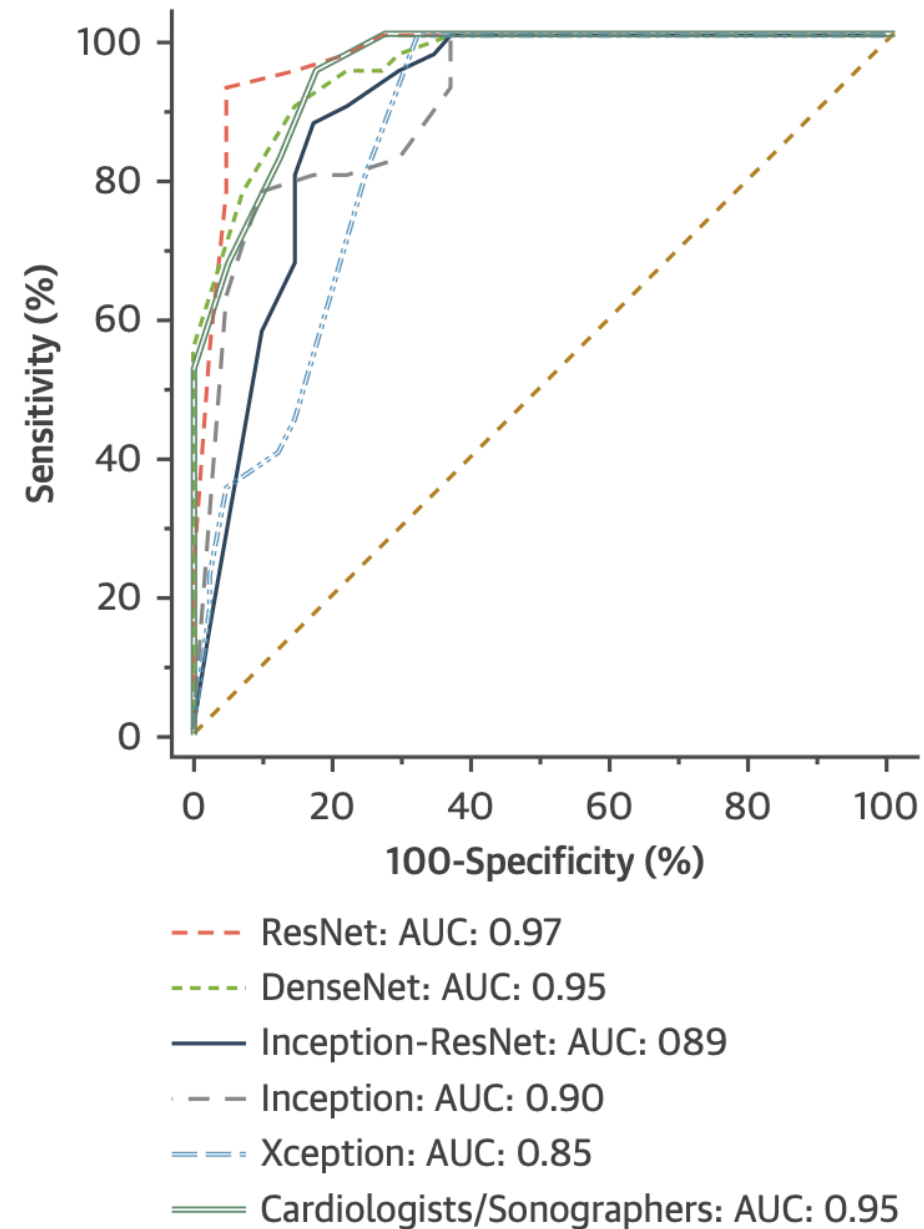
The fully connected layers transform the image features into the final scores by adjusting weights for neuron activations during training. Echo = echocardiography; End-D = end diastolic (phase); LAD = left anterior descending artery; LCX = left circumflex artery; Mid/End-S = mid-/end systolic (phases); RCA = right coronary artery; RWMA = regional wall motion abnormality.

FIGURE 1 Import Data



There were a total of 400 cases from which 1,200 images were split with 256 cases (786 images) as the training set, 64 cases (192 images) as the validation set, and 80 cases (240 images) as the test set. OMI = old myocardial infarction.

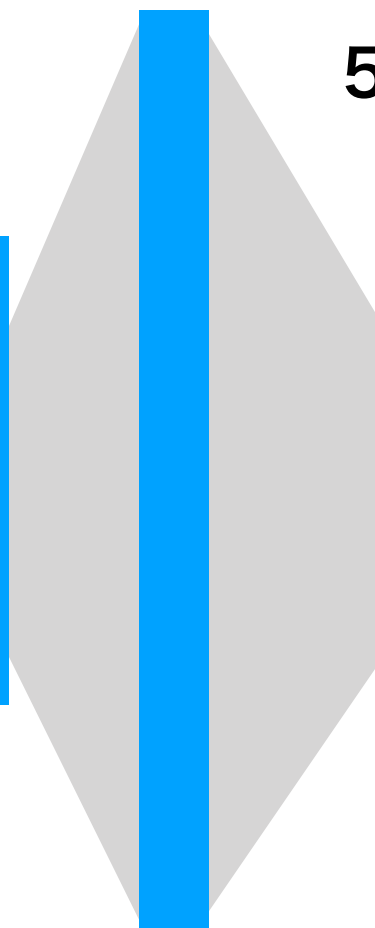
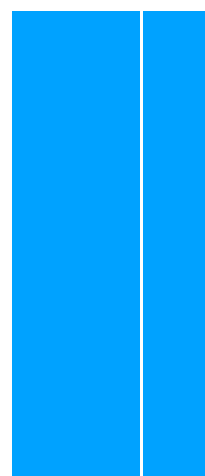
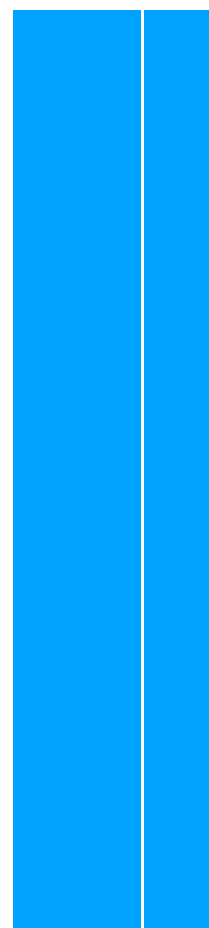
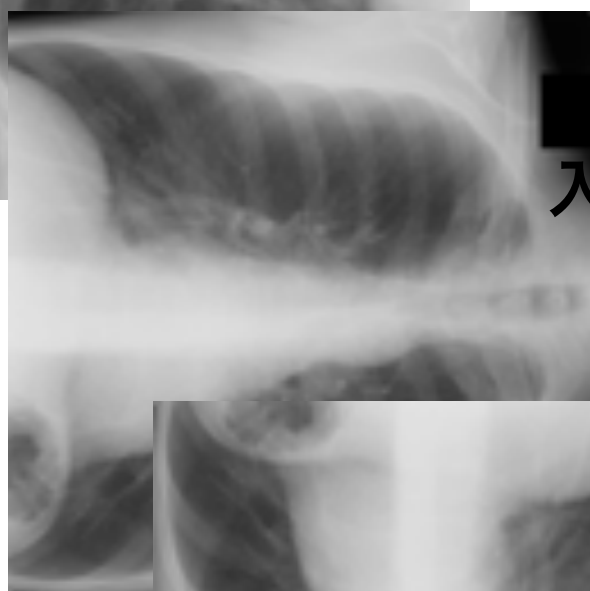
FIGURE 3 Diagnostic Ability to Detect the Presence of RWMAs



The area under the curves by several deep learning algorithms for detection of territories of wall motion abnormality were good. AUC = area under the curve; ResNet = residual neural network; other abbreviation as in [Figure 2](#).

4 方向の分類問題

4クラスの画像分類



上
左
右
下

512

2

512

1000

多クラス分類の例

$x \longrightarrow y$

反応

ステージ（多分類）

実数

	I	II	III	IV	V	VI
I	26	2	4	6	2	2
II	2	34	5	3	10	2
III	4	5	44	6	2	1
IV	5	2	1	26	4	6
V	11	3	6	12	16	8
VI	6	3	17	2	19	86

混同行列で評価

精度 = (対角成分) / 全体
分類ごとの精度

論文での取り扱い



Fully Automated Echocardiogram Interpretation in Clinical Practice

Feasibility and Diagnostic Accuracy

Editorials, see p 1636 and p 1639

BACKGROUND: Automated cardiac image interpretation has the potential to transform clinical practice in multiple ways, including enabling serial assessment of cardiac function by nonexperts in primary care and rural settings. We hypothesized that advances in computer vision could enable building a fully automated, scalable analysis pipeline for echocardiogram interpretation, including (1) view identification, (2) image segmentation, (3) quantification of structure and function, and (4) disease detection.

METHOD: Using 14 035 echocardiograms spanning a 10-year period, we trained and evaluated convolutional neural network models for multiple tasks, including automated identification of 23 viewpoints and segmentation of cardiac chambers across 5 common views. The segmentation output was used to quantify chamber volumes and left ventricular mass, determine ejection fraction, and facilitate automated determination of longitudinal strain through speckle tracking. Results were evaluated through comparison to manual segmentation and measurements from 8666 echocardiograms obtained during the routine clinical workflow. Finally, we developed models to detect 3 diseases: hypertrophic cardiomyopathy, cardiac amyloid, and pulmonary arterial hypertension.

Jeffrey Zhang, BA
Sravani Gajjala, MBBS
Pulkit Agrawal, PhD
Geoffrey H. Tison, MD, MPH
Laura A. Hallock, BS
Lauren Beussink-Nelson, RDCS
Mats H. Lassen, BM
Eugene Fan, MD
Mandar A. Aras, MD, PhD
Charran Jordan, MD, PhD
Kirsten E. Fleischmann, MD, MPH
Michelle Melisko, MD
Atif Qasim, MD, MSCE
Sanjiv J. Shah, MD
Ruzena Bajcsy, PhD
Rahul C. Deo, MD, PhD

x

超音波画像 1 枚

y

分割画像

多クラス分類

$n = 14035$

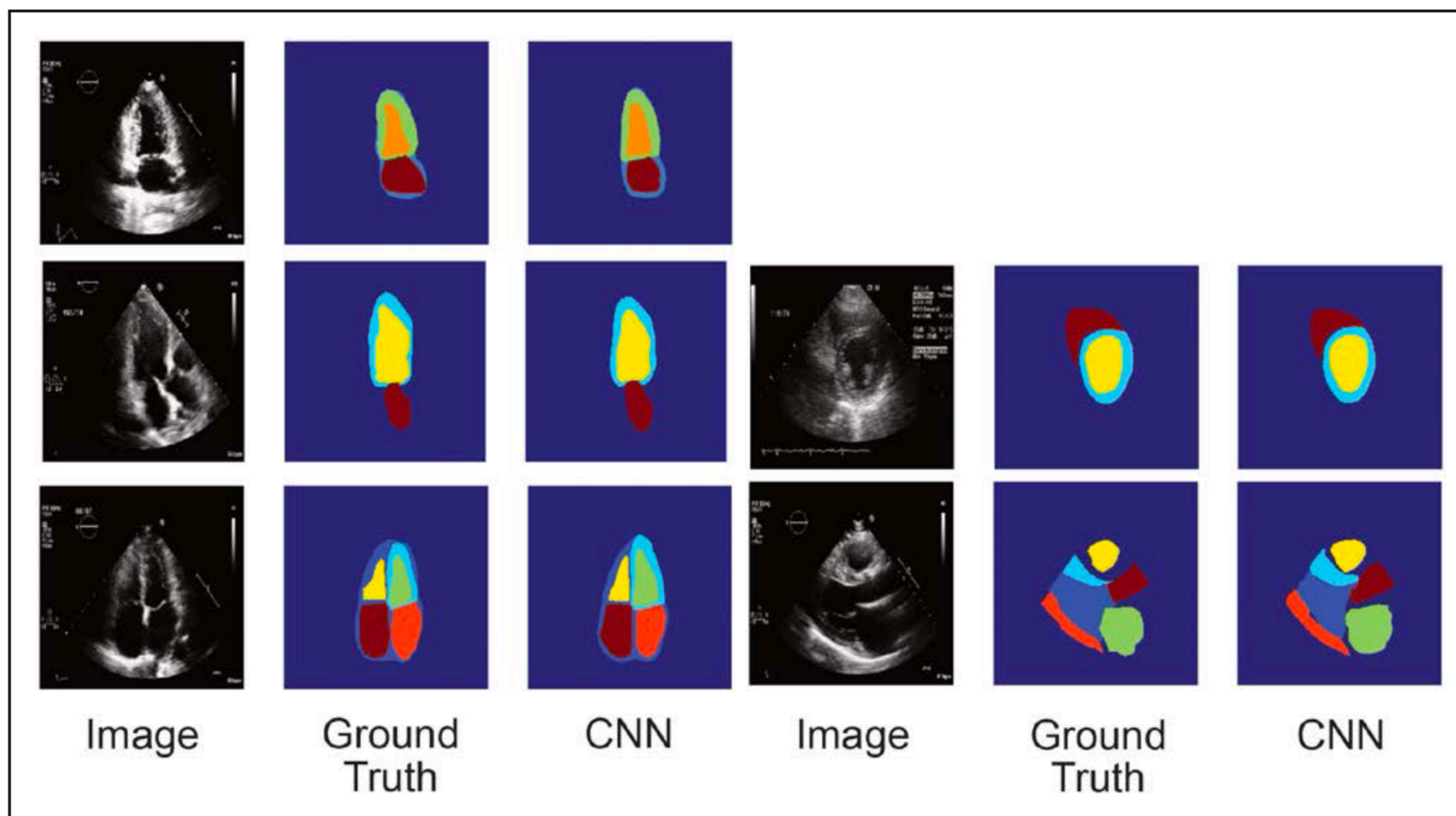


Figure 3. Convolutional neural networks successfully segment cardiac chambers.

We used the U-net algorithm to derive segmentation models for 5 views: A2c, A3c, A4c (left: top, middle, and bottom, respectively), parasternal short axis at the level of the papillary muscle (right, middle), and PLAX (right, bottom). For each view, the trio of images, from left to right, corresponds to the original image, the manually traced image used in training (ground truth), and the automated segmented image (determined as part of the cross-validation process). A2c indicates apical 2-chamber; A3c, apical 3-chamber; A4c, apical 4-chamber; CNN, convolutional neural network; and PLAX, parasternal long axis.

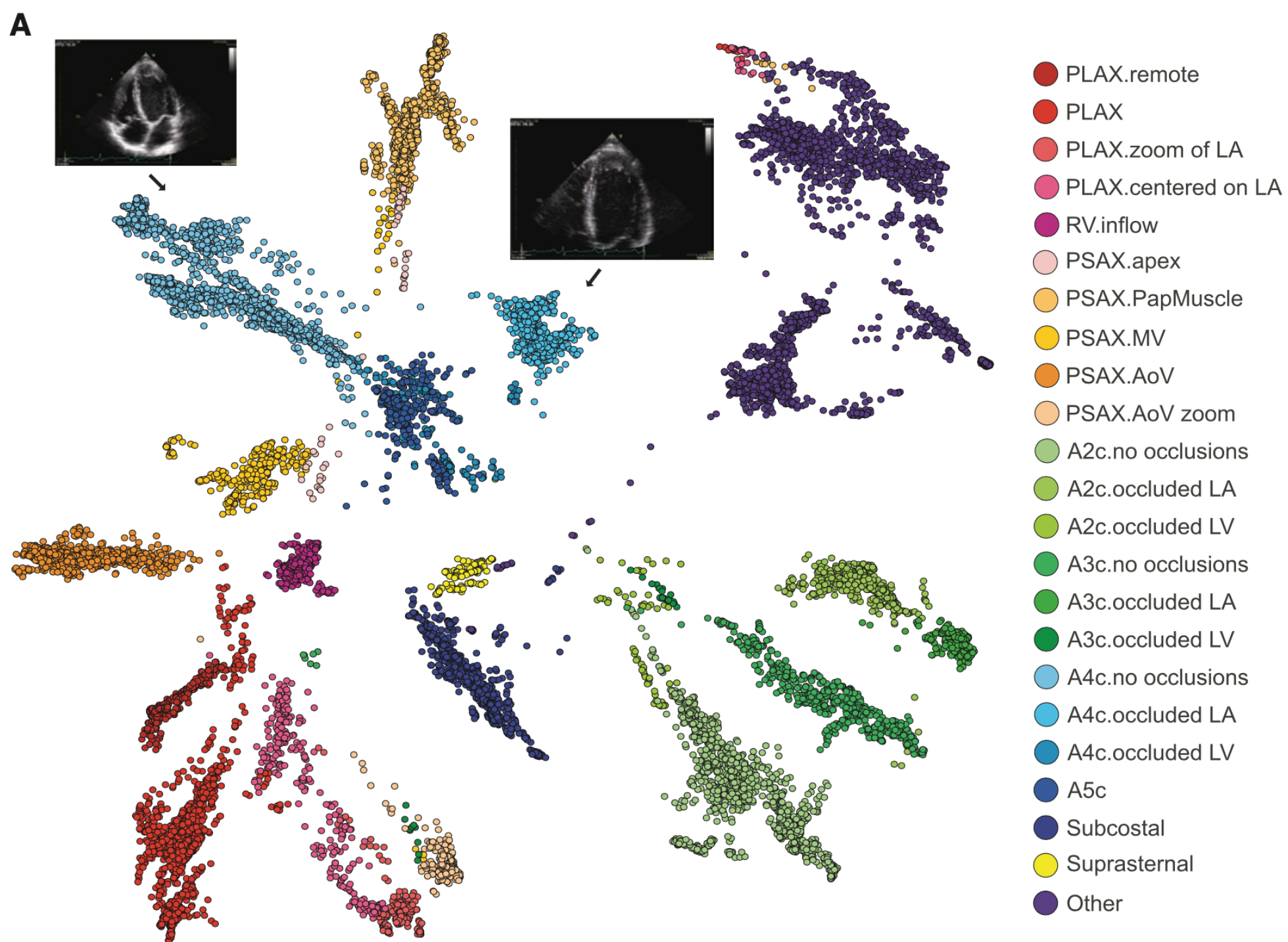


Figure 2. Convolutional neural networks successfully discriminate echocardiographic views.

A, t-Distributed Stochastic Neighbor Embedding (t-SNE) visualization of view classification. t-SNE is an algorithm used to visualize high-dimensional data in lower dimensions. It depicts the successful grouping of test images corresponding to 23 different echocardiographic views. Echocardiographic still images indicate the distinct clustering of images of A4c views without occlusions and those with occlusion of the left atrium. **B**, Confusion matrix demonstrating successful and unsuccessful view classifications within the test data set. Numbers along the diagonal represent successful classifications, whereas off-diagonal entries are misclassifications. A2c indicates apical 2-chamber; A3c, apical 3-chamber; A4c, apical 4-chamber; echo, echocardiogram; LV, left ventricular; and PLAX, parasternal long axis.

Zhang J, Gajjala S, Agrawal P, Tison GH, Hallock LA, Beussink-Nelson L, Lassen MH, Fan E, Aras MA, Jordan C, Fleischmann KE, Melisko M, Qasim A, Shah SJ, Bajcsy R, Deo RC. Fully Automated Echocardiogram Interpretation in Clinical Practice. *Circulation*. 2018 Oct 16;138(16):1623-1635. doi: 10.1161/CIRCULATIONAHA.118.034338. PMID: 30354459; PMCID: PMC6200386.

B

Prediction of echo views for individual images

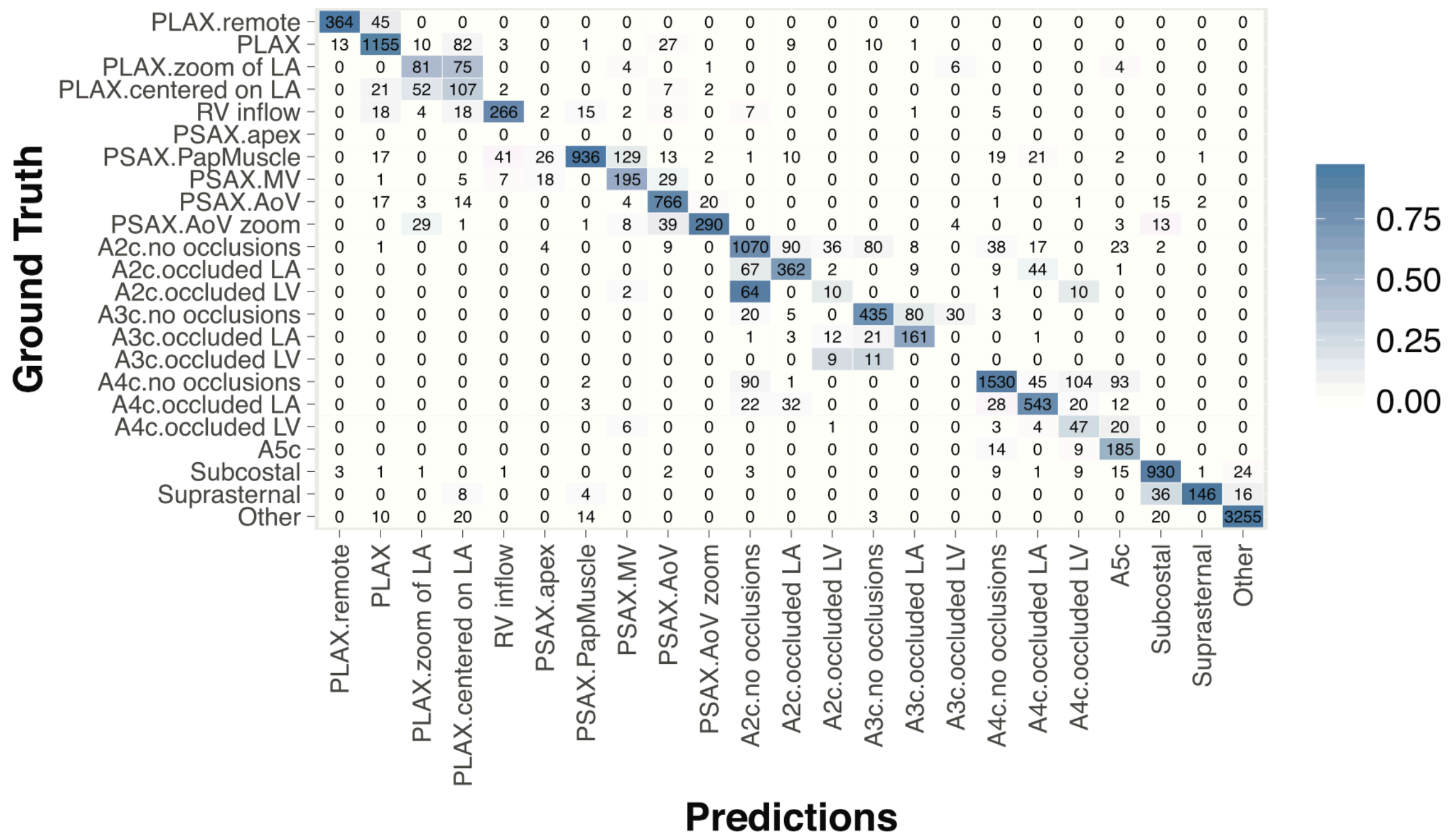


Figure 2. Convolutional neural networks successfully discriminate echocardiographic views.

A, t-Distributed Stochastic Neighbor Embedding (t-SNE) visualization of view classification. t-SNE is an algorithm used to visualize high-dimensional data in lower dimensions. It depicts the successful grouping of test images corresponding to 23 different echocardiographic views. Echocardiographic still images indicate the distinct clustering of images of A4c views without occlusions and those with occlusion of the left atrium. **B**, Confusion matrix demonstrating successful and unsuccessful view classifications within the test data set. Numbers along the diagonal represent successful classifications, whereas off-diagonal entries are misclassifications. A2c indicates apical 2-chamber; A3c, apical 3-chamber; A4c, apical 4-chamber; echo, echocardiogram; LV, left ventricular; and PLAX, parasternal long axis.

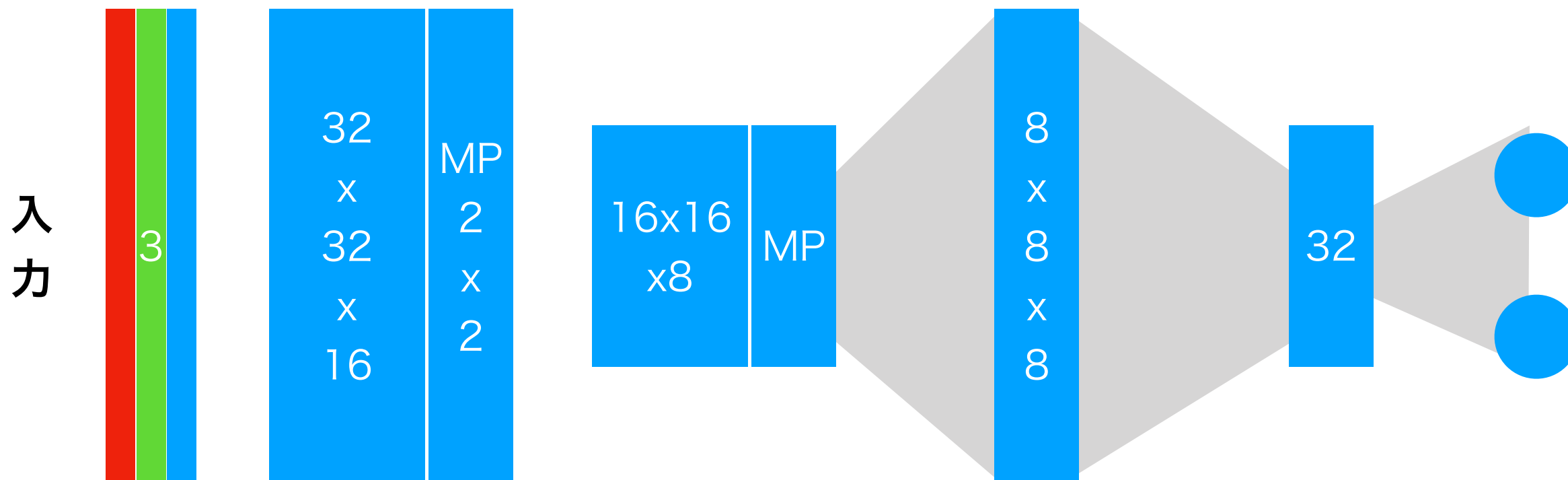
Zhang J, Gajjala S, Agrawal P, Tison GH, Hallock LA, Beussink-Nelson L, Lassen MH, Fan E, Aras MA, Jordan C, Fleischmann KE, Melisko M, Qasim A, Shah SJ, Bajcsy R, Deo RC. Fully Automated Echocardiogram Interpretation in Clinical Practice. *Circulation*. 2018 Oct 16;138(16):1623-1635. doi: 10.1161/CIRCULATIONAHA.118.034338. PMID: 30354459; PMCID: PMC6200386.

モデルの構築

構築するモデル

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1) # 32x32x3ch -> 16x16x16
        self.conv2 = nn.Conv2d(16, 8, kernel_size=3, padding=1) # 16x16x16 -> 8x8x8
        self.fc1 = nn.Linear(8 * 8 * 8, 32) # 8x8x8 -> 32
        self.fc2 = nn.Linear(32, 2) # 32 -> 2

    def forward(self, x):
        out = F.max_pool2d(torch.relu(self.conv1(x)), 2)
        out = F.max_pool2d(torch.relu(self.conv2(out)), 2)
        out = out.view(-1, 8 * 8 * 8)
        out = torch.tanh(self.fc1(out))
        out = self.fc2(out)
        return out
```



構築するモデル

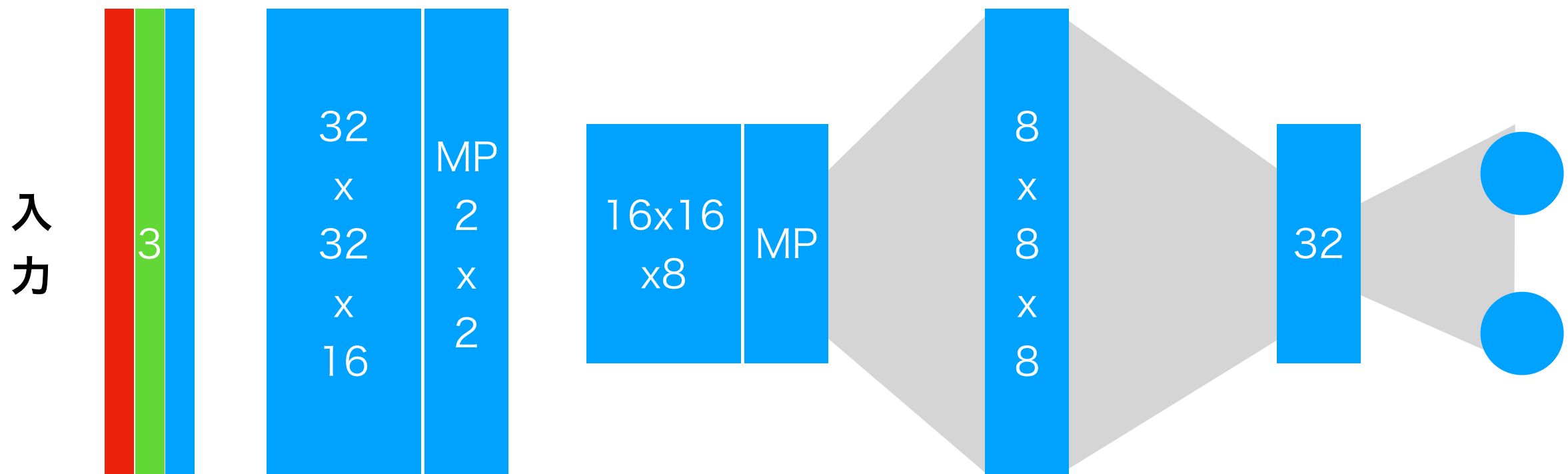
```
class Net(nn.Module):
```

```
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(16, 8, kernel_size=3, padding=1)
        self.fc1 = nn.Linear(8 * 8 * 8, 32)
        self.fc2 = nn.Linear(32, 2)
```

部品を書く部分

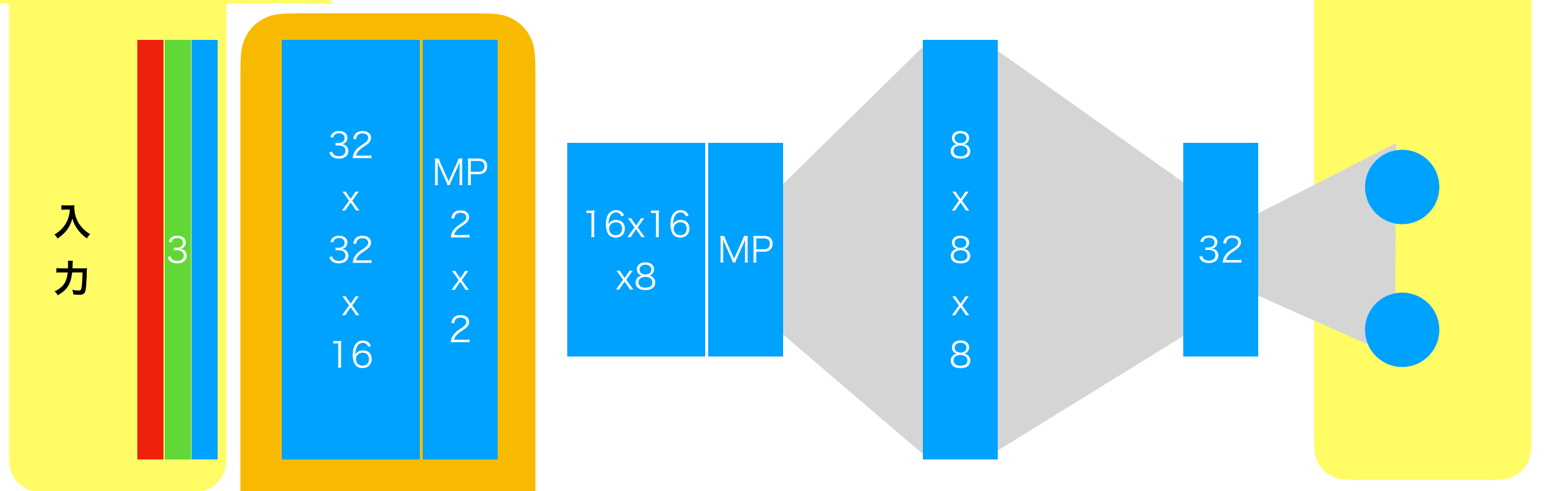
```
    def forward(self, x):
        out = F.max_pool2d(torch.relu(self.conv1(x)), 2)
        out = F.max_pool2d(torch.relu(self.conv2(out)), 2)
        out = out.view(-1, 8 * 8 * 8)
        out = torch.tanh(self.fc1(out))
        out = self.fc2(out)
        return out
```

処理順を書く部分



構築するモデル

画像が入力される
32x32x3chにする



`nn.Conv2d(3, 16, kernel_size=3, padding=1)`
3chの入力を3x3のカーネルで16chにする
画素数はそのまま
カーネルのパラメタは学習で決定する

そのあとMaxPoolingで2x2画素の最大値を抽出する
2x2x16の中から16種類の最大値が抽出される
その結果16x16画素, 16chになる

構築するモデル

画像が入力される
32x32x3chにする

入力

3

32
x
32
x
16

MP
2
x
2

16x16
x8

MP

8
x
8
x
8

32

2分類なので
出力は2つ

```
nn.Conv2d(16, 8, kernel_size=3, padding=1)
```

16chの入力を3x3のカーネルで8chにする

画素数はそのまま

カーネルのパラメタは学習で決定する

そのあとMaxPoolingで2x2画素の最大値を抽出する

2x2x8の中から8種類の最大値が抽出される

その結果8x8画素, 8chになる

構築するモデル

画像が入力される
32x32x3chにする

入力

3

32
x
32
x
16

MP
2
x
2

16x16
x8

MP

8
x
8
x
8

32

2分類なので
出力は2つ

```
nn.Linear(8 * 8 * 8, 32)
```

8x8画素, 8chになった画像を平坦に並べる

それを全結合型ニューラルネットワークの入力にする

その出力を32ノードに接続する

構築するモデル

画像が入力される
32x32x3chにする

入力

3

32
x
32
x
16

MP
2
x
2

16x16
x8

MP

8
x
8
x
8

32

2分類なので
出力は2つ

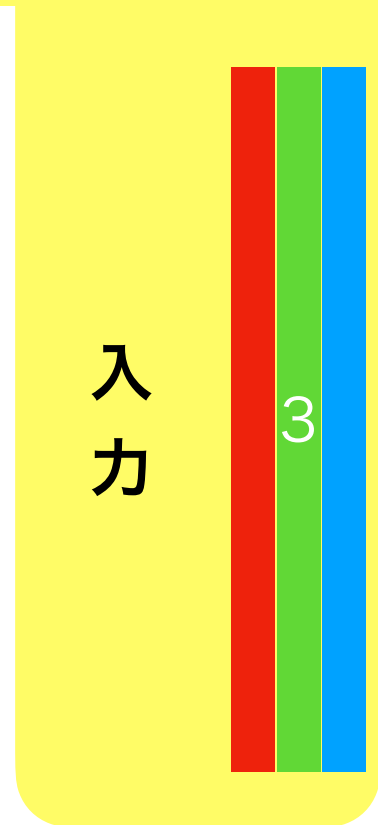
`nn.Linear(32, 2)`

32特徴を
ニューラルネットワークの入力にする

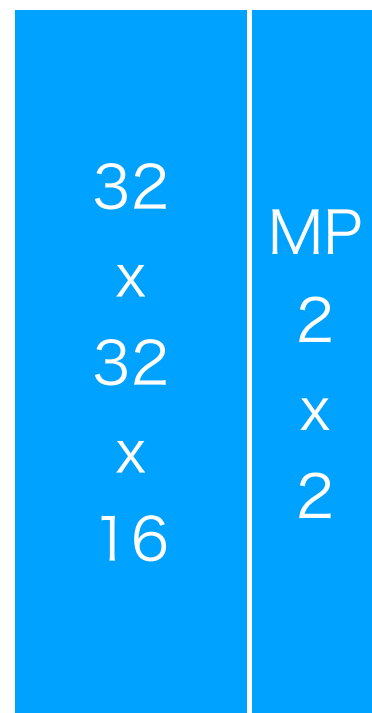
その出力を2ノードに接続する

構築するモデル

画像が入力される
32x32x3chにする

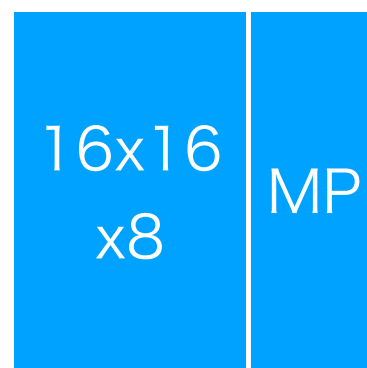


RGB画像が入力されて



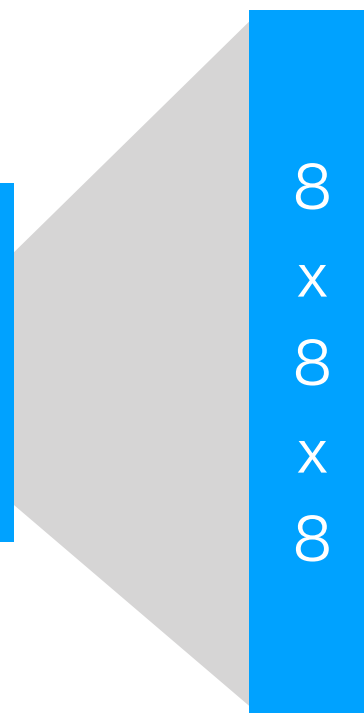
16のフィルタで
32x32x16特徴に分解されて

MaxPoolingで
16x16x16に削減されて



8のフィルタで
16x16x8特徴に分解されて

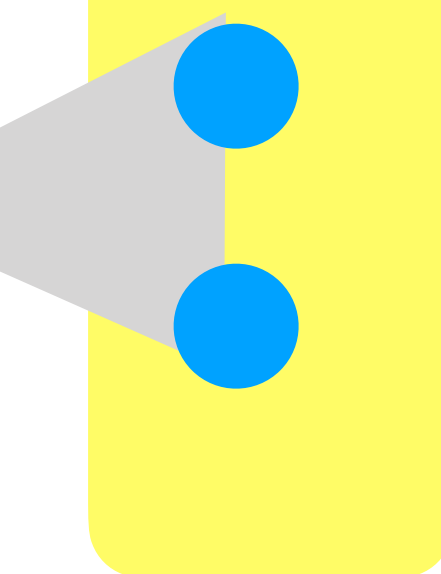
MaxPoolingで
8x8x8に削減されて



8x8x8が
平坦に並べられて



32ノードに
接続されて



2分類なので
出力は2つ

2ノードで
出力

構築するモデル

```
class Net(nn.Module):
```

```
    def __init__(self):
```

```
        super(Net, self).__init__()
```

```
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1)
```

```
        self.conv2 = nn.Conv2d(16, 8, kernel_size=3, padding=1)
```

```
        self.fc1 = nn.Linear(8 * 8 * 8, 32)
```

```
        self.fc2 = nn.Linear(32, 2)
```

部品を書く部分

```
    def forward(self, x):
```

```
        out = F.max_pool2d(torch.relu(self.conv1(x)), 2)
```

```
        out = F.max_pool2d(torch.relu(self.conv2(out)), 2)
```

```
        out = out.view(-1, 8 * 8 * 8)
```

```
        out = torch.tanh(self.fc1(out))
```

```
        out = self.fc2(out)
```

```
        return out
```

処理順を書く部分

学習済みモデルの利用

- 大規模な画像データベースを利用して学習済みのモデルを利用する
- 「学習済み」とは、モデルの重みが決定されていること
- 大規模なデータでの学習は時間がかかるので重みが公開されている場合が多い

学習済みモデルの利用

(常にうまくいくとはかぎりません！)

- **転移学習**

別の分野で学習されたデータを積極的に利用する方法

例：自然画像で学習されたモデルを医用画像に適用する

モデルの構造の出力層を変更する

出力層に近い重みを更新して実現

- **ファインチューニング**

別の分野で学習されたデータを初期値として利用する方法

例：A社製画像装置で学習したモデルをB社製に適用する

モデルの構造の出力層を変更する

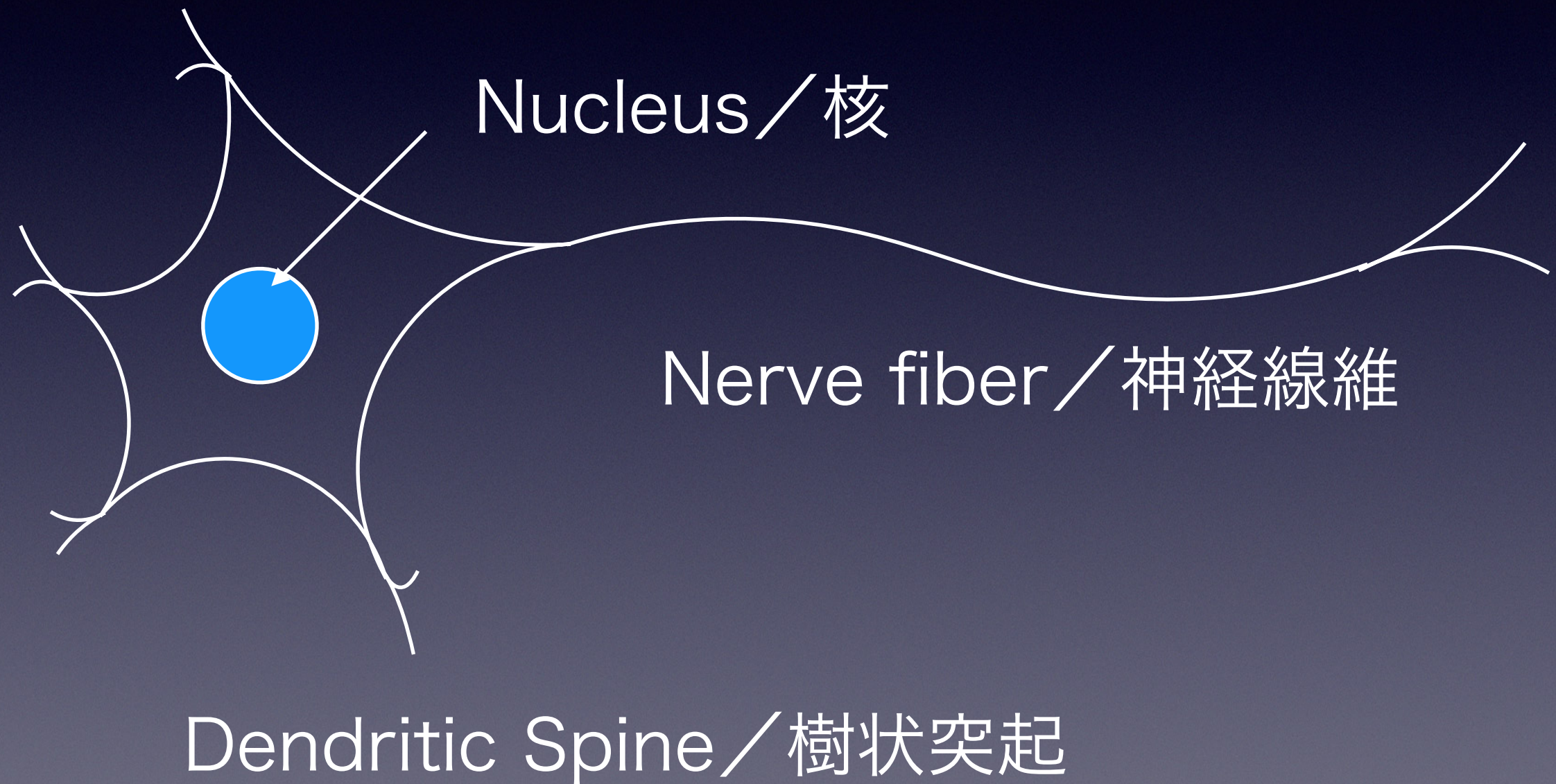
出力層に近い重みのみならず全体を更新して実現

- **知識蒸留**

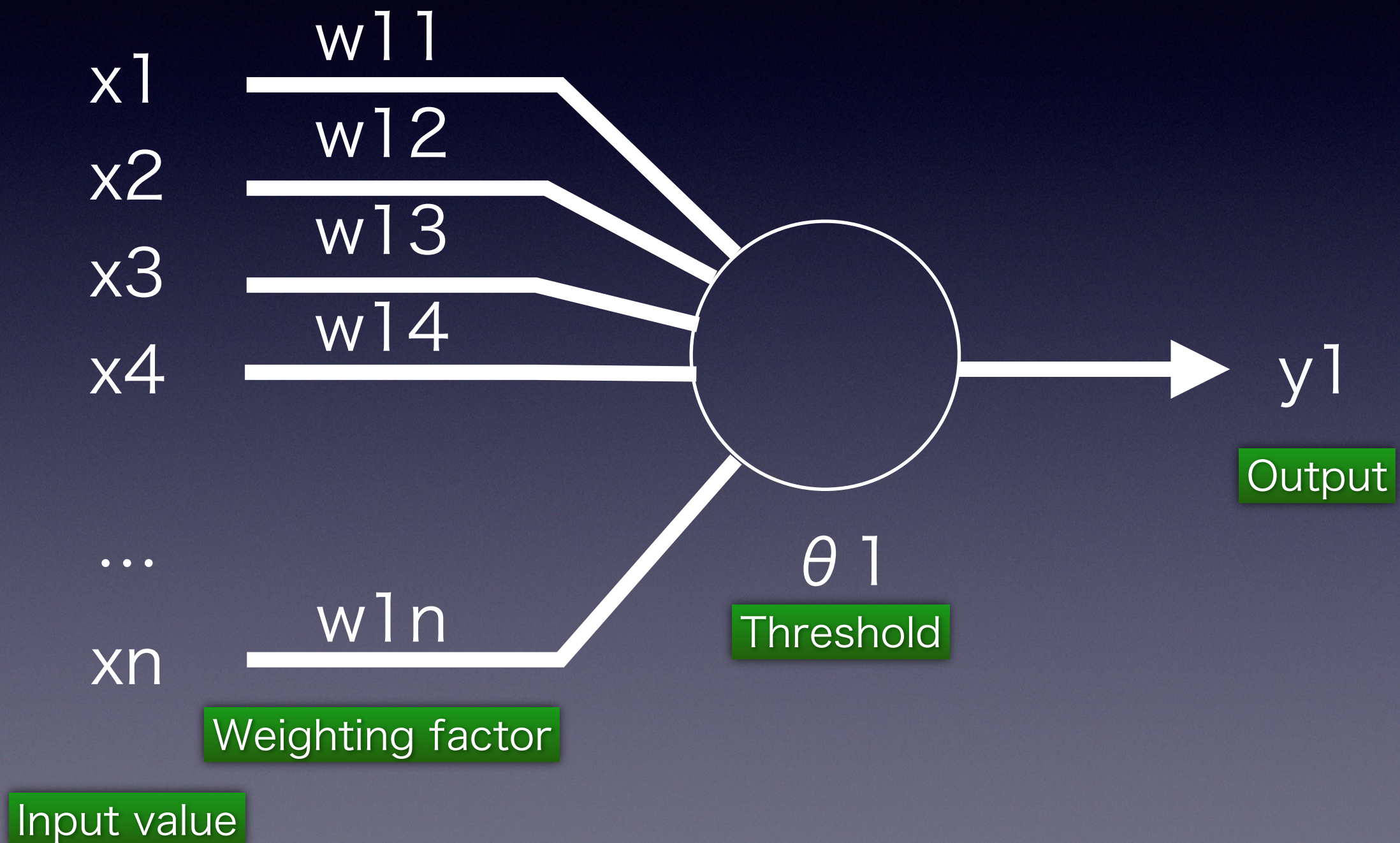
すでにある学習済みモデルと同じ出力が得られるように

元よりも小さなモデルを作ること（今回は取り扱いません）

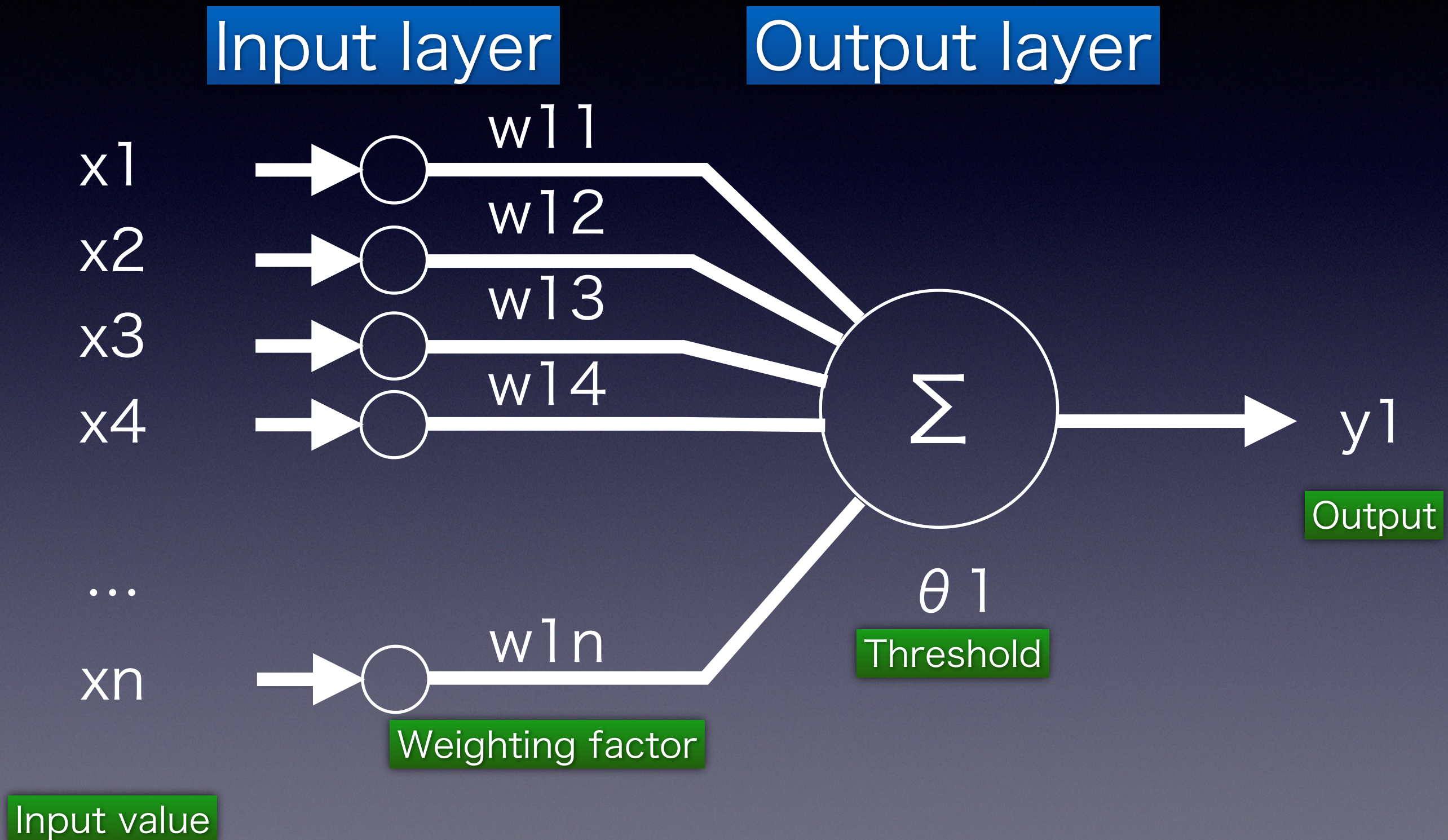
Perceptron



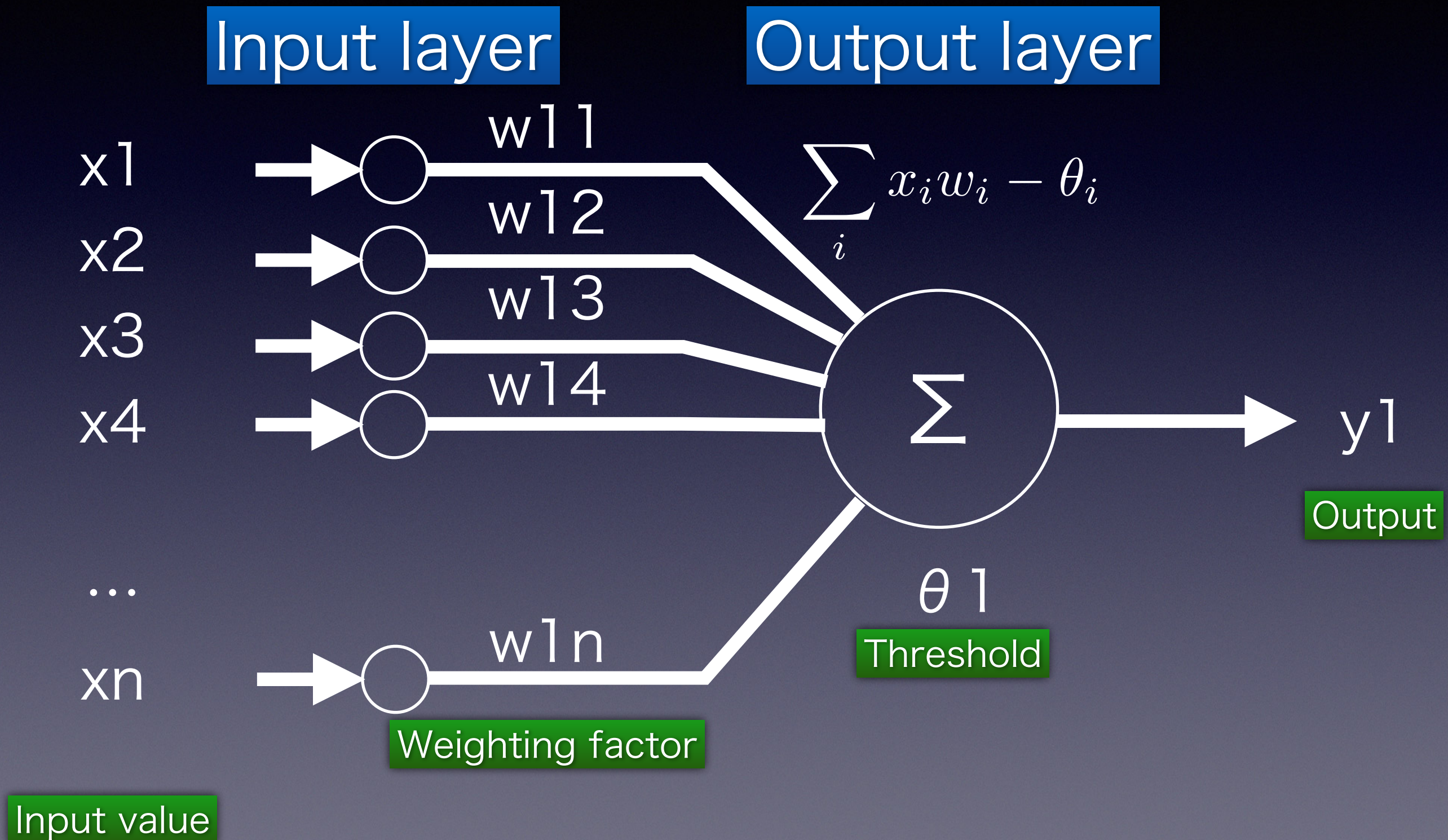
Neuron Model



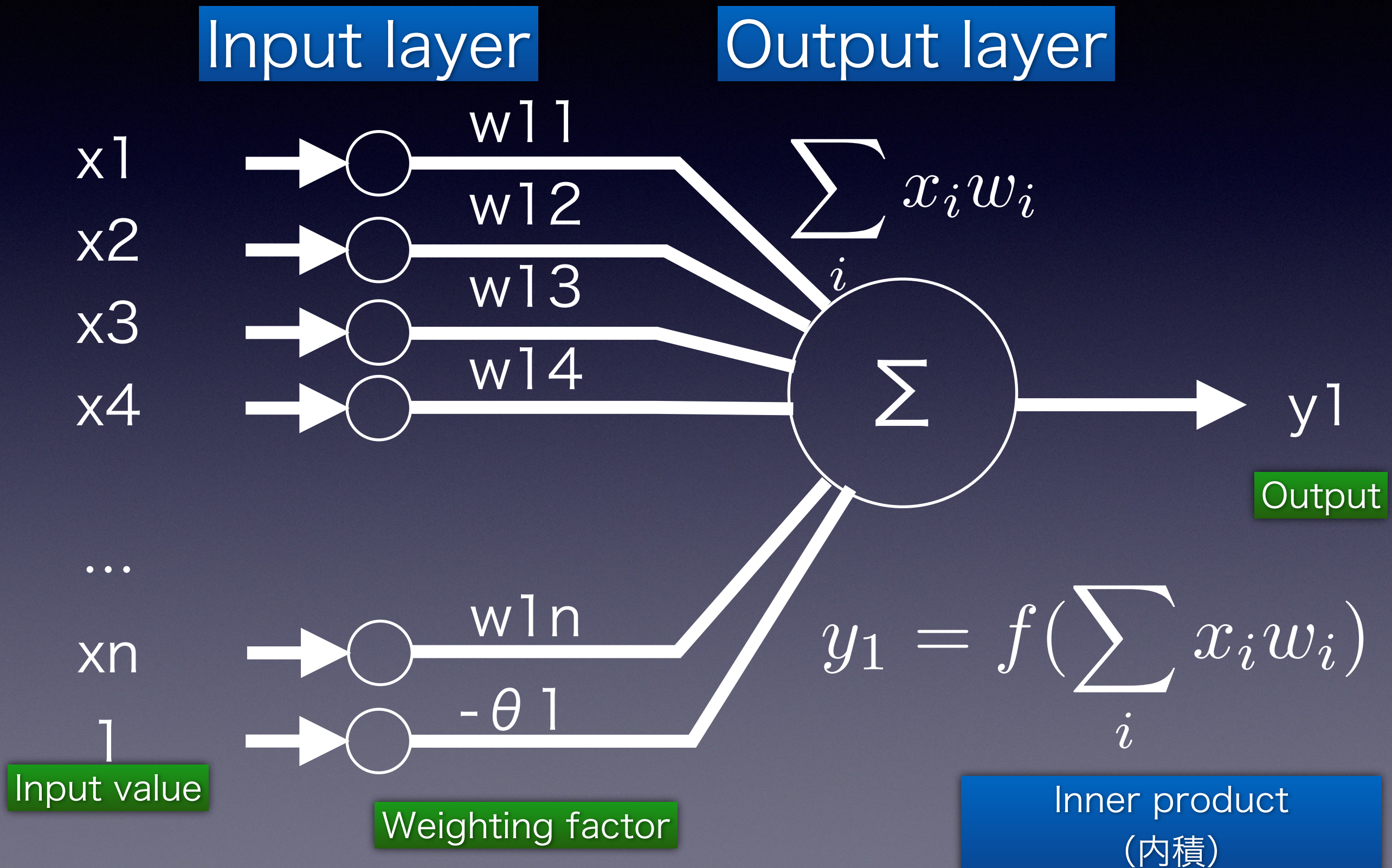
Simple Perceptron

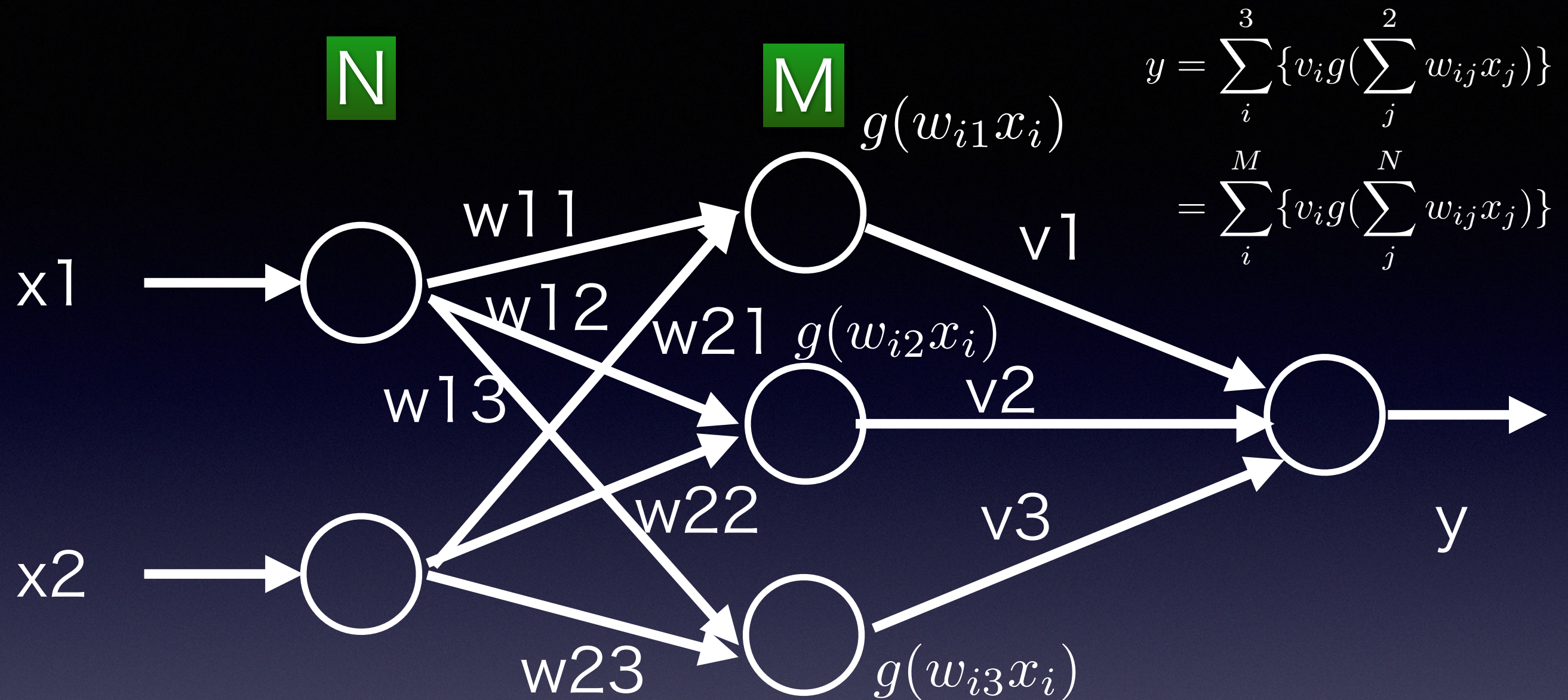


Simple Perceptron



Simple Perceptron





$$y = \sum_i^3 \{v_i g(\sum_j^2 w_{ij} x_j)\}$$

$$= \sum_i^M \{v_i g(\sum_j^N w_{ij} x_j)\}$$

$$g(wx) = \frac{1}{1 + \exp(-\beta wx)}$$

Sigmoid Function

$$\neq w_1 x_1 + w_1 x_1 + \dots + w_n x_n$$

**Fine Tuningの
2クラス問題を
やってみましょう**


```
1  # 学習を行うためには requires_grad = Trueである必要があります
2  # このままですべて学習ができます
3
4  for name, param in model_ft.named_parameters():
5      # print(name)
6      # print(param)
7      print(name, param.requires_grad)
```

```
conv1.weight True
bn1.weight True
bn1.bias True
layer1.0.conv1.weight True
layer1.0.bn1.weight True
layer1.0.bn1.bias True
layer1.0.conv2.weight True
layer1.0.bn2.weight True
layer1.0.bn2.bias True
```

Transfer Learningの 2クラス問題を やってみましょう

```
1 # この2つだけが重要で あとはFalseにしたいよね
2 print(model_ft.fc.weight.requires_grad)
3 print(model_ft.fc.bias.requires_grad)
```

True
True

```
1 # 面倒なので全部Falseにします
2 for name, param in model_ft.named_parameters():
3     param.requires_grad = False
```

```
1 # ぜんぶFalseになったようです
2 for name, param in model_ft.named_parameters():
3     print(name, param.requires_grad)
```

conv1.weight False
bn1.weight False
bn1.bias False
layer1.0.conv1.weight False
layer1.0.bn1.weight False
layer1.0.bn1.bias False
layer1.0.conv2.weight False

```
1 # この2つだけが重要ですがFalseのようです
2 print(model_ft.fc.weight.requires_grad)
3 print(model_ft.fc.bias.requires_grad)
```

False
False

```
1 # なのでTrueにしてあげます
2 model_ft.fc.weight.requires_grad=True
3 model_ft.fc.bias.requires_grad=True
4
```

```
1 # 無事にTrueになりました
2
3 print(model_ft.fc.weight.requires_grad)
4 print(model_ft.fc.bias.requires_grad)
```

True
True

領域分割

ディープラーニングを評価

$x \longrightarrow y$



図形で評価

検出（箱型）

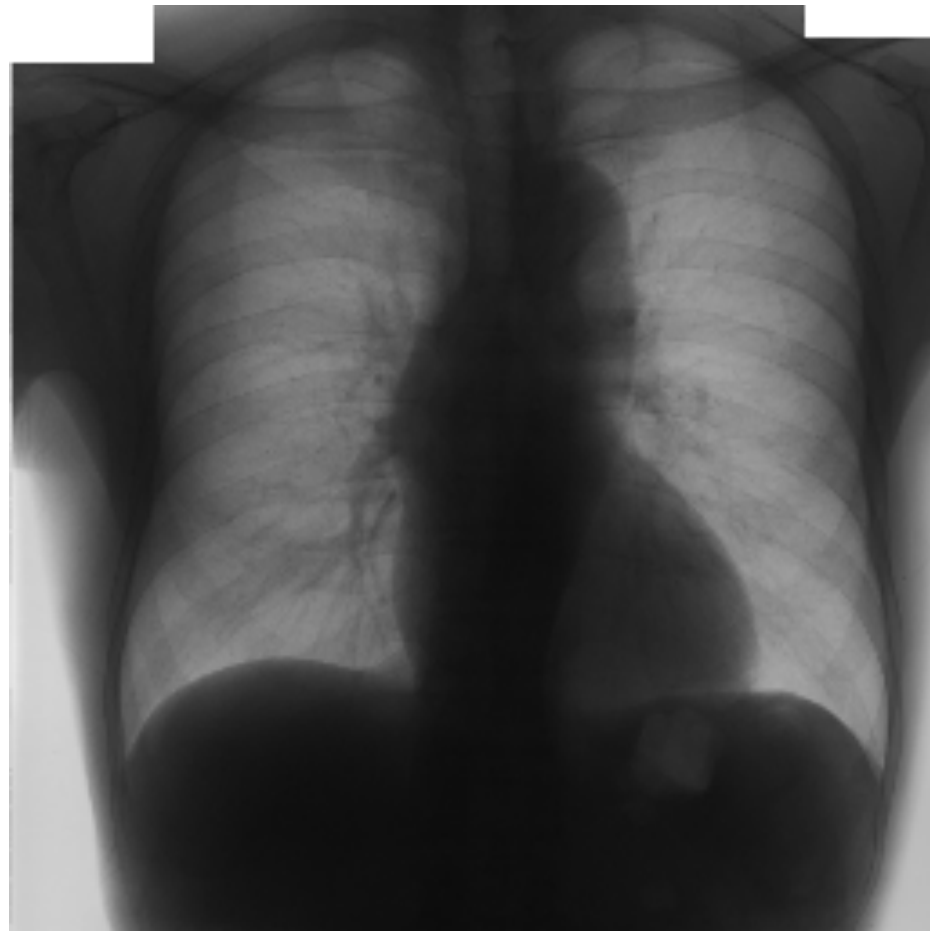
領域抽出（精密抽出）

この2つは
どのくらい重なる？

一致率を計測する

演習 画像の領域分割

肺野領域抽出



U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOS Centre for Biological Signalling Studies,
University of Freiburg, Germany

`ronneber@informatik.uni-freiburg.de`,

WWW home page: `http://lmb.informatik.uni-freiburg.de/`

Abstract. There is large consent that successful training of deep networks requires many thousand annotated training samples. In this paper, we present a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently. The architecture consists of a contracting path to capture

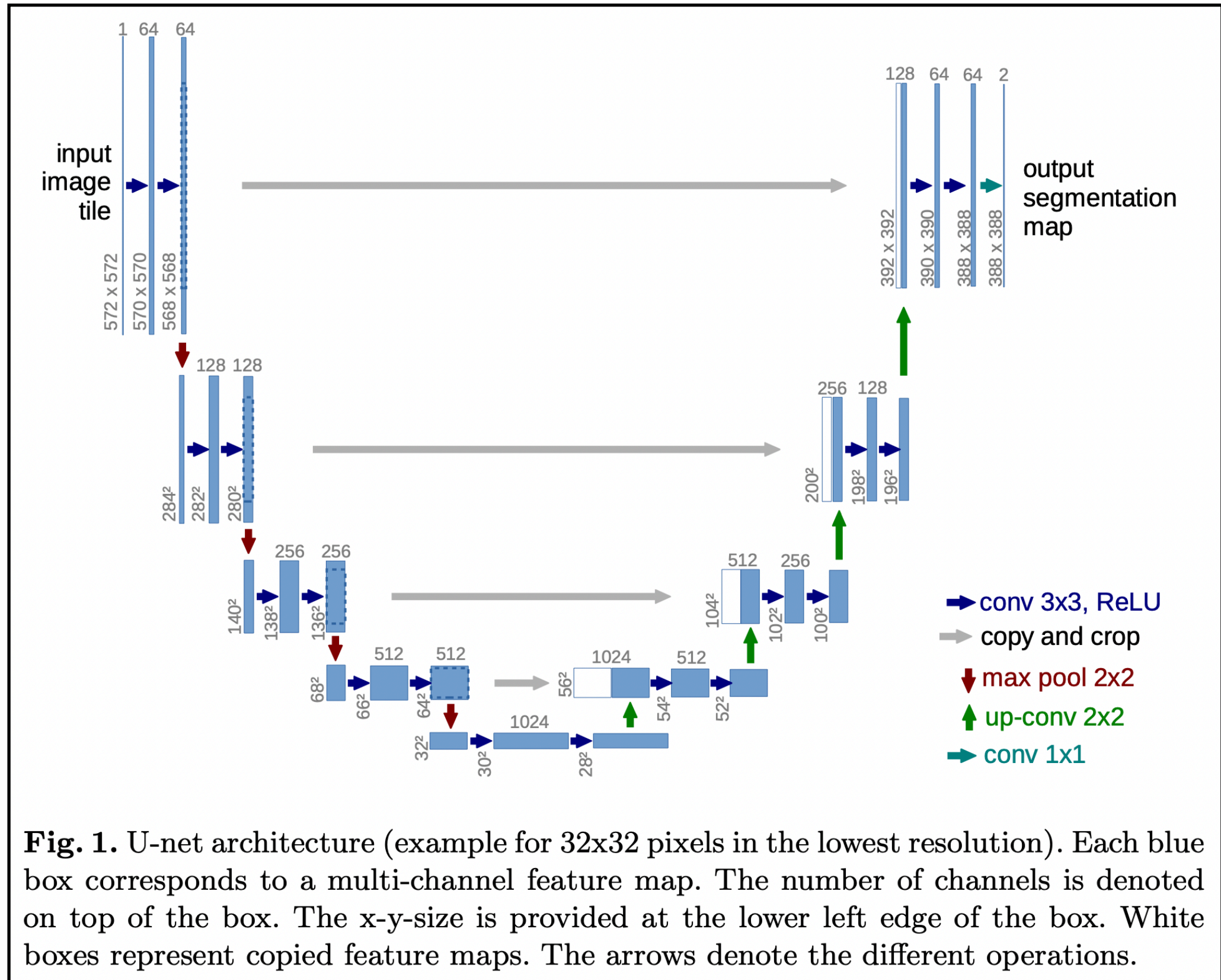


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

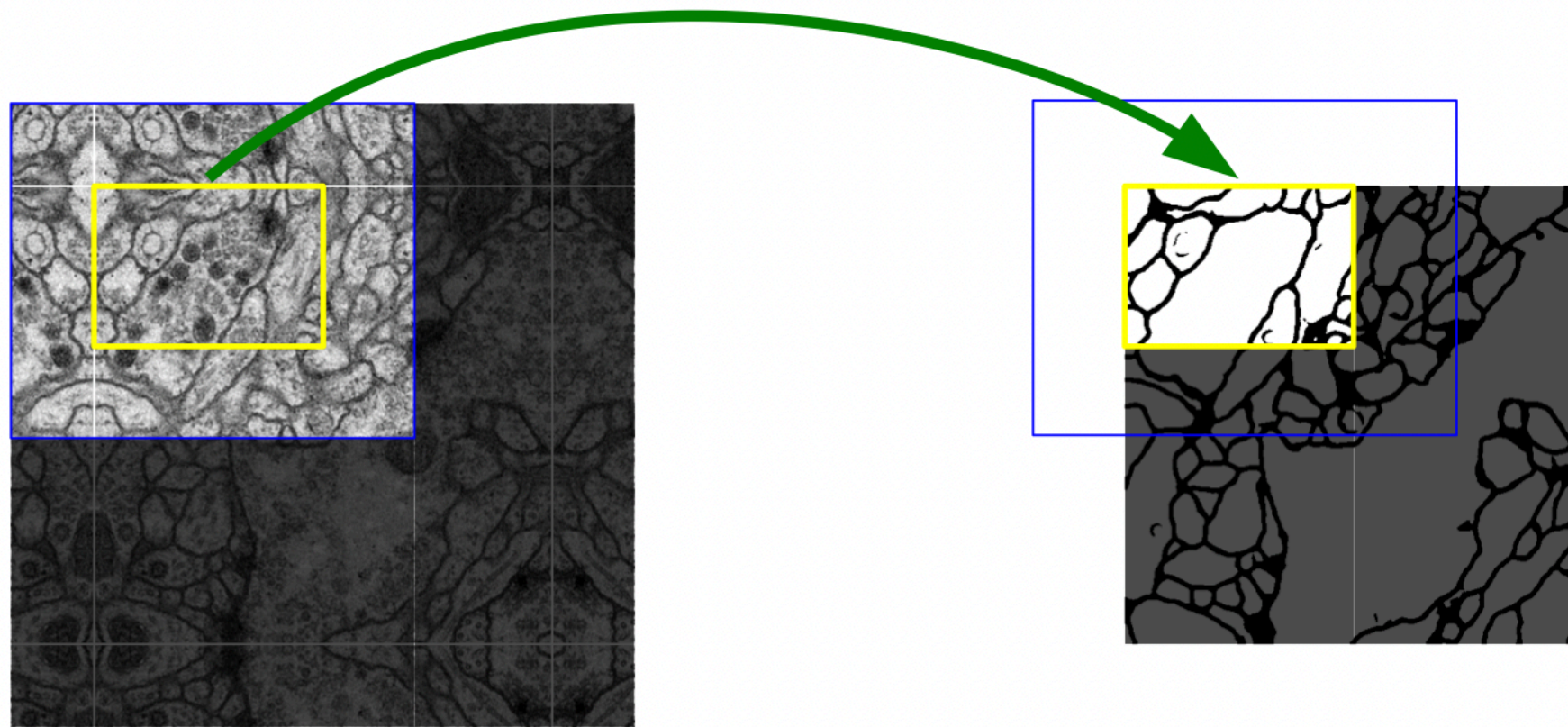


Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

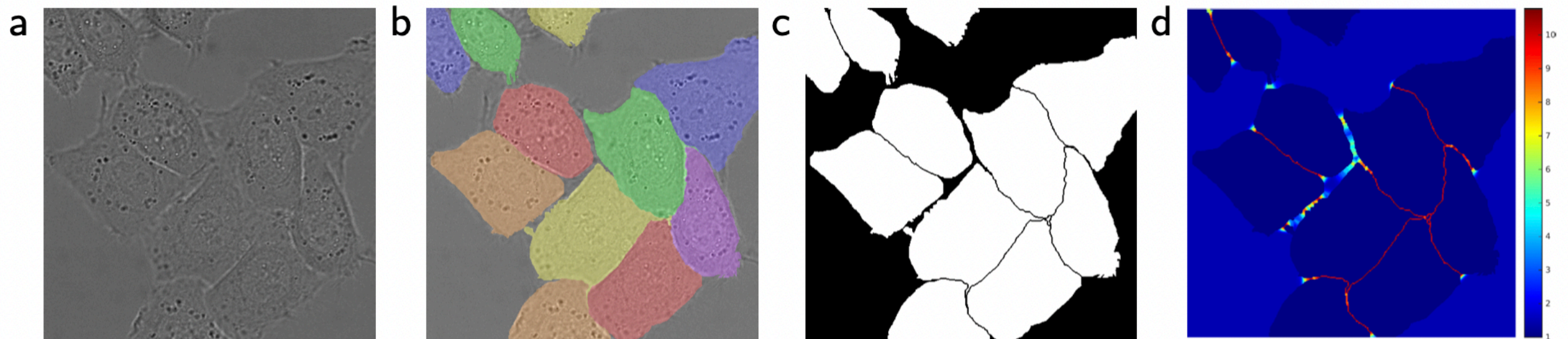


Fig. 3. HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (a) raw image. (b) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (c) generated segmentation mask (white: foreground, black: background). (d) map with a pixel-wise loss weight to force the network to learn the border pixels.

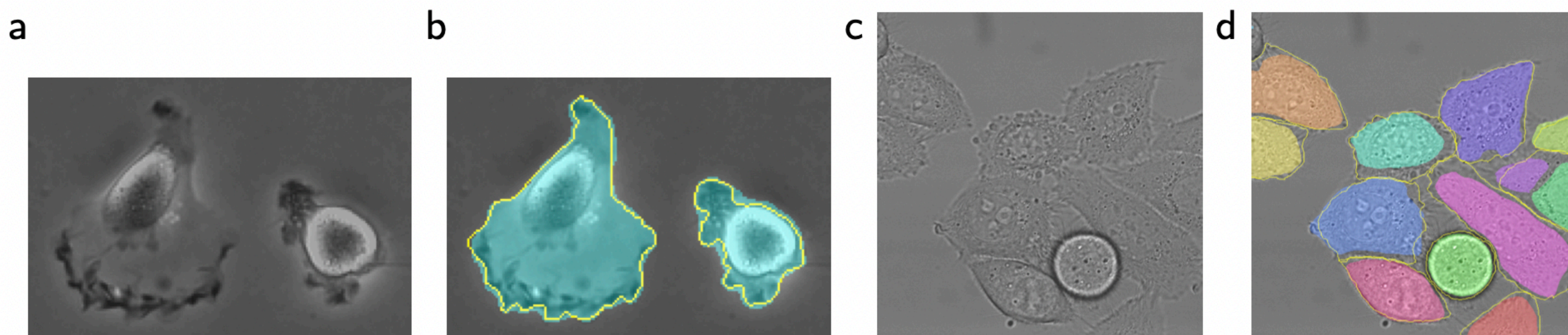


Fig. 4. Result on the ISBI cell tracking challenge. (a) part of an input image of the “PhC-U373” data set. (b) Segmentation result (cyan mask) with manual ground truth (yellow border) (c) input image of the “DIC-HeLa” data set. (d) Segmentation result (random colored masks) with manual ground truth (yellow border).

実際に動かしてみましよう

Image Input

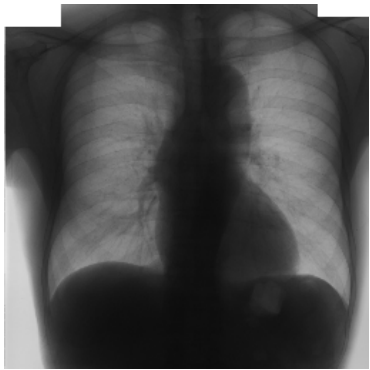


Image Output



Conv 2D

Batch
Normalization

Max
pooling

Concatenate

Up-
sampling

正解の作成方法

2 値画像を作る / 多値画像にする

ラベル番号に意味があるので可逆圧縮を使うこと

VoTT

<https://github.com/Microsoft/VoTT/releases>

PLUTO

[http://www.suenaga.cse.nagoya-u.ac.jp/wiki/index.php?
PLUTO%A5%D7%A5%E9%A5%B0%A5%A4%A5%F3](http://www.suenaga.cse.nagoya-u.ac.jp/wiki/index.php?PLUTO%A5%D7%A5%E9%A5%B0%A5%A4%A5%F3)

特集論文／CADの新展開

PLUTO：医用画像診断支援共通プラットフォーム

PLUTO: A common Platform for Computer-aided Diagnosis

二村 幸孝^{*1} 出口 大輔^{*2} 北坂 孝幸^{*1}

Yukitaka NIMURA Daisuke DEGUCHI Takayuki KITASAKA

森 健策^{*1} 末永 康仁^{*1}

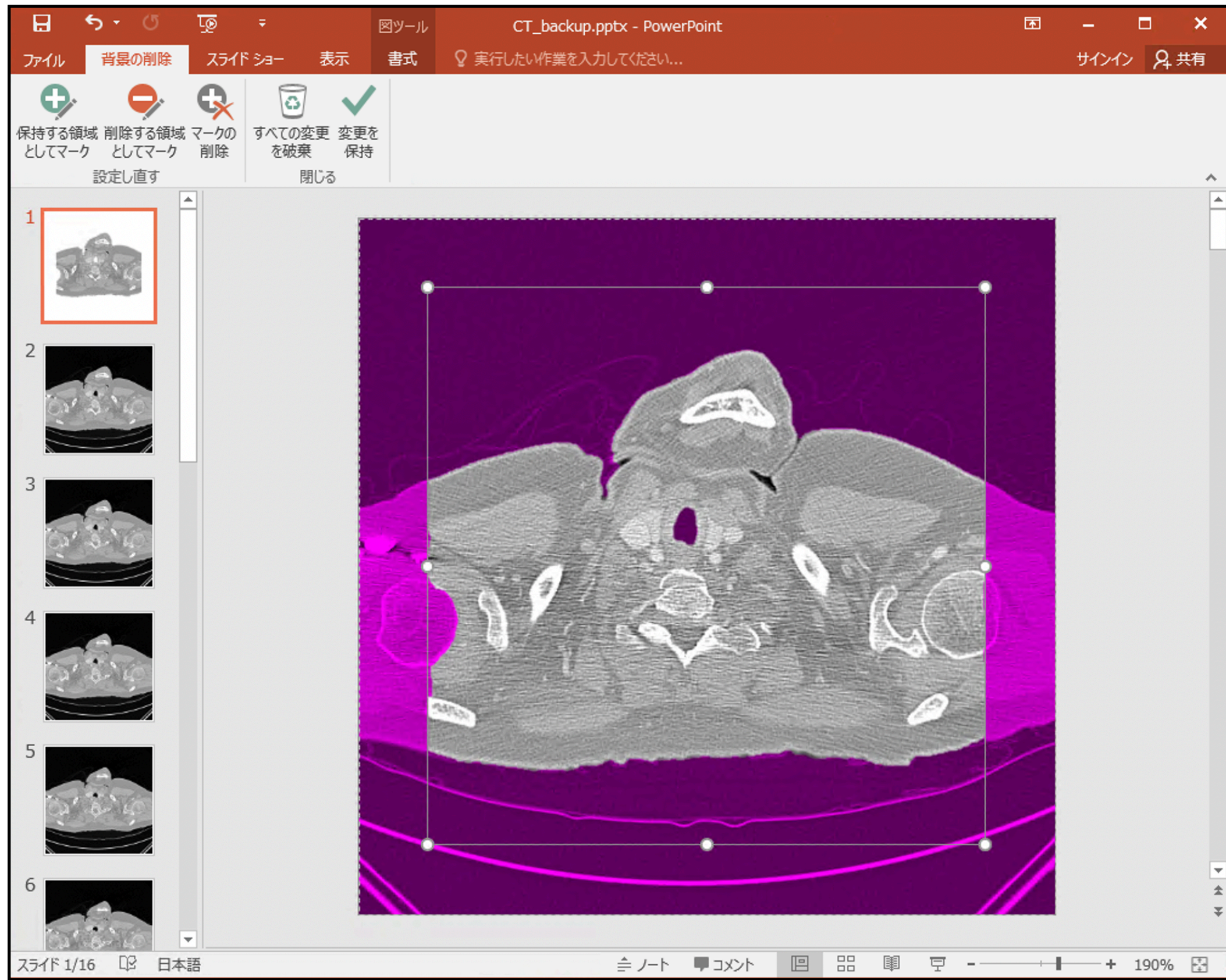
Kensaku MORI Yasuhito SUENAGA

要 旨

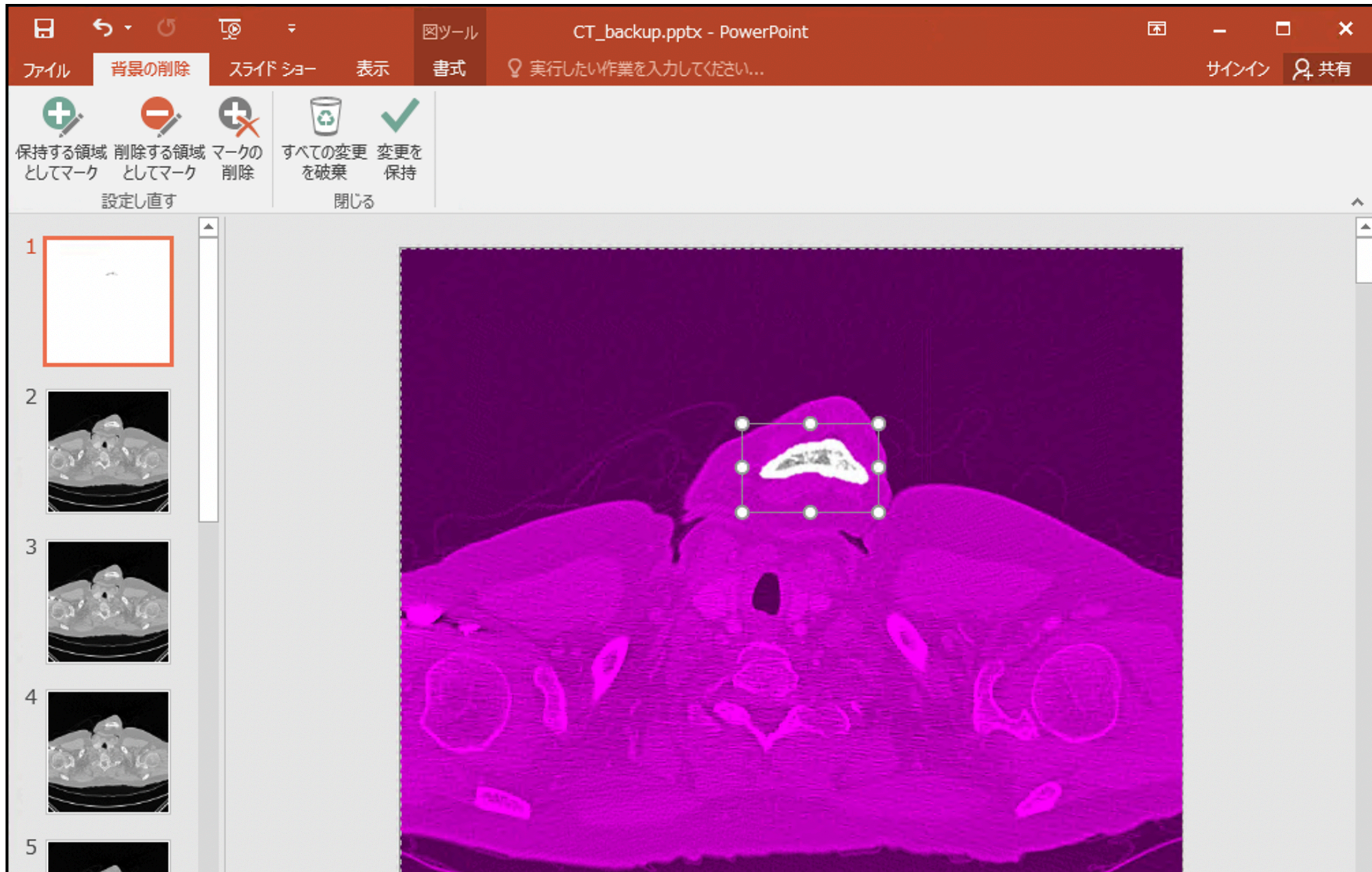
本稿では文部科学省科学研究費補助金特定領域研究「多次元医用画像の知的診断支援」において、多臓器・多疾病計算機支援医用画像診断システムを実現するための共通開発基盤および診断支援アルゴリズムの共有化のために開発された、医用画像診断支援共通プラットフォーム「PLUTO」について述べる。PLUTOは、診断支援システムを開発するための共通プラットフォームであり、多臓器・多疾病計算機支援医用画像診断システムが必要とするさまざまな診断支援機能を容易に実現できるように設計されている。ここでは、このPLUTOのソフトウェア構成とこれまでに実現されている機能について紹介する。

キーワード：計算機支援医用画像診断、共通プラットフォーム、多臓器、多疾病

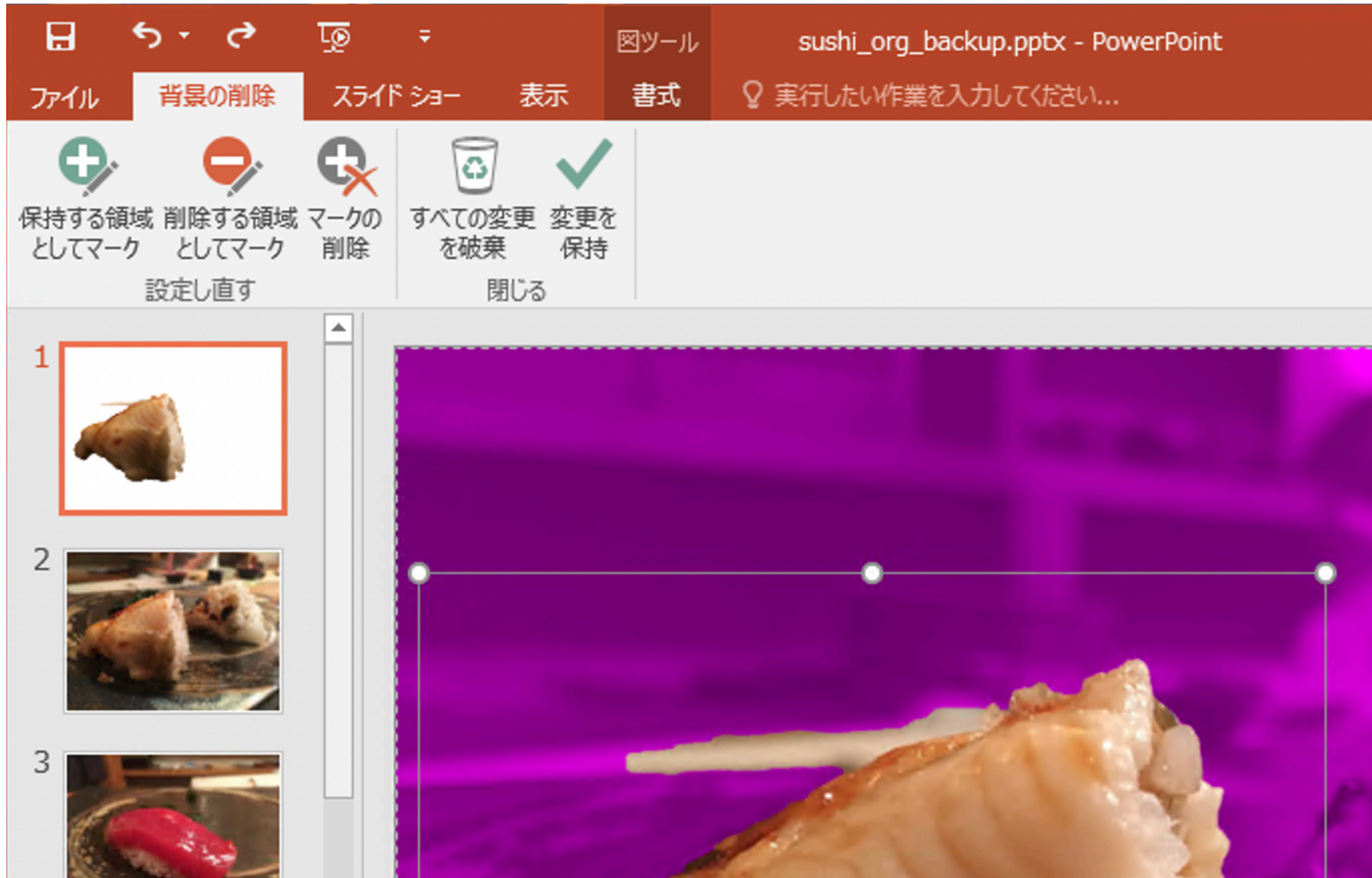
PowerPointも使える！



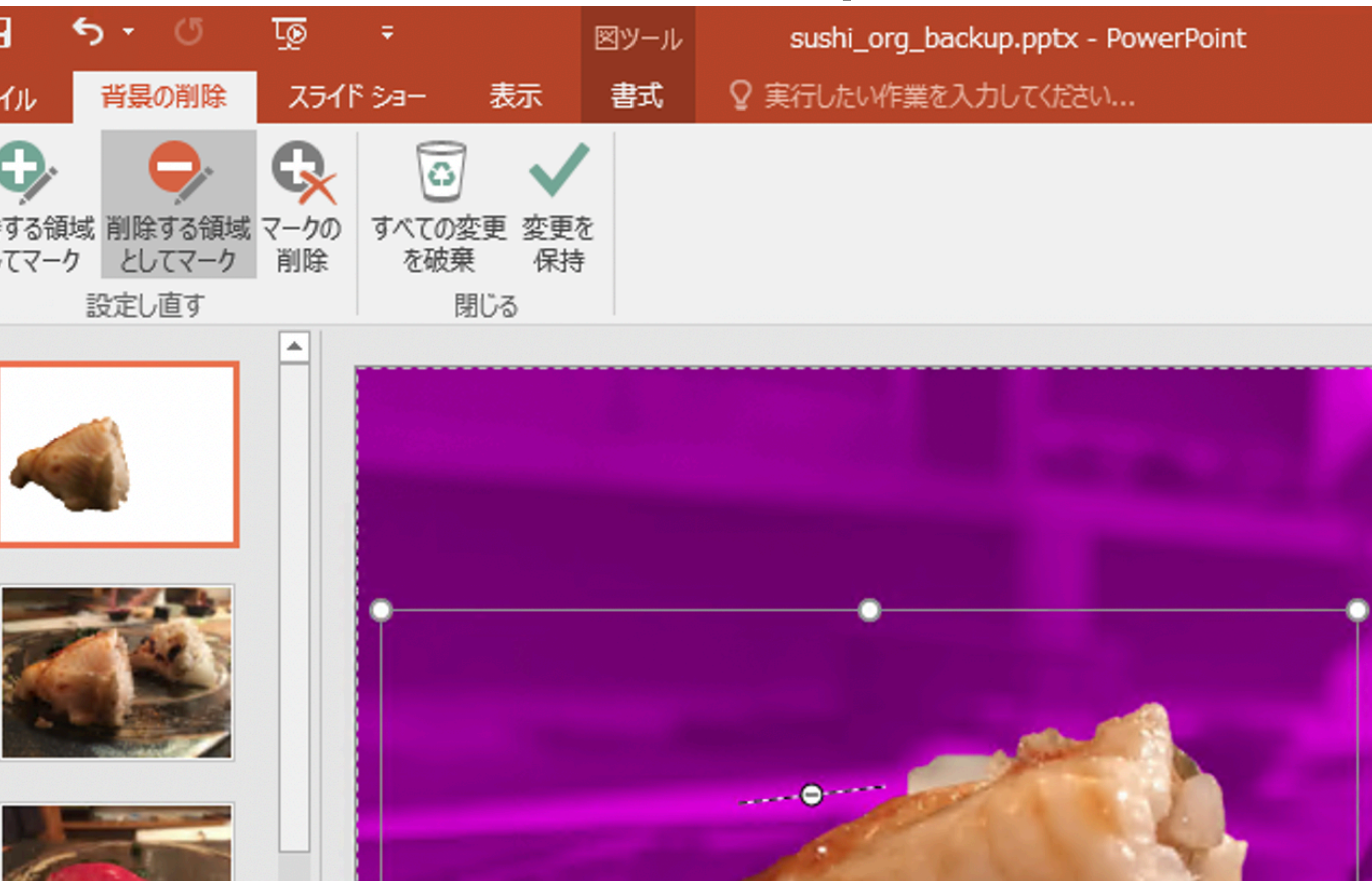
PowerPointも使える！



PowerPointも使える！



PowerPointも使える！



背景の削除

スライドショー

表示

書式

💡 実行したい作業を入力してください...



領域
ク 削除する領域
としてマーク



マークの
削除



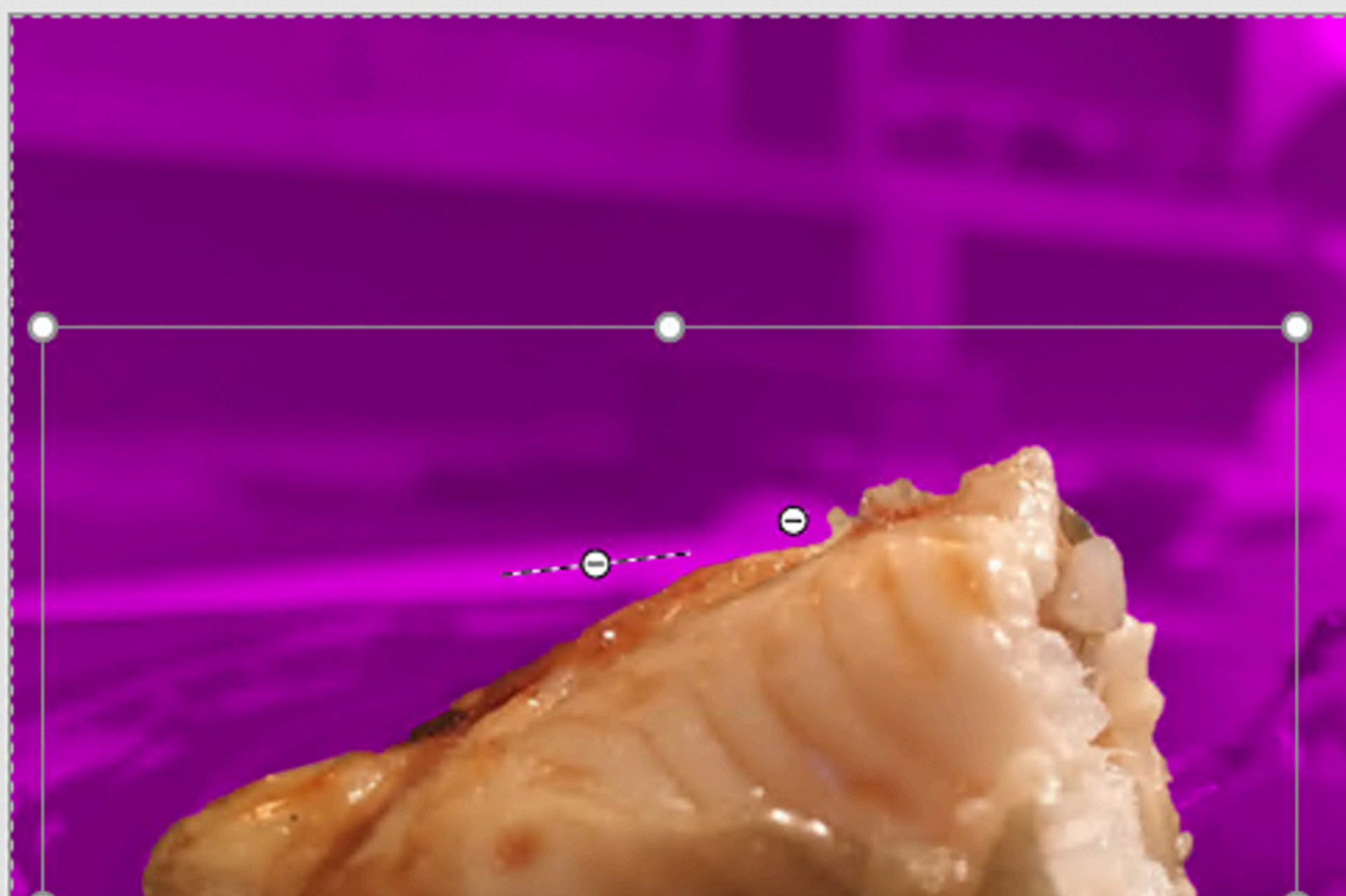
すべての変更
を破棄



変更を
保持

閉じる

設定し直す



“GrabCut” — Interactive Foreground Extraction using Iterated Graph Cuts

Carsten Rother*

Vladimir Kolmogorov[†]
Microsoft Research Cambridge, UK

Andrew Blake[‡]



Figure 1: **Three examples of GrabCut.** The user drags a rectangle loosely around an object. The object is then extracted automatically.

Abstract

The problem of efficient, interactive foreground/background segmentation in still images is of great practical importance in image editing. Classical image segmentation tools use either texture (colour) information, e.g. Magic Wand, or edge (contrast) information, e.g. Intelligent Scissors. Recently, an approach based on optimization by graph-cut has been developed which successfully combines both types of information. In this paper we extend the graph-cut approach in three respects. First, we have developed a more powerful, iterative version of the optimisation. Secondly, the power of the iterative algorithm is used to simplify substantially the

free of colour bleeding from the source background. In general, degrees of interactive effort range from editing individual pixels, at the labour-intensive extreme, to merely touching foreground and/or background in a few locations.

1.1 Previous approaches to interactive matting

In the following we describe briefly and compare several state of the art interactive tools for segmentation: Magic Wand, Intelligent Scissors, Graph Cut and Level Sets and for matting: Bayes Matting and Knockout. Fig. 2 shows their results on a matting task, together with degree of user interaction required to achieve those results.

Magic Wand

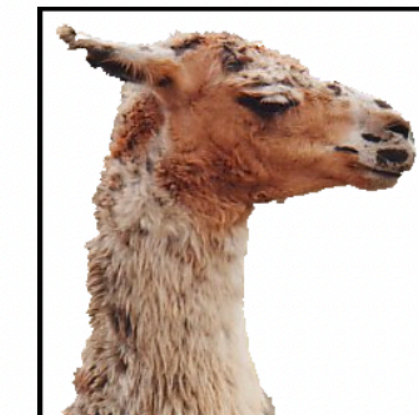
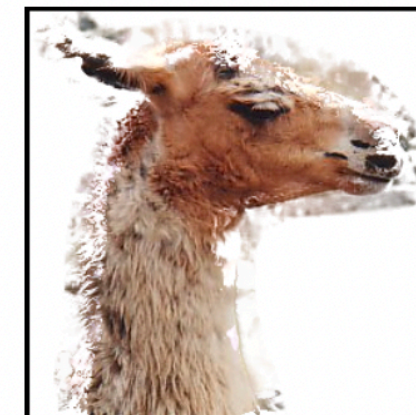
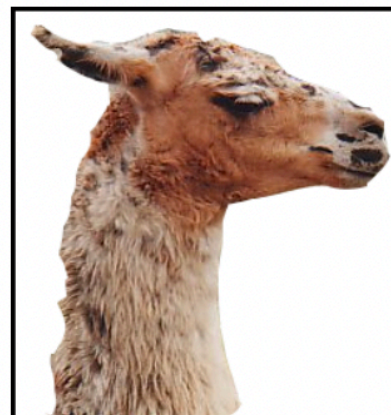
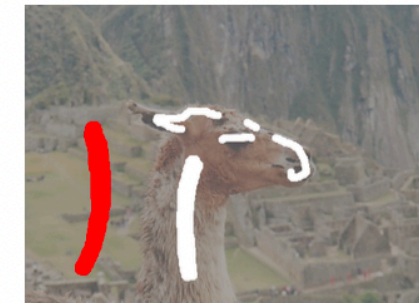
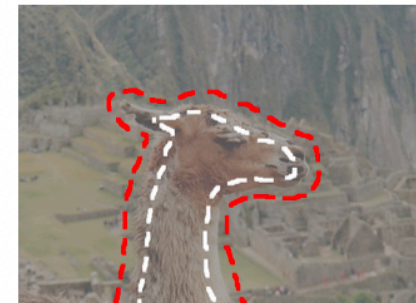
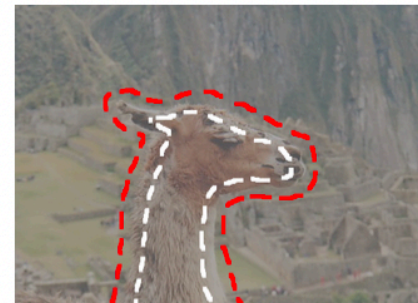
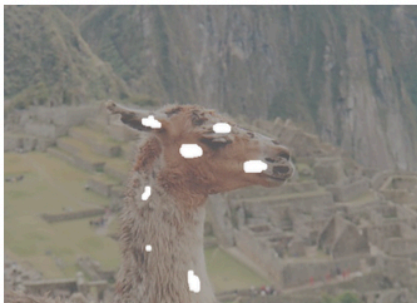
Intelligent Scissors

Bayes Matte

Knockout 2

Graph cut

GrabCut



(a)

(b)

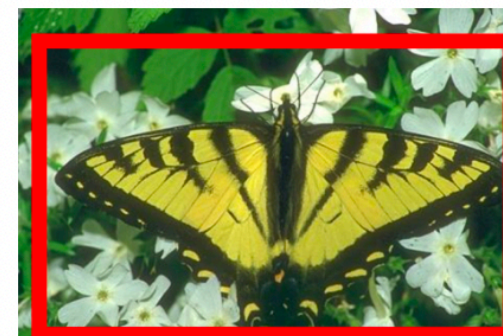
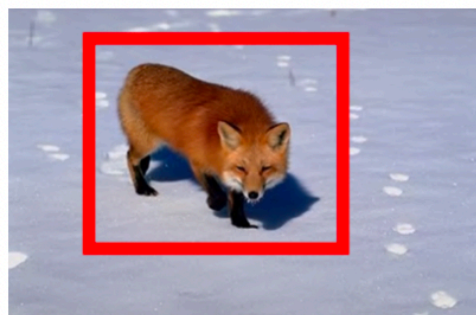
(c)

(d)

(e)

(f)

Figure 2: **Comparison of some matting and segmentation tools.** The top row shows the user interaction required to complete the segmentation or matting process: white brush/lasso (foreground), red brush/lasso (background), yellow crosses (boundary). The bottom row illustrates the resulting segmentation. GrabCut appears to outperform the other approaches both in terms of the simplicity of user input and the quality of results. Original images on the top row are displayed with reduced intensity to facilitate overlay; see fig. 1. for original. Note that our implementation of Graph Cut [Boykov and Jolly 2001] uses colour mixture models instead of grey value histograms.



No User
Interaction

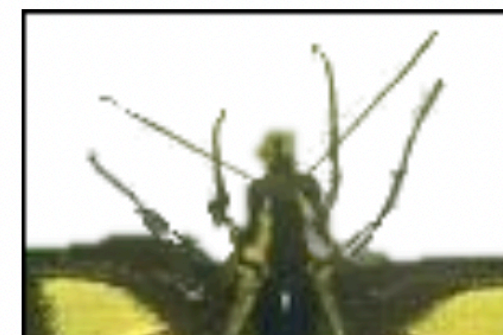
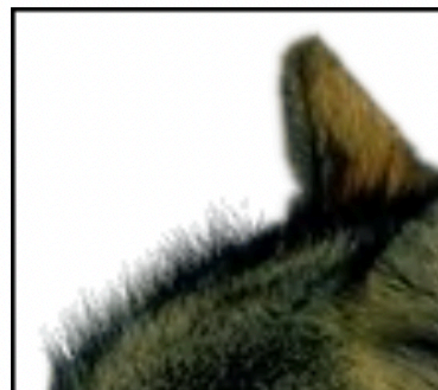
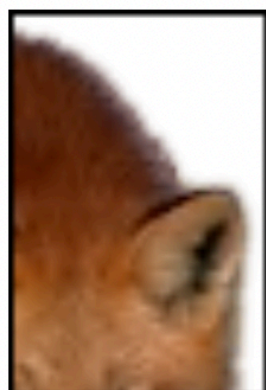
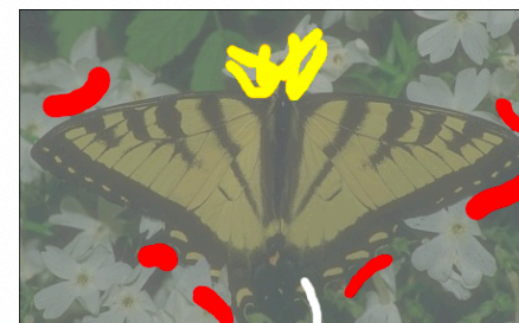
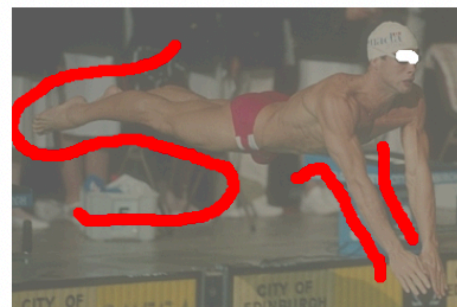
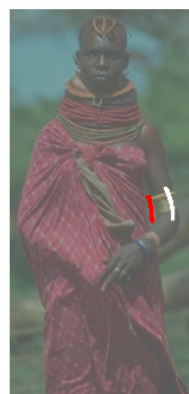


Figure 8: **Results using GrabCut**. The first row shows the original images with superimposed user input (red rectangle). The second row displays all user interactions: red (background brush), white (foreground brush) and yellow (matting brush). The degree of user interaction increases from left to right. The results obtained by GrabCut are visualized in the third row. The last row shows zoomed portions of the respective result which documents that the recovered alpha mattes are smooth and free of background bleeding.

<https://dl.acm.org/doi/10.1145/1186562.1015720>

評価方法

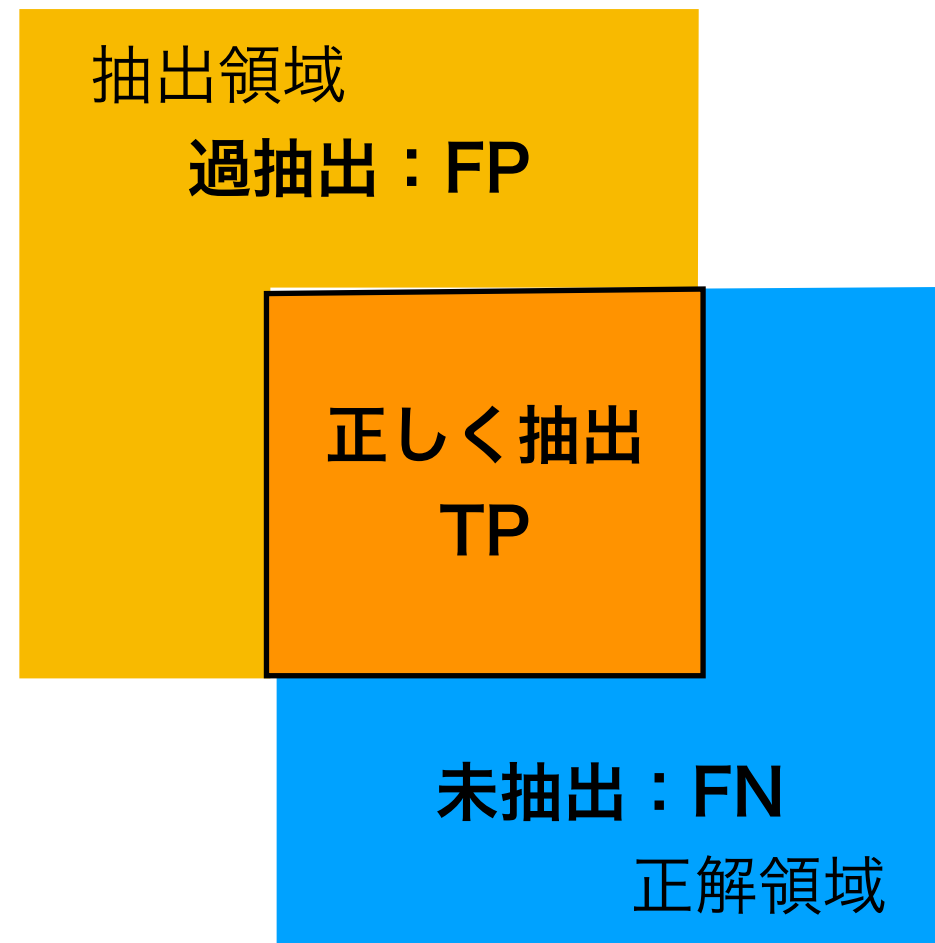
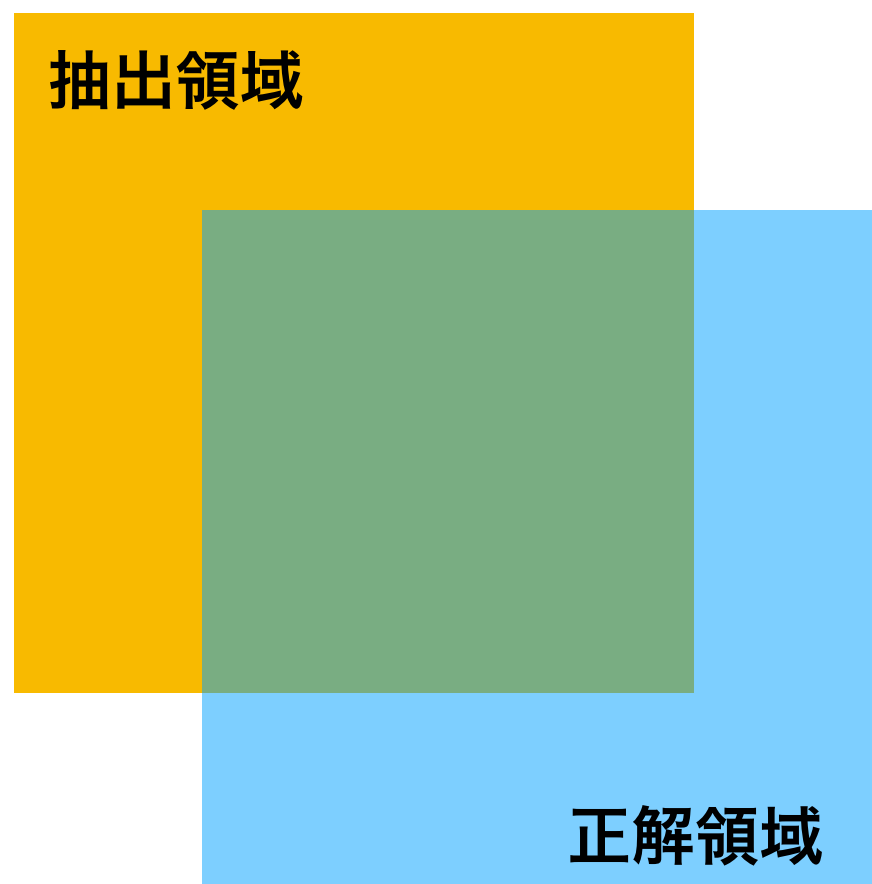
IoU

Precision

Recall

F1

評価方法：領域の評価



$$\text{IoU (Jaccard)} = \text{TP} / (\text{TP} + \text{FP} + \text{FN})$$

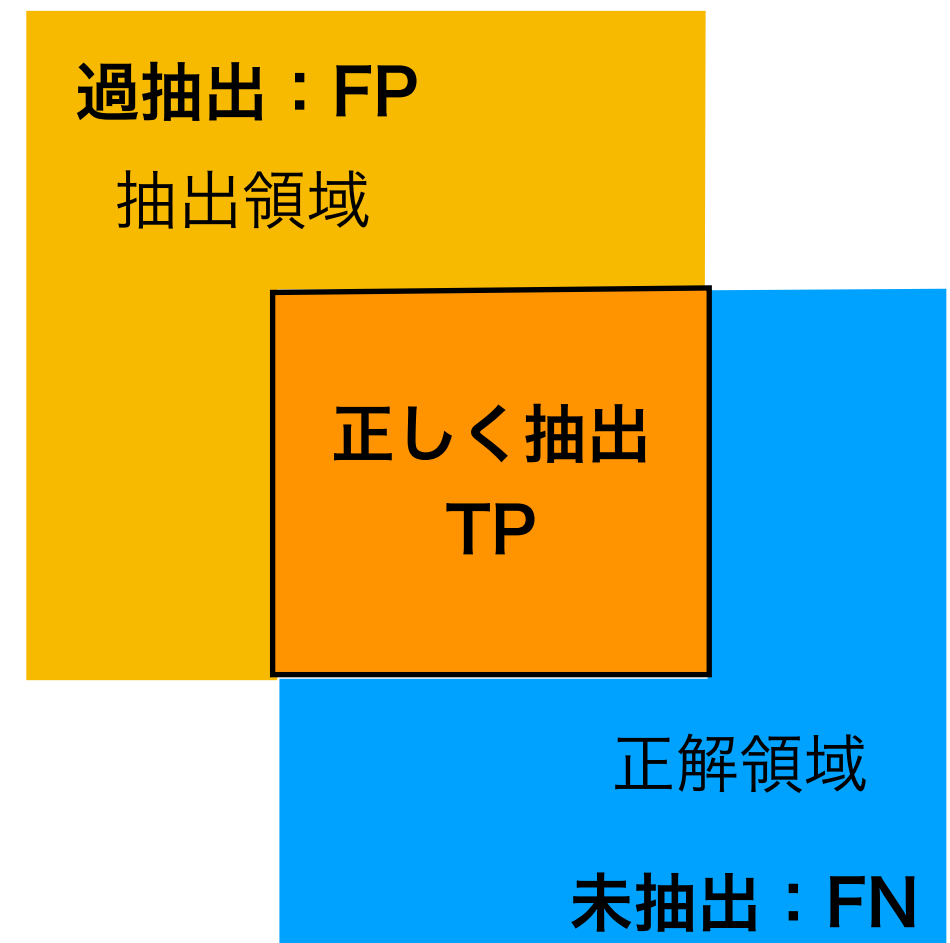
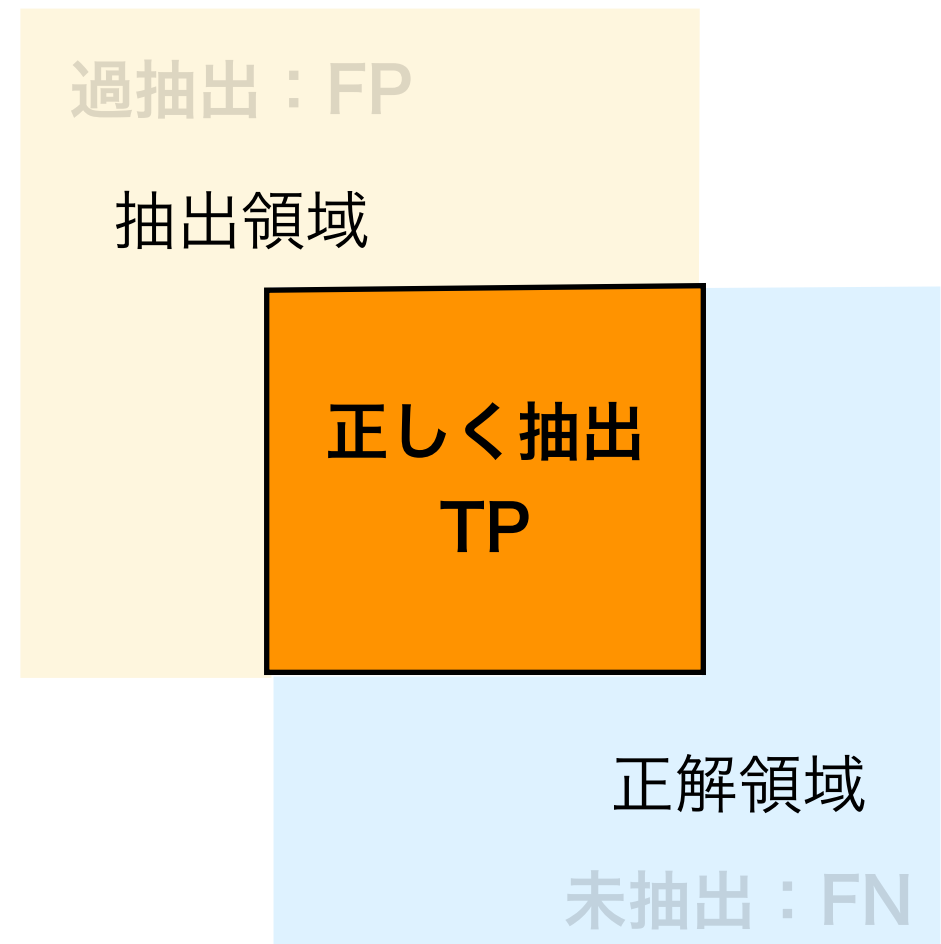
$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{F1 (Dice)} = 2 / \{1 / \text{Precision} + 1 / \text{Recall}\} = 2\text{TP} / (2\text{TP} + \text{FP} + \text{FN})$$

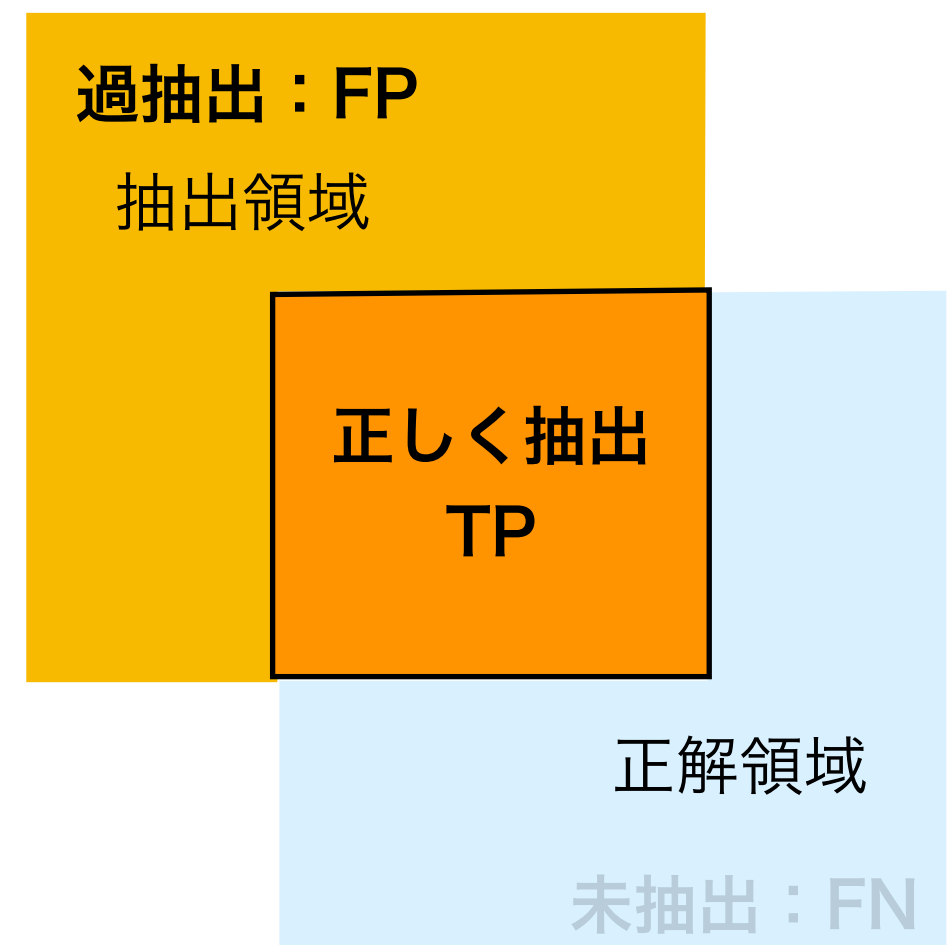
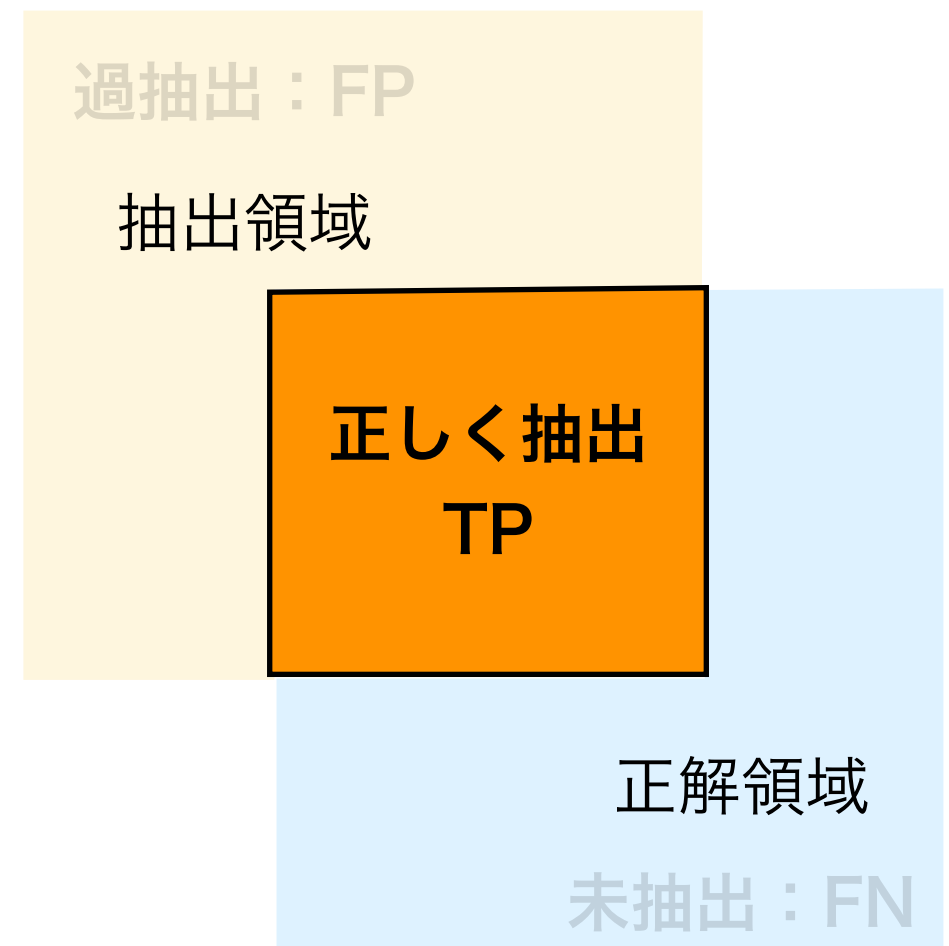
IoU

$$\text{IoU (Jaccard)} = \text{TP} / (\text{TP} + \text{FP} + \text{FN})$$



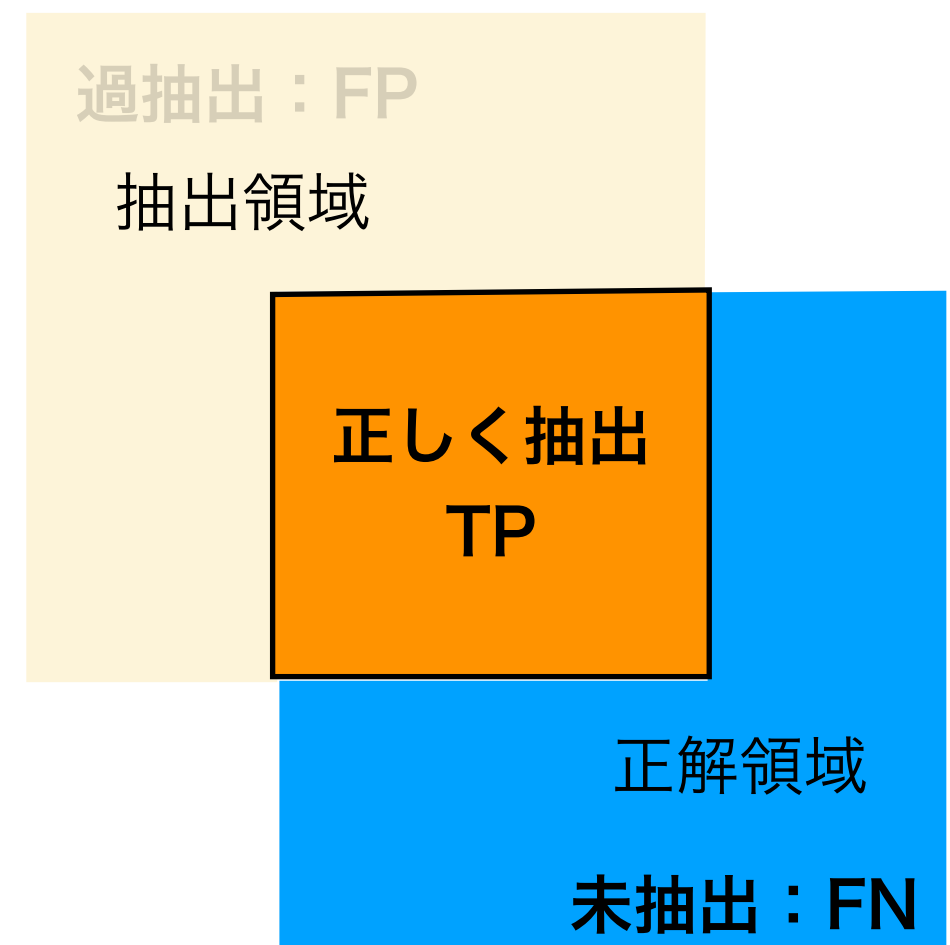
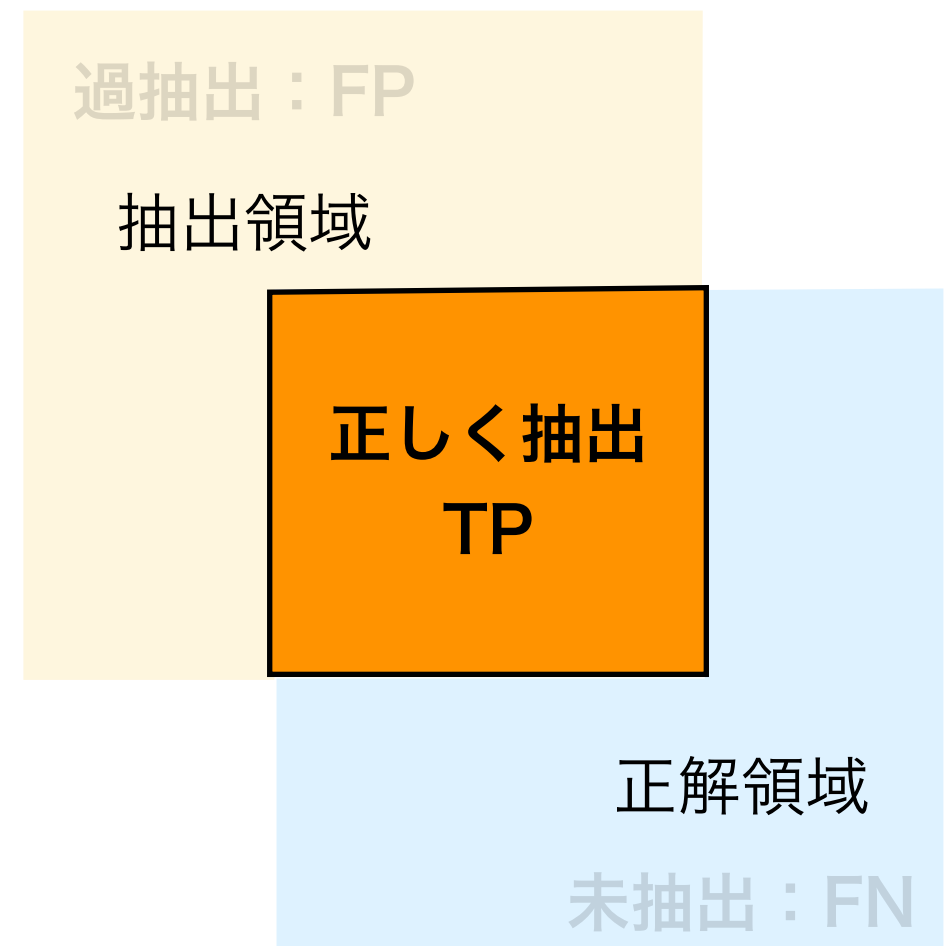
Precision

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$



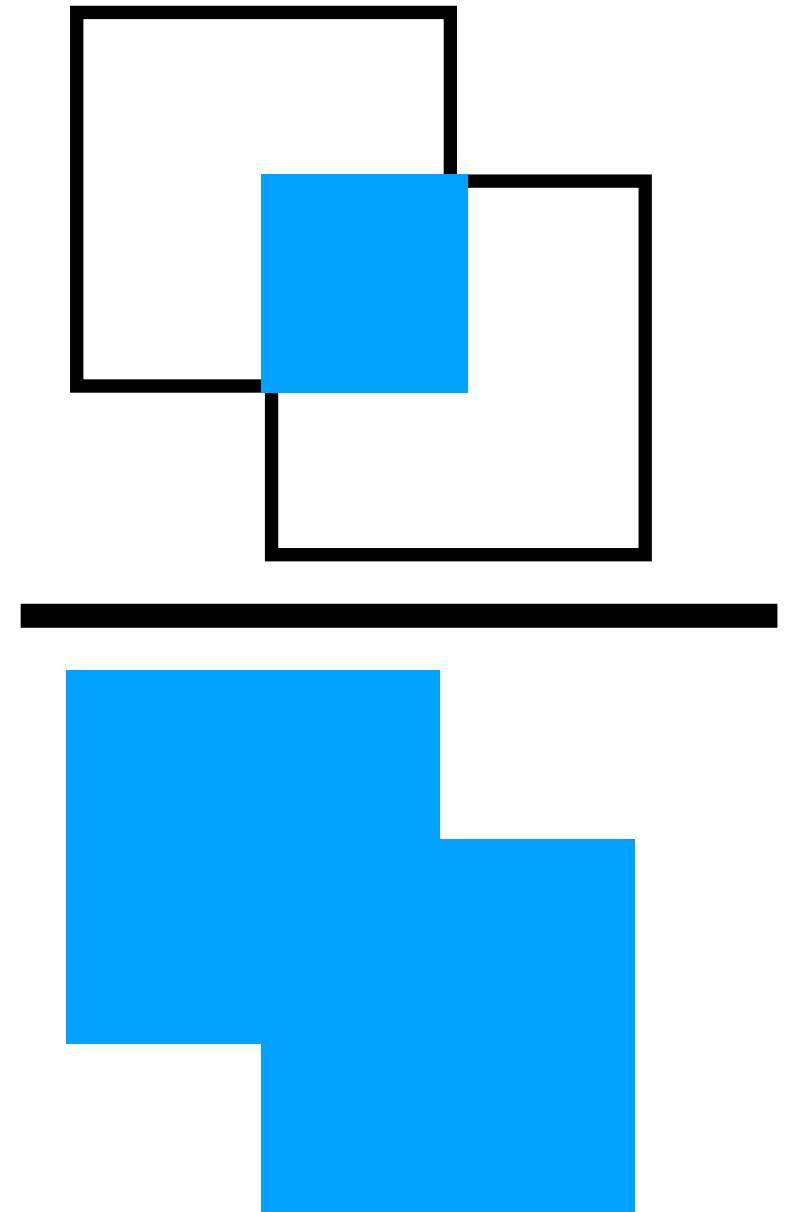
Recall

$$\text{Recall} = \text{TP} / (\text{TP} + \text{TN})$$



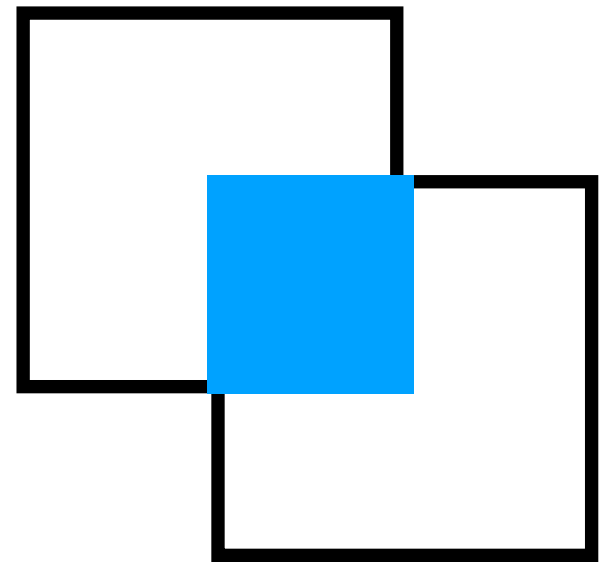
評価方法：領域の評価

Intersection-over-Union (IoU) =
(Jaccard index)



評価方法：領域の評価

$$|A \cap B|$$



Intersection-over-Union (IoU) =
(Jaccard index)



A, Bは集合.

$|A|$ は, A の要素数を表すよ.

画像の場合には画素数を用いるよ.

$$|A \cup B|$$

評価方法：領域の評価

$$\text{Intersection-over-Union (IoU) (Jaccard index)} = \frac{|A \cap B|}{|A \cup B|}$$

$$\text{Sørensen–Dice index} = \frac{2|A \cap B|}{|A| + |B|}$$

$$\text{Simpson coefficient (Overlap coefficient)} = \frac{|A \cap B|}{\min\{|A|, |B|\}}$$

$\min\{a, b\}$ は, a, b の小さい方を表すよ

評価方法：領域の評価

2つの集合において、一部が共通要素の場合

$$\text{IoU} < \text{Dice} < \text{Simpson}$$

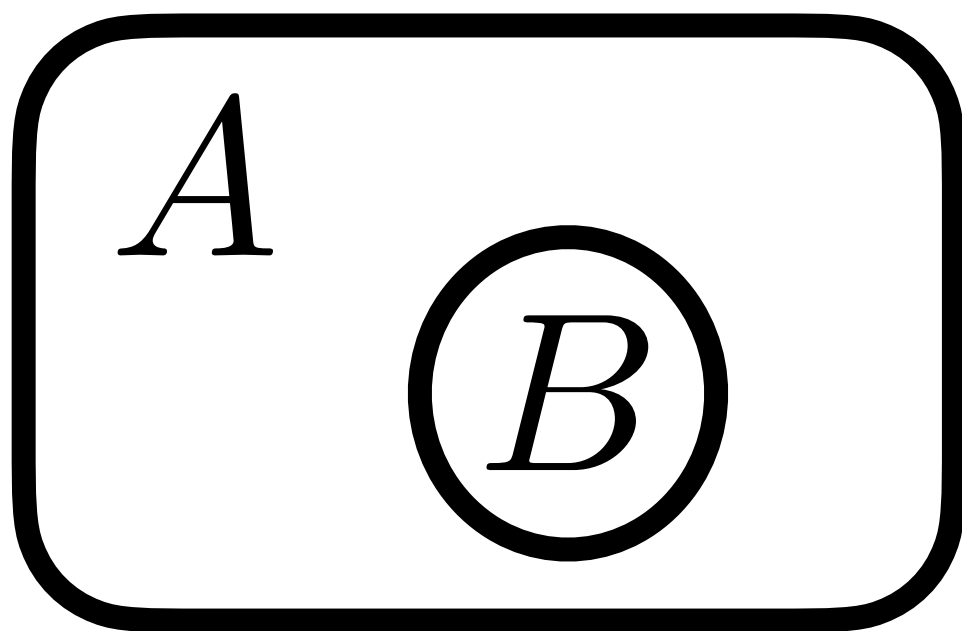
2つの集合が、空集合の場合 $|A \cup B| = 0$

$$\text{IoU} = \text{Dice} = \text{Simpson} = 0$$

2つの集合において、一つが他に完全に含まれる（真部分集合）の場合

$$\text{IoU} \neq 1, \text{Dice} \neq 1, \text{Simpson} = 1$$

評価方法：領域の評価



$$|A| = 100$$

$$|B| = 10$$

Intersection-over-Union (IoU)
(Jaccard index) $= \frac{|A \cap B|}{|A \cup B|} = \frac{10}{100} = 0.1$

2つを似ていないと評価したいとき

Sørensen-Dice index $= \frac{2|A \cap B|}{|A| + |B|} = \frac{2 \cdot 10}{100 + 10} = \frac{20}{110} = 0.182$

2つを似ていると評価したいとき

Simpson coefficient $= \frac{|A \cap B|}{\min\{|A|, |B|\}} = \frac{10}{\min\{100, 10\}} = \frac{10}{10} = 1$

課題に取り組みましょう

画像分類のポイント

- 分類問題を深層学習で解決する：**出力層の作り方**
- ファインチューニングを理解する：**モデルの改造**
- 転移学習を理解する：**モデルの選択と重みの固定**
- 評価方法：**2クラス分類／多クラス分類**

領域分割のポイント

- 領域分割のモデルを作る：U-Net, U-Net++
- 結果を可視化する：ラベル番号をカラーで表示
- 必要な領域を抽出する：出力の整数化
- 評価方法：IoUなど