物体検出法SSDを用いた遠くの道路標識の評価

2017SC076 谷口純輝　2017SC078 手塚悠
指導教員：河野浩之

1　はじめに

近年交通事故の交通事故関与人数は年々減少しており, 2019 年においては 472,165 人と前年の 499,201 人より約 25,000 人減少している. しかし依然として発生件数は多く, 警視庁の発表によると事故の原因の中でも安全不確認と脇見運転が TOP2 になっている [1].

また, 一般物体認識の研究の発展も目覚ましく, 近年では特にディープラーニングを用いた研究が主流になっており, 一部では人間の目を超えるほどの結果を得られている事例もある. 画像の認識精度を競う 2010 年から 2017 年まで開催されたコンペティションである ILSVRC (ImageNet Large Scale Visual Recognition Challenge) で 2017 年 [2] に SEnet と呼ばれるネットワークがエラー率 0.023 と人間の 0.051 を超えた.

顔認証システムや医療分野における癌や骨折などの発見, 画像の中の文字をテキストとして出力してくれる OCR (Optical Caracter Recognition) という技術, さらには自動車業界においても障害物の位置や動きを認識したり周辺の歩行者や道路の状況を確認など多くのものに使われている. このような自動運転技術が近年の事故発生数の減少の一因を担っており, 自動運転車の実現には先進技術が不可欠である.

事故の原因の 1 つとして前方不注意による交通道路標識の発見の遅れがあり, 道路標識を検出するといった研究がなされてきた. 前方の道路標識を認識し運転者に知らせるようなシステムにより, こういった事故を減らすことは可能である. これからの運転支援システム開発において, こういった細かいことの積み重ねが事故を起こす可能性を下げることにつながると考える. しかし, 先行研究の問題点として遠くの標識の検出が困難といった問題があるがこれは距離が遠くなることで標識が小さくなってしまうからである. そのため, 本研究では遠くの道路標識の検出について試みる.

2　関連研究

物体検出の方法は Hog 特徴量やフィルタリング機能を用いたものや Deep Learning を使用した検出がある. 道路標識の検出を行っている先行研究を表 1 に示す.

表 1　道路標識検出の先行研究

先行研究	検出手法	特徴
神田ら [3]	SSD	領域抽出法より高精度
印ら [4]	色フィルタリング	高速だが低速
結城ら [5]	SSD	位置推定とクラス分類を同時に実行

表 1 において神田ら [3] はピクトグラムにおいて 2 つの分類処理を行っている. 一つ目は領域抽出方式で特徴としては上下左右のずれがないものの外枠を囲まれているものに対して検出ができないことがあげられる. 二つ目は SSD (Single Shot Multibox Detector)[4] を用いたものであり特徴としては外枠の有無にかかわらず検出できるものの背景との誤検知が発生し精度が落ちる傾向にあることがあげられる.

次に, 印ら [5] の研究では, 円型道路標識の抽出手法の提案をしており, 標識部分の候補領域を指定するために色フィルタを適用して赤色以外の部分をすべて黒に変換し, モノクロ処理を施して円の候補を抽出するという手法である. 検出は高速である反面木やビルなどの細かい模様が判定されてしまうといった精度の低さが欠点となっている.

また, 結城ら [6] の研究では十分な量の学習データによって学習した SSD のモデルを用いて, 凍結範囲を変えた 3 種類の方法にて転移学習を行うという手法を提案した. 実験の結果, 凍結範囲を減らし学習層を増やすことで精度の向上が見られた一方で, ある程度の精度は出せるもののより高精度のモデルには有効ではない.

CNN での物体検出に用いられるアルゴリズムのうち SSD, YOLO, R-CNN が一般手法として知られるためそれぞれの特長と問題点を表 2 にまとめた.

先行研究においては背景を含んでいる標識画像や, 小さい標識に対しての検出についてあまり芳しくない結果になっていた. この問題に対し, 私たちは背景の多いデータセットや, 検出物体が小さい画像をデータセットに用いて学習したモデルを用いることで検出ができるのではないかと考えた. そこで我々は背景が多く, 標識の小さいものと背景が少なく標識の大きいものデータセット作成し, 検出比較, 評価を行うことにした.

表 2　検出アルゴリズムの比較

アルゴリズム	特長 問題点
SSD	高速な検出が可能 小さい物体の検出が不安定
YOLO	検出と識別を同時に行う 小さい物体の検出が不安定, 低精度
R-CNN	物体位置推定, クラス分類 処理速度が遅い

3　提案手法

3 節では提案手法として使用するものについて述べる. 3.1 では検出アルゴリズムについて, 3.2 ではフレームワークについて, 3.3 ではデータセットについて, 3.4 ではモデル学習について, 3.5 ではモデル評価について説明する.

3.1 検出アルゴリズム

実際の一般道路での運転を想定した時, いち早く道路標識の検出をするかが重要であり, また画像上の物体を検出する上で物体が一つしか存在せず, 領域が一定であることは考えにくい. そのため, 様々なサイズで写っている複数の物体を上手く切り出すバウンディングボックスを探し, クラス分類の問題に持ち込む必要がある. そこで考えられるのは Sliding Window 法である.

これはバウンディングボックスの位置を動かし, すべてのパターンにおいて探索を行う. この方法では検出をすることができるが時間がかかりすぎてしまうため一般道路を想定した時には使うことが難しい. また Selective Search 法を利用した R-CNN があげられる. これは似た特徴を持った小さい領域を統合させていき, 複数のサポートベクターマシンによって学習しカテゴリ識別した後に物体が存在しそうな領域に当たりをつけるものである. この手法だと工夫が加えられているため Sliding Window 法よりも高速ではあるが, 一方で冗長性があるためより高速に処理することが可能である.

そこであらかじめ画像をグリッドで分割し, 各領域ごとにクラス識別とバウンディングボックスを求める YOLO や SSD があげられる. この二つの手法では上記の手法に比べて無駄が省かれるため高速になっている. また YOLO と SSD の比較では SSD がより高速かつ低解像度でも認識しやすい. しかし小さい物体の検出の不安定さがある. 以上より本研究に使用する検出アルゴリズムは SSD とする. SSD を使う際小さい物体の検出精度を上げるため, 学習データを背景の占める範囲を増やし標識の部分を小さくすることで標識が小さい時での精度が向上すると考えた.

SSD の模式図を図 1 に示す. SSD では VGG-16 という畳み込み層の後に Extra Feature Layers という畳み込み層を追加している. VGG-16 は Visual Geometry Group が作った ImageNet データベースの 100 万枚を超えるイメージで学習済みの畳み込みニューラルネットワークのことで畳み込み 13 層＋全結合層 3 層＝ 16 層で構成されている. この追加した Extra Feature Layers があることでマルチスケールでの検出を可能にしている.

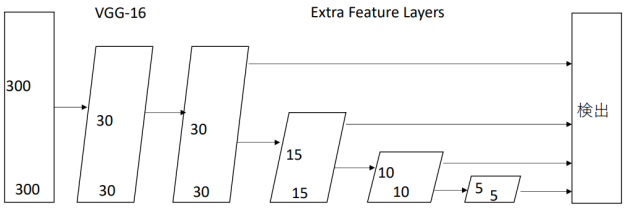

図 1 SSD のモデル図

3.2 フレームワーク

フレームワークの比較表を表 3 に示す. TensorFlow は Google が提供しているオープンソースソフトウェアで対応言語は Python, C, Java, Go となっている. 分散処理に優れており, また Tensorboard というツールを使うことができる. TensorBoard はデータフローグラフの可視化や学習の履歴の可視化, 途中過程で生成される画像や音声の表示を行うことができる. これにより, モデルの最適化やパラメーターチューニングがしやすくなるという利点がある. 他にも Intel の CPU 向けにディープラーニング用の最適化が行われており, それを TensorFlow で利用できるようにしたものを Intel が配布している. これは pip で配布したものと比較すると 8.6 倍高速で学習することができる. Keras は various が提供しているオープンソースソフトウェアで対応言語は Python となっている. Keras で書いたコードをどのライブラリで実行するかを選ぶことができるので TensorFlow などの, 素のままだと Deep Learning を書くには粒度の小さすぎるライブラリをより手軽に使うことができる.

PyTorch は Facebook が提供しているオープンソースソフトウェアで対応言語は Python, C++, CUDA となっている. これは強力な GPU のサポートを備えることでより高度なテンソル計算を行うことができる. 以上の比較より, 本研究では Intel の CPU を使用するためより高速に学習できる TensorFlow を用いて Keras を実行するという手法を採用する.

表 3 フレームワーク比較

フレームワーク	開発元 特長
TensorFlow	Google 分類処理に優れており Tensorboard という学習可視化ソフトを持つ
Keras	various Keras のコードをどのライブラリで実行するか選ぶことができる
Pytorch	Facebook 強力な GPU サポートを備えたテンソル演算ができる

3.3 データセット

道路標識のデータセットとして GTSRB を使用して研究を行っているものもあるが, GTSRB は画像内に占める標識の領域が多いものしかないため今回の実験には向かない. そのため実験に使用する道路標識のデータセットは自分たちで作成する. データセットにはさまざまな角度や明るさにおける道路標識の画像データが必要であり, また, 画像内の物体の位置座標とその物体が何なのかというラベル付けを行うアノテーションという作業をする. さらに今回の実験では標識の画像における範囲による精度の比較を行う. 使用する標識は「とまれ」, 「駐車禁止」, 速度標識の「40」, 「50」の 4 クラスとし「40」のデータは画像内の標識の範囲を大きくし, 逆に「50」のデータは小さくする. また, 「とまれ」と「駐車禁止」の 2 つはどちらも標識の範囲を小さいものも大きいものも同程度の量にして作成する. また 1 クラス当たりに用意する画

像の枚数として GTSRB では 210 枚から 2250 枚で作成されており, 我々も 1 クラス 420 枚, 4 クラスで合計 1680 枚の画像データとそれに対応するアノテーションデータの 2 つを用いてデータセットを作成する．また, データセットには, データに対し「45 度回転」,「315 度回転」,「上下左右の平行移動」の画像処理を行う．図 2 に本研究で使用するデータの例を示す．なお, 画像の左上は「下に 50 ピクセル平行移動」, 右上は「右に 50 ピクセル平行移動」, 左下は「45 度回転」, 右下は「315 度回転」である．

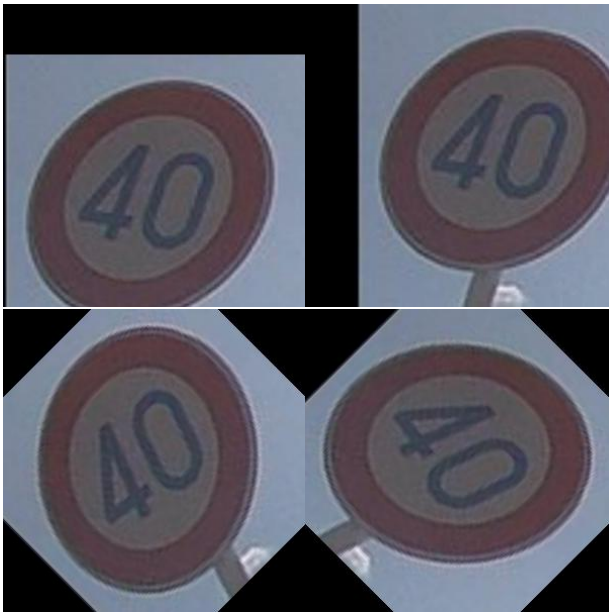

図 2 作成したデータセットの例

3.4 モデル学習

データセットを作成した後, 次に検出を行うためのモデルを学習する必要がある．モデル学習は前節で述べたアノテーションデータに示された画像中の位置とそれが何であるというラベルから, その物体がどういう形であるかを学習していく．この流れをデータセットの全データから繰り返し行っていくことでモデルの精度を上げていく．その際, エポック数とバッチサイズの調整をすることで精度の向上を望めるため, 適切なパラメータを見つけるがそれは次の 4 節で述べていく．

3.5 モデル評価

作成したモデルを用いて標識の検出を行っていくうえでモデルの出来を評価する必要がある．本研究においては道路標識を正しく検出できた, 誤検出だった, 未検出だったの 3 通りあるため, ただ何枚中何枚検出できたというような測定では不適．そこで今回は精度を評価する手法の中で *F1 Score* という手法を用いる．*F1 Score* とは F 値とも呼ばれ, *Precision* 手法と *Recall* 手法の二つの手法のバランスをとった手法である．*F1 Score* の数式を式 (1) に示す．これをもとに作成したデータセットの性能を評価していく．

$$
\begin{aligned}
F1\ Score &= \frac{2 \times\ Precision \times\ Recall}{Precision + Recall} \\
Precision &= \frac{TP}{TP + FP} \\
Recall &= \frac{TP}{TP + FN} \qquad (1) \\
TP &: \text{検出できた物体の数} \\
FP &: \text{誤検出した物体の数} \\
FN &: \text{未検出の物体の数}
\end{aligned}
$$

4 実験

今回の実験では実験環境として OS は Windows10, 言語は Python を使用し, TensorFlow で Keras を使い SSD を動かす．4.1 ではモデル学習について, 4.2 ではプログラムを実行した検出結果について, 4.3 で結果をもとに考察をする．

4.1 モデル学習

前節で述べたようにエポック数とバッチサイズのパラメータの調整をする．「とまれ」の標識に対し, エポック数を 1, 5, 10 の 3 段階, バッチサイズを 2, 10, 20, 30, 42, 60 の 6 段階でモデルを学習させ, その際に出た損失を比較して最適なパラメータを見つける．パラメータの比較の一部を表 4 に示す．

表 4 パラメータ変更時の精度比較

エポック	バッチサイズ	loss	val_loss	時間 [s]
1	10	8.08	6.05	90
1	20	9.71	10.47	90
1	30	10.27	10.81	90
5	10	3.04	2.99	437
5	20	3.19	3.45	443
5	30	3.43	3.98	450
10	10	2.82	2.71	960
10	20	2.82	2.81	980
10	30	2.87	2.85	1005

以上から loss と val_loss がともに低いエポック 10 の時のバッチサイズ 10, 20 あたりが望ましいと考えられる．また, この作業をほかの 3 つの標識に対しても同様に行う．

4.2 プログラムの実行

前節で得られたパラメータを用いることによって出力された検出の結果を図 3, 図 4 に示す．バッチサイズ 10 の時 0.49, バッチサイズ 20 の時 0.62 であるという結果になった．

次に, 得られたパラメータをもとに作成したモデルの評価を *F1 Score* を用いて行う．

図 3　バッチサイズ 10 の時の検出結果

図 4　バッチサイズ 20 の時の検出結果

4.3　モデル評価

4 つの標識に対し最適なパラメータで *F1 Score* を計測した結果を表 5 に示す. *F1 Score* の結果によると, 「40」と「駐車禁止」が同程度の高さで一番高く, 「とまれ」が 44%程度, 「50」が一番精度が低かったことがわかる. 今回の実験では「40」と「50」の標識に対しデータセットに変化を加えており, 画像内の標識の範囲を大きくしたり, 小さくしたりして精度の比較を行った. 標識の範囲を大きくした「40」とのデータセットによる検出ではモデル評価が 69%という精度が出ている一方で, 範囲を小さくした「50」のデータセットでは 8%程度と非常に低精度という結果が得られた. また, 「とまれ」と「駐車禁止」の 2 種類の標識に対しては標識の範囲が大きいものと小さいものを同じくらいにしてデータセットを作成しておりどちらも 44%の精度が得られていることがわかる. 表 5 によると「40」の検出では「とまれ」や「駐車禁止」に比べ誤検出は大きく変わらなく, 正しく検出できた数が大きく増え未検出が減っているためすべてが向上していることがわかる. 「50」の検出では誤検出が非常に多く原因として標識の領域を小さくしすぎることで標識の特徴量を正しく抽出できていないことが考えられる. そのため背景の標識以外の領域を標識だと認識してしまっている.

表 5　*F1 Score* 比較

標識	バッチサイズ	score	TP	FP	TN
とまれ	20	0.4458	188	232	232
駐車禁止	10	0.6972	274	92	146
40	20	0.6905	290	130	130
50	20	0.0877	34	321	396

5　むすび

今回我々は遠くの交通道路標識の検出を SSD という手法によって実現するという研究に取り組んできた. 遠くの標識の検出をするために, 画像に対して標識の割合が小さい標識を用いることで検出精度の向上を図ったが, 「50」に関しては 8%とかえって位置推定がうまくいかなくなり検出精度が落ちてしまうといった結果になってしまった.

一方で「とまれ」においては約 45%, 「40」, 「駐車禁止」に関しては約 70%と高い評価を得られることができた.

参考文献

[1] 警察庁, 「平成 29 年中の交通事故の発生状況, 」 https://www.npa.go.jp/publications/statistics/koutsuu/H29zennjiko.pdf, 参照日 2021-01-19.

[2] ILSVRC, 「IMAGENET Large Scale Visual Recognition Challenge (ILSVRC) 2017 Overview, 」 http://image-net.org/challenges/talks_2017/ILSVRC2017_overview.pdf, 参照日 2020-09-10.

[3] 神田剛志, 伊藤一成, “深層学習を用いたピクトグラム画像における 多クラス分類問題とその応用, ”エンタテインメントコンピューティングシンポジウム（EC2019）, pp. 73-79, 2019 年 9 月.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg. “Ssd: Single shot multibox detector, ”In ECCV, pp. 21-37. 2016.

[5] 印 芳, 片山 健太朗, 吉田 英司, “物体検出システムの性能向上に向けた円型標識抽出手法の提案, ”信学技報, vol. 119, no. 324, IE2019-57, pp. 41-46, 2019 年 12 月.

[6] 結城洸太, 新納治幸, “物体検出モデル SSD に対する転移学習, ”人工知能学会全国大会論文集 第 34 回全国大会 (2020).