Vol.2018-UBI-59 No.11
Vol.2018-ASD-13 No.11
2018/9/1

CityFlow: 超分散エッジ環境における機械学習システム実現のための知的都市開発環境

河野 慎[1,a)] 米澤 拓郎[1,b)] 谷村 朋樹[1,c)] Giang Nam[3,d)] Broadbent Matthew[2,e)]
Lea Rodger[2,f)] 中澤 仁[1,g)]

概要：都市から収集したデータからの都市の状況推定やイベント検出などの研究が盛んに試みられている．これらの研究の多くは，都市の課題を機械学習タスクとして定式化し，実際に設置した物理センサや SNS などの Web サービスからデータを収集し，機械学習モデルの構築により行われている．都市の課題へ機械学習適用による実運用解決を試みた際，次の問題が生じる．1）データの収集・処理，2）機械学習ライフサイクル，3）時空間的特徴による共変量シフト．本研究では，これらの問題を解決する開発環境 CityFlow の設計と実装を行う．CityFlow では，処理やハードウェアをノードで表現し，タスク間でデータの流れやエッジデバイスの処理の記述が容易になる．本稿では，道路の損傷検出と参加型センシングにおける自動ラベリングをユースケースとして挙げ，CityFlow の有用性について議論する．

CityFlow: Smart City IDE for Machine Learning Flow with Distributed Edge Computing

Kawano Makoto[1,a)] Yonezawa Takuro[1,b)] Tanimura Tomoki[1,c)] Giang Nam[3,d)]
Broadbent Matthew[2,e)] Lea Rodger[2,f)] Nakazawa Jin[1,g)]

1. はじめに

スマートシティの実現には，都市の状況把握や様々な課題解決などが求められる．多くの研究者は，スマートシティ実現に向けて，機械学習を用いた状況推定や課題解決を試みてきた．例えば，市民がもつスマートデバイスの位置情報やタクシーの GPS 情報から都市における市民の行動予測を行う研究 [12] や，自動車に搭載されたドライブレコーダーの映像から道路の損傷を検出する研究 [6] などが挙げられる．これらの研究は，クラウドコンピューティング技術などを用いて，一箇所にデータを集中させて分析することを想定している．一方で，NVIDIA 製 Jetson TX2 [*1]をはじめとした組み込みコンピュータや FPGA などが小型化・高性能化してきたことや，様々なものがインターネットに接続される Internet of Things（IoT）の普及に伴い，エッジコンピューティング技術が注目されている．都市には，自動車をはじめ，市民が持つモバイルデバイス，ロボットなど多種多様なものが分散して存在しており，相互通信によるコミュニケーションなどが行われる．各デバイスが接続しているネットワークは，WiFi や LTE 回線，有線などそれぞれ異なっている．以上のことから，都市は，多種多様でデータ流通が非常に多く，エッジデバイスの設置環境がそれぞれ異なる超分散エッジ環境である

[1] 慶應義塾大学
Graduate School of Media and Governance Keio University, Fujisawa, Kanagawa 252–0882, Japan
[2] ランカスター大学
Lancaster University, Bailrigg, Lancaster, U.K.
[3] ブリティッシュ・コロンビア大学
University of British Columbia, Vancouver, BC, Canada
[a)] makora@ht.sfc.keio.ac.jp
[b)] takuro@ht.sfc.keio.ac.jp
[c)] tanimu@ht.sfc.keio.ac.jp
[d)] kyng@ece.ubc.ca
[e)] m.broadbent@lancaster.ac.uk
[f)] r.lea1@lancaster.ac.uk
[g)] jin@ht.sfc.keio.ac.jp

[*1] https://www.nvidia.com/ja-jp/autonomous-machines/embedded-systems-dev-kits-modules/

といえる．そして，超分散エッジ環境における都市の課題解決が期待される．

しかしながら，機械学習を取り入れたシステム（以降，機械学習応用システムと呼ぶ）を超分散エッジ環境である都市で運用していくには様々な問題を解決しなければならない．大学や企業などが開発し，運用されている一般的なIT システムでは，問題の分析からシステムに必要な機能の洗い出しから設計を行い，その設計を元に実装，そして運用が行われている．一方で，機械学習応用システムの場合，機械学習の特性から一度運用して完成ではなく，問題の定式化をした後の試験的な取り組みや運用・維持管理を繰り返すこと（機械学習ライフサイクル）が非常に重要となる．機械学習応用システムを都市全域において運用を試みた場合，扱うべきデバイスの数は非常に多く，都市全体に分散しているため，機械学習ライフサイクルのコストは非常に大きいものとなってしまう．したがって，都市での機械学習応用システムを実現するために，超分散エッジ環境に対応した新しい開発環境を設計する必要がある．

本研究では，都市での機械学習応用システムの統合開発環境 CityFlow の設計と実装をする．CityFlow は，都市における機械学習応用システムの開発に必要なデータの流れの管理や，訓練させた機械学習モデルのデプロイなどの容易な記述を可能にする．これらの記述が容易になることで，従来の機械学習応用システムを開発する際に，ボトルネックとなっていたデータの収集から前処理，保存などの流れや機械学習応用システムを運用した際に得られる精度について検証する Proof of Concept（PoC）[13] を素早く繰り返すことが可能となる．CityFlow は Distributed Node-RED（DNR） [3], [7] を応用する．Node-RED は，ハードウェアデバイスおよび Web サービス API やオンラインサービスを相互に接続するためのツールであり，DNR は Node-RED をつまりエッジコンピューティング環境において利用可能に拡張したものである．

本稿では，CityFlow の有用性について検証および考察を行うため，2 種類のケーススタディを行った．1 種類目は，ゴミ清掃車に取り付けられたドライブレコーダーの映像から道路標示の損傷を検出するものである．物体検出技術を用いて損傷を検出し，プライバシー保護を行った結果を可視化アプリケーションで提示する．2 種類目は，参加型センシングにおける添付画像の自動ラベリングである．参加型センシングを用いて，都市で発生してしまったゴミの不法投棄や回収忘れ，落書きなどを市の職員が報告する．その際に，添付された画像からそれが何であるかを自動分類することで，職員が報告しやすくなる．本ケースを用いて機械学習応用システムの開発の容易さについて検証する．

本研究の貢献は以下の通りである．

- 超分散エッジ環境である都市における機械学習応用システムの開発・運用における問題を定義した．
- 都市における機械学習応用システム開発・運用のための統合開発環境 CityFlow の設計と実装を行った．
- CityFlow を実際に利用し，2 種類のケーススタディを行った．

残りの本稿の構成は以下の通りである．2 章で都市の課題解決のために，機械学習応用システムを開発・運用する際の問題について述べる．次に 3 章と 4 章で本研究が提案する統合開発環境 CityFlow の設計と実装について説明する．そして 5 章で，CityFlow を用いた 2 種類のユースケースについて紹介する．6 章で CityFlow の限界について議論し，7 章にて本稿をまとめる．

2. 都市での機械学習応用システム

本節では，都市の特徴が超分散エッジ環境であることを説明し，その後超分散エッジ環境における機械学習応用システム開発・運用における課題について述べる．

2.1 超分散エッジ環境

近年の IT 技術革新によりクラウドコンピューティング技術などが発展し，都市の情報が Web サービス[*2]やオープンデータ[*3]という形で一般公開され，都市の統計情報も得ることが可能となってきている．こうした様々な形式でデータを提供しているサービスなどを仮想センサとしてみなし，全てのデータを XML 形式で扱えるようにする取り組みもある．つまり，既に都市には仮想的なセンサが設置されているといえる．

また，NVIDIA 製 Jetson TX2 をはじめとした組み込みコンピュータや FPGA，Google 製の Tensor Processing Unit（TPU）など，エッジデバイスが注目されている．従来のエッジデバイスのスペックは非常に低かったため，計算量が多い機械学習など高負荷がかかる処理を行うことは難しかった．しかし，近年のエッジデバイスは小型化・高性能化してきており，その上非常に安価で購入が可能になってきている．そのため，農業分野など様々なアプリケーションに応用され[*4]，高負荷な深層学習を実行可能にする研究 [5] なども行われている．

現在の都市にある様々なものに組み込みコンピュータが搭載され，Internet of Things（IoT）技術によってインターネットに接続されることで，Web サービスなどの仮想センサによる情報に加えて，エッジデバイスおよびその周辺に関するデータを大量に取得することが可能となる．また自動運転技術など，より高性能な機能が搭載された自動車 [1]

[*2] http://soramame.taiki.go.jp/
[*3] http://www.city.fujisawa.kanagawa.jp/joho006/shise/kekaku/kakushu/datalibrary.html
[*4] https://cloud.google.com/blog/products/gcp/how-a-japanese-cucumber-farmer-is-using-deep-learning-and-tensorflow

やアプリで配車が可能なタクシー[*5]などが走行され始めており，道路交通情報なども徐々に得られるようになってきている．

以上のように現在の都市では，物理・仮想センサや組み込みコンピュータが設置されるようになってきており，非常に多種多様なデータが大量に得られる．一方で，ネットワークは研究開発・普及が進められているが，都市やデバイスの設置場所によって，ネットワーク環境（WiFi や 3G 回線，イーサネット接続など）が異なる．そのため，同機種のデバイスであっても，取得可能なデータ量などが異なることが想定される．したがって，IoT 技術やエッジコンピューティング技術が普及した都市は超分散エッジ環境にあるといえる．

2.2 機械学習応用システムのライフサイクル

機械学習応用システムの開発・運用におけるライフスタイルを**図 1** に示す．最初に，与えられた都市の問題を機械学習タスクとして取り組むため，統計量を算出するなどのデータ分析を行い，問題の特徴を掴み，定式化を行う．定式化の多くの場合は，実数値を予測する回帰問題か，何らかのカテゴリに分類する分類問題の 2 種類になる．その際に，手に入るデータ全てに教師ラベルが付与されているのか（教師あり学習），一部なのか（半教師あり学習），一切付与されていないのか（教師なし学習）によって，適用する機械学習アプローチを決定する．

次に，定式化したタスクを解くために，与えられたデータセットを用いて機械学習モデルを設計し，訓練させ，実際にタスクを解くことが可能であるか評価する．本稿では，この機械学習モデルで実際にタスクを解けるか検証する工程を PoC と呼ぶ [13]．本工程で，精度が出ない場合，問題の定式化からやり直す．

PoC の結果，可能であると評価できた場合，都市の一部のエッジ環境に実装し，実データで PoC と同等の精度でタスクが解けるか検証する（パイロット運用）．パイロット運用の結果，精度が悪化した場合，PoC に戻り，原因がデータの量や質の不足であったときはデータの改善を試み，モデルの容量であったときはモデルの改善を試み，実データに対する機械学習モデルの精度を高める．

十分に実データに対してモデルが適応できていると判断した場合，本番運用に移行する．そして随時メンテナンスに移行し，運用中に得られたデータを新たに含めてデータセットを再構築し，モデルの再学習により本番環境により最適化させる．著しくモデルの精度が悪化した際は，問題の定式化や PoC に戻る．

以上のように，機械学習応用システムの開発・運用では，探索的な工程 {PoC，パイロット運用，本番運用，メンテナンス } を繰り返し行っていく．

[*5] https://japantaxi.jp/

2.3 エッジ環境における機械学習応用システムの課題

本節では，超分散エッジ環境である都市において，機械学習応用システムを開発し，運用して行く際に生じる課題について説明する．

2.3.1 都市データの収集・前処理およびプライバシー

統計的機械学習において，訓練用のデータセットとして扱うために，データは整形され，正規化される．整形では，まずあらゆるデータを収集し，フィルタリングを用いて，地域ごとや時間ごとにデータを分ける．その後，正規化が解く機械学習タスクと扱うデータに合わせて行われる．結果的に使わないデータも収集されてしまうため，非常に時間がかかり非効率である．特に都市で収集されるデータは多種多様のため，余計にコストがかかる．その上，都市によっては，新しい機械学習タスクが与えられるため，その度に整形および正規化を行うのはコストが高い．

また，都市データの特性として，プライバシーの問題が挙げられる．エッジデバイスから取得されるデータは，Web データに比べて実世界に紐付きが強いため，場所に関してや個人情報につながってしまうことが多い．欧州で個人情報の扱いに関する規則である GDPR が施行されるなど，プライバシーは注目されている．したがって，機械学習応用システムを実現する際は，プライバシーを考慮し，あらかじめ機械学習モデルの訓練を行う前かモデルが予測を行なった後にプライバシー保護の処理が求められる．

2.3.2 都市データにおける共変量シフト

一般的に統計的機械学習では，学習データとテスト（予測）データを生成される周辺確率分布が変化しないことが求められる．すなわち，学習データの確率分布を p，テストデータの確率分布を p' とすると，$p(\mathbf{x}) = p'(\mathbf{x})$ であることが求められる．しかし，超分散エッジ環境である都市において，エッジデバイスが設置されているネットワークや状況は随時変化するため，予測したい結果 y との関係性，つまり条件付き確率 $p(y|\mathbf{x})$ は変わらないが，周辺確率分布 $p(\mathbf{x})$ と $p'(\mathbf{x})$ が異なってしまう．この確率分布が異なってしまうことは，共変量シフト [9], [10] と呼ばれている．

$$p(\mathbf{x}) \neq p'(\mathbf{x}), p(y|\mathbf{x}) = p'(y|\mathbf{x}) \quad (1)$$

よって，都市データを用いた機械学習応用システムでは，時間および空間的特徴の変化によって生じてしまう共変量シフト（以降，時空間共変量シフトと呼ぶ）を考慮する必要がある．その一例を**図 2** および**図 3** に示す．商業施設で収集したデータを用いて学習したモデルの場合，商業施設での予測精度は高くなる．一方で，商業施設以外の住宅街や大学，海岸沿いなど，商業施設で収集されるデータと空間的特徴が異なる場合は機械学習モデルの予測精度が異なってしまう．また同一の商業施設であっても，朝と夜で

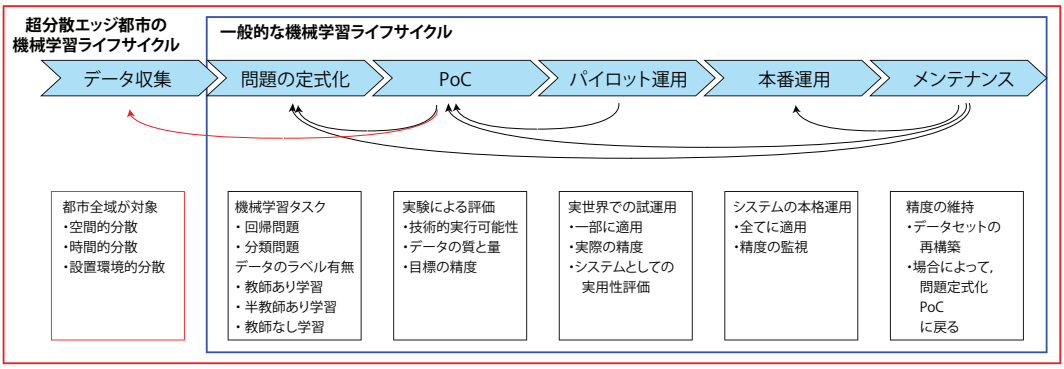

図 1　機械学習応用システムのライフサイクル．

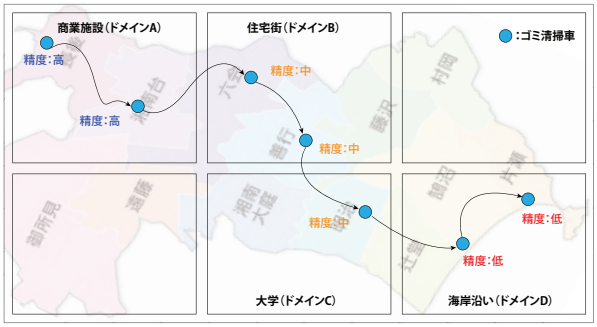

図 2　時空間共変量シフトによる精度への影響．商業施設周辺のデータで学習したモデルの場合，同様の商業施設では予測精度は高い．しかし空間的特徴が異なるにつれて，予測精度が下がってしまう．

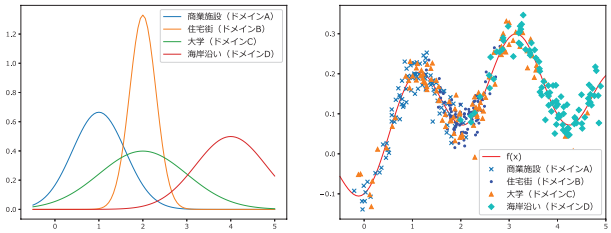

図 3　入力データの確率分布はドメインによって変化する（左図）が，学習したい真の関数は変化しない（右図）．

はデータが異なってしまうことが起こりうる．以上のことから，都市のデータにおける時空間共変量シフトの解決が求められる．

2.3.3　都市における機械学習モデルの利活用

問題の定式化，PoC およびパイロット運用による機械学習モデルの検証後は，都市全域に設置されたエッジデバイス上に訓練済みの機械学習モデルを配信し，デバイス上で動作する必要がある．既存の Web サービスでは，GPU が搭載されたクラウド上にサーバを設け，予測に利用するデータをバッチ形式でサーバに送信し，サーバ上で計算し，計算結果を再びクライアントデバイスに送信する流れになっている．これは，多くの機械学習モデルのパラメータ数が非常に多く，大容量メモリや高性能 CPU を搭載したコンピュータが必要となるためである．しかしながら，前節で述べたとおり，都市では時空間共変量シフトが発生しやすいため，一つの共通モデルのみでは対応できない．つまり，エッジデバイスが設置されている環境に合わせた機械学習モデルを用意する必要があるといえる．そして，各エッジデバイスの性能やネットワークなどの設置環境にはばらつきがあるため，これらを考慮した機械学習モデルの設計と配信をしなければ非効率となってしまう．

3.　CityFlow 構築のためのアプローチ

上記の問題を解決するため，本研究では，超分散エッジ都市において，機械学習応用システムのライフサイクルを柔軟に実現可能な機械学習応用システム開発環境 CityFlow を構築する．従来の機械学習応用システムは，基本設計として集中処理型となっている．多種多様なデータが，分散している大量のエッジデバイスから送信されてしまう都市では，従来の機械学習応用システムの効率的な運用は難しい．CityFlow では，機械学習応用システムを分散処理型として開発を可能にする．本章では，CityFlow 構築に向けたアプローチについて説明する．

3.1　都市における機械学習ライフサイクル

一般的な機械学習ライフサイクルでは，すでに対象とするデータが蓄積している一方で，都市は収集できていない．CityFlow では，大量のエッジデバイスの制御およびデバイス間の通信を可能にし，データ収集を行う（図 4 赤枠）．その際にデバイスの性能や設置環境の違いを考慮する必要がある．また物理センサから取得されたデータや Web データのログや API からの大量のデータに前処理などを行う必要がある．しかし，データによってフォーマットなどが異なるため，前処理の準備に非常に時間がかかってしまう．

そこで，CityFlow では，データリソースが物理センサなどハードウェアなのか，Web サービスなどソフトウェアなのか開発者が意識する必要がないように，同一の概念で抽象化する．その際に，ハードウェアであれば CPU やメモリなどの性能を，ソフトウェアである場合は，API 制限などをメタ情報として付与する．この抽象化により，デバイス間の違いなどを隠すことができ，開発者は機械学習モデ

ルの開発に集中することが可能になる．また前処理や，機械学習モデルの予測などの処理もデバイスなどと同じ概念で抽象化する．この抽象化により，データ前処理や学習済みのモデルをモジュールのように扱うことが可能になるため，そのほかの開発者が作成したものを流量したり，複数のタスク間で用いることが可能となる．

3.2 機械学習ライフサイクルにおける効率化

機械学習ライフサイクルにおける問題の定式化工程は，機械学習応用システムを構築する上で非常に重要である．問題の定式化の際に様々なデータを収集し，分析を行われる．問題の定式化が終わった後，PoC に入る．この工程に入った時点で，必要とするデータは決まっている．したがって，データを収集する時点で必要とする地域のデータにフィルタをかけることで、ネットワークやハードディスクの利用効率という点で有効であるといえる．具体的なデータの指定収集方法として次のものが考えられる．

- 明示的にデバイスを固有 ID などを用いて指定する．
- 位置などで物理空間的な範囲や時間帯によって指定し，その範囲にあるデバイスからデータを収集する．

一般的にエッジデバイスは，クラウド技術に比べて，圧倒的に CPU やメモリなどの性能が低い．そのため，前節で述べたデータの前処理やモデルの予測がエッジデバイス上で処理しきれない可能性がある．そこで，処理きれない場合のロードバランサーとしての機能を CityFlow に実装する必要がある．

3.3 時空間共変量シフトにおけるモデルのドメイン適応

前章で述べた時空間共変量シフトに対応する必要がある．時空間共変量シフトへの適応の際に考えるべき課題は二つある．一つ目は，一般的な深層学習をはじめとした統計的機械学習は外挿ができないため，時空間共変量シフトには適応できないことである．この課題に取り組んでいる手法として，Ganin ら [2] の研究をはじめとしたドメイン適応学習手法がある．Ganin らは，学習データ（＝ソースドメイン）とテストデータ（＝ターゲットドメイン）間に共変量シフトが存在するとし，二つのドメインで同じ確率分布の特徴量の出力が可能な特徴量抽出ネットワークと，実際にタスクを解くネットワーク，そして二つのドメインを判別するネットワークの三つを同時に学習させる．Ganin らの研究の場合，想定しているドメインは二つであるが，都市においては図 2 のように複数ドメインがあり，二つのドメインの組み合わせは指数的に増えてしまい，従来の機械学習応用システムのように集中処理型では扱えきれない．そこで，CityFlow では，特徴量抽出ネットワークと各ドメインデータごとにタスクを解くネットワークがそれぞれ動作するようにデバイスを分散させることが可能な機能の実装が必要である．

二つ目の課題は，時空間共変量シフトの検出である．共変量シフトは，訓練データで学習したモデルが出力する確率分布と異なる確率分布が出力されたデータにある．従来のシステムの場合，データは全て包括的に収集されてモデルに入力されるため，共変量シフトの検出は難しい．そこで CityFlow では，機械学習フローにおける運用・メンテナンス工程において精度をモニタリングする際に，時空間ごとにモニタリングすることで徐々に共変量シフトが生じる境界を発見することが可能となる．したがって，時空間共変量シフトのモニタリングが可能な機能を設計する．

4. CityFlow の実装

本章では，CityFlow を構成するシステムの説明を行う．CityFlow は, オープンソースである Distributed Node-RED と Sensor of XMPP によって構成される．

4.1 Distributed Node-RED

Node-RED [*6]は，データフローベースの視覚的プログラミングツールおよび IoT アプリケーションを開発するための言語である．キャンバス上でノードをドラッグ&ドロップで配置し，ノード同士をパスで接続することでアプリケーションを開発することができる．パスは，ノード同士の通信を表す．ノードとパスで表現されたアプリケーションはフローと呼ばれ，一つのデバイス上で一つのプロセスとして実行される．しかしながら，Node-RED は分散環境には対応していない．

Distributed Node-RED（DNR）は，Node-RED を分散環境向けに拡張されたものである [3][7]．DNR は，主に三つの機能が拡張されている（図 4）．一つ目の機能は，デバイスを示す固有 ID をノードに付与するものである．この機能により，携帯端末やエッジデバイス，クラウドサーバなど任意のデバイスでの処理を記述することが可能となる．

二つ目の機能は，異なるデバイス（ノード）間で通信を可能にする遠隔パスである．遠隔パスはパブリッシュ・サブスクライブ（Publish-Subscribe）方式で実装されており，ノードの固有識別子がトピックとして扱われる．具体例として，任意の二つのノード A と B において，A から B へのデータ送信を考える．A は，B が示すデバイスを特定する必要はなく，通信を仲介するブローカーに自身のノード識別子とデータを送信する．一方で B は，予め知らされているノード識別子をトピックとして持つデータをブローカーから受信する．これにより，異なるデバイス間の通信を開発者が意識することなく，データ処理の流れに集中することが可能となる．さらに，遠隔パスを用いることで，従来一つのエッジデバイスでは処理することができなかった高負荷の処理も，分割して複数のエッジデバイスに割り

[*6] http://nodered.org/

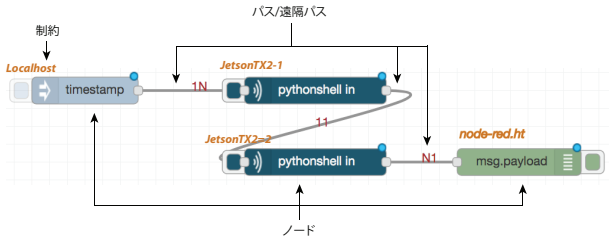

図 4 Distributed Node-RED におけるノード，パス/遠隔パス，制約．複数ノードをパスで接続することでデータの流れを，制約を用いることで各ノードのデプロイ条件などを記述できる．

当てることで，クラウドサーバを利用することなくエッジ側のみで可能となる．

三つ目の機能は，制約機能である．この機能は，アプリケーションのより複雑なルールの記述を可能にする．具体的には，エッジデバイスの CPU やメモリなどのリソースにおける制約や，物理空間上の位置情報にもとづいてデバイスへのデプロイを決めることが可能となる．この制約によって，都市全体における大量のエッジデバイスに対して，デプロイをすることが可能となる．具体的には，特定の位置に移動してきた自動車に搭載されたセンサのみデータを送信することが可能となる．

CityFlow に DNR を応用することで，機械学習モデルが動作するデバイスや，データリソースに関して詳細に知ることなく，開発者は機械学習ライフサイクルに専念することが可能となる．DNR のその他の機能については，論文 [3] を参照されたい．

4.2 Sensor-Over-XMPP ノード

CityFlow では，Sensor-Over-XMPP（SOX）を扱うノードを実装し，提供する．SOX は，ユニバーサルセンサデータ流通システム SOXFire[11] で使われている仕様であり，チャット通信などで使われているオープン XML 形式のインターネットプロトコルである XMPP を用いてメタ情報を表現することができる．これにより，物理センサや仮想センサからのデータを一律で扱うことが可能となる．機械学習で用いるデータは，画像データを始め，Web データやセンサデータなど多種多様である．したがって，CityFlow でも SOX を用いることは有用といえる．さらに，CityFlow では，エッジデバイス上で局所的な予測を行う機械学習モデルを何度も利用する．この SOX を用いることで，学習済みの機械学習モデルを都市全域に設置してあるエッジデバイスに読み込むことが可能となる．より詳細な SOXFire については，論文 [11] を参照されたい．

4.3 その他

CityFlow では，SOX ノードから Web データを利用可能にするため，Sensorizer[8] を用いる．また，機械学習モデルの実装については，特に技術的制約を CityFlow では設けていない．しかしながら，深層学習ライブラリとエッジデバイスの CPU アーキテクチャの組み合わせなどによっては，動作しないことがあるため，この点については，開発する際に注意する必要がある．例えば，Chainer[*7]は，python が動作するデバイスであれば，インストールおよび動作することが可能であり，多くのデバイスに対応が可能である．

5. ケーススタディ

5.1 道路の損傷検出

本稿では，河野らのゴミ清掃車に搭載されたドライブレコーダー映像を用いた道路の損傷検出技術 [6] をケーススタディとして採用する．現在の業務では，道路の損傷を市の職員の目視によって確認しているが，人件費もかかり，時間もかかってしまう．そこで河野らは，ゴミ清掃車のドライブレコーダー映像を用いた道路点検手法を提案した．

この道路点検手法を実際の業務に取り入れる場合，最終的に職員に映像を確認してもらう必要がある．しかしながら，河野らの提案手法では，NVIDIA 製 JetsonTX2 などエッジデバイスで処理することを想定しているため，映像を全て保存するのは現実的ではない．また，映像全てをネットワークを介してクラウドサーバなどに保存するのは，データの通信量・通信費の観点から効率的ではない．さらに，プライバシーの観点などから，個人が特定可能な状態で保存することも難しい．

以上のことを踏まえ，本稿で開発する道路点検アプリケーションは，次の流れで動作する．

(1) ゴミ清掃車に搭載されたエッジデバイスを用いて，ドライブレコーダーから道路損傷を検出する．
(2) 道路の損傷が検出された場合，損傷がある道路の位置情報を送信する．
(3) 複数回位置情報が登録されている道路は修復の必要性があるため，改めて周辺を走行しているゴミ清掃車などのドライブレコーダー映像を送信する．
(4) 送信された映像に，人の姿や自動車など個人情報に紐づく情報が含まれている場合，それらを取り除く．
(5) 確認用可視化アプリケーションに表示する．

この流れを，提案する CityFlow を用いて実装する．**図 5** に道路点検アプリケーションのフローを示す．道路点検アプリケーションは，主に二つのフローで構成されている．まず（1）と（2）を実現するため，図 5（上段）のフローを構築する．ドライブレコーダーの映像を分析する周期を決定するタイムスタンプノード，道路損傷検出ノード，そして SOX に位置情報を送信する SOX ノードが接続されている．

そして（3）（4）（5）を実現するため，図 5（下段）のフローを構築する．SOX から位置情報を取得し，現在のゴミ

*7 https://chainer.org/

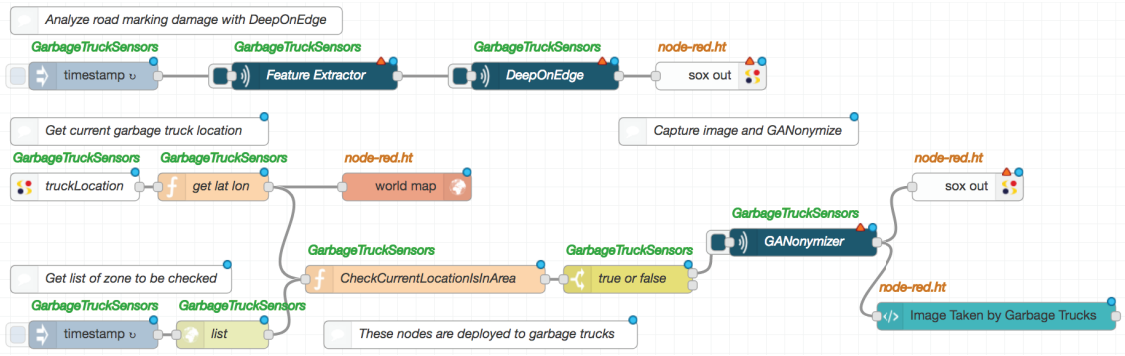

図 5　CityFlow によって開発されたゴミ清掃車による道路の損傷検出システム．

清掃車の位置情報と照合するノードを用意し，損傷した道路の周辺であった場合，プライバシー保護ノードに接続する．そして可視化アプリケーションのノードに接続される．

5.2　参加型センシングにおける自動ラベリング

本稿では，市の職員のための参加型センシングプラットフォーム「みなレポ」[14] を二つ目のケースとして用いる．みなレポは，藤沢市の職員が通常業務で市内を移動している際に，ゴミの不法投棄や回収忘れ，落書きなどを発見した際に報告するシステムとなっている．現在も実際の業務で使用されており，日々情報は集まってきている．しかしながら，現在のみなレポは，発見したものの分類（全 12 種類），対応の緊急度合い（緊急対応/通常対応/対応なし），コメント（任意，自由記述）を職員が手動で入力する必要があり，利便性が低い．

そこで，本稿ではこのみなレポでの報告を，画像から自動判断するアプリケーションを開発する．

(1) みなレポでカメラを使って撮影された報告画像の自動判断を各職員がもつデバイス上で行う．
(2) 自動ラベリングが間違ってる場合は，修正してもらう．その後 SOX に情報として送信・保存される．
(3) SOX に報告が一定数保存された場合，データセットを更新し，クラウドでモデルの再学習を行う．
(4) 学習したモデルを SOX に送信，各デバイスは新しいモデルを受信する．
(5) 特定の報告率が高い地域にいる職員には，周辺で報告が起こりやすいことを通知する．

この流れにもとづいて CityFlow で開発したフローを**図 6** に示す．自動ラベリングアプリケーションは，主に三つのフローで構成されている．まず（1）（2）を実現するフローを図 6（上段）に示す．SOX 上に新しいモデルがある場合は，それをダウンロードし，撮影された画像の自動ラベリングを行う．ラベリングの結果が間違っていた場合は，職員の方に修正をしてもらい，最終的な結果を SOX に送信する．次に（3）（4）を実現するフローを図 6（下段）に示す．一定時間ごとに SOX 上にある報告数を算出し，閾値以上のデータがあった場合はデータセットを更新し，モデルを再学習させる．ただし，この学習のフローはエッジデバイス上ではなく，クラウド上で行われる．最後に（5）を実現するフローを図 6（中段）に示す．定期的に職員の位置情報を取得し，報告件数が高い地域の範囲内にいるかどうか判断する．範囲内だった場合は，職員に通知をし，注意深く周辺を確認することを促す．

6.　議論

本研究で提案した CityFlow を用いた二種類のケーススタディでは，ドメイン適応と位置情報を考慮したアプリケーションを開発した．DNR 導入によって，データの流れと処理にのみ集中して開発することが可能となった．DNR 以外にも Hong らの Mobile Fog[4] が挙げられる．Mobile Fog 導入によっても複数のエッジデバイスへのアプリケーションのデプロイが可能となる一方で，一つのアプリケーションを複数のデバイスで処理することができないため，この点において DNR がより適しているといえる．一方で，CityFlow では，機械学習応用システムを構築する上で，重要である訓練とデータへのアノテーション作業を超分散エッジ環境に対応できていない．

モデル訓練．現在，入手可能といえるエッジデバイスの中で，性能が高いものでは本稿でも利用している NVIDIA 社 JetsonTX2 が挙げられる．JetsonTX2 は，組み込みコンピュータでありながら，GPU を搭載しており，深層学習の推論なども高速で行うことができる．しかし，搭載されている GPU のメモリは，RAM と共有で 8GB しかないため，深層学習モデルと入力データ，そしてモデルパラメータの勾配情報をすべて GPU メモリに乗せることが難しい．ゆえに JetsonTX2 で訓練させることは，現時点では現実的ではない．一方で，近年 FPGA を用いたエッジ側での訓練を実現しようとしている取り組み[*8]もあり，今後エッジデバイス上での学習が可能になることが期待される．

アノテーション．特定の機械学習タスクを解く場合，一般的に教師あり学習が最も性能が良いとされている．教師

[*8] https://www.fixstars.com/ja/news/1999/

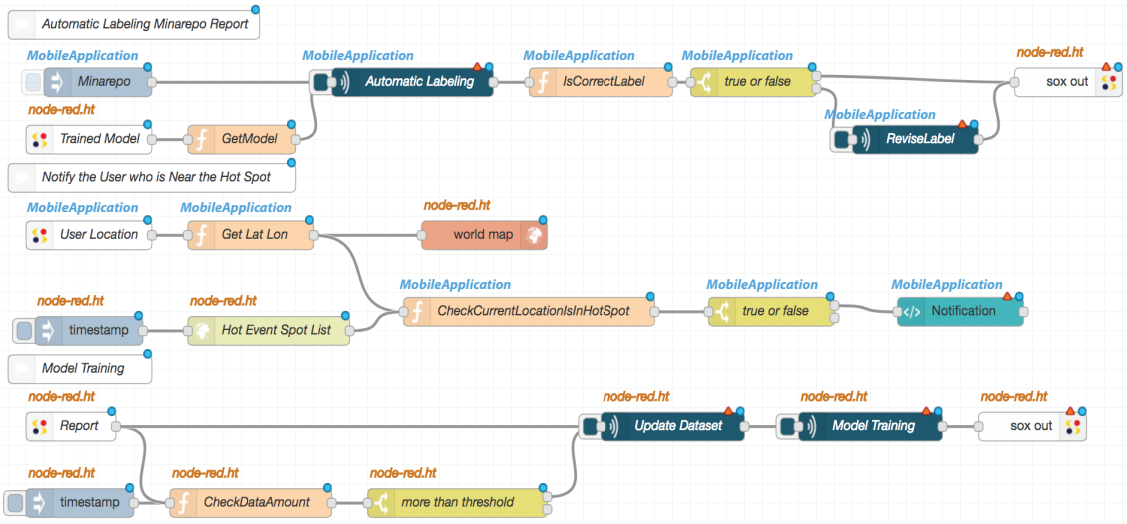

図 6　CityFlow によって開発された参加型センシングの自動ラベリング．

あり学習を用いるためには，入力データ $\mathbf{x}$ に対する正解ラベル $\mathbf{y}$ を用意する必要があるが，都市のデータに対してのアノテーションは，人手に依存しており，本稿でも人手によるアノテーションを想定した．

7. まとめ

本稿では，全域に様々な性能を持つエッジデバイスが設置されている超分散エッジ環境である都市の諸問題を解決する機械学習応用システムの開発環境 CityFlow について述べた．従来の機械学習応用システムを都市に適用した際に生じる問題を挙げ，CityFlow がそれらの問題を解決可能であることを説明した．そして実際に CityFlow の活用事例として二つのケーススタディを行った．今後，CityFlow を用いて開発された機械学習応用システムの長期運用を試み，その有効性について検証していく．

謝辞　本研究の一部は国立研究開発法人情報通信研究機構に支援頂いた．

参考文献

[1] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J. et al.: End to end learning for self-driving cars, *arXiv preprint arXiv:1604.07316* (2016).

[2] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M. and Lempitsky, V.: Domain-adversarial training of neural networks, *The Journal of Machine Learning Research*, Vol. 17, No. 1, pp. 2096–2030 (2016).

[3] Giang, N. K., Lea, R. and Leung, V. C. M.: Exogenous Coordination for Building Fog-Based Cyber Physical Social Computing and Networking Systems, *IEEE Access*, Vol. 6, pp. 31740–31749 (online), DOI: 10.1109/ACCESS.2018.2844336 (2018).

[4] Hong, K., Lillethun, D., Ramachandran, U., Ottenwälder, B. and Koldehofe, B.: Mobile fog: A programming model for large-scale applications on the internet of things, *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, ACM, pp. 15–20 (2013).

[5] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications, *arXiv preprint arXiv:1704.04861* (2017).

[6] Kawano, M., Mikami, K., Yokoyama, S., Yonezawa, T. and Nakazawa, J.: Road marking blur detection with drive recorder, *2017 IEEE International Conference on Big Data (Big Data)*, pp. 4092–4097 (online), DOI: 10.1109/BigData.2017.8258427 (2017).

[7] Lea, R. J.: Fog at the Edge: experiences building an Edge computing platform (2018).

[8] Nakazawa, J., Tokuda, H. and Yonezawa, T.: Sensorizer: an architecture for regenerating cyber physical data streams from the web, *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, ACM, pp. 1599–1606 (2015).

[9] Shimodaira, H.: Improving predictive inference under covariate shift by weighting the log-likelihood function, *Journal of statistical planning and inference*, Vol. 90, No. 2, pp. 227–244 (2000).

[10] Sugiyama, M. and Kawanabe, M.: *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*, MIT press (2012).

[11] Yonezawa, T., Ito, T., Nakazawa, J. and Tokuda, H.: Soxfire: A universal sensor network system for sharing social big sensor data in smart cities, *Proceedings of the 2nd International Workshop on Smart*, ACM, p. 2 (2016).

[12] Zheng, Y., Zhang, L., Xie, X. and Ma, W.-Y.: Mining interesting locations and travel sequences from GPS trajectories, *Proceedings of the 18th international conference on World wide web*, ACM, pp. 791–800 (2009).

[13] 丸山宏：機械学習工学に向けて，日本ソフトウェア科学会第 34 回大会 (2017 年度) 講演論文集. 日本ソフトウェア科学会 (2017).

[14] 坂村美奈，米澤拓郎，伊藤友隆，金子義之，　中澤仁ほか：みなレポ: 地方自治体の日常的な行政業務における参加型センシングによる情報収集・共有システム，デジタルプラクティス，Vol. 9, No. 2, pp. 550–572 (2018).