画像分野におけるディープラーニングの新展開

MathWorks Japan

アプリケーション エンジニアリング部　テクニカルコンピューティング

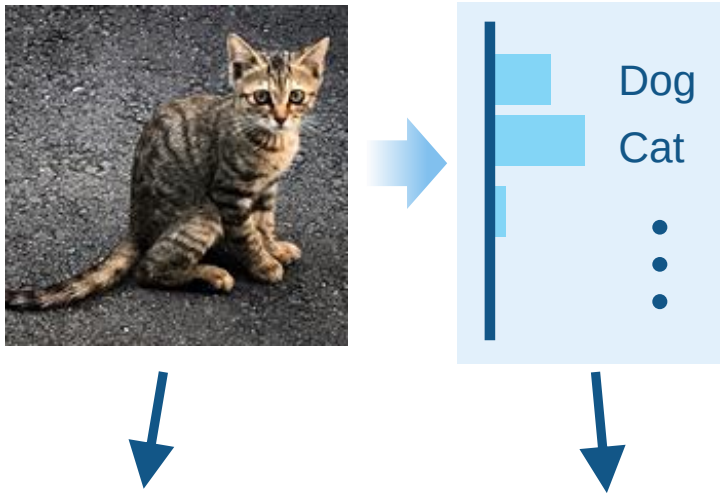
太田　英司

画像分野におけるディープラーニングの新展開

R2017a
R2017b

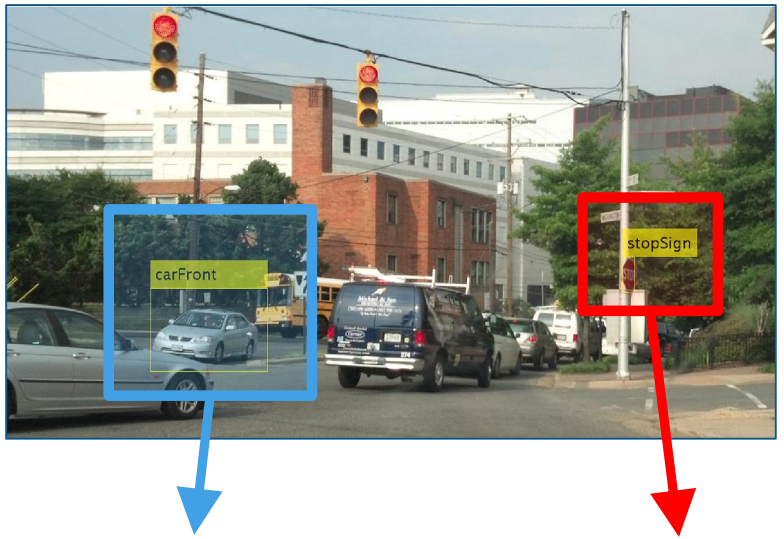
物体認識（画像全体）

CNN (Convolutional Neural Network)



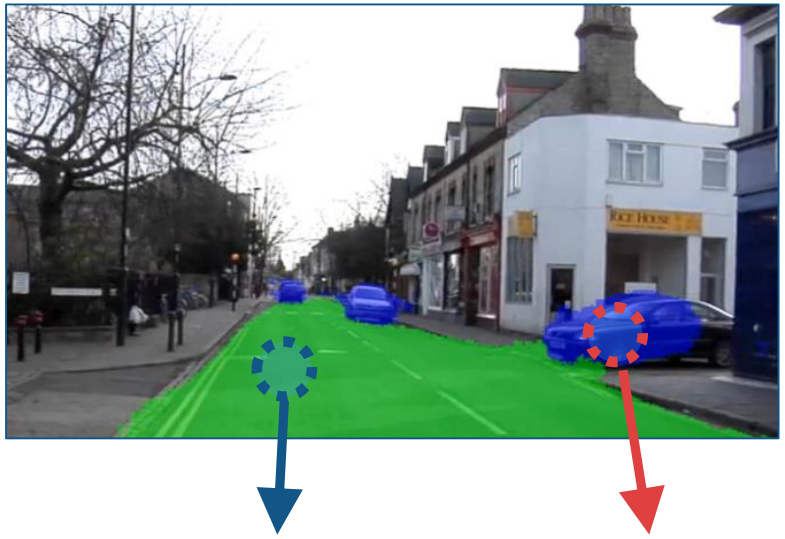
物体の検出と認識

R-CNN / Fast R-CNN / Faster R-CNN



物体認識（ピクセル単位）

SegNet / FCN

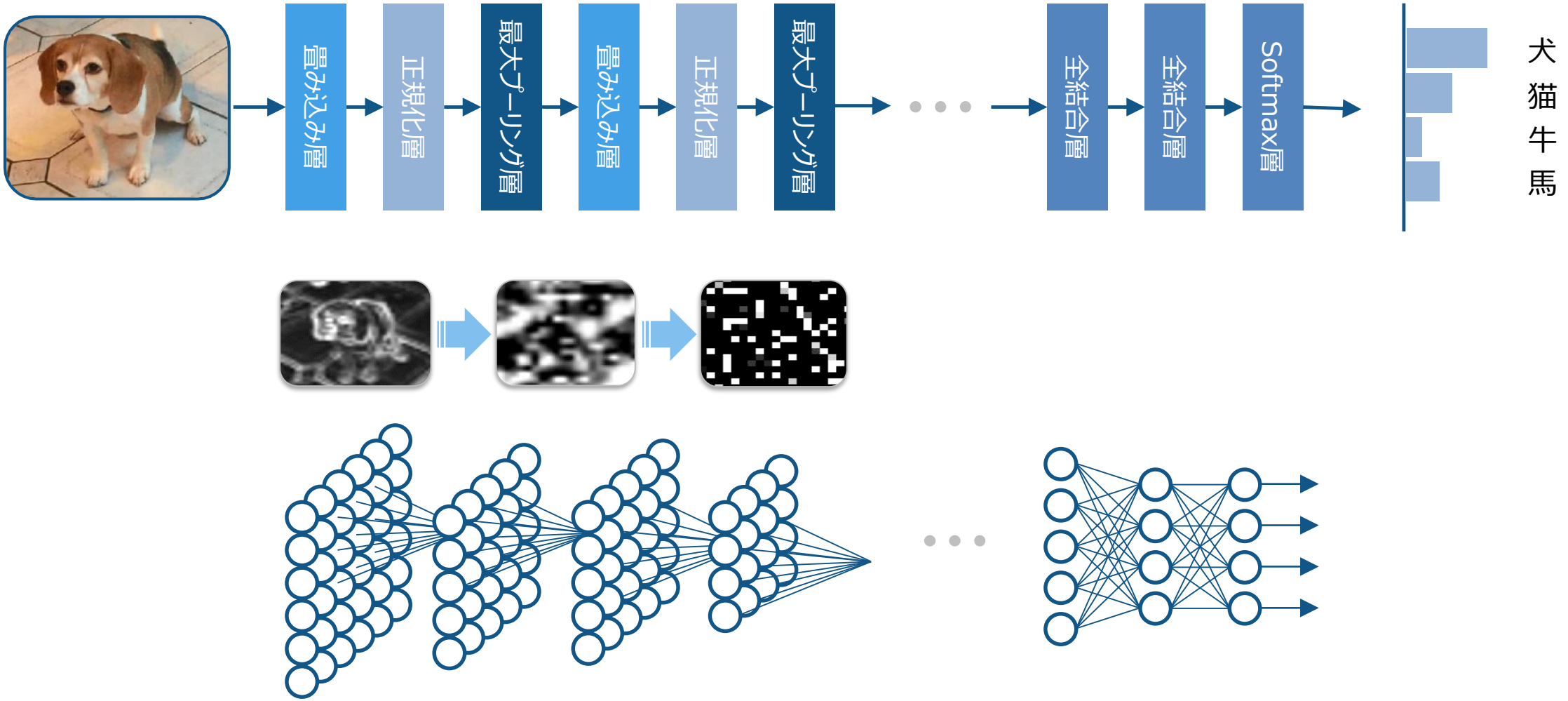


Agenda

- 物体認識（画像全体）
  - CNN の基礎（復習）
- 物体の検出と認識
  - R-CNN / Fast R-CNN / Faster R-CNN
- 物体認識（ピクセル単位）
  - Semantic Segmentation (SegNet)
- その他の新機能
  - CNN の回帰
  - 学習済みモデル / インポート機能

画像認識（画像全体）
CNN (Convolutional Neural Network)

R2016a



物体の検出と認識
R-CNN / Fast R-CNN / Faster R-CNN

R2017a

R-CNNに自動車の前面と停止標識を学習させた場合の検出例

停止標識（Stop Sign）

自動車の前面（Car Front）

物体認識（ピクセル単位）
Semantic Segmentation

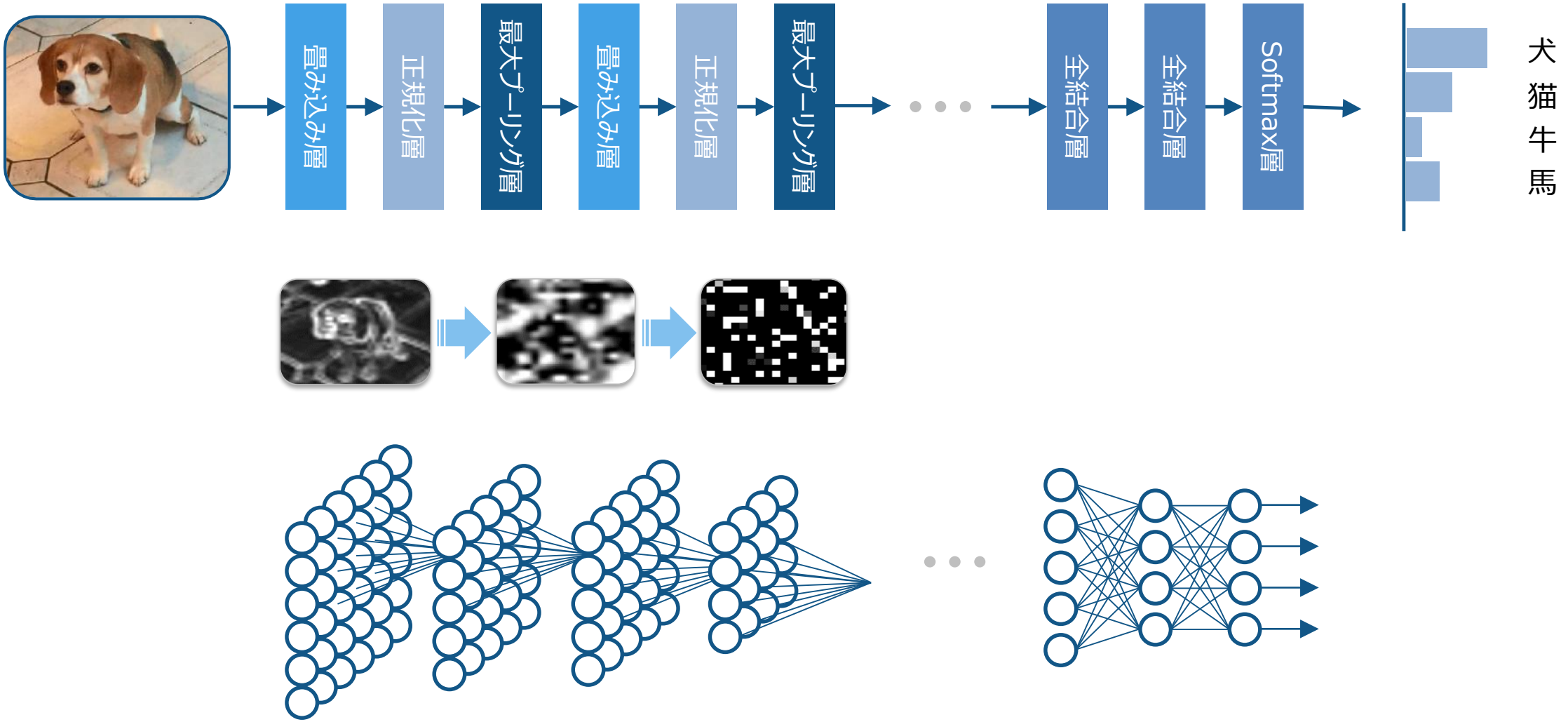
R2017b

物体識別（画像全体）

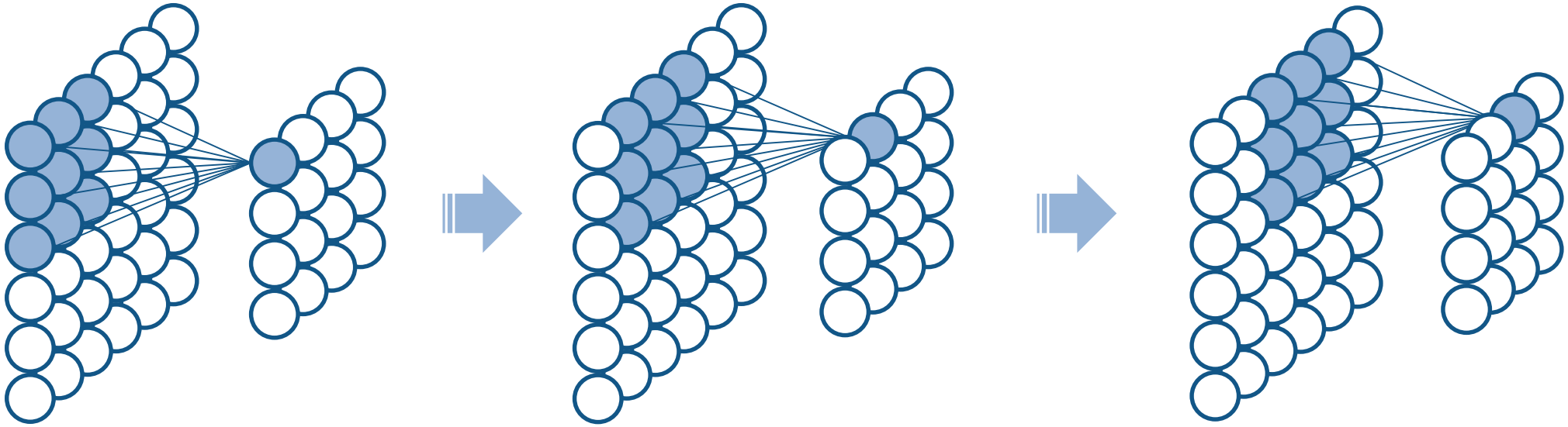
CNN (Convolutional Neural Network)

画像認識（画像全体）
CNN (Convolutional Neural Network)

R2016a



Convolution Layer（畳み込み層） / Pooling Layers (プーリング層)



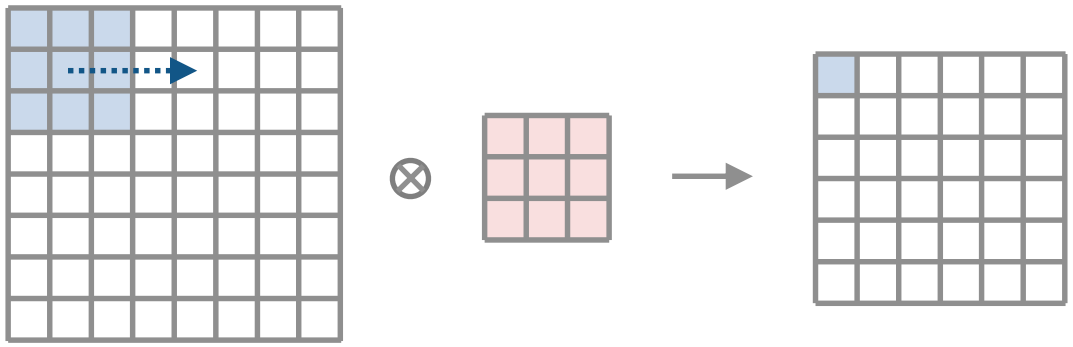
層と層の間を一部のみ連結して、ウェイトを共有すると、ニューラルネットで畳み込みが表現できる

Convolution Layer（畳み込み層） / Pooling Layer（プーリング層）

Convolution Layer（畳み込み層）

→ 画像のフィルタ処理に相当する処理

→ 特徴抽出器としての役割

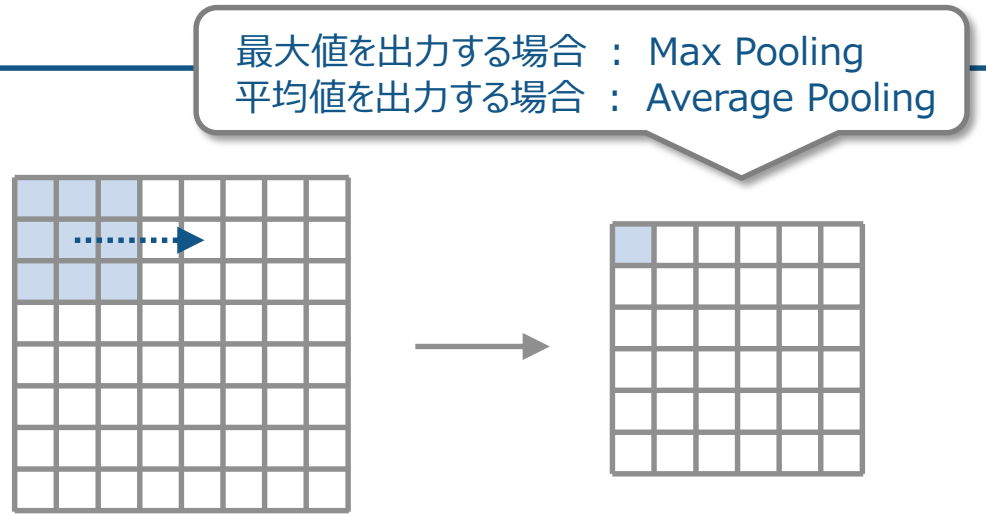


Pooling Layer（プーリング層）

→ 領域内の最大値または平均値を出力

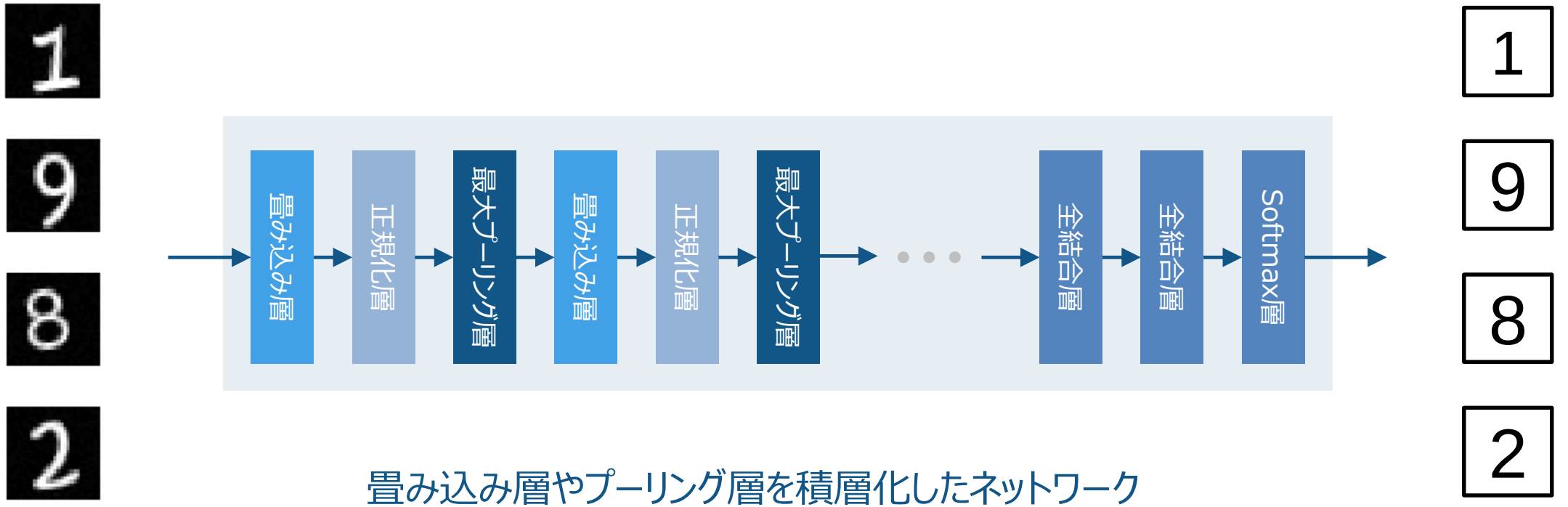
→ 平行移動等に対するロバスト性に関係

→ ストライドと呼ばれる間引きを行うこともある



【例題】手書き文字の認識

畳み込みニューラルネットによる手書き文字の認識



畳み込み層やプーリング層を積層化したネットワークを定義し、誤差逆伝搬法により学習を行う

手書き文字
28 x 28 pixel

整数 (0-9)

畳み込みニューラルネットの構築と学習

28×28 ピクセルの画像（数字）を認識させる例題でのネットワーク構築の例

```
layers = [ ...
    imageInputLayer([28 28 1], 'Normalization', 'none');
    convolution2dLayer(5, 20);
    reluLayer();
    maxPooling2dLayer(2, 'Stride', 2);
    fullyConnectedLayer(10);
    softmaxLayer();
    classificationLayer()];
```

畳み込み層・プーリング層・正規化層
などの層を積み上げて定義

```
opts = trainingOptions('sgdm', 'MaxEpochs', 50);
net = trainNetwork(XTrain, TTrain, layers, opts);
```

学習率や最大反復数などを定義して
学習の関数を呼び出す

http://www.mathworks.com/help/releases/R2016a/nnet/ref/trainnetwork.html

ILSVRC に登場した有名なネットワーク（Alex Net, VGG Net）

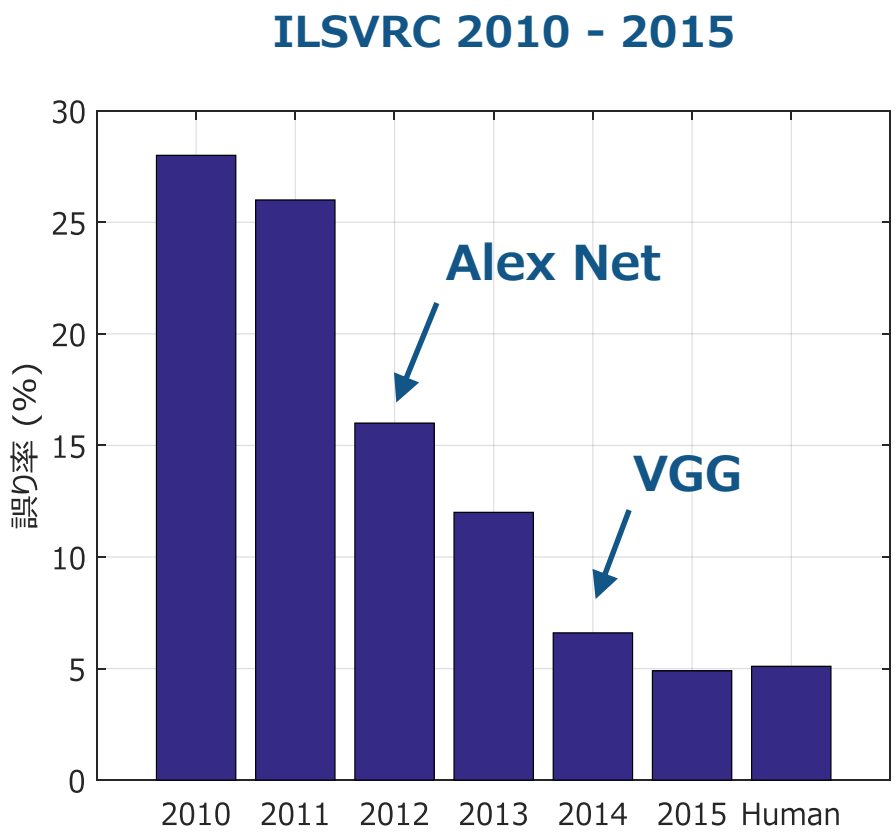
Alex Net の登場後、より深いネットワークが試されるようになった

Alex Net

- トロント大学 Hinton のチームにより発表
- NVIDIA® GeForce® GTX 580 2機 による 5～6日間の学習
- ILSVRC 2012 において優勝した記念碑的なネットワーク

VGG Net

- Oxford大学 Visual Geometry Group により発表
- NVIDIA® GeForce® TITAN Black 4機 による 2～3週間の学習
- ILSVRC 2014 において２位の記録を残したネットワーク



Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton "ImageNet Classification with Deep Convolutional Neural Networks" In NIPS, pp.1106-1114, 2012

K. Simonyan, A. Zisserman "Very Deep Convolutional Networks for Large-Scale Image Recognition" arXiv technical report, 2014

物体の検出と識別

R-CNN / Fast R-CNN / Faster R-CNN

R-CNN（Regions with CNN features） とは？

R2016b

CNNにコンピュータビジョンの手法を組み合わせた物体検出・識別の手法

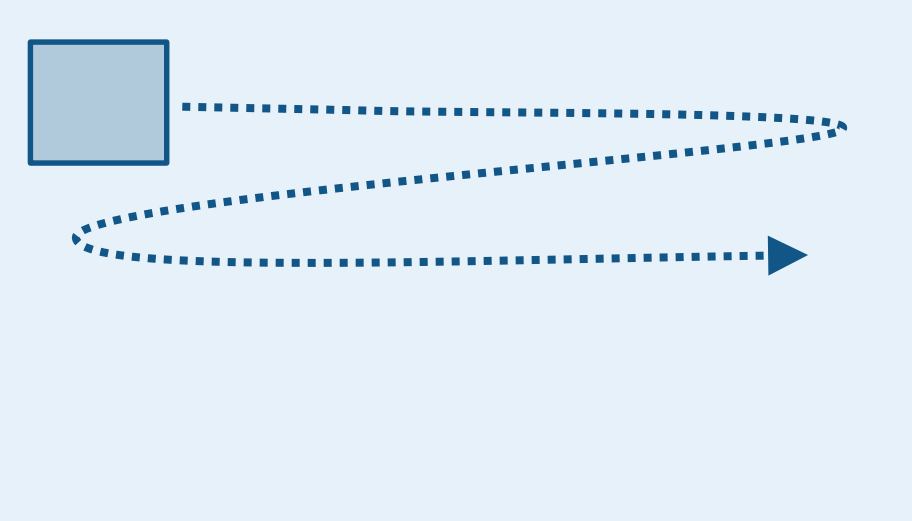
R-CNNに自動車の前面と停止標識を学習させた場合の検出例

停止標識（Stop Sign）

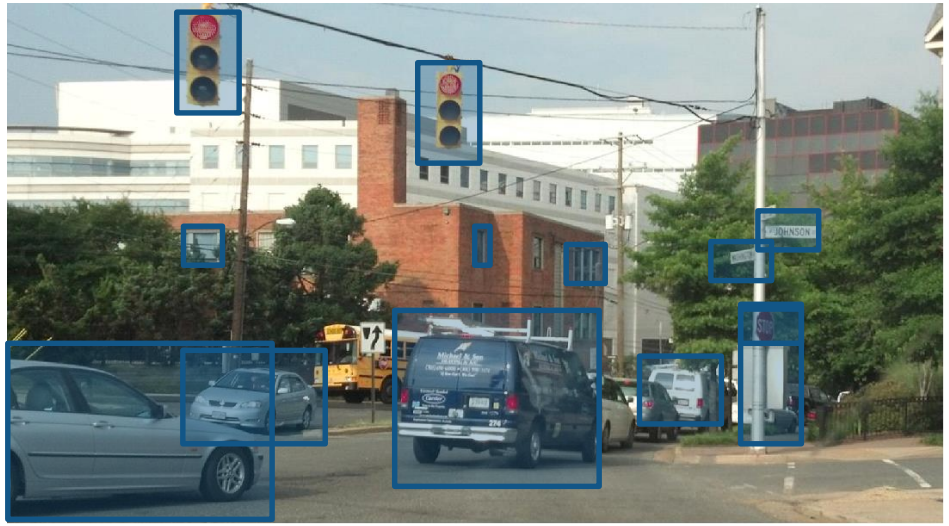
自動車の前面（Car Front）

R-CNN（Regions with CNN features） とは？

Exhaustive Search では、領域の候補が非常に多くなり、高性能な識別器との組み合わせが難しかった

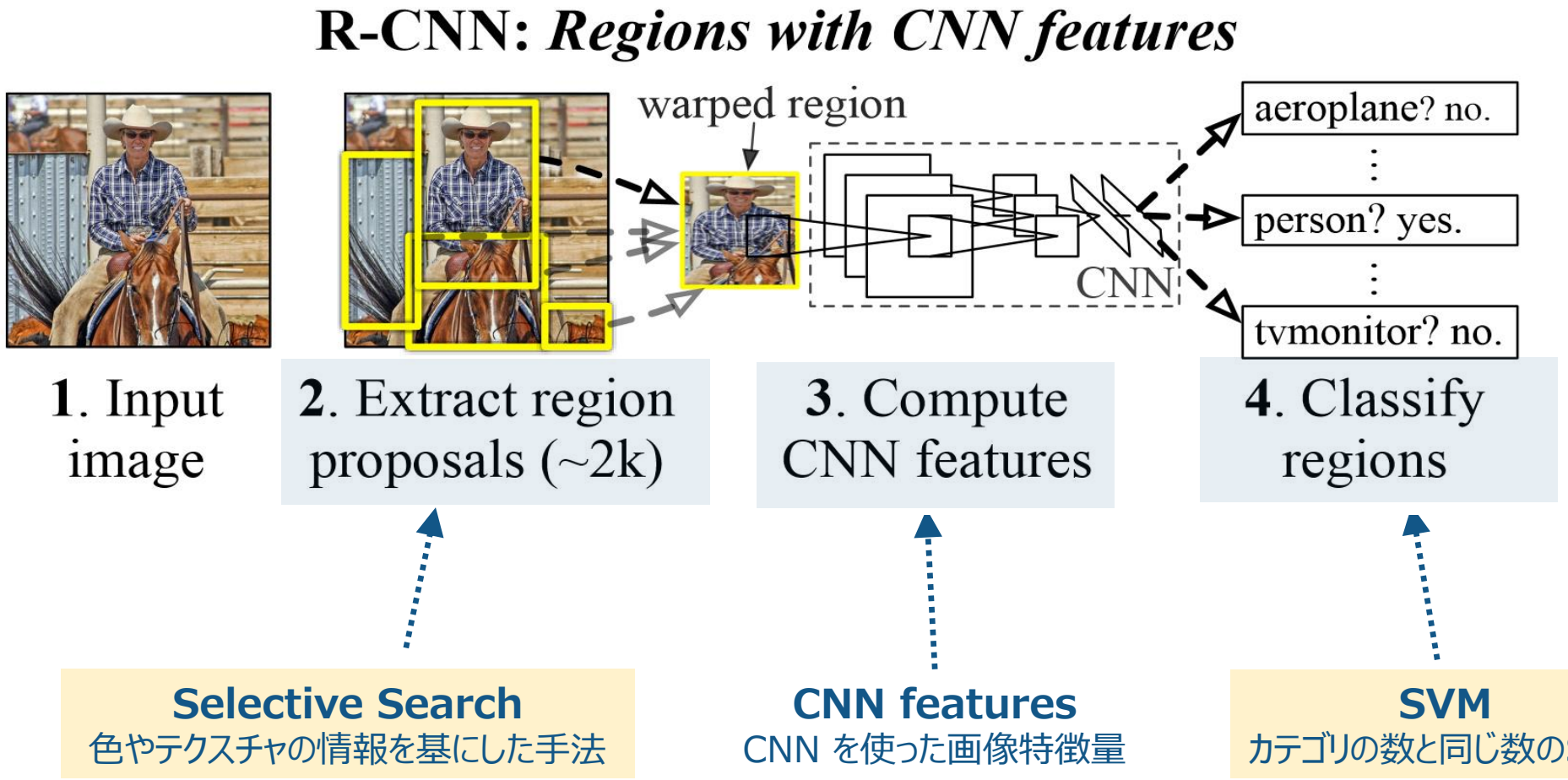


Exhaustive Search
Sliding Window を使ったアルゴリズム。サイズや場所を変えながら網羅的に探索する。顔検出などのアルゴリズムなどでもよく利用されている。



Selective Search
色やテクスチャの情報を基にしたアルゴリズム。物体らしき部分を選んで提案してくれる（通常2000個くらいの候補を生成することが多い）

R-CNN（Regions with CNN features） とは？



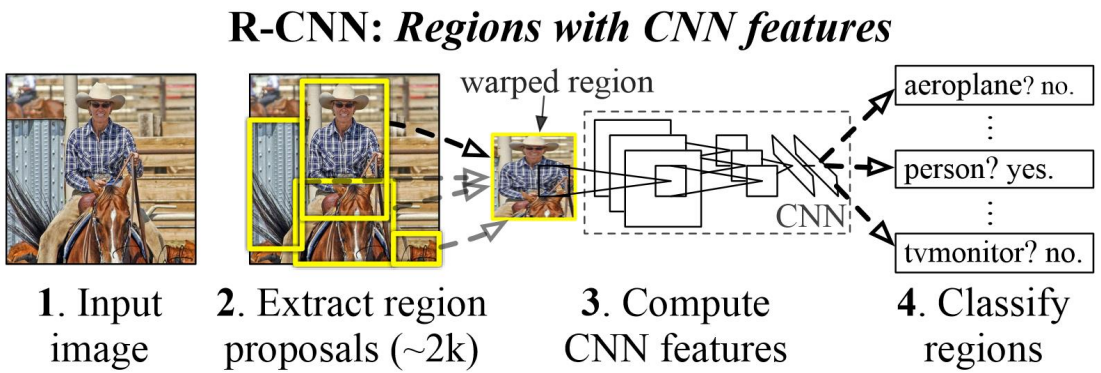
Selective Search
色やテクスチャの情報を基にした手法

CNN features
CNN を使った画像特徴量

SVM
カテゴリの数と同じ数のSVM

Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." CVPR 2014

MATLAB® における R-CNN



◀····· **元論文のアルゴリズム**

MATLAB では、実はR-CNN の元論文のアルゴリズムを若干改良して実装している

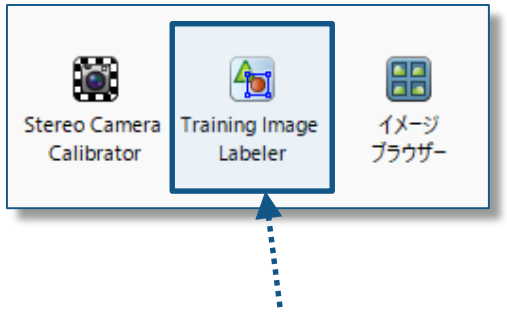
	領域候補 **Region Proposals**	**特徴抽出** **Feature Extraction**	**分類** **Classification**
元論文 ·····▶	**Selective Search** 色やテクスチャの情報を基にした手法	**CNN features** CNN を使った画像特徴量	**SVM** カテゴリの数と同じ数のSVM
MATLAB ·····▶	**Edge Boxes** 画像のエッジの情報を基にした手法		**Neural Network** CNNの後段を付け替え

[1] Girshick, R., J. Donahue, T. Darrell, and J. Malik. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation."*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587

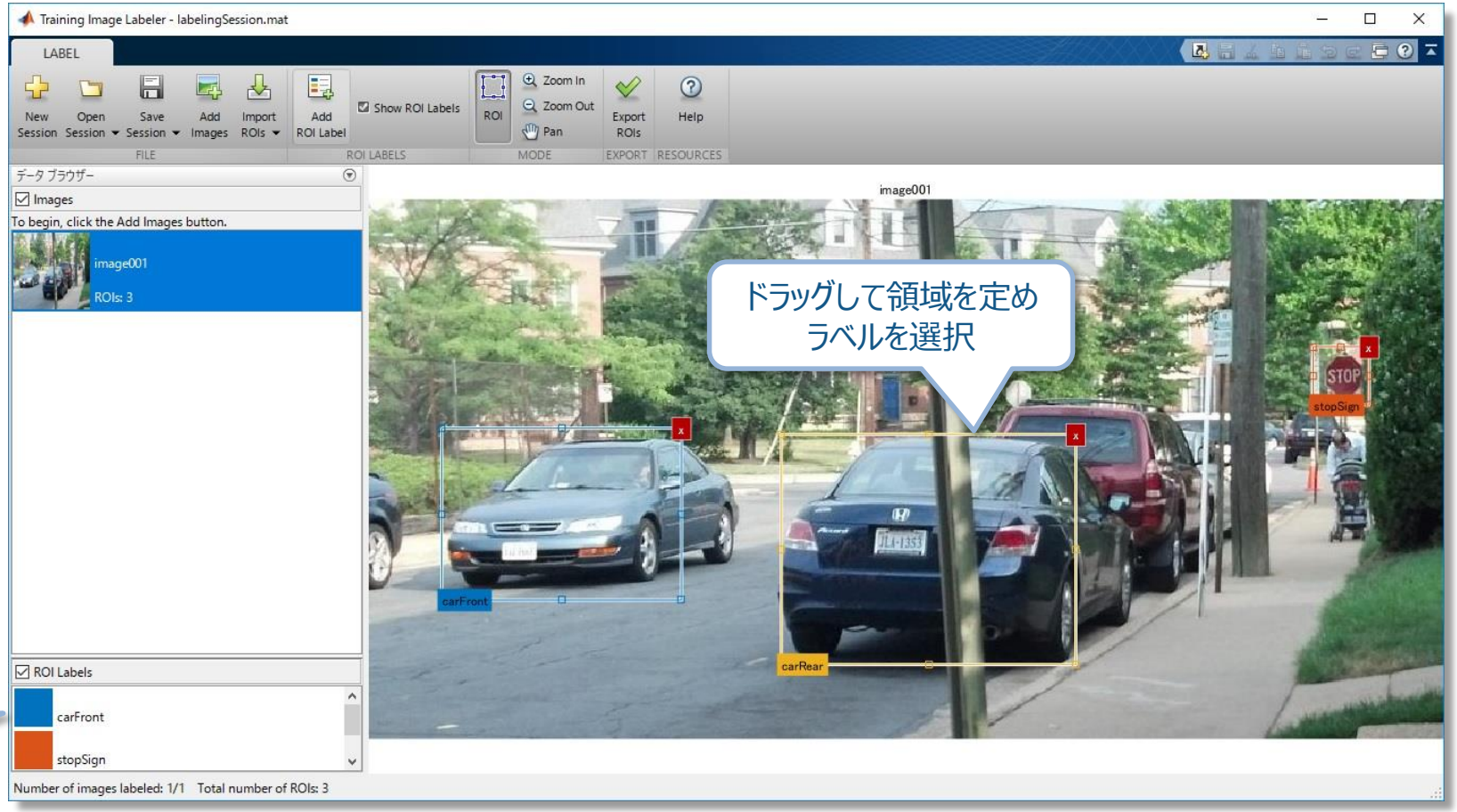
[2] Zitnick, C. Lawrence, and P. Dollar. "Edge Boxes: Locating Object Proposals from Edges." Computer Vision-ECCV, Springer International Publishing. 2014, pp. 391–405.

Image Labeler App

面倒で手間のかかるラベリングも専用ツールで誰にでも簡単に行うことができます



起動はアプリケーションタブにある
上記のアイコンをクリックするだけ



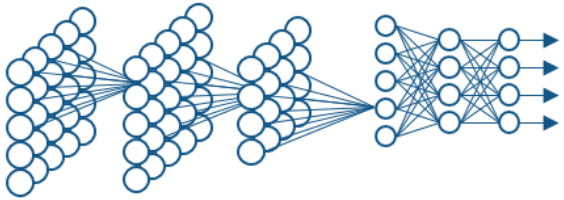
ラベルは自由に設定できます

R-CNN の学習（関数：trainRCNNObjectDetector）

```
detector = trainRCNNObjectDetector(groundTruth, network, options)
```

1 imageFilename	2 stopSign	3 carRear
'stopSignImages/image001.jpg'	[856,318,39,41]	[398,378,315,210]
'stopSignImages/image002.jpg'	[445,523,52,54]	[332,633,691,287]
'stopSignImages/image003.jpg'	[897,365,49,48]	[718,409,74,66;1...
'stopSignImages/image004.jpg'	[948,424,34,44]	[757,503,143,69]
'stopSignImages/image005.jpg'	[980,393,31,56]	[]
'stopSignImages/image006.jpg'	[1.0408e+03,35...	[]

Ground Truth



SeriesNetwork または 層の配列

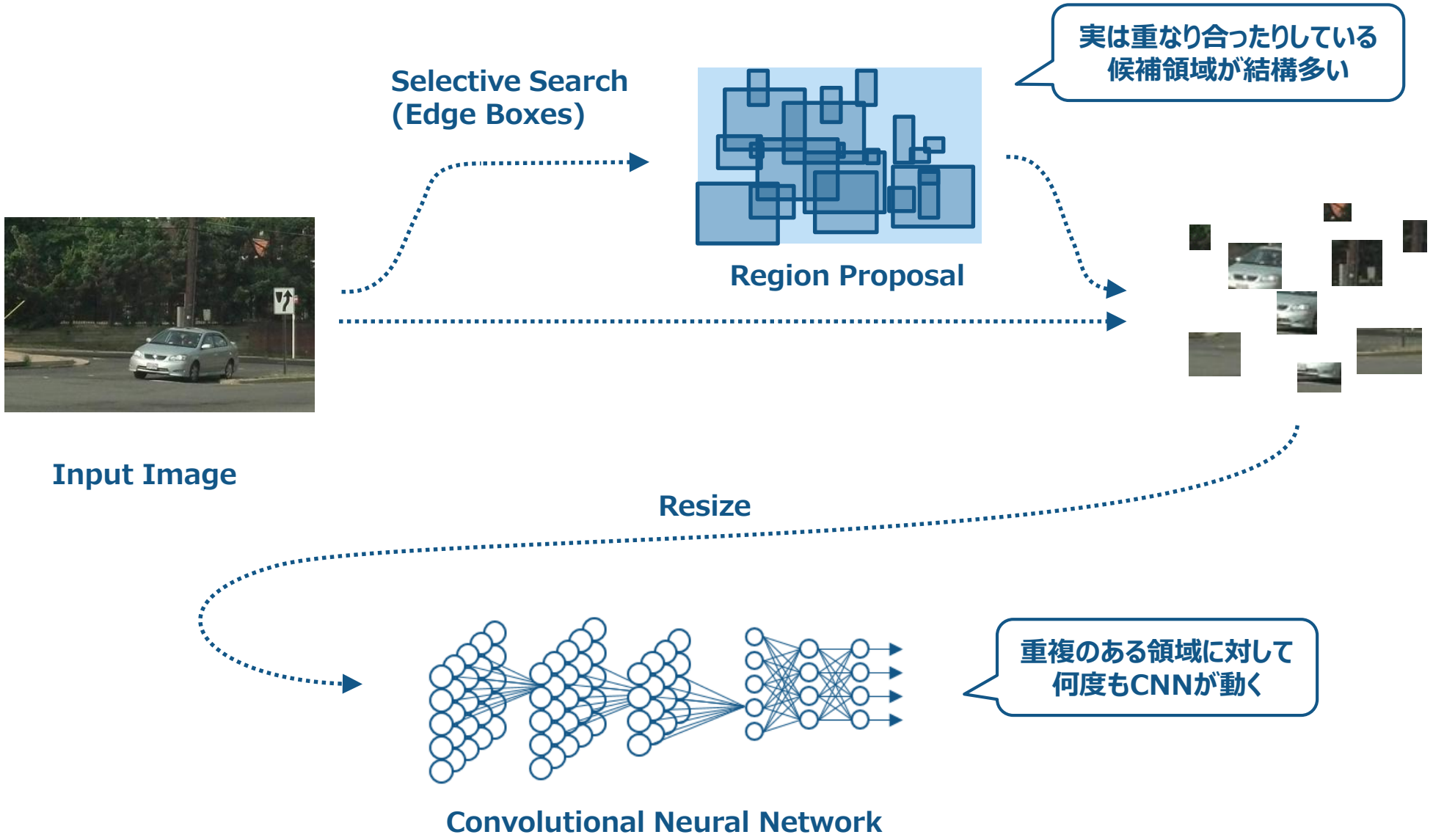
※　引数として渡すネットワークの型により関数の動作が変わることに注意！

SeriesNetwork の場合　⇒　ネットワークは自動的に変更される（学習率の倍率等も自動に設定される）

層の配列の場合　⇒　学習率の倍率等を手動で設定したい場合などはこちらを使う

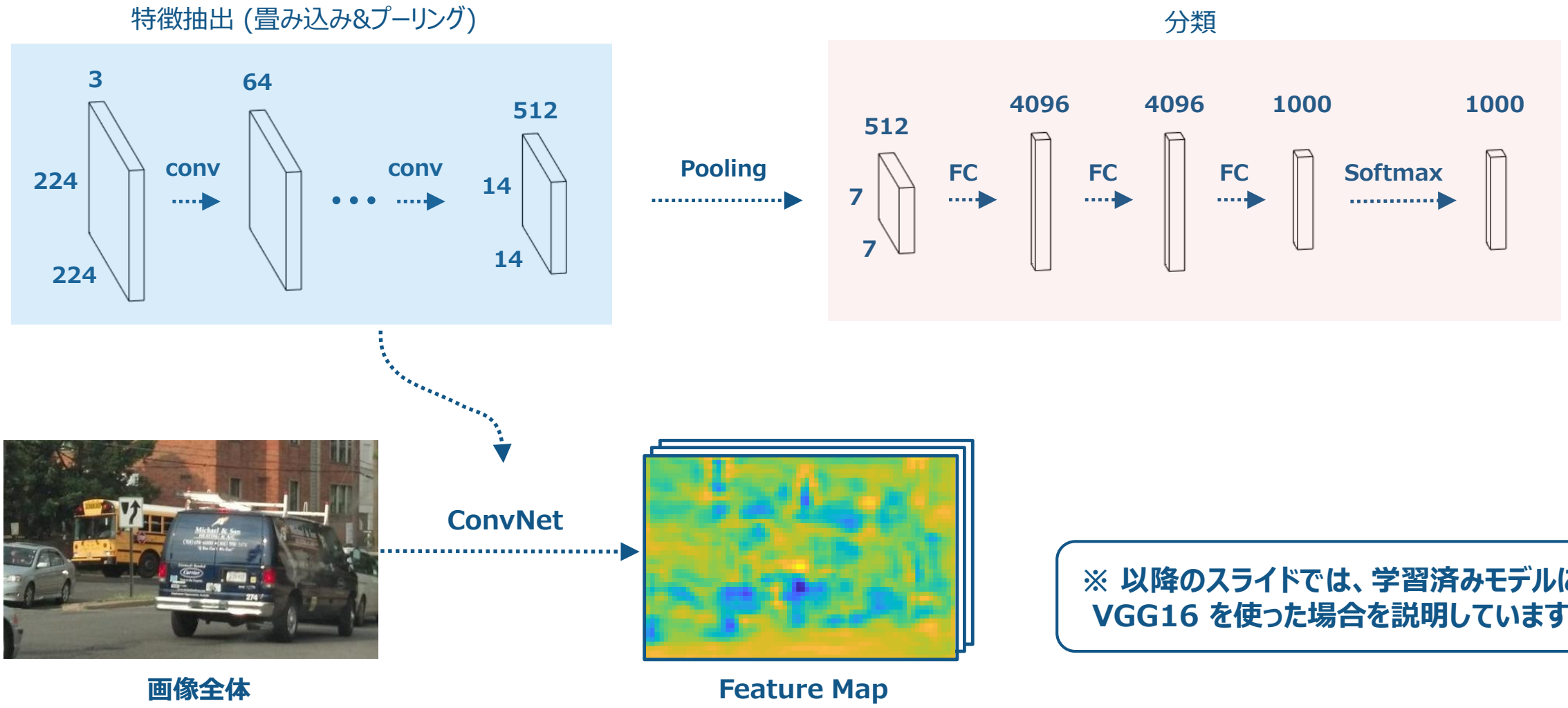
Fast R-CNN（R-CNN の高速化）

R-CNN はなぜ遅くなってしまうのか？



Fast R-CNN とは？

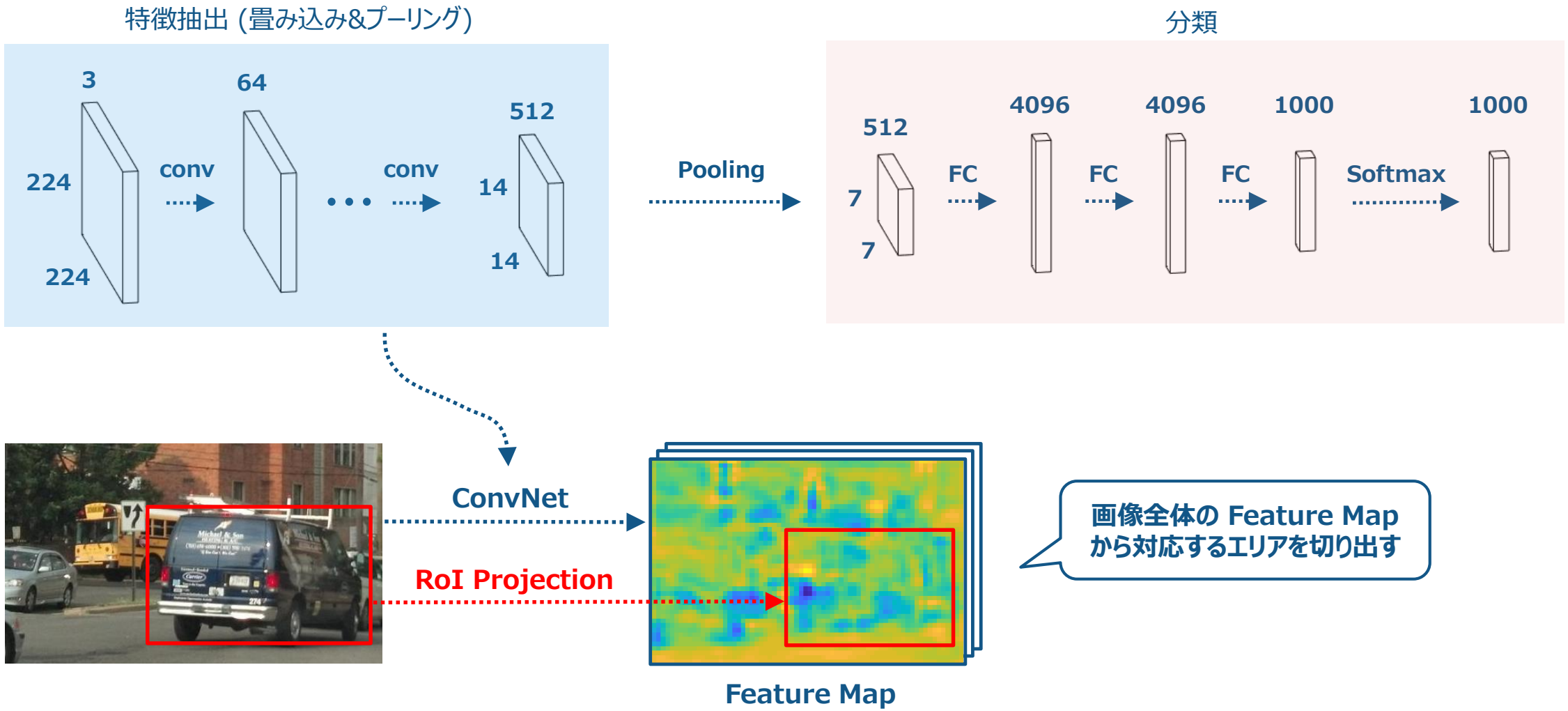
Step 1) 画像全体に CNN の前半部分（畳み込み&プーリング）を実行して、Feature Map を生成する



※ 以降のスライドでは、学習済みモデルに VGG16 を使った場合を説明しています

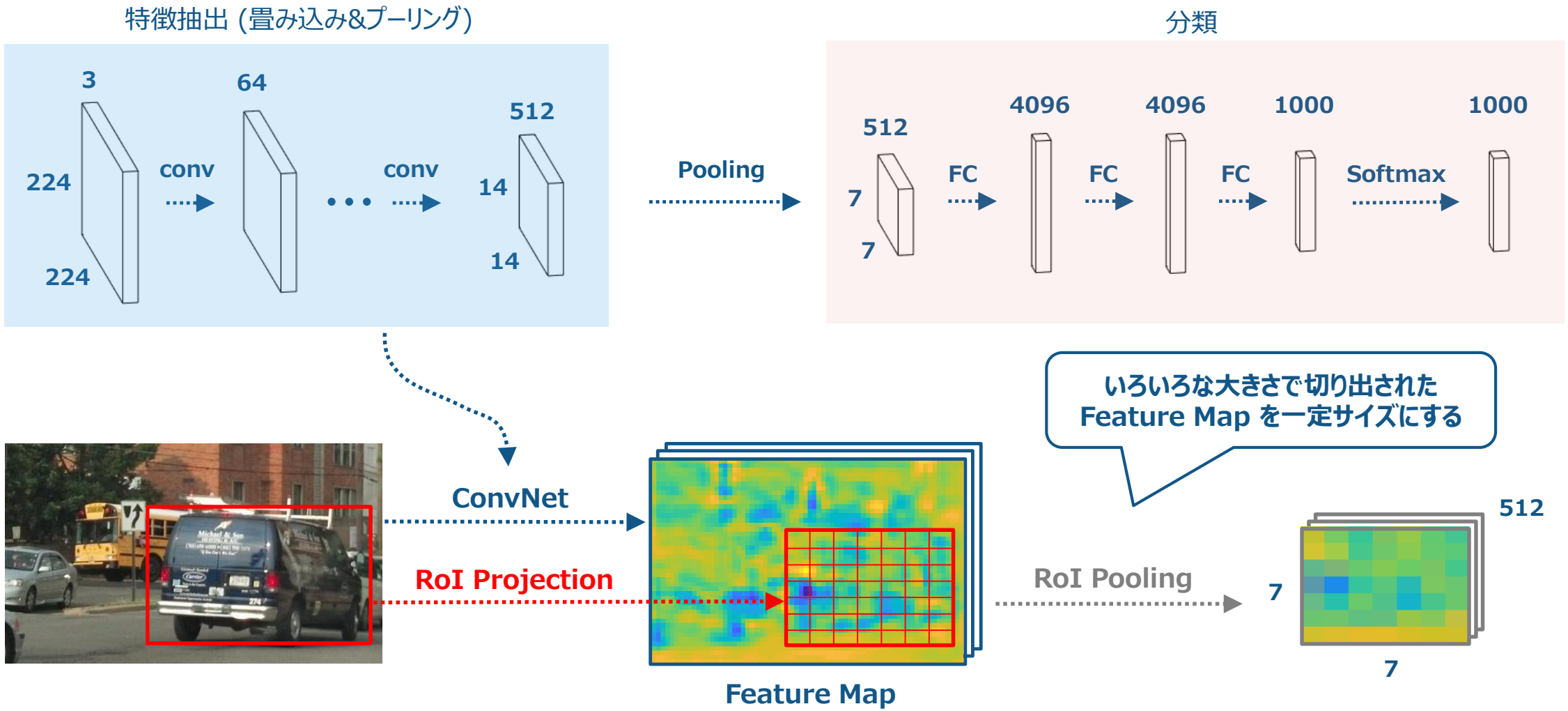
Fast R-CNN とは？

Step 2) Region Proposal に対応する部分の Feature Map を切り出す



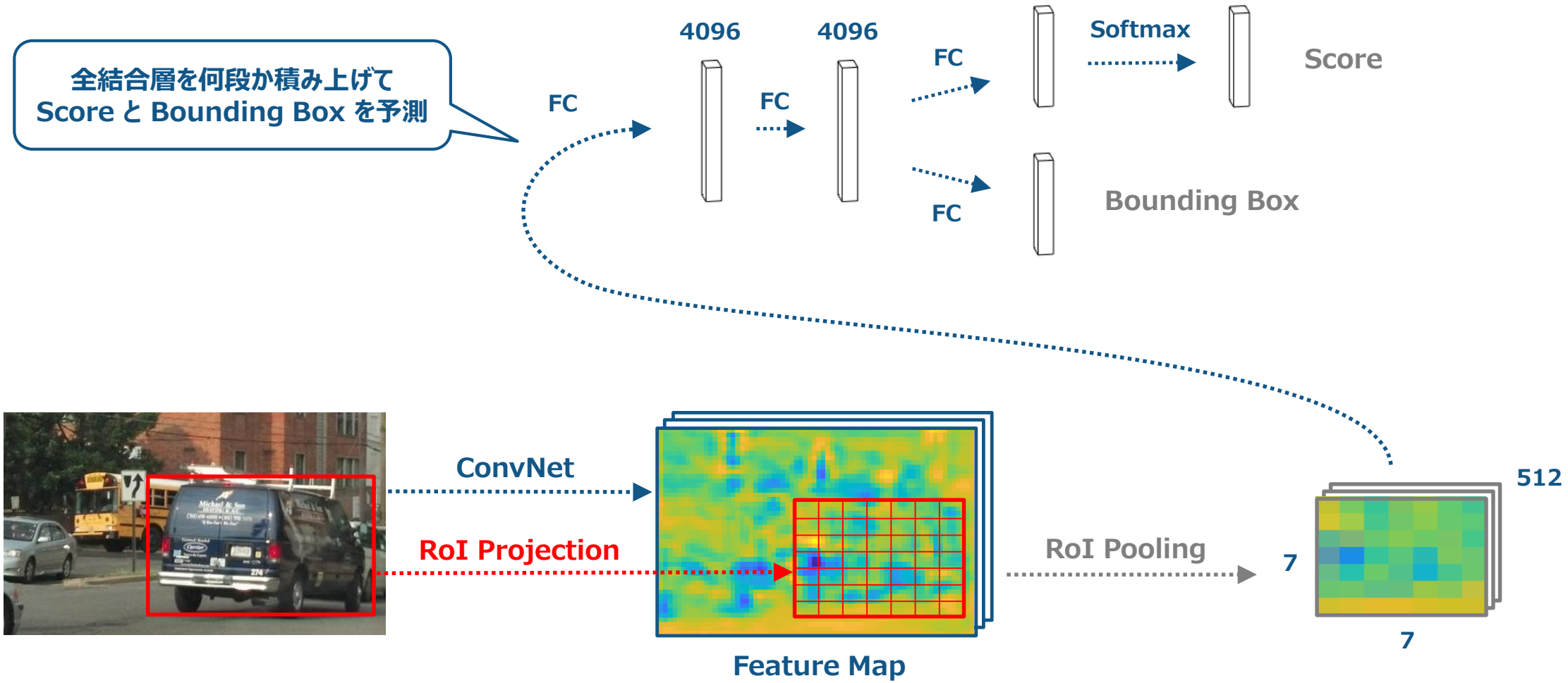
Fast R-CNN とは？

Step 3) Feature Map の対応部分を固定サイズに切り分けて、プーリングを行う（RoI Pooling）

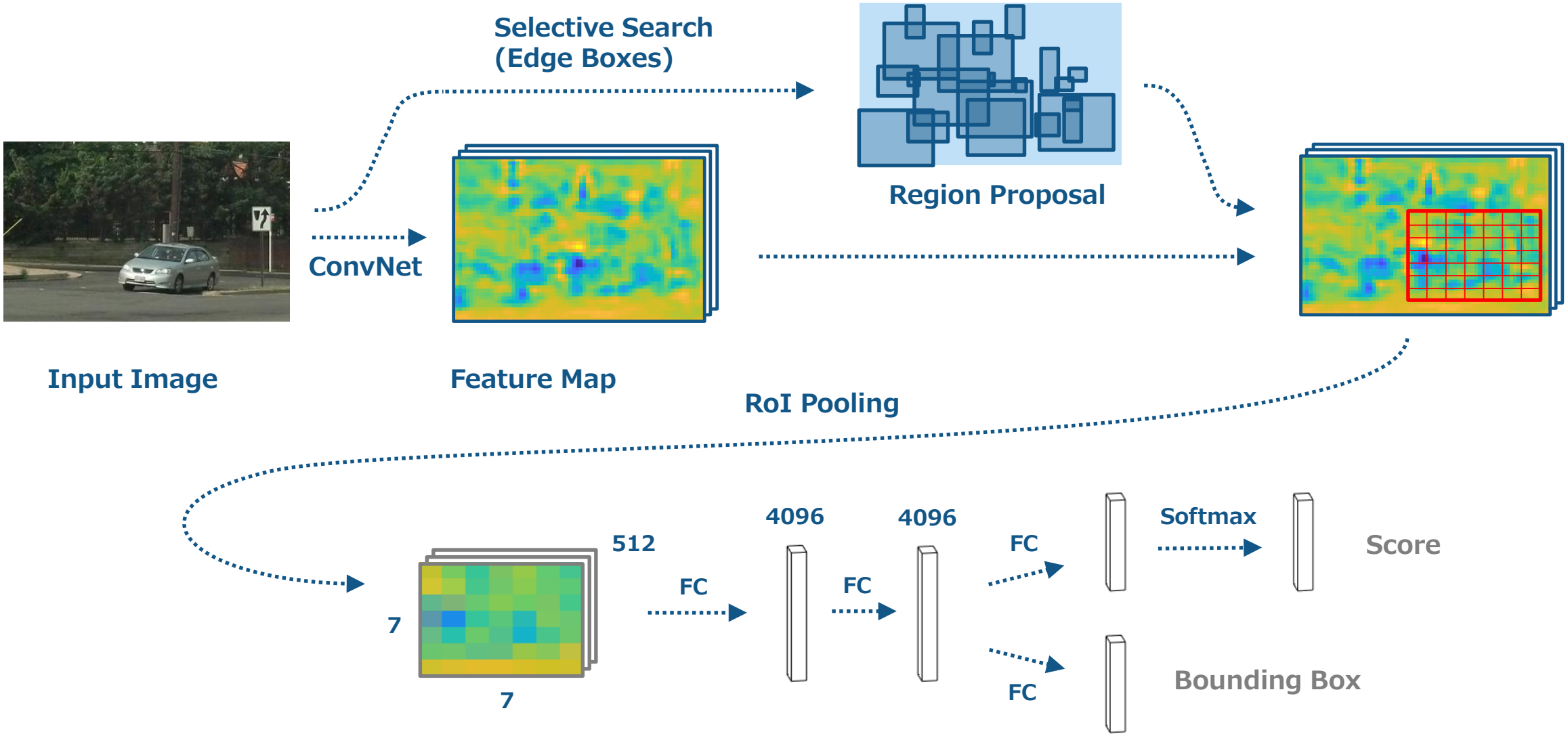


Fast R-CNN とは？

Step 4) 切り抜き & プーリングした後の Feature Map から、Score と Bounding Box を算出する



Fast R-CNN とは？


Selective Search
(Edge Boxes)
Region Proposal
ConvNet
Input Image
Feature Map
RoI Pooling
512
7
7
FC
4096
FC
4096
FC
FC
Softmax
Score
Bounding Box

Fast R-CNN の学習（関数の呼び出し例）

基本的な呼び出し方は R-CNN と大差ないが、必要に応じてメモリを節約するためのパラメータ等は設定するとよい

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate', 1e-6, ...
    'MaxEpochs', 10, ...
    'CheckpointPath', tempdir);
```

Ground Truth

```
frcnn = trainFastRCNNObjectDetector(data, layers , options, ...
    'NegativeOverlapRange', [0 0.1], ...
    'PositiveOverlapRange', [0.7 1], ...
    'SmallestImageDimension', 400);
```

学習済みモデル or 層の配列

画像の短辺のサイズ
（入力画像をリサイズする）

メモリを節約したい場合に設定すると良いパラメータ

Fast R-CNN は画像全体から Feature Map を生成するため、大量のGPUメモリを消費し易い

Faster R-CNN（Fast R-CNN の高速化）

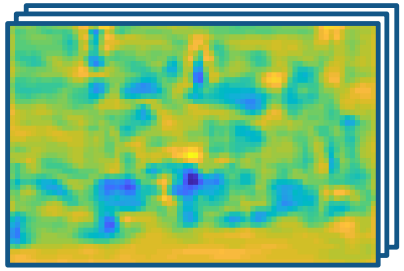
Fast R-CNN はなぜ遅くなってしまうのか？

実はRegion Proposal の生成に
時間がかかっていた！

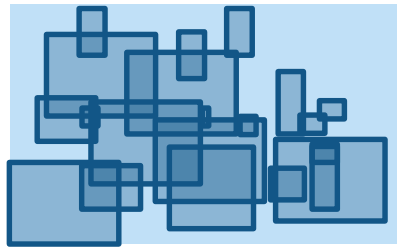
Selective Search
(Edge Boxes)

Input Image

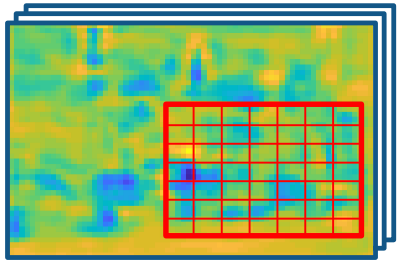
ConvNet



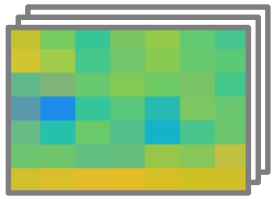
Feature Map



Region Proposal



RoI Pooling



7

7

512

FC

4096

FC

4096

FC

FC

Softmax

Score

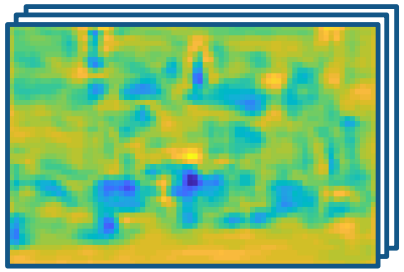
Bounding Box

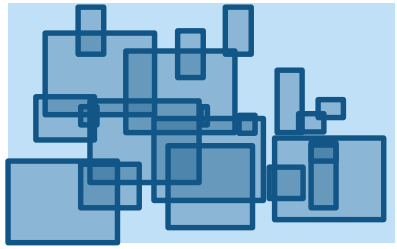
Faster R-CNN とは？

途中までの演算を共有化すること
で追加の演算コストを最小化！

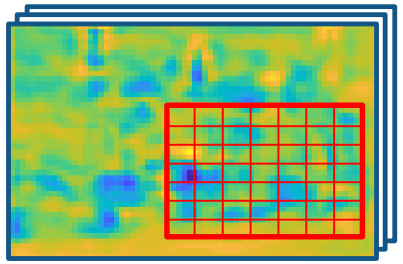
Region Proposal
Network

ConvNet





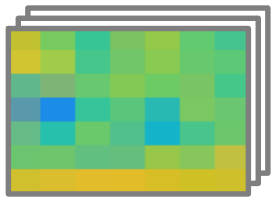
Region Proposal



Input Image

Feature Map

RoI Pooling



7

7

512

FC

4096

FC

4096

FC

FC

Softmax

Score

Bounding Box

Region Proposal Network とは？

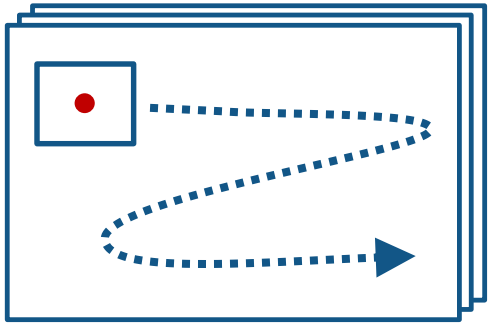
Faster R-CNN とは？

Fast R-CNN に Region Proposal Network を導入して、更に高速化したアルゴリズム

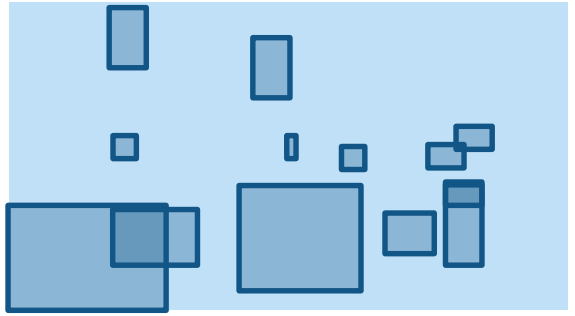
CNN の畳み込み層を使って
Feature Map を生成する

Sliding Window で切り出した
Feature Map から領域候補を生成

ConvNet



Regression



Input Image

Feature Map

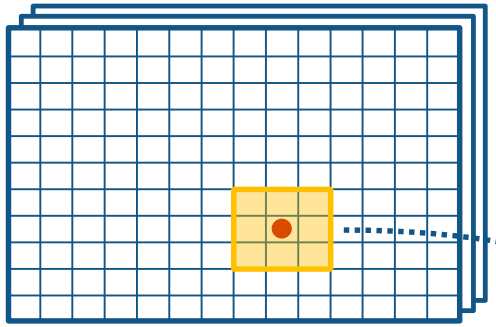
Region Proposal

Region Proposal Network とは？ → Feature Map を使って効率よく領域候補を生成する仕組み

Region Proposal Network とは？

Input Image

ConvNet



Feature Map

例えば、VGG16 の場合は 512ch の Feature Map を利用するため 3 x 3 x 512 次元の情報が使える

FC

256

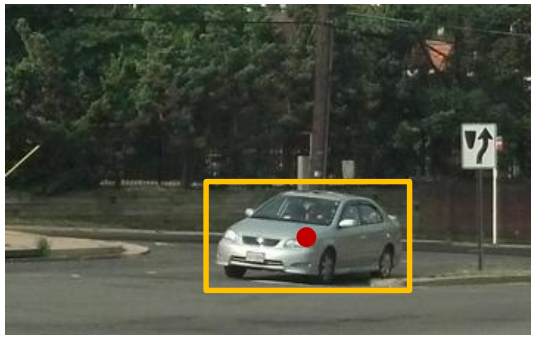
中間層

Bounding Box

(t_x, t_y, t_w, t_h)

確率値（物体・非物体）

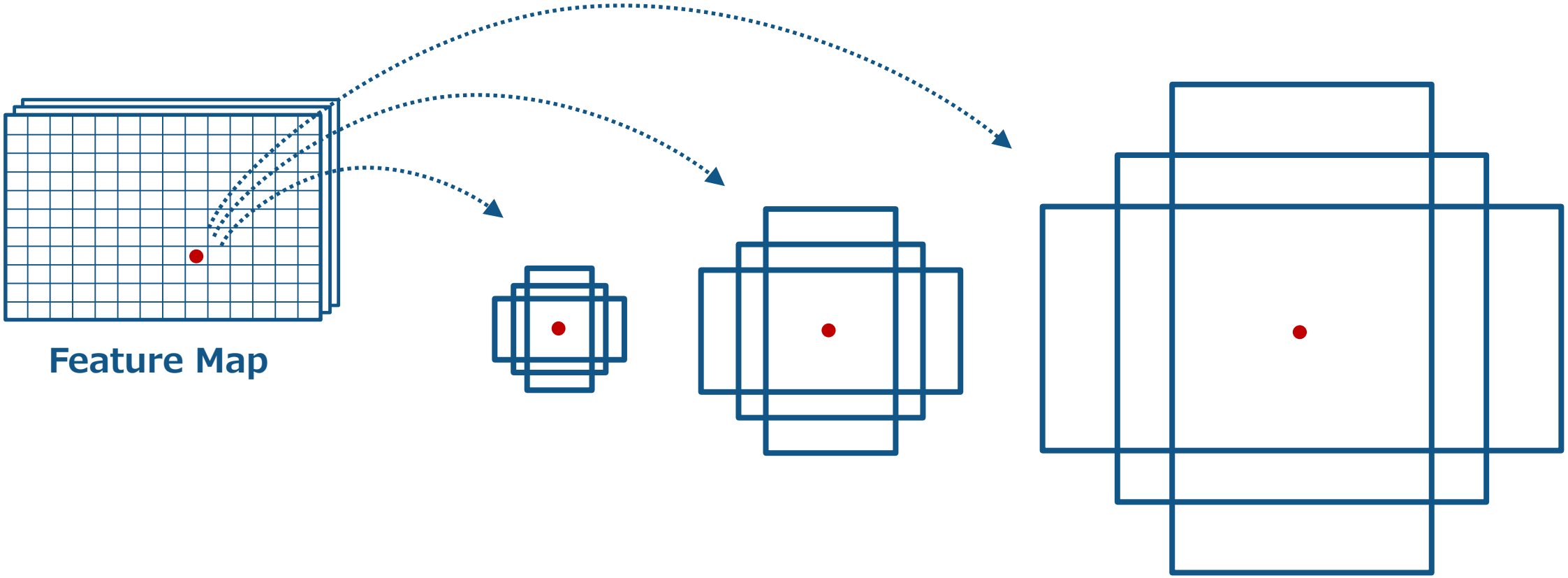
(p_{pos}, p_{neg})



領域候補（Region Proposal）

Region Proposal Network とは？

論文では Feature Map の各点に対して、９種類のサイズの Anchor Box を定義している

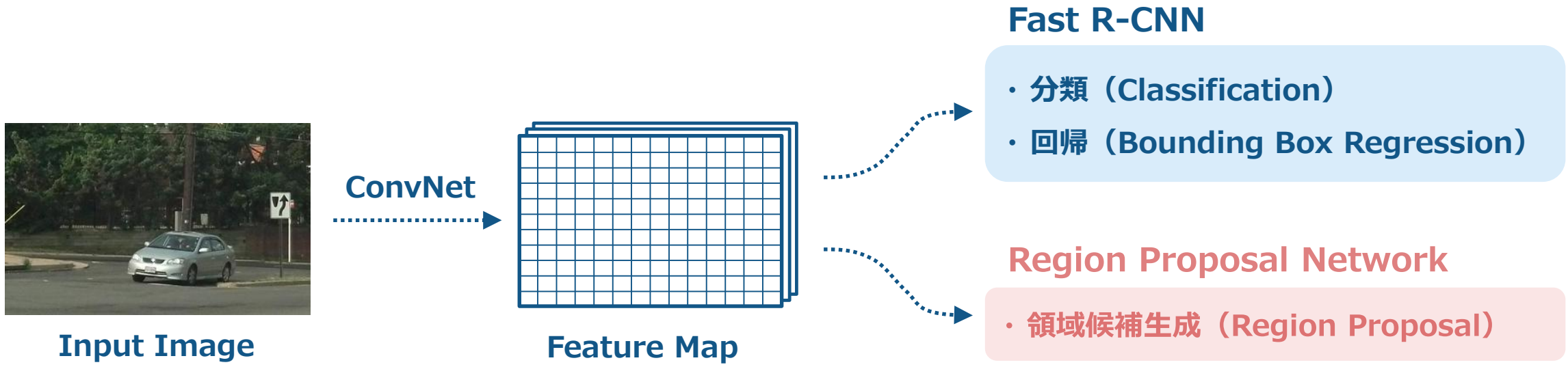


Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." Advances in Neural Information Processing Systems . Vol. 28, 2015.

Faster R-CNN の学習

次のような４ステップの手順に沿って、畳み込み部分を共有化したネットワークを学習させる

1．Region Proposal Network (RPN) を構築する
2．構築した RPN を使って Fast R-CNN を学習させる
3．Fast R-CNN に RPN を接続して、接続した部分のウェイトの Fine Tune を行う
4．最後に Fast R-CNN の FC部分のウェイトの Fine Tune を行う



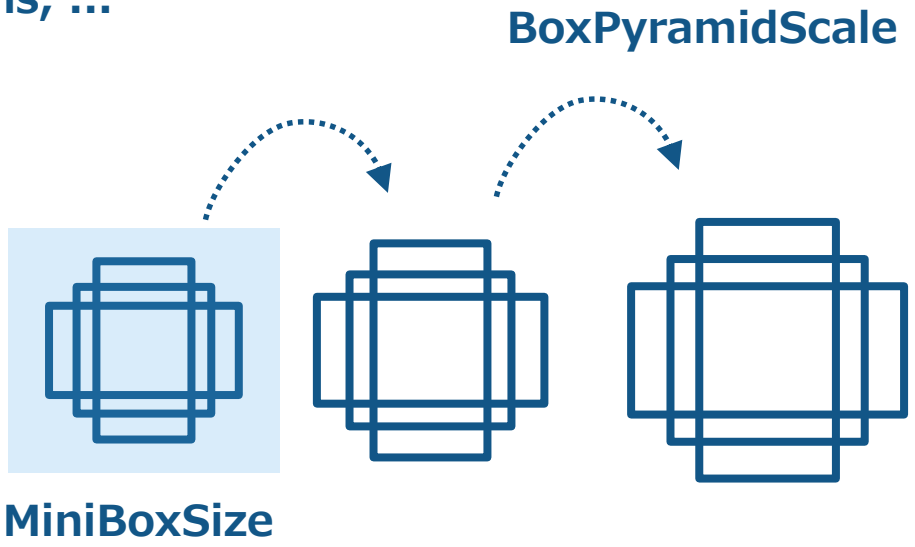
Faster R-CNN の学習（関数の呼び出し例）

学習のステップ毎に学習率等を変えたい場合は、次のような形で設定することができる

```
options(1) = trainingOptions('sgdm', 'MaxEpochs', 10, 'InitialLearnRate', 1e-5);      % Step 1
options(2) = trainingOptions('sgdm', 'MaxEpochs', 10, 'InitialLearnRate', 1e-5);      % Step 2
options(3) = trainingOptions('sgdm', 'MaxEpochs', 10, 'InitialLearnRate', 1e-6);      % Step 3
options(4) = trainingOptions('sgdm', 'MaxEpochs', 10, 'InitialLearnRate', 1e-6);      % Step 4
```

Anchor Box を自動で設定しない場合は、次のような形で設定することができる

```
detector = trainFasterRCNNObjectDetector(data, layers, options, ...
    'MiniBoxSizes', [90 180; 128 128; 180 90] , …
    'BoxPyramidScale', 1.2, …
    'NumBoxPyramidLevel, 3);
```



R-CNN / Fast R-CNN / Faster R-CNN の選び方

	R-CNN	Fast R-CNN	Faster R-CNN
認識・検出の速度	×（遅い）	△（割と速い）	○（速い）
必要なGPUメモリ	○（少なめ）	×（多い）	×（多い）
小さな物体の認識	○（得意）	×（不得意）	×（不得意）
カスタムの領域候補	○（可）	○（可）	×（不可）
学習のさせ易さ	○（簡単）	○（簡単）	×（難しい）
学習に必要な時間	○（短め）	○（短め）	×（長い）

画像全体の Feature Map を生成するため大量の GPU メモリを消費しやすい

４回の学習が必要であり、時間がかかる
うまく収束させるにはコツがいる

切り出した領域をリサイズする操作が入るため
小さな領域では拡大が行われる

物体識別（ピクセル毎）
Semantic Segmentation (SegNet)

Semantic Segmentation

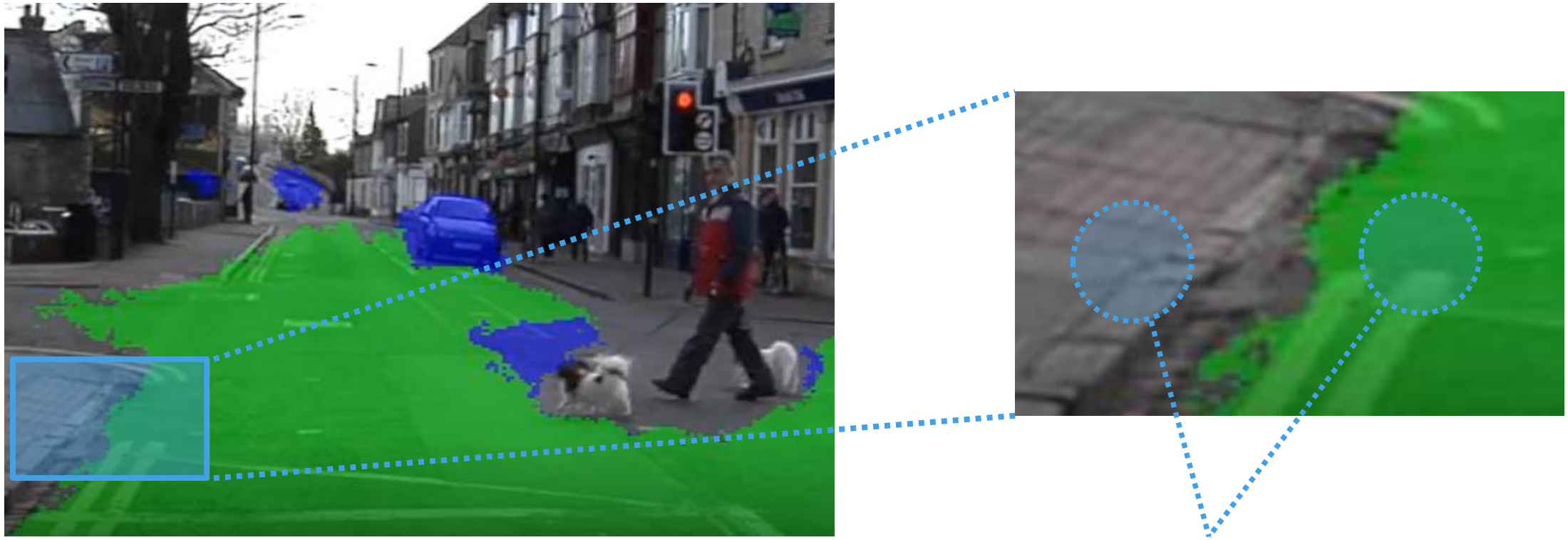
R2017b

畳み込みニューラルネットによるセグメンテーション

Semantic Segmentation とは？

R2017b

各ピクセルをその意味（周辺のピクセルの情報）に基づいて、カテゴリ分類する手法

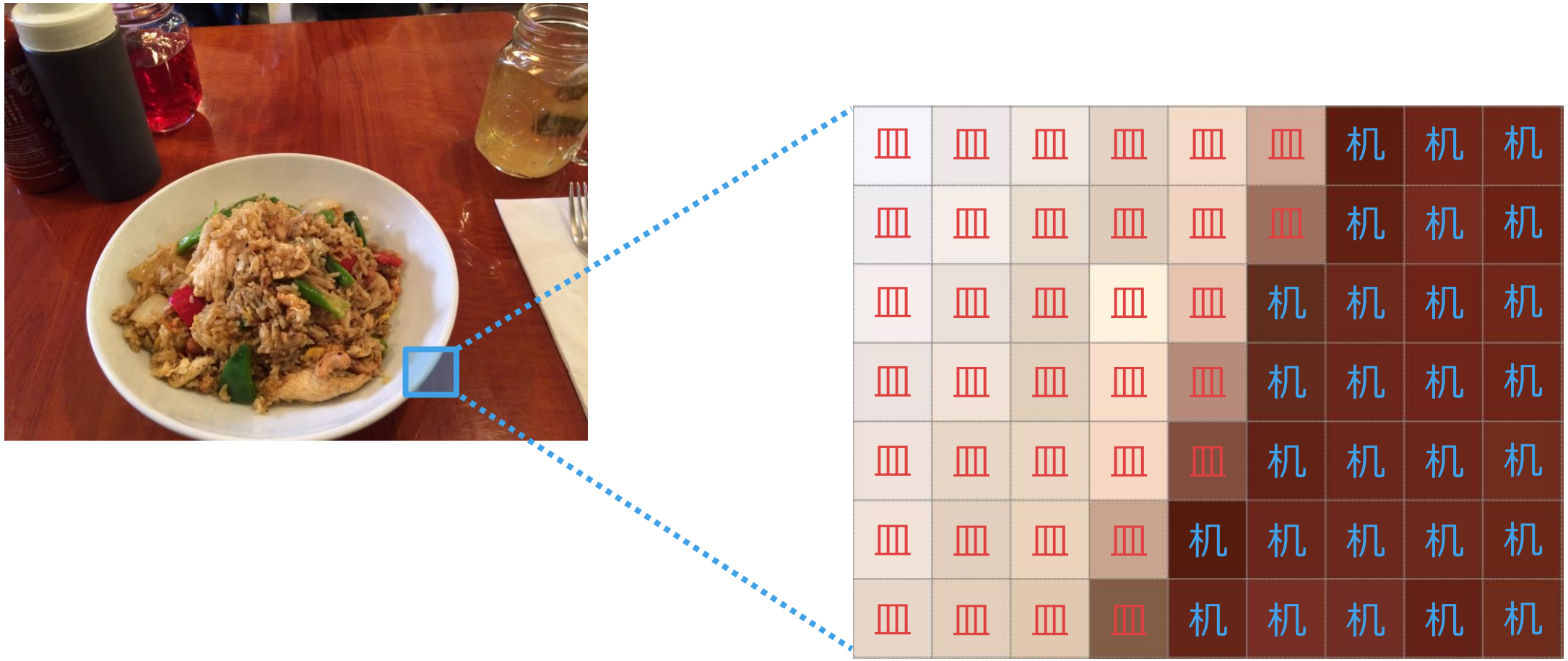


ちゃんと歩道と車道を区別できている！
色だけを見ているわけではない

Semantic Segmentation とは？

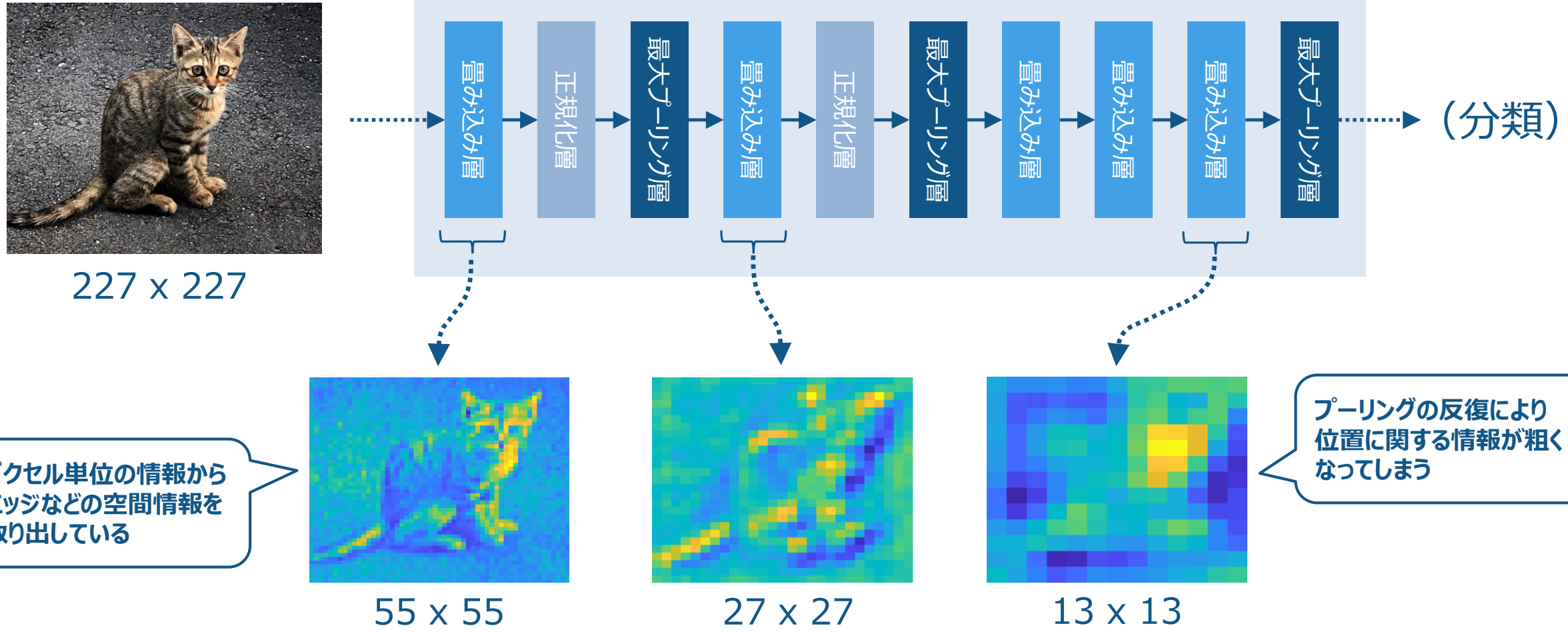
R2017b

各ピクセルをその意味（周辺のピクセルの情報）に基づいて、カテゴリ分類する手法



Feature Map とは？

畳み込みニューラルネットワークの計算過程で出てくる畳み込みの出力

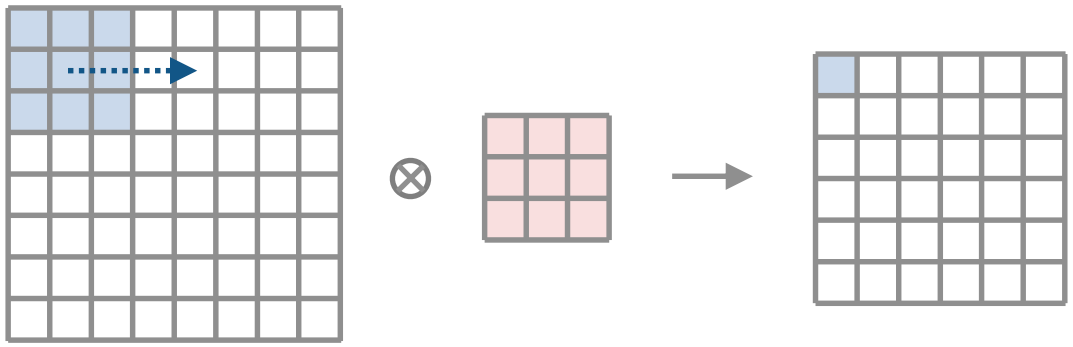


Convolution Layer（畳み込み層） / Pooling Layer（プーリング層）

Convolution Layer（畳み込み層）

→ 画像のフィルタ処理に相当する処理

→ 特徴抽出器としての役割

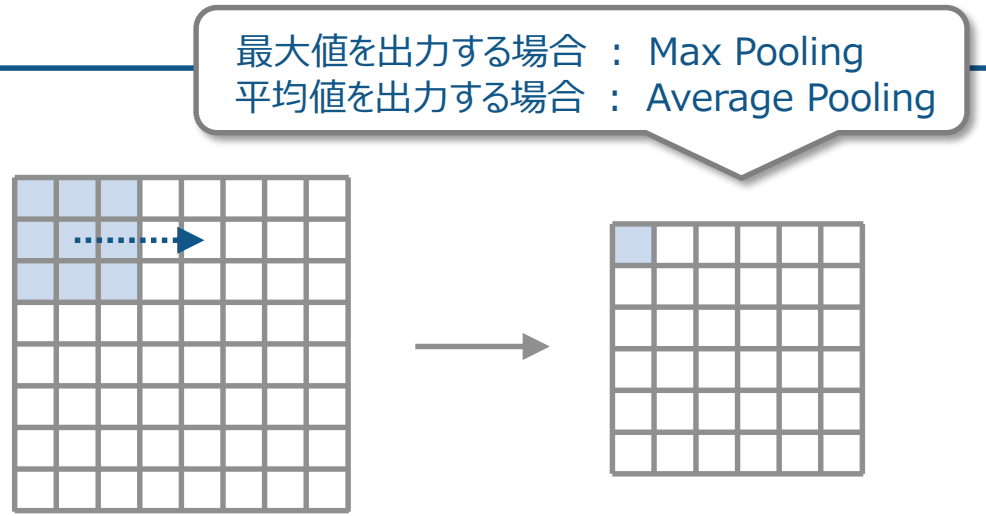


Pooling Layer（プーリング層）

→ 領域内の最大値または平均値を出力

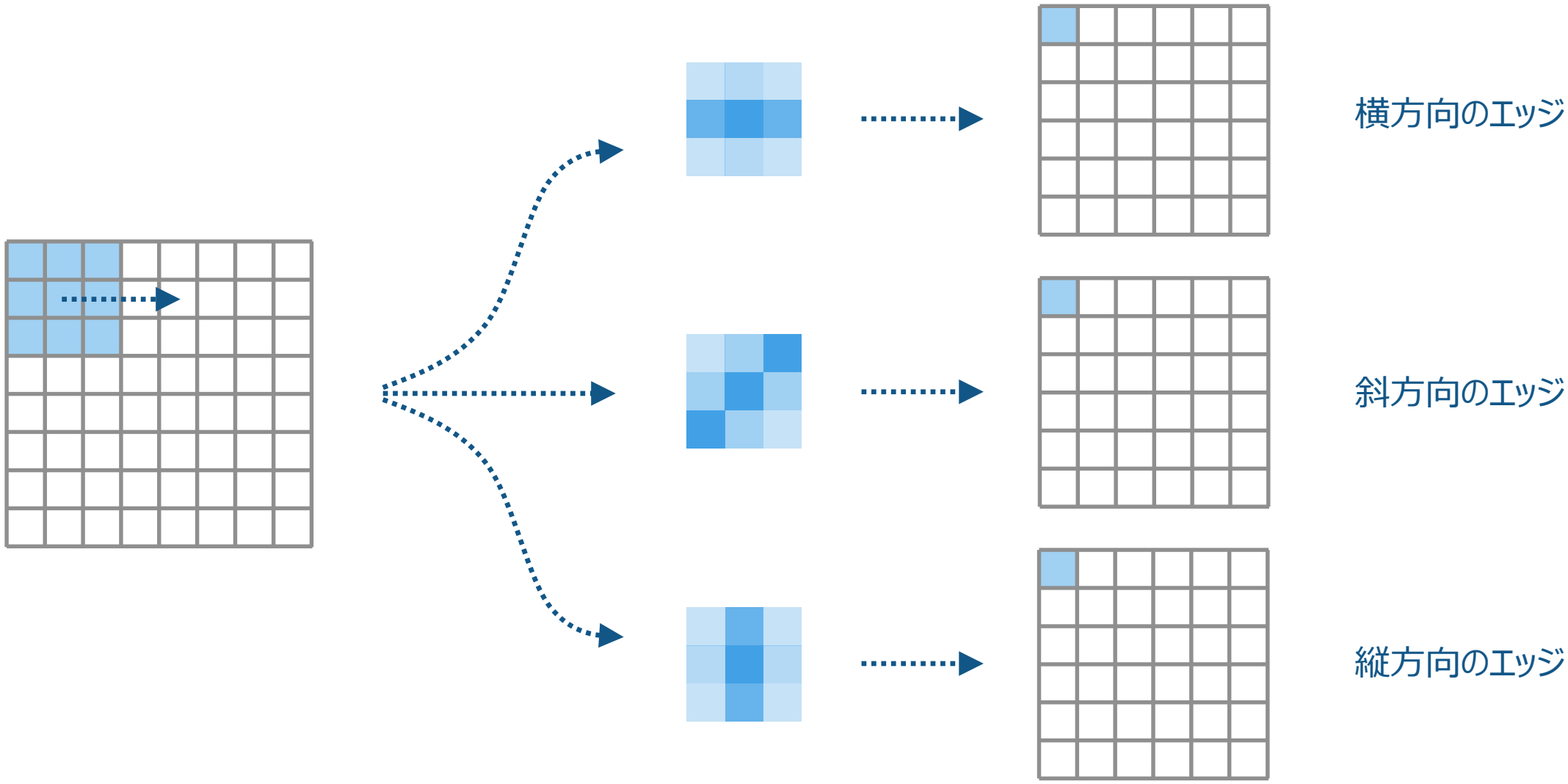
→ 平行移動等に対するロバスト性に関係

→ ストライドと呼ばれる間引きを行うこともある



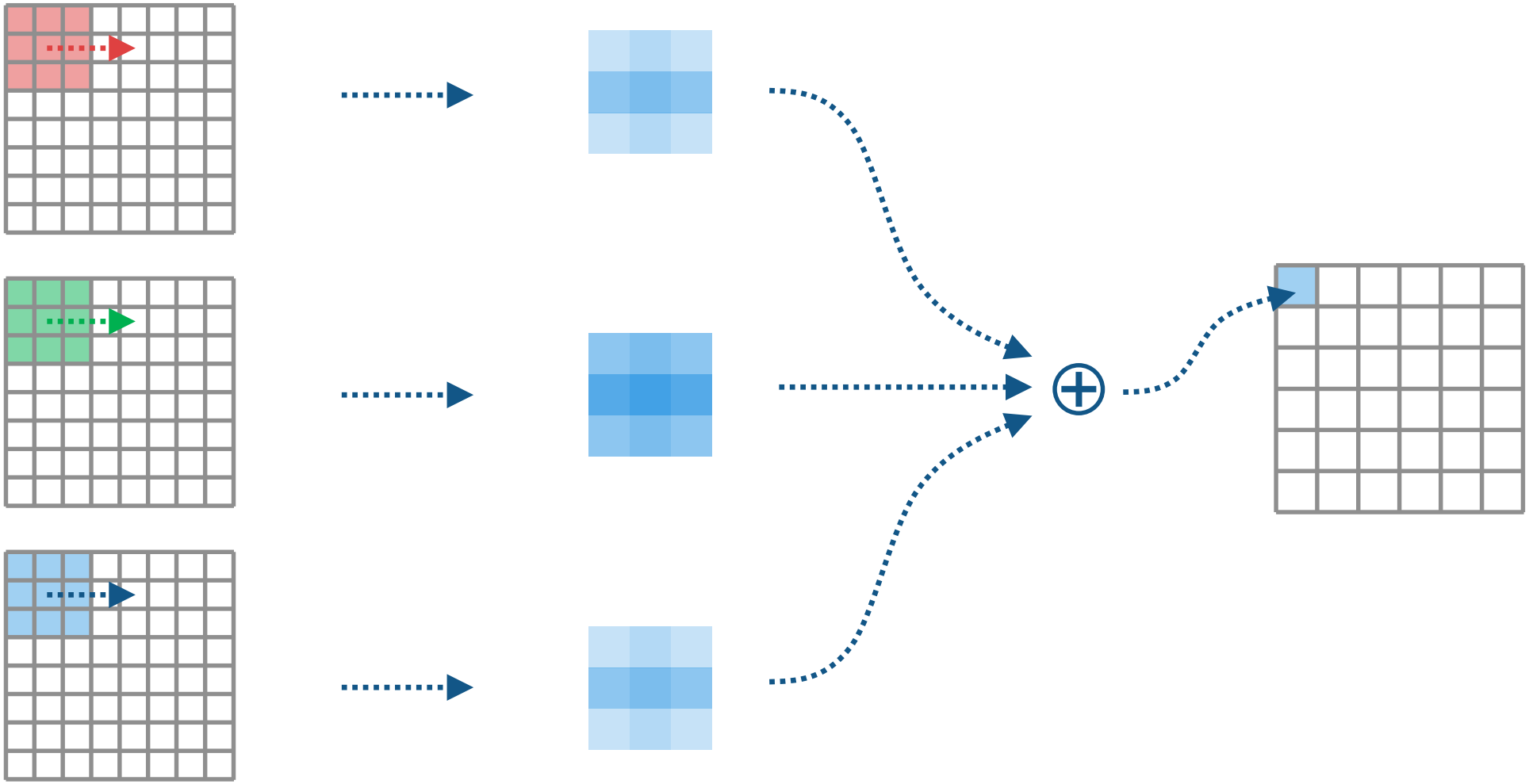
Convolution Layer（畳み込み層）

複数のフィルタにより、画像の空間方向のさまざまなパターンを抽出することができる



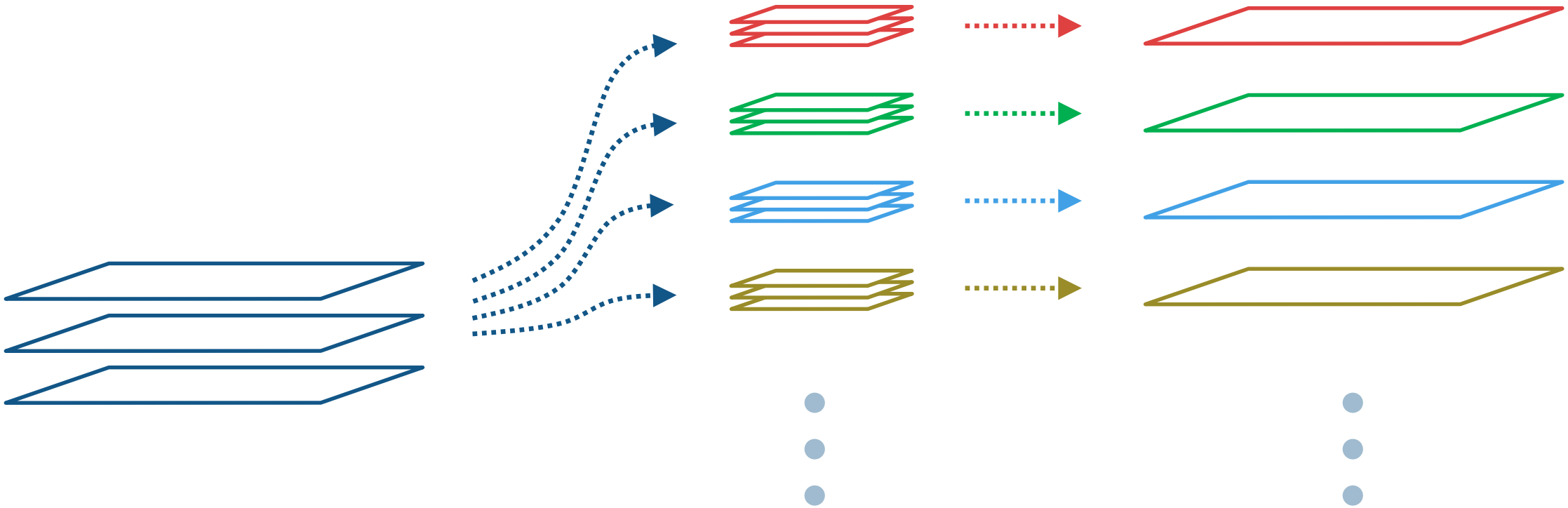
Convolution Layer（畳み込み層）

空間方向だけでなく、チャネル方向のパターンも抽出することができる



Convolution Layer（畳み込み層）

AlexNet の 1 段目の畳み込みでは 3 ch の特徴マップから 96 ch の特徴マップを生成している

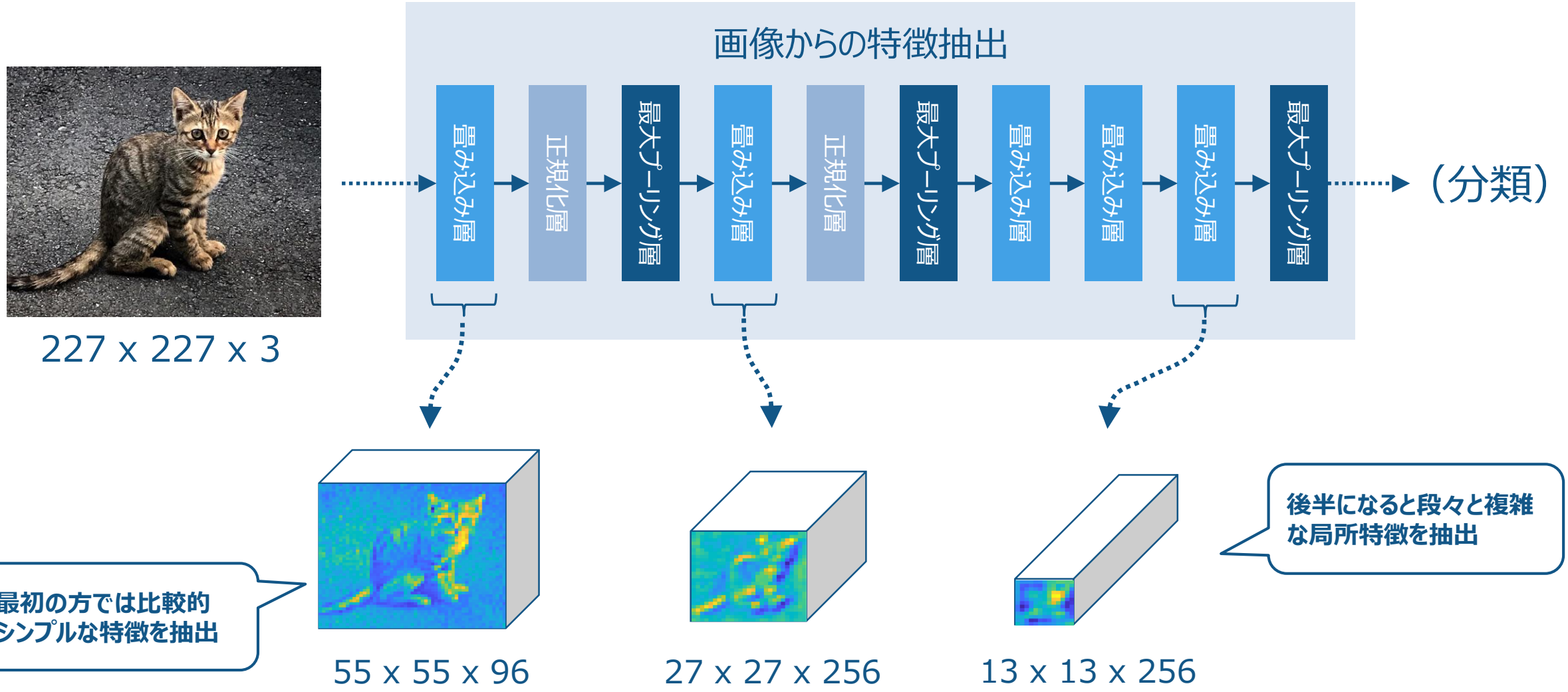


入力特徴マップ
（チャネル数: 3）

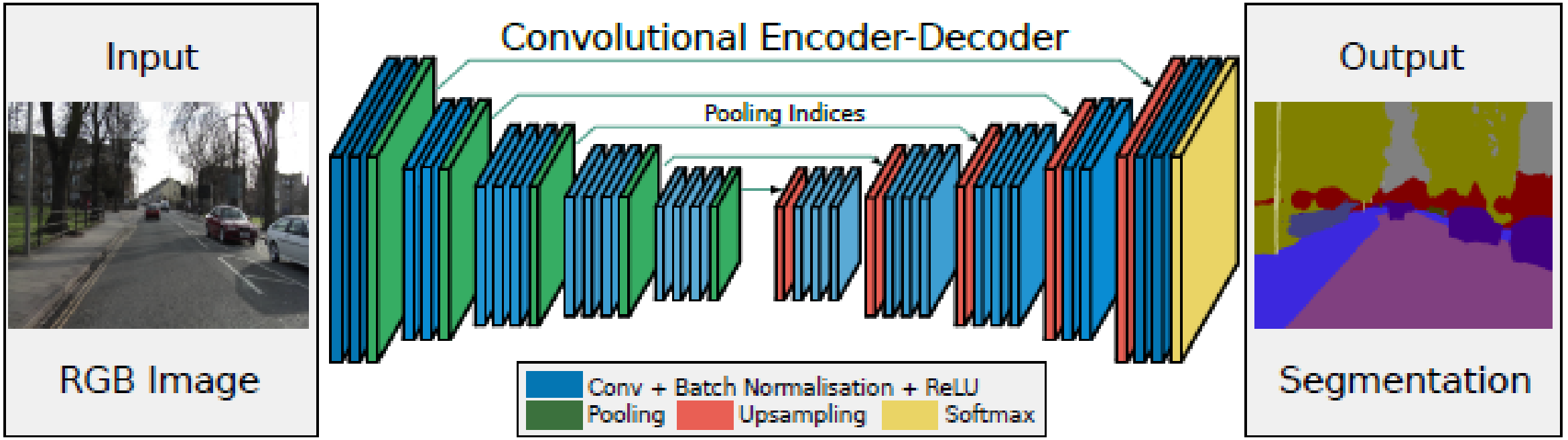
出力特徴マップ
（チャネル数: 96）

Feature Map とは？

畳み込みニューラルネットワークの計算過程で出てくる畳み込みの出力



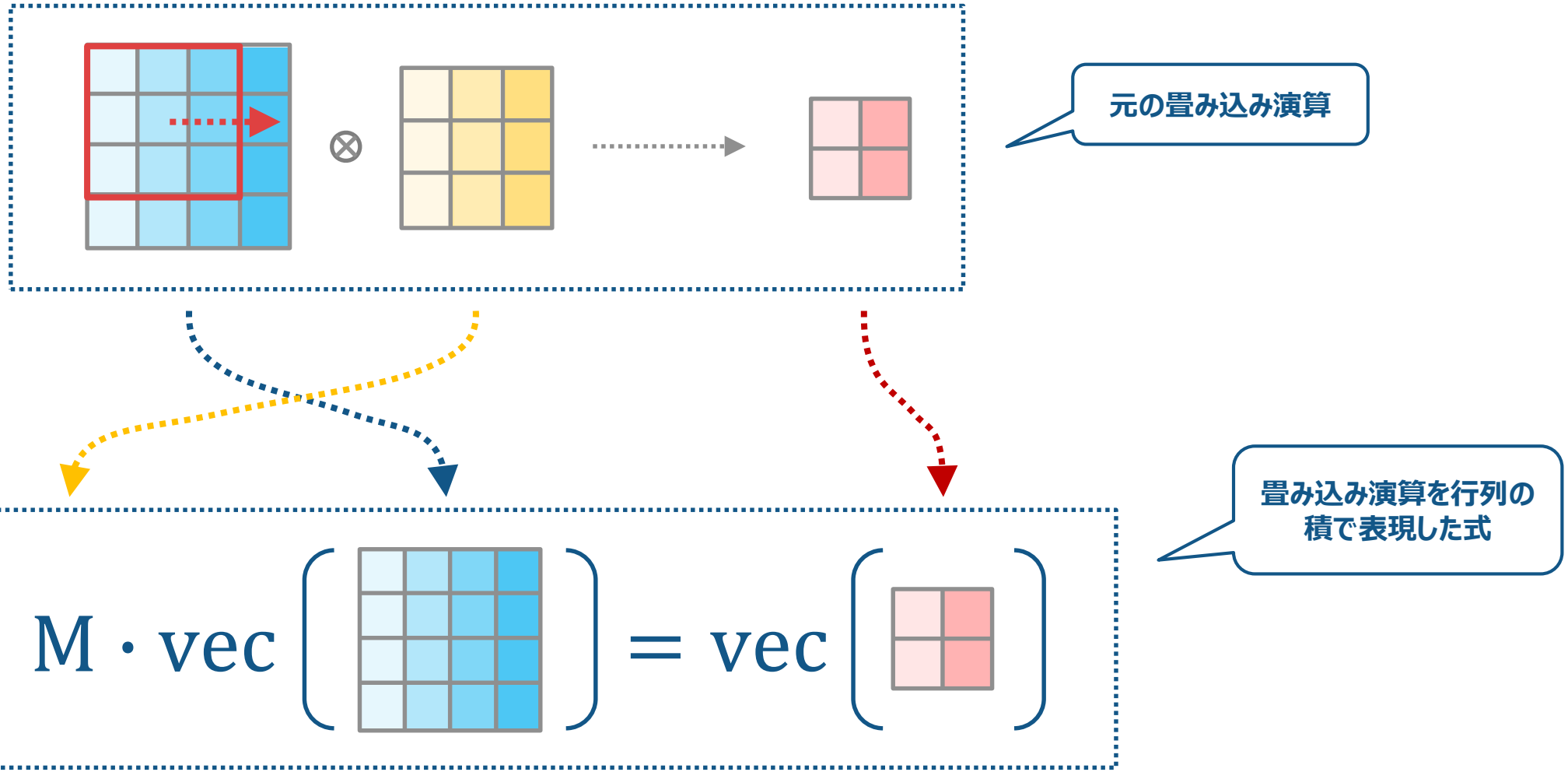
SegNet (Semantic Segmentation)



Badrinarayanan, V., A. Kendall, and R. Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." arXiv. Preprint arXiv: 1511.0051, 2015.

逆畳み込み（Deconvolution）

畳み込み演算は、Feature Map と Kernel に適当な変換を施すことで行列の積で表せる



逆畳み込み層（Deconvolution Layer）

畳み込み演算に相当する行列を転置すると、逆畳み込み演算に対応する行列となる

畳み込み：
Convolution

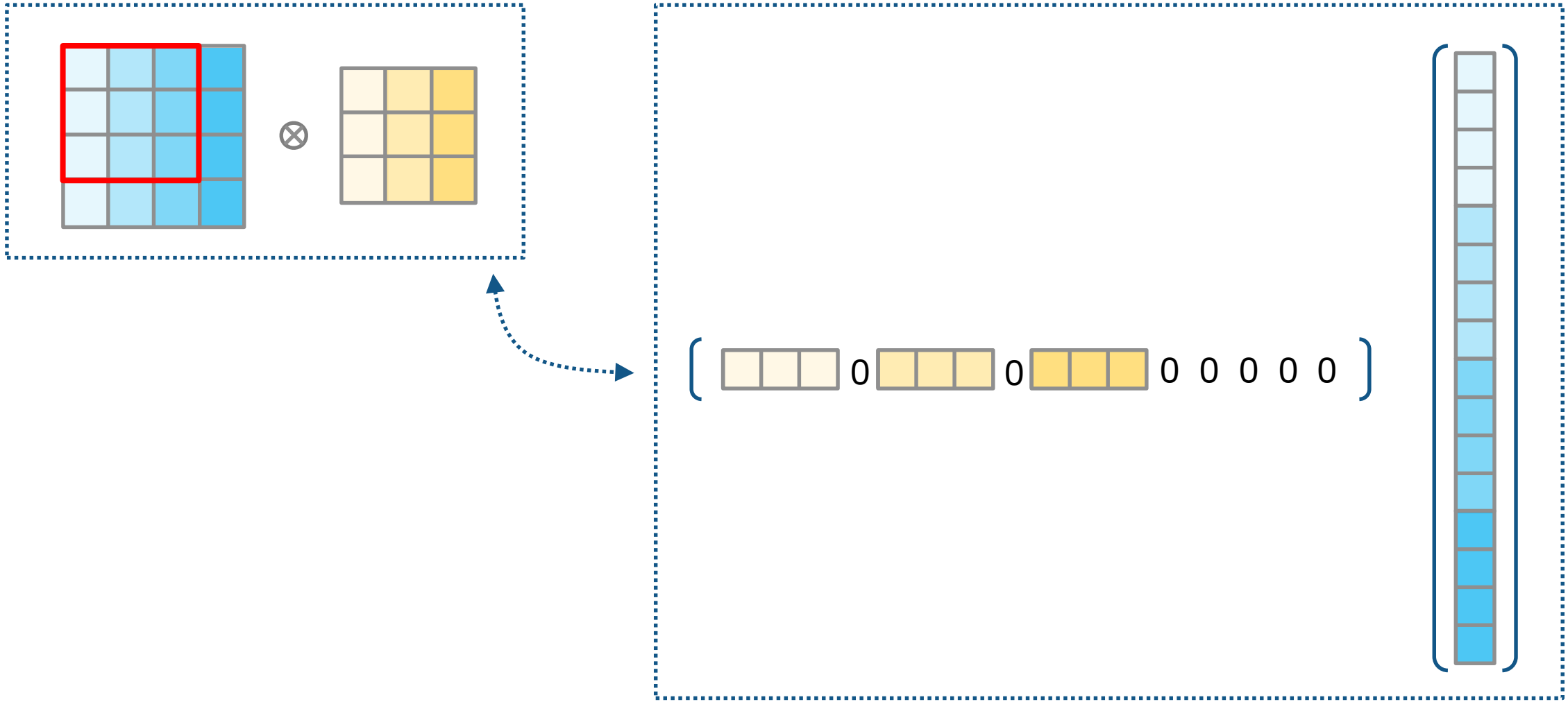
$$M \cdot \mathrm{vec}\left(\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}\right) \dashrightarrow \mathrm{vec}\left(\begin{bmatrix} & \\ & \end{bmatrix}\right)$$

逆畳み込み：
Deconvolution

$$M^{T} \cdot \mathrm{vec}\left(\begin{bmatrix} & \\ & \end{bmatrix}\right) \dashrightarrow \mathrm{vec}\left(\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}\right)$$

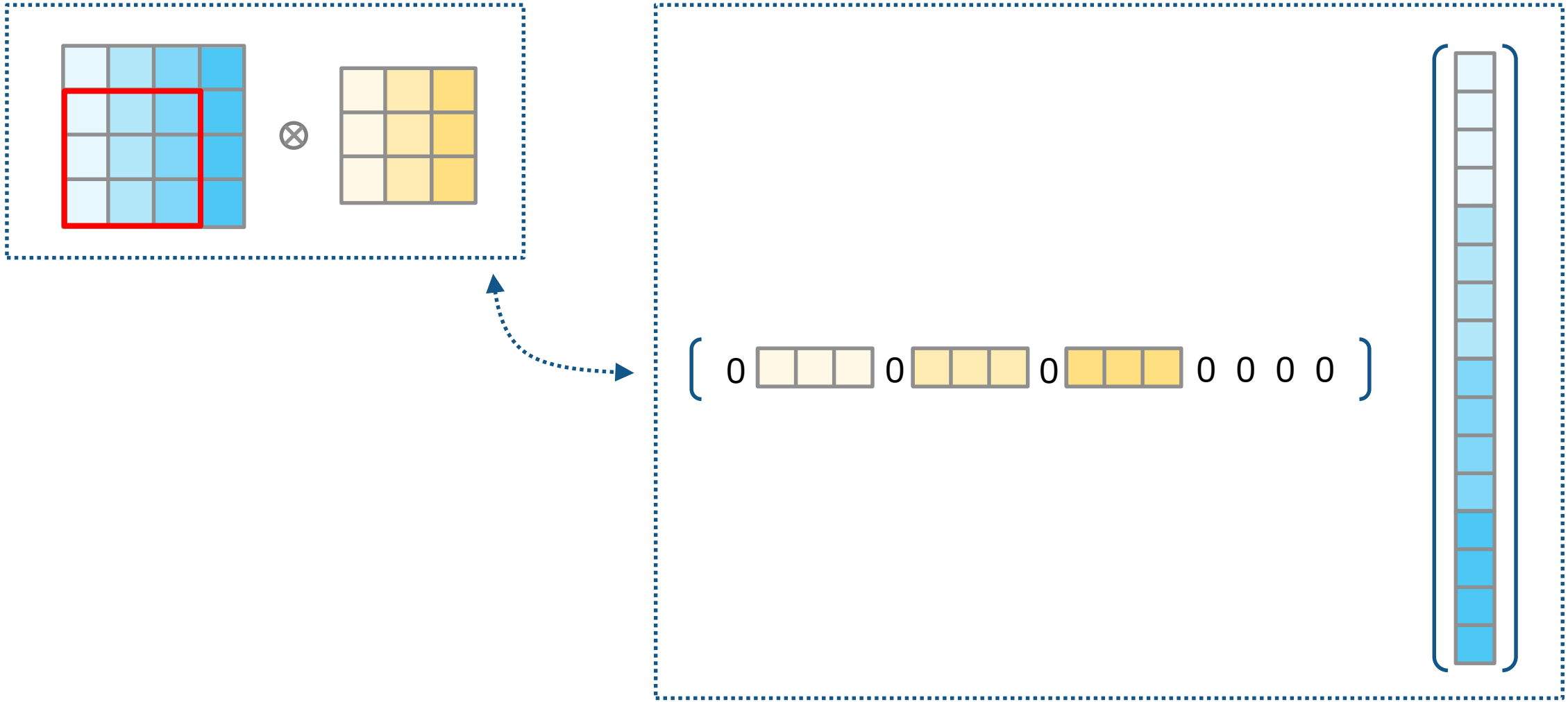
逆畳み込み（Deconvolution）

Kernel が赤枠の場所にいるときの畳み込み演算は、左図の内積演算に等しい



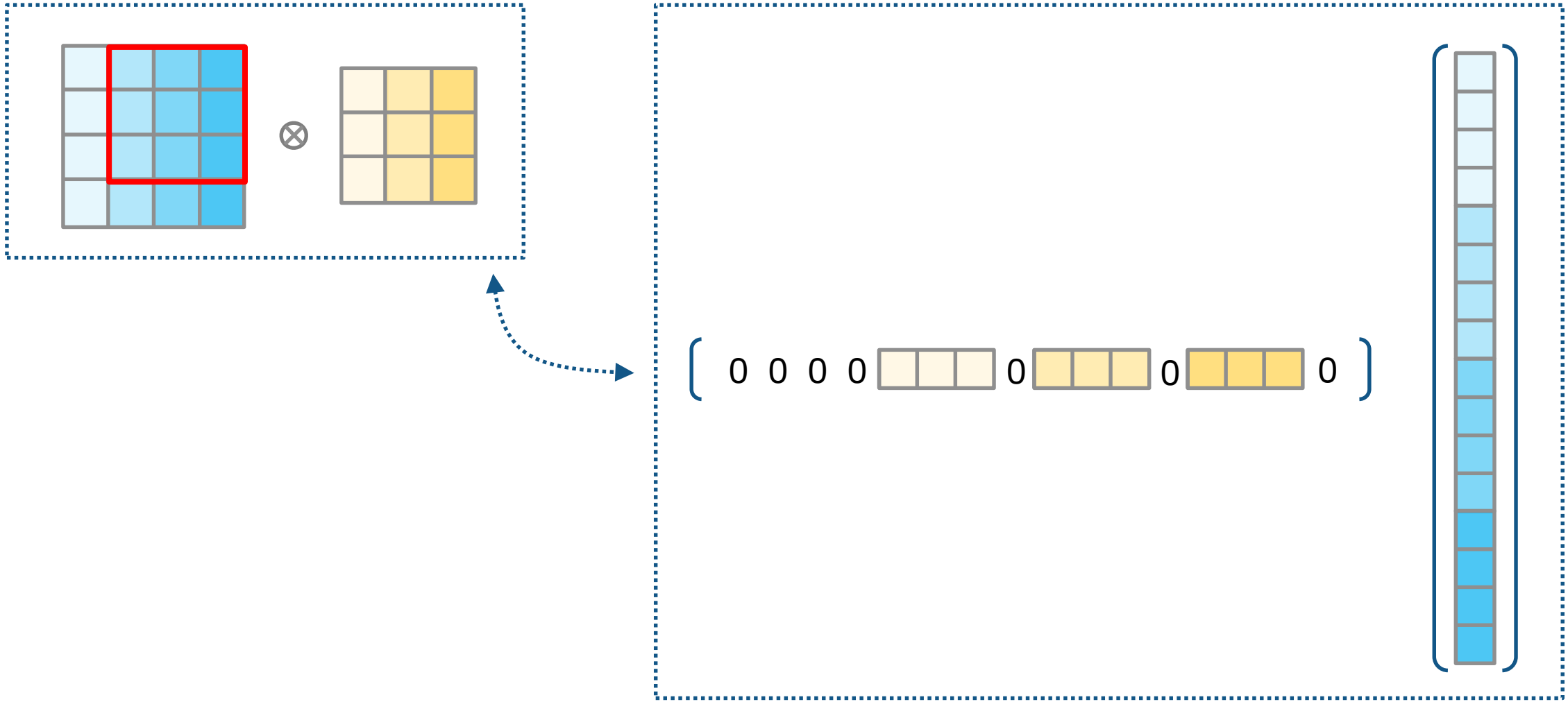
逆畳み込み（Deconvolution）

Kernel が赤枠の場所にいるときの畳み込み演算は、左図の内積演算に等しい



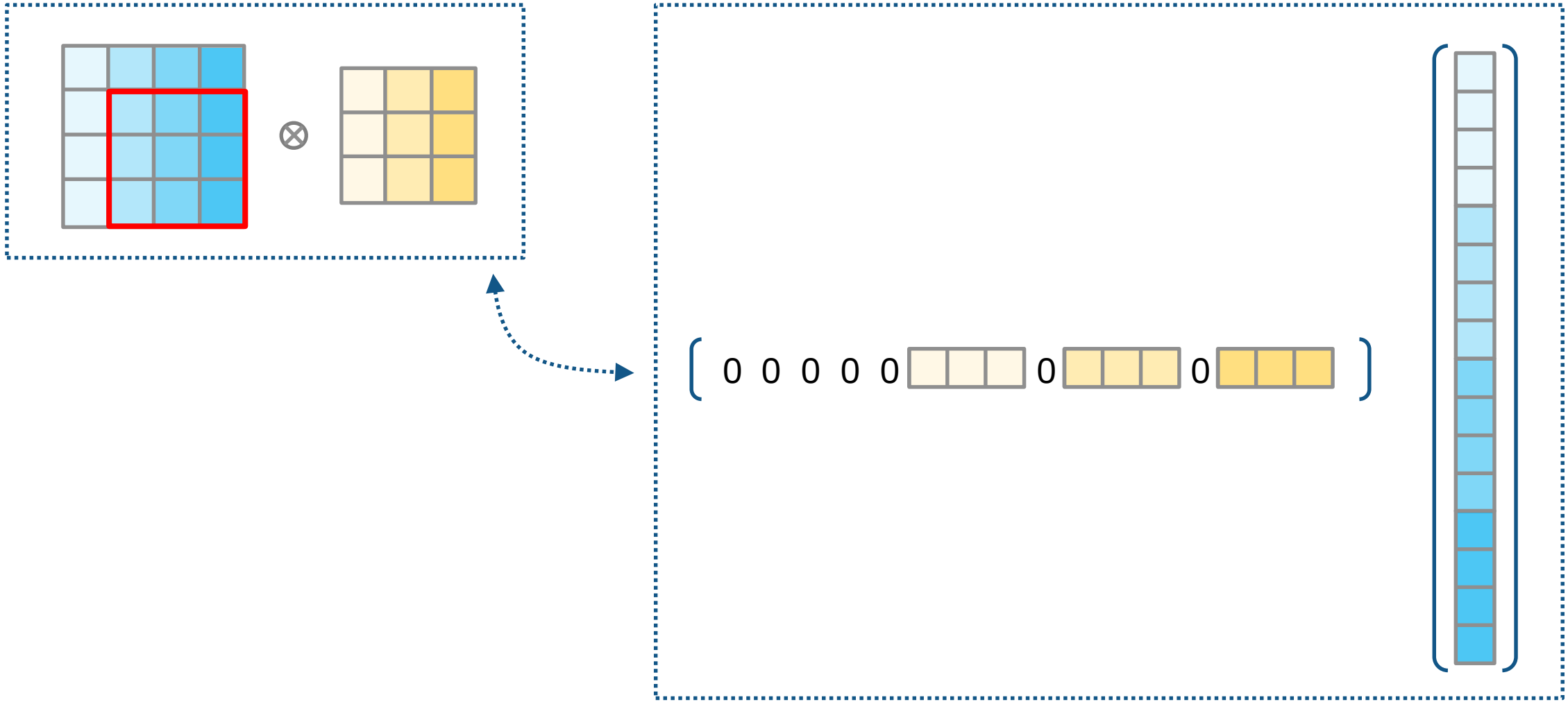
逆畳み込み（Deconvolution）

Kernel が赤枠の場所にいるときの畳み込み演算は、左図の内積演算に等しい

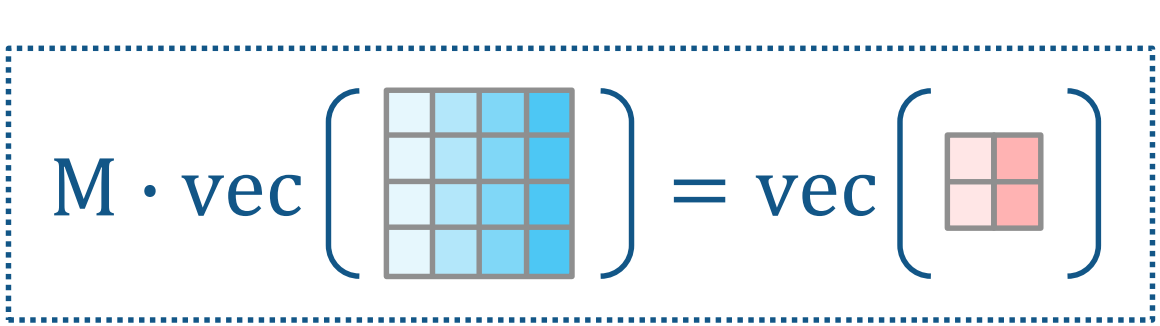


逆畳み込み（Deconvolution）

Kernel が赤枠の場所にいるときの畳み込み演算は、左図の内積演算に等しい

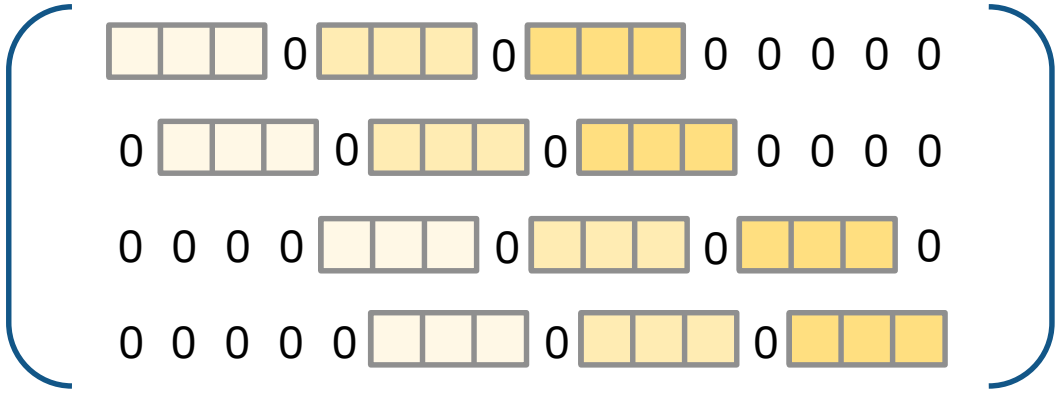


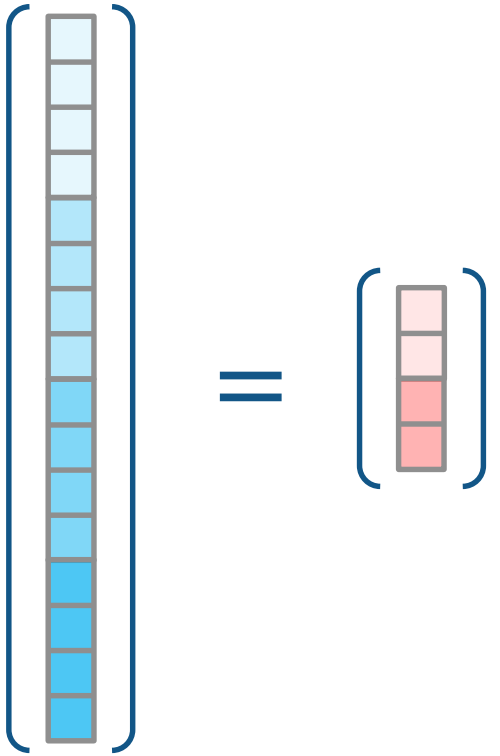
逆畳み込み（Deconvolution）



Kernel

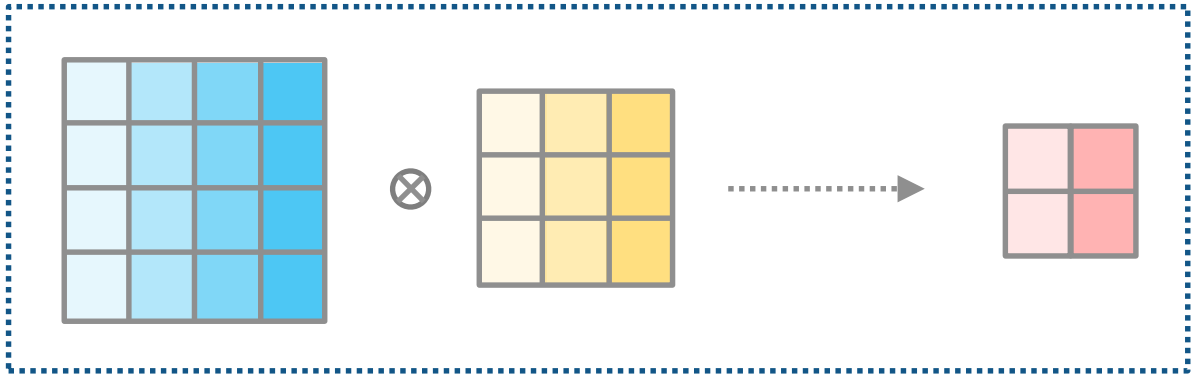
Kernel をばらして
行列 M 構築する





=

逆畳み込み（Deconvolution）



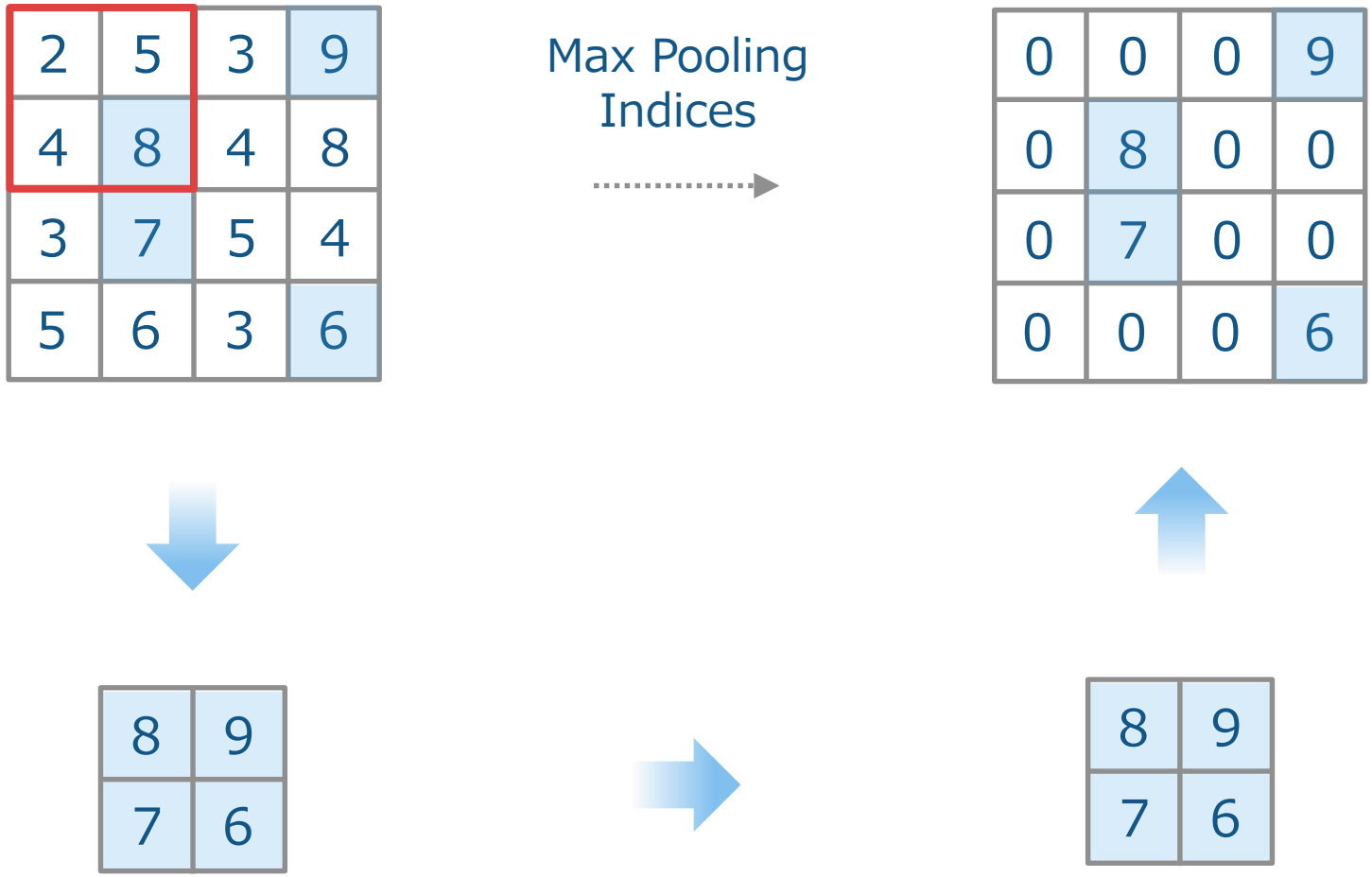
畳み込み：
convolution

$$M \cdot \mathrm{vec}\left(\quad\right) \longrightarrow \mathrm{vec}\left(\quad\right)$$

逆畳み込み：
deconvolution

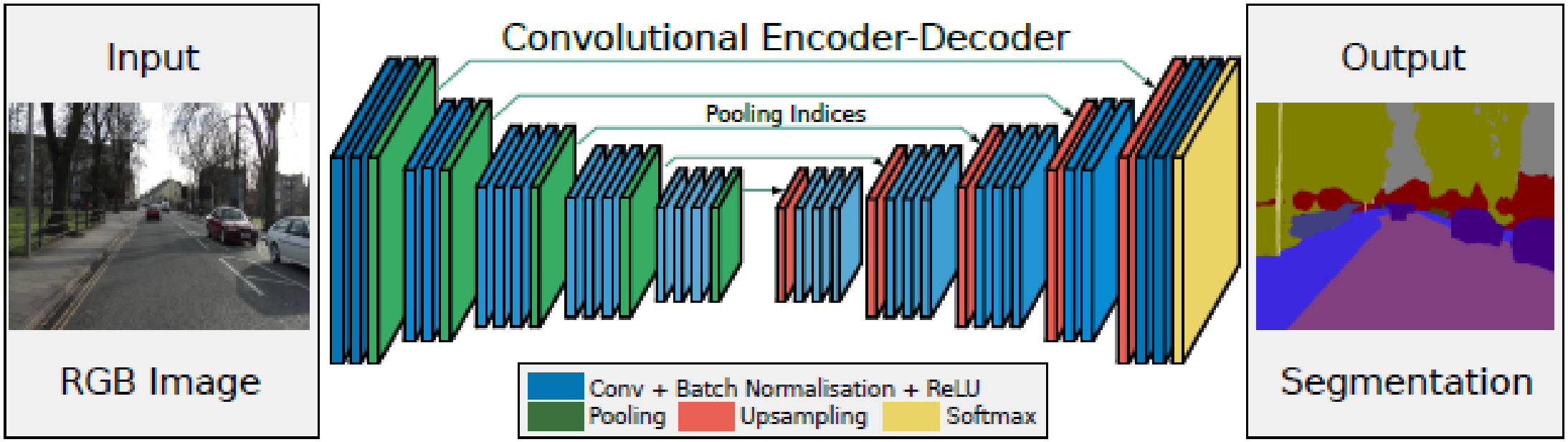
$$M^{T} \cdot \mathrm{vec}\left(\quad\right) \longrightarrow \mathrm{vec}\left(\quad\right)$$

逆プーリング（Unpooling）



SegNet (Semantic Segmentation)

Max Pooling時のIndexを転送して
位置に関する情報を補充している



Badrinarayanan, V., A. Kendall, and R. Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." arXiv. Preprint arXiv: 1511.0051, 2015.

学習データの定義（SegNet）

Step1) ラベルの ID番号とラベルのマッピングを決めておく

```
classNames = ["sky" "grass" "building" "sidewalk"];
pixelLabelID = [1 2 3 4];
```

Step2) 画像とピクセルラベルの組を定義する

```
imds = imageDatastore(imageDir);
pxds = pixelLabelDatastore(labelDir, classNames, pixelLabelID);
```

Step3) 画像とピクセルラベルの組から学習データを定義する

```
datasource = pixelLabelImageSource(imds, pxds);
```

学習と推論（SegNet）

Step1) SegNet のレイヤーを定義

VGG16 または VGG19 を選択

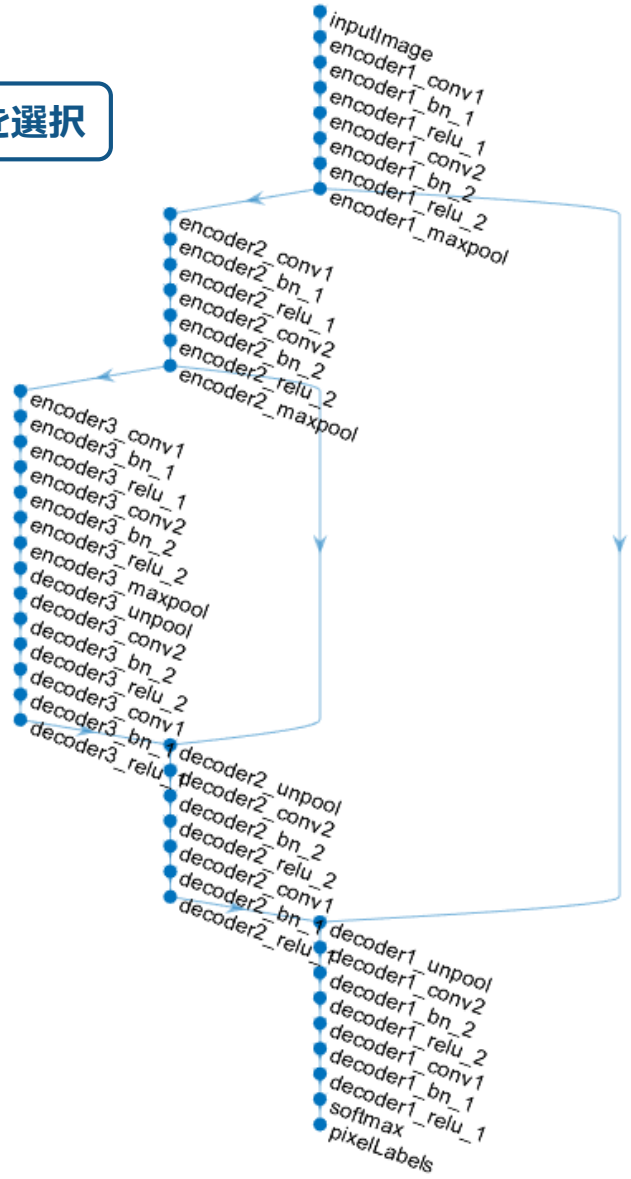
```
lgraph = segnetLayers(imageSize, numClasses, model);
```

Step2) データ源を指定して、学習を実行する

```
net = trainNetwork(datasource, lgraph, options);
```

Step3) 学習させたネットワークでセグメンテーションを行う

```
C = semanticseg(I, net);
```



その他の新機能

Regression with CNNs

R2017a

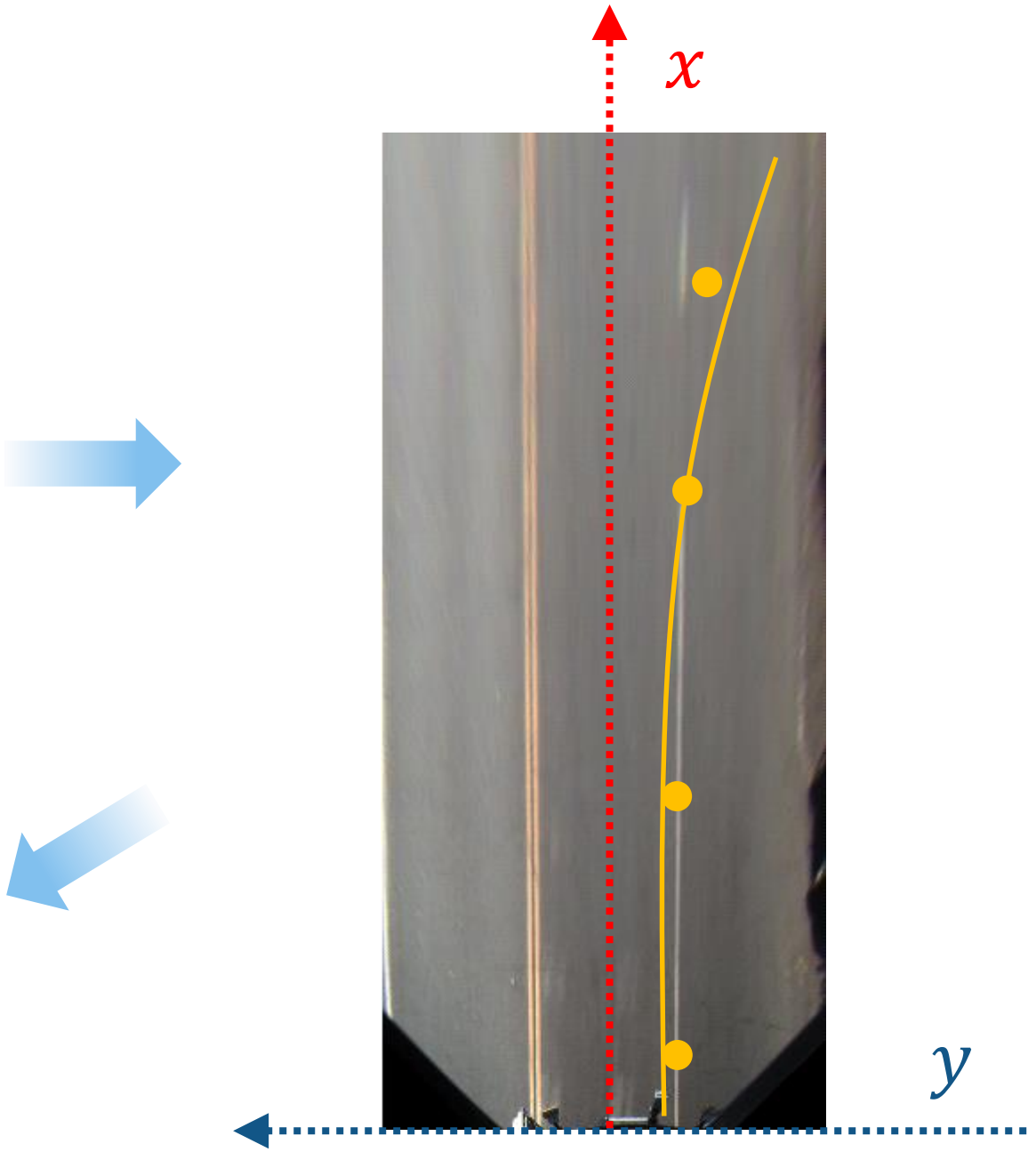
畳み込みニューラルネットによる回帰

ラベル付けされた白線

CNN により推定した白線

CNNによる回帰のデモ

$$y = ax^2 + bx + c$$



CNNによる回帰のデモ

画像

右側の白線： (a_1, b_1, c_1)

左側の白線： (a_2, b_2, c_2)

6 次元ベクトル

学習済みモデル / インポート機能

R2017b

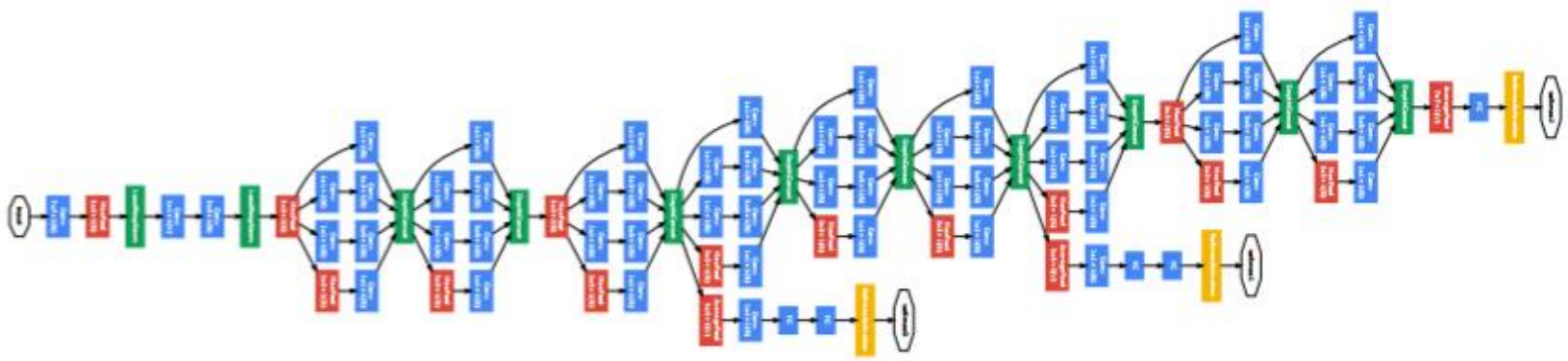
学習済みモデル（Pretrained Model）
- AlexNet / VGG-16 / VGG-19
- GoogLeNet / Resnet50 / ~~Inception V3~~

インポート機能（Model Importer）
- Caffe Model Importer
- TensorFlow/Keras Model Importer

たった一行で学習済みモデルを呼び出せる

コマンド ウィンドウ

```
>> net = googlenet

net =

  DAGNetwork のプロパティ:

       Layers: [144×1 nnet.cnn.layer.Layer]
  Connections: [170×2 table]
```



C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015

実行環境の切り替え / 高速化

MathWorks

R2017a

オプションひとつで CPU / GPU / Multi-GPU / Cluster を切り替えることが可能

`ExecutionEnvironment` — Hardware resource for training network

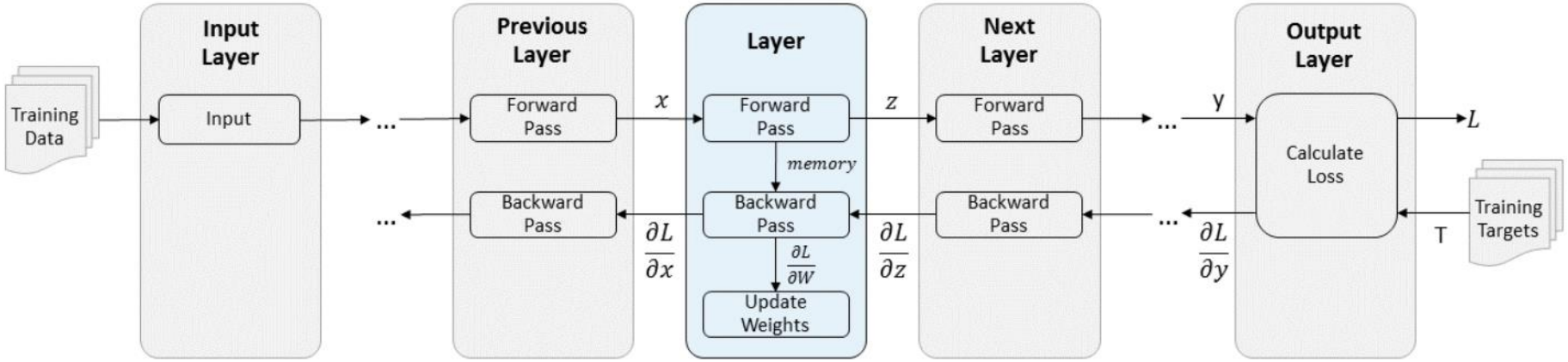
`'auto'` | `'cpu'` | `'gpu'` | `'multi-gpu'` | `'parallel'`

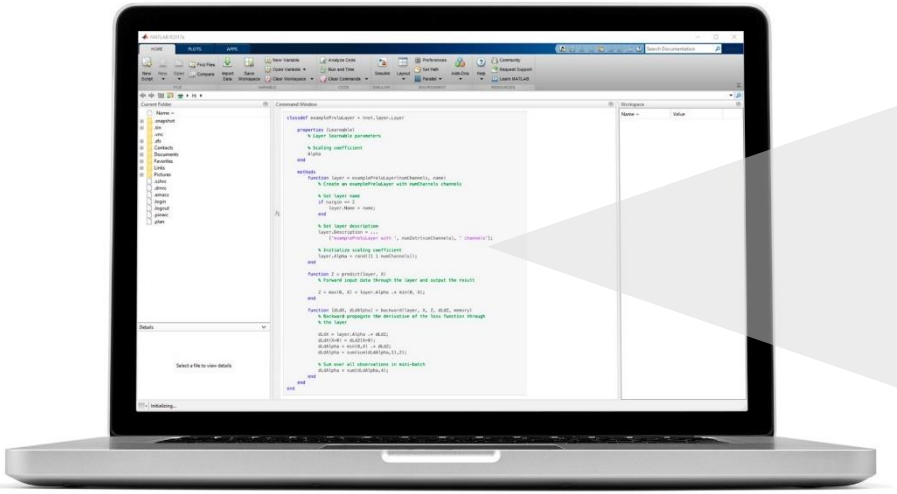
Hardware resource for training network, specified as one of the following:

- `'auto'` — Use a GPU if it is available, otherwise uses the CPU.
- `'cpu'` — Use the CPU.
- `'gpu'` — Use the GPU.
- `'multi-gpu'` — Use multiple GPUs on one machine, using a local parallel pool. If no pool is already open, `trainNetwork` opens one with one worker per supported GPU device.
- `'parallel'` — Use a local parallel pool or compute cluster. If no pool is already open, `trainNetwork` opens one using the default cluster profile. If the pool has access to GPUs, then `trainNetwork` uses them and excess workers are left idle. If the pool does not have GPUs, then the training takes place on all cluster CPUs.

User Define Custom Layer

R2017b





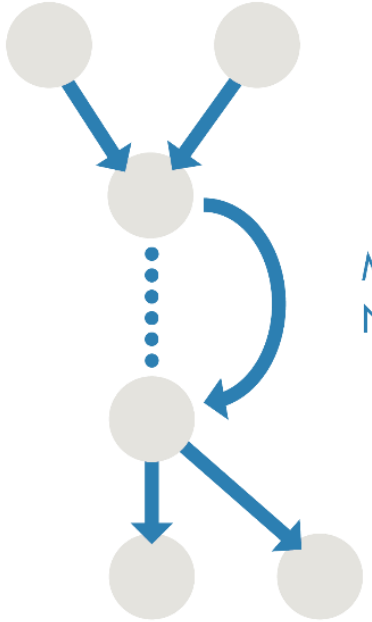
```
function [dLdX, dLdAlpha] = backward(layer, X, Z, dLdZ, memory)
    % Backward propagate the derivative of the loss function through
    % the layer

    dLdX = layer.Alpha .* dLdZ;
    dLdX(X>0) = dLdZ(X>0);
    dLdAlpha = min(0,X) .* dLdZ;
    dLdAlpha = sum(sum(dLdAlpha,1),2);
```

Various Deep Neural Networks

Series Network

Single-in
single-out

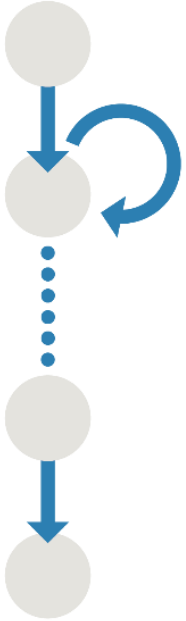
DAG Network



R2017b

Multi-in, multi-out
No feedback loops

Recurrent Network



R2017b

Memory

画像系ディープラーニングのための構成

深層学習に必要な Toolbox と Hardware

MATLAB ← R2017a以降のMATLABを推奨

Neural Network Toolbox™ ← 必須

Parallel Computing Toolbox™ ← GPUを使う場合 ⋯⋯▶

NVIDIA® のチップを搭載したGPU
（Compute Capability 3.0以降）

Statistics and Machine Learning Toolbox™ ← R-CNN で必須

Image Processing Toolbox™

Computer Vision System Toolbox™ ← R-CNN, Fast R-CNN, Faster R-CNN
Semantic Segmentation 等で必須