# 人工知能
# （畳込みニューラルネットワーク）
# の医療への応用

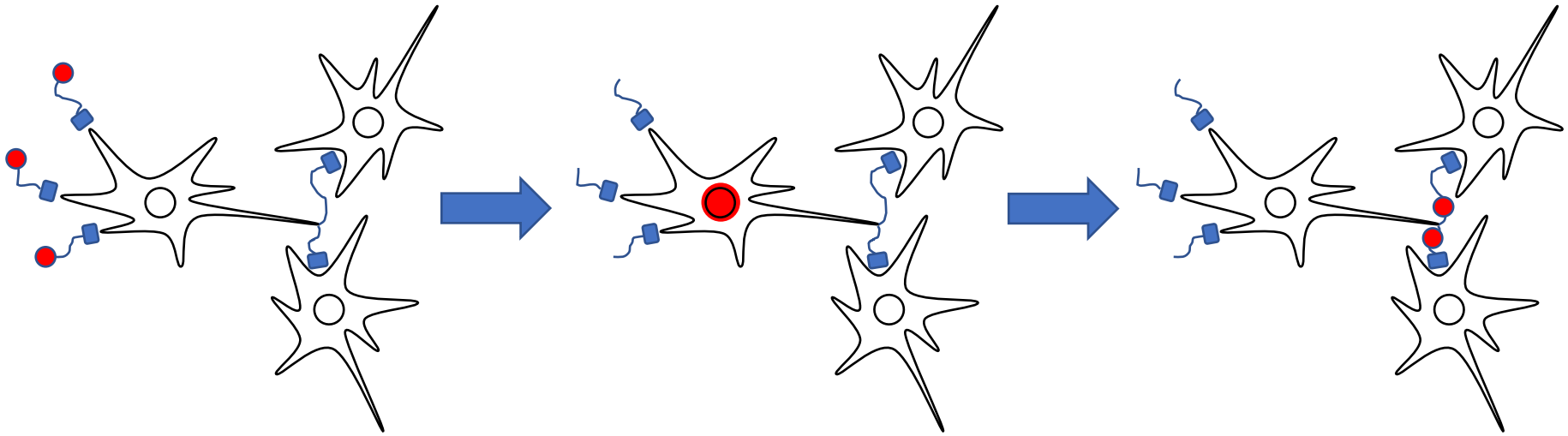
立命館大学 理工学部 電子情報工学科
中山 良平

# 人間の脳

他の神経細胞から
興奮状態（信号）
が伝達

細胞体の電位レベル
（≒ 信号の和）
が上昇，閾値と比較

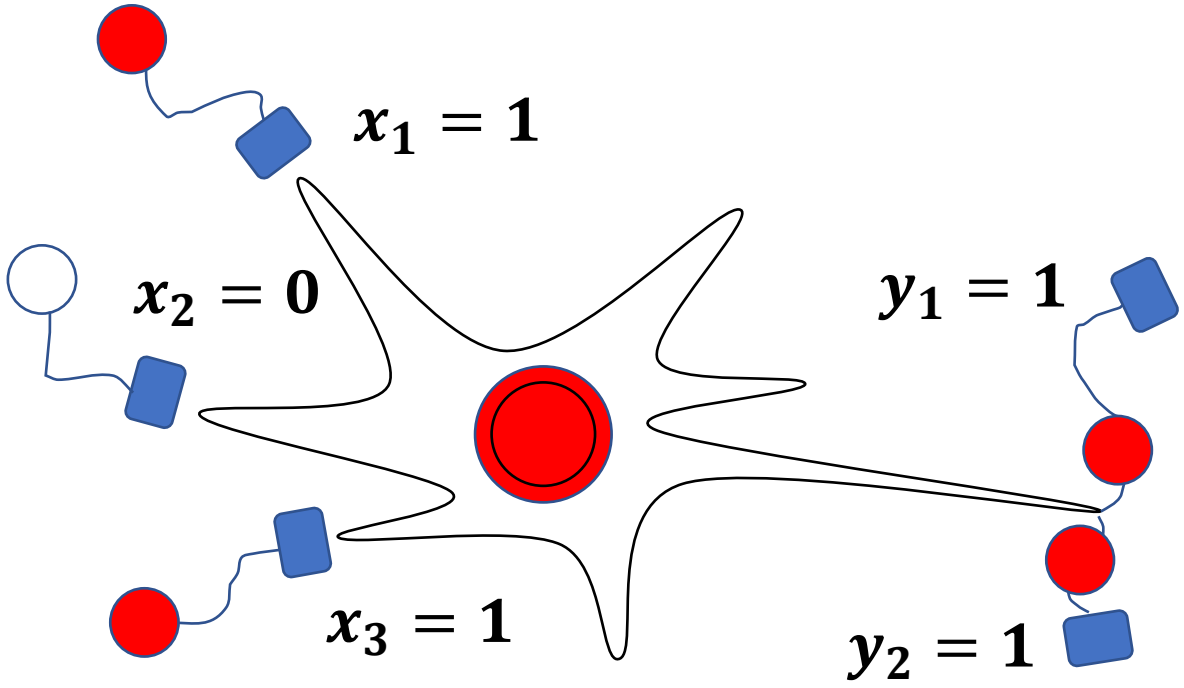
閾値より大のとき，
発火し，隣の
ニューロンへ伝達



# 入出力信号を数学的に表現

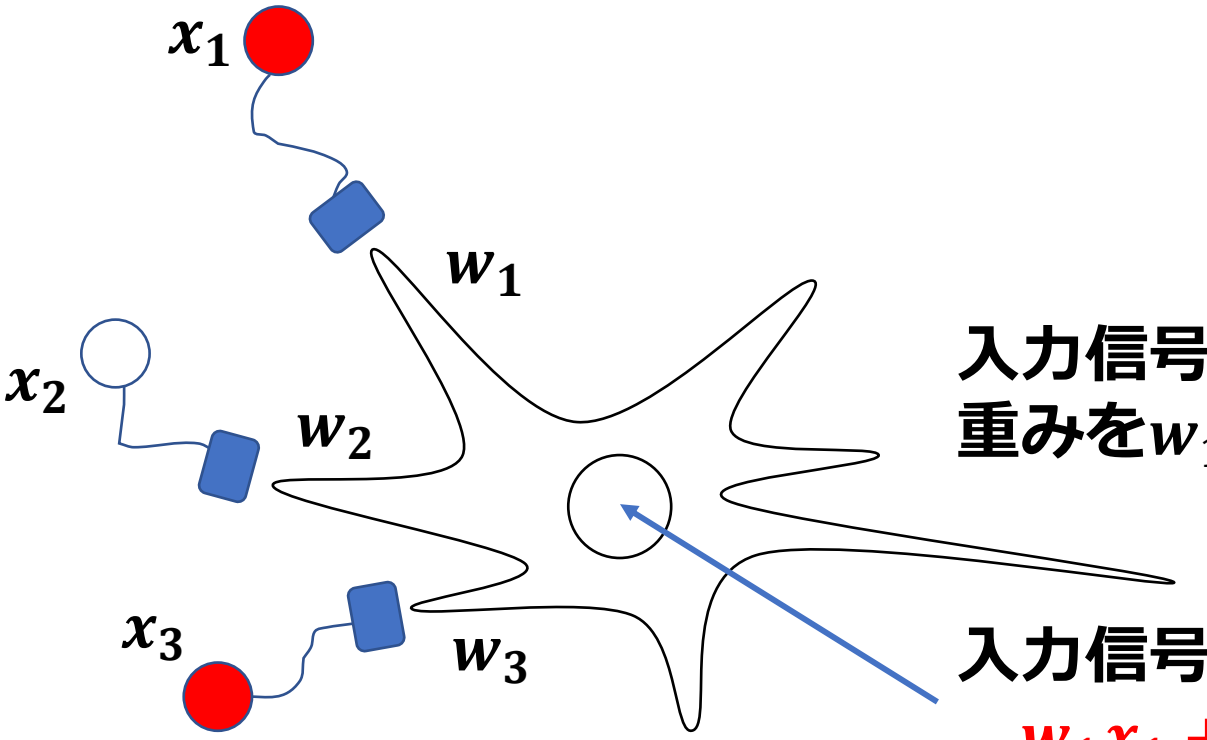
- 入力（出力）信号は「あり」「なし」の2情報で表現可

$$\begin{cases} \text{入力信号なし：} x = 0 \\ \text{入力信号あり：} x = 1 \end{cases} \qquad \begin{cases} \text{出力信号なし：} y = 0 \\ \text{出力信号あり：} y = 1 \end{cases}$$



# 入力信号に対する重みを数学的に表現

- ニューロンの発火の有無は入力信号の和で判定
- 和のとり方は単純でないはず！脳は重みを変えて処理




入力信号$x_1, x_1, x_1$に対する
重みを$w_1, w_2, w_3$で表現

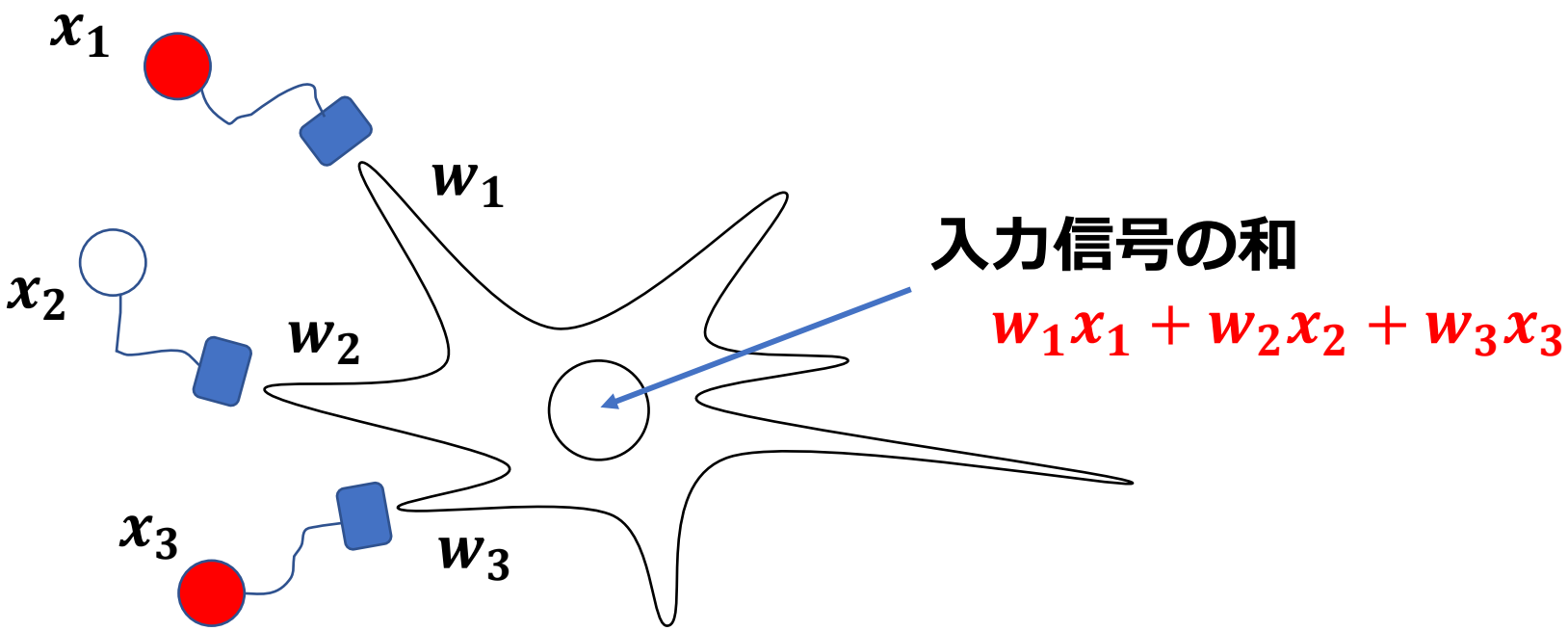
入力信号の和

$$w_1x_1 + w_2x_2 + w_3x_3$$

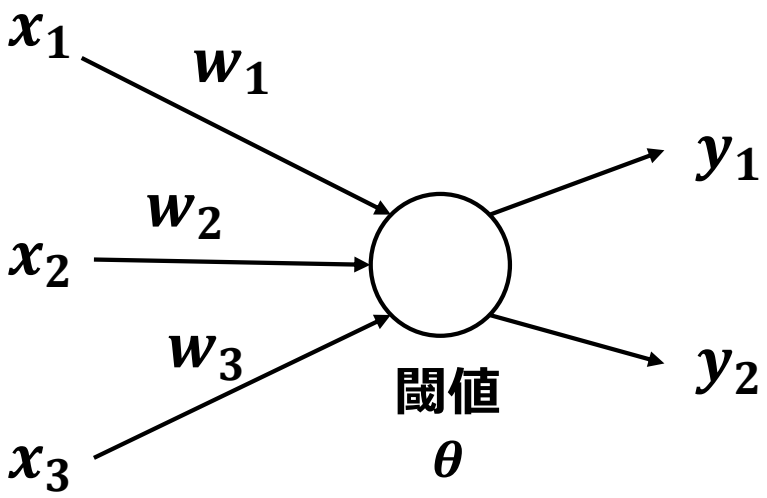
# 発火条件を数学的に表現

- ニューロンは入力信号の和が閾値を超えると発火
- 発火条件（$\theta$：ニューロン固有の閾値）

$$\begin{cases} \text{出力信号なし}(y=0): w_1x_1 + w_2x_2 + w_3x_3 < \theta \\ \text{出力信号あり}(y=1): w_1x_1 + w_2x_2 + w_3x_3 \geq \theta \end{cases}$$



# unitの出力値




unit（ニューロン）

生物学的ニューロン再現のため
単位ステップ関数$u$により

$$y = u(w_1x_1 + w_2x_2 + w_3x_3 - \theta)$$

⇒ 出力値が0または1

⇒ 入力値も0または1

作成者が定義する関数$h$で一般化

$$y = h(w_1x_1 + w_2x_2 + w_3x_3 - \theta)$$

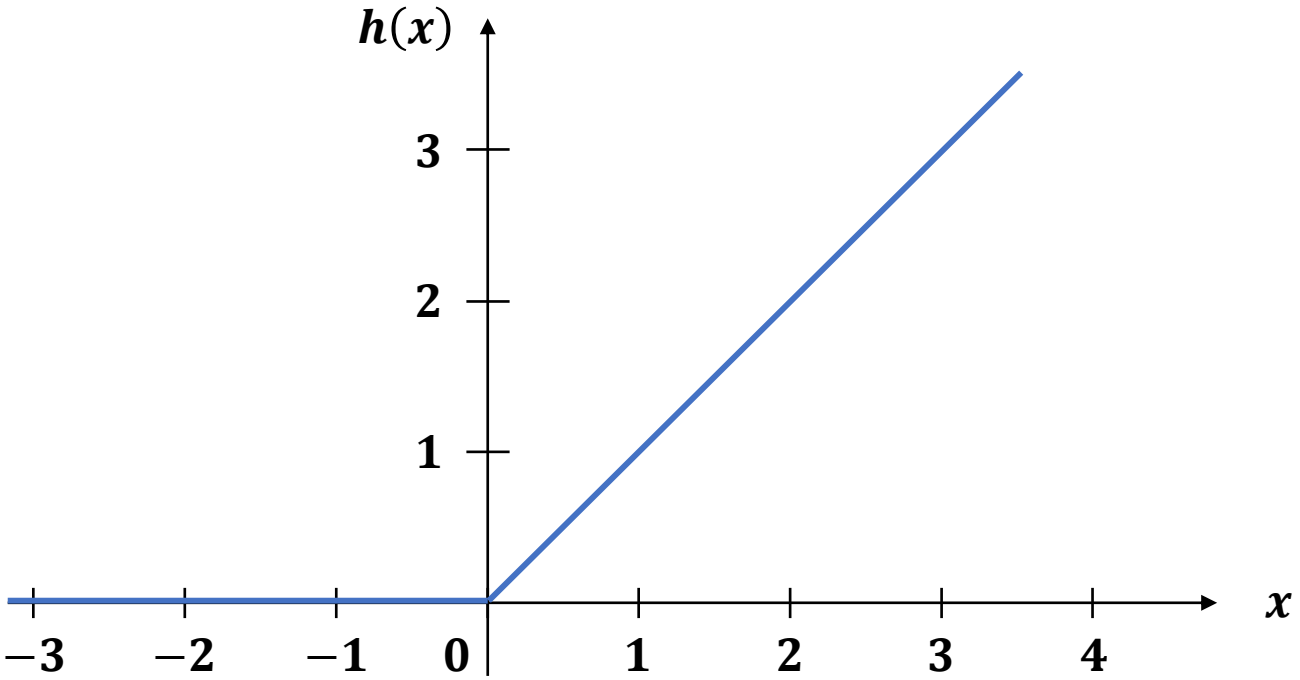
⇒ 出力値が0,1と限らない

⇒ 入力値も0,1とは限らない

関数$h$を活性化関数と呼ぶ

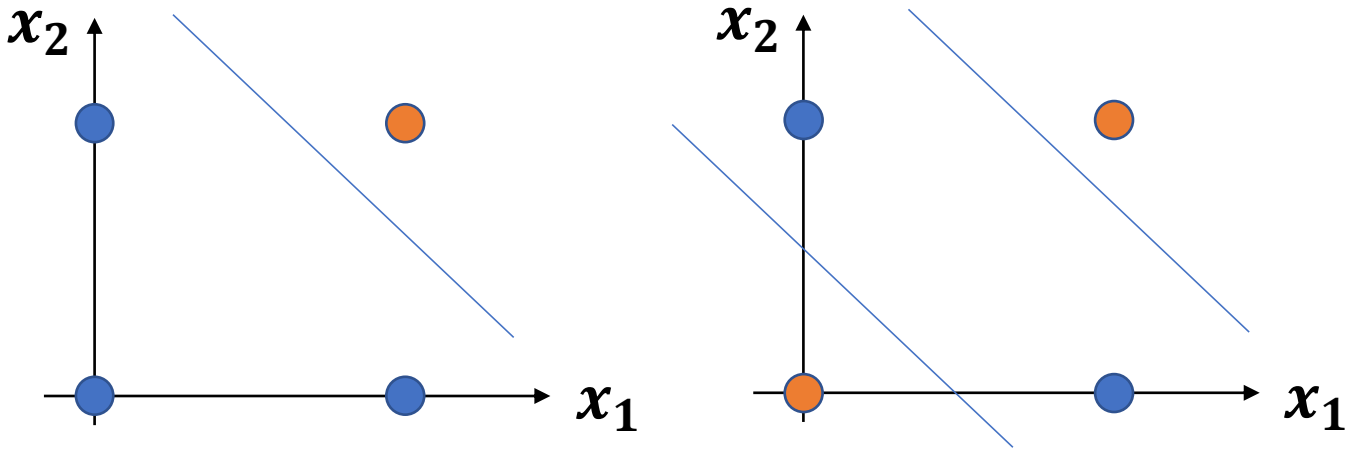
# 活性化関数

ReLU（Rectified Linear Unit）：$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$



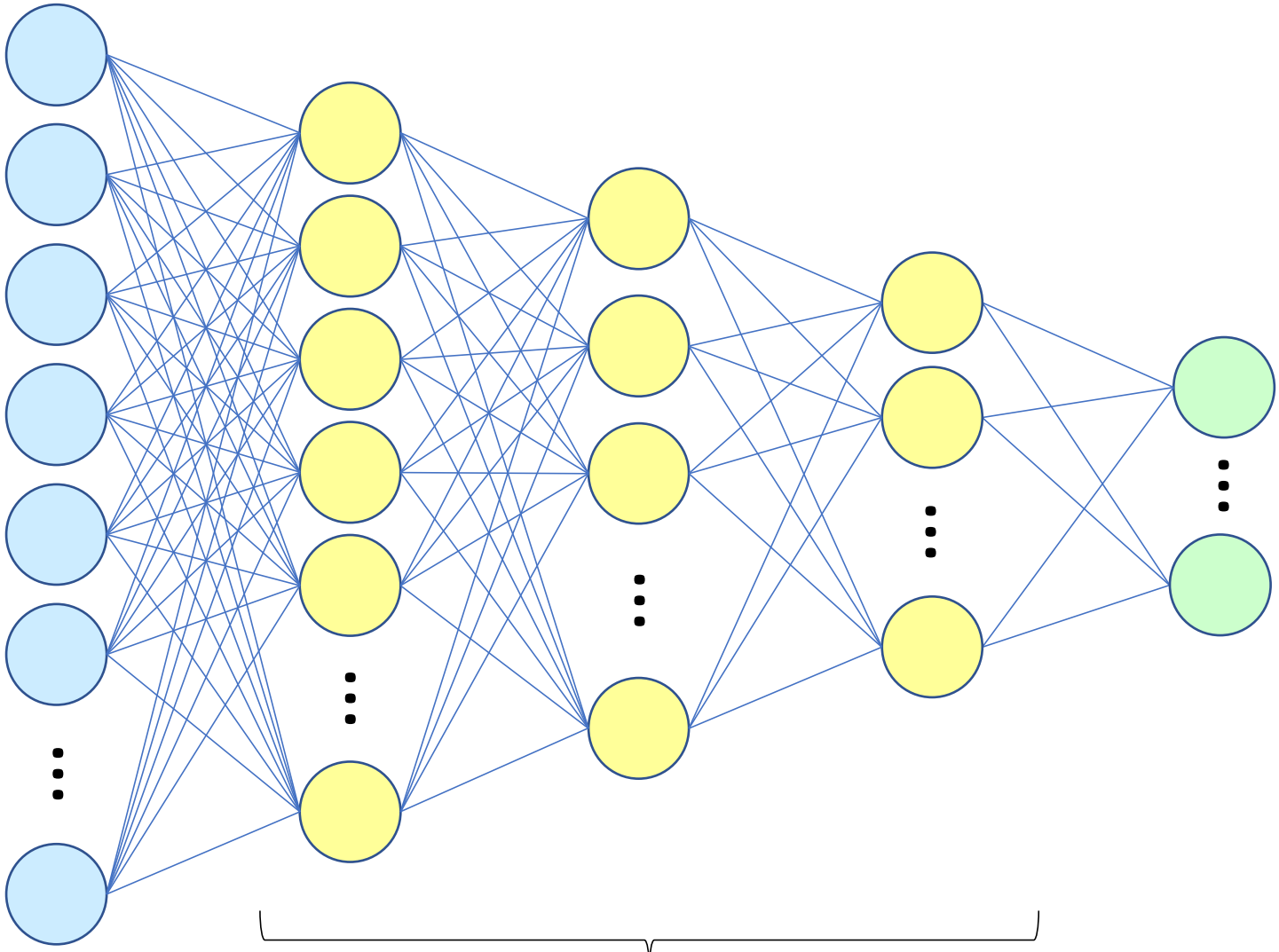
# unitの問題

- 線形分離可能な問題にしか対応できない




⇒ 複数のunitを階層的に構成することで対応
⇒ ニューラルネットワーク
（ANN: Artificial Neural Network）

# ニューラルネットワーク

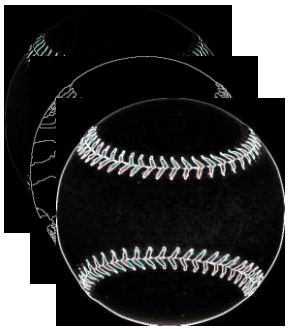


# 入出力層の設計

入力画像

フィルタ処理による特徴強調

定量化（特徴量）

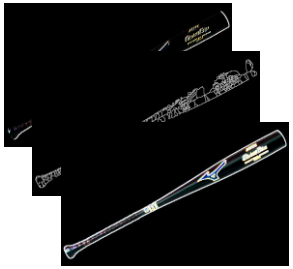


| | |
|---|---|
| 長軸 | 8.5 |
| 短軸 | 8.5 |
| 円形度 | 1.0 |
| 角数 | 0 |

教師信号

ボール 1

バット 0

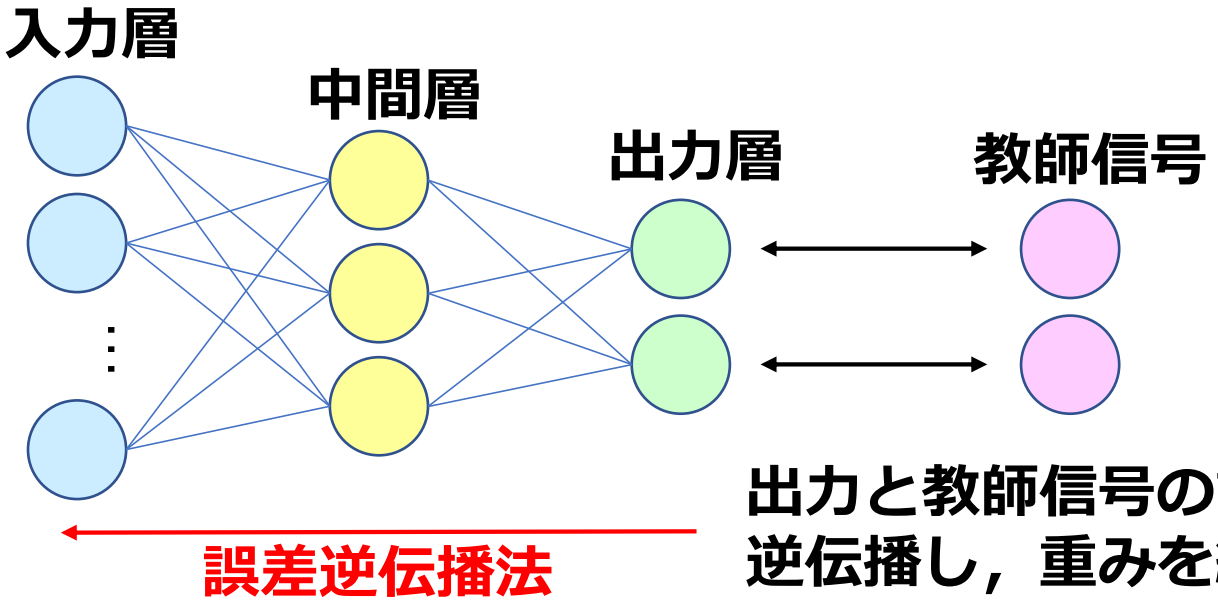


| | |
|---|---|
| 長軸 | 50 |
| 短軸 | 5 |
| 円形度 | 0.2 |
| 角数 | 6 |

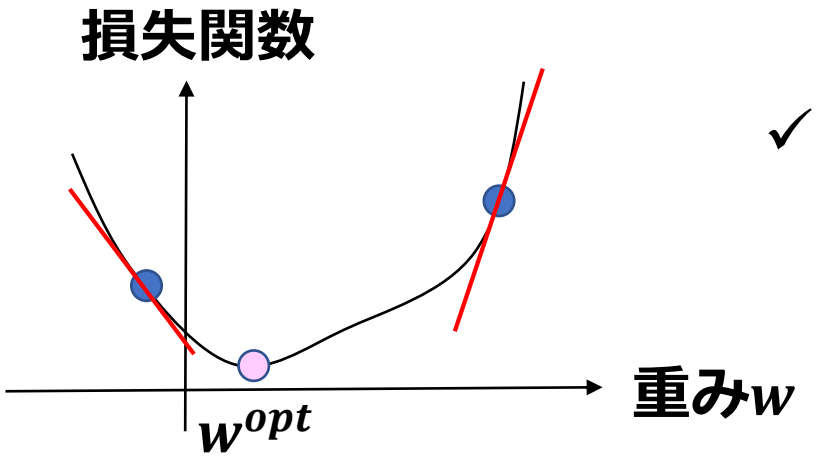
ボール 0

バット 1

# ニューラルネットワークの学習




出力と教師信号の誤差を各層に逆伝播し，重みを繰り返し更新



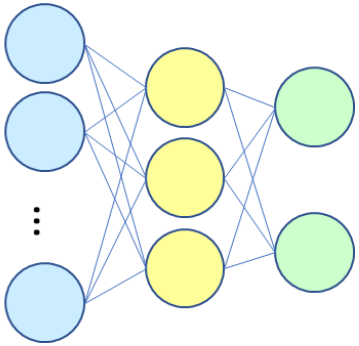

- ✓ 損失関数を微分し，重みの更新方向を決定

  ⇒ 微分値0で，weightの更新終了

# 従来のニューラルネットワーク

| | |
|---|---|
| 長軸 | 8.5 |
| 短軸 | 8.5 |
| 円形度 | 1.0 |
| 角数 | 0 |

中間層数：1～3



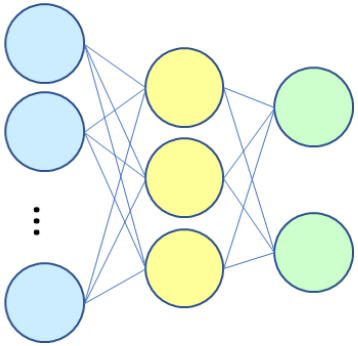
入力 → 特徴抽出 → 定量化 → 分類 → 出力

**人間が教え込む型ニューラルネットワーク**

データに適した特徴の違いを見つけ，
特徴を正確に定量化する手法を定義することが重要

# 畳み込みニューラルネットワーク

特徴抽出部分もデータから学習



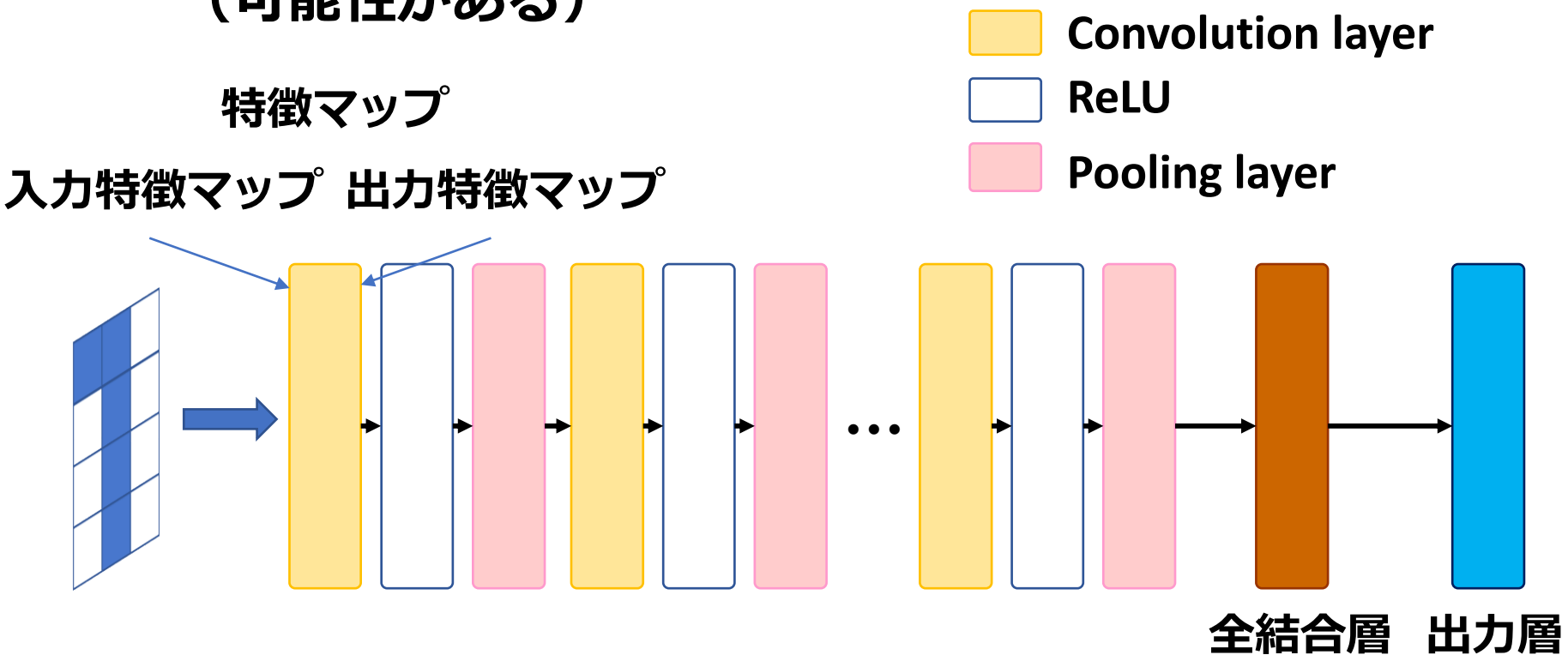
入力 → 特徴抽出 → 定量化 → 分類 → 出力

**勝手に学ぶ型ニューラルネットワーク**

Big data と computer の性能向上で実現

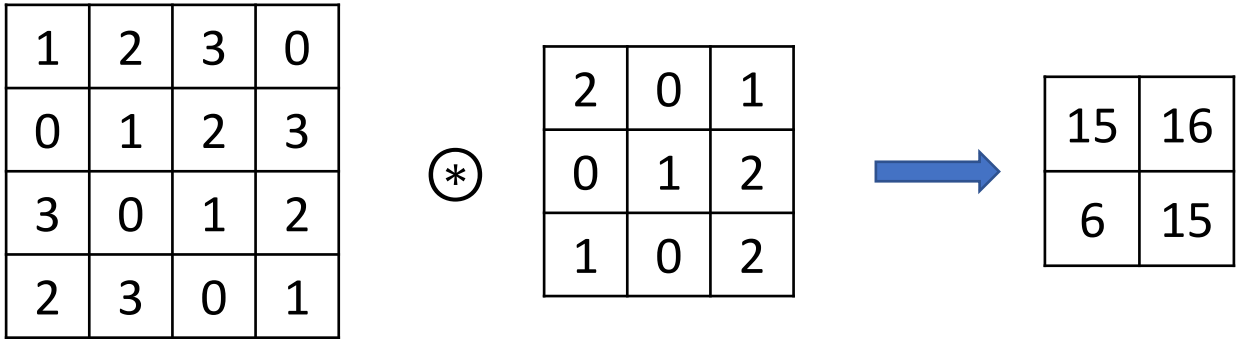
# 畳み込みニューラルネットワーク

- CNNはANNに畳み込み（convolution）層とプーリング（pooling）層が加わったもの
- CNNは2次元データを2次元データとして出力
  ⇒ 画像などの形状を有したデータを正しく理解できる（可能性がある）



# 畳み込み層

- 畳み込み演算（フィルタ演算）

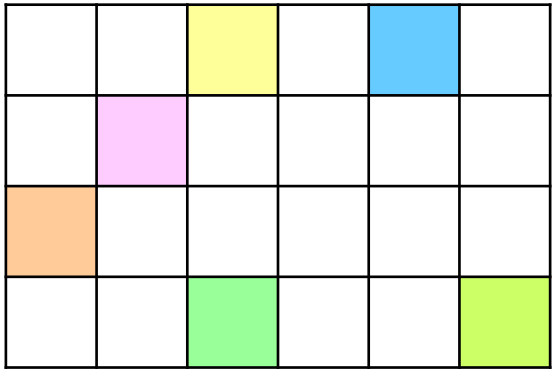



- 入力データに対し，フィルタのウィンドウを一定の間隔でスライドさせながら適用
- フィルタの要素と入力の対応する要素を乗算し，その和を求める　⇒　積和演算
- CNNではフィルタの要素がweightに対応

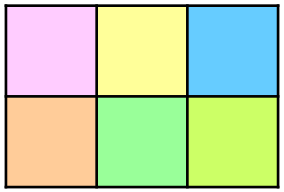
# プーリング層

特徴マップを縮小（次元の低減）
⇒ 幾何学的変化に対する不変性を獲得
⇒ 特徴の位置感度を低下させる役割

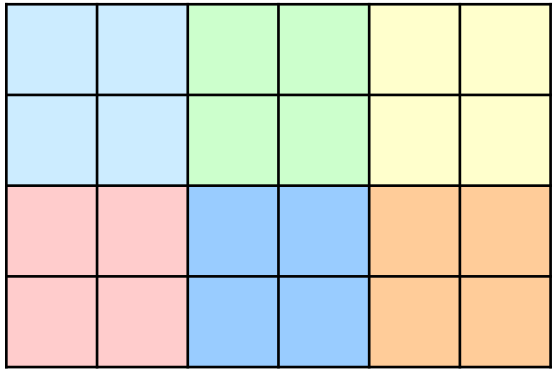
## 最大値プーリング



2×2の領域
での最大値



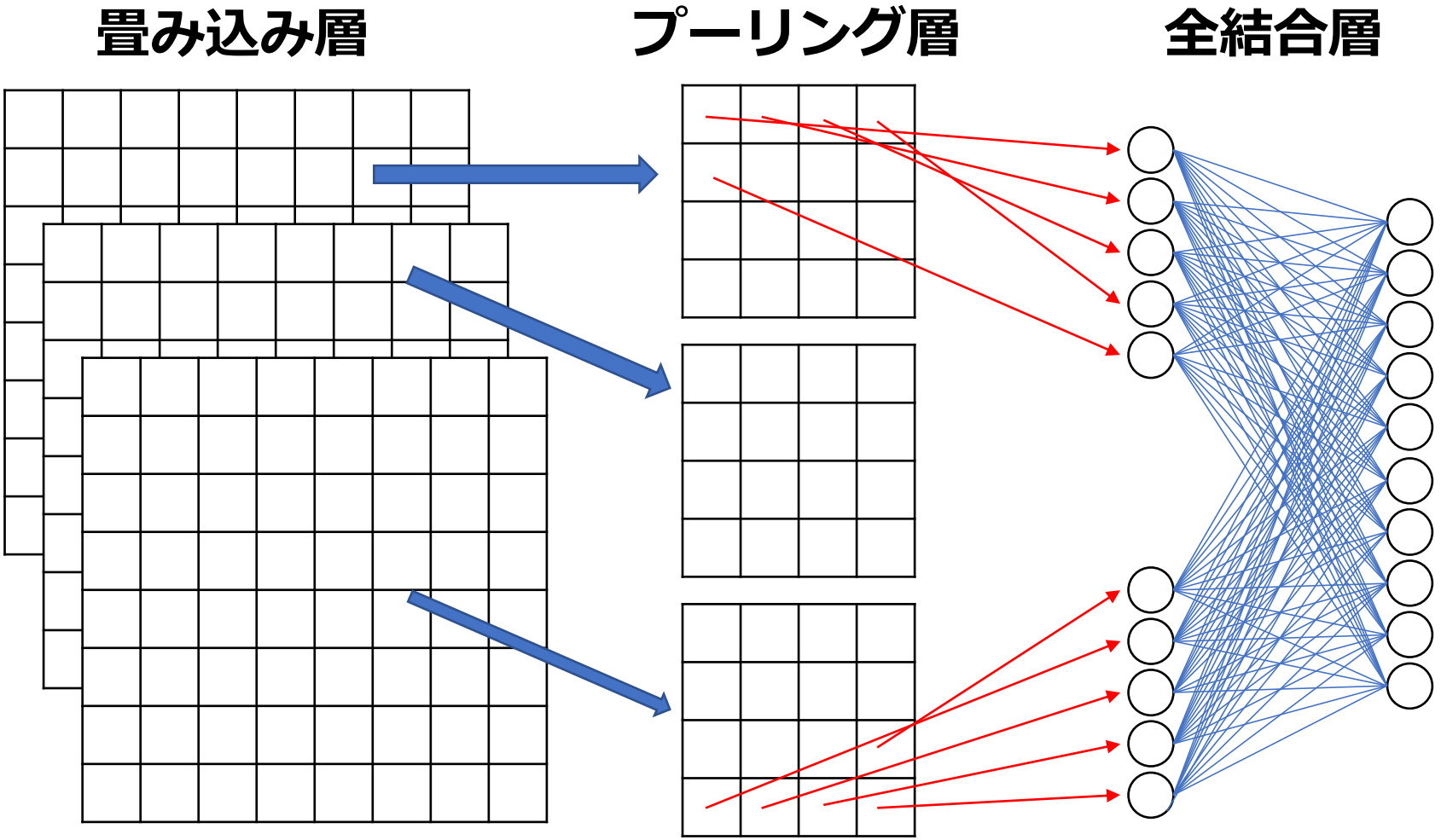
## 平均値プーリング



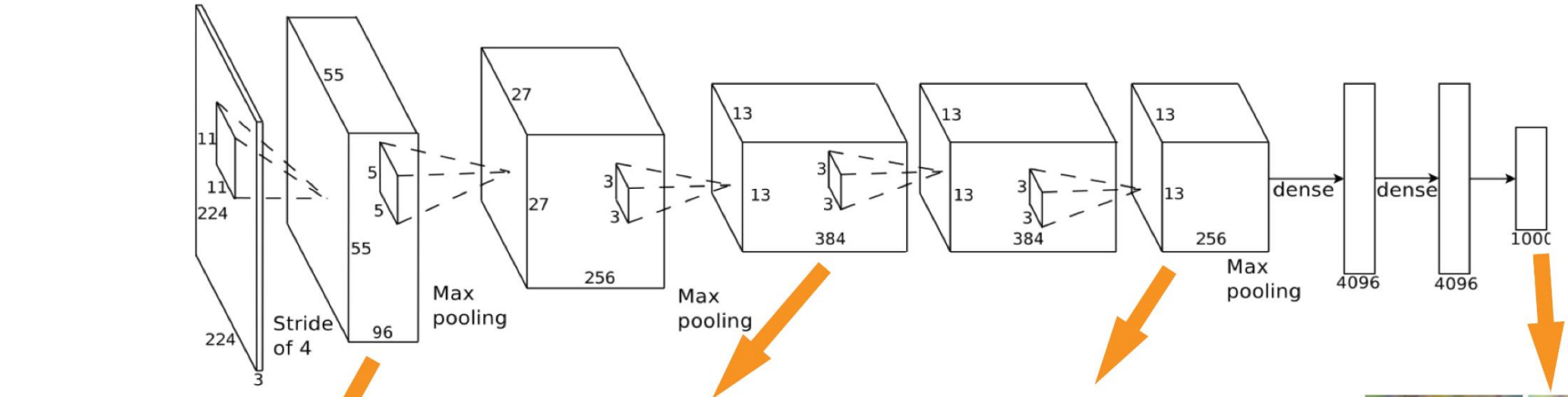
2×2の領域
での平均値

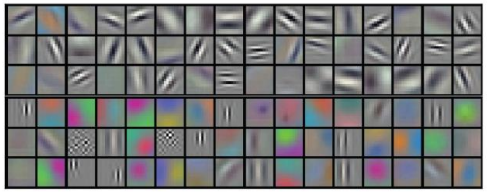
# 全結合層

畳み込み層またはプーリング層の出力である
2次元特徴マップを1次元に展開

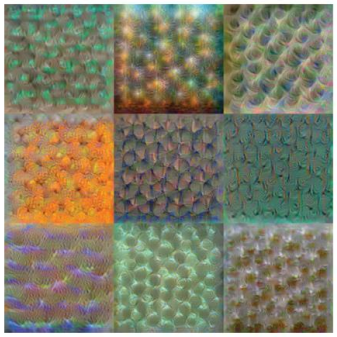


# 階層構造による情報抽出



Conv 1: Edge+Blob



Conv 3: Texture

Numerical

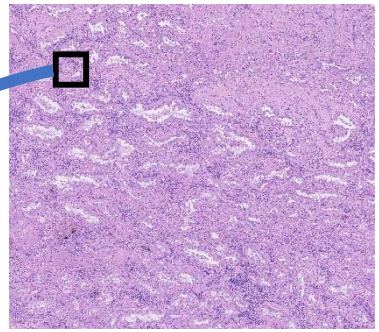
Data-driven

Conv 5: Object Parts


Fc8: Object Classes

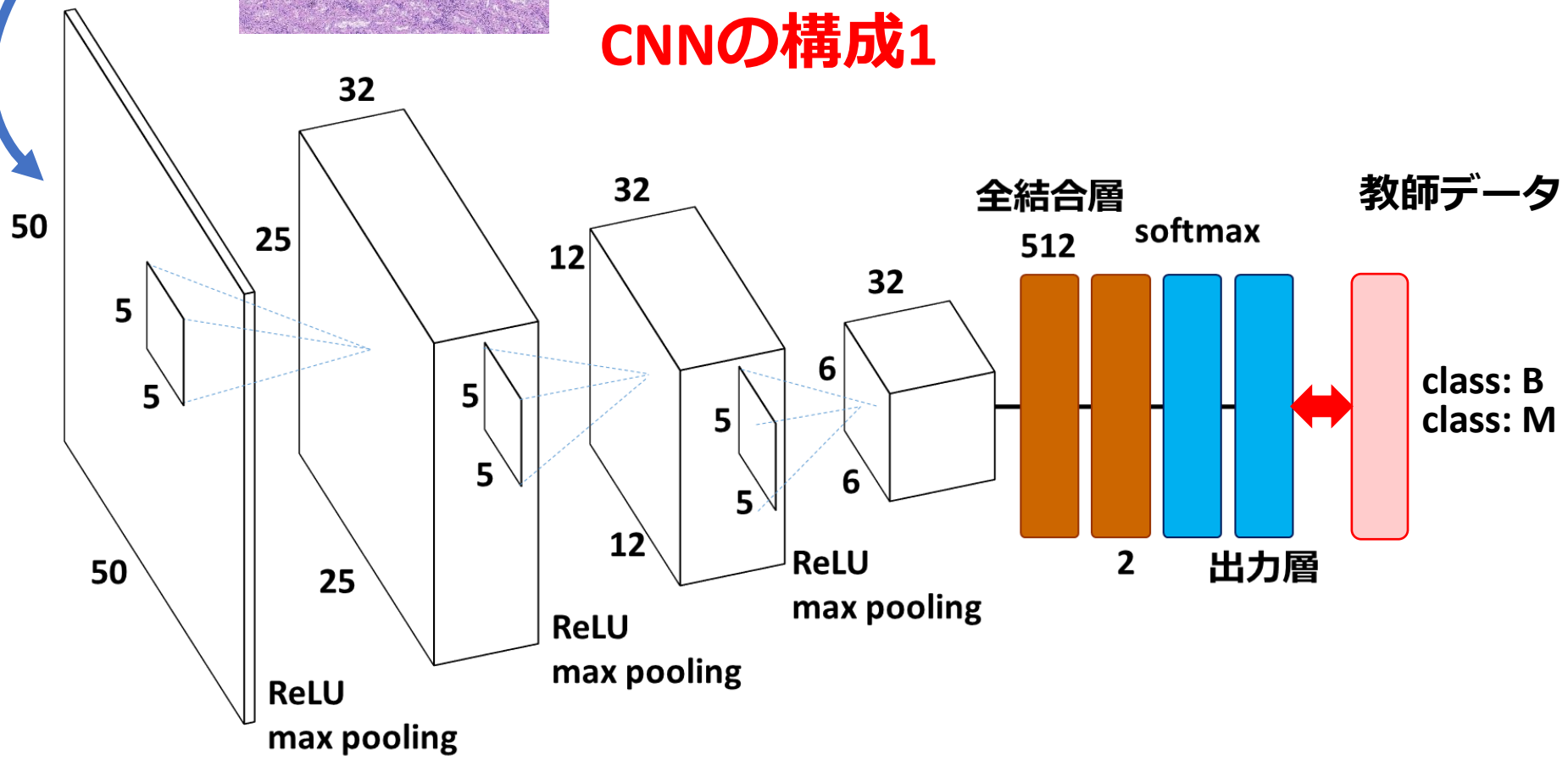
http://vision03.csail.mit.edu/cnn_art/

# 病理組織像の良悪性自動分類



良性ROI：32,484
悪性ROI：33,789

CNNの構成1



# CNNの構成1

C:¥Users¥Nakayama¥Documents¥MATLAB¥CNN_pathology1014.m*

エディター　パブリッシュ　表示

```
76      % Create A Convolutional Neural Network(CNN)
77      % 入力層の定義
78 -    inputLayer  = imageInputLayer([ROI_size, ROI_size, nChannel]);
79
80      % 中間層の定義
81 -    middleLayers = [
82          convolution2dLayer([5 5], 32)
83          reluLayer()
84          maxPooling2dLayer(2, 'Stride', 2)
85          convolution2dLayer([5 5], 32)
86          reluLayer()
87          maxPooling2dLayer(2, 'Stride',2)
88          convolution2dLayer([5 5], 32)
89          reluLayer()
90          maxPooling2dLayer(2, 'Stride',2) ];
91
```

スクリプト　行 150　列 9

# CNNの構成1

C:¥Users¥Nakayama¥Documents¥MATLAB¥CNN_pathology1014.m

エディター　パブリッシュ　表示

```
92      % 出力層の定義
93 -    finalLayers = [
94          fullyConnectedLayer(512)
95          reluLayer()
96          fullyConnectedLayer(numImageCategories)
97          softmaxLayer
98          classificationLayer ];
99
100     % CNNの統合
101 -   layers = [
102          inputLayer
103         middleLayers
104         finalLayers
105     ];
```

スクリプト　行 150　列 9

# 出力層の設計

- ニューラルネットワークは**分類問題**と**回帰問題**で使用
  - 分類問題：データがどのクラスに属するか
  - 回帰問題：ある入力データから数値の予測
- 問題に応じ，出力層の活性化関数を変更する必要
  - ✓ 分類問題 ⇒ ソフトマックス関数
  - ✓ 回帰問題 ⇒ 恒等関数

- **ソフトマックス関数**

$$y_k = \frac{e^{a_k}}{\sum_{i=1}^{n} e^{a_i}}$$

  - ソフトマックス関数の各出力は0～1の実数
  - 出力の総和は1　⇒　確率と解釈可能

# CNNの構成1

C:¥Users¥Nakayama¥Documents¥MATLAB¥CNN_pathology1014.m*

エディター　パブリッシュ　表示

```
107     % Train CNN Using Training Image Data
108     % Set the network training options
109 -   opts = trainingOptions('sgdm', ...
110         'Shuffle','every-epoch', ...
111         'MaxEpochs',50, ...
112         'Plots','training-progress');
113
114     % Train Network
115 -   learnedNet = trainNetwork(train_stack, train_label, layers, opts);
116
117     % Test Network
118 -   [class, likelihood] = classify(learnedNet, test_ROI);
119
```

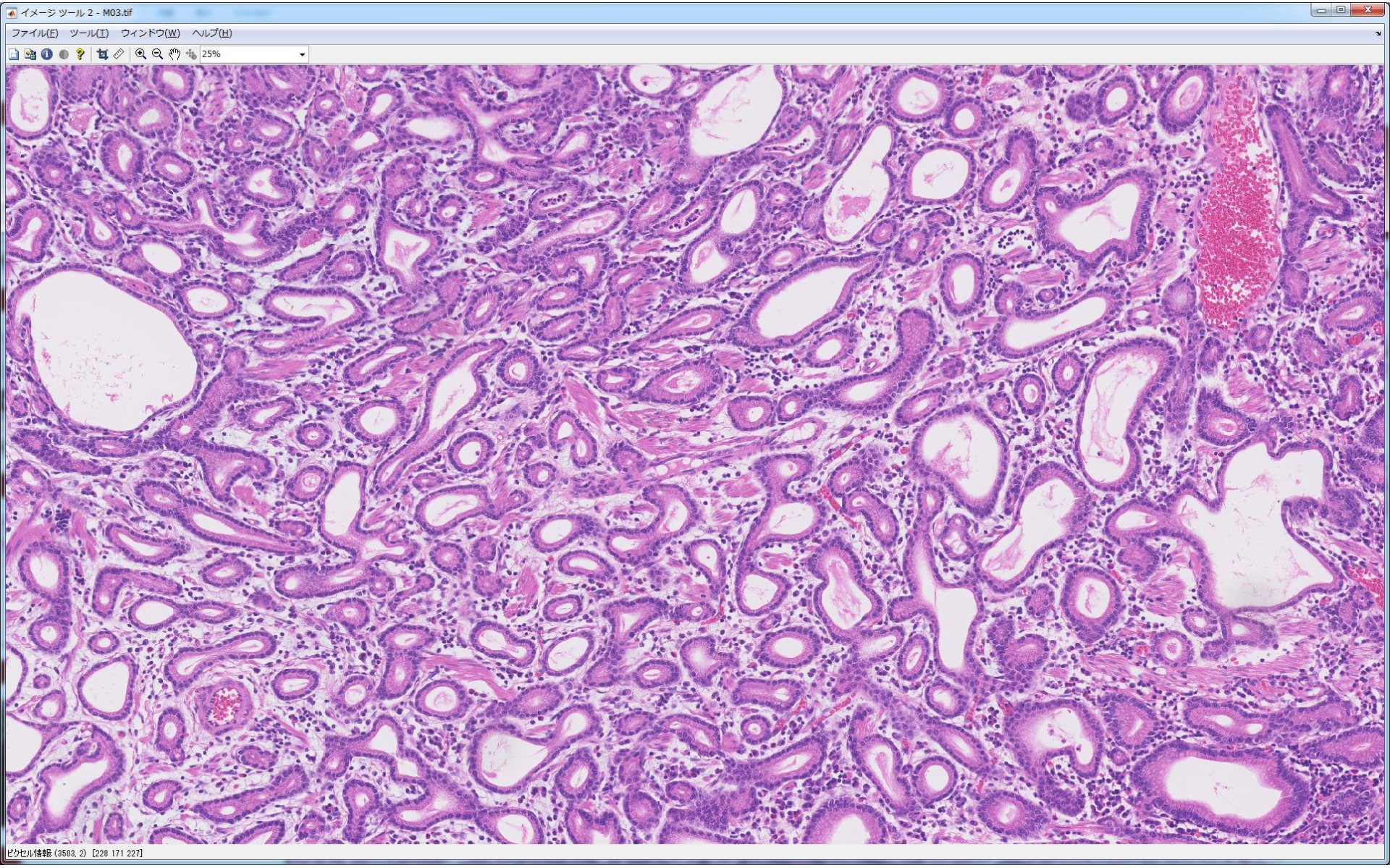
スクリプト　行 129　列 16

# CNNの学習

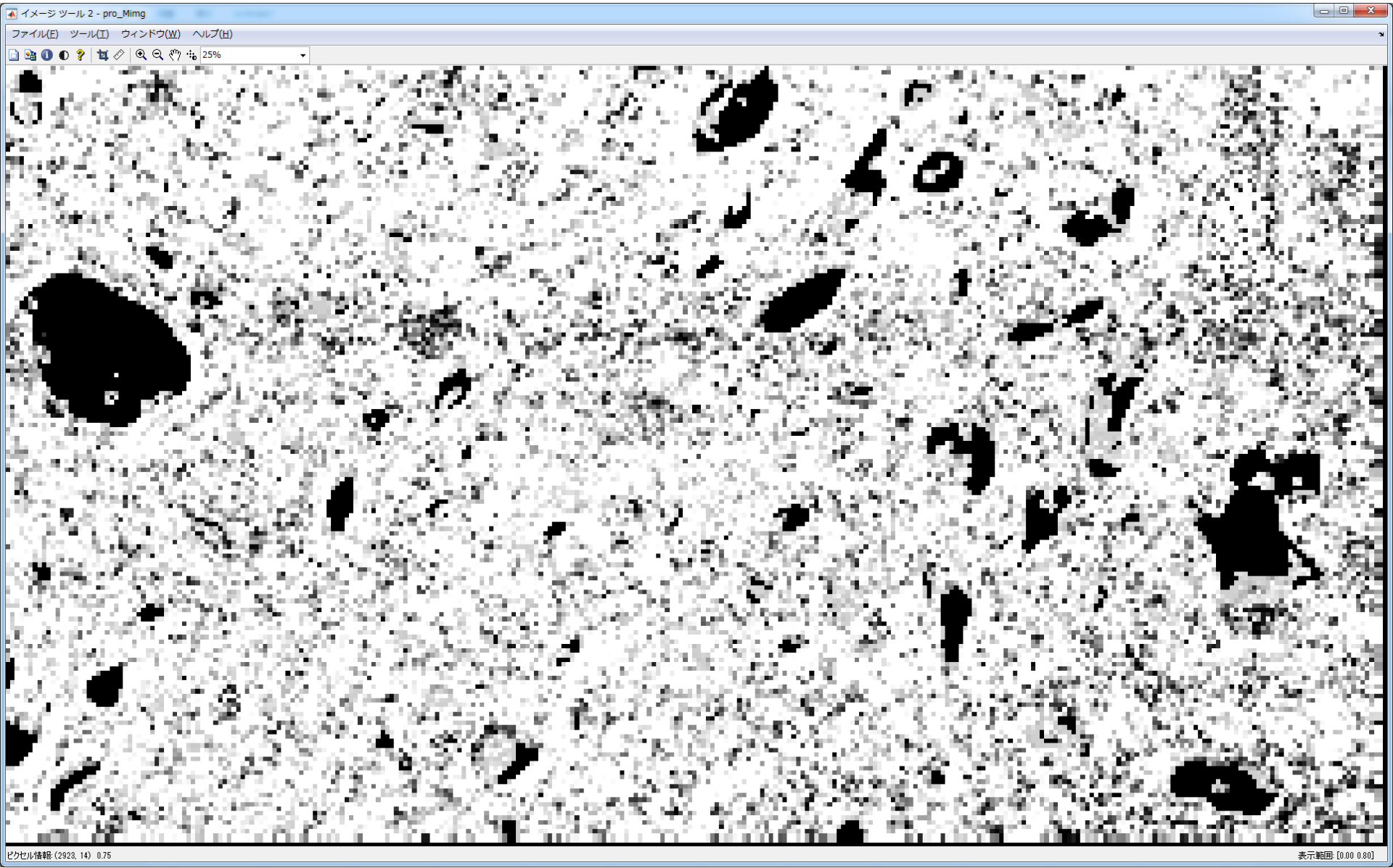
1. 学習データの中からランダムに1部データを選択
   - 選択された1部データをミニバッチと呼ぶ
   - ミニバッチの損失関数の値を減らすことを目的
2. 各weightパラメータにおける損失関数の勾配を計算
3. weightパラメータを勾配方向に微小量だけ更新
4. 1., 2., 3. の繰り返し
   - 1., 2., 3. を1回実施＝学習回数1回
   - epoch: 全ての訓練データが使用された回数

   Ex. 1,000の学習データに対し，100個データを含むミニバッチで10回学習すると1 epoch

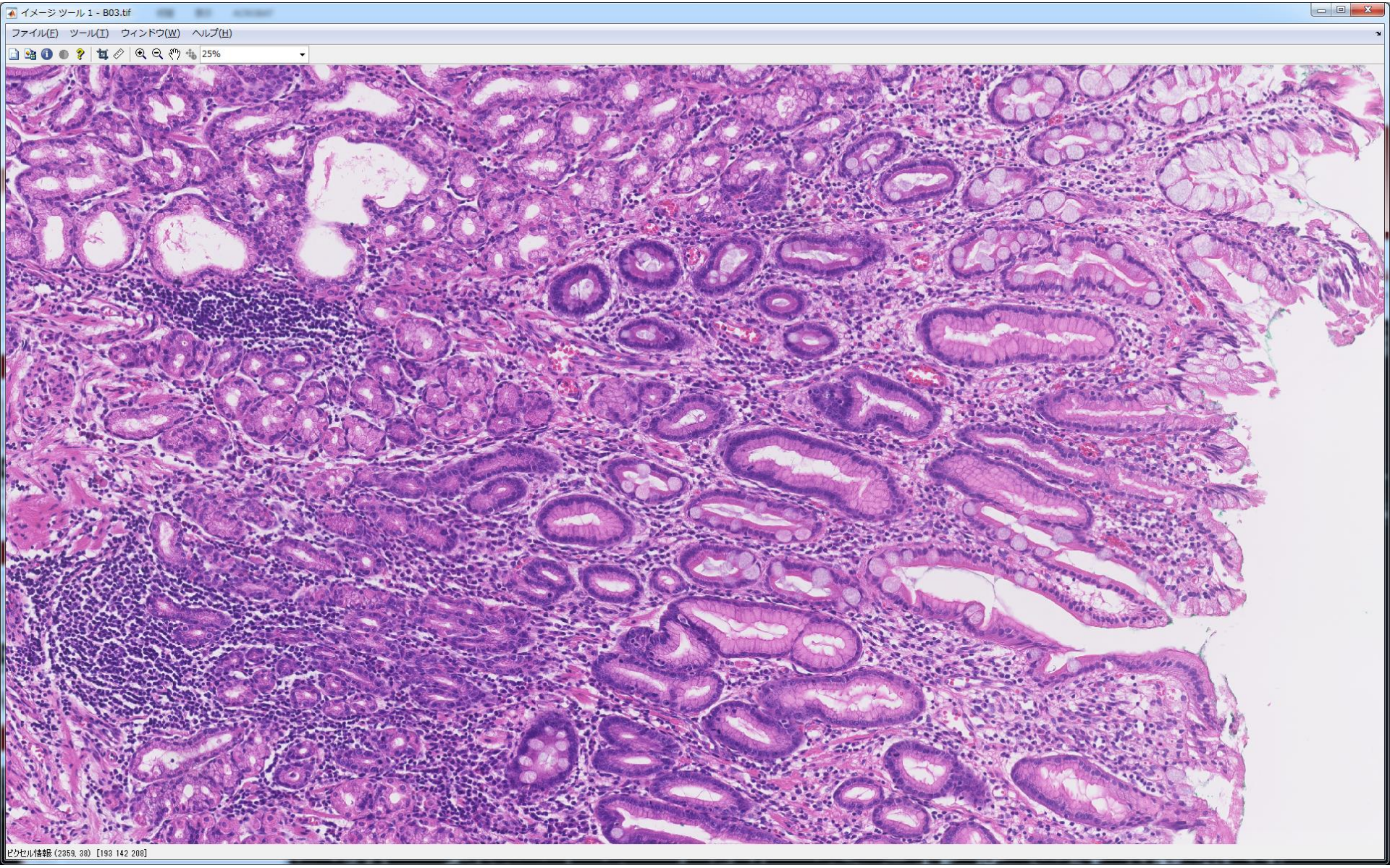
# CNNの構成1


イメージ ツール 2 - M03.tif
ファイル(F) ツール(T) ウィンドウ(W) ヘルプ(H)
25%
ピクセル情報: (3503, 2) [228 171 227]

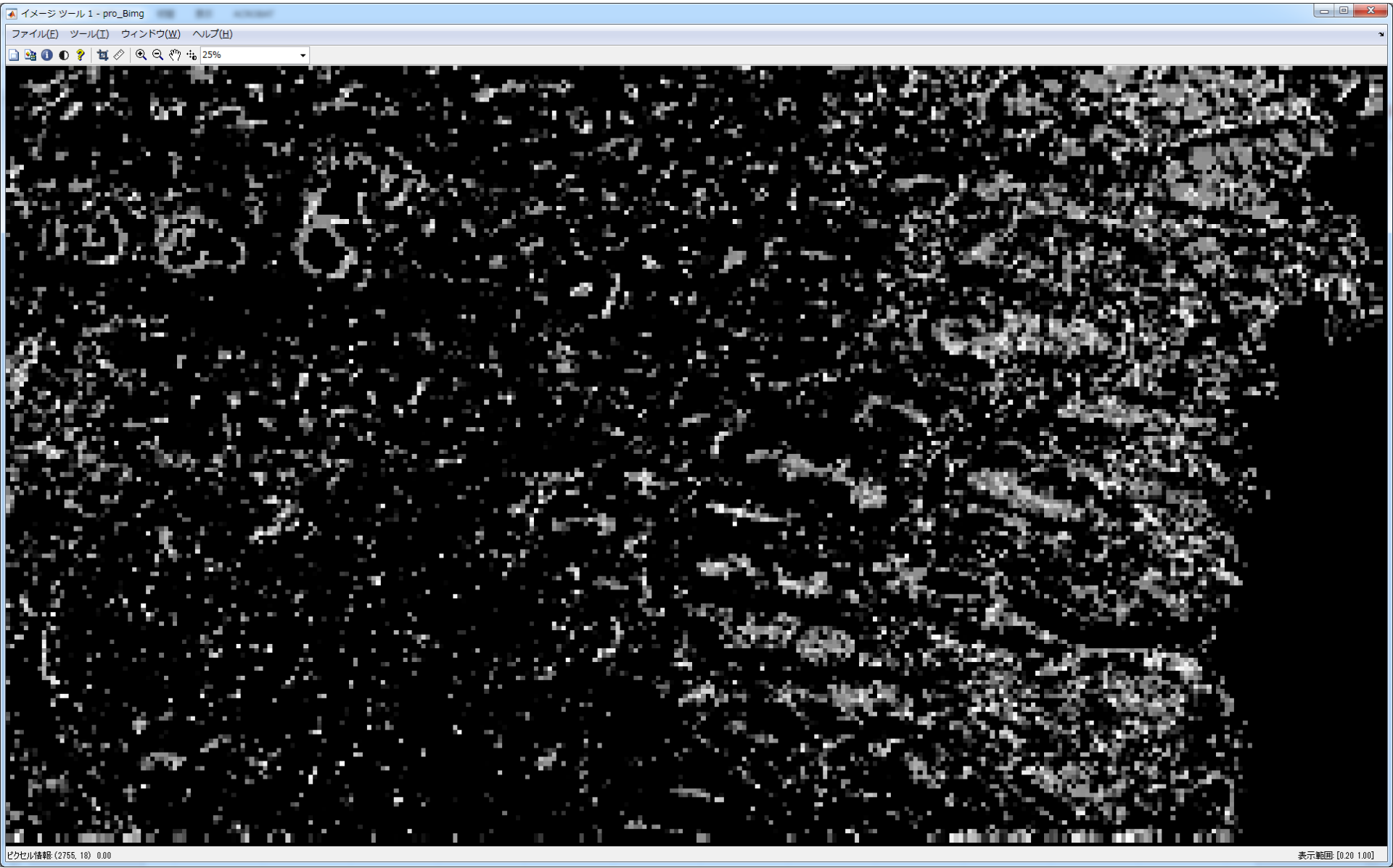
# CNNの構成1


イメージ ツール 2 - pro_Mimg
ファイル(F) ツール(T) ウィンドウ(W) ヘルプ(H)
25%
ピクセル情報: (2923, 14) 0.75
表示範囲: [0.00 0.80]

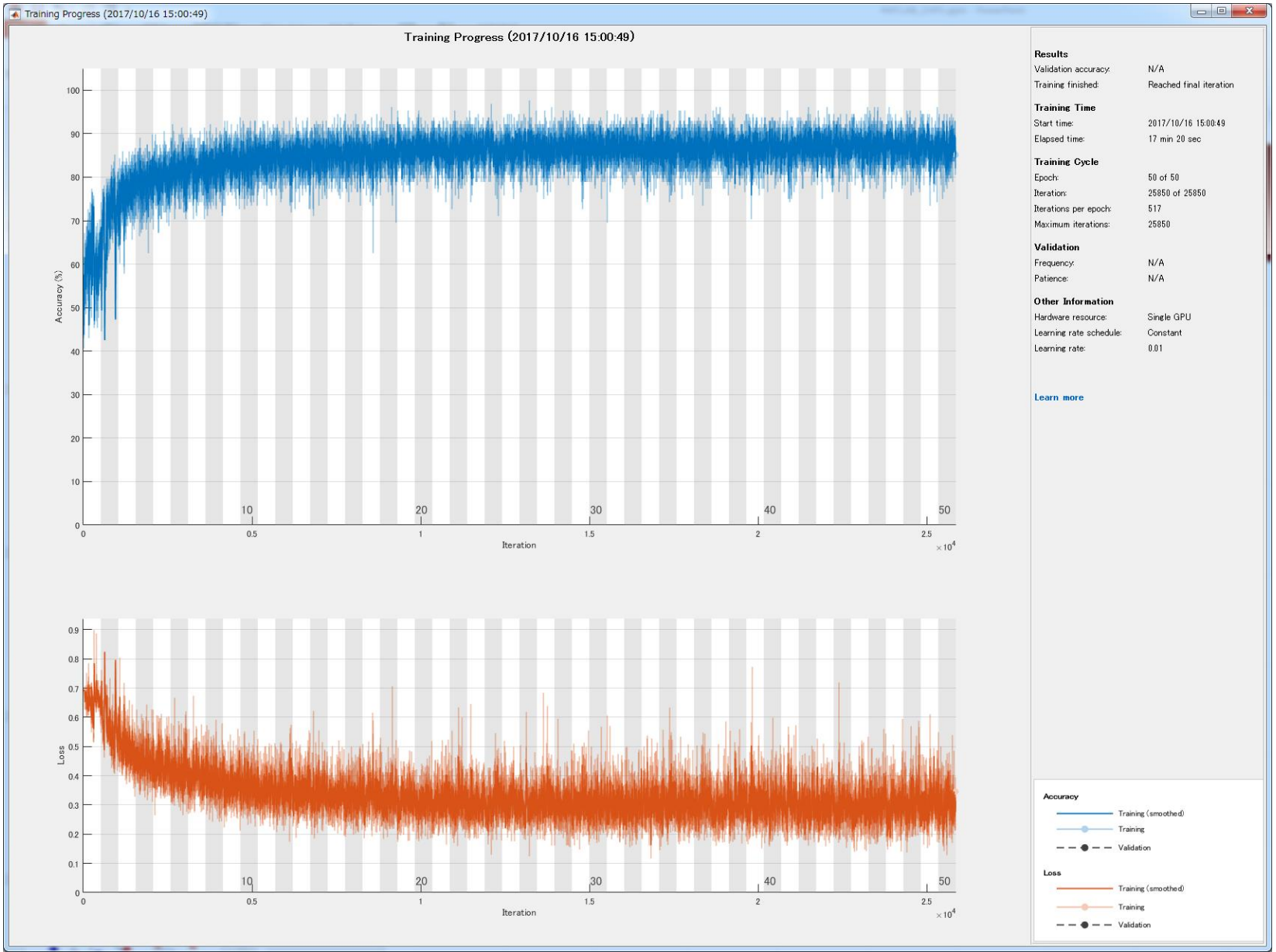
# CNNの構成1


イメージ ツール 1 - B03.tif
ファイル(F) ツール(T) ウィンドウ(W) ヘルプ(H)
25%
ピクセル情報: (2359, 38) [193 142 208]

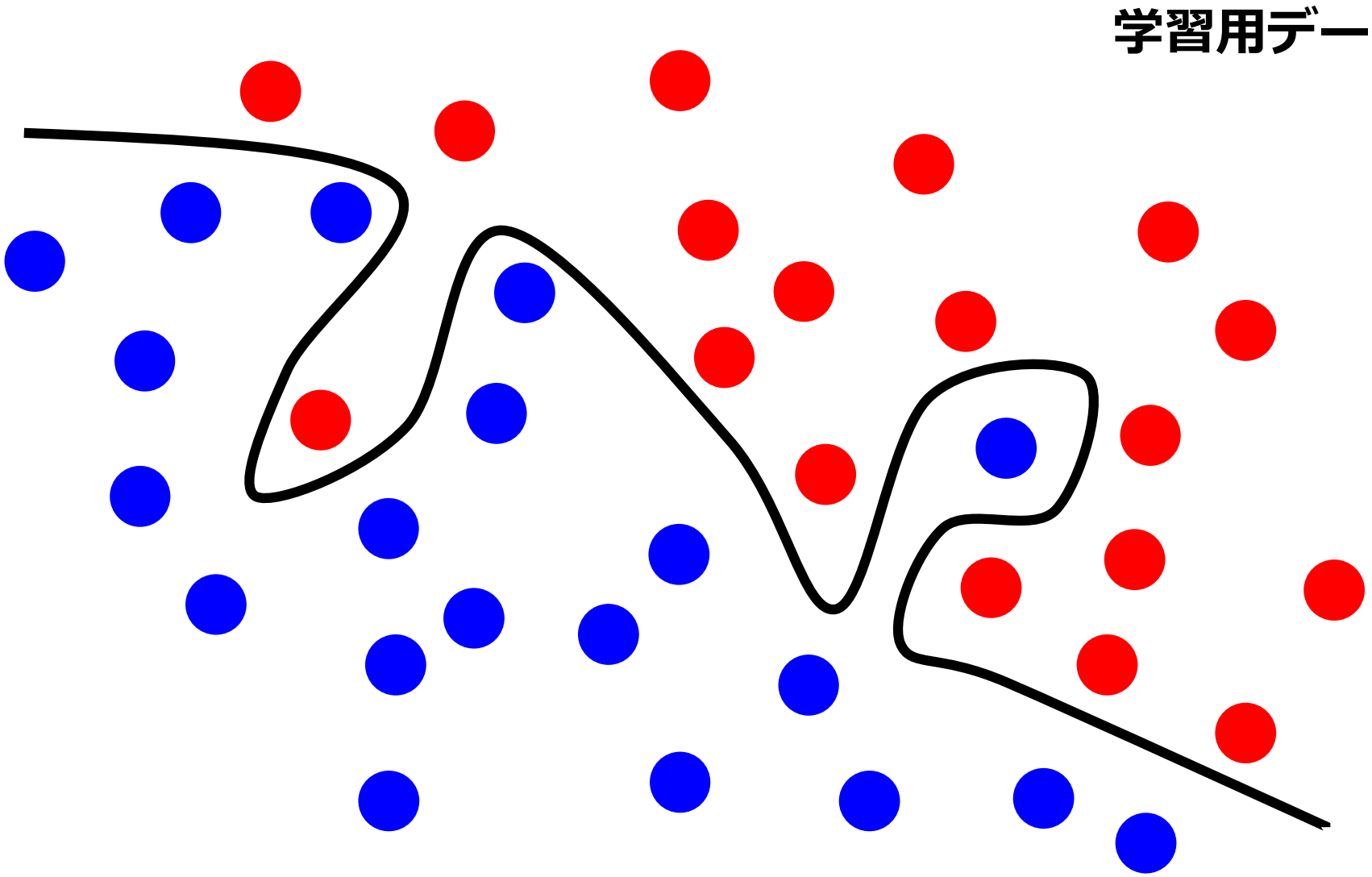
# CNNの構成1


イメージ ツール 1 - pro_Bimg
ファイル(F) ツール(T) ウィンドウ(W) ヘルプ(H)
25%
ピクセル情報: (2755, 18) 0.00
表示範囲: [0.20 1.00]

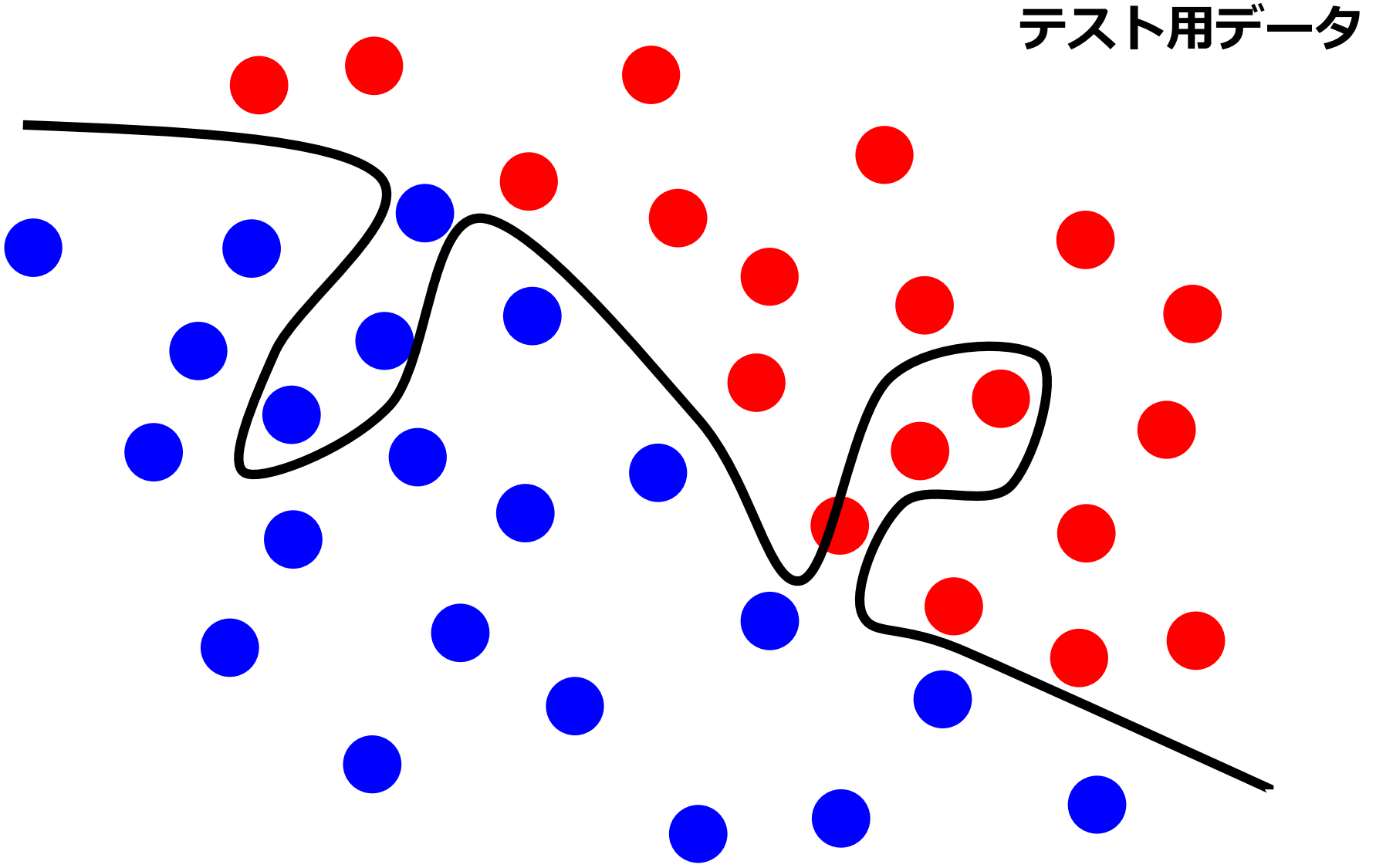
# CNNの構成1


Training Progress (2017/10/16 15:00:49)
Training Progress (2017/10/16 15:00:49)
Accuracy (%)
Iteration
Loss
Iteration
Results
Validation accuracy: N/A
Training finished: Reached final iteration
Training Time
Start time: 2017/10/16 15:00:49
Elapsed time: 17 min 20 sec
Training Cycle
Epoch: 50 of 50
Iteration: 25850 of 25850
Iterations per epoch: 517
Maximum iterations: 25850
Validation
Frequency: N/A
Patience: N/A
Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.01
Learn more
Accuracy
Training (smoothed)
Training
Validation
Loss
Training (smoothed)
Training
Validation

# 過学習



# 過学習



# 過学習の回避

C:¥Users¥Nakayama¥Documents¥MATLAB¥CNN_pathology1014.m

エディター　パブリッシュ　表示

```
75      % 検証用データの作成
76 -    idx = randperm(size(train_stack,4),2000);
77 -    valImages = train_stack(:,:,:,idx);
78 -    train_stack(:,:,:,idx) = [];
79 -    valLabels = train_label(idx);
80 -    train_label(idx) = [];
```

スクリプト　行 101 列 40

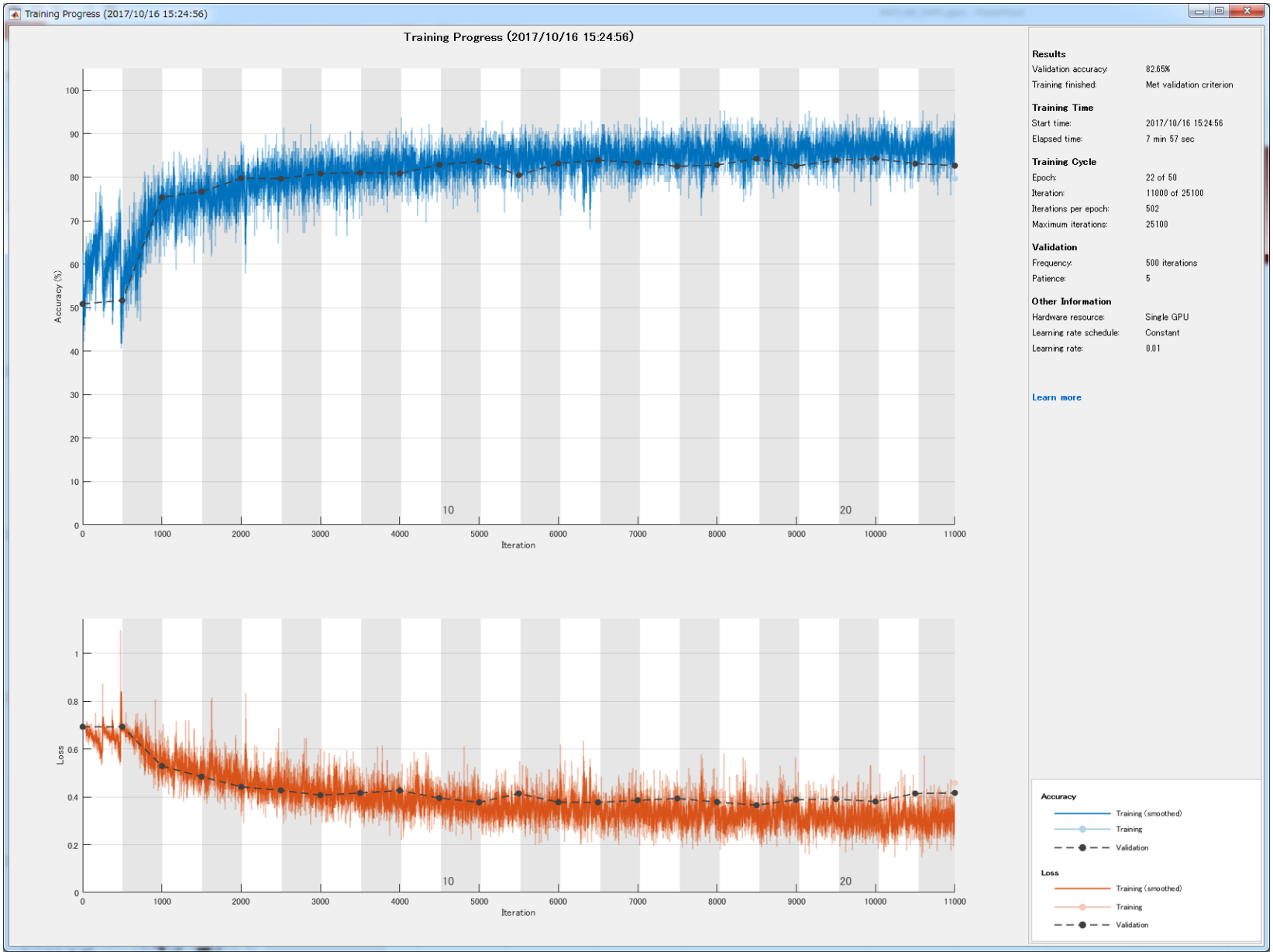
C:¥Users¥Nakayama¥Documents¥MATLAB¥CNN_pathology1014.m

エディター　パブリッシュ　表示

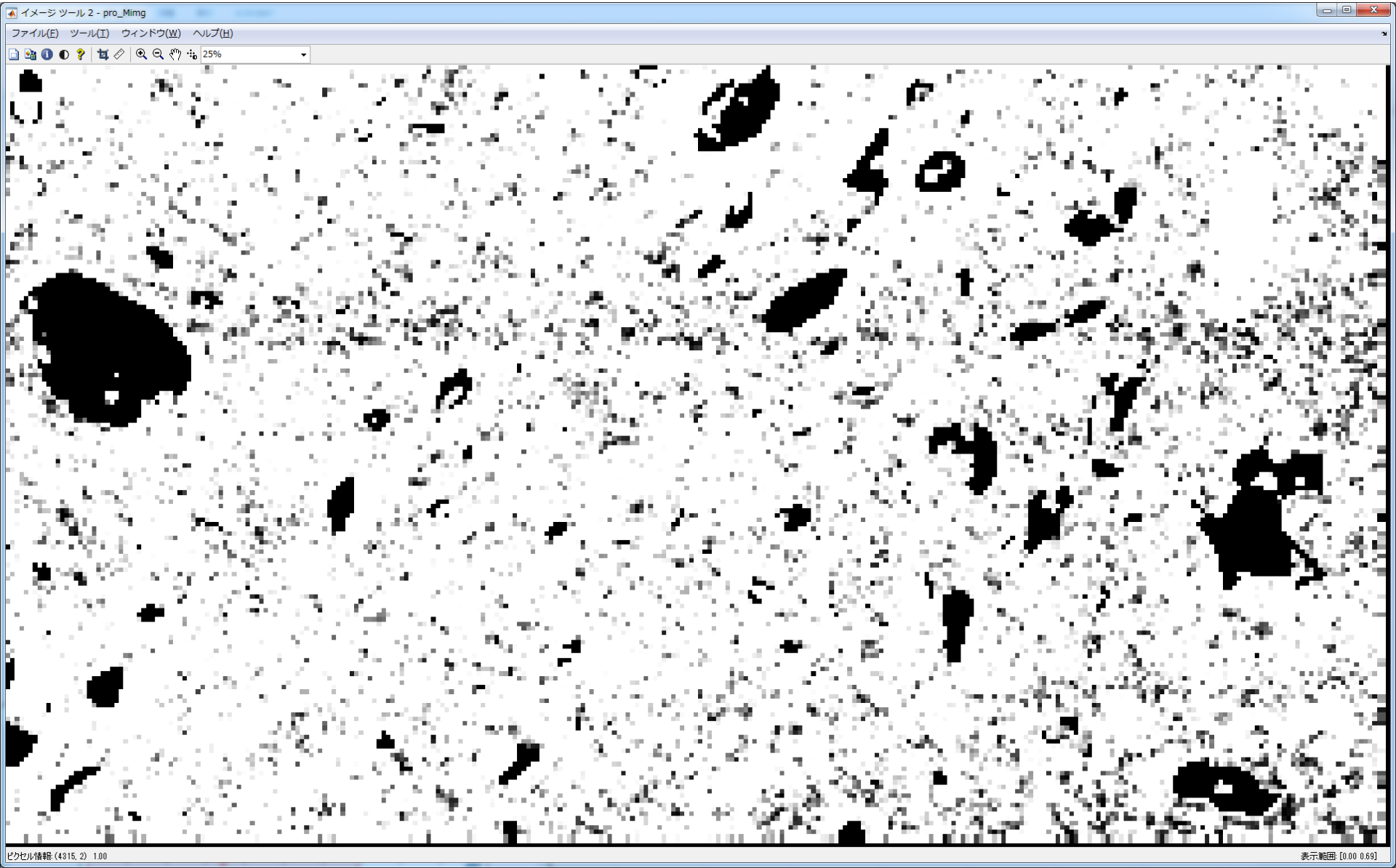
```
118     % Train CNN Using Training Image Data
119     % Set the network training options
120 -   opts = trainingOptions('sgdm', ...
121         'Shuffle','every-epoch', ...
122         'MaxEpochs',50, ...
123         'Plots','training-progress', ...
124         'ValidationData', {valImages,valLabels},...
125         'ValidationFrequency',500);
```

スクリプト　行 81 列 1

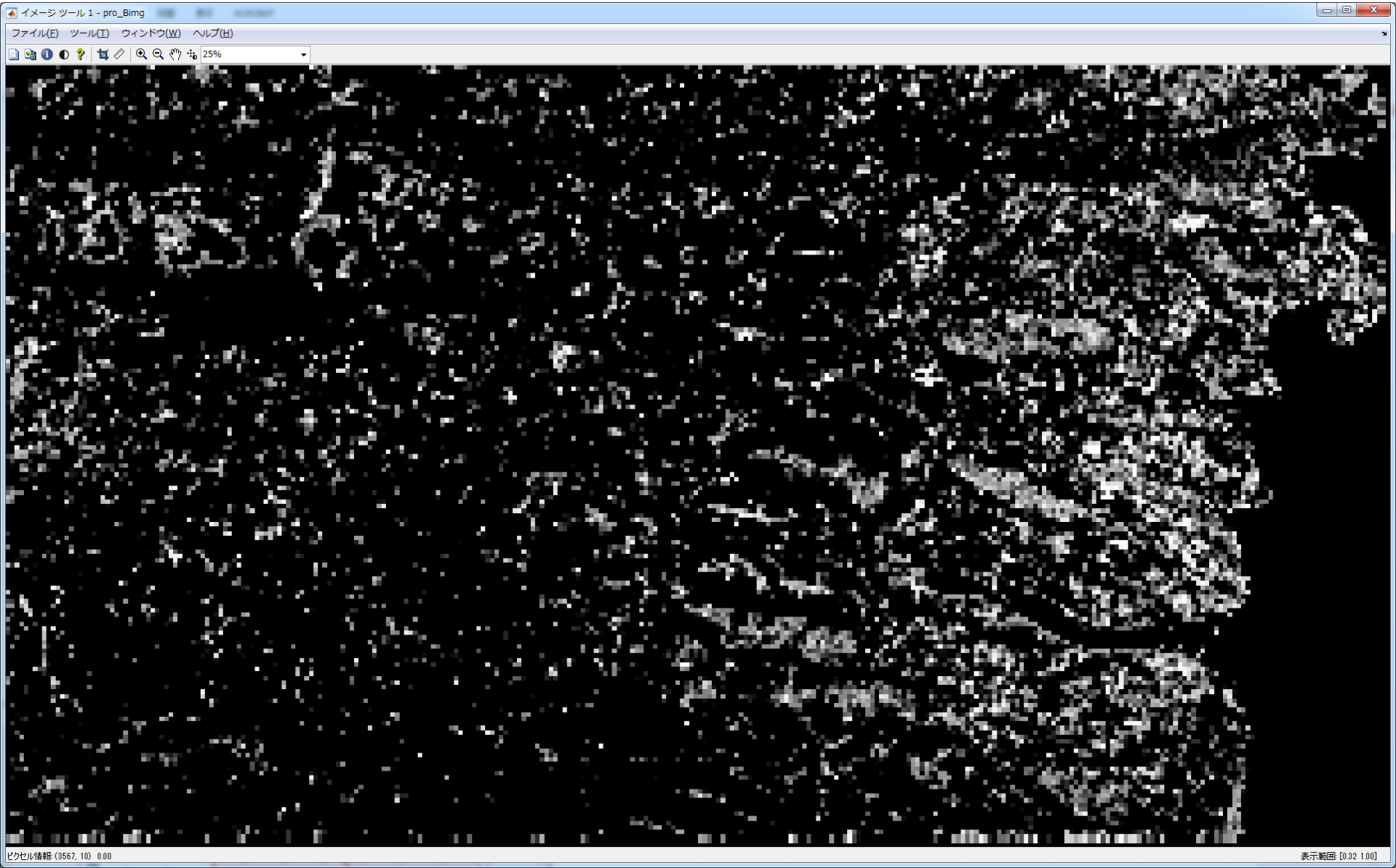
# 過学習の回避


Training Progress (2017/10/16 15:24:56)
Training Progress (2017/10/16 15:24:56)
Accuracy (%)
Iteration
Loss
Iteration
10
20
Results
Validation accuracy: 82.65%
Training finished: Met validation criterion
Training Time
Start time: 2017/10/16 15:24:56
Elapsed time: 7 min 57 sec
Training Cycle
Epoch: 22 of 50
Iteration: 11000 of 25100
Iterations per epoch: 502
Maximum iterations: 25100
Validation
Frequency: 500 iterations
Patience: 5
Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.01
Learn more
Accuracy
Training (smoothed)
Training
Validation
Loss
Training (smoothed)
Training
Validation

# 過学習の回避


イメージ ツール 2 - pro_Mimg
ファイル(F) ツール(T) ウィンドウ(W) ヘルプ(H)
25%
ピクセル情報: (4315, 2) 1.00
表示範囲: [0.00 0.69]

# 過学習の回避


イメージ ツール 1 - pro_Bimg
ファイル(F) ツール(T) ウィンドウ(W) ヘルプ(H)
25%
ピクセル情報: (3567, 10) 0.00
表示範囲: [0.32 1.00]

# CNNの構成

対象画像，目的に最適なCNNの構成を見つけることが肝要

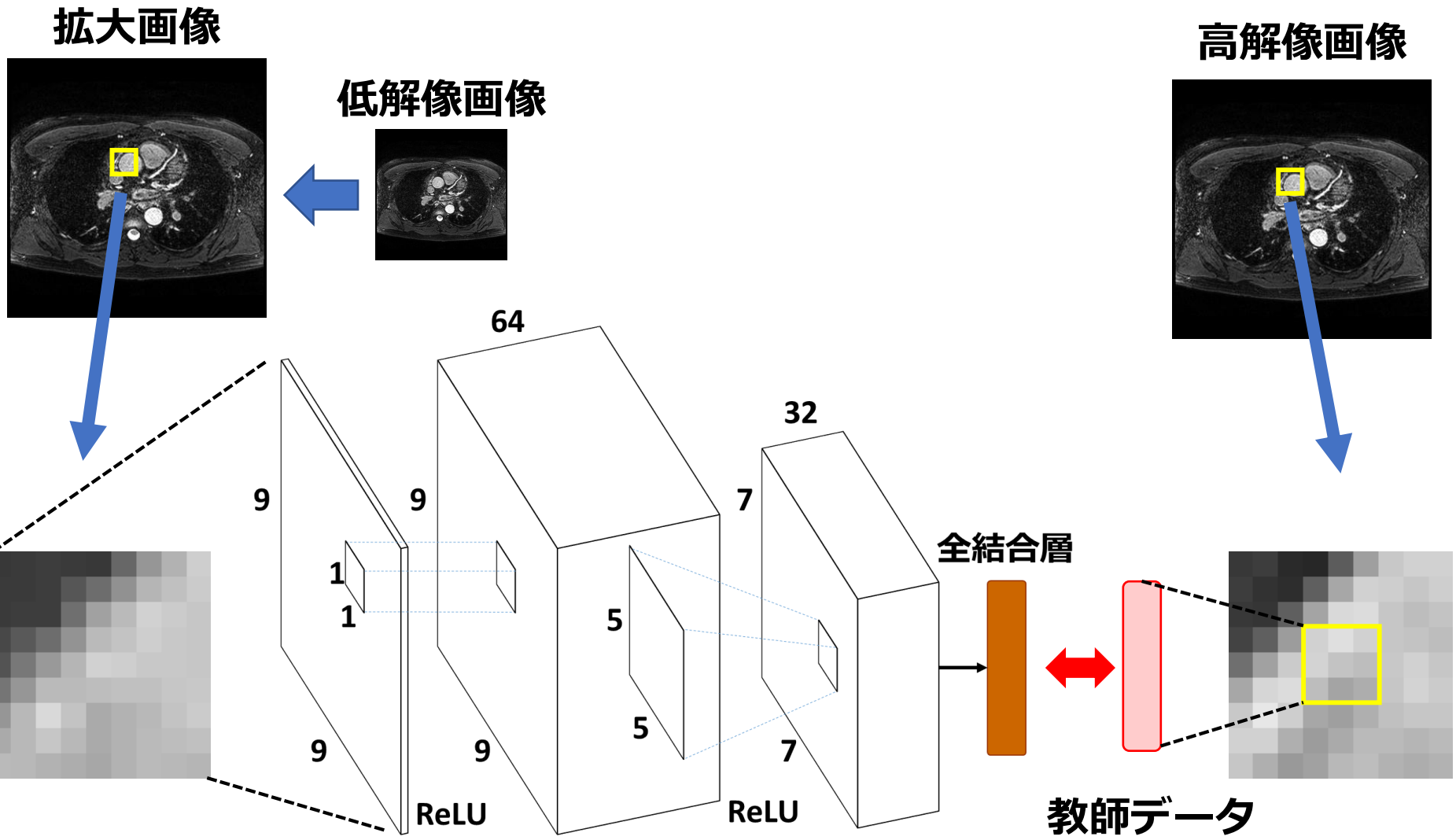
- ✓ 畳み込み層の数
- ✓ 各畳み込み層におけるフィルタサイズ，フィルタ数
- ✓ プーリングの有無
- ✓ 入力サイズ（ROIサイズ）
- ✓ 識別器の選択

```
[class, likelihood] = classify(learnedNet, test_ROI);
```

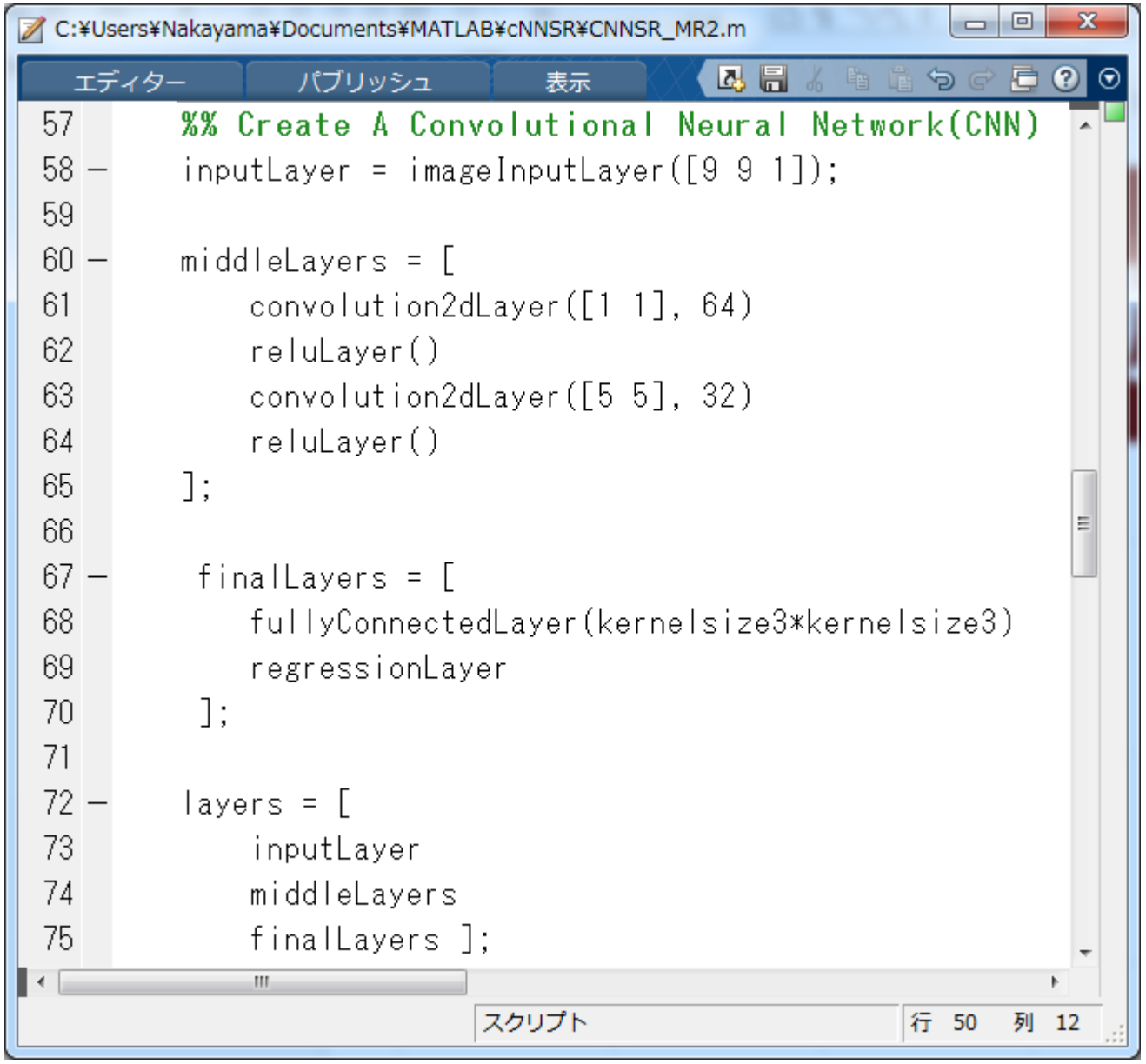
↓　サポートベクターマシンへ

```
class = fitcsvm(learnedNet, test_ROI);
```

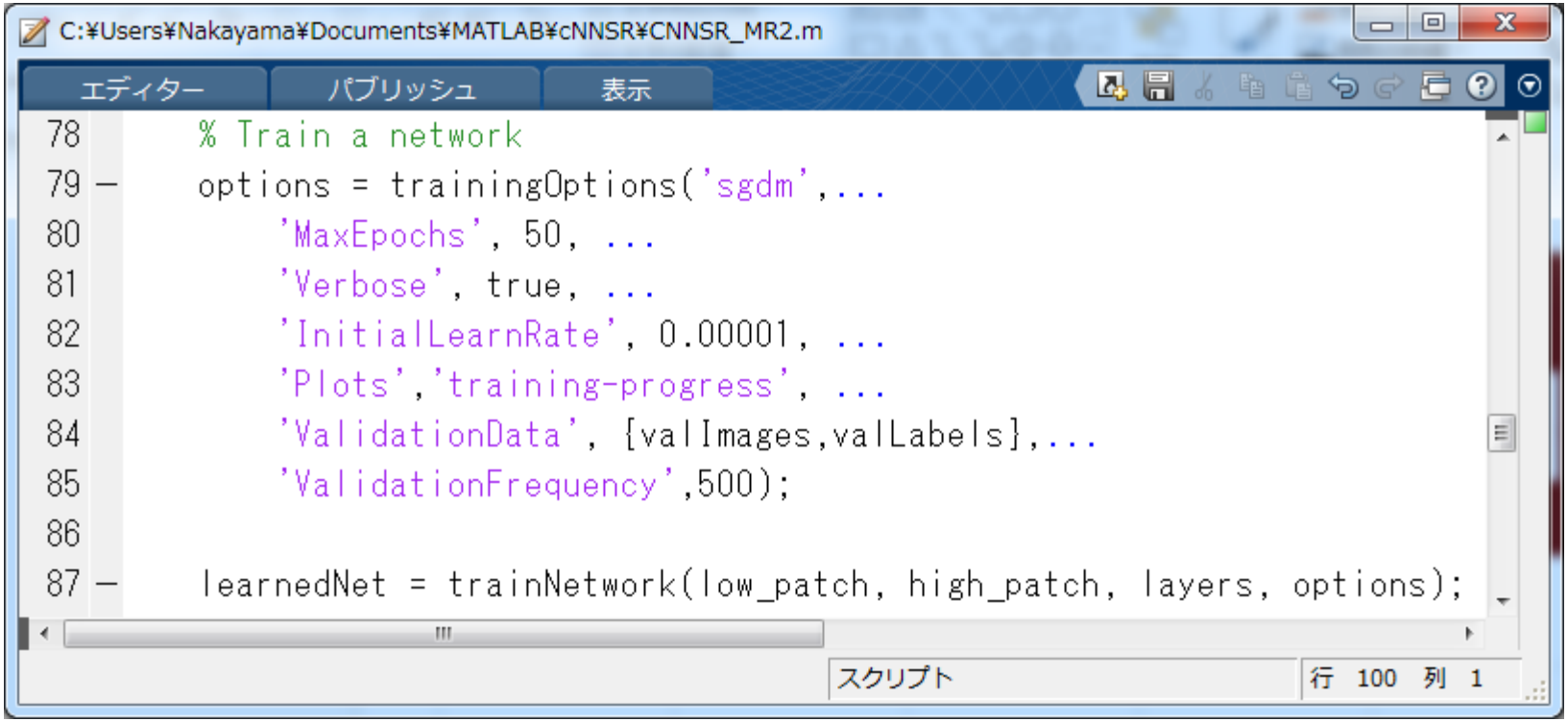
# 冠動脈MRAの高解像度化



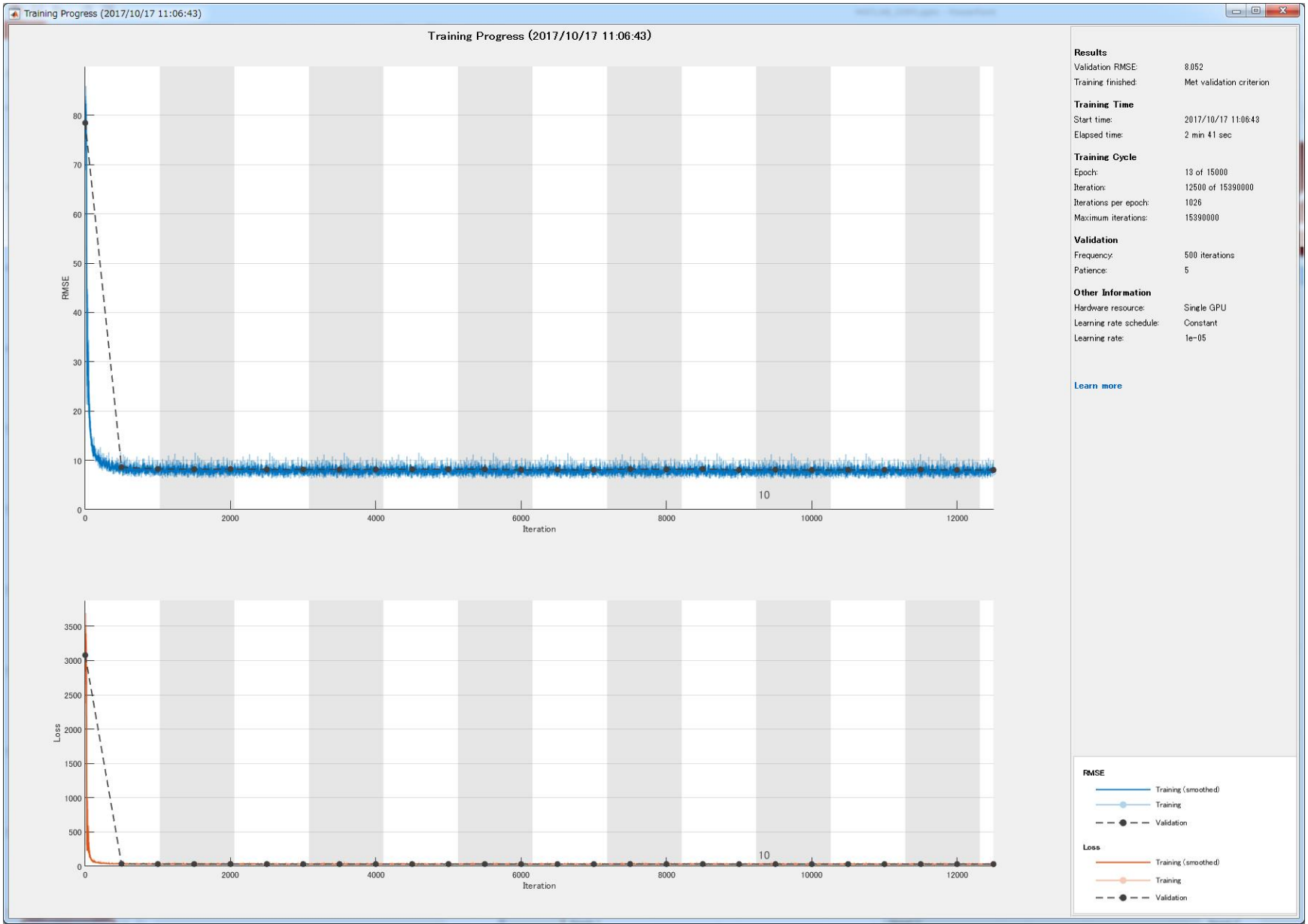
# 冠動脈MRAの高解像度化



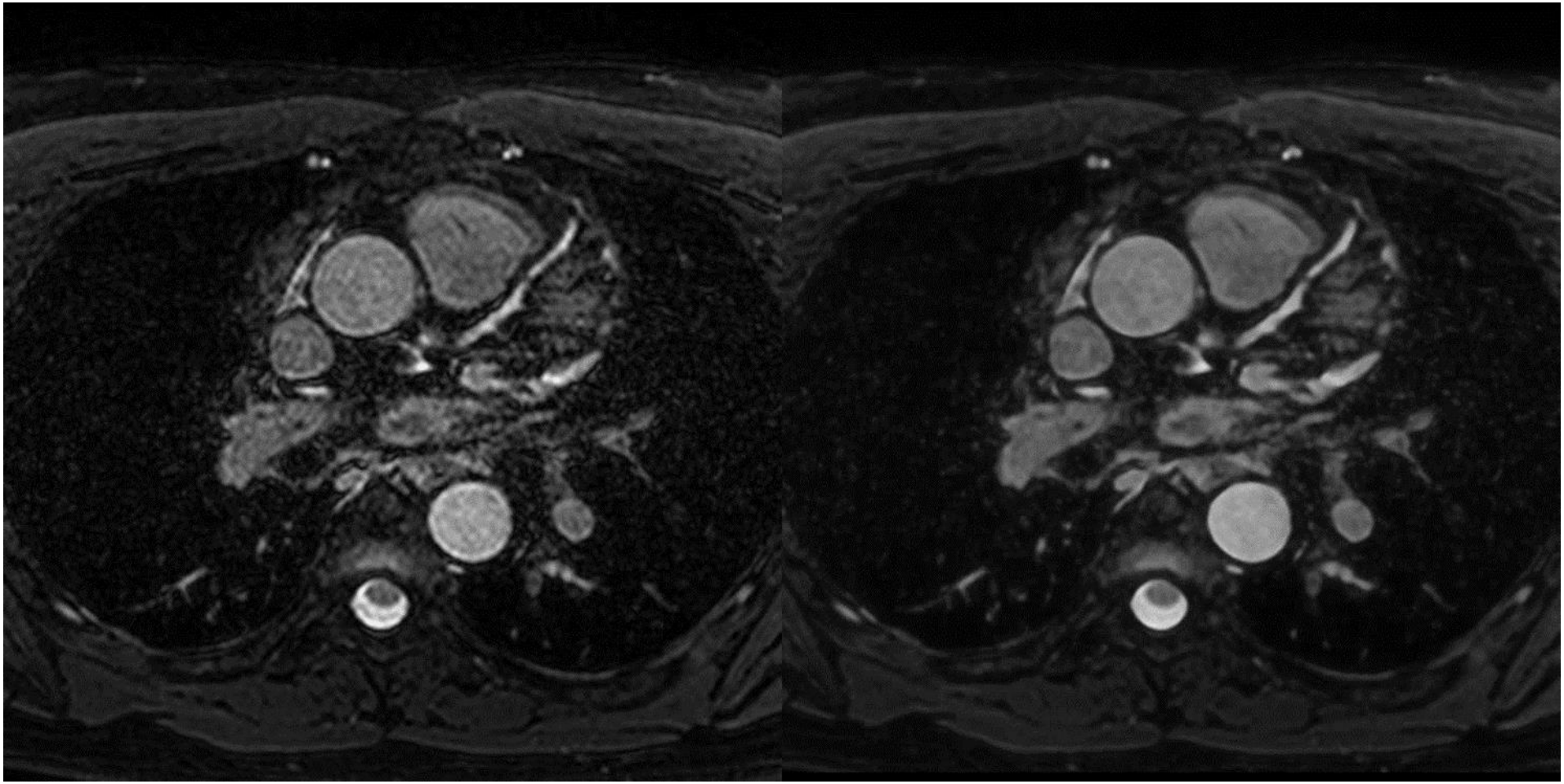
# 冠動脈MRAの高解像度化



# 冠動脈MRAの高解像度化



# 冠動脈MRAの高解像度化



冠動脈MRA

高解像度化MRA

# なぜMATLAB®か？

- デメリット
  - ✓ 高額
  - ✓ 機能追加までの時間

- メリット
  - ✓ 再構成法や医療画像解析ソフトなどMATLABで開発
  - ✓ 開発環境の構築が容易
  - ✓ プログラムが簡単
  - ✓ アルゴリズムを理解できていなくても利用可
  - ✓ File Exchangeなどユーザーコミュニティ
  - ✓ テクニカルサポート