令和元年度茨城大学工学部情報工学科卒業研究論文

物体検出モデル SSD に対する転移学習

所属　　情報工学科

著者　　結城洸太 (16T4063F)

指導教員　　新納浩幸教授

令和 2 年 2 月 5 日

令和元年度茨城大学工学部情報工学科卒業研究

物体検出モデル SSD に対する転移学習

著者

結城洸太 (16T4063F)

指導教員

新納浩幸教授

論文要旨

SSD は 2016 年に提案された物体検出のアルゴリズムである。物体検出のモデルの学習では、一般に膨大な訓練データを必要とする。そのため、訓練データを用意するのはコストがかかる。そこで、それを解消するために少量の訓練データで学習する方法を提案する。

本研究の目的は、十分な量のデータを用いて学習した 3 クラスの SSD のモデルを用いて、1 クラスを追加した 4 クラスのモデルを、少量の訓練データのみで作成することである。

これを実現する手法として、転移学習を用いた。SSD のモデルに対して、凍結、初期化の範囲を変えた三つの転移学習の方法を用いて学習した。そして、これらのモデルの精度を比較した。

その結果、モデルの精度は、十分なデータを用いて作成したモデルには及ばなかったが、同数のデータで学習したモデルに比べると高い値であった。

これに基づき、提案手法ではある程度の精度を出すことはできるが、十分な量のデータを用いた場合と同等の精度を求めるならば、別のアプローチが必要であると結論付けた。

1　序論

物体検出は、画像を入力として物体のカテゴリと画像における位置を出力する画像処理のタスクである。物体検出のモデルは一般に教師あり学習によって生成される。教師あり学習とは、訓練データと正解ラベルを基に機械学習させる方法である。教師あり学習に用いる訓練データには、データに加え正解ラベルとなるアノテーションが必要である。また、教師あり学習は、膨大な訓練データを用いて学習させることが一般的である。そのため、訓練データを用意するにはコストがかかる。

本研究では、訓練データのコストを削減することを目的に、少ない訓練データのみで物体検出モデルを作成する方法を提案する。具体的な方法としては、転移学習を利用する。転移学習は、学習済みのモデルを利用し新たなモデルを学習させる手法の一つである。本研究では、物体検出のアルゴリズムの一つである SSD(Single Shot MultiBox Detector) [1] のモデルを用いた転移学習を行う。SSD は 2016 年に提案された物体検出アルゴリズムである。物体検出アルゴリズムのうち、SSD のモデルは検出速度が特に優れ、トレードオフを最小に抑えることで検出精度も高さを保っている。本研究の目的は、SSD のモデルを転移学習させることで、もとのモデルに含まれるクラスに新たなクラスを追加した SSD のモデルを、少ない訓練データのみで作成することである。

2　関連研究

[2] における研究では、物体検出の学習に使用する訓練データの質を向上させることで、モデルの精度を高める方法を述べている。その実験において、トレーニング時間を短縮する目的で、オープンソースの物体検出モデルを使った転移学習を行っている。アーキテクチャは、Faster-RCNN、SSD、R-FCN を利用しており、それぞれの精度を比較している。Faster-RCNN と R-FCN は、位置推定とクラス分類の処理が分けられた手法である。それに対して、SSD はそれらを同時に処理する手法である。

この論文の主題はデータセットの生成についてである。結論としては、優れたデータセットを明確に定義することはできないが、訓練データにおける物体の向き、シーンの多様性、ノイズなどに関してバランスが取れていることが必要であると述べている。

また、ネットワークの観点からも考察している。それによると、RCNN 系の二つが精度が高く、SSD の精度は低いという結果となっている。これに関しては、位置推定の手法の違いに原因があると述べている。RCNN 系の場合、位置推定は処理を分けて行うため予測されるボックスは非常に正確であるのに対し、SSD で予測されるデフォルトボックスには、速度以外の利点はない。従って、RCNN 系の精度が高くなっているのである。ただし、SSD は実行速度の面では優位に立っていると述べられている。

3　物体検出モデル SSD

3.1　教師あり学習

物体検出のモデルは、教師あり学習によって作成される。教師あり学習は機械学習の手法の一つである。訓練データとして、入力データに加え正解ラベルを用いる。この正解ラベルが教師の役割りをしていることから、教師あり学習と呼ばれる。教師あり学習での学習方法は、入力データからあるルールに従って推論された出力と正解ラベルを比べることを繰り返し、それらが同じになるようにルールを調整していくというものである。これによって、未知のデータに対しても正しい分類ができるようなルールを生成する。

なお、機械学習の分類としては、教師あり学習の他に教師なし学習、強化学習がある。また、教師あり学習の代表的な手法としては回帰と分類がある。本研究で扱う物体検出のタスクは分類に当たる。分類は名前のとおり、データをクラスごとに自動分類するものである。

教師あり学習は膨大な訓練データを用いるのが一般的であり、データが多いほど学習の精度が向上する。

3.2　畳み込みニューラルネットワーク (CNN)

畳み込みニューラルネットワーク (CNN: Convolutional Neural Network) は、畳み込み演算を使ったニューラルネットワークである。CNN は、画像処理において最も頻繁に使われているモデルである。

3.2.1　ニューラルネットワーク

ニューラルネットワーク (NN: Neural Network) は、人間の脳の神経構造をもとに考案された機械学習のモデルである。ニューラルネットワークの構造は、複数の層によって構成されている。層は入力した値を、層自身が持つパラメータに従い変換し出力する。この層がいくつも重なっており、最初の入力から複数の層を経て最終的な出力が得られる。これによって複雑な推論を可能にしている。訓練データの入力をネットワークに通して得られた結果が正解ラベルと同じになるように、ネットワークのパラメータを更新していくことでモデルを調整するのが、ニューラルネットワークでの学習方法である。

なお、多層構造にしたニューラルネットワークをディープラーニングという。

3.2.2　畳み込み演算

畳み込み演算は、数学的には二つの関数の積の積分であるが、CNN においては単なる二次元行列の演算として表される。その計算方法は、入力された二次元行列に対してフィルタの二次元行列を少しずつずらしながら掛け合わせて合算し、出力である二次元行列を得るというものである。畳み込み演算における二次元行列を特徴マップという。

具体的な畳み込み演算の方法を次に示す。

入力特徴マップを 5×5 の配列 [5,5]、フィルタを 3×3 の配列 [3,3] とした場合を例に考える。これによって得られる出力特徴マップは、3×3 になる。

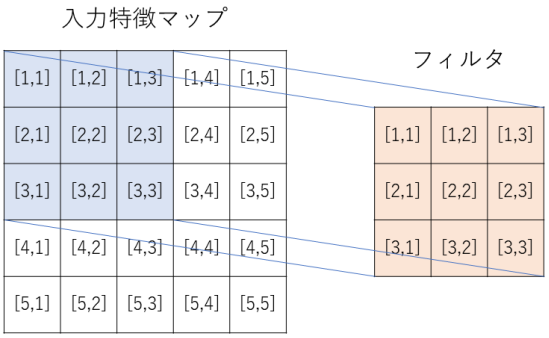

図 1　畳み込み演算 (1)

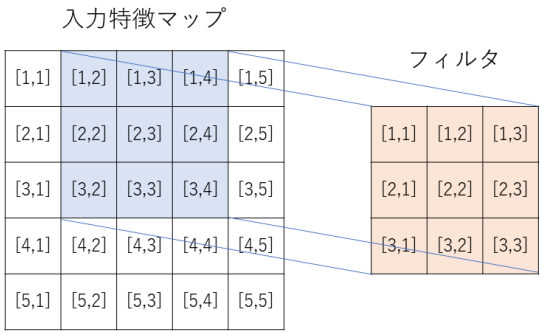

図 2　畳み込み演算 (2)

まず、図 1 に示したように、入力特徴マップの [1,1] から [3,3] の 3×3 マスとフィルタ 3×3 を見る。そして、それらの行列の積を取り、その合計を出力特徴マップの [1,1] とする。

次に、図 2 に示したように、見ている入力特徴マップを一つずらし、[1,2] から [3,4] の 3×3 マスも同様に計算し、出力特徴マップの [1,2] に結果を入れる。

このようにして、入力特徴マップをすべて見ると、3×3 の出力特徴マップが得られる。

通常の畳み込み演算は以上であるが、畳み込み演算には、いくつかのパラメータを設定することができる。特に頻繁に使用する二つのパラメータについて説明する。

一つは、パディングである。これはデフォルトでの値は 0 となっており、入力特徴マップの上下左右のマスに、パディングの値分だけ余白のタイルを作ることができる。パディング 1 で上記に示した入力特徴マップを 5×5、フィルタを 3×3 の畳み込み演算を行うと、入力特徴マップの上下左右に 1 マスずつ余白が生成され、7×7 の特徴マップとなる。つまり、入力特徴マップ 5×5 に対して、同じ大きさである 5×5 の出力特徴マップを得られる。

パディングを設けることによって生まれる利点としては、出力特徴マップが小さくならないということである。

もう一つは、ストライドである。これのデフォルトの値は 1 である。ストライドは、特徴マップを移動するマス数を表す。

ストライド 2 の場合、上記に示した入力特徴マップを 5×5、フィルタを 3×3 の畳み込み演算を行うと、入力特徴マップの [1,1] から [3,3] を見た次は右に 2 つずれて、入力特徴マップの [1,3] から [3,5] を見ることになる。更にその次は、下に 2 つずれた行の [3,1] から [5,3] となる。

ストライドを増やすことにより、特徴マップの縮小を加速することができる。

3.2.3　CNN のネットワーク構造

最後に、CNN のネットワーク構造について述べる。畳み込み演算によって得られた特徴マップは、次の層への入力となる。このように、CNN も NN 同様に複数の層が重なった構造となっている。CNN ではフィルタを更新するパラメータとして扱い、それをニューラルネットワークと同じように調整することでモデルを学習している。すべてデータを結合して値を変換する全結合層が大域的なパターンを学習するのに対し、畳み込み層は局所的なパターンを学習することができる。

さらに、局所的なパターンを学習した特徴マップをさらに畳み込むことで、より大きなパターンを学習できる。これによって畳み込み層は、視覚的な画像の特徴を抽象化することができる。

3.3 物体検出

物体検出は画像処理のタスクの一つである。

画像処理のタスクのうち、基本的なものとして画像認識がある。画像認識は、画像を入力し、出力としてクラス、つまり画像の種類を得るというタスクである。

それに対して物体検出では、入力は画像認識と同様に画像であるが、出力は物体の位置とクラスの二つである。画像認識は、入力に対して出力が一つに定まるが、物体検出では出力である位置とクラスの組が複数存在する場合がある。これは、一つの画像の中に複数の物体が存在するとき、それぞれ別の物体として出力するためである。物体検出の出力であるクラスは、具体的にはクラス確信度で出力される。クラス確信度は、検出された物体がそのクラスであることにどのくらい確信があるのかを最小値 0、最大値 1 で表したものである。　また、もう一つの出力である物体の位置は、バウンディングボックスで表される。バウンディングボックスは、中心座標 (x, y)、幅 w、高さ h で表された長方形のボックスである。

画像認識や物体検出といった画像処理のタスクでは、一般に CNN を使った教師あり学習が用いられる。(節 3.1、節 3.2 を参照)

3.4 SSD

SSD(Single Shot MultiBox Detector) は、物体検出の手法の一つである。物体の領域候補を探してからクラス分類をする R-CNN とは違い、SSD での物体の検出方法は、ボックスをあらかじめ用意することで位置推定とクラス分類を同時に実行するものである。これにより、他の手法と比較して優れた実行速度を実現している。

3.4.1 SSD の仕組み

SSD の位置推定の方法について説明する。SSD における位置、つまりバウンディングボックスは、デフォルトボックスとオフセットの組み合わせによって表される。デフォルトボックスは以下の式に従い、画像に対して 8732 個生成される。

k 番目の特徴マップのスケール s_k

特徴マップの数 m

$s_{max} = 0.9, s_{min} = 0.2$(入力画像のスケールを 1.0)

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}(k - 1), k \in [1, m]$$

アスペクト比 a_r

$$a_r = 1, 2, 3, \frac{1}{2}, \frac{1}{3}$$

$$\text{幅 } (w_k^a = s_k\sqrt{a_r}), \text{高さ } (w_k\ a = s_k/\sqrt{a_r})$$

アスペクト比が 1 の場合は以下のスケールを追加

$$s'_k = \sqrt{s_k s_{k+1}}$$

各デフォルトボックスの中心座標 k 番目の特徴マップのサイズ f_k

$$(\frac{i+0.5}{|f_k|}, \frac{j+0.5}{|f_k|})$$
$$i, j \in [0, |f_k|)$$

SSD では、複数の異なる分割数の特徴マップに対してデフォルトボックスを作成する。分割数が変わると特徴マップのセルの大きさが変わり、その大きさに応じてボックスのスケールを定める。これによりスケールの違うボックスが作成される。スケールの違いにより、様々な大きさの物体に対応し検出することができる。

また、ボックスは一つのセルにおいても複数個生成される。スケールは同じだがアスペクト比の異なるボックスを生成することで、様々な形の物体に対応している。

これによって生成された 8732 個のデフォルトボックス $priors(cx, cy, w, h)$ ごとに、オフセット $loc\_p(\Delta cx, \Delta cy, \Delta w, \Delta h)$ とクラス確信度 $conf\_p(c0, c1, c2, \cdots, cn)$ を推論する。そして、クラス確信度が閾値を超えたボックスを、バウンディングボックス $(cx+\Delta cx, cy+\Delta cy, w+\Delta w, h+\Delta h)$ として表示する。

ただし、この方法では多くのデフォルトボックスが閾値を超え、一つ物体に対して複数のバウンディングボックスが集中してしまう。その問題を解消する方法として、NMS(Non Maximum Suppression) がある。これは、表示されたバウンディングボックス同士の重複面積が閾値を超えた場合、クラス確信度が最大のボックス以外を削除するという手法である。これにより、一つの物体に対して表示されるバウンディングボックスは一つに定まる。

3.4.2 SSD のネットワーク

SSD のネットワークは CNN で構成されている。図 3 は、SSD のネットワーク構造を図式化して示したものである。SSD のネットワークは、ベースネット層、追加ネットワーク層、位置推定層、クラス推定層に分けることができる。

ベースネット層は、vgg16 などの画像認識のモデルから最終層の全結合層を除いたネットワークとなっている。この部分は、既存のモデルを利用することになる。特徴マップは、入力として 300×300 の画像が入力され、複数の CNN 層への入出力を経て、最終的に 38×38 の特徴マップを出力する。

追加ネットワーク層は、SSD 独自のネットワークで、畳み込みにより 38×38 の特徴マップを 1×1 まで小さくしていく。

最後に、位置推定層、クラス推定層は、どちらもパディング 1、3×3 のフィルタの畳み込み層 6 層からなり、特徴マップの大きさを一定に保持したまま、ネットワーク最終層までつながる。この二つの層は、ベースネット層、または追加ネットワーク層から分岐している。異なるサイズの特徴マップから分岐することで、様々な物体の大きさに対応したボックスの作成を実現している。

3.4.3 SSD における学習の処理概要

SSD における学習では以下の処理が繰り返される。

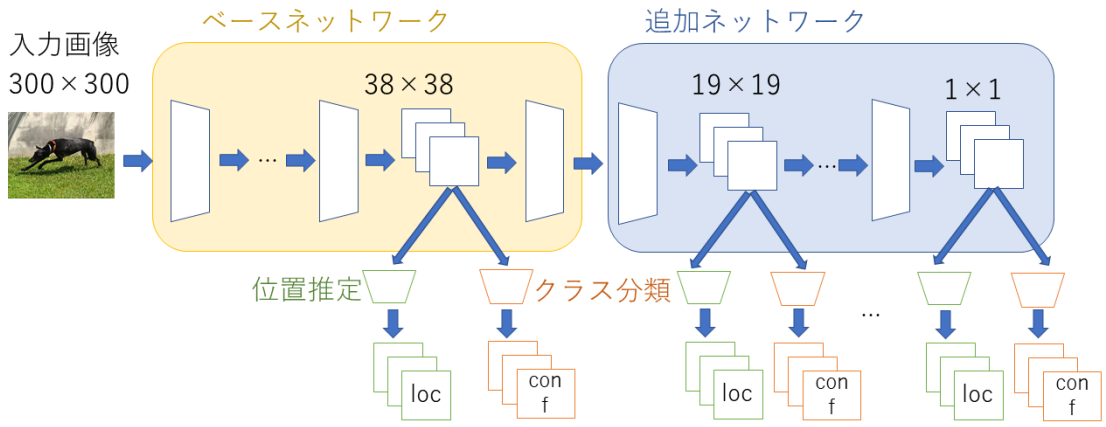

図 3　SSD のネットワーク構造

1. 訓練データの読み込み
2. ネットワークでの推論
3. 正解座標とボックスのマッチング
4. 損失関数の計算

ステップ 3 を除いては、基本的な機械学習の流れと同じとなっている。

ステップ 1 では、訓練データから、画像 $images$ とアノテーション $target$ のテンソルを作成する。$image$ は 300×300 にリサイズされ、(C, H, W) の画像データ形式に変更される。

$images = (\text{バッチサイズ} : B, Channel : 3, Height : 300, Width : 300)$

$targets = (\text{バッチサイズ} : B, \text{物体数} : O, (xmin, ymin, xmax, ymax, label\_ind) : 5)$

ステップ 2 では、ネットワークの推論を行う。ステップ 1 で作成した $images$ のデータからネットワークで推論を行い、推論結果 out を作成する。推論結果 out は、以下で構成されている。

オフセット $loc\_p : (\Delta cx, \Delta cy, \Delta w, \Delta h)$
クラス確信度 $conf\_p : (c0, c1, c2, \cdots, cn)$
ボックス座標 $priors : (cx, cy, w, h)$

ステップ 3 では、ボックスのマッチング処理を行う。ステップ 1 で作成したアノテーション $targets$ から、正解座標 $truths$ と正解ラベル $labels$ を作成する。正解座標 $truths$ と正解ラベル $labels$、ステップ 2 で作成した推論結果 out を用いて、マッチング処理を行う。

マッチング処理は 8732 個のデフォルトボックスに対して行われ、ボックス座標 $priors$ とオフセット loc_p から求められるデフォルトボックスの面積と正解座標 $truths$ の面積を IoU でマッ

チングする。IoU は、二つの領域がどれくらい重なっているかを表す指標である。IoU については、節 6.1.2 で詳しく説明する。マッチングにおける IoU の閾値は 0.5 で、それを超えた場合マッチング成功となる。

マッチング成功 (Positive) の場合、正解ラベル $conf\_t$：ラベル番号 $[1, n]$、正解オフセット loc_t : $(\Delta cx, \Delta cy, \Delta w, \Delta h)$ を作成する。

マッチング失敗 (Negative) の場合、正解ラベル $conf\_t$: ラベル番号 0 を作成する。

ステップ 4 では、損失関数の計算を行う。

SSD における損失関数は、位置推定の損失関数とクラス確信度の損失関数を組み合わせた、以下の式で表される。

損失関数 $L(x, c, l, g)$

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

ただし、

$$\text{位置推定の損失関数} : (localazationloss)L_{loc}(x, l, g)$$

$$\text{クラス確信度の損失関数} : (confidenceloss)L_{conf}(x, c)$$

$$\text{位置推定とクラス分類の重要度を制御するパラメータ} : \alpha(\text{ここでは}\alpha = 1)$$

マッチングに成功した $Positive$ のボックスの数 N (N が 0 の場合、例外として損失関数は 0)

また、位置推定の損失関数 $L_{loc}(x, l, g)$ は、以下の式で表される。

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{(cx,cy,w,h)} x_{ij}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m) \tag{1}$$

ただし、

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

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

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

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

$$smooth_{L1}(x) = \begin{cases} -0.5x^2, & |x| < 1 \\ |x| - 0.5, & |x| \geq 1 \end{cases}$$

式 1 において、i はデフォルトボックス 8732 個を区別し、j は画像中の物体の正解座標を区別している。l_i^m は推論結果のオフセット、$\hat{g}_j^m$ は正解座標のオフセットを表す。x_{ij}^k は i と j がマッチング成功なら 1、それ以外は 0 となる。$\hat{g}_j^{cx}$ は、正解座標 g_j^{cx} とデフォルトボックス座標 d_i^{cx} のオフセットを、d_i^w で正規化している。$\hat{g}_j^w, \hat{g}_j^h$ は、ボックス幅 d_i^w と高さ d_i^h で正規化され、log スケールになっている。

位置推定の損失関数は、$Negative$ の場合 x_{ij}^k が 0 となるので、$Positive$ のボックスが計算対象であり、$smooth_{L1}$ で位置推定の誤差を計算する。

クラス確信度の損失関数は $L_{conf}(x,c)$ は、以下の式で表される。

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} \log\left(\hat{c}_i^p\right) - \sum_{i \in Neg}^{N} \log\left(\hat{c}_i^0\right) \tag{2}$$

$$where \hat{c}_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)}$$

式 2 において i はデフォルトボックス 8732 個を区別し、j は画像中の物体の正解座標を区別している。x_{ij}^k は i と j がマッチング成功なら 1、それ以外は 0 となる。p はクラスのラベル番号で、1 項目はデータセットの 1 から n のクラス番号、2 項目はバックグラウンドのクラス番号 0 になる。$\hat{c}_i^p$ は $softmax$ 関数で正規化している。

$Positive$ ボックス (1 項目) と $Negative$ ボックス (2 項目) が計算対象であり、その合計が損失関数となる。クラス分類において一般的なクロスエントロピーで、クラス確信度 $conf\_p$ と正解ラベル $conf\_t$ の誤差を計算している。

なお、損失関数の計算に使う $Negative$ のボックスは、クラス確信度の上位 30 個のみを使用する。これにより、正解と間違いやすい不正解データを学習することができ、精度を向上させる効果がある。

3.4.4 データ入出力

SSD の学習、推論におけるデータの入出力の方法を示す。本実験で使用した SSD のプログラムは、6.2 節で説明するオープンソースのプログラムを利用した。

学習に利用するデータセットは画像デファイル (jpg) とアノテーションファイル (xml) の組からなる。

アノテーションファイルは XML 方式のデータで、その中身は、対応する画像データのファイル名、正解ボックスのデータが木構造になっている。アノテーションファイルの読み込みは、xml.etree.ElementTree モジュールを使用して行われる。

画像データの読み込みは、cv2 モジュールを使用して行われる。300 × 300 の大きさにリサイズなどし、ネットワークに入力可能な形式に直す。

推論での入力は、画像ファイル (jpg) である。この読み込み方法は学習時と同様である。推論では、求めた座標とクラス確信度のデータから、画像内にバウンディングボックスを出力する。画像のプロットは、matplotlib モジュールを使用して行われる。任意の閾値以上の確信度のボックスを選択し、それらをもとの画像に重ねたものを出力する。

4 転移学習

転移学習とは、学習されたモデルを利用することで少量のデータで学習させる方法である。画像認識などの機械学習のモデルを高い精度で作成するには、膨大な訓練データと学習が必要であり、コストがかかる。そこで、すでに学習されたデータを利用することで学習コストを削減しようと考えられた手法の一つが転移学習である。

基本的な転移学習の方法は、ネットワーク最終層以外のパラメータを凍結し、最終層のパラメータを初期化した上で、新たな訓練データを用いて学習する、というものである。パラメータを凍

結することで、元のモデルの精度の高い特徴抽出を残したまま、最終層で新たなデータにおけるクラス分類を学習する。転移学習のメリットとしては、少量のデータで高い精度のモデルが作成できること、パラメータを凍結する分パラメータを更新する計算が少なくなり、学習時間が短くなることが挙げられる。

5　SSD に対する転移学習

5.1　SSD に対する転移学習の方法

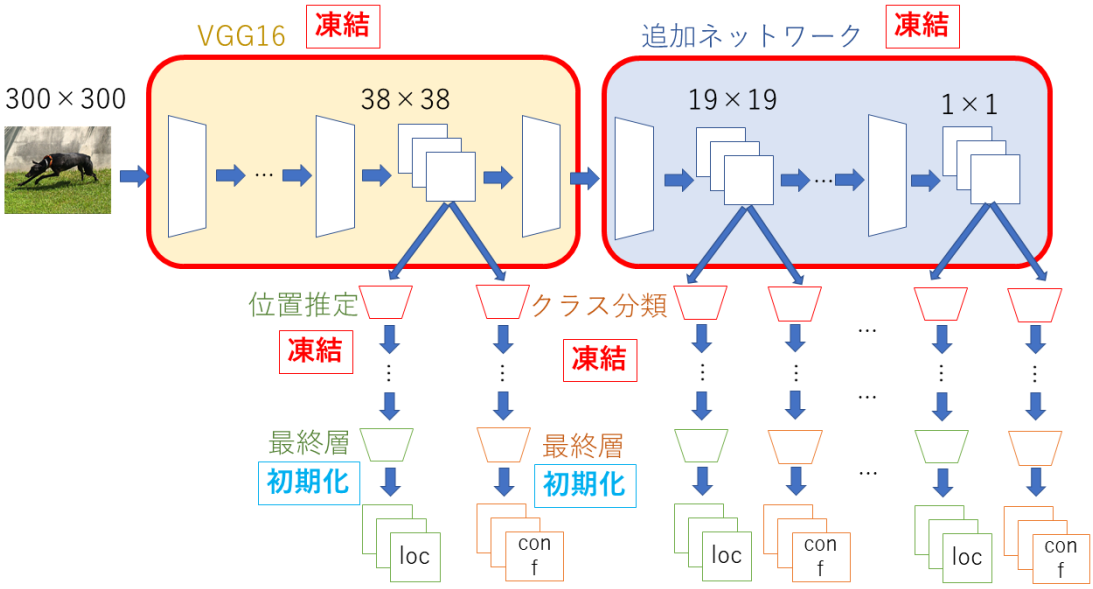

図 4　転移学習：方法 1

一般的な転移学習では、最終層以外のパラメータを凍結させ、最終層である全結合層のパラメータを初期化し学習させる。SSD の場合、最終層は全結合層ではなく、畳み込み層となっている。本実験では、凍結する範囲を変えた三種類の方法を用いて転移学習を行い比較する。

SSD のモデルのネットワークは、ベースネットワーク層、追加ネットワーク層、位置推定層、クラス分類層に分けられる。

一つ目の方法 (図 4) は、ベースネットワーク層、追加ネットワーク層の全てと、位置推定層、クラス分類層の最終層以外を凍結させ、位置推定層、クラス分類層の最終層を初期化させるものである。これは、ネットワークの最終層を凍結させるという一般的な転移学習の凍結方法である。

二つ目の方法 (図 5) は、ベースネットワーク層、追加ネットワーク層を凍結させ、位置推定層、クラス分類層を初期化させるものである。これは、ネットワークを大まかな部分で見たときの最終層に当たる位置推定層とクラス分類層部分を初期化させたものである。位置推定層とクラス分類層はどちらも CNN6 層で構成されており、それをすべて初期化している。

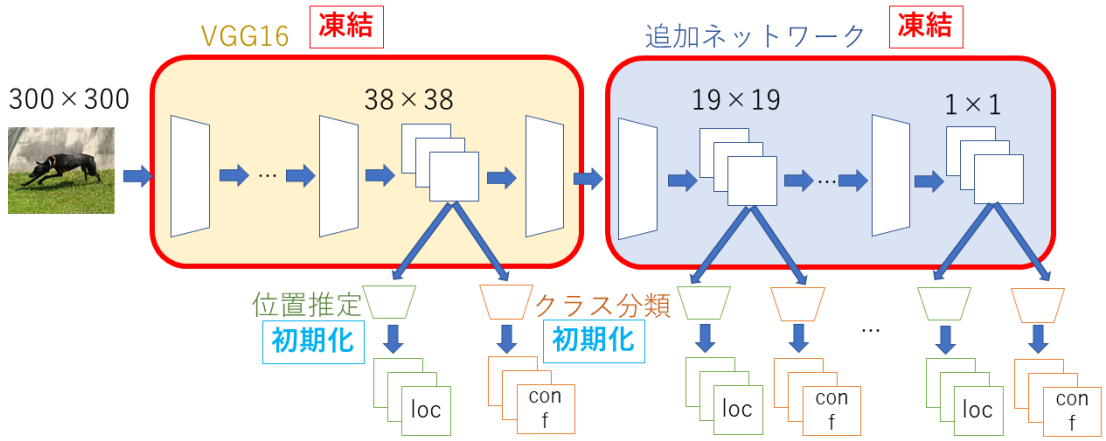

図 5　転移学習：方法 2

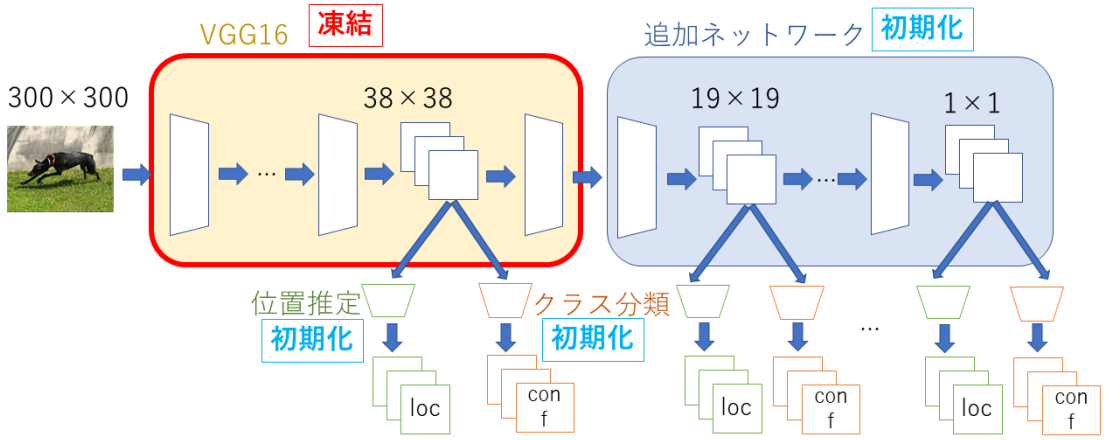

図 6　転移学習：方法 3

三つ目の方法は (図 6)、ベースネットワーク層を凍結させ、追加ネットワーク層、位置推定層、クラス分類層を初期化させるものである。この方法は、三つの方法のなかで最も凍結範囲が狭い。この方法では、もともと強い特徴抽出のネットワークであるベースネットワークのみを凍結させることで、方法 1、2 とどのような違いが現れるかを見る。

5.2　実装方法

SSD のモデルに転移学習を実装する手順を説明する。6.2 節に示す SSD のプログラムでは、以下に示すのが学習用のプログラムである。

train.py

これは、訓練データを読み込んで学習を回し、モデルを作成するプログラムとなっている。これを転移学習ができるように書き換える。

転移学習の実装は、PyTorch のチュートリアルとして以下のリンクに公開されている”TRANSFER LEARNING FOR COMPUTER VISION TUTORIAL”を参考にした。

https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

転移学習を実装するには、モデルを凍結する必要がある。モデルを凍結するには、パラメータの一つである requires_grad を False に設定する。例えば、ssd_net モデルの vgg 層を凍結したい場合は、次のようにする。

```
for param in ssd_net.vgg.parameters():
param.requires_glad = False
```

また、層の途中までを凍結したい場合は、次のようにすればよい。

```
i = 0
for param in ssd_net.vgg.parameters():
if i>= 10: #10 層目まで
break
param.requires_glad = False
i+=1
```

なお、ベースネットワーク層は、プログラム上 vgg 層と L2Norm 層の二つとなっており、ベースネットワーク層を凍結したい場合はともに凍結を行う。

パラメータの初期化は SSD のプログラムに関数として既に備わっている。例えば、ssd_net モデルの extras 層を初期化したい場合は、次のようにする。

```
ssd_net.extras.apply(weights_init)
```

これにより、extras 層のすべて層のパラメータを初期化できる。

また、層の一部を初期化したい場合は、次のようにすることができる。

```
ssd_net.extras[5].apply(weights_init)   #5 層目のみ
```

以上により、モデルの凍結と初期化ができる。

凍結、初期化の方法が分かった上で、SSD の転移学習のプログラムを実装する方法を説明する。

まず、転移学習をするには学習済みのモデルを読み込む必要がある。これは、train.py の args の resume に指定することで読み込むことができる。本来このパラメータは、学習を再開するときに、途中まで学習したモデルを指定するものである。例えば、VOC_3class.pth のモデルを読み込みたい場合は、args のパラメータを次のようにする。

```
’resume’:’VOC_3class.pth’
```

これにより、指定したモデルのパラメータをロードすることができる。

次に、モデルを凍結、初期化する。SSD のプログラムには、resume のパラメータ指定しなかった場合に、パラメータ初期化する部分がある。

```
if not args.resume:
print('Initializing weights...')
# initialize newly added layers' weights with xavier method
ssd_net.extras.apply(weights_init)
ssd_net.loc.apply(weights_init)
ssd_net.conf.apply(weights_init)
```

今回、resume を指定したので、この if 文は False となる。よって、次のように else 文を加え、モデルの凍結、初期化を行う。

```
if not args.resume:
print('Initializing weights...')
# initialize newly added layers' weights with xavier method
ssd_net.extras.apply(weights_init)
ssd_net.loc.apply(weights_init)
ssd_net.conf.apply(weights_init)
else:
#位置推定、クラス分類を初期化
ssd_net.loc.apply(weights_init)
ssd_net.conf.apply(weights_init)
#ベース、エクストラネットを凍結
for param in ssd_net.vgg.parameters():
param.requires_glad = False
for param in ssd_net.L2Norm.parameters():
param.requires_glad = False
for param in ssd_net.extras.parameters():
param.requires_glad = False
```

これで、SSD における転移学習が可能となる。

6 実験

6.1 実験設定

6.1.1 実験の流れ

(car, person, motorbike) の 3 クラスを学習させた SSD のモデルを作成する。3 クラスの訓練データを用いて 120,000 エポック学習させる。次に、3 クラスの SSD モデルに、元の 3 クラス + 追加の 1 クラス (car, person, motorbike, bicycle) の少量の訓練データを用いて転移学習させ、

3 + 1 クラスの SSD モデルを作成する。

節 5 で示した方法 1、方法 2、方法 3 について同様に転移学習を行い、エポック数は、12,000、120,000 の精度を計測する。また、方法 2 と方法 3 に関しては、追加学習によるさらなる精度向上の余地が見られたため、追加で 240,000 エポックのモデルも作成した。さらに、比較のため、十分な量のデータを用いて学習させた 4 クラスのモデルと、転移学習に用いた少量のデータで学習させた 4 クラスのモデルも作成する。

6.1.2 評価指標

物体検出では、出力ボックスに実際のボックスとの多少の誤差が生じるため、どのくらいの誤差であれば、正解か不正解かが曖昧である。物体検出の正誤評価は、という性質上、一般的に IoU(Intersection over Union) を用いて正誤判定が行われる。

IoU は、実際のボックスと検出したボックスがどれくらい重なっているかを表す指標であるり、以下の式で表される。

$$IoU = \frac{AreaofOverlap}{AreaofUnion}$$

$$AreaofOverlap = 2\ \text{つのボックスの共通部分}$$

$$AreaofUnion = 2\ \text{つのボックスの全体部分}$$

IoU が 1 のとき、二つのボックスは完全に重なっている。また、IoU が 0 のとき、二つのボックスに重なってる部分はない。SSD では IoU の閾値を 0.5 として、それを超えたものを正解 (positive) としている。

物体検出の精度の評価は mAP(mean Average Precision) といった評価値が用いられる。

次に mAP の導出について説明する。導出方法については、[3] にある方法を用いた。

まず、mAP の導出における予備知識として、TP(True Positive),FP(False Positive) ,FN(False Negative),TN(True Negative) の指標を示す。

TP (True Positive)　実際にボックスに対して、検出されたボックスが存在し、重なっている。($IoU > 0.5$)。

FP(False Positive)　実際にボックスに対して、検出されたボックスが存在するが、外れた位置にある。($IoU < 0.5 \bigcap IoU \neq 0$)

FN(False Negative)　実際はボックスがあるが、検出したボックスはない。

TN(True Negative)　検出したボックスも、実際のボックスもない。

次に、$Precision$ と $Recall$ を示す。$Precision$ が示すのは、検出した物体がどれくらい正しいのかであり、$Recall$ が示すのは、検出すべき実際の物体がどれくらい検出できたかである。これらは、TP、FP、FN によって、次のように表される。

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

以上を踏まえたうえで、mAP の導出する。まず、SSD モデルで得られた結果を、表 1 のように、クラス別に確信度の高いボックス順に並べる。そして、表 1 にあるように上から順にボックスを見ていき、検出されたボックスを現在のボックスまでとしたうえでの *Precision* と *Recall* を求める。

このとき、*Precision* と *Recall* は次の式で表される。

$$Precision = \frac{\text{現在まで正解したボックス数}}{\text{現在まで見た検出ボックス数}}$$

$$Recall = \frac{\text{現在まで正解したボックス数}}{\text{すべての実際のボックス数}}$$

表 1　*Precision* と *Recall* の導出例

順位	クラス確信度 (%)	正誤	*Precision*	*Recall*
1	100	正解	$1/1 = 1$	$1/10 = 0.1$
2	99	正解	$2/2 = 1$	$2/10 = 0.2$
3	96	不正解	$2/3 = 0.67$	$2/10 = 0.2$
4	92	不正解	$2/4 = 0.5$	$2/10 = 0.2$
5	89	不正解	$2/5 = 0.4$	$2/10 = 0.2$
6	88	正解	$3/6 = 0.5$	$3/10 = 0.3$
7	80	正解	$4/7 = 0.57$	$4/10 = 0.4$
8	77	正解	$5/8 = 0.63$	$5/10 = 0.5$
⋮	⋮	⋮	⋮	⋮

求めた *Precision* と *Recall* をもとに、縦軸を *Precision*、横軸を *Recall* とした、PR 曲線 (Precision-Recall Curve) を描く。(図 7) すると、*AP* は次の式で表される。ただし、p は *Precision* の変数、r は *Recall* の変数とする。

$$AP = \int_0^1 p(r)dr$$

なお、これは PR 曲線の下側の面積を表している。次に、積分を簡単にするため、グラフを以下の式で均す。(図 8)

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r})$$

最後に積分を求めるが、簡略化し、$r = (0, 0.1, \cdots, 1)$ の計 11 箇所の点を取る。(図**??**) そして、その点についてのみ計算を行う。これにより、縦 × 横の掛け算で簡単に求めることができる。

すると、以下の式で *AP* が求められる。

$$AP = \frac{1}{11} \sum_{r\in(0,0.1,\cdots,1)} p_{interp}(r)$$

つまり、

$$AP = \frac{1}{11} * (p_{interp}(0) + p_{interp}(0.1) + \cdots + p_{interp}(1.0))$$

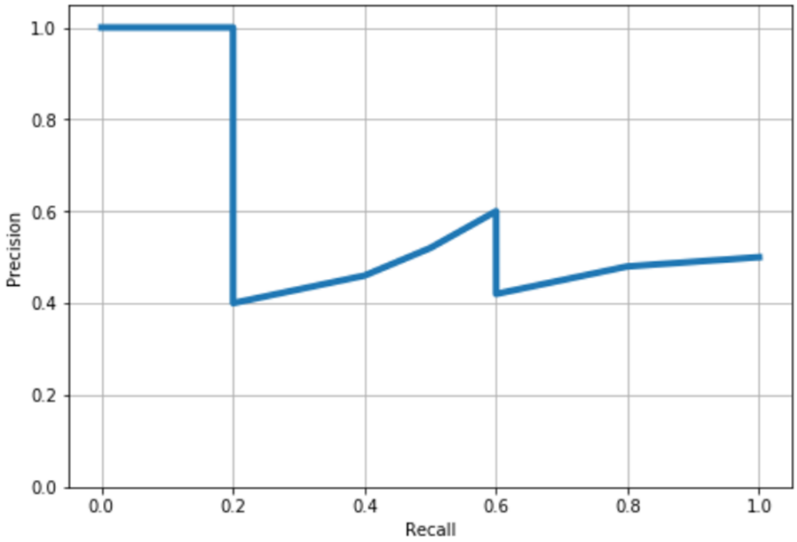

図 7　Precision/Recall 曲線

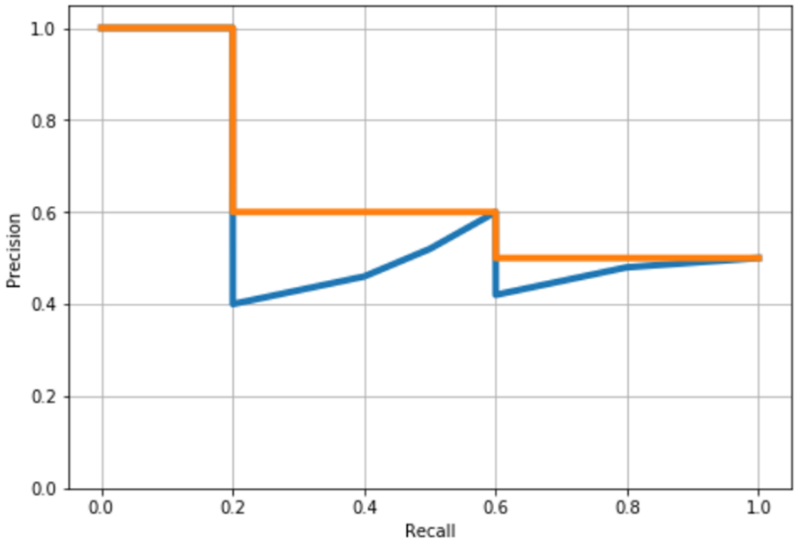

図 8　PR 曲線の簡略化

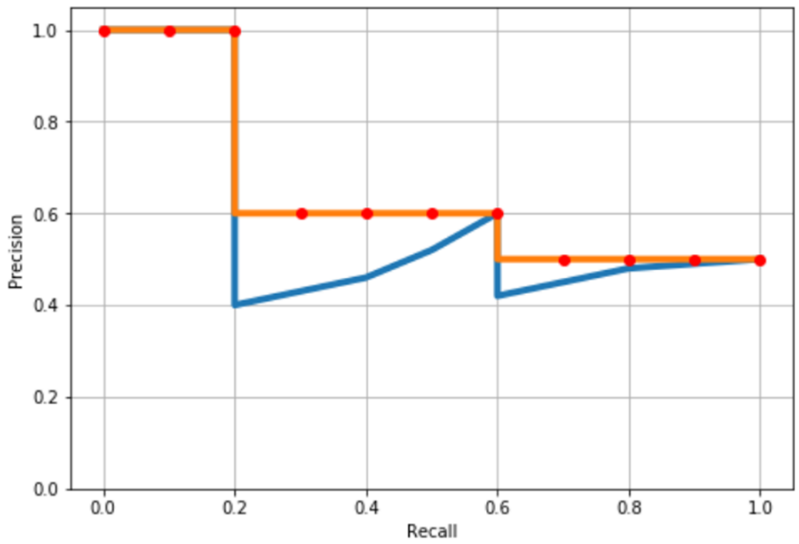

図 9　AP の導出

これをすべてのクラスに対して導出し、全てのクラスの AP の平均が mAP となる。本実験ではこの mAP を用いてモデルを評価する。

補足として、mAP は正誤判定に用いる IoU の閾値によって、同じモデルに対しても異なる値を示す。これは、IoU の閾値が高くなると不正解となる場合が増えてしまうため、mAP は低くなるからである。よって、どの閾値の IoU を用いて算出した mAP であるかを明記する必要がある。これは一般に、IoU 閾値が 0.5 の場合、$mAP_{0.5}$ のように示したり、@0.5 と書き加えたりすることで表す。

IoU の閾値は基本的にどんな値でも設定できるが、0.5 や 0.7 に設定することが多い。また、0.5 から 0.95 までの mAP を 0.05 間隔で求め、それを平均したものが一般的な指標として用いられている。

本実験では、$mAP_{0.5}$ と $mAP_{0.7}$ を指標として使用した。

6.2　実験データ

今回使用したデータセットは、PASCAL Visual Object Classes(VOC) 2007, 2012 である。このデータは、画像ファイル (.jpg) と位置、クラスを持つアノテーションファイル (.xml) の組で構成された、物体検出用のデータセットである。以下の URL からダウンロードできる。

```
http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtrainval_06-Nov-2007.tar
http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtest_06-Nov-2007.tar
http://host.robots.ox.ac.uk/pascal/VOC/voc2012/VOCtrainval_11-May-2012.tar
```

VOC は計 20 種類のクラスのデータからなり、cat, dog といった動物クラス、car, ship といっ

た乗り物クラス、chair, tvmonitor といった家具クラスを含む。

VOC データセットは、図 10 に示すファイル構造になっている。

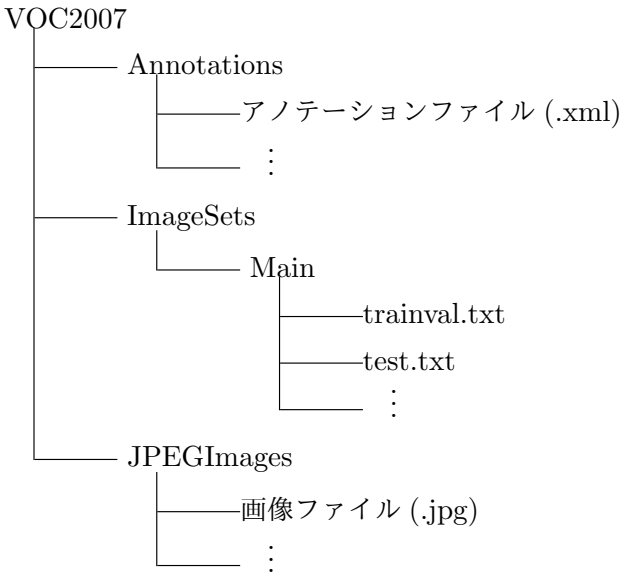

図 10　VOC データセットの構造

ImageSets の Main ファイルには、アノテーションファイルが一覧をテキスト形式で書かれたファイルがある。これはデータを読み込む際に参照する。

また、アノテーションファイルは、図 11 に示す構造になっている。

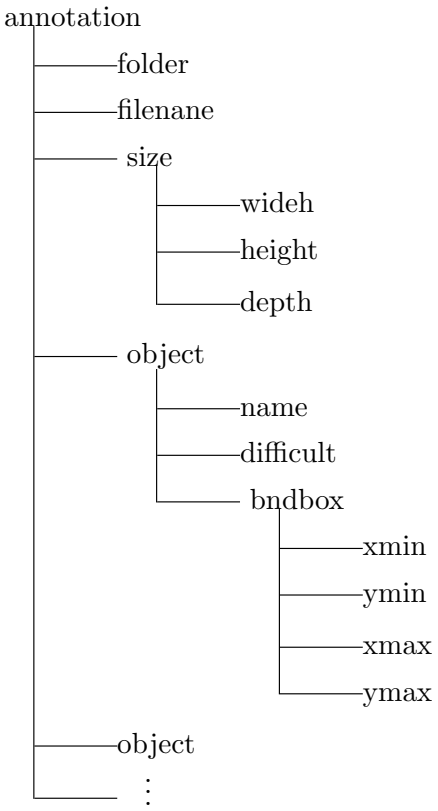

図 11　アノテーションファイルの構造

folder の属性値は、VOC2007 か VOC2012 になる。filename の属性値は、対になる画像ファイル名である。size 属性は、画像のサイズを示す。object 属性は、正解ボックスの情報を表す。name 属性がクラス名、bndbox 属性がボックスの位置を表す。

その中から次のデータを取り出した。

- 訓練データ：3 クラス (car, person, motorbike) のデータ計 6,909 個。
- 転移学習に用いる少量の訓練データ：4 クラス (car, person, motorbike, bicycle) のデータ計 200 個。
- 比較用の 4 クラスのモデルに用いるデータ：4 クラス (car, person, motorbike, bicycle) のデータ計 7,423 個。
- テストデータ (訓練データと被り無し)：4 クラス (car, person, motorbike, bicycle) のデータ計 2,390 個

データの取り出しは、xml.etree.ElementTree モジュールを使用し、アノテーションファイルを操作する。まず、アノテーションファイルを一つずつ読み込んでいく。object 属性の name 属性値を読み取り、取り出したいクラスでない場合は object を削除する。すべての object が削除された場合は、ファイルごと削除し、対になる画像データも削除する。これによって、指定したクラスのみのデータセットを作成する。

次に、取り出したデータのうち、difficult 属性値が 1 の object しかないファイルを削除する。difficult 属性値が 1 のオブジェクトは、学習には使用されないようになっている。そのため、このファイルを読み込む場合オブジェクトの数が 0 個になってしまうため、エラーを起こす。それを防ぐためにこのファイルを削除しておく。

最後に、ImageSets/Main 内のアノテーションファイル一覧のテキストを、削除した内容に更新する。以上がデータセットの生成方法である。

ベースネットワークとして、vgg16 を使用した。vgg16 は、画像認識の学習済みモデルで、20,000 クラスの分類ができる、高性能なモデルである。ネットワーク構造は、畳み込み 13 層と全結合 3 層の計 16 層で構成されたニューラルネットワークである。

SSD モデルのソースコードは以下のリンク先から取得したものを利用した。このコードは pytorch で書かれており、Jupyter Notebook で作成されたノートブック形式のプログラムである。

```
https://github.com/amdegroot/ssd.pytorch
Copyright (c) 2017 Max deGroot, Ellis Brown
Released under the MIT license
```

6.3 実験結果

実験結果を表 2 と表 3 に示す。それぞれの表にある@0.5 と@0.7 は、正誤判定に用いた IoU の閾値閾値を示している。@0.5 の場合、IoU が 0.5 以上の場合を正解と判定して導出した評価値となっている。

各表の car, notorbike, person, bicycle の項目は、クラス別の AP を示している。すべてのクラ

スの AP を平均したものが mAP となる。

表 2 に示した IoU の閾値閾値が 0.5 の実験では、方法 3 のモデルが最大の mAP となった。追加クラスである bicycle クラスも、方法 3 のモデルが最大であった。表 3 に示した IoU の閾値閾値が 0.7 の実験でも、ほぼ同様の結果が得られた。

表 2　実験結果 mAP@0.5

メソッド (epoch)	Train 画像数	$mAP_{0.5}$	car	motorbike	person	bicycle
4 クラス 12k	7,423	80.3	85.2	77.4	76.8	81.0
4 クラス 120k	7,423	83.7	87.0	86.7	74.5	86.7
4 クラス 12k	200	49.4	60.2	41.3	19.1	22.6
4 クラス 120k	200	51.3	60.0	47.9	47.9	49.5
方法 1 12k	200	62.2	87.3	83.6	74.3	3.4
方法 1 120k	200	62.7	87.3	83.6	74.4	5.4
方法 2 12k	200	67.2	86.0	81.4	78.1	23.3
方法 2 120k	200	69.6	85.5	80.7	77.3	34.9
方法 2 240k	200	68.7	85.8	80.7	77.3	34.9
方法 3 12k	200	68.6	85.3	81.6	77.8	30.8
方法 3 120k	200	75.7	85.3	76.9	77.3	63.5
方法 3 240k	200	75.8	85.2	77.4	76.8	63.8

表 3　実験結果 mAP@0.7

メソッド (epoch)	Train 画像数	$mAP_{0.7}$	car	motorbike	person	bicycle
4 クラス 12k	7,423	50.3	63.1	46.7	38.1	53.1
4 クラス 120k	7,423	65.2	70.5	69.5	50.9	69.8
4 クラス 12k	200	27.8	48.8	20.8	19.1	22.6
4 クラス 120k	200	31.8	49.7	27.0	23.0	27.3
方法 1 12k	200	47.8	70.4	66.4	51.1	3.2
方法 1 120k	200	48.3	70.4	66.4	51.1	5.4
方法 2 12k	200	45.0	69.7	55.5	44.9	10.1
方法 2 120k	200	45.8	69.1	55.5	44.9	12.5
方法 2 240k	200	45.2	69.3	54.3	45.2	12.1
方法 3 12k	200	41.7	68.6	48.9	44.4	5.1
方法 3 120k	200	46.8	68.8	50.1	44.0	24.2
方法 3 240k	200	45.6	68.5	44.8	43.7	25.6

また、25 ページ以降の付録資料、図 12 から図 35 に示すのは、各モデルの PR 曲線である。方法と学習数、IoU 閾値を分けてにグラフ化し、それぞれの図には 4 クラスの PR 曲線を重ねてプロットしている。縦軸の Precision は、検出した物体がどれくらい当たっているかを意味し、横軸

の Recall は、検出すべき物体をどれくらい検出できたかを意味する。なお、曲線の下面積は AP を表している。

7　考察

7.1　転移学習の凍結範囲

本実験では、転移学習の凍結範囲を変えた三つのパターンを比較している。

方法 1 は節 5 にあるとおり、位置推定層とクラス分類層の最終層 1 層ずつを初期化し、それ以外を凍結する方法である。方法 1 の結果は、追加クラスである bicycle クラスの AP の値が 3 から 5 程度であり、低い値である。図 30 から図 27 にある方法 1 における全ての bicycle クラスの PR 曲線からは、Precision と Recall がどちらも低いことが読み取れる。それに対して他のクラスの AP は、@05 であればどれも 80 前後と高い値である。つまり、bicycle の物体がほとんど検出されず、転移学習前のモデルに近いモデルであるといえる。これは、位置推定層、クラス分類層での学習が不十分であったと考えられる。方法 1 は物体検出の転移学習の方法としては不適であるといえる。画像認識における全結合層は、SSD において位置推定 6 層とクラス分類 6 層で表されていると考えられる。

次に、方法 2 と方法 3 の比較をする。方法 2 は、ベースネットと追加ネットワークを凍結し、位置推定とクラス分類を初期化するもので、方法 3 は、ベースネットを凍結し、それ以外初期化するものであった (11 章参照)。mAP においては、方法 3 の方が方法 2 に比べて高い値を示している。追加クラスである bicycle クラスにおいても、方法 3 が高い値を示している。これは、更新部分の大きい方法 3 において、より学習が行われたからであると考えられる。逆に方法 2 は、追加クラス以外のクラスの精度が方法 3 より高い。これは、更新部分の小さい方法 2 では、元の 3 クラスのモデルの形がより大きく現れたからだと考えられる。方法 2、3 は、十分なデータを用いて学習した 4 クラスのモデルに比べると精度は劣るが、同数のデータを用いて学習させたものに比べると、精度は高い。従って、ともに転移学習は成功したといえる。

全体的な結果としては、凍結範囲を減らし学習する層を増やすほど精度の向上が見られた。しかしこれは、転移学習の基本的な考えに反する。このような結果になった理由としては、転移学習前後のクラスに原因があると考えられる。一般的な転移学習においては、転移学習後のクラスは元のモデルのクラスに含まれず、クラスの偏りは生じない。しかし、本実験では転移学習後の 4 クラスのうち 3 クラスが、転移学習前の学習済みモデルに含まれるクラスである。よって、4 クラスのうち 3 クラスの特徴が色濃く残ったことにより、追加クラスの学習はほかの 3 クラスに比べて劣ってしまい、クラスによって学習の差が生じてしまった。それにより、追加クラスをより学習できる凍結範囲が小さい方法が、より精度が高まったと考えられる。しかしこれは、転移学習の本来の性能が発揮できたとはいえない結果である。

7.2　エポック数

方法 2、3 において、12,000 エポックから 120,000 エポックに学習数を増やすと、精度の向上が見られた。従って、エポック数を増やすことでさらなる制度の向上が図れることを確かめるため、240,000 エポックまで学習を増やして実験した。しかし、結果は 120,000 エポックで学習し

たときの値と比べ、わずかな増加、減少が見られる。つまり、追加学習による精度の向上は見られなかったということである。これからいえることは、学習数を増やしてもこれ以上の制度の向上は望めないということである。したがって、これ以上精度を上げるには、[2] にあるような訓練データの生成、あるいは学習に関して別の方法を取る必要がある。

7.3　クラス別 AP

方法 2、3 において、クラス別の AP に着目する。転移学習によって追加したクラスである bicycle クラスの AP は、学習数を増加させることで増加していることが見える。一方、元の学習済みモデルに含まれる、car、motorbike、preson クラスの AP は、学習数を増やすことで下がっている。その中でも、motorbike が他の二つに比べ減少が大きいことが表 3 から読み取れる。その理由として考えられるのは、クラス同士の特徴の偏りである。4 つのクラスのうち、motorbike と bicycle はどちらも二輪車であるため、学習すべき特徴が似通っていると思われる。これによって、新規で学習する bicycle クラスの特徴が、すでに motorbike クラスで学習された特徴であるということが生じている可能性がある。したがって、bicycle クラスの学習に伴って、motorbike クラスの精度化下がったと考えることができる。逆に、bicycle クラスの学習に、motorbike クラスの特徴が妨げになった可能性も考えられる。その場合、別の組み合わせのクラスを選択すると精度が向上する可能性がある。これを確かめるには、これらのクラスを全く別のものに変えて追実験する必要がある。

8　結論

本研究では、学習済みの物体検出のモデルに転移学習を利用することで、1 クラス追加したモデルを少量の訓練データで作成した。そして、凍結範囲を変えて学習した 3 つのモデルを mAP で比較した。それにより、凍結範囲を抑えて学習させたモデルが精度がより優れているという結果が得られた。しかし、この結果からは転移学習の効果が十分であったとは言えず、別のパターンで再実験する必要がある。結論としては、本研究で提案する方法は、ある程度の精度は出せるものの、より高精度のモデルを求めるとなると有効であるとは言えない。

謝辞

最後に、本論文を作成するにあたり、指導、助言を頂いた、指導教官の新納浩幸教授に心より感謝申し上げます。また、研究を進めるにあたり協力していただいた、日立オートモティブシステムズ株式会社の伊藤浩朗様、土井宏治様に、深く感謝申し上げます。

参考文献

[1] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg”SSD: Single Shot MultiBox Detector”(2016).

[2] J. Talukdar, S. Gupta,P. S. Rajpura, R. S. Hegde.: ”Transfer Learning for Object Detection using State-of-the-Art Deep NeuralNetworks”2018 5th International Conference

on Signal Processing and Integated Networks (SPIN) (2018).

[3] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, "The PASCAL Visual Object Classes (VOC) Challenge" Int J Comput Vis (2010) 88: 303 - 338 Andrew Zisserman

[4] 宮本　圭一郎, 大川　洋平, 毛利　拓也『PyTorch ニューラルネットワーク実装ハンドブック』秀和システム,(2019).

付録資料

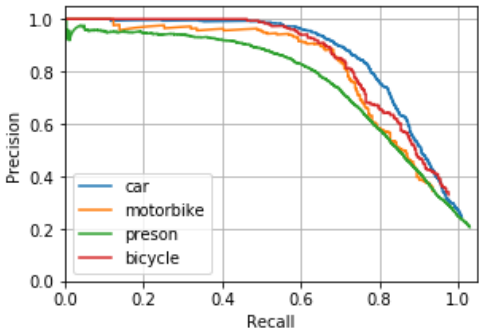

図 12　PR 曲線：4 クラス/12,000 エポック@0.5

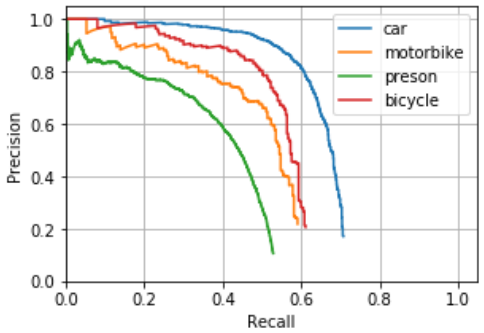

図 13　PR 曲線：4 クラス/12,000 エポック@0.7

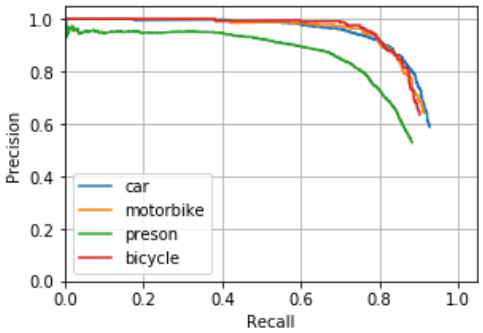

図 14　PR 曲線：4 クラス/120,000 エポック@0.5

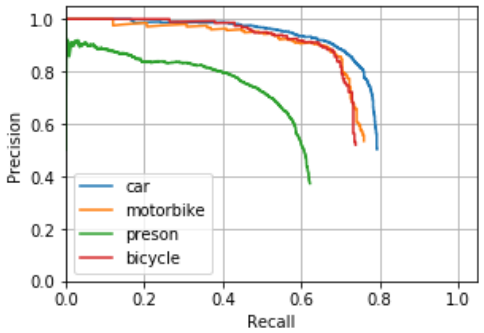

図 15　PR 曲線：4 クラス/120,000 エポック@0.7

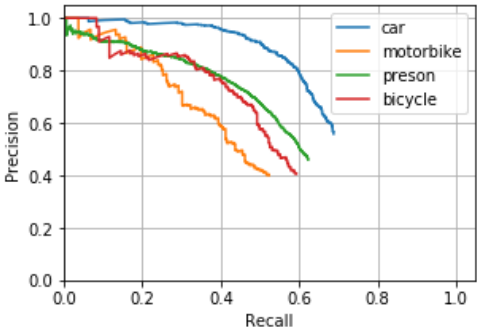

図 16　PR 曲線：4 クラス (少データ)/12,000 エポック@0.5

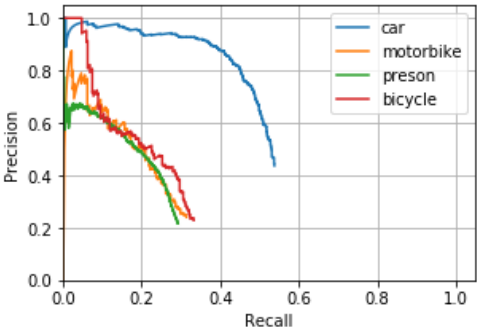

図 17　PR 曲線：4 クラス (少データ)/12,000 エポック@0.7

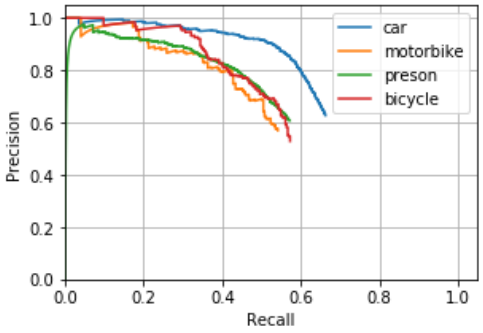

図 18　PR 曲線 : 4 クラス (少データ)/120,000 エポック@0.5

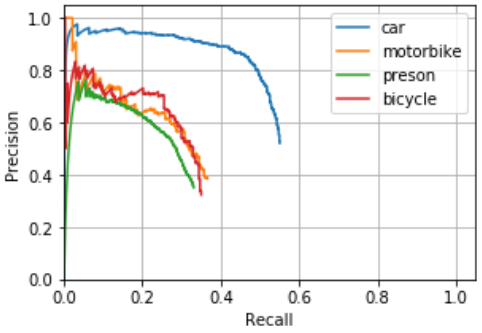

図 19　PR 曲線 : 4 クラス (少データ)/120,000 エポック@0.7

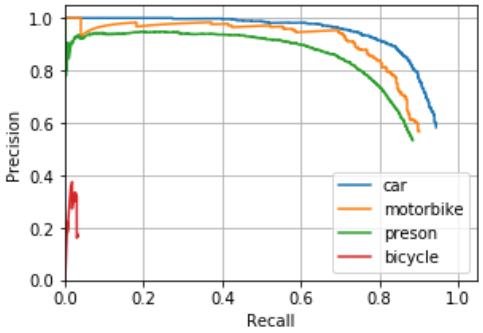

図 20　PR 曲線 : 方法 1/12,000 エポック@0.5

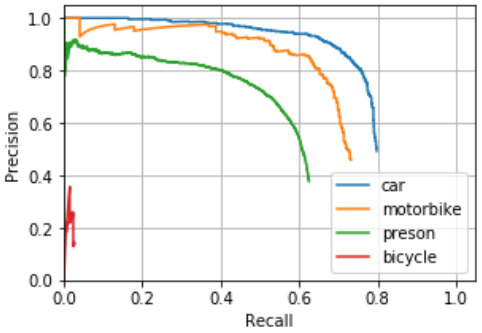

図 21　PR 曲線 : 方法 1/12,000 エポック@0.7

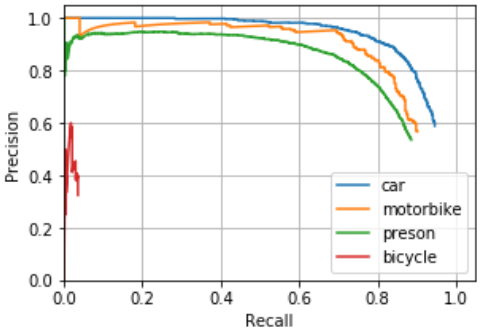

図 22　PR 曲線 : 方法 1/120,000 エポック@0.5

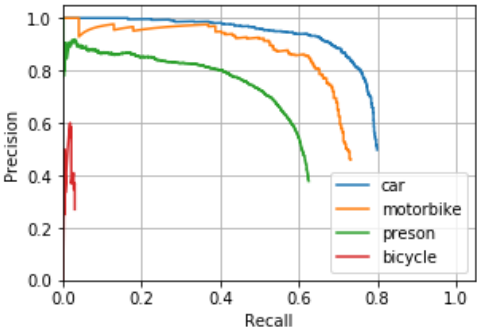

図 23　PR 曲線 : 方法 1/120,000 エポック@0.7

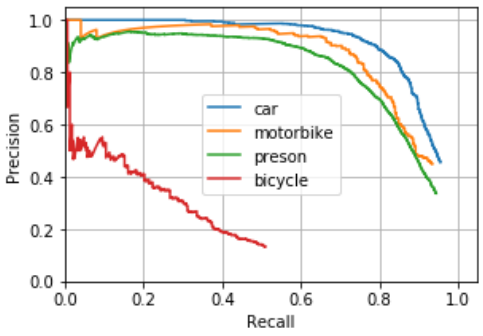

図 24　PR 曲線：方法 2/12,000 エポック@0.5

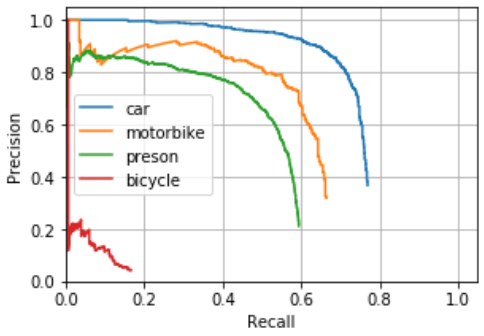

図 25　PR 曲線：方法 2/12,000 エポック@0.7

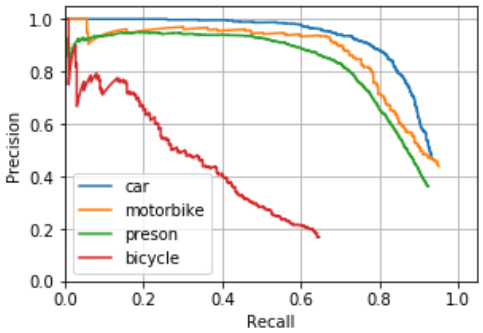

図 26　PR 曲線：方法 2/120,000 エポック@0.5

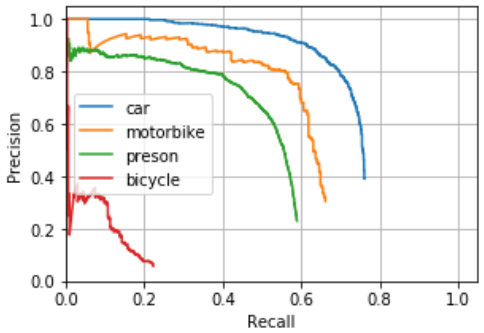

図 27　PR 曲線：方法 2/120,000 エポック@0.7

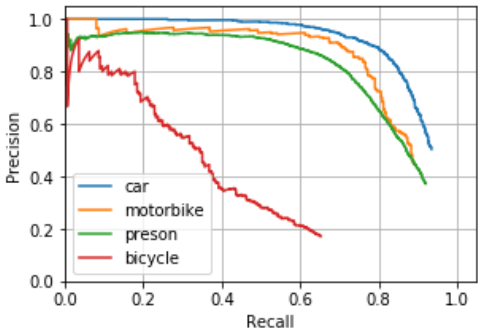

図 28　PR 曲線：方法 2/240,000 エポック@0.5

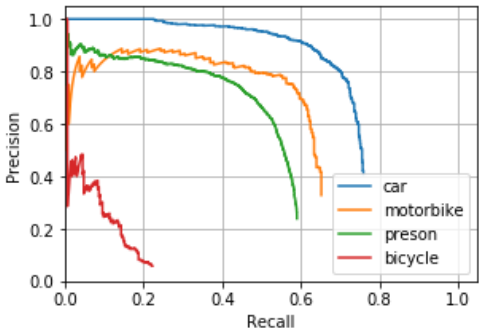

図 29　PR 曲線：方法 2/240,000 エポック@0.7

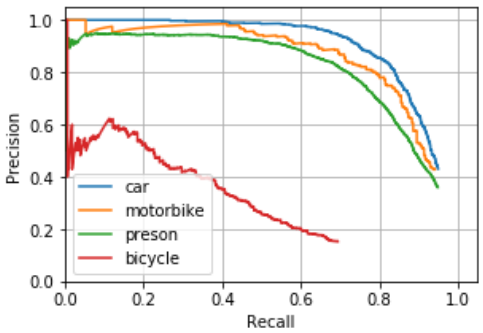

図 30　PR 曲線：方法 3/12,000 エポック@0.5

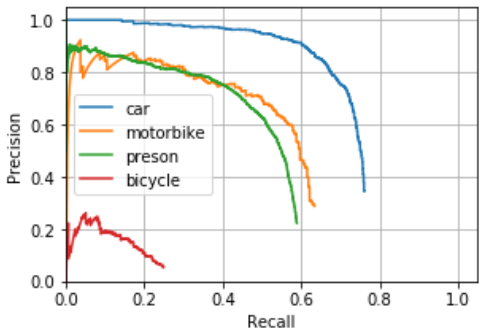

図 31　PR 曲線：方法 3/12,000 エポック@0.7

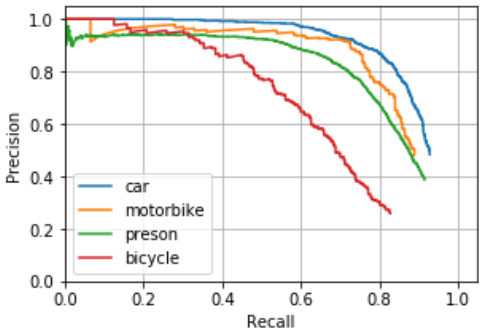

図 32　PR 曲線：方法 3/120,000 エポック@0.5

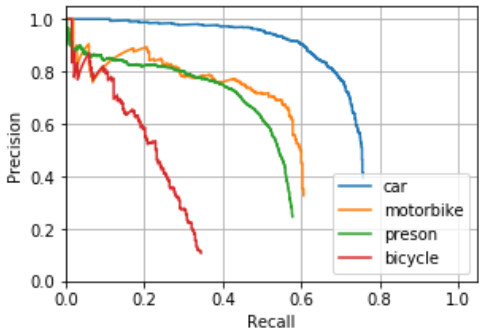

図 33　PR 曲線：方法 3/120,000 エポック@0.7

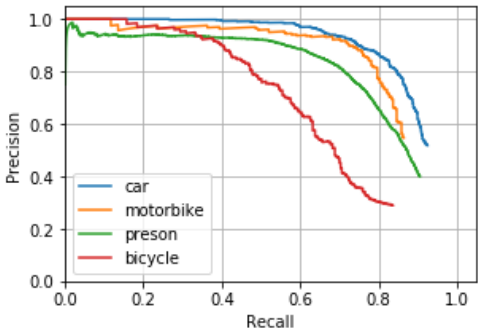

図 34　PR 曲線：方法 3/240,000 エポック@0.5

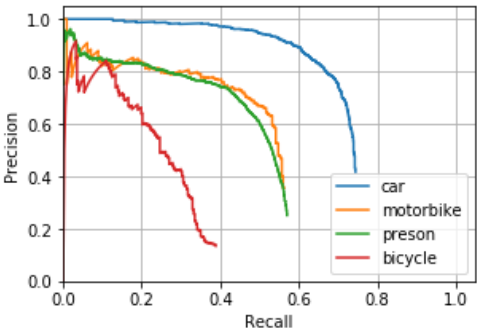

図 35　PR 曲線：方法 3/240,000 エポック@0.7