強化学習を用いた分散画像解析技術の効率化手法

Efficiency Improvement Methods for Distributed Image Analysis Using Reinforcement Learning

中川 竣介 † 水谷 后宏 ‡,§
Shunsuke Nakagawa Kimihiro Mizutani

1. はじめに

近年，映像解析の技術が向上し，様々な場所でカメラデバイスを用いた顔認証システムが用いられている．多くの顔認証システムでは，取得した映像データを外部のサーバに送信し，解析を行う．したがって，解析速度の高速化を実現するためには，高性能なサーバを用いる必要がある．しかし，高性能なサーバは高価かつ費用対効果が小さい．そこで，一般的には比較的安価な汎用サーバを複数台利用し，並列処理により映像データを解析することが多いが，並列処理に利用する複数のサーバ間での性能の差が大きければ，性能が低いサーバの解析速度がボトルネックとなり，全体の解析速度に影響を及ぼす可能性がある．先行研究では，様々なアプリケーションからの解析要求に対して各サーバの資源の割当を行う研究 [1] が行われているが，画像解析に特化したものではない．また，画像解析において解析要求を平準化するバッファを利用することにより優先度制御を行い，負荷を抑える研究 [2] なども行われているが，バッファを利用して優先度制御を行うにはサーバに解析画像を振り分けるロードバランサにもメモリの性能が求められる．そこで，本研究では，クライアントからの解析対象となる画像をバッファリングし，各サーバの性能や解析状況に応じて，解析する映像データを動的に適切に割り当てるシステム及び，そのアルゴリズムの実装と検証を目的とする．

2. 強化学習を用いた計算資源割当手法

本研究では，ロードバランサと解析サーバ群からなる分散システムを提案する．このシステムでは，ロードバランサがクライアントから送信された解析対象の画像をバッファリングし，サーバの CPU 負荷に応じて画像の解析を割り当てる手法を採用する．システム上での処理は以下のように行われる．

1. ロードバランサがクライアントのカメラデバイスから映像データを受信する．
2. ロードバランサは映像データをフレーム単位に分割する．
3. ロードバランサは各計算資源 (解析を行うサーバ) の解析状況に応じて適切に計算資源をフレーム解析タスクに割り当てる．
4. 各計算資源での解析結果をクライアントが受信し，描画と表示を行う．

各計算資源を CPU 負荷に応じて動的に割り当てる手法として，強化学習 [3] を採用した．強化学習は機械学習の一種であり，エージェントが環境との相互作用を通じて行動を学習するフレームワークである．エージェントとは，環境内で行動を選択し，その結果として報酬を受け取る主体である．エージェントは各時間ステップで状態を観察し，行動を選択する．その行動により，エージェントは環境の状態を変更し，報酬を受け取る．本システムでは，エージェントは，計算資源割当を行うアルゴリズム，環境は計算資源群となる．エージェントの目標は，累積報酬を最大化するポリシーを学習することである．価値関数を求めるアルゴリズムには，Q 学習 [4] と SARSA 法 [5] を用いた．また，それぞれの構成要素が持つ変数を以下のように定義した．

- 状態 (State)
  各計算資源の CPU の割当可能な割合を表す．例えば，コンテナが 4 つ存在する場合，状態は $[10, 21, 98, 0]$ にような配列にて表現される．この時，1 番目の要素である 10 は，CPU の非使用率が 10%であることを意味しており，これは，(100% - CPU 利用率) によって算出することとした．
- 行動 (Action)
  ロードバランサが解析対象のフレームを，どの計算資源に割り当てるかを決定し，該当の計算資源にフレームを送信する．送信された計算資源は画像解析と特徴量抽出を行い，結果を返す．
- 報酬 (Reward)
  計算資源が，フレームを受信し，解析結果をロードバランサに送信するまでの時間を計測する．直近の解析時間平均と同じか，それよりも短ければ+1，長くなっている場合は，-1 を報酬として与える．

† 近畿大学大学院総合理工学研究科, Graduate School of Science and Engineering Research, Kindai University
‡ 近畿大学情報学部, Faculty of Informatics (KDIX), Kindai University
§ 近畿大学情報学研究所, Cyber Informatics Research Institute, Kindai University

以上の変数を使用して強化学習を行う．Q 学習では式 (1) を用いて学習を行う．時刻 t にて，行動 a_t で，計算資源を割り当てることによって，各計算資源でのリソース使用状況が変化し状態 s_{t+1} へ遷移する．このとき，エージェントが報酬 r_t を獲得し，状態 s_t での行動 a_t を評価する関数 $Q(s_t, a_t)$ を更新する．Q 関数における方策は，Q 関数の状態 s_t で最大価値が期待される行動 a_t となる．しかし，期待される価値が最大の行動のみを取り続けると，学習が進んだ際に報酬の期待値に差が出る状況が生じ，常に $argmaxQ(s_t, a_t)$ となってしまう．そこで，一定の確率 ε でランダムな行動をとり，確率 $(1-\varepsilon)$ のときのみ最大行動価値の行動を選択する ε-greedy 法 [6] を利用する．Q 学習は，方策と実際に行う行動が異なるため，方策オフ (Off-Policy) の学習手法と呼ばれる．

$$
\begin{aligned}
Q(s_t, a_t) \leftarrow\ & Q(s_t, a_t) \\
& + \alpha[r_{t+1} + \gamma \cdot maxQ(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]
\end{aligned}
\tag{1}
$$

一方で，SARSA 法では式 (2) を用いて学習を行う．SARSA 法は Q 学習と同様に強化学習の代表的な手法の 1 つである．SARSA 法では，Q 関数の値 $Q(s_t, a_t)$ が報酬 r_{t+1} と次の状態 s_{t+1} での行動価値 $\gamma Q(s_{t+1}, a_{t+1})$ の和となるように行動価値関数を更新する．価値関数を更新したあとは，行動 a_{t+1} を実行するが，SARSA 法では Q 学習と異なり，ε-greedy 法による実際の方策に従って行動価値関数を更新する．そのため，SARSA 法は方策オン (On-Policy) の学習手法と呼ばれる．

$$
\begin{aligned}
Q(s_t, a_t) \leftarrow\ & Q(s_t, a_t) \\
& + \alpha[r_{t+1} + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]
\end{aligned}
\tag{2}
$$

強化学習で用いるその他のパラメータとして学習率 α は，行動によって変化した更新量を現在の Q 値へどれだけ反映させるかを決定し，割引率 γ は，将来与えられる報酬をどれくらい現在の価値として考慮に入れるかを決定するパラメータである．

本節では，強化学習の一種である Q 学習と SARSA 法を用いた計算資源の動的割当手法を提案した．それぞれの方法は，環境との相互作用を通じて学習し，最適な資源割当を達成することを目指す．強化学習を利用することで，ロードバランサは各計算資源の負荷を考慮して適切にタスクを割り当て，全体のシステムパフォーマンスを最大化することが可能となる．

3. 実験・評価

3.1 Docker を用いたエミュレーション環境

2 節で提案した手法の検証を行うために，Docker[7] によるコンテナ型仮想環境を利用し，図 1 のようなエミュレーション環境を作成した．コンテナ型仮想化とは，サーバの OS 上に仮想化ソフトウェアをインストールし，仮想化ソフトウェア上でコンテナを動作させる技術である．コンテナはアプリケーションの実行環境のみを仮想化し，ホスト OS 上で動作するため，アプリケーションとその依存関係を一つのパッケージとして扱うことができ，環境間での移植性が高まると言う利点がある．Docker はこのコンテナ型仮想化を利用して，アプリケーションを構築，実行，共有するためのソフトウェアである．各コンテナでは CPU やメモリなどのハードウェアに制限を行うことができる．さらに，Docker Compose と呼ばれるツールを使用することで，複数のコンテナで構成されるアプリケーションについて，Docker イメージのビルドや各コンテナの起動，停止などをより簡単に実行したり，各コンテナ間でネットワークを作成したりすることができる．

本研究では，Docker を使用して複数のコンテナを稼働させ，それぞれのコンテナを仮想ネットワークで接続した．また，ロードバランサとなるエッジデバイスとして Raspberry Pi 4 (メモリ 4GB) を使用し，Ubuntu サーバ上で稼働させた Docker コンテナと物理的にネットワーク接続を行った．学習の条件を固定するため，カメラデバイスではなく，Raspberry Pi 4 上に保存した動画ファイルを利用した．Docker コンテナでは，Python の Web アプリケーションフレームワークである FastAPI[8] を使用して解析画像の送受信を行った．一方，Raspberry Pi 4 では Flask を利用し，Web ブラウザから解析結果を表示する機能を実装した．

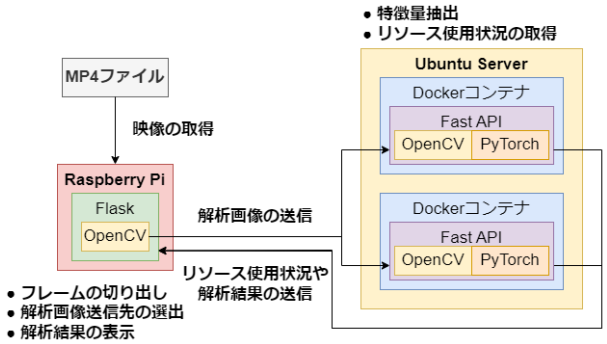

図 1: Docker コンテナを利用したエミュレーション環境

3.2 実験環境

3.1 節で述べたエミュレーション環境で計算資源割当を行うアルゴリズムを実行した．4 つの Docker コンテナを稼働させ，各 Docker コンテナに表 1 のような性能制限を設定した．また，各 Docker コンテナでは解析処理として以下の処理を実行した．

1. OpenCV のカスケード分類 [9] によってフレーム中の顔を検出する．
2. 検出された顔の部分のみを切り出し，リサイズとグレースケール変換を行う．
3. 18 層の畳み込みニューラルネットワークである Resnet-18[10, 11] で学習したモデルを利用して顔画像から男女の性別分類を行う．
4. 顔の検出位置と，性別の分類結果を response として返す．

表 1: 各コンテナの制限

コンテナ	CPU	メモリ
A	2 スレッド	4GB
B	4 スレッド	8GB
C	6 スレッド	8GB
D	6 スレッド	16GB

エピソード数 (学習・評価回数) を 2000 回に設定し，学習を行った．また，Q 学習のみ割引率 γ と探索率 ε のパラメータを変化させた．割引率 γ は，将来もらえる報酬をどれくらい現在の価値として考慮に入れるかを表す．一方，探索率 ε はランダムにサーバを選出する確率を表す．

3.3 提案手法の評価

3.2 節で設定した実験環境に対して，2 節で述べた計算資源割当の手法を実行した．以下の図 2, 図 3, 図 4 は横軸に学習のエピソード数を示し，縦軸に行動によって得られた累積報酬 (Cumulative reward) を示している．図 2 は学習の手法による違いを示しており，学習なし，Q 学習，SARSA 法のそれぞれの累積報酬の変化を表している．学習なしの場合は $\varepsilon = 1.0$ で資源を割り当てるサーバを完全にランダムで選出している．また，図 3 は Q 学習において，割引率を変化させた場合の累積報酬について表しており，図 4 は探索率を変化させた場合の累積報酬について表している．実験結果から，Q 学習と SARSA 法では大きな差は見られなかったが，どちらの学習でもエピソードが進むにつれて累積報酬が増加しており，計算資源の効率的な割当が行われていることが確認できた．

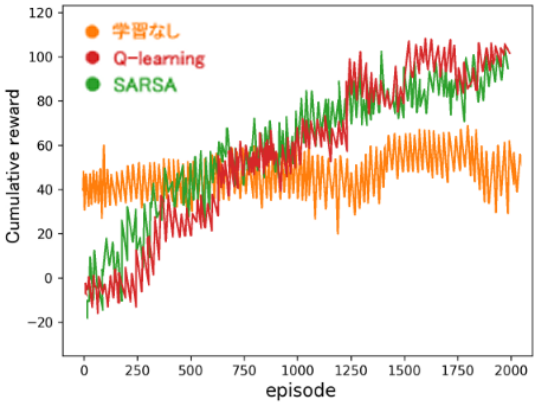

図 2: 各学習手法による効果 ($\gamma = 0.9, \varepsilon = 0.2$)

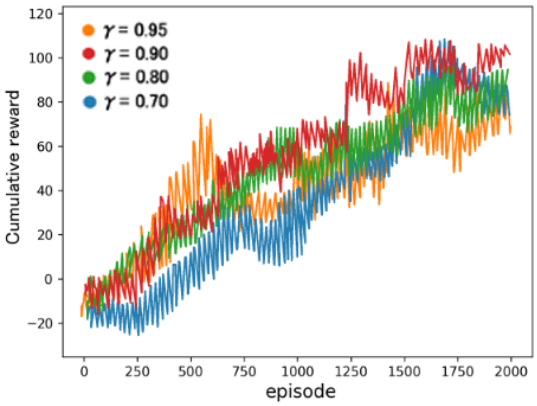

図 3: Q 学習における割引率の変化が及ぼす影響 ($\varepsilon = 0.2$)

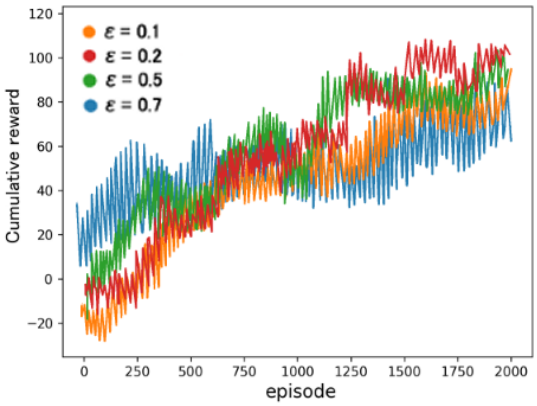

図 4: Q 学習における探索率の変化が及ぼす影響 ($\gamma = 0.9$)

4. 結論と今後の課題

本研究では，汎用サーバを複数台利用した並列処理において，各サーバの性能や解析状況に基づいて画像解析処理に計算資源を適切に割り当てるアルゴリズムの提案と実装を行った．実験結果より，学習が進むにつれて累積報酬が増加していることから，計算資源の効率的な割り当てが行われ，性能が向上していることが確認できた．しかし，学習時に実際にフレームを送信して分類を行う必要があるため，学習にかなり時間がかかってしまうという問題も見られた．また，全てのフレームを等しくサーバに送信し続けると，ネットワーク上のトラフィックが多くなってしまうため，エッジデバイス側での前処理を最適化したり，映像の監視とイベントの検知を行ったりすることで，イベント発生時のみサーバへの処理要求を増やすといった処理を実装することも今後の課題として挙げられる．

参考文献

[1] W. Wei, H. Gu, K. Wang, J. Li, X. Zhang and N. Wang, "Multi-Dimensional Resource Allocation in Distributed Data Centers Using Deep Reinforcement Learning," in IEEE Transactions on Network and Service Management, vol. 20, no. 2, pp. 1817-1829, 2023

[2] 小山和也, 岩松洋介, "大規模映像解析システム向けのサーバ資源割当制御手法," 情報処理学会第 77 回全国大会講演論文集, pp. 37–38, 2015.

[3] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," vol.1, MIT press Cambridge, 1998.

[4] C. J. Watkins and P. Dayan, "Q-learning," Machine learning, Vol. 8, No. 3-4, pp. 27–292, 1992.

[5] M. Grzes and D. Kudenko, "Robustness analysis of SARSA(lambda): Different models of reward and initialisation," doi:10.1007/978-3-540-85776-1 13, pp. 144–156, 2008.

[6] A. dos Santos Mignon and R. Luis de Azevedo da Rocha, "An Adaptive Implementation of ε-Greedy in Reinforcement Learning," Procedia Computer Science, Vol. 109, pp. 1146–1151, 2017.

[7] Docker, https://www.docker.com/.

[8] FastAPI, https://fastapi.tiangolo.com/ja/.

[9] S. Syed Ameer Abbas, P. Oliver Jayaprakash, M. Anitha, and X. Vinitha Jaini, "Crowd detection and management using cascade classifier on ARMv8 and OpenCV-Python," 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), pp. 1–6, 2017.

[10] K. Lyman Sasha Targ and D. Almeida, "Resnet in Resnet: Generalizing Residual Architectures," arXiv preprint arXiv:1603.08029v1, 2016.

[11] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," arXiv preprint arXiv:1512.03385, 2015.