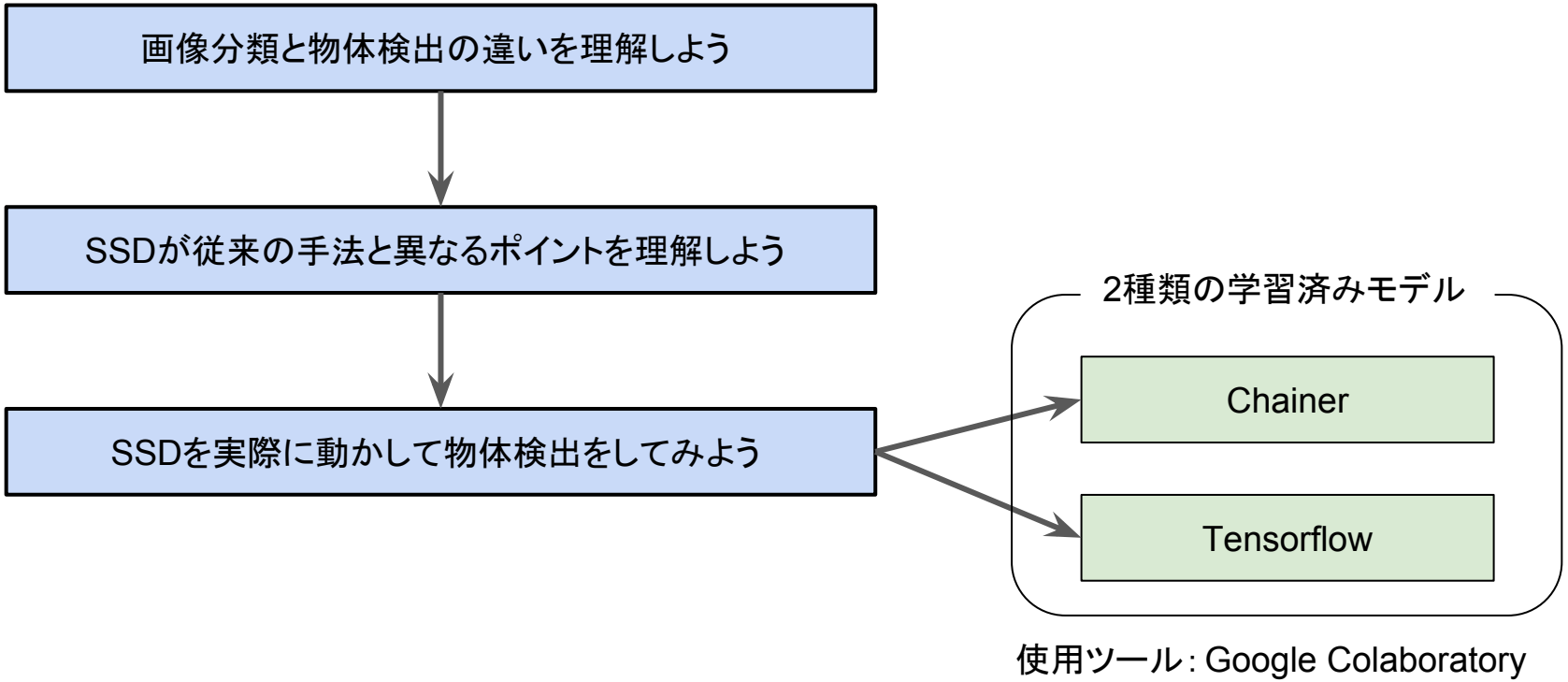
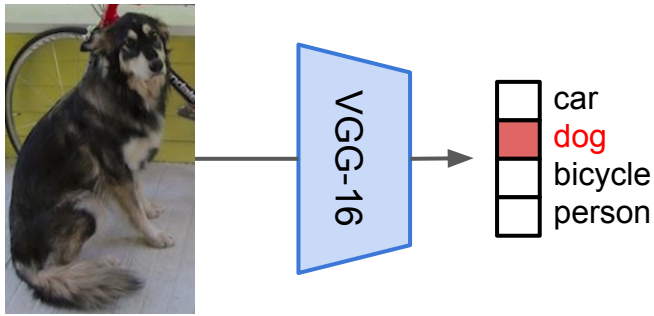
本セクションのモチベーション



使用ツール：Google Colaboratory

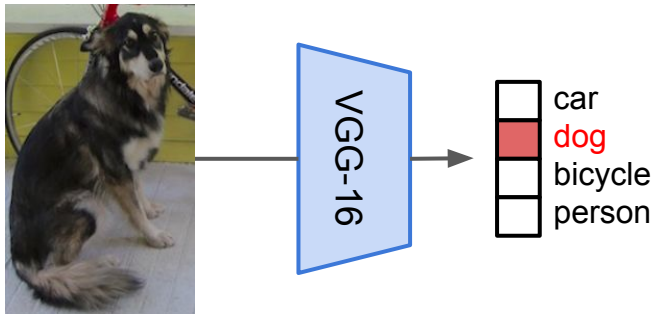
画像分類から物体検出へ

画像全体に**何が**写っているか

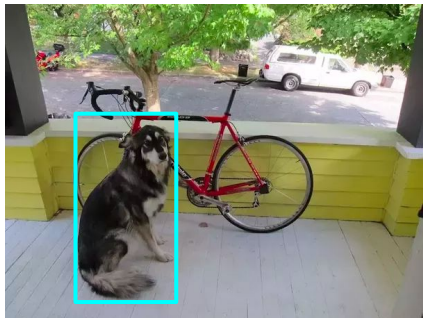


画像分類から物体検出へ

画像全体に**何が**写っているか

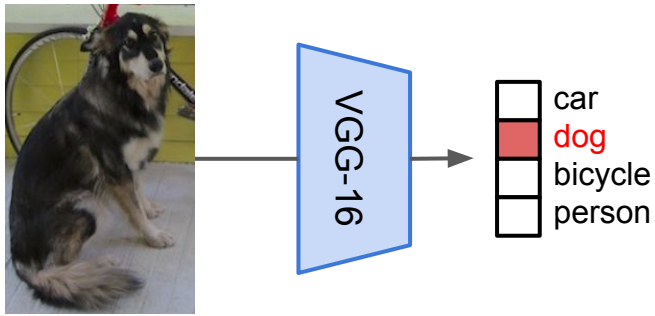


画像の**どこ**に**何が**写っているか

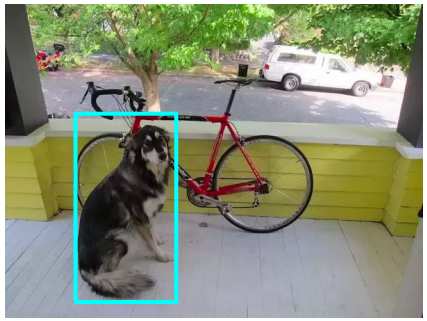


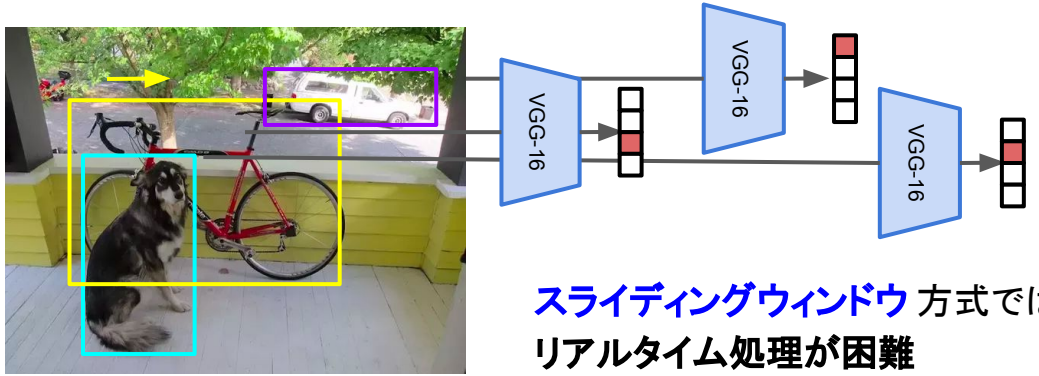
画像分類から物体検出へ

画像全体に**何が**写っているか



画像の**どこ**に**何が**写っているか





スライディングウィンドウ 方式では計算量が多く
リアルタイム処理が困難

DeepLearningで物体検出

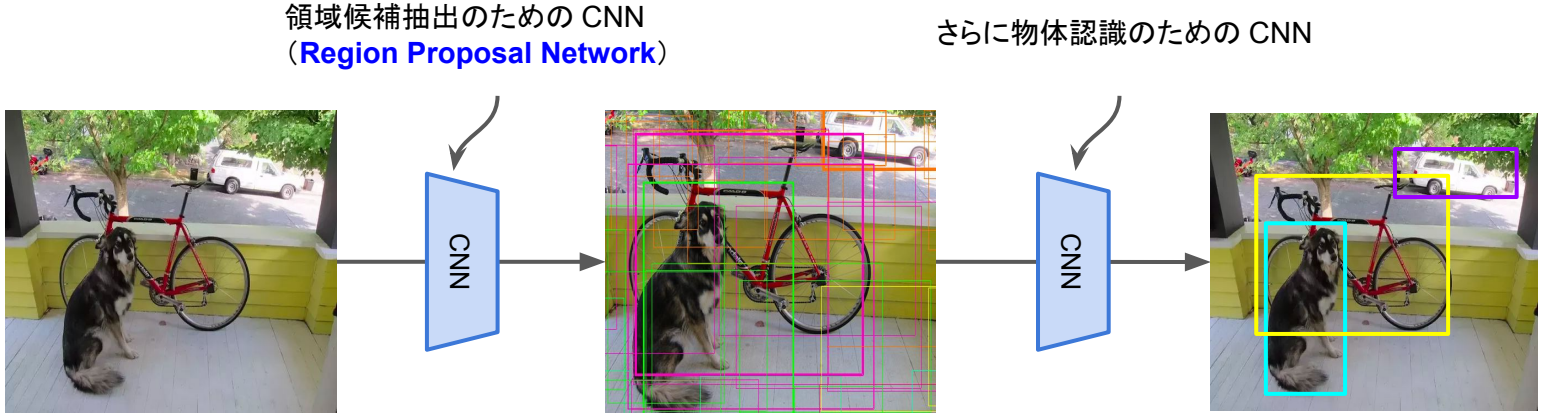
- 従来の手法
  - R-CNN、Fast R-CNN、Faster R-CNN など
    これらは「物体らしい箇所の検出」のための処理を含む

DeepLearningで物体検出

- 従来の手法
  - R-CNN、Fast R-CNN、Faster R-CNN など
    これらは「物体らしい箇所の検出」のための処理を含む
  - 計算コストが大きくリアルタイム処理が難しかった

DeepLearningで物体検出

- 従来の手法
  - R-CNN、Fast R-CNN、Faster R-CNN など
    これらは「物体らしい箇所の検出」のための処理を含む
  - 計算コストが大きくリアルタイム処理が難しかった

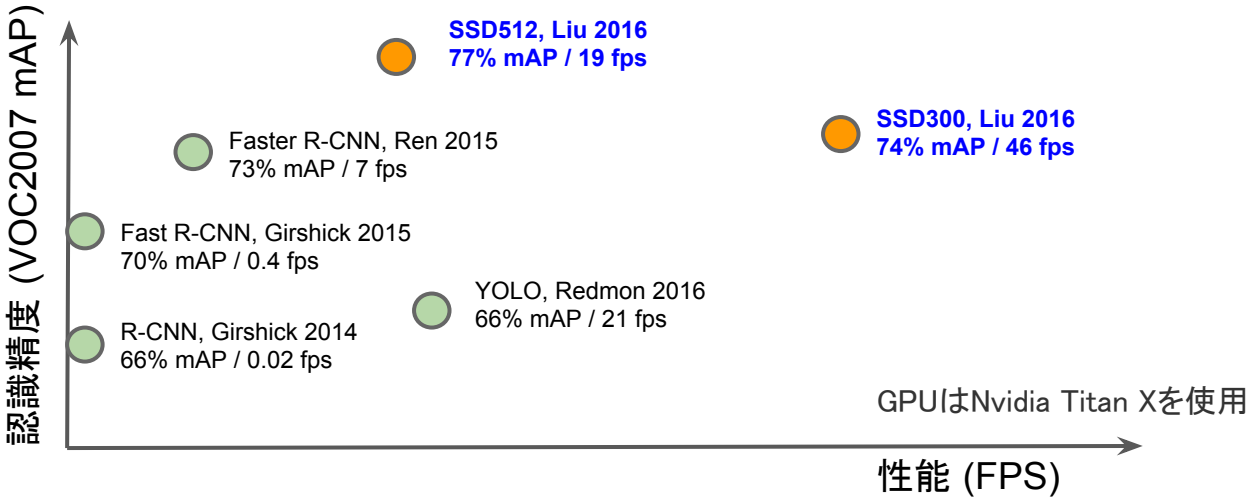


Single Shot Multibox Detector (SSD)

- 注目領域を絞り込まない物体検出のためのCNNアーキテクチャ
  - 1度のCNN演算で**クラス分類**と**位置検出**の両方を行うことができる
  - 余分な処理がないため**高い認識精度**で**高速処理**が可能

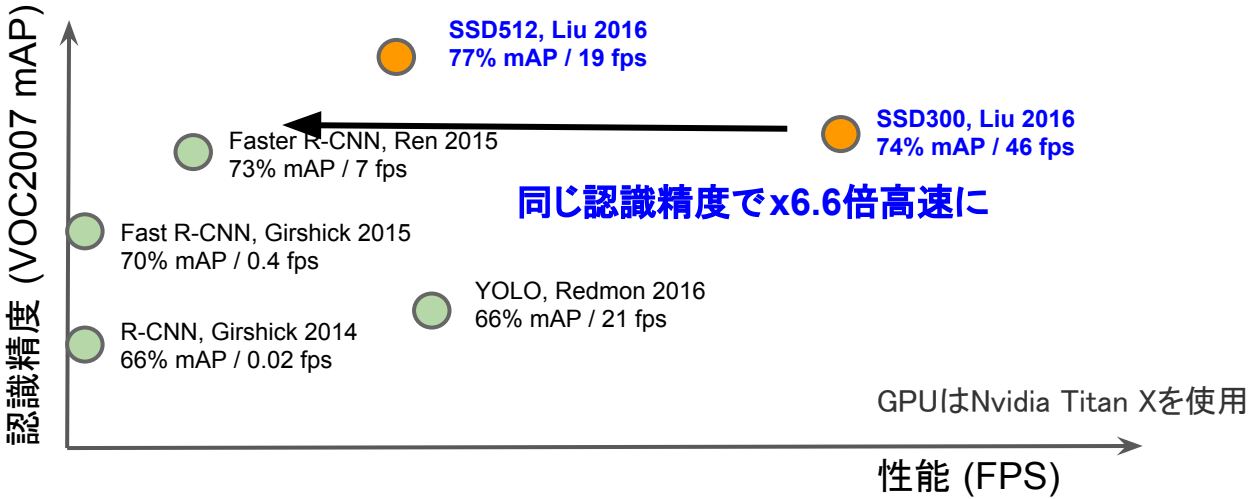
Single Shot Multibox Detector (SSD)

- 注目領域を絞り込まない物体検出のためのCNNアーキテクチャ
  - 1度のCNN演算で**クラス分類**と**位置検出**の両方を行うことができる
  - 余分な処理がないため**高い認識精度**で**高速処理**が可能
- 演算性能の比較



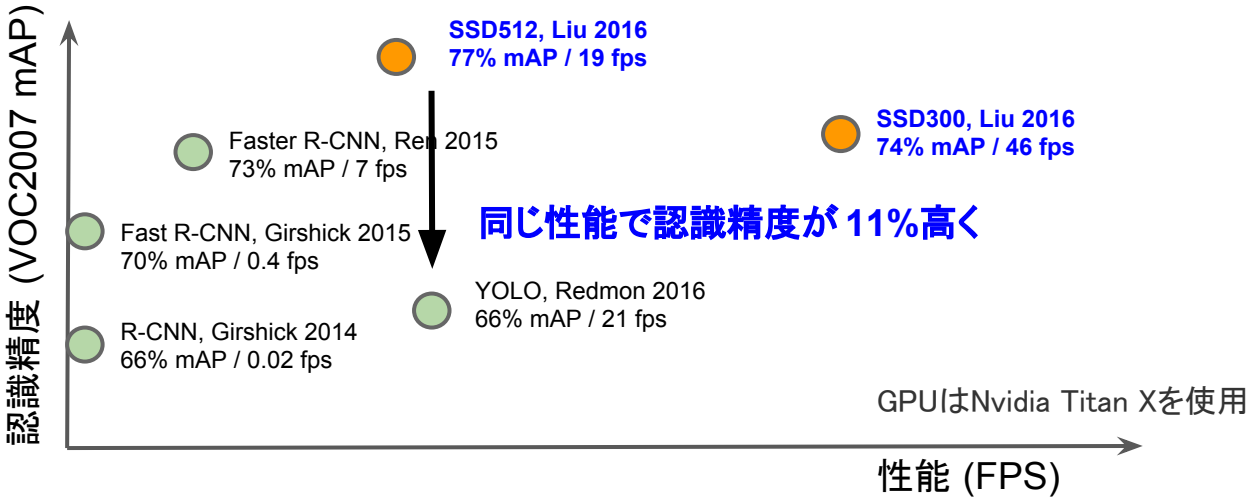
Single Shot Multibox Detector (SSD)

- 注目領域を絞り込まない物体検出のためのCNNアーキテクチャ
  - 1度のCNN演算で**クラス分類**と**位置検出**の両方を行うことができる
  - 余分な処理がないため**高い認識精度**で**高速処理**が可能
- 演算性能の比較



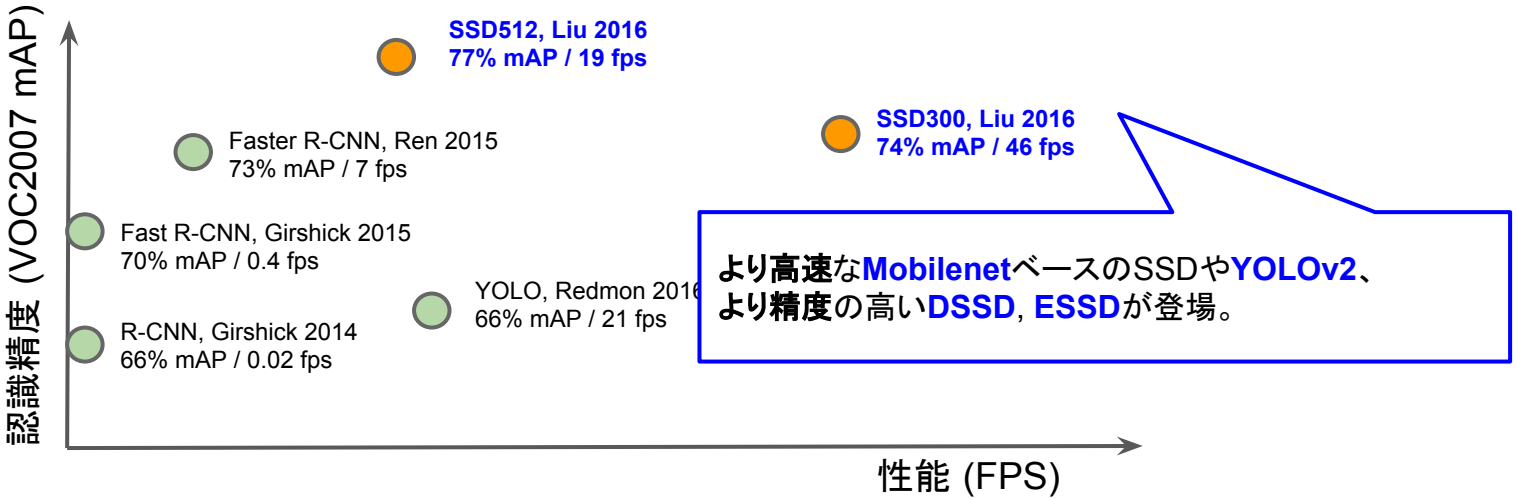
Single Shot Multibox Detector (SSD)

- 注目領域を絞り込まない物体検出のためのCNNアーキテクチャ
  - 1度のCNN演算で**クラス分類**と**位置検出**の両方を行うことができる
  - 余分な処理がないため**高い認識精度**で**高速処理**が可能
- 演算性能の比較



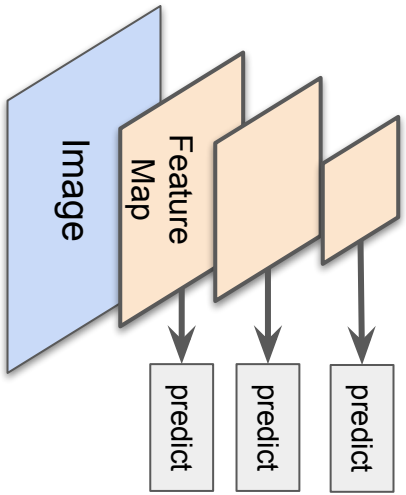
Single Shot Multibox Detector (SSD)

- 注目領域を絞り込まない物体検出のためのCNNアーキテクチャ
  - 1度のCNN演算で**クラス分類**と**位置検出**の両方を行うことができる
  - 余分な処理がないため**高い認識精度**で**高速処理**が可能
- 演算性能の比較



SSDとYOLOの違い

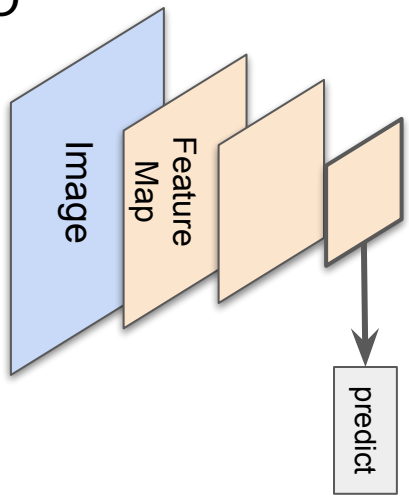
SSD



o 多オブジェクトの検出に強い
× 大きなスケールの特徴抽出が弱い

https://github.com/weiliu89/caffe/tree/ssd
BSD 2-Clause License

YOLO



o カテゴリ数が多く検出速度が速い
× 多オブジェクト検出に弱い
× 特徴マップの解像度が低い

https://github.com/pjreddie/darknet
License Free

問題

SSDの特徴

- **画像分類と物体検出の違いはなんでしょうか？**

- **物体検出の従来の手法と比べてSSDは何が新しいでしょうか？**

答え

SSDの特徴

- **画像分類と物体検出の違いはなんでしょうか？**
  ・画像分類＝画像全体に何が写っているかを分類すること
  　物体検出＝画像のどこに何が写っているかを分類する
  ・物体検出は画像に複数の物体が映っていても検出できる

- **物体検出の従来の手法と比べてSSDは何が新しいでしょうか？**
  ・「物体らしさ」の検出処理を排除し、一度のCNN演算で物体検出を行う

SSD300を動かしてみる

- GoogleのColaboratoryを使用

  環境構築がほぼ不要で、GPUが無料で利用可能

  https://colab.research.google.com/

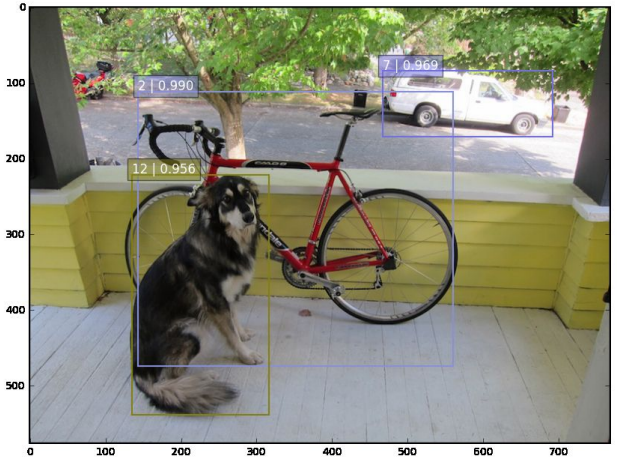
- 2種類のDeepLearning Framwork の学習済みモデルで試してみる

  ChainerCV :

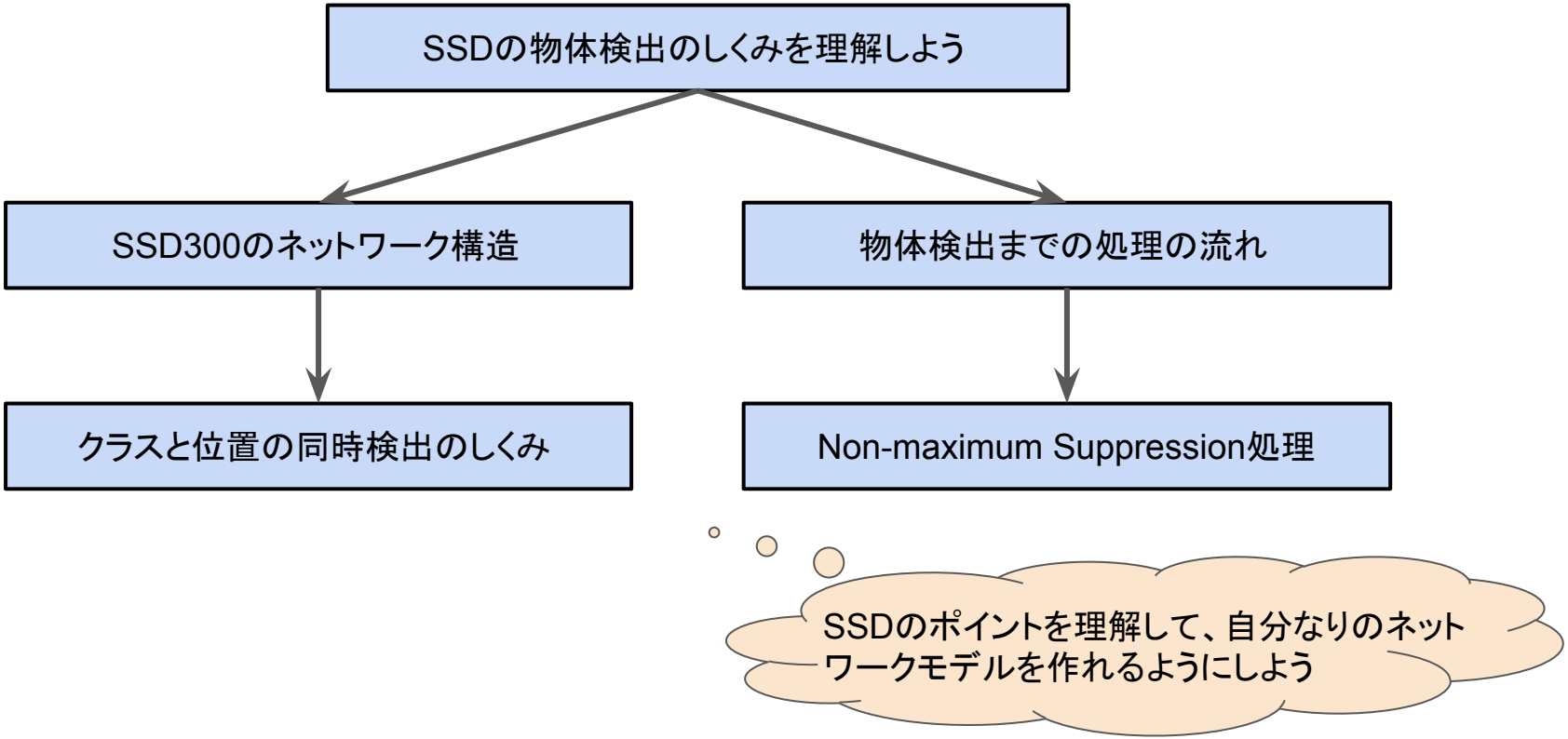
  https://github.com/chainer/chainercv

  Tensorflow Object Detection API :

  https://github.com/tensorflow/models/tree/master/research/object_detection

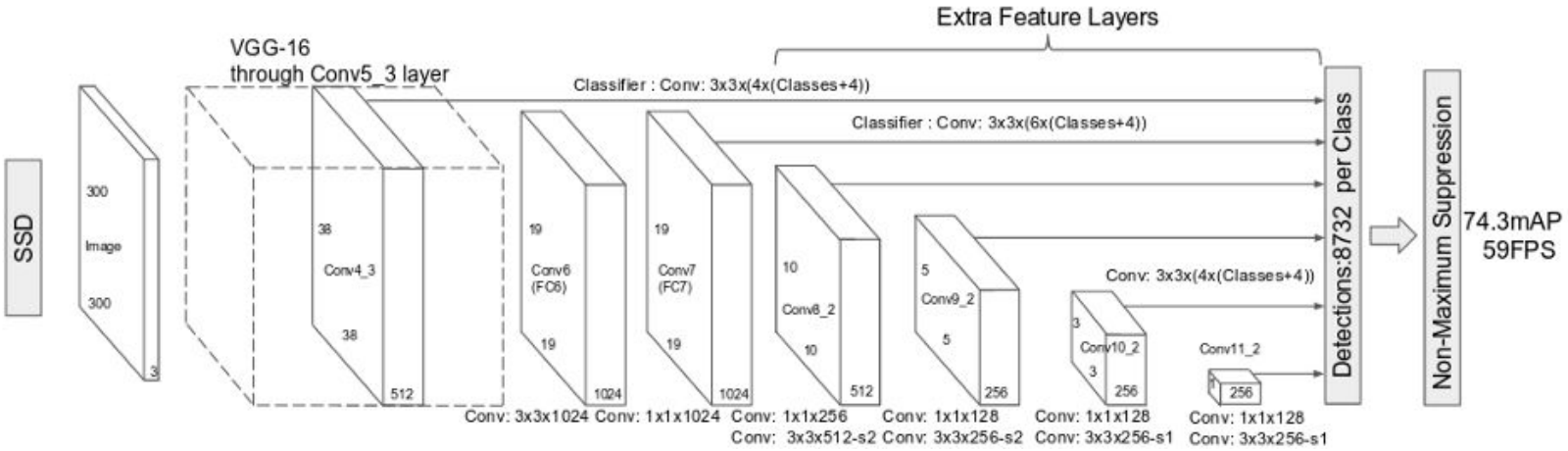


本セクションのモチベーション



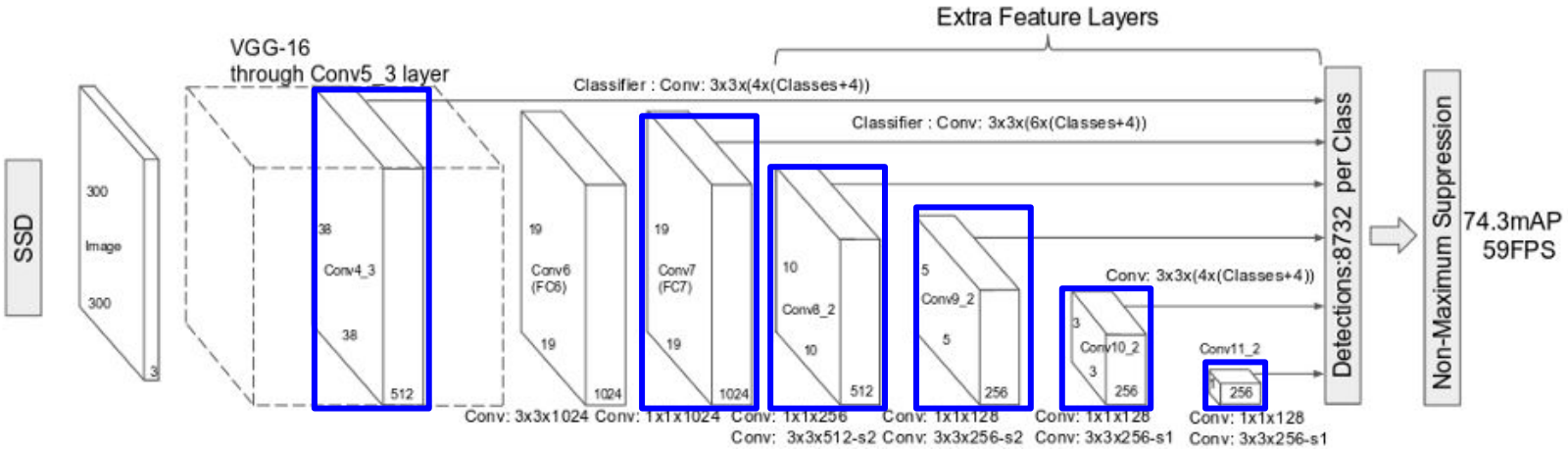
SSD300のネットワーク構造

- 入力サイズは300x300
- ベースネットワークは**VGG-16**
- **畳込み層**さらに追加
- 途中の特徴マップから①クラス分類と②物体位置検出のための**2種類の畳込み層**を追加



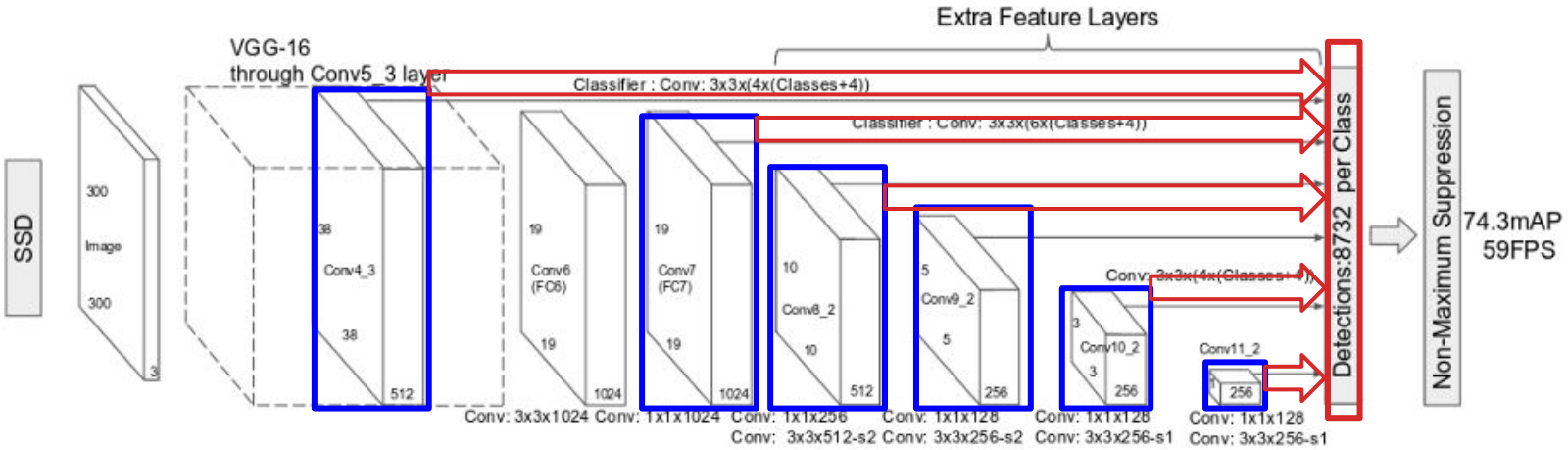
SSD300のネットワーク構造

- 入力サイズは300x300
- ベースネットワークは**VGG-16**
- **畳込み層**さらに追加
- 途中の特徴マップから①クラス分類と②物体位置検出のための**2種類の畳込み層**を追加



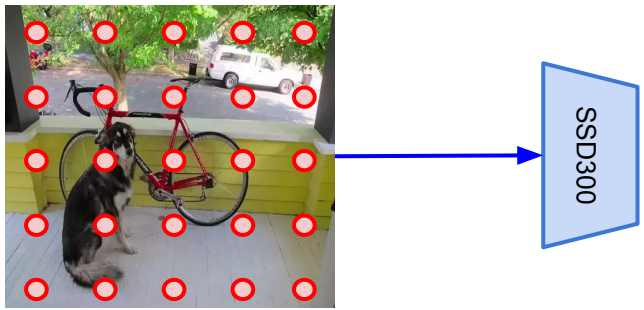
SSD300のネットワーク構造

- 入力サイズは300x300
- ベースネットワークは**VGG-16**
- **畳込み層**さらに追加
- 途中の特徴マップから①クラス分類と②物体位置検出のための**2種類の畳込み層**を追加



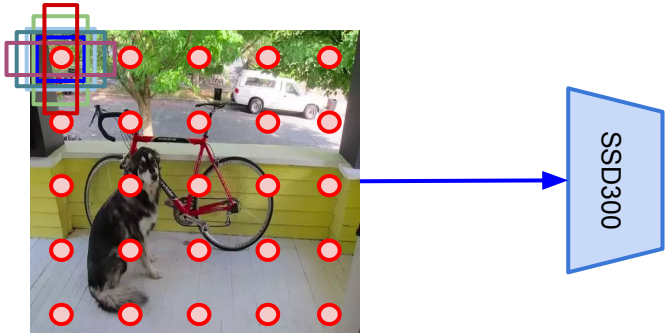
SSD300がどうやって物体検出するか

1. 画像に目印(**Anchor**)を複数のスケールで等間隔に設定



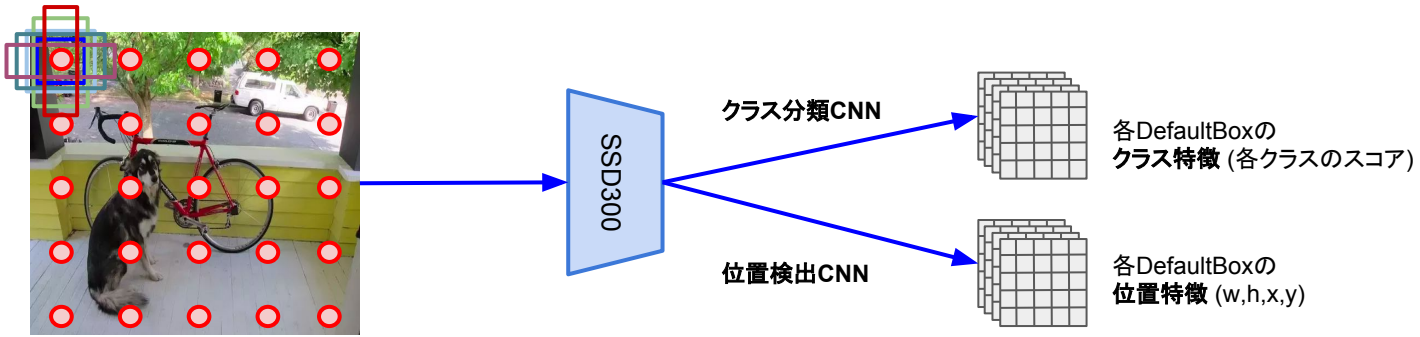
SSD300がどうやって物体検出するか

1. 画像に目印(**Anchor**)を複数のスケールで等間隔に設定
2. 様々なアスペクト比の**DefaultBox**を複数のアスペクト比で設定



SSD300がどうやって物体検出するか

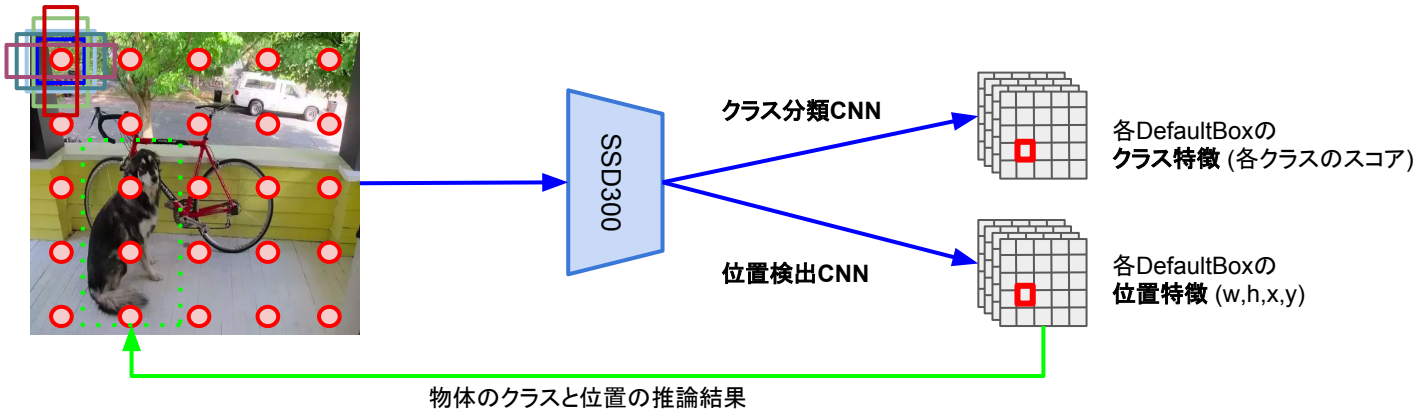
1. 画像に目印(**Anchor**)を複数のスケールで等間隔に設定
2. 様々なアスペクト比の**DefaultBox**を複数のアスペクト比で設定
3. 各DefaultBoxの**クラス特徴**と**位置特徴**をSSD300で抽出



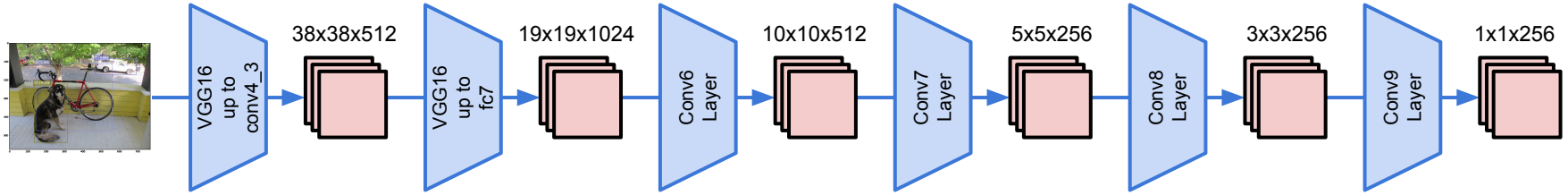
SSD300がどうやって物体検出するか

1. 画像に目印(**Anchor**)を複数のスケールで等間隔に設定
2. 様々なアスペクト比の**DefaultBox**を複数のアスペクト比で設定
3. 各DefaultBoxの**クラス特徴**と**位置特徴**をSSD300で抽出

⇒ 画像の中の**様々なスケール・アスペクト比の物体**を検出する



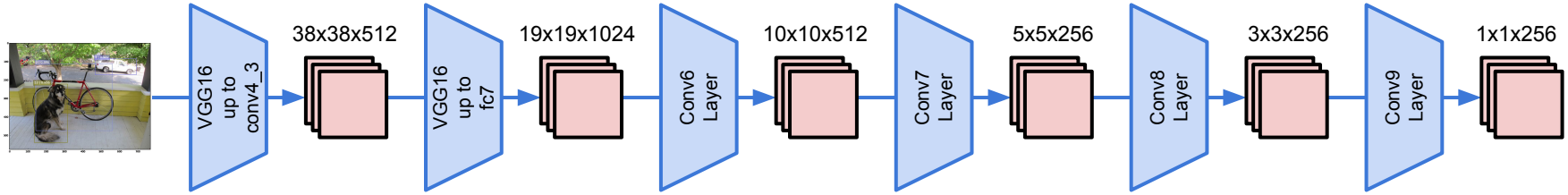
問題 DefaultBoxの合計数は？



	①	②	③	④	⑤	⑥
Anchorの個数	38x38	19x19	10x10	5x5	3x3	1x1
アスペクト比の種類	4	6	6	6	4	4

答え

DefaultBoxの合計数は？

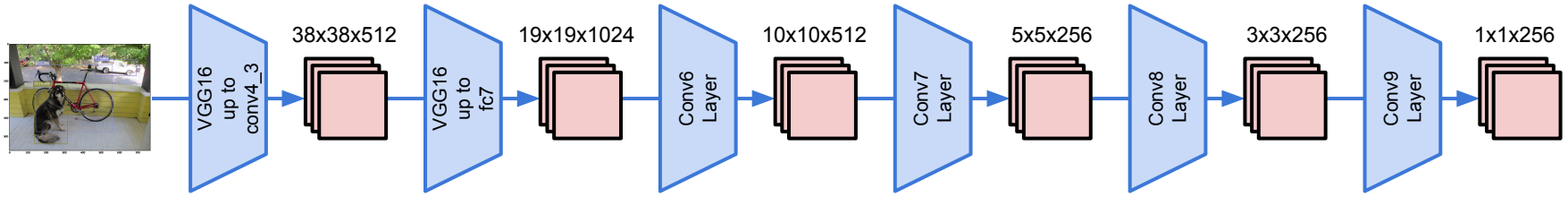


	①	②	③	④	⑤	⑥
Anchorの個数	38x38	19x19	10x10	5x5	3x3	1x1
アスペクト比の種類	4	6	6	6	4	4
DefaultBox数	5776	2166	600	150	36	4

5776 + 2166 + 600 + 150 + 36 + 4 = 8732

答え：合計 8732 個のDefaultBox

異なるスケールの特徴を抽出するしくみ

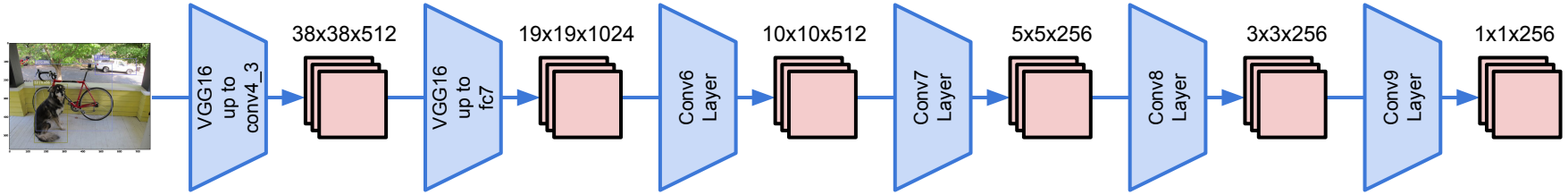


VGG-16と同じ

VGG-16の全結合層を
畳込み層に変更

4つの畳込み層を追加

異なるスケールの特徴を抽出するしくみ



VGG-16と同じ

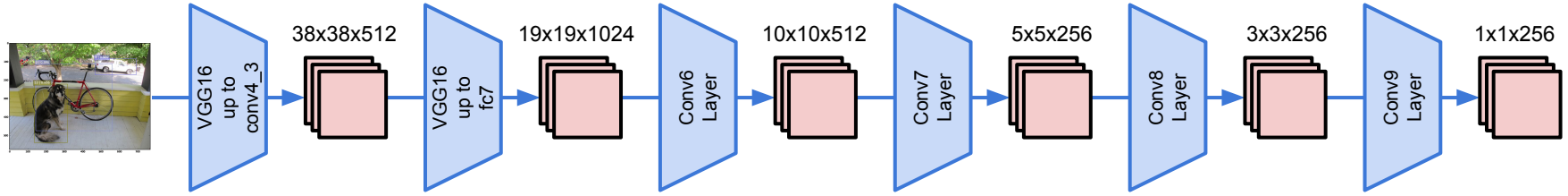
VGG-16の全結合層を畳込み層に変更

4つの畳込み層を追加

浅い層からは小さい物体を検出

深い層からは大きい物体を検出

異なるスケールの特徴を抽出するしくみ



VGG-16と同じ

VGG-16の全結合層を畳込み層に変更

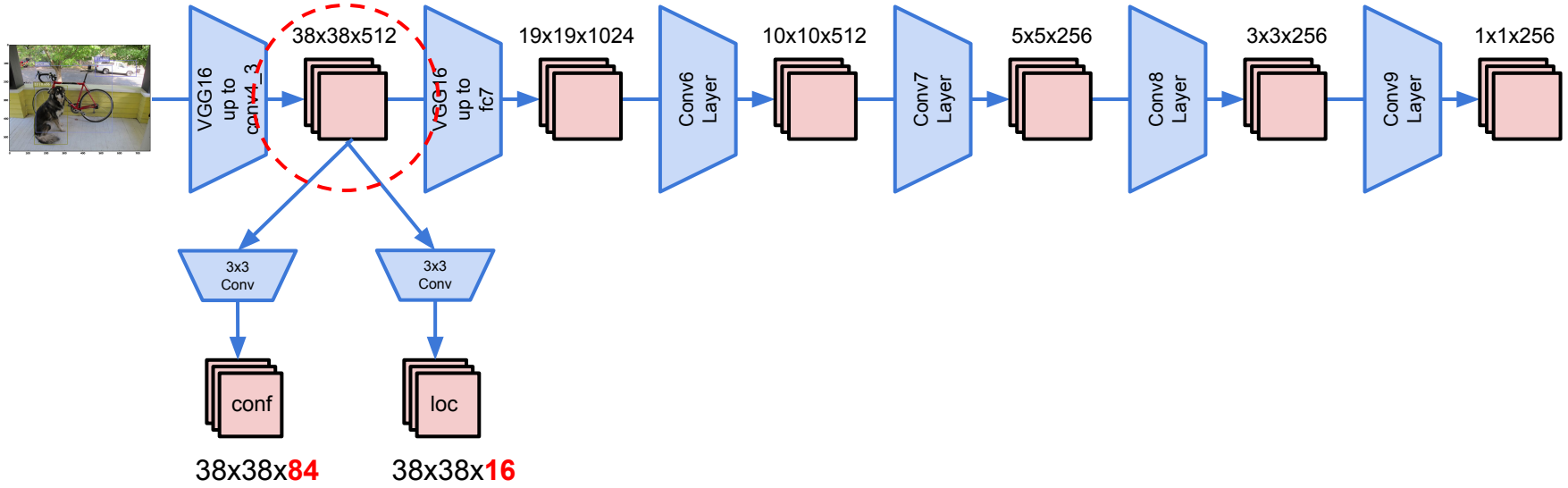
4つの畳込み層を追加

浅い層からは小さい物体を検出

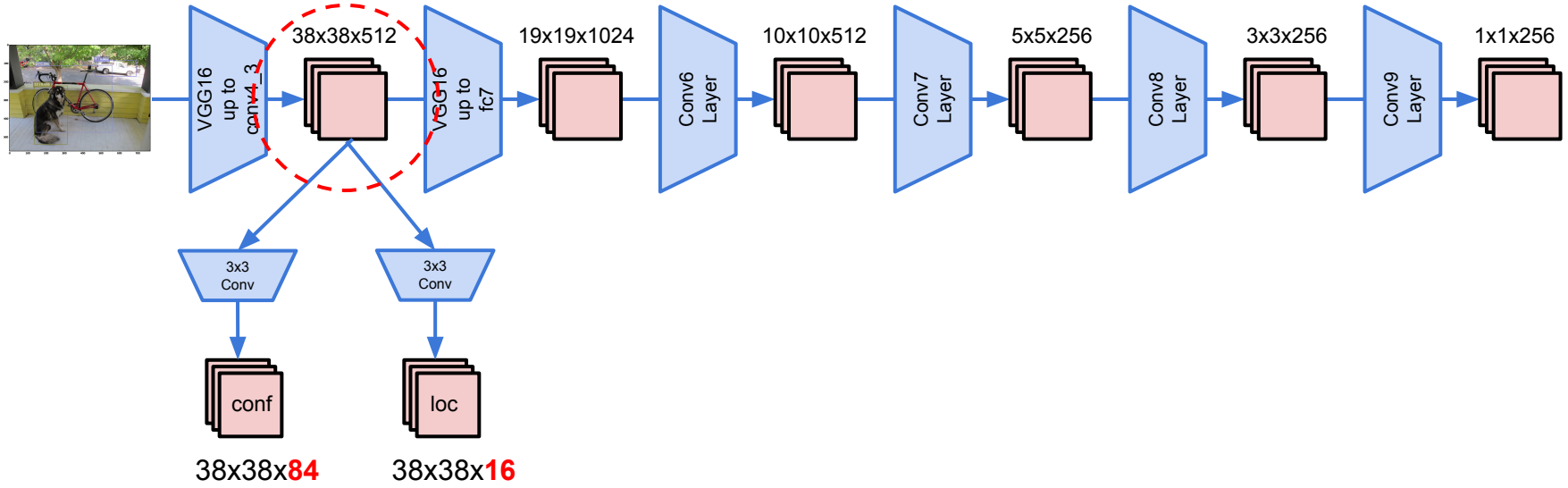
深い層からは大きい物体を検出

異なるスケールの特徴を一度の畳込み演算で抽出

異なるスケールの特徴を抽出するしくみ



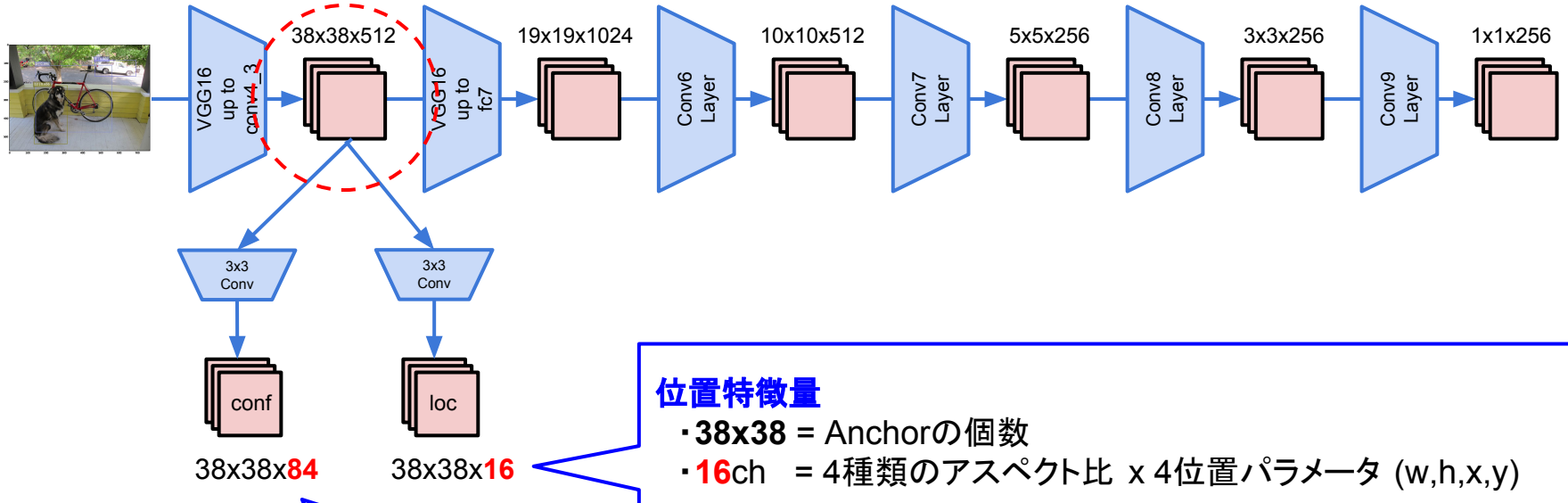
異なるスケールの特徴を抽出するしくみ



クラス特徴量

- **38x38** = Anchorの個数
- **84**ch = 4種類のアスペクト比 x 21クラス

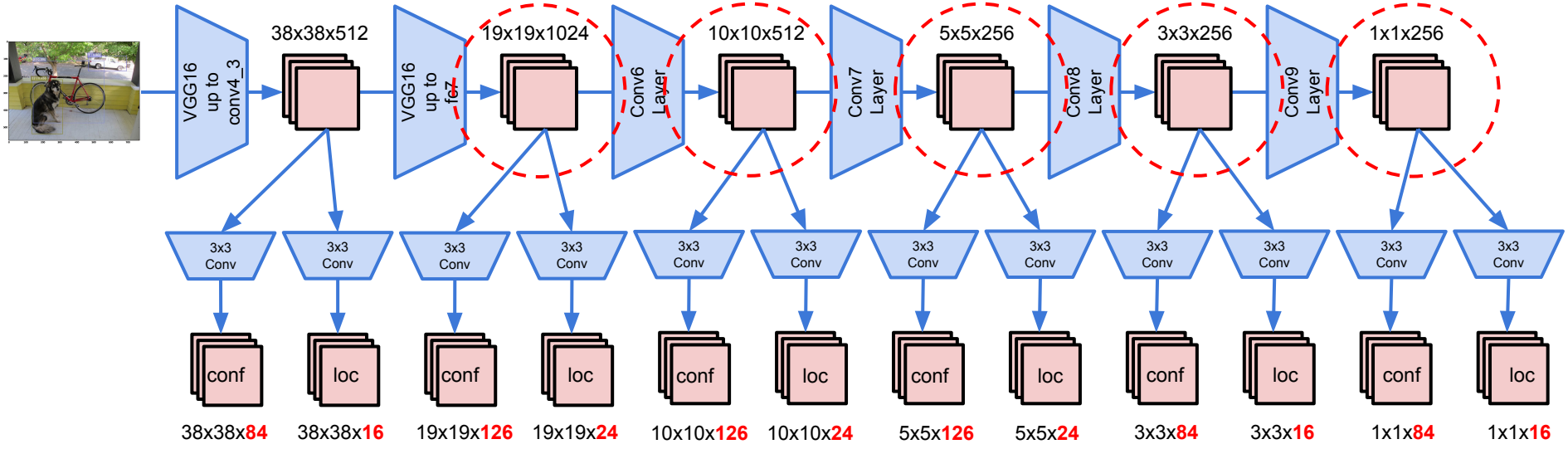
異なるスケールの特徴を抽出するしくみ



位置特徴量
- **38x38** = Anchorの個数
- **16**ch = 4種類のアスペクト比 x 4位置パラメータ (w,h,x,y)

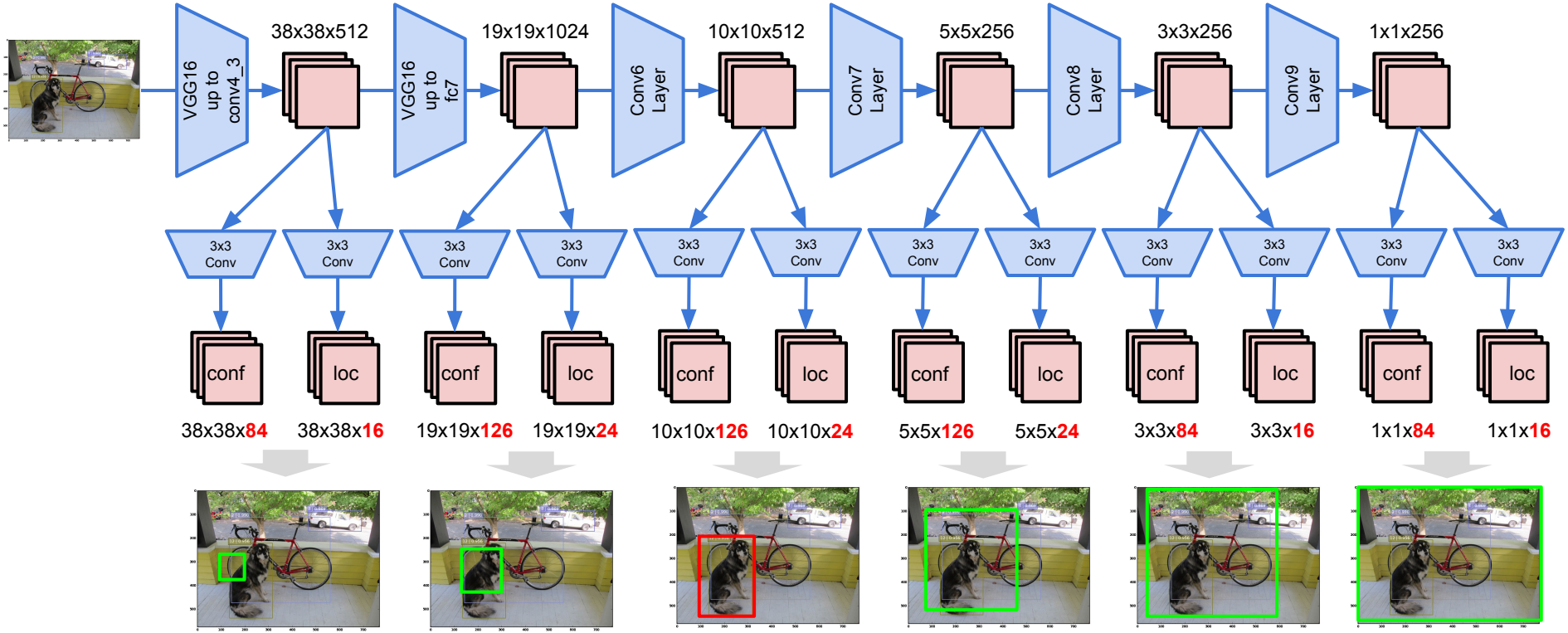
クラス特徴量
- **38x38** = Anchorの個数
- **84**ch = 4種類のアスペクト比 x 21クラス

異なるスケールの特徴を抽出するしくみ

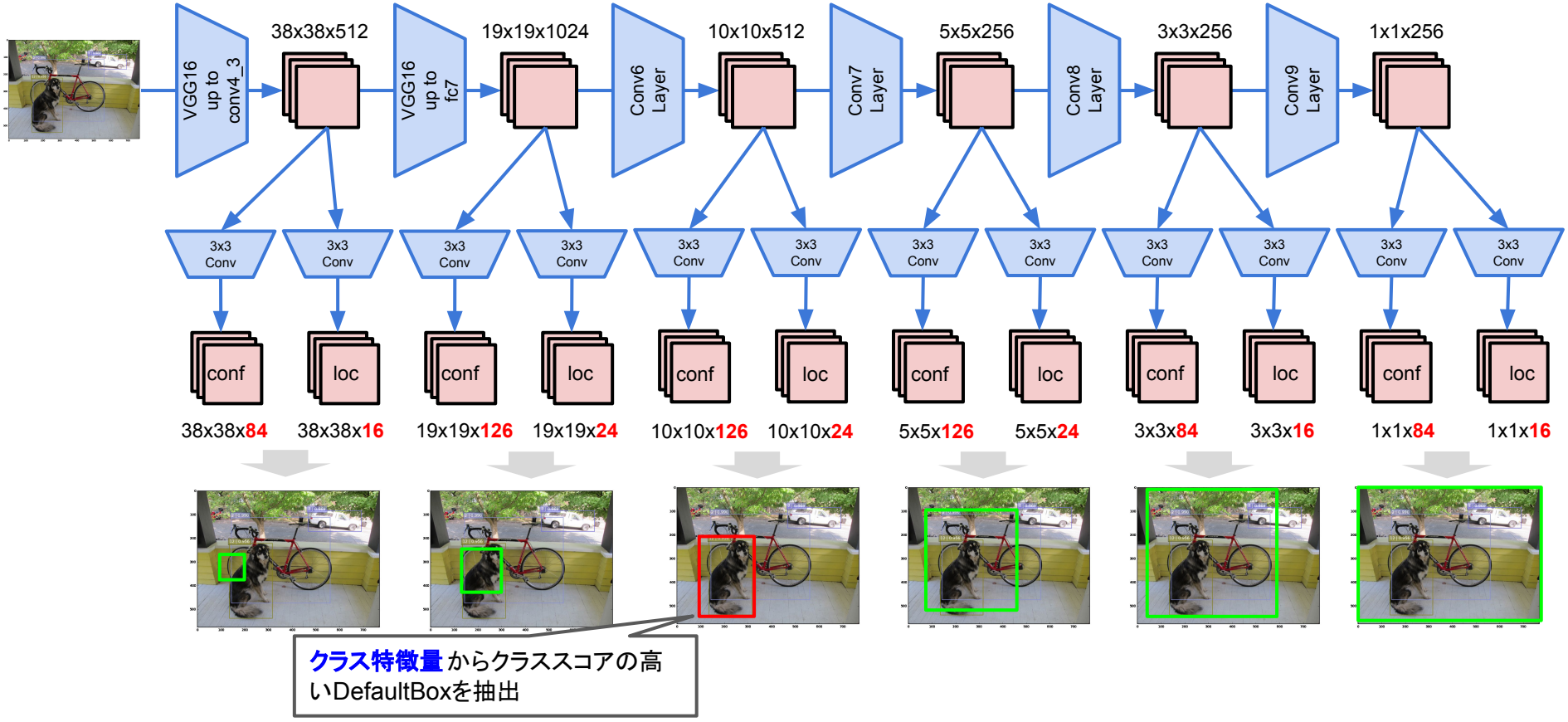


他の特徴マップにも同様に2種類の畳込み層を追加

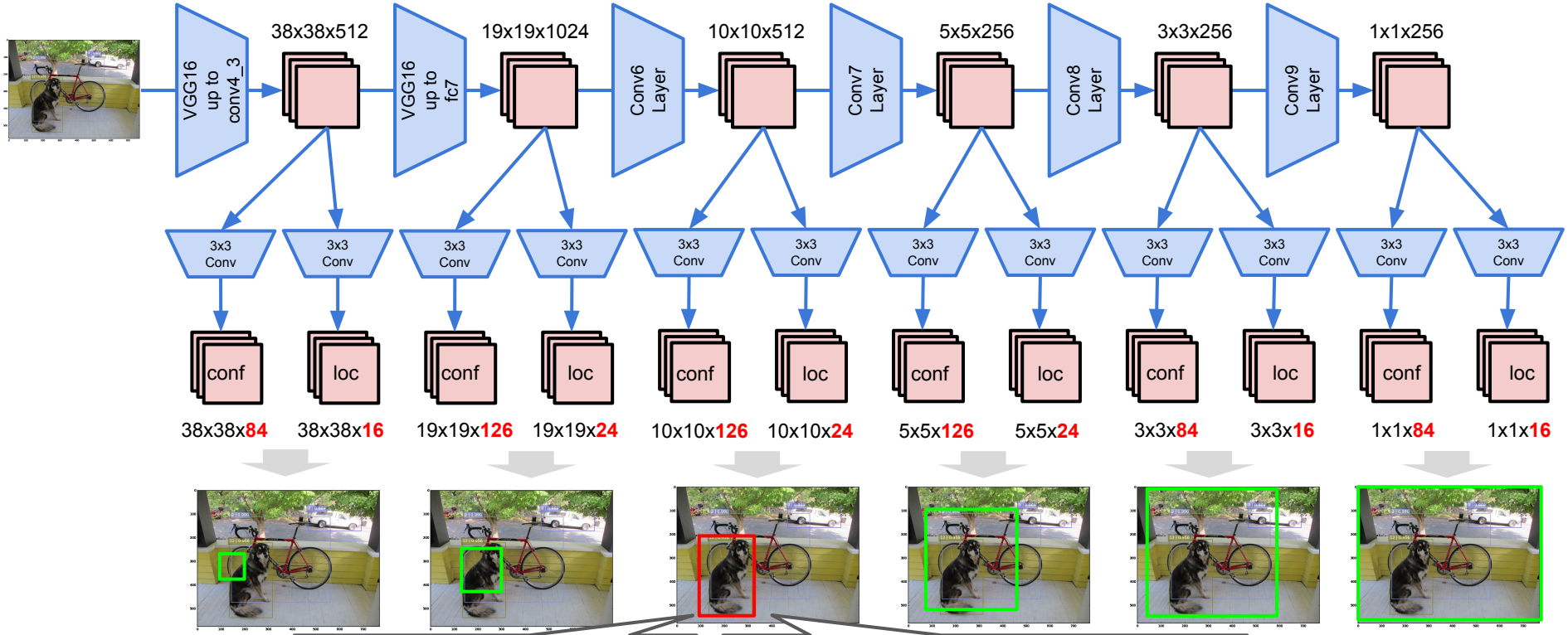
異なるスケールの特徴を抽出するしくみ



異なるスケールの特徴を抽出するしくみ



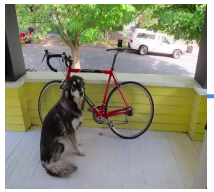
異なるスケールの特徴を抽出するしくみ



クラス特徴量 からクラススコアの高いDefaultBoxを抽出

位置特徴量 からDefaultBoxからの物体位置のずれを算出して正確な物体位置を求める

Non-maximum Suppression



SSD300

mbox_loc

mbox_conf

DetectionOutput layer

Priorbox parameter

Multibox Decoder

実際の画像上の座標に変換

nms_threshold=0.45
conidence_threshold=0.01
top_k=20

Non-maximum supression

重複する物体位置候補を除外

detection_out

スコアが高い
物体位置候補 (BoundingBox) を抽出

各カテゴリごとに
BoundingBoxの重なり率 (**IoU**)を計算

重なり率が高いBoundingBoxは
スコアの低い方を除外

IoU=Intersection over Union
:2領域の画像の重なりの割合

問題

NMS処理

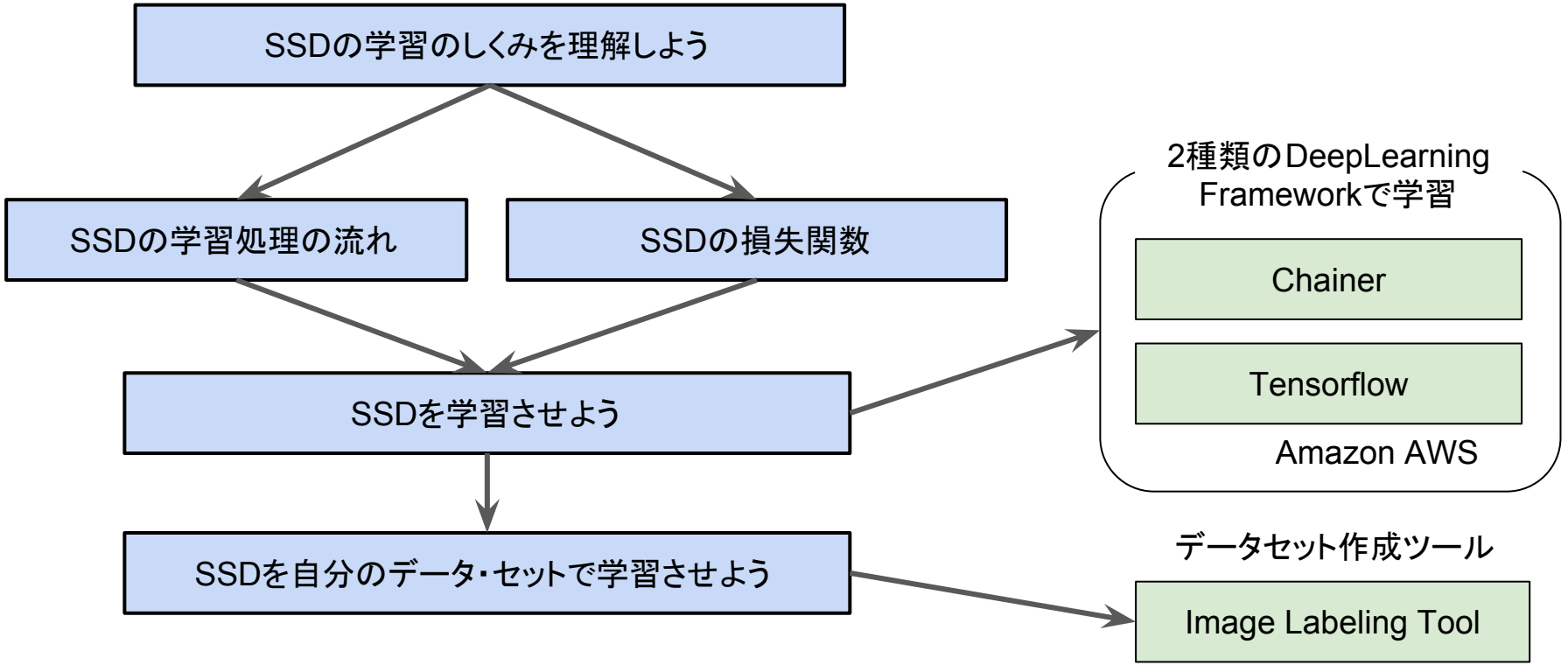
- NMS処理の目的は何でしょうか？
- NMSに関わるパラメータを挙げてください。
- NMSの処理内容を説明してください。

答え

NMS処理

- **NMS処理の目的は何でしょうか？**
  - 重複するBoundingBox候補を除外するため
- **NMSに関わるパラメータを挙げてください。**
  - クラススコアの閾値
  - BoundingBoxの重なり率 (IoU) の閾値
- **NMSの処理内容を説明してください。**
  - クラススコアが閾値以上のBoundingBoxを抽出する
  - 同一クラスのBoundingBoxのIoUを計算
  - IoUが閾値以上のBoundingBoxについてスコアの低い方を削除

本セクションのモチベーション



SSDの学習

SSD300

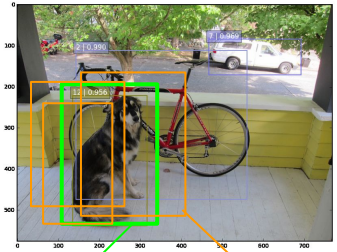
ネットワークの出力

mbox_loc

mbox_conf

位置情報

各クラスのスコア



Annotation Data
1) 正解BoundingBox
2) 正解クラス

SSDの推論結果

GroundTruth

BoundingBox位置の誤差

各クラスのスコアの誤差

smoothL1誤差計算

softmax損失計算

SSDの学習

SSD300

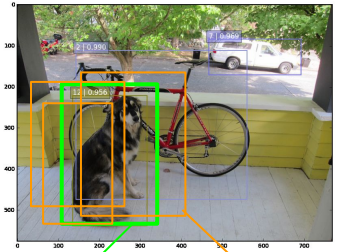
ネットワークの出力

mbox_loc

mbox_conf

位置情報

各クラスのスコア



Annotation Data
1) 正解BoundingBox
2) 正解クラス

SSDの推論結果

GroundTruth

BoundingBox位置の誤差

各クラスのスコアの誤差

smoothL1誤差計算

softmax損失計算

合成損失関数 計算

SSDの学習

SSD300

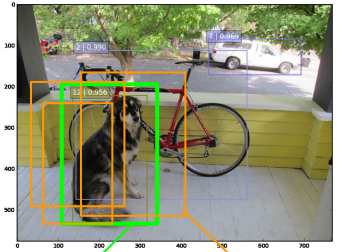
ネットワークの出力

mbox_loc

mbox_conf

位置情報

各クラスのスコア



Annotation Data
1) 正解BoundingBox
2) 正解クラス

SSDの推論結果

GroundTruth

BoundingBox位置の誤差

各クラスのスコアの誤差

smoothL1誤差計算

softmax損失計算

誤差逆伝搬
BackPropagation

合成損失関数 計算

SSDの損失関数

① 位置検出の損失関数: **smoothL1損失**

$$L_{loc}(x,l,g) = \sum_{i \in Pos} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^k \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

② クラス分類器の損失関数: **softmax損失 (交差エントロピー)**

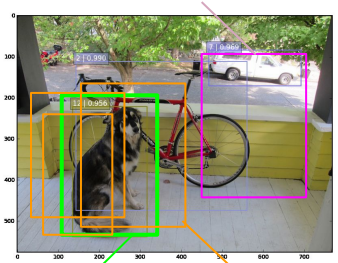
$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

合成損失関数

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$

SSDの損失関数

それ以外の大部分が Neg(負例:背景)



GroundTruth

Pos(正例)データ
GroundTruthとIoUが閾値以上

① 位置検出の損失関数: **smoothL1損失**

位置の誤差関数は
正例についてのみ 積算

$$L_{loc}(x,l,g) = \sum_{i \in Pos} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^k \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

BoundingBoxの中心位置のずれと
サイズのずれの両方を考慮

② クラス分類器の損失関数: **softmax損失 (交差エントロピー)**

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

合成損失関数

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$

SSDの損失関数

① 位置検出の損失関数：**smoothL1損失**

DefaultBox位置の推論結果と正解位置とのsmoothL1誤差を次式で計算し加算

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \mathrm{smooth}_{\mathrm{L1}}(l_i^m - \hat{g}_j^m)$$

$$\mathrm{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

画像上の座標からSSDの出力に合わせて変換
- x, yのズレはw,hで規格化
- w,hのズレは対数をとる

② クラス分類器の損失関数：**softmax損失 (交差エントロピー)**

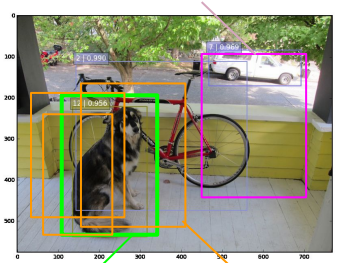
$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

合成損失関数

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$

SSDの損失関数

それ以外の大部分が Neg(負例:背景)



GroundTruth

Pos(正例)データ
GroundTruthとIoUが閾値以上

① 位置検出の損失関数: **smoothL1損失**

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

② クラス分類器の損失関数: **softmax損失 (交差エントロピー)**

各DefaultBox・各クラスについて
のスコア出力値の softmax損失を計算

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

合成損失関数

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$

GroundTruthとあるIoUが閾値以上の BoundingBoxを
Pos(正例)、それ以外を**Neg(負例)**とする。
実際には負例が大量にあり学習を妨げるので **Pos:Neg=1:3**に
なるように調整 (ハードネガティブマイニング)。

SSDの損失関数

① 位置検出の損失関数：**smoothL1損失**

$$L_{loc}(x,l,g) = \sum_{i \in Pos} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^k \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h \qquad (2)$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

② クラス分類器の損失関数：**softmax損失 (交差エントロピー)**

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \qquad (3)$$

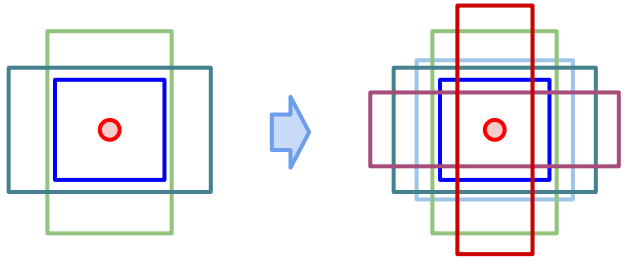
合成損失関数

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$

①と②の重み付き和 を損失関数とすることにより物体クラスと位置を両方学習できる

SSDで調整可能なパラメータ

- **DefaultBoxの設定**
  例：スケールやアスペクト比の種類を増やす
  メリット　= **認識精度が向上する**
  デメリット = **処理が重くなる**



- **ベースネットワーク**
  SSD300はVGG-16をベースとしているが、
  より軽量なネットワークや精度の高いネットワークにしてもよい
- **各種の閾値**
  例：NMS処理時の重なり率やスコア閾値を変えて検出感度を上げる
- **分類クラス**
  クラスを変えて再学習したいときは、**学習済みのベースネットワーク**
  を用いてSSD固有の畳込み層のみを再学習する

問題

SSDの学習

- SSDの損失関数は何と何の合成関数でしょうか？
- 位置の損失関数はどのように定義されているでしょうか？
- クラス分類の損失関数はどのように定義されているでしょうか？

答え

SSDの学習

- **SSDの損失関数は何と何の合成関数でしょうか？**
  - 位置損失 (localization loss) とクラス分類の損失 (confidence loss)
- **位置の損失関数はどのように定義されているでしょうか？**
  - BoundingBoxの位置と幅に対して正解位置とのsmoothL1損失を計算
- **クラス分類の損失関数はどのように定義されているでしょうか？**
  - クラスのスコアに対して正解クラスとの交差エントロピーを計算

まとめ

- **SSDの特徴**
  - 畳込み演算だけで物体の**クラス**と**位置**を検出するCNN
  - 途中の特徴マップを利用して**様々なスケール**の物体を検出可能
  - 高精度を保ったまま**リアルタイム**物体検出が可能
- **SSDの物体検出のしくみ**
  - DefaultBoxを**様々なスケールとアスペクト比**で敷き詰めて、全DefaultBoxについてクラススコアと物体位置のずれを推論
  - **Non-maximum Suppression**で最もスコアの高い物体位置を抽出
- **SSDの学習の特徴**
  - 物体位置は**smoothL1損失**、クラススコアは**softmax損失**として2つの**合成損失関数**を用いて学習