日本データベース学会和文論文誌
Vol. 20-J, Article No. 10, 2022 年 3 月

軽量・高精度な物体検出が可能な Feature Sharing Network(FSN)の提案

山重 雄哉[1]　青野 雅樹[2]

深層学習に基づく画像上の物体検出技術は盛んに研究が行われている．近年ではドローンや小型ロボットなどの自動運転に向けた軽量で高速なエッジ AI も注目されている．代表的な手法である YOLOv4 は 65FPS の検出速度で従来手法を上回る精度を記録したが，計算コストは高く，高性能な GPU を必要とする．これに対してより軽量なモデルである YOLOv4-tiny は，GPU を用いずともリアルタイムな検出を可能とするが，YOLOv4 と比較して精度は劣り，特に小物体の検出精度に課題がある．そこで本稿では，YOLOv4-tiny の高速性を維持した高精度化改良を行う．具体的には，特徴抽出を行う Backbone 部および特徴量を補強する FPN 機構を改良する．これらの改良を行ったネットワークを YOLOv4-tiny-3det-light と呼び，新たな特徴量補強機構を Feature Sharing Network (FSN)として提案する．実験では，Udacity Annotated Driving Dataset を用いて精度，推論速度，モデルパラメータ数，計算量 (FLOPs)の側面から評価を行い，従来手法に対して優位性を示した．また，検出結果の可視化によって検出精度の向上を視覚的に確認した．

1. はじめに

深層学習を用いた画像上の一般物体検出技術が注目を集めている．物体検出は様々なビジネスシーンに応用可能であり，自動車の自動運転をはじめ監視カメラによる自動監視，工場での外見検査，がん検出等の様々な応用先が考えられる．特に，近年ではドローン等の小型ロボットにおける自動運転や，スマートフォンを車載カメラとして用いた運転補助アプリケーション等の実現に向けて，エッジ AI の需要も高まっている．そのため，高い精度だけでなく，計算量が少なく軽量で，高速に動作する物体検出モデルが求められている．

一般的に深層学習による物体検出は精度と速度のトレードオフが課題とされ，特に自動運転シーンでは精度およびリアルタイム性が求められる．また，モバイル端末やドローンなどの小型デバイスを用いる場合はマシンコストの側面も考慮しなければならない．You Only Look Once (YOLO) [1]は比較的高速な物体検出手法として提案され，現在も改良が続いている．YOLO の改良モデルである YOLOv4 [2]は，検出精度および速度で従来手法を上回る性能を記録した．一方，YOLOv4 は特徴抽出を行う Backbone 部の計算量が非常に多く，リアルタイムな検出には高性能な GPU を必要とする．そのため，物理的な制約が多く，消費電力や費用等のコストが高いという問題を抱えている．そこで，本稿では新たな物体検出手法として YOLOv4-tiny-3det-light を提案する．本手法は，YOLOv4 の畳み込み層を削減した YOLOv4-tiny [24]を用いることで計算量を大幅に削減し，Backbone 部の変更によって YOLOv4-tiny の課題であった小物体の検出精度の低下を改善している．また，新たな特徴量補強機構として，Feature Pyramid Network (FPN) [4]を改良した Feature Sharing Network (FSN)を提案する．

比較実験ではデータセットとして Udacity Annotated Driving Dataset [25]を用いて精度，速度，計算量の側面から提案手法の性能を評価し，高速な検出速度を維持した精度の向上を確認した．加えて，従来の FPN 機構との比較も行い，従来手法に対する優位性を示した．また，GPU 未使用時の検出においても 27 [FPS]のリアルタイム性を獲得し，検出結果の可視化によって検出精度の向上を視覚的に確認した．

2 節では，関連研究として深層学習を用いた代表的な物体検出手法，YOLOv4 および FPN 機構について説明する．3 節では提案手法について説明する．4 節では比較実験の結果を示し，5 節では本実験の考察，6 節では結論および今後の課題について述べる．

2. 関連研究

本節では関連研究を示す．2.1 節では深層学習を用いた代表的な物体検出手法を説明する．また，2.2 節において YOLOv4 および YOLOv4-tiny の概要を，2.3 節において FPN について説明する．

2.1. 深層学習を用いた物体検出手法

深層学習を用いた物体検出手法は two stage 型と one stage 型に大別され，前者は物体の候補領域を提案するステップと，候補領域内の物体クラスを特定するステップに分かれている点が特徴的である．代表的な手法としては Regions with CNN features (R-CNN) [5]，Fast R-CNN [6]，Faster R-CNN [7]，Mask R-CNN [8]などがある．Faster R-CNN は The PASCAL Visual Object Classes Challenge [18]における Pascal VOC 2007 test Dataset で 73.2 mAP の高い精度を記録したが，検出速度は GPU を用いた場合でも約 5 FPS であり，リアルタイム性に欠けるという問題がある．

one stage 型は領域提案とクラス分類のステップを同時に行うため，two stage 型と比較して高速である．代表的な手法として，Single Shot MultiBox Detector (SSD) [3]，YOLO，EfficientDet [14] などが挙げられる．Redmon らの YOLO (YOLOv1) は，R-CNN 系の手法と比較して高速な検出が可能な手法として提案された．Backbone 部には Google LeNet [27]を使用し，解像度が 7×7 の特徴マップ の各セルを畳み込むことで，バウンディングボックスの座標値とクラス確率を計算する．なお，YOLOv1 は Pascal VOC 2007 test において 45FPS の速度で 63.4mAP の検出を可能とした．一方，特徴マップのセル内に小さな物体が複数存在した場合の検出が困難になるという欠点もあった．

[1] 豊橋技術科学大学 情報・知能工学専攻 yamashige@kde.cs.tut.ac.jp
[2] 正会員 豊橋技術科学大学　大学院工学研究科 情報・知能工学系 aono@tut.jp

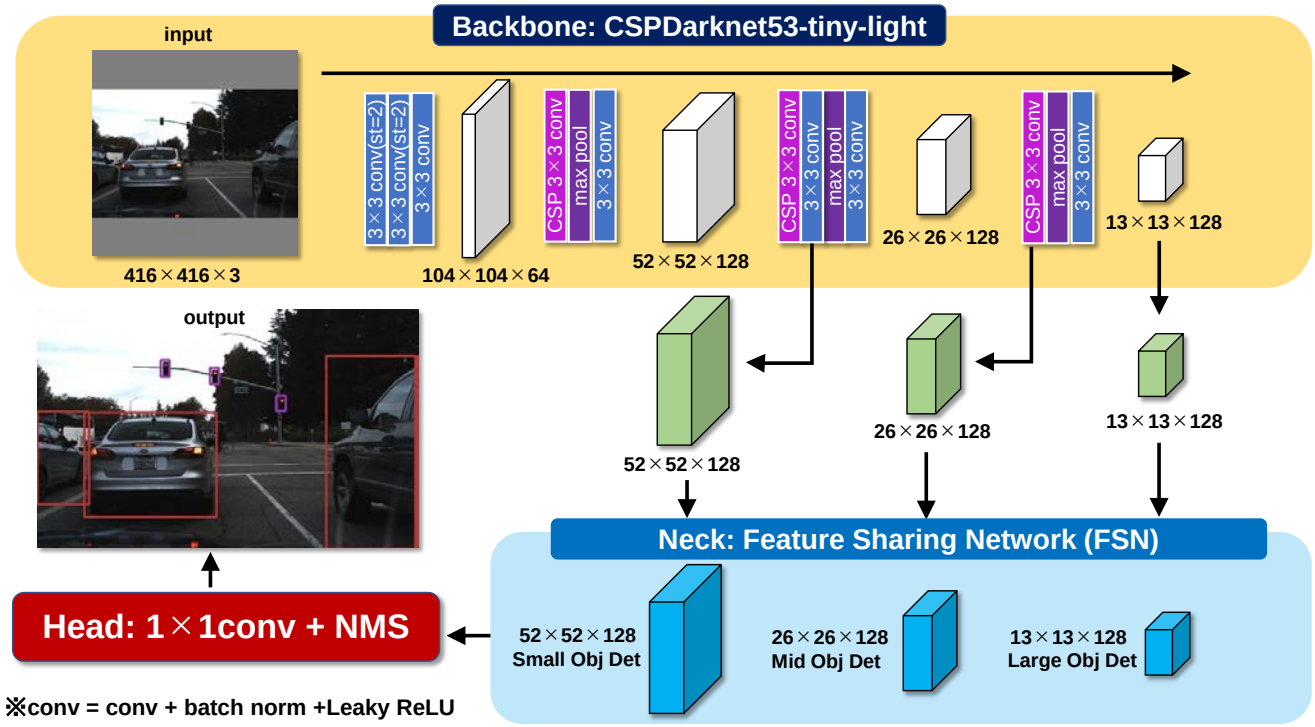


図 1. 提案手法 YOLOv4-tiny-3det-light のネットワークアーキテクチャ

SSD は，YOLOv1 における小物体の検出精度が低い問題を改善し，物体サイズにロバストな検出を可能とした手法である．YOLOv1 と異なり，Backbone 部に VGG-16 [19]を使用している．また，物体予測を行う際に一つの特徴マップを用いるのではなく，様々な解像度を持つ複数の特徴マップを用いる点が特徴である．これは，浅い層の高解像度特徴マップでは小物体の検出を，深い層の低解像度特徴マップでは大物体の予測を行うため，物体のサイズにロバストな検出が可能となっている．ここで，特徴マップのグリッド上にデフォルトボックス（アンカーボックス)と呼ばれる基準のバウンディングボックスを定めておき，畳み込みでボックスのオフセットを計算することで物体領域を予測する．また，SoftMax 関数を用いて，物体のクラス確率も同時に予測する．さらに，重なったバウンディングボックスは Non-Maximum Suppression (NMS)と呼ばれる手法を用いて統合する．これらの SSD における物体検出の一連の流れは，後述する改良モデルの基盤となっているアルゴリズムである．なお，SSD (SSD300)は Pascal VOC 2007 test において 46FPS の速度，77.2mAP の高い精度を記録した．

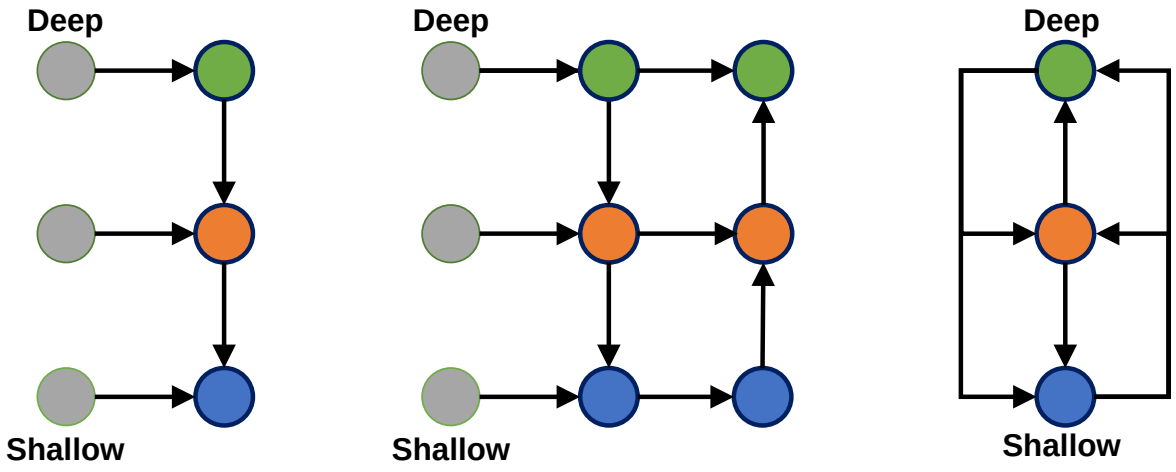
上述したアルゴリズムをベースとして様々な改良モデルが提案されている．例えば，SSD ベースの改良モデルとしては DSSD [11]， RefineDet [12]，M2Det [13]等がある．YOLO ベースの手法には，YOLO9000 (YOLOv2) [9]や，YOLO v3[10]，YOLOv4 がある．YOLO の改良手法は，Backbone 部が Google LeNet から Darknet に変更され，SSD と同様に複数の解像度の特徴マップを用いて物体予測を行うことで精度を向上している．

2.2. Feature Pyramid Network (FPN)

解像度の異なる特徴マップを用いる手法では，小物体の検出に浅い層の特徴マップを用いる．そのため精度が低下するという問題が発生する．この問題を解決したのが Feature Pyramid Network (FPN) [4]である．FPN は図 2(a)に示すように，深い層にある特徴マップをアップサンプリングし，浅い層の特徴マップと要素和融合する．これを繰り返すことで，深い層の特徴量を伝播する．これによって，浅い層の特徴マップの意味的な特徴量が補強され，小物体の検出精度の向上が可能となる．FPN は多くの物体検出モデルに用いられ，アーキテクチャの改良が続けられている．例えば，図 2(b)に示す浅い層から深い層への逆方向への特徴量伝播を加えたに示す PANet [15]，Neural Architecture Search を用いて，最適な FPN アーキテクチャの探索を行う NAS-FPN [16]，PANet の処理を繰り返すことで，さらに特徴マップを洗練化し，スキップ接続を追加した BiFPN 等がある．

2.3. YOLOv4, YOLOv4-tiny

YOLOv4 は，近年の様々な研究成果を統合した YOLO ベースの最新手法である．YOLOv4 の Backbone 部は Darknet53 と呼ばれるネットワークに CSPNet [29]の畳み込み手法を組み込んだ CSPDarknet53 で構成されている．また，Neck 部と呼ばれる部分において，Spatial Pyramid Pooling (SPP) [32] および PANet を用いた特徴量の補強を行っている．物体予測を行う Head 部では，各アンカーボックスに関してオフセット値，物体か背景かを分類する物体性確率，およびクラス確率を計算する．



(a) FPN (YOLOv4-tiny) (b) PANet (YOLOv4) (c) FSN (proposed)

図 2. 従来の FPN 機構と提案手法の比較

また，損失関数として，領域の損失には CIoU-Loss [31]を，物体性確率の損失には Focal Loss [30]を，クラス確率の損失にはクラスごとの Binary Cross Entropy Loss を用いる．また，訓練時の工夫として，学習率を線形的に増加させる warm-up や，学習率の減衰に Cosine Annealing Scheduling [28]を使用している．

YOLOv4 は，最高性能を記録していた EfficientDet の性能を上回る性能を記録した．一方，計算量は 59.7 GFLOPs (FLoating-point OPerations)であり，リアルタイムな検出には高性能な GPU を必要とする．これに対して，より軽量なモデルとして提案された YOLOv4-tiny は，YOLOv4 の畳み込み層を削減し，PANet を FPN に変更することで計算量を 6.8GFLOPs に削減している．そのため，GPU を用いずともリアルタイムな検出が可能であるが，YOLOv4 と比較して精度は劣っており，小物体の検出精度に課題がある．

3. 提案手法

我々は YOLOv4-tiny の計算量を削減し，高速性を維持しつつ高精度化改良を行った YOLOv4-tiny-3det-light を提案する．

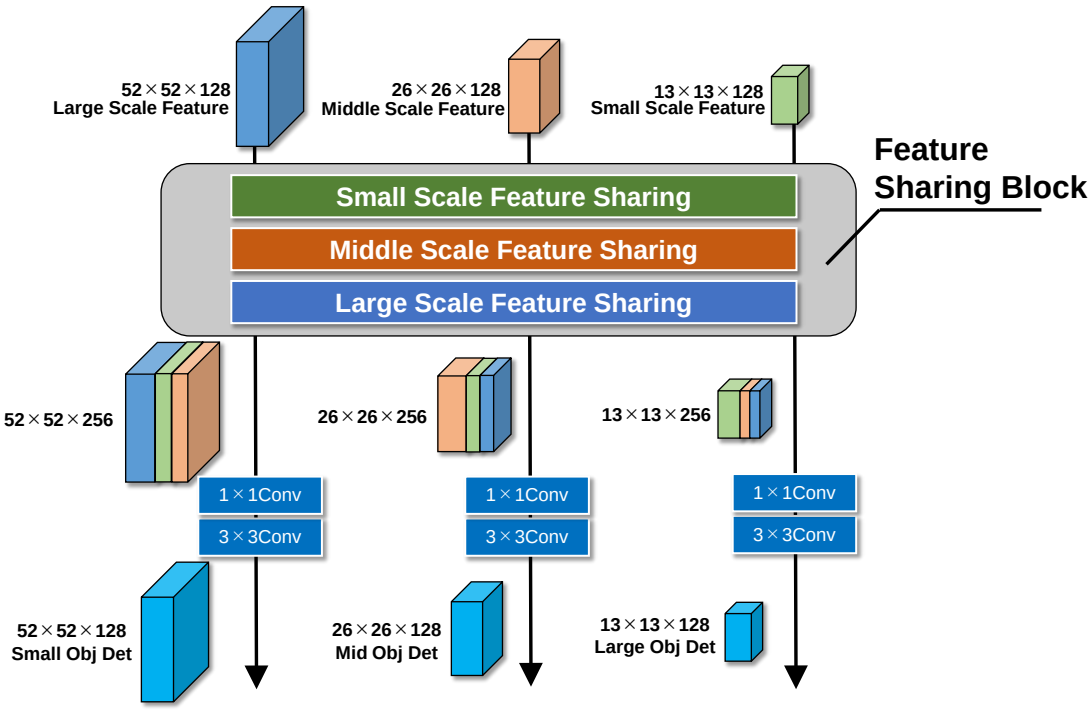
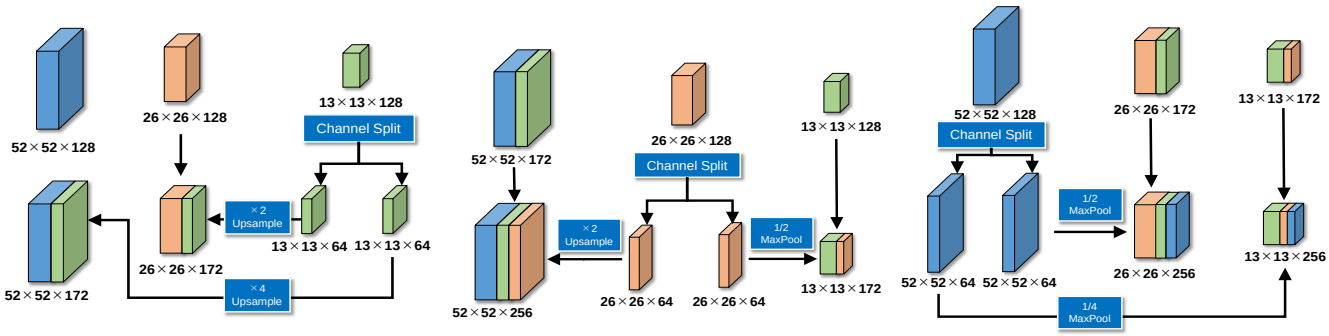


図 3. 1. Feature Sharing Network (FSN)のアーキテクチャ



(a)Small Scale Feature Sharing　(b) Middle Scale Feature Sharing　(c) Large Scale Feature Sharing

図 3. 2. Feature Sharing Network における各モジュールのアーキテクチャ

3.1 節では提案手法の概要を示す．3.2 節では，従来の FPN に代わる新たな特徴量補強機構として，Feature Sharing Network (FSN)を提案する．

3.1. YOLOv4-tiny-3det-light

2.3 節で述べたように，YOLOv4-tiny は YOLOv4 と比較して計算量は少ないが，低精度であり，特に小物体の検出精度が低い．そこで，YOLOv4-tiny を高精度化した YOLOv4-tiny-3det-light を提案する．主なアプローチは高解像度特徴マップの追加および FPN の改良に基づいている．なお，ネットワークアーキテクチャは**図 1** に示す通りである．

まず，高解像度特徴マップの追加について説明する．YOLOv4-tiny の Head 部では，解像度が異なる複数の特徴マップを用いて，物体のサイズにロバストな検出を行う．ここで，YOLOv4-tiny は 13×13 および 26×26 の特徴マップを物体予測に用いており，この低解像度の特徴マップが，小物体の検出精度の低さの原因ではないかと考えた．そこで，**図 1** の緑色の特徴マップで示すように 52×52 の高解像度な特徴マップを加え，3 層の特徴マップで予測を行うようにアーキテクチャを変更する．しかし，この変更によって計算量は 6.8 GFLOPs から 9.7 GFLOPs に増加し，YOLOv4-tiny の軽量性が失われる．そこで，物体予測に用いる特徴マップの次元数を 256 から 128 に変更することで計算量を削減する．この変更は，自動運転等の応用シーンでは検出すべき物体の種類数は比較的少ないため，チャンネル次元数を減らしても精度に影響が出ないという仮説に基づいている．なお，3 層の特徴マップは Neck 部へと入力され，FSN に基づく特徴量の補強を行う．

3.2. Feature Sharing Network (FSN)

従来の YOLOv4-tiny では，結合によって特徴マップを融合する**図 2**(a)の FPN が用いられている．ここで，FPN に代わる新たな特徴量補強機構として Feature Sharing Network (FSN)を提案する．本機構のアイデアは，**図 2**(b)の PANet において双方向の特徴量伝播が精度の向上に有効であること，および層をスキップして特徴量を補強する NAS-FPN の構造から着想を得た．FSN の構造は，従来の特徴量を段階的に伝播する構造とは異なり，**図 2**(c)に示すように，相互に特徴量を分配・共有し合うような構造になっている．この構造によって，空間的から意味的といった様々なレベルの特徴量を，従来の機構と比較して効率的に補強することができると考えらえる．なお，FSN は本提案ネットワークだけでなく，その他のネットワークにも導入可能な汎用的なモジュールである．以下では，本提案における導入例を説明する．

FSN のアーキテクチャを**図 3. 1** に，各モジュールのアーキテクチャを**図 3. 2** に示す．ここで，Backbone 部で抽出した 52×52×128，26×26×128，13×13×128 の特徴マップはそれぞれ Large Scale Feature，Middle Scale Feature，Small Scale Feature と呼ぶ．これらの特徴マップは，Feature Sharing Block に入力した後，**図 3. 2**(a)に示す Small Scale Feature Sharing (SSFS) の処理を適用する．SSFS では，最も解像度の低い Small Scale Feature が持つ意味的な特徴量を Middle Scale および Large Scale Feature に分配することで特徴量の補強を行う．

具体的には，Small Scale Feature をチャンネル次元で 2 分割し，一方には解像度が 2 倍になるアップサンプリングを，もう一方には解像度が 4 倍になるアップサンプリングを適用して，解像度を一致させる．これらを Middle Scale Feature および Large Scale Feature と結合(Concatenation)する．SSFS 適用後，**図 3. 2** (b)に示す Middle Scale Feature Sharing (MSFS)の処理に移行する．MSFS は Middle Scale Feature が持つ特徴量を，解像度の異なる特徴マップへ分配する．解像度の調整には，アップサンプリングおよび Max Pooling を用いている．ここで，Large Scale Feature への結合は，先の SSFS で結合した特徴量(緑色で示す特徴マップ) に付け加えるようにして行う．最後に**図 3. 2** (c)の Large Scale Feature Sharing (LSFS)を適用する．上記と同様に Large Scale Feature をチャンネル次元で分割後，

表 1. YOLOv4 および YOLOv4-tiny との検出性能の比較

method	VOC-mAP	COCO-AP	Inference time [ms]		FPS		Params[M]	FLOPs[G]
			w/ GPU	w/o GPU	w/ GPU	w/o GPU		
YOLOv4 [2]	46.7	21.2	26.381	302.387	37.906	3.307	64.0	59.7
YOLOv4-tiny [24]	34.6	13.8	7.514	36.656	133.085	27.281	5.9	6.8
YOLOv4-tiny-3det-light (proposed)	41.1	16.9	8.700	**36.564**	114.943	**27.349**	**1.7**	**6.4**

表 2. 従来手法との検出性能の比較

method	Inference time [ms]		FPS		Params	FLOPs[G]	VOC-mAP
	w/ GPU	w/o GPU	w/ GPU	w/o GPU			
SSD7 [22]	15.330	-	65.233	-	213,904	-	25.4
FPSSD7 [20]	13.043	-	76.669	-	299,472	-	27.8
AFPSSD7 [21]	15.444	-	64.751	-	384,176	-	31.8
SSD300 [3]	23.446	-	42.650	-	24,280,556	-	34.6
YOLOv3 [10]	28.649	-	34.905	-	61,597,882	65.4	45.0
YOLOv3-tiny [23]	11.763	-	85.012	-	8,685,484	5.5	29.8
YOLOv4 [2]	26.381	302.387	37.906	3.307	64,025,530	59.7	46.7
YOLOv4 -tiny [24]	7.514	36.656	133.085	27.281	5,889,564	6.8	34.6
YOLOv4-tiny_3det_light (ours)	8.700	36.565	114.943	27.349	1,703,098	6.4	41.1

表 3. クラス別精度（VOC Class AP）の比較

method	car	truck	pedestrian	bicyclist	light	VOC-mAP
YOLOv4 [2]	71.8	47.8	35.4	19.2	59.1	46.7
YOLOv4 -tiny [24]	64.1	38.3	16.7	10.9	43.2	34.6
YOLOv4-tiny-3det-light (proposed)	66.9	45.4	23.5	20.4	49.5	41.1

Max Pooling を用いて解像度を一致させ，スケールの異なる特徴マップと結合する．以上により，特徴量を分配し合ったチャンネルが 256 次元の特徴マップが 3 つ得られ，それぞれにカーネルサイズが 1×1 の畳み込み(1×1Conv)および 3×3 の畳み込み(3×3Conv)を適用して処理を終える．

4. 比較実験

比較実験では，車載カメラ画像データセットの Udacity Annotated Driving Dataset を用いて各モデルの検出性能を比較し，物体クラス・サイズ別の精度も計測した．さらに，従来の FPN 機構との比較によって FSN の有効性を検証した．また，検出結果の可視化例も示す．

4.1. 実験環境・評価指標

Udacity Annotated Driving Dataset には物体のクラスとして car，truck，pedestrian，bicyclist，light の 5 クラスが存在し，18000 枚の訓練データ，4241 枚の未知データで構成されている．解像度は 480×300 であり，訓練データには物体の矩形領域およびクラスが与えられている．なお，ベースラインモデルは YOLOv4-tiny とし，計算量の多い YOLOv4 も加えて精度，速度，計算量の側面から比較を行った．また，SSD ベースの従来手法である SSD300，SSD7，FPSSD7，および AFPSSD7 に加えて，YOLO ベースの YOLOv3，YOLOv3-tiny との比較結果も示す．

精度の評価には The PASCAL Visual Object Classes Challenge に基づく VOC-mAP，COCO 2020 Object Detection Task [26] 基準の COCO AP を用いた．また，速度に関しては，GPU 使用時・未使用時の推論時間および FPS を用いた．ここで推論時間は，1 枚の画像をモデルに入力し，出力が得られるまでの平均処理時間と定義する．計算量の評価に関してはモデルの訓練パラメータ数および FLOPs (Floating-point OPerations) を用いた．これは，モデルの入力から出力までに必要な浮動小数点演算回数を意味し，1 秒当たりに実行可能な浮動小数点演算回数を示す FLOPS (FLoating-point Operations Per Second) とは異なる指標である．

開発環境として，GPU は NVIDIA Quadro RTX 8000，CPU

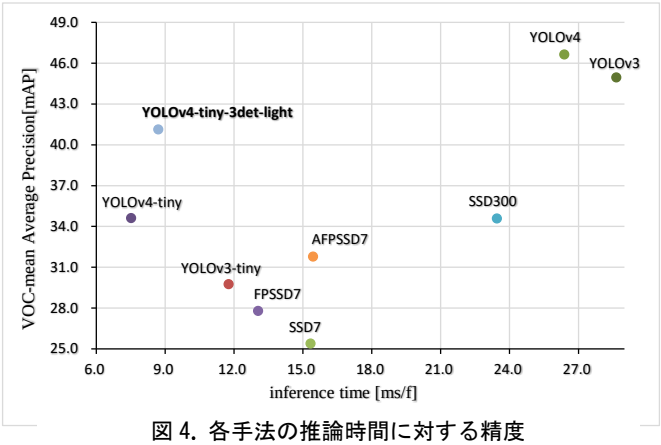


図 4. 各手法の推論時間に対する精度

は Intel(R) Xeon(R) Silver 4108 を用い，ライブラリは CUDA 10.1，cuDNN 7.6.4，TensorFlow 2.1 を使用した．なお，純粋なモデルのアーキテクチャの優劣を比較するため，量子化や枝刈り等を用いたモデルの圧縮は行っていない．

4.2. 検出性能比較実験

Udacity Annotated Driving Dataset を用いてモデルを訓練し，検出性能を比較した．提案手法の最適化手法には Adam [17] を使用し，バッチサイズは 16，学習率の初期値は 0.001，最大エポック数は 150 とした．なお，YOLOv4 の訓練方法に則り，1 エポック目は学習率を線形的に増加させる warm-up を用い，上限学習率の減衰には Cosine Annealing Scheduling [28] を用いた．また，損失関数も YOLOv4 と同様のものを用いた．推論時のバッチサイズは 1，入力解像度は 416×416，確信度閾値は 0.005 とした．ここで，確信度閾値とは NMS におけるハイパーパラメータであり，閾値より低い確信度のバウンディングボックスは統合前に破棄される．

表 1 に YOLOv4 および YOLOv4-tiny との比較結果を示す．提案手法はベースラインの YOLOv4-tiny に対して，VOC-mAP が 34.6 から 41.1 に，COCO-AP が 13.8 から 16.9 に向上し，YOLOv4 の精度に大きく近づいた．また，GPU 未使用時の推論速度は YOLOv4-tiny より僅かに高速な 27.349 FPS を記録した．これより，提案手法は GPU を用いずとも，高精度なリアルタイム検出が可能であることを示した．さらに，提案手法のモデルパラメータ数は 1.7M，FLOPs は 6.4 G で計算量が削減されており，従来手法に対して優れた手法であることが分かる．SSD や YOLOv3 等の従来手法との比較結果を表 2 に示し，横軸を GPU 推論時間，縦軸を VOC-mAP としてプロットした結果を図 4 に示す．これより提案手法は従来手法に対して精度と速度のトレードオフを改善しつつ，計算量の多い YOLOv3 に匹敵する高い精度を獲得できていることが確認できる．

4.3. 物体クラス・サイズ別精度の比較

表 3 に VOC-mAP における物体クラス別精度の比較結果を示す．提案手法は YOLOv4-tiny に対して，全クラスで精度の向上が見られ，bicyclist に関しては YOLOv4 を上回る精度を獲得した．また，表 4 に COCO-AP における物体サイズ別精度の比較結果を示す．提案モデルは，従来の FPN 機構を用いた場合においても，小物体精度(S-AP)と中物体精度(M-AP)の向上が見られ，FSN を導入することで，さらに精度が向上する結果を得た．一方，大物体精度(L-AP)に関しては YOLOv4-tiny を下回る結果となった．

表 4. 物体サイズ別精度の比較

architecture	VOC_mAP	COCO_AP	model_FLOPs[G]	model_params
FPN_concat	38.5	15.3	7.9	2,046,778
FPN_ew-sum	40.6	16.5	6.2	1,604,410
FPN_ew-product	40.5	16.5	6.2	1,604,410
PANet	40.0	17.0	7.7	2,492,218
FSN (proposed)	**41.1**	16.9	6.4	1,703,098

表 5.物特徴量補強機構の性能比較

method	S-AP	M-AP	L-AP	COCO-AP
YOLOv4	17.1	29.0	36.2	21.2
YOLOv4-tiny	10.0	22.1	30.2	13.8
YOLOv4-tiny-3det-light (FPN)	11.7	23.6	27.2	15.3
YOLOv4-tiny-3det-light (FSN)	12.9	26.9	27.2	16.9

4.4. 特徴量補強機構の性能比較

FSN の有効性を確認するため，従来の FPN および PANet との比較を行った．なお，FPN については従来の結合(Concatenation) に加えて，要素和，要素積による特徴マップの融合も比較した．表 5 に精度および計算量の計測結果を示す．まず，従来の結合を用いる FPN は精度が低いだけでなく，計算量も多いことが分かる．要素和や要素積を用いた FPN は比較的精度が高く，YOLOv4-tiny よりも少計算量だが，COCO-AP は PANet に劣っている．これに対して FSN を用いた場合，最高精度の VOC-mAP と PANet に匹敵する COCO-AP を獲得しつつ，6.4 GFLOPs の比較的少ない計算量であった．これより，FSN が精度と計算量の両面において優れた機構であることが示された．

4.5. 検出結果の可視化

検出結果の可視化例を図 5 に示す．左から YOLOv4-tiny，FSN 未導入（従来の FPN を使用）の提案手法，FSN を導入した提案手法である．ここで，car (車)が赤色，truck (トラック)が黄緑色，pedestrian (歩行者)が緑色，light (信号機)が紫色の矩形領域に対応する．まず一段目の例において，YOLOv4-tiny は信号機の検出漏れが見られるが，提案手法では改善されている．さらに FSN を導入することで，全ての信号機を検出できているだけでなく，画像右端に映る車の重複(オクルージョン)を識別し，複数台の車の検出に成功している様子が見て取れる．二段目の

YOLOv4-tiny　　提案手法（FSN なし）　　提案手法（FSN あり）

図 5. 検出結果の可視化例

例においては, YOLOv4-tiny では画像中央部に歩行者の誤検出, 右下に車の誤検出が見られる．また，FSN 未導入の提案手法は歩行者の検出漏れは改善しているが，車の誤検出は改善されていない．これに対して，FSN を導入した提案手法はこれらの誤検出を改善できている．三段目の例では，YOLOv4-tiny および FSN 未導入の提案手法で検出できなかった歩行者を検出できている様子が確認できる．

5. 考察

表 4 や図 5 より，提案手法は特徴マップのチャンネル次元数を削減しているにも関わらず, 小物体の精度向上が見られた．これは, 高解像度な特徴マップを物体の予測に用いたことで, 小物体の検出が容易になったためであると推測される．そのため, 自動運転シーンの物体検出においては，特徴マップのチャンネル次元数よりも解像度の方を優先すべきであると考えられる．また, 小規模なパラメータの探索空間が, 比較的少クラスな物体の分類に適していたことも要因の一つと思われる．一方で, 大物体の検出精度は低下しているが, これは 13×13 の特徴マップのチャンネル次元数が少なく，大物体の分類が困難になったためと推測される．

FSN を導入することで, さらに小物体の検出精度が向上した．これは，Large Scale Feature が Small Scale Feature の意味的な特徴量を直接利用できたことが，精度の向上に寄与したためと考えられる．また, 中物体は特に大きく精度が向上していた．これは，中物体の検出を行う Middle Scale Feature が深いレベルの特徴量だけでなく，浅いレベルの空間的な特徴量も必要としており，FSN によってこれを補強できたためと考えられる．一方, 大物体に関しては精度の向上が見られなかった．これは, 大物体の検出を行う Small Scale Feature では空間的な特徴量の必要性が低いと考えられ，この改善には Backbone 部の段階でネットワークチューニングが必要になると思われる．

6. おわりに

本稿では，軽量かつ高精度な検出手法として YOLOv4-tiny-3det-light を提案し，従来の FPN に代わる新たな特徴量補強機構として Feature Sharing Network (FSN)を提案した．車載カメラ画像の Udacity Annotated Driving Dataset を用いた比較実験では，YOLOv4-tiny の計算量を削減しつつ精度の向上を確認した．特に, 小物体および中物体の検出精度が向上しており, 検出結果を可視化することで誤検出や検出漏れの改善を確認した．さらに，従来の FPN 機構との比較を行った結果，FSN は比較的少ない計算量で PANet に匹敵する精度を獲得できることを示した．今後の課題としては，大物体の精度向上，クラス数の多いデータセットでの精度向上，モバイル端末や安価なマシンを用いたベンチマーク等が挙げられる．

参 考 文 献

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. “You Only Look Once:Unified, Real-Time Object Detection”, In CVPR, 2016, pp. 779-788

[2] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao. “YOLOv4: Optimal Speed and Accuracy of Object Detection”, arXiv preprint arXiv: 2004.10934, 2020

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. ”SSD: Single Shot MultiBox Detector”, In ECCV, 2016, pp. 21-37

[4] T. Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. “Feature Pyramid Networks for Object Detection”, In CVPR, 2017, pp. 2117-2125

[5] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. “Rich feature hierarchies for accurate object detection and semantic segmentation”, In CVPR, 2014, pp. 580-587

[6] R. B. Girshick. “Fast R-CNN”, In ICCV, 2015

[7] S. Ren, K. He, R. B. Girshick, and J. Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, In NIPS, 2015, pp. 91–99

[8] K. He, G. Gkioxari, P. Doll´ar, R. Girshick. “Mask R-CNN”, In ICCV, 2017, pp. 2961-2969

[9] J. Redmon and A. Farhadi. “YOLO9000:Better, Faster, Stronger”, In CVPR, 2017, pp. 7263-7271

[10] J. Redmon and A. Farhadi. “yolov3: An incremental improvement”, arXiv preprint arXiv:1804.02767, 2018

[11] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. ”DSSD : Deconvolutional single shot detector”, arXiv: 1701.06659, 2017

[12] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li. “Single-Shot Refinement Neural Network for Object Detection”, In CVPR, 2018, pp. 4203-4212

[13] Q. Zhao, T. Sheng, Y. Wang1, Z. Tang, Y. Chen, L. Cai, and H. Ling. “M2Det: A Single-Shot Object Detector based on Multi-Level Feature Pyramid Network”, In AAAI, 2019

[14] M. Tan, R. Pang, and Q. V. Le. “EfficientDet: Scalable and Efficient Object Detection”, In CVPR, 2020, pp. 10781-10790

[15] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. “Path Aggregation Network for Instance Segmentation”, In CVPR, 2018, pp. 8759-8768

[16] G. Ghiasi, T. Y. Lin, and Q. V. Le. “NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection”, In CVPR, 2019, pp. 7036-7045

[17] D. P. Kingma, J. Ba. “Adam: A Method for Stochastic Optimization”, In ICLR, 2015

[18] M. Everingham, L. J. V. Gool, C. K. I. Williams , J. M. Winn , and A. Zisserman. ”The PASCAL Visual Object Classes (VOC) Challenge", In IJCV, 2010, 88(2), pp.303-338

[19] K. Simonyan, and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”, In ICLR, 2015

[20] Y. Yamashige, and M. Aono. “FPSSD7: Real-time Object Detection using 7 Layers of Convolution based on SSD”, In ICAICTA, 2019

[21] 山重雄哉, 青野雅樹. “Attention と意味情報補強処理による高速な物体検出手法の提案”, DEIM, 2020

[22] Pierluigiferrarr. “keras_ssd7.py” [Online] Available: https://github.com/pierluigiferrari/ssd_keras/tree/master/models/keras_ssd7.py, [Accessed: 20-December 2020]

[23] AlexeyAB. “yolov3-tiny.cfg” [Online] Available: https://github.com/AlexeyAB/darknet/blob/master/cfg/yolov3-tiny.cfg, [Accessed: 20-December 2020]

[24] AlexeyAB. “yolov4-tiny.cfg” [Online] Available: https://github.com/AlexeyAB/darknet/blob/master/cfg/yolov4-tiny.cfg, [Accessed: 10-December 2020]

[25] “ Udacity Annotated Driving Dataset” [Online] Available:https://drive.google.com/open?id=1tfBFavijh4UTG4cGqIKwhcklLXUDuY0D, [Accessed: 10-December 2020]

[26] “COCO 2020 Object Detection Task” [Online] Available: https://cocodataset.org/#detection-2020, [Accessed: 8-December 2020]

[27] 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

[28] I. Loshchilov, and F. Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts”, In ICLR, 2017

[29] C.Y. Wang, H. Y. M. Liao, Y. H. Wu, P. Y. Chen, J. W. Hsieh, and I. H. Yeh. “CSPNet: A New Backbone That Can Enhance Learning Capability of CNN”, In CVPR, 2020, pp. 390-391

[30] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. “Focal Loss for Dense Object Detection”, In ICCV, 2017

[31] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren. “Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression”, In AAAI, 2020

[32] K. He, X. Zhang, S. Ren, and J. Sun. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2015,37(9):1904–1916