小口研究室　研究紹介　(2021年度)
(お茶の水女子大学理学部情報科学科)

ROS準拠ロボット及びエッジサーバを活用した環境情報収集・処理を行うIoTシステムの検討
（研究担当:佐々木 怜名）

研究概要

■ 背景
- IoT機器に付属するセンサによって収集された環境情報や動画像等のライフログデータの活用が進んでいる
- IoT機器から収集したセンサデータは大量のストリームデータとしてクラウドに集約される

■ 課題
- 室内全体を網羅的にセンシングし多種センサデータを収集するため複数箇所にセンサを設置する必要
- クラウドにおける処理負荷・通信遅延の問題
- データ送信時におけるプライバシの問題

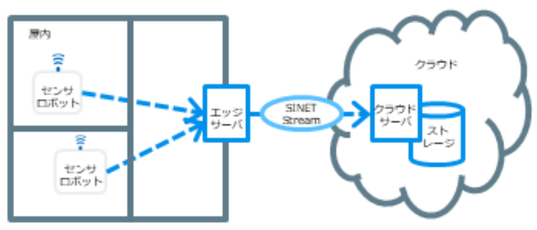
実験環境

■ ロボット
- Raspberry Pi ROS SLAM Robot (XiaoR Geek)
- ROS KInetic

■ エッジサーバ（代用）
- Surface Pro 4 Ubuntu20.04
- ROS Noetic

提案システム

■ ROS（Robot Operating System）で実装された車輪型移動センサロボットを用いて室内環境情報を収集
■ 収集したセンサデータをエッジサーバに集約し、必要に応じてデータの前処理や一部の解析処理をエッジサーバで行う
■ SINETSteamを利用して広域ネットワークを介したデータ送信を行う
■ データを欠損なく確実にクラウドへの収集し解析処理を行う



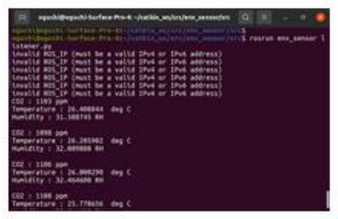
試行実験

■ PC端末をエッジサーバと見立たロボットとエッジ間における処理方法の検討のために実施した試行実験

■ **ロボットの遠隔移動操作**
- エッジからロボットの遠隔制御が可能であることを確認
- エッジにおいて走行操作パッケージ（teleop_twist_keyboard）を用いてロボットの走行方向・速度をキーボードで操作

■ **カメラ画像収集とエッジへの転送**
- 動画像データの収集の確認
- USBカメラノード用のパッケージ（uvc_camera_node）を用いてエッジで動作する表示ツールに画像をパブリッシュ
- エッジ上で画像を表示

■ **CO2濃度・温度・湿度の収集とエッジへの転送**
- 環境情報の収集の確認
- センサSCD4x（Sensirion）と自作パッケージを用いて収集した環境情報をエッジにパブリッシュ
- エッジ上でデータを表示



今後の課題

■ エッジサーバを介したクラウドへの収集・センサロボットの制御の方式の検討
■ カメラから収集した動画像のエッジサーバでの処理方法の検討
■ エッジサーバを活用した屋内環境情報の収集におけるプライバシ侵害の課題解決

完全準同型暗号を用いたゲノム秘匿情報検索のSSD適用に向けた調査
（研究担当:辻 有紗）

研究概要

- 完全準同型暗号(FHE)
  暗号化した状態で任意回加算乗算が可能な暗号方式
  FHEの実行で必要となるbootstrap処理などにより時間空間計算量が膨れあがることが課題
- クラウドの使用効率
  従来FHEの大きな空間計算量に対して、クラウド上のDRAMが使用されているが, Containerなどの利用により1台のリソースを分け合うため、使用効率は65%に止まる
- ハードウェア(DRAM・SSD)の特性の違い
  アクセス速度：DRAM 50ns, SSD 1ms　1GBあたりのコスト：DRAM 35円, SSD 0.72円
  並列IO(CPUへのload/store命令)：DRAM 直列処理, SSD：並列処理可能

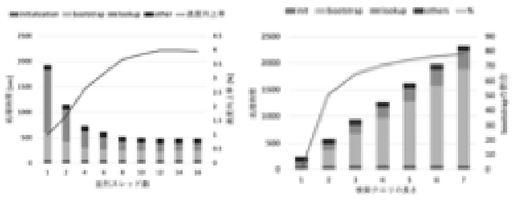
→ SSDとDRAMの特性を活かした使い分けによりFHE処理にかかるコストを削減し、高速化を検討

先行研究

ゲノム秘匿情報検索：クライアントがサーバのゲノムデータベース(PBWT)に対して特定の塩基配列が最長マッチを持つか問い合わせを行う
・お互いに持っているゲノム塩基配列や調べたい内容を秘匿しながら行うことがFHE により実現
・クエリ長や検索ポジション数に関わらず実行を可能にするには, FHE の演算でbootstrap の導入が必要

評価

- 並列度の調査
  OpenMPを用いた並列処理により, 並列数12(論理CPU 数) まで減少しその後は増加
  →論理CPU数以上の並列度ではCPUコストに関するオーバーヘッドが大きいことが原因
  ・処理全体として(特にbootstrap処理の)並列度が大きいため、SSDの並列IO利用により高速化

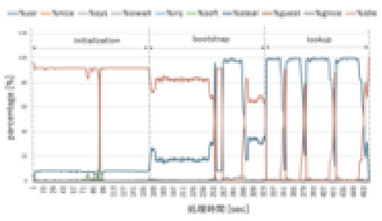


評価

- 処理のボトルネック
  hyperthread (計算性能は約2倍となるが、メモリへのload/store は逐次処理のままとなる機能)を有効にしCPU 使用率が高い箇所について、CPUの演算処理とメモリへのload/store処理の重さを比較

  評価結果：無効時(並列数12) < 有効時(並列数12) < 有効時(並列数24)

  ・12 以上の並列化に伴うCPU コストに変化が見られないことや, IPC の低下から，メモリへのload/store がボトルネック
  → CPU 使用率が高い箇所について, メモリへのload/store を並列に行うため, 処理を並列化し, DRAM を並列IOが可能なSSD に取り替えることが有効



	無効時(並列数12)	有効時(並列数12)	有効時(並列数24)
総実行時間(s)	477.33	605.67	623.67
initialization(s)	69.81	95.2	94.40
lookup(s)	144.17	155.43	166.41
bootstrap(s)	164.96	223.12	230.30
IPC	2.56	2.49	2.19
context switch	163,206	294,530	540,457
iTLB load数	35,309,006	66,520,758	104,796,253
iTLB load misses(%)	207.75	153.90	115.66
L3 cache load数	5.77×10^{10}	6.04×10^{10}	6.29×10^{10}
L3 cache load misses(%)	8.45	8.70	9.22

▲ hyperthread 有効/無効時のCPUコストの変化

◀ 実行時間に伴うCPU処理内訳の推移

今後の課題

・より細かい(関数ごとの)処理の効率化を検討
・非同期IOライブラリを用いたSSD並列処理の実装

モバイル端末上でのプライバシに配慮した画像認識モデルを構築する手法の提案と実装
（研究担当:近藤 華）

研究背景

Androidなどのスマートフォンを始めとするモバイル端末は高機能化と高性能化が進んでいる

❖モバイル端末の高機能化
→ 深層学習技術を用いた画像認識技術が様々な問題を端末内で処理する

❖モバイル端末の高性能化
→ CPUのみならずGPU、NPUの搭載
→ メモリ、ストレージの大容量化

プライバシの保護のためモバイル端末に保存した画像データを外部に送信せずに、それらの画像データから得られた情報を用いた画像認識を行う手法が生まれた

❖転移学習技術を用いる手法
→ 演算にかかる時間は短いが精度があまり高くない

❖端末のデータを外部のサーバに送信し、演算させた結果を受け取る手法
→ 外部のサーバにデータを送信するため機密性の高い情報で学習を行いたい場合に適さない

モバイル(Android)端末内のみでファインチューニングを行いモデルの精度を高める、プライバシに配慮した画像認識モデルを構築する手法の提案

❖精度を高める
→ 転移学習よりも精度が高くなるであろうファインチューニングを使用
→ アーキテクチャには画像認識モデルとして精度が良いことが確認されているMobileNetを使用

❖ 個人情報になり得るデータを外部に送信しない
→ 端末内のみで端末内データを使用した学習処理を行うことで、学習に使用するデータのみならず学習後のモデルも端末外に出さない

実験概要

以下の２つの実験を行った。

① モバイル端末内のデータ（各種類100枚と少ないデータ量）とプロセッサのみが使用可能な環境下においてもファインチューニングを使用することで精度の高いモデルが作成可能であるかの検証実験
①の実験は、モバイル端末より実装が容易なPC上で行った

② モバイル端末内にモデルを組み込んだ場合にNNAPIを使用することで速度が向上するか確認を行う実験
学習しない層のみのベンチマークを測定した

実験結果

実験①
図1に矢印で示した層から出力層までの再学習を十分に行い、再学習後のモデルにおける精度を測定した。
結果は表1に示すようになった。
これにより、再学習用のデータが各種類100枚と少ない枚数でも方法2の手法である畳み込み層も再学習するファインチューニングを行う手法により精度の良いモデルを構築可能であることが確認できた。

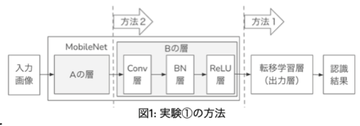


図1: 実験①の方法

表1: 実験①の結果

	方法 1	方法 2
精度 [%]	91.00	95.99

実験②
環境は右の表2のものを使用した。
それぞれのベンチマークの結果は表3に示すようになった。
いずれの端末でも、
NNAPI使用時の方が未使用時より実行速度が速くなることが確認できた。

表2: 実験②で使用した端末

Model number	Pixel4	Pixel5	POCO F2 Pro
OS	Android 12	Android 11	Android 10
SoC	Snapdragon 855	Snapdragon 765G	Snapdragon 865
Memory	6 GB	8 GB	8 GB

表3: ベンチマークの測定結果

Model number	Pixel4	Pixel5	POCO F2 Pro
NNAPI 未使用時	16203.4	16697.2	13664.8
NNAPI 使用時	5660.83	13826.7	6593.78

まとめ

モバイル（Android）端末向けのプライバシに配慮した精度の良い結果がえられる個人用の画像認識モデルを構築する手法として、モバイル端末上で畳み込み層も再学習するファインチューニングを行う手法を提案した。
また、モバイル端末上で実行可能なモデルの実装において、重みを固定する層ではAndroid Neural Networks API（NNAPI）を利用することで学習速度が向上することが確認できた。

今後の課題

Android端末上で実行可能なモデルの実装とアプリケーションの作成を行う。

❖モバイル端末上で実行可能なモデルの実装
→ 実験1の方法2の手法に従い、再学習を行う部分の実装も行う
→ 実装後に学習速度の計測を行い、実装方法やモデルの改善を行う

❖アプリケーションの作成
→ 実装したモデルがモバイル端末上で実用可能か調査するために作成する

http://ogl.is.ocha.ac.jp/
oguchi@is.ocha.ac.jp