エッジコンピューティングを用いた交通量計測システムの性能評価

椎名昂[1]・柳沼秀樹[2]・寺部慎太郎[3]・海野遥香[4]・鈴木雄[5]

[1]学生非会員　東京理科大学　創域理工学部社会基盤工学科（〒 278-8510 千葉県野田市山崎 2641）
E-mail: 7620052@ed.tus.ac.jp
[2]正会員　東京理科大学准教授　創域理工学部社会基盤工学科（〒 278-8510 千葉県野田市山崎 2641）
E-mail: yaginuma@rs.tus.ac.jp (Corresponding Author)
[3]正会員　東京理科大学教授　創域理工学部社会基盤工学科（〒 278-8510 千葉県野田市山崎 2641）
E-mail: terabe@rs.tus.ac.jp
[4]正会員　茨城大学大学院助教　理工学研究科（〒 316-8511 茨城県日立市中成沢町 4-12-1）
E-mail: haruka.uno.uj38@vc.ibaraki.ac.jp
[5]正会員　東京理科大学助教　創域理工学部社会基盤工学科（〒 278-8510 千葉県野田市山崎 2641）
E-mail: yusuzuki@rs.tus.ac.jp

昨今の AI を活用した画像解析技術は，データの取得から利活用に至るまでのプロセスに大きな変革を与えており，様々な分野で適用が意欲的に進められている．交通分野においては，車両の自動検出や追跡による断面交通量の常時観測が実現しつつある．一方，CCTV カメラ動画の送受信による通信量や AI 解析処理の計算量が増大により，サーバーへの負荷が大幅に上昇する．これにより現状の集中処理システムでは性能低下およびコストの増加が実務上の問題となる．本研究では，エッジコンピューティング技術を活用した分散処理型システムを導入することで課題の解決に取り組む．複数の市販デバイスと AI モデルを組み合わせたエッジシステムを構築し，交通量計測の処理速度や検出精度等の比較検証を通じて，システムの有効性に関する基礎的な検証を行う．その上で，交通量計測に特化したエッジシステムの設計・開発に資する知見を得ることを目指す．

Key Words: *Traffic Census, Edge Computing, Convolutional Neural Network (CNN), GPGPU*

1. はじめに

AI を活用した画像解析技術は，データの取得から利活用に至るまでのプロセスに大きな変革を与えている．Hinton ら[1)2)] が Deep Neural Network (DNN) による学習の弱点を克服する手法を提案したことにより訪れた第三次人工知能ブームの中で，画像中から対象の物体を検出する画像解析の技術も大きく進歩してきた．AI モデルの性能向上に伴って大規模かつ高速な計算処理を行うことができる環境が必要となり，現在多くの AI 研究が高性能サーバや大規模データセンタを用いるクラウド集約型アーキティクチャの形を取っている．

国土交通省は，最先端の情報通信技術を用いて人・道路・車両の情報を集約することにより、交通事故、渋滞などといった道路交通問題を解決するすることを目的とした高度道路交通システム (ITS[3)]) の一環として全国の道路上に約 14,000 台の CCTV カメラを設置している．CCTV カメラで撮影された映像は，道路の維持管理を目的としたモニタリング等に利用されており，撮影データはストリームデータの受信時に動画形式へと変換し保管されている．これらの動画像に対し，今井ら[4)] の断面交通量の常時観測などを目標にした車両検出・追跡に関する研究は，東ら[5)] が示した近年研究が加速する ICT 技術による道路交通センサスの無人観測手法の一つとして注目されている．

しかし CCTV カメラを用いて各地点における断面交通量や車両速度などの交通情報を常時観測し記録するためには，システムに対してリアルタイムでの処理の実行が要求される．現在主流のクラウド集約型のシステムを用いてリアルタイムの画像処理を実現するために，サーバには高度な処理能力，保管能力に加え安定した動作が同時に求められる．

そこで，本研究では Edge Computing(EC) による画像処理の導入を試みる．EC によってセンサレベルで画像処理を行うことによって，サーバへ送付するデータを動画像データから検出結果のみに置換し，データ容量を大幅に削減することが可能である．EC はセンサ単位で処理を行うため，エッジサーバの故障により全てのセンサに対して画像処理が停止してしまうことなども生じない．また，センサ単位の処理により，システム拡張のために新たにセンサを増設する際にもメインサーバの負荷を考慮する必要が殆ど無い．更に交通量のカウントや車種分類の記録のみを検出結果としてメインサーバに保存することによって，動画像に記録されて

しまうナンバープレートや顔などの個人情報を排除することができ，情報の秘匿性も向上する．

ただし，EC による交通量計測システムをクラウドコンピューティングへの代替として構築する際にも，システムと同様の要件が課されるが，EC の実務への適用可能性は定量的に評価されていない．

本研究では，CCTV カメラで撮影されたストリーム画像に対して行う常時観測の実現に向けた第一歩として，EC によるリアルタイムでの物体検出の実現可能性についての評価を行う．

2. 既往研究の整理と研究の位置づけ

EC とは，ネットワーク内部のセンサに最も近いエッジ部分にサーバを分散して配置する技術の総称である．飯田[6] が示すように，第 5 世代携帯電話網 (5G) を利用した超低遅延通信サービスの実現，IoT 技術の発展に伴うエッジセンサの増加などに伴い EC の研究は近年急速に活性化している．畑ら[7] のエッジサーバを用いた動画像処理の実行時間を CPU の性能差から予測する研究，塩田ら[8] によるセンサで得られたデータをエッジサーバとクラウドサーバで分散して処理することで AR システムを高速化する研究など，特にトラフィックが多くなる動画像データへの処理に着目した研究が多くなされている．

また，エッジサーバの利用によるトラフィックの削減はデータの分散収集によるグローバル学習の阻害をもたらす．これを解決するために Niwa ら[9] が行った大量のデータを多くのエッジサーバーに分散蓄積したままでもモデル学習を可能にする技術の研究，荻野ら[10] によるエッジサーバのローカル最適化とクラウドサーバのグローバル最適化のバランスを取り，システムアーキテクチャを動的に最適化する研究など，EC に特有のシステム形状に対する研究も盛んに行われている．

さらに Szen ら[11] によるエッジサーバに利用可能な小型デバイスの種類による性能差を CNN を用いて比較した研究や，Basulto-Lautsova ら[12] によるテンプレートマッチングを用いて比較した研究など，ハードウェアに着目した研究も見られる．

実際に EC による車両検出において，その速度や精度は利用の可否に大きな影響を与える．

本研究は，検出モデルやエッジサーバとなるデバイスを用いて車両検出を行う際の精度や速度を比較することで，CCTV カメラに対する EC を用いた常時車両検出システム構築の技術的可能性を検討する．

3. 物体検出モデル

(1) YOLO-v3 / YOLO-v3-tiny

You Only Look Once(YOLO)[13] は高速に動作する物体検出アルゴリズムの一つであり，YOLO-v3 をはじめとするその後継モデルは現在の画像解析においてデファクトスタンダードとなっている．YOLO は，Convolutional Neural Network(CNN) を利用した画像解析技術であるが，画像の領域推定と分類を同時に行うことにより単一の CNN ネットワークによる物体検出を可能にした．

YOLO-v3[14] は 53 層の CNN により構成されている．入力画像をグリッドセルに分割し，各グリッドの内部に存在する物体が属するクラスをロジスティック回帰によって推定する．同時に k-means 法を用いてグリッド毎に推定したスケールやアスペクト比の異なる複数の Bounding-box(B-box) を，位置を表す座標値と物体が含まれる確率を表す信頼スコアによって示す．

YOLO-v3 は 3 次元のテンソルによってスケールの違うデフォルト B-box を提案するので，スケールのちがう物体の検出に対して頑健である．また，全結合層を用いないことによって任意の画素数の入力画像に対応することが可能である．

YOLO-v3 はより軽い負荷で実行することができる tiny モデルも提案している．YOLO-v3-tiny モデルは 13 層の CNN であり，YOLO-v3 よりも精度は劣るものの高速に物体検出を行うことができる．

YOLO-v3 は入力画像の画素数を問わない汎用性の高さにくわえて，高速に動作する単一の CNN ネットワークによって高速な処理が可能である．またスケールの違いに対する頑健さは車両の遠近の違いに依存しないモデル精度の確保に寄与する．以上を踏まえて本研究では YOLO のモデルから YOLO-v3 および YOLO-v3-tiny を利用することとする．

4. エッジコンピューティングデバイス

本研究では，NVIDIA®が提供する Jetson™ プラットフォームをエッジサーバとして使用する．Jetson™ は NVIDIA®が AI を用いた自律マシンの開発・展開を想定して提供する Graphics Processing Unit(GPU) を搭載した組み込み型シングルボードコンピュータである．搭載される GPU などの性能の違いにより複数のラインを展開しており，本研究ではこれらのシリーズの中から後述するモデルを利用する．

AI は，繰り返し行われる大規模な行列計算により高い計算負荷を生じさせる．GPU は軽度な計算を大量に並列処理することを目的として Central Processing Unit(CPU) と比較して小さなコアが大量に搭載されている．この

表–1 Jetson Nano の性能表[18)]

	Jetson Nano
GPU(Core)	Maxwell™(128)
CPU	Quad-core ARM Cortex-A57
Memory	4GB LPDDR4

表–2 動画データの詳細

場所	茨城県潮来市洲崎 国道 51 号 神宮橋
撮影日	2021/01/19
天候	晴れ
画素数	1920 × 1080
動画時間	10:00
jpeg 画像	8990 枚

GPU に備わった計算リソースを映像処理以外に利用する General-Purpose computing on Graphics Processing Units(GPGPU)[15)] と呼ばれる技術の利用は，画像解析をはじめとした並列での行列計算を必要とするアーキテクチャに対して有用性が高く，YOLO においても GPGPU によって処理速度が大幅に向上することが示されている[16)17)]．

Compute Unified Device Architecture(CUDA) もまた NVIDIA®が提供するサービスであり，NVIDIA が開発する GPU において GPGPU の実行を記述することができる開発プラットフォームである．本研究においても GPGPU を利用するために CUDA を利用している．

本研究では，Jetson™ から Nano developer Toolkit を用いて比較を行うこととする．代表的な性能を**表–1** に示す．

5. CCTV カメラ画像を用いた数値実験

(1) 分析データの概要

本実験には茨城県 神宮橋に設置されている CCTV カメラによって撮影された動画データを用いる．使用した動画データの詳細な情報を**図–2** に示す．また画角を動画の 1 フレーム目を用いて**図–1** に示す．

本研究はストリームデータに対する処理を最終的な目的としており，想定される環境では動画データを画像データに変換する工程は発生しない．よって本実験では，事前に動画をフレーム毎に jpeg 画像として書き出したのち，画像に対して物体検出を行う．また，モデルのトレーニングによる精度向上は本研究の目的とするところではないため，Microsoft cocodataset[19)] による学習済みモデルを使用する．



図–1 動画データから作成したフレーム画像

YOLO-v3 および YOLO-v3-tiny の学習済みモデルは，80 種の class を分類することが可能である．本実験では ‘car’,‘motorbike’,‘truck’,‘bus’ の 4 種類の class を検出対象とした．

(2) 実験内容

物体検出モデルおよびデバイスを変化させて物体検出を実行する．推定された検出結果に対し，Average Precision(AP) から算出される mAP(mean AP) を用いた精度比較や, 実行時間やシステムへの負荷などをモニターすることによってシステムの有用性を検討する．本実験において，稼働時間は time コマンドを，システム稼働状況は NVIDIA が提供する tegrastats Utility[20)] を用いた．

検出精度を評価するために用いる mAP は，物体検出をはじめとする多クラス分類問題に対する評価指標である．クラスごとの適合率および再現率から算出される Average Precision(AP) の平均によって計算される．

EC を用いたリアルタイムの交通量計測システムには，断面交通量を常時観測し続けられる検出速度と車両台数を適切にカウントする精度が要求される．また，地点ごとにシステムを導入することを考慮すると，より安価なデバイスでの実装が求められる．よって本実験では，現状の軽量なモデルおよび安価なデバイスで実装可能な交通量計測の速度および精度の確認を行う．

6. 数値実験の結果と考察

(1) デバイス実行時間

それぞれのモデルにおける実行時間は**表–3** であった．

ここで，real は動作にかかった実際の時間，user は CPU が動作している時間を示している．また，FPS は実際にかかった時間を秒数に換算した $time_{real}[sec]$ および解析した画像の枚数 $frames$ を用いて式 (1) によっ

表–3 Jetson Nano におけるモデルごとの実行時間

	YOLO-v3	YOLO-v3-tiny
real	116m27.715s	55min14.443s
user	49m59.036s	46m 2.176s
FPS	1.29 f/sec	2.71 f/sec

て算出した．

$$FPS = \frac{time_{real}}{frames} \tag{1}$$

この結果から YOLO-v3-tiny は YOLO-v3 と比較しておよそ 2 倍の速さで検出を行うことができていることが確認できる．しかしながら，いずれの処理速度であっても見通しが悪い場所や車両が高速域で走行する場所などでは処理を行っている間に車両が通過することによる検出漏れが出てしまう可能性がある．今回の実行環境はいずれも十分な処理速度を有しているとは言えない結果となった．

(2) デバイスシステム負荷

モデル実行中における CPU,GPU および RAM の動作状況を，モデル別に**図–2** に示す．

CPU はコアによる著しい処理の偏りはなく，適切に分散処理が行われていると判断できる．CPU は最大負荷での連続使用がされていないことが確認でき，YOLO-v3 および YOLO-v3-tiny の両モデルにおいて，処理のボトルネックになっているとは判断できない．

GPU は 100%に近い使用率で動作していることが確認でき，GPGPU としてハードウェアリソースが利用されている．一方，GPU がほぼ最大負荷で連続動作していることから検出速度のボトルネックが GPU にあるとも考えられる．

RAM は使用量が両モデルともほぼ一定で推移している．YOLO-v3-tiny は解析の開始直後には YOLO-v3 とほぼ同じ値を取るが，すぐに減少したのち，YOLO-v3 よりも 400MB 程度小さい値で安定動作していることが確認できる．いずれのモデルにおいても，最大メモリ容量の 3964MB には到達していないことが確認でき，検出速度のボトルネックになっているとは判断できない．

(3) 物体検出精度

検出モデルごとの class 別 AP および mAP を**図–3** に示す．な，mAP の算出にあたり，Interesection Over Unioon(IoU)[21]=0.5 としている．また，正解データとしてより検出精度の高い YOLO-v5 を利用した．

YOLO-v3 の mAP は 27.76%と決して高くないものの，車両に対して一定の精度を確認した．最も大きい AP の値を取った class である ‘car’ は，AP=58%であった．その一方で YOLO-v3-tiny は，YOLO-v3 に対して mAP の値が著しく低くなった．‘car’ においても AP=1%と，正常に検出できているとは言えない結果となった．ここで，正解データおよび各モデルにおける class の検出数を**表–4** に示す．

表–4 モデルごとの検出 class

	YOLO-v5 (Ground Truth)	YOLO-v3	YOLO-v3 -tiny
car	11290	11969	1178
motor cycle	1	0	0
bus	717	1493	74
truck	4907	5401	850

表–4 から YOLO-v3-tiny の物体の検出数は，YOLO-v3 の検出数に対して class を問わず大きく劣っていることが確認できる．対象物体の検出数が正解データの 1/10 程度であったことも mAP の値を小さくした要因と考えられる．

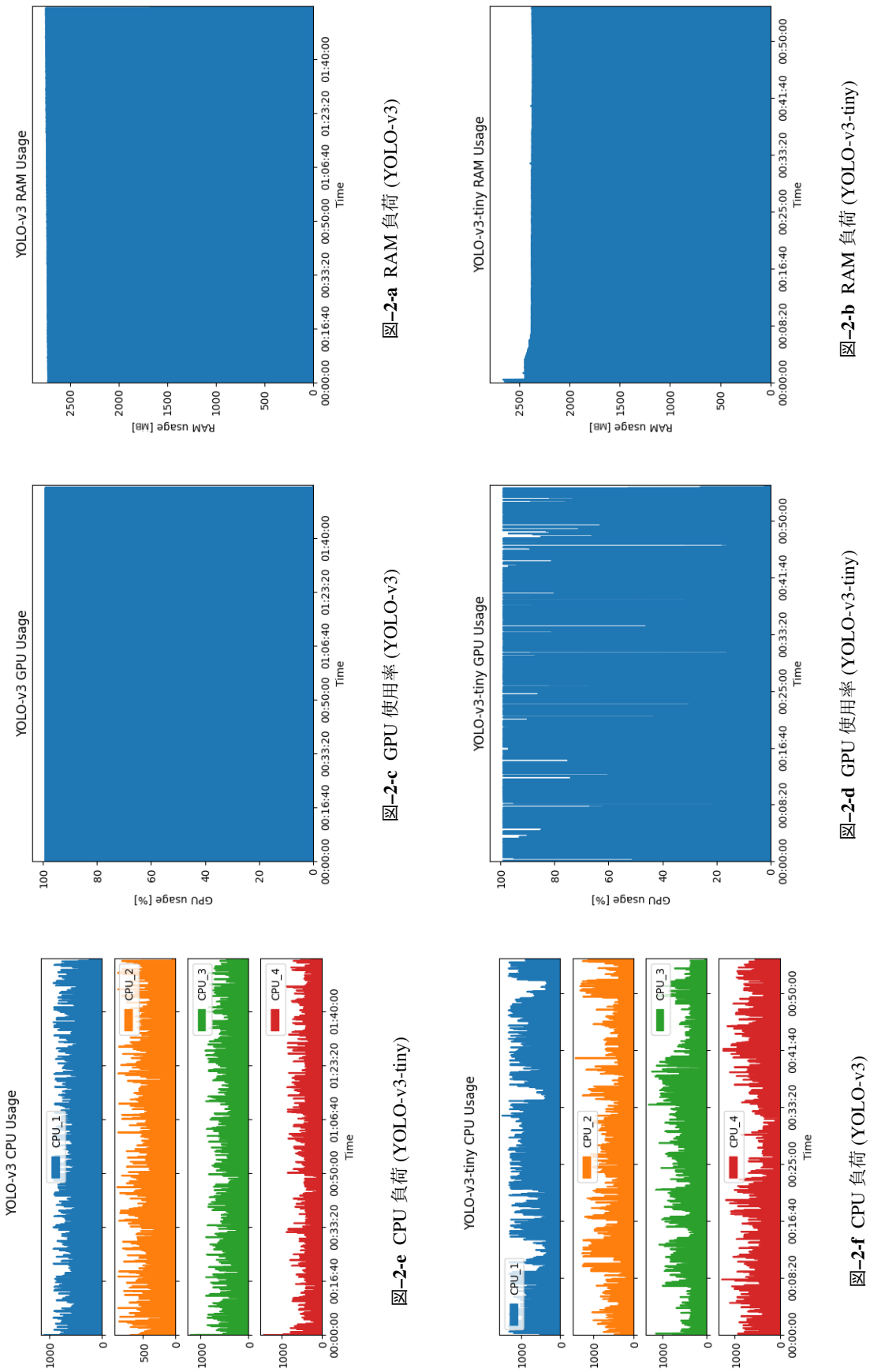
検出した物体の大きさやその精度をモデルごとに比較したものを**図–4** に示す．横軸に用いた $\sqrt{S}$ は検出された B-box の横幅 w と高さ h を用いて式 (2) により算出した．また横軸の階級幅を 100 としている．縦軸は検出された B-box の総数である．

$$\sqrt{S} = \sqrt{w \times h} \tag{2}$$

今回用いた YOLO のモデルは検出した B-box に対する出力値として，その B-box の信頼度を示す confidence を百分率で出力する．階級幅を 10%として**図–4** に同時に示した．今回使用したモデルにおいて，B-box を出力する confidence の下限値は 0.25 である．

YOLO-v3 と YOLO-v3-tiny で B-box の大きさの違いによる検知数の比率は大きく異なってはいない．しかし YOLO-v3-tiny では，$\sqrt{S} < 100$ の範囲において検出された B-box のうち，約 2/3 の B-box で confidence の値が 40%を下回る結果となった．これは，畳み込み層が少なくなったことにより，小さく映る遠くの物体の特徴量把握が不十分になったことが原因であると考えられる．

mAP の値から，YOLO-v3-tiny の学習済みモデルを利用した車両検出は現状では非常に厳しいと言える．YOLO-v3 は実用に対して十分とは言えないものの，YOLO-v3-tiny と比較すると高い mAP を記録した．



図–2-a RAM 負荷 (YOLO-v3)

図–2-b RAM 負荷 (YOLO-v3-tiny)

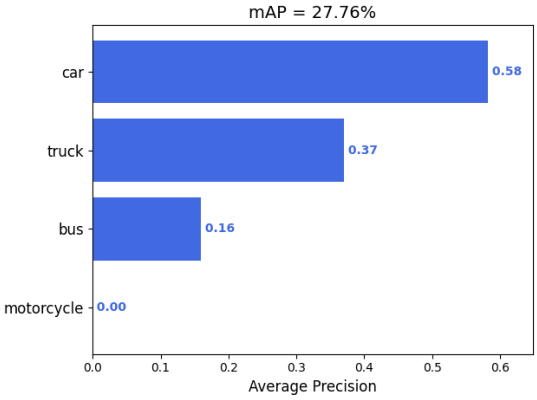
図–2-c GPU 使用率 (YOLO-v3)

図–2-d GPU 使用率 (YOLO-v3-tiny)

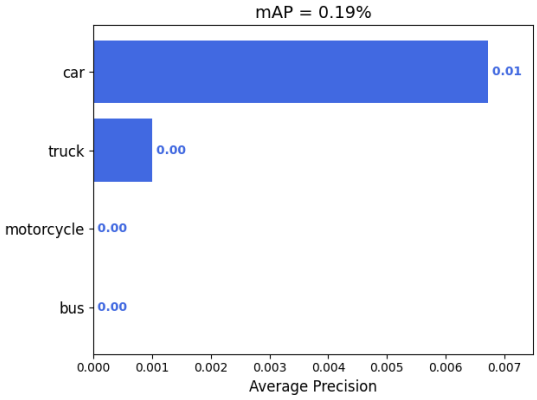
図–2-e CPU 負荷 (YOLO-v3-tiny)

図–2-f CPU 負荷 (YOLO-v3)

図–2 モデル別システム負荷

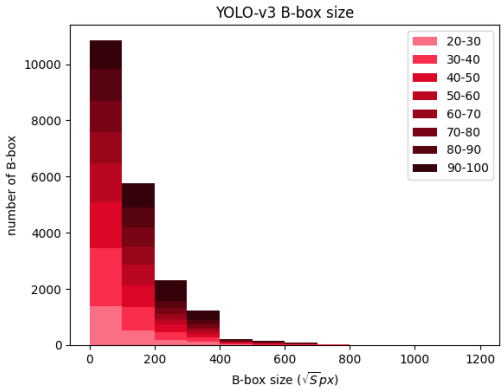


図–3-a mAP YOLO-v3

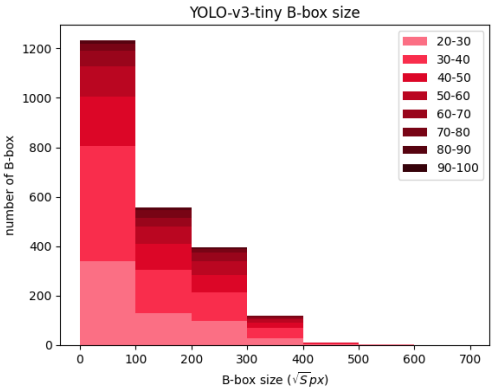


図–3-b mAP YOLO-v3-tiny

図–3 モデルごと Class 別 AP



図–4-a B-box and confidence YOLO-v3



図–4-b B-box and confidence YOLO-v3-tiny

図–4 モデルごと B-box の大きさと accuracy

7. おわりに

(1) 結論

本論文では，市販のエッジデバイスである Jetson Nano によって構築されるシステムによって画像解析によるリアルタイムの車両の検出が可能か否かを検証した．くわえて軽量な検出モデルである YOLO-v3, YOLO-v3-tiny を用いて現状の車両に対する検出精度が断面交通量の計測に利用可能かどうかを，モデル間の比較を通して検討した．その結果，YOLO-v3 は class によっては既存の学習済みモデルでも，高い mAP の数値が確認された．一方，YOLO-v3-tiny は class を問わず検出精度が低く，特に小さい検出対象に対する検出が YOLO-v3 よりも低精度であると考察される．ただし，車両の検出漏れなどの観点から考えると，リアルタイムで常時検出を実行することは両モデルとも難しく，特に YOLO-v3 は軽量なモデルである YOLO-v3-tiny に対して約 2 倍の実行時間が必要であった．またシステムの使用状況から，本論文で用いた環境では GPU の性能がボトルネックとなっていると考えられる．検出の速度を向上させる，またはより高度なモデルを使用することにより検出速度を低下させずに検出精度を向上させるためには GPU の性能向上を第一に考える必要がある．

(2) 今後の展望

EC によるリアルタイム交通量計測システムの実現に向け，要求される検出速度や検出精度を過不足なく満たすモデルを把握するために，エッジデバイスおよび検出モデルの双方で，対象を増やした比較検討が必要である．また，転移学習によって計算負荷を増加させることなく精度向上を図ることができるかどうかを検証する必要がある．その際，エッジデバイスの複数ヶ所への設置を考慮し，地点に依存しない検出精度の向上が図れるかが重要となる．また，想定される実装環境により近い状態の再現として，外部サーバへのデータ送信やストリームデータに対する処理を想定したネットワークを構築しての検証，評価なども求められる．

謝辞

本研究は国土交通省新道路技術会議「カメラ画像および複数の観測データを融合した次世代交通計測手法に関する研究開発」の助成を受けたものです．ここに記して感謝いたします．

参考文献

1) Hinton, Geoffrey E and Osindero, Simon and Teh, Yee-Whye: A fast learning algorithm for deep belief nets, *Neural computation*, Vol.18, No.7, pp.1527–1554, 2006.
2) Hinton, Geoffrey E and Salakhutdinov, Ruslan R: Reducing the dimensionality of data with neural networks, *science*, Vol.313, No.5786, pp.504–507, 2006.
3) 国土交通省: 道路交通センサスのデータ収集の現状と課題, *ICT* を活用した新道路交通調査体系検討会, 2018.
4) 今井龍一, 神谷大介, 山本雄平, 田中成典, 中原匡哉, and 中畑光貴: 汎用的な深層学習器を用いた交通量調査手法に関する基礎的研究, 土木学会論文集 *F3 (*土木情報学*)*, Vol.75, No.2, pp.I_150–I_159, 2019 [A basic study on traffic census using generic deep learning].
5) 東俊孝, 高田知典, 井坪慎二, and 内田淳: 道路交通センサスのための次世代情報収集システムの開発, 土木情報システム論文集, Vol.15, No.0, pp.103–110, 2006 [Development of a next generation information gathering system for road traffic censuses].
6) 飯田勝吉: エッジコンピューティング研究開発の現状と今後の課題, 電子情報通信学会技術研究報告, Vol.117, No.187, pp.25–30, 2017 [Current R and D Status of Edge Computing and its Future Direction].
7) 畑徹, 棟朝雅晴, et al.: 隠消現実感アプリケーションに基づくエッジコンピューティングの性能推定, 研究報告数理モデル化と問題解決 *(MPS)*, Vol.2018, No.23, pp.1–2, 2018 [Performance Estimations of Edge Computing with Diminished Reality Applications].
8) 塩田純, 滝澤允, 田中裕之, 高橋紀之, 小林英嗣, et al.: 画像処理におけるエッジコンピューティングを用いた垂直分散処理方式の検討, 研究報告マルチメディア通信と分散処理 *(DPS)*, Vol.2016, No.2, pp.1–7, 2016 [A Study of Vertical Distributed Image Processing based on Edge Computing].
9) Niwa, Kenta and Harada, Noboru and Zhang, Guoqiang and Kleijn, W Bastiaan: Edge-consensus learning: Deep learning on p2p networks with nonhomogeneous data, *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 668–678, 2020.
10) 荻野正, 北上眞二, 白鳥則郎, et al.: フレキシブル iot エッジコンピューティングモデルと応用, 研究報告グループウェアとネットワークサービス *(GN)*, Vol.2018, No.1, pp.1–8, 2018 [Flexible IoT Edge Computing Model and Its Applications].
11) Szen, A. A., Duman, B., and en, B.: Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-cnn, *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pp. 1–5, 2020.
12) Basulto-Lantsova, A., Padilla-Medina, J. A., Perez-Pinal, F. J., and Barranco-Gutierrez, A. I.: Performance comparative of opencv template matching method on jetson tx2 and jetson nano developer kits, *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0812–0816, 2020.
13) Redmon, Joseph and Divvala, Santosh and Girshick, Ross and Farhadi, Ali: You only look once: Unified, real-time object detection, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
14) Redmon, J. and Farhadi, A.: Yolov3: An incremental improvement, 2018.
15) Ghorpade, J.: GPGPU processing in CUDA architecture, *Advanced Computing: An International Journal*, Vol.3, No.1, pp.105–120, jan 2012.
16) Qiang, Z., Yuanyu, W., Liang, Z., Jin, Z., Yu, L., and Dandan, L.: Research on real-time reasoning based on jetson tx2 heterogeneous acceleration yolov4, *2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, pp. 455–459, 2021.
17) Shin, D.-J. and Kim, J.-J.: A deep learning framework per-

formance evaluation to use yolo in nvidia jetson platform, *Applied Sciences*, Vol.12, No.8, 2022.
18) Nvidia, 2023 [https://developer.nvidia.com].
19) Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Doll'a r, P., and Zitnick, C. L.: Microsoft COCO: common objects in context, *CoRR*, Vol.abs/1405.0312, 2014.
20) tegrastats utility.
21) Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

(Received ?)
(Accepted ?)

PERFORMANCE EVALUATION OF TRAFFIC COUNTING SYSTEM USING EDGE COMPUTING

Kou SHIINA, Hideki YAGINUMA, Shintaro TERABE and Haruka UNO

Improvements in image analysis technology using AI have led to significant changes in the data to be analyzed and the methods used in prediction models in a short period of time. In the field of traffic engineering, it has become possible to constantly observe the amount of traffic by automatically detecting and tracking vehicles in images. However, constant video observation increases the amount of data traffic to send and receive images. The current centralized processing system is insufficient for server performance due to the increased load. However, installing a server with sufficient performance is difficult due to its high cost. This study aims to solve this problem by introducing a distributed processing system using edge computing. We will construct an edge system that combines several commercially available devices and AI models, and verify whether the system is practical by comparing processing speed and detection accuracy under the same conditions.