クロスアプリケーションフィンガープリンティング —同一端末上のアプリケーション間の紐付け—

齋藤 祐太[1]　細谷 竜平[1]　齋藤 孝道[2,4]　森 達哉[3]

概要：モバイルデバイスの利用において，別々のアプリケーションからのアクセスをサーバ側で紐づけることは一般に困難とされている．たとえば，モバイルデバイス利用者は，アプリケーションによる通信を伴う広告表示を契機として，同一デバイスのブラウザを使って関連する情報を検索することがある．このような 2 つの通信の紐付けを，クロスアプリケーショントラッキングと呼び，アプリケーションを跨いだ識別技術の一つとして知られている．しかし，クッキーを用いず，もしくは，利用者を認証せずに，2 つの通信の紐付けを実現することは一般には容易ではない．本論文では，同一デバイスにおいて，アプリケーションで取得できる情報，および，ブラウザからサーバへの通信で取得できる情報を用いて，フィンガープリンティング技術により，2 つのソフトウェア主体を紐付けることを行う．そして，その精度を検証する．

Cross-App Fingerprinting : Linking Mobile Apps and Browsers on the Same Device

Yuta Saito[1]　Ryohei Hosoya[1]　Takamichi Saito[2,4]　Tatsuya Mori[3]

Abstract: In mobile devices, it is generally considered difficult to link access from different apps on the server side.For example, a mobile user who watched an advertisement displayed on the app sometimes searches related information using browsers on the same device.Such linking of more than two apps is called Cross-App Tracking and is known as one of identification technologies beyond apps, but it is not easy to realize it without authenticating a user.In this paper, we realize to link two software entities by Fingerprinting using information that can be acquired by apps and browsers on the same device, then verify its accuracy.

1. はじめに

2018 年までに，世界総人口の 36 %以上がスマートフォンを利用すると予測されている [4]．また，別の調査によると，2018 年 3 月現在，日本国内でのスマートフォン利用率は 64%であるとのことである [6]．このように，スマートフォンの利用が一般化する一方で，フランスの Web 広告配信企業 Criteo による 2016 年の調査 [5] によると，利用者の 64%は 1 つのデバイスのみで，Web サイトに初めて訪問してから購入に至ることが報告された．たとえば，スマートフォン利用者はオンラインゲームをプレイする際，そのアプリケーションが表示する広告を見て，同じデバイス内のブラウザにて広告に掲載された商品を検索するなどの行動が想定できる．しかしながらそのような場合，たとえ同じデバイス内であっても利用者の操作が 2 つのアプリケーションに跨るので，上述のような行動を識別すること，いわゆるクロスアプリケーショントラッキングは困難である．

そこで，本論文では，同一デバイスにおける，アプリケーションで取得できる情報，および，ブラウザからサーバへの通信で取得できる情報を用いて，フィンガープリンティング技術により，2 つのソフトウェア主体を紐付けることを行う．これは，クロスアプリケーショントラッキングの一種であるが，本論文では，特に，アプリケーションとブ

1 明治大学大学院
Graduate School of Meiji University
2 明治大学
Meiji Univevrsity
3 早稲田大学
Waseda University
4 レンジフォース株式会社
Rangeforce Inc.

ラウザの紐付け（以降，クロス App-Web トラッキングと呼ぶ）について調査する．

本論文の成果は，以下の通りである．

- クロス App-Web トラッキングが，Precision（適合率）98.0%で実現できることを示した．
- パッシブフィンガープリンティングでのトラッキングが，Precision 99.9%で実現できることを示した．

特に，既存類似研究である Zimmeck ら [9] の実験と比較して，対象としたサンプル数が 5 桁多く，大規模な実験であるだけでなく，Precision および Accuracy（正解率）などについても，良い結果が得られた．

2. 背景知識

2.1 フィンガープリンティング

フィンガープリンティングはデバイスから採取可能な情報を用いてアプリケーション，ブラウザやデバイスなどを，サーバ側で，確率論的手法で識別する技術である．デバイスから採取可能な個々の情報の種類を特徴点，採取された情報（値）の組み合わせをフィンガープリントと定義する．

ここで，確率論的手法とは，UserAgent の情報などのフィンガープリントを用いて，確率モデルにより，複数の主体を紐付ける手法である．ここでの確率モデルには，フィンガープリント間の類似性を用いたものや，本論文で採用する教師あり機械学習を用いるものの他，多種存在する．

2.2 クロスアプリケーショントラッキング

クロスアプリケーショントラッキングとは，利用者が使用しているデバイスの中で，複数のアプリケーションを跨いでそれらを紐付けることである．2 つのモバイルアプリからのアクセスは，同一のサーバへのアクセスとは限らず，別のサーバへのアクセスをそれぞれ別々に採取し，それらをフィンガープリンティングを用いて紐付けることもある．

クロスアプリケーショントラッキングで利用するフィンガープリントの例としては，一般に，以下がある：

- IP アドレス
- タイムスタンプ
- UserAgent の情報
- 広告 ID（モバイルアプリケーションの場合）
- （Web サイトへの）アクセス履歴
- クッキー

ここで，広告 ID は，モバイルデバイスだけで取得される．特に，モバイルの場合，広告 ID が利用できれば，一意にデバイスを特定できるが，利用者に利用を拒否されることや更新されることを想定する．

2.3 クロスブラウザトラッキング

クロスブラウザトラッキングとは，クロスアプリケーショントラッキングの一種だが，デバイスの中でブラウザを跨いでそれらを紐付けることである．

クロスアプリケーショントラッキングとの違いとしては，JavaScript 等の実行はサンドボックス化されており，通常のアプリのように端末識別情報等を取得できない．また，クッキーは，ブラウザ間で共有できないだけでなく，利用者に削除されることを想定する．

2.4 クロス App-Web トラッキング

本論文では，同一デバイスにおいて，アプリケーションで取得できる情報，および，ブラウザからサーバへの通信で取得できる情報を用いて，フィンガープリンティング技術により，2 つのソフトウェア主体を紐付けることを行う．これは，クロスアプリケーショントラッキングおよびクロスブラウザトラッキングの混合であるが，本論文では，クロス App-Web トラッキングと呼ぶ．

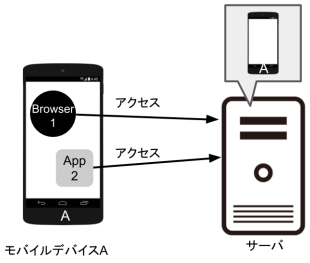

図 1 クロス App-Web トラッキングの概念図

3. 関連研究

Boda ら [7] は，ブラウザをまたいで利用者を識別するクロスブラウザフィンガープリンティングという技術を提案した．クロスブラウザフィンガープリンティングには，グローバル IP アドレスや UserAgent の情報，およびフォントリストが使用された．

Zimmeck ら [9] は，126 の利用者から収集した 27,000 のサンプルを用いて，実際にクロスデバイストラッキングを実施した．この実験により，モバイル端末からデスクトップ端末へのクロスデバイストラッキングの精度（F_1）が 0.91 にも上昇したことを示した．このトラッキングは IP アドレス，Web サイトの閲覧履歴の類似性により達成できることがわかった．また，彼らは広告配信を観察することによって，クロスデバイストラッキングの有無を検知することが出来ることを示した．

Brookman ら [8] は，世界的に人気な 100 の Web サイトに対し，クロスデバイストラッキングの実施状況を調査した．調査結果として，第 1 に，クロスデバイストラッキングを行うことが知られているサードパーティのライブラリが，100 の Web サイトのうちの少なくとも 87 に埋め込ま

れていたことが判明した．第 2 に，クロスデバイストラッキングを行うことが知られているサードパーティが 861 も存在していることが判明した．第 3 に，100 の Web サイトのうち 96 が利用者に利用者の氏名またはメールアドレスを送信させていることが判明した．さらに，調査した 100 の Web サイトのうち 16 の Web サイトが，サードパーティと利用者の氏名またはメールアドレスを共有していることが判明した．

高橋ら [10] は，JavaScript によって採取する特徴点を使用せず，HTTP ヘッダのみから得られる特徴点のみを用いたパッシブフィンガープリンティングの実験を行った．実験結果によると，HTTP ヘッダから得られる情報のみでも，9 割前後の精度でブラウザの識別が可能であることが判明した．特徴点の中でも，特にグローバル IP アドレスと HTTP UserAgent の情報が識別に大きく貢献していることが判明した．

4. データセット

本節では，実験に用いたデータセットについて解説する．

4.1 データセットの概要

本論文では，以下の 2 種類のデータセットを扱う：

- **アプリ由来データ**
  - モバイルアプリケーションから取得したデータ
  - サンプル数は約 354,000,000
- **Web 由来データ**
  - モバイルブラウザから取得したデータ
  - サンプル数は約 22,000,000

なお，アプリ由来データおよび Web 由来データにおいて採取した特徴点は，表 1 の通りである．

表 1 データセットにおける特徴点
（✓：当該データセットに存在する特徴点であることを示す）

特徴点	例	Web 由来	アプリ由来
タイムスタンプ	1534225576	✓	✓
クッキー値	123...abc（32 桁）	✓	
識別子	123...ABC (32 桁)	✓	✓
IP アドレス	192[.]168.1.168	✓	✓
OS 名	Android	✓	✓
OS のバージョン	Android6.0	✓	✓
機種名	iPhone7,2	✓	✓

4.2 データセットの詳細

本節では，採取したデータセットにおける特徴点ごとの詳細情報をまとめる．ここで，本節にて登場する ‘割合’ とは，該当するデータの数をそれらが属するデータセットの総数で除算した値を示す．

4.2.1 タイムスタンプ

データセットにおけるタイムスタンプの期間は 2017 年 2 月 27 日から 2017 年 3 月 13 日の 15 日間である．また，データセットにおける日付ごとの割合を表 2 に示す．

表 2 日付ごとのアクセス数の割合

Web 由来データ		アプリ由来データ	
日付	割合	日付	割合
2017-02-27	0.0727	2017-02-27	0.0666
2017-02-28	0.0668	2017-02-28	0.0669
2017-03-01	0.0645	2017-03-01	0.0662
2017-03-02	0.0678	2017-03-02	0.0667
2017-03-03	0.0625	2017-03-03	0.0683
2017-03-04	0.0670	2017-03-04	0.0669
2017-03-05	0.0701	2017-03-05	0.0658
2017-03-06	0.0687	2017-03-06	0.0661
2017-03-07	0.0657	2017-03-07	0.0657
2017-03-08	0.0661	2017-03-08	0.0659
2017-03-09	0.0636	2017-03-09	0.0669
2017-03-10	0.0621	2017-03-10	0.0682
2017-03-11	0.0671	2017-03-11	0.0667
2017-03-12	0.0678	2017-03-12	0.0655
2017-03-13	0.0675	2017-03-13	0.0667

このデータセットにおいては，各日付ごとにほぼ均等に，0.06 から 0.08 の範囲の割合でデータが採取されていることが分かる．

4.2.2 個体識別子

データセットのアクセスを識別するために，全ての通信に，端末に紐付く個体識別子を付与した．個体識別子は，Web 由来データでは 616,336 種類，アプリ由来データでは 3,468,891 種類存在し，両方に存在する個体識別子は 49,309 種類である．個体識別子ごとの出現回数について，最大のもの，最小のもの，平均，および，中央値について，表 3 に示す．

表 3 個体識別子の出現回数

データの種類	最大回数	最小回数	平均	中央値
Web 由来データ	328,653	1	35.704	10
アプリ由来データ	408,162	1	101.967	82

4.2.3 IP アドレス

データセットにおける IP アドレスは，Web 由来データでは 1,203,266 種類，アプリ由来データでは 4,999,830 種類存在する．

4.3.1 節で解説する手段で，IP アドレスから Internet Service Provider (ISP) に変換した場合のデータセットにおける ISP の集計結果を表 4 に示す．

本論文のデータセットのサンプルはすべて国内のものであり，表 4 にあるように，Web 由来データとアプリ由来データでは，どちらとも日本国内の大手 3 大通信キャリア

表 4　ISP の分布（上位 3 つのみ）

データの種類	ISP 名	割合
Web 由来データ	KDDI KDDI CORPORATION	0.293
	GIGAINFRA Softbank BB Corp.	0.189
	DOCOMO NTT DOCOMO, INC.	0.108
アプリ由来データ	DOCOMO NTT DOCOMO, INC.	0.327
	KDDI KDDI CORPORATION	0.235
	GIGAINFRA Softbank BB Corp.	0.180

が上位を占めている．

4.2.4　OS 名・OS のバージョン

データセットにおける OS 名と，そのバージョンの割合を表 5 に示す．ここで，表 5 に示されている ‘OS のバージョン’ の割合は，Web 由来データおよびアプリ由来データのそれぞれにおける上位 3 つの OS バージョンのみの割合である．

表 5　OS 名・バージョンの分布（各 OS 上位 3 つずつ）

データの種類	OS 名	割合	OS のバージョン	割合
Web 由来データ	iOS	0.575	iOS10.2.1	0.324
			iOS10.2	0.061
			iOS10.1.1	0.046
	Android	0.425	Android6.0.1	0.126
			Android6.0	0.078
			Android5.0.2	0.060
アプリ由来データ	iOS	0.816	iOS10.2.1	0.072
			iOS10.2	0.064
			iOS10.1.1	0.017
	Android	0.184	Android6.0.1	0.197
			Android5.0.2	0.156
			Android6.0	0.099

このデータセットは，iOS と Android という 2 つの OS のみで構成されている．また，Web 由来データにおいては，iOS の割合が Android を上回っていることが分かる．その一方で，アプリ由来データにおいては，Android の割合が iOS を上回っていることが分かる．また，OS のバージョンに関しては，Web 由来データにおいては iOS10.2.1 が最も多いことが分かる．それに対し，アプリ由来データにおいては Android6.0.1 が最も多いことが分かる．

4.2.5　機種名

データセットにおける機種名の割合を表 6 に示す．ここで，表 6 に示されている割合は，Web 由来データおよびアプリ由来データのそれぞれにおける上位 3 つの機種名のみの割合である．

Web 由来データにおいてはバージョンが指定されていない ‘iPhone’ が最も多いことが分かる．それに対し，アプリ由来データにおいては ‘iPhone 7,2’ が最も多いことが分かる．

表 6　機種名の分布（上位 3 つのみ）

データの種類	機種名	割合
Web 由来データ	iPhone	0.566
	SOV32	0.038
	SOV33	0.015
アプリ由来データ	iPhone7,2	0.059
	iPhone8,1	0.057
	SO-01G	0.031

4.2.6　アクセス数の分布

データセットにおける，個体識別子ごとのアクセス数の分布を図 2，および，図 3 に示す．なお，図 2 は Web 由来データのアクセス数の分布図を示しており，図 3 はアプリ由来データのものを示している．これらのグラフの横軸は個体識別子ごとのアクセス数に常用対数をとった値を表しており，縦軸は個体識別子数を表している．

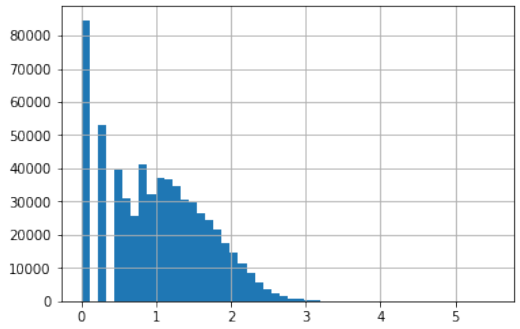

図 2　Web 由来データにおける個体識別子ごとのアクセス数の分布

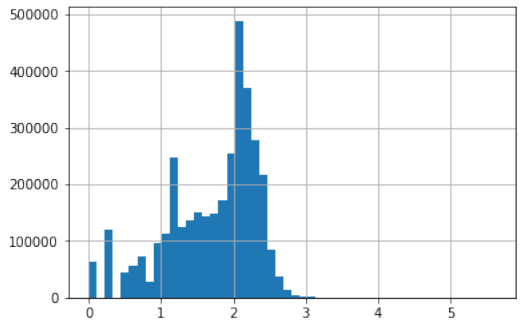

図 3　アプリ由来データにおける個体識別子ごとのアクセス数の分布

図によると，データセットにおける個体識別子のアクセス数の分布は，Web 由来データとアプリ由来データとで異なることが分かる．

4.3　実験データの作成

4.3.1　加工特徴点の作成

本研究では，表 1 で示した特徴点に加えて，それらの特徴点に基づいた付加的な特徴点（以降，加工特徴点と呼ぶ）を追加する．加工特徴点の種類は以下である：

- **タイムスタンプの加工特徴点**

- 年，月，日，時，分，秒，曜日
（UNIX タイム形式から年月日の形式に変換）

- **IP アドレスの加工特徴点**
  - 第一オクテット，第二オクテット，第三オクテット，第四オクテット
  （IP アドレスをオクテットごとに分け，それらを別々の特徴点として使用）
  - ISP 名
  （pyisp[3] を使用して変換）
  - 国名，県名，市区町村名，緯度，経度
  （GeoIP2[2] を使用して変換）
- **OS バージョンの加工特徴点**
  - メジャーバージョン，マイナーバージョン，メンテナンスバージョン
  （OS のバージョンをメジャーバージョンとマイナーバージョンに分け，別々の特徴点として使用）

4.3.2 ベクトルデータの作成

本論文における実験では，フィンガープリントの特徴点を，機械学習のため，ベクトルデータに変換する．ベクトルデータの構造は，Web 由来データの特徴点の値，および，アプリ由来データの特徴点の値からなる組みとした．

さらに，作成したデータセットに，Web 由来データとアプリ由来データを比較した情報である特徴点（以降，組み合わせ特徴点と呼ぶ）を追加する．組み合わせ特徴点の種類は以下である．

- 表 1 及び 4.3.1 で示した特徴点それぞれに対して，Web 由来データとアプリ由来データで特徴点の値が一致しているか否かを表すラベル
- 組み合わせた Web 由来データとアプリ由来データの日にちと時刻の差
- タイムスタンプと OS バージョンの前後関係の矛盾を表すラベル
  （例えば，Web 由来データのタイムスタンプがアプリ由来データのタイムスタンプよりも小さいときに，Web 由来データの OS バージョンがアプリ由来データの OS バージョンよりも大きい場合は矛盾していると言える）
- 緯度経度から計算した直線距離

Web 由来のデータとアプリ由来のデータを組み合わせたベクトルデータに含まれる特徴点の値を数値に変換する．特徴点ごとの数値への変換方法は以下である：

- **OS 名，OS のバージョン，機種名，ISP 名，国名，県名，市区町村名**
  ハッシュ化を行い数字の列に変換する．
- **その他の特徴点**
  そのまま使用する．

上記の通りで変換後，すべての特徴点の値は標準化する．ただし，本論文で使用するデータセットにおいて，出現する国名は日本だけであり，同様に出現する年は 2017 年のみである．よって，実験では，国名の特徴点とタイムスタンプの年の特徴点は使用しない．またクッキー値は特徴点として使用しない．

4.3.3 教師データの作成

Web 由来データとアプリ由来データには，予め，同じデバイスからのものである場合，同一の個体識別子が付与されている．つまり，本論文の実験では，正解データが与えられている．

ベクトルデータにおいては，Web 由来データとアプリ由来データの個体識別子が等しいとき，正解のラベルとして 0 のラベルを割り当て，そうでなければ不正解のラベルとして 1 のラベルを割り当てる．このラベルを教師データとして使用する．

5. 実験方法

5.1 システム環境

本論文での実験に用いたシステム環境は次のとおりである：

- CPU: Xeon E5-2603v4, 1.7GHz/6 cores/15MB
- Memory: 80GB
- GPU: NVIDIA Tesla P100
- OS: Ubuntu 16.04.3 LTS
- 言語: Python 3.6.1
- 利用ライブラリ: Keras 2.1.6(Tensorflow-gpu 1.3.0), scikit-learn 0.19.1

5.2 ニューラルネットワークの構造

本論文での実験に用いたニューラルネットワークの構造及びハイパーパラメータを表 7 に示す．

それぞれの層は全結合層である．ただし，本論文の実験では，最適値の探索をしていない．損失関数として交差エントロピー関数，最適化関数として Adaptive Moment Estimation（Adam）を使用した．Adam のハイパーパラメータはデフォルトの値を使用した．

表 7 ニューラルネットワークの構造

層	ユニット数	活性化関数	DropOut
入力層	特徴点の数	relu	無し
中間層 1	1024	relu	0.3
中間層 2	2048	relu	0.2
中間層 3	1024	relu	0.2
出力層	2	sigmoid function	無し

5.3 ランダムフォレストの構造

本論文での実験に用いたランダムフォレストの学習には

scikit-learn の RandomForestClassifier を使用した．ハイパーパラメータ min_samples_split を学習データ数の平方根である 2300 を使用した．その他のハイパーパラメータはデフォルトの値とした．

5.4 提案手法の精度の検証方法

正解データを元に，各実験において，Precision，Recall，Accurarcy，F_1，F_2，$F_{0.5}$ の各値を求める．

5.5 実験概要

本論文では以下のとおり，大きく 6 つの実験を行う．なお，実験 A，C，D，E，F では，4 節で示した特徴点を全て用いる．

実験 A（深層学習・全特徴点利用）： データセットから Web 由来データとアプリ由来データの組み合わせをランダムに抽出し，深層学習を用いたクロス App-Web トラッキングの実験を行う．この実験では，4 節で紹介した特徴点を全て用いる．

実験 B（深層学習・組み合わせ特徴点のみ利用）： 実験 A と同様に，深層学習を用いたクロス App-Web トラッキングの実験を行う．ただし，この実験では，ISP 名や機種名などを特徴点として用いず，4.3.2 節で定義した組み合わせ特徴点を使う．すなわち，Web 由来データとアプリ由来データのそれぞれの特徴点が完全一致しているか否かを表すラベル，タイムスタンプと OS バージョンの前後関係の矛盾を表すラベル，日にちと時刻の差，緯度経度から計算した直線距離のみを特徴点として用いる．

実験 C（深層学習・200 種限定）： 深層学習を用いたクロス App-Web トラッキングの実験を行う．ただし，この実験では，実験 A や実験 B とは違い，扱う個体識別子の種類をランダムに選択した 200 種類に限定し，それらのデータから生成される組みすべてを使って，学習と検証を行う．

実験 D（ランダムフォレスト・全特徴点利用）： ランダムフォレストを用いて実験 A と同様に，クロス App-Web トラッキングの実験を行う．この実験では，4 節で紹介した特徴点を全て用いる．

実験 E（深層学習・サンプル数 4 種）： 実験 A と同じ内容で，学習用のデータの組み合わせ数を変えて実験を行う．

実験 F（パッシブフィンガープリンティング）： Web 由来データのみを組み合わせ，ブラウザのトラッキングの実験を行う．

5.6 実験において使用するデータセット

実験ごとに，それぞれ以下のようにデータの組み合わせを用いる．それぞれの実験で，特に記載がなければ，抽出した組み合わせデータを 8 対 2 に分割し，それぞれを学習データとテストデータとする．

実験 A，B，D： 以下のように作成した 600 万通りの組み合わせデータセットを用いる．

(1) データセットから出現回数が 2 桁以上 3 桁以下の個体識別子をランダムに 800 種類抽出する
(2) 抽出した個体識別子の Web 由来データとアプリ由来データから，正解のラベルと不正解のラベルが同数になるように，ランダムに組み合わせ 600 万通りの組み合わせデータを作成する

実験 C： 以下のように作成した約 2 億通りの組み合わせデータセットを用いる．

(1) データセットから個体識別子をランダムに 200 種類抽出する
(2) それらの個体識別師を持つデータから全組み合わせを作成する．これを学習データとする
(3) それとは別に，個体識別子をランダムに 50 種類を抽出し，その全組み合わせをテストデータとする

実験 E： 実験 A，B，D のデータセットの作成方法の (2) で，組み合わせデータ数を 1,000 万（E1），100 万（E2），10 万（E3），3 万（E4）の 4 種類に分けて作成したデータセットを用いる．実験 A，B，D のデータセットの作成方法と同様に正解のラベルと不正解のラベルが同数になるように，ランダムに組み合わせる．

実験 F： 以下のように作成した 1,600 万通りの組み合わせデータセットを用いる．

(1) Web 由来データから出現回数が 2 桁以上 3 桁以下の個体識別子を 800 種類ランダムに抽出する
(2) 抽出した個体識別子の Web 由来のデータから，正解のラベルと不正解のラベルが同数になるように，1,600 万通りの組み合わせデータを作成する

6. 実験の結果

実験 A〜F の結果を表 8 にまとめる．

実験 C 以外の結果は，概ね 0.9 以上となり良好であった．

6.1 実験 A（深層学習・全特徴点利用）について

本論文では，深層学習を用いてクロス App-Web トラッキングの実験を行い，精度については，F_1 などが 0.9 以上といずれも良好であった．

とても高精度であるので，結果への疑いの意見も予想されるが，以下により，実験 A の結果はある程度信頼に足りると判断した．

深層学習やランダムフォレストを用いたクロス App-Web トラッキングの実験に先駆けて，著者らは，Web 由来データ，および，アプリ由来データそれぞれの UserAgent の情報，IP アドレス，タイムスタンプを比較して，Web 由来アクセスおよびアプリ由来アクセスの紐付けを試みた．そ

表 8 実験の結果

実験	データセット数	Pre	Rec	Acc	F_1	F_2	$F_{0.5}$
A（深層学習・全特徴点利用）	600 万	0.980	0.999	0.989	0.989	0.984	0.995
B（深層学習・組み合わせ特徴点のみ利用）	600 万	0.929	0.940	0.934	0.935	0.938	0.931
C（深層学習・200 種限定）	約 2 億	0.792	0.487	0.983	0.603	0.527	0.704
D（ランダムフォレスト・全特徴点利用）	600 万	0.977	0.992	0.985	0.985	0.989	0.980
E1：実験 E（深層学習・サンプル数 4 種）	1,000 万	0.999	0.999	0.999	0.999	0.999	0.999
E2：実験 E（深層学習・サンプル数 4 種）	100 万	0.980	0.998	0.988	0.989	0.994	0.999
E3：実験 E（深層学習・サンプル数 4 種）	10 万	0.980	0.975	0.977	0.977	0.976	0.979
E4：実験 E（深層学習・サンプル数 4 種）	3 万	0.973	0.988	0.980	0.981	0.985	0.976
F（パッシブフィンガープリンティング）	1,600 万	0.998	0.999	0.999	0.9999	0.999	0.999

の際，UserAgent のマイナーバージョンを無視して比較したり，タイムスタンプで分・秒を無視して比較したりしたが，結果として，Precision および Recall が同時に 0.6 を同時に超えることはなかった．このことは，フィンガープリントを単純に編集距離等の類似性だけで紐付けることは限界がある一方で，深層学習および，実験 D のランダムフォレストを用いた手法の効果があったとみなせる．

さらに，クロスデバイストラッキングの調査研究 [11] によれば，類似のクロスデバイストラッキングでも同程度の精度であるとのことである．たとえば，Drawbridge 社 [1] の手法の精度として，Precision が 97.3%と示されている．

6.2 実験 B（深層学習・組み合わせ特徴点のみ利用）について

この実験の狙いは，既知のデータセットで学習済みの判別器に対して，未知のデータセットを判別させる際の影響を少なくすることである．すなわち，実験 B では具体的な ISP 名や機種名などを特徴点として使わず，4.3.2 節で定義した組み合わせ特徴点を用いた．

実験 A では F_1 が 0.989 に対し，実験 B では 0.935 だったので，結果は概ね良好であったと言える．ただし，具体的な ISP 名や機種名を特徴点として使用した実験 A の方が F_1 が高いので具体的な ISP 名や機種名そのものにもトラッキングに寄与する情報があると言える．このことは実験 D と符合する．

6.3 実験 C（深層学習・200 種限定）について

この実験は，当該データセット母集団（376M 個）における，Web 由来データおよびアプリ由来データが同一端末からの場合（正解ラベルが付与されるケース）と，異なる端末からの場合（不正解ラベルが付与されるケース）の割合を，再現した状況でのトラッキングを行うのが目的である．

ランダムに選択した 200 種類の個体識別子に限定した上で，データセット母集団からランダムサンプリングではなく，全ての組み合わせを用意して，クロス App-Web トラッキングを実施した．

200 種類の個体識別子に限定した場合，全組み合わせは 261,106,902 通りで，正解ラベルが付与されるケースの割合はその 0.54%であった．ちなみに，個体識別子を 50 種類に限定した場合，全組み合わせは 10,941,812 通りで正解ラベルが付与されるケースの割合は 2.7%であった．

Precision，Recall，Accuracy，および，F_1 は，それぞれ，0.792，0.487，0.983，および，0.603 だった．実験 A に比べて全体的に精度が下がる結果になった．大多数の不正解ラベルが付与されるケースに関しては，十分な学習ができたので，正しく「紐付かない」と判定できており，Accuracy は 0.983 と高い結果になった．その一方，Recall が 0.5 未満になっている．これは，学習において，正解ラベルが付与されるケースの割合が，小さかったのでその影響が精度に表れたのだと推察される．

6.4 実験 D（ランダムフォレスト・全特徴点利用）について

実験 A と同一のデータセットに対してランダムフォレスト用いて実験した．

この実験の目的はクロス App-Web トラッキングにおいて重要な特徴点を抽出することである．なお，F_1 は実験 A の方が高かったので，本論文の実験においてはランダムフォレストよりも深層学習の方が有効であったと言える．

ランダムフォレストによる分類において重要な変数は，その重要度の順で，以下の通りとなった：

(1) OS バージョンが一致したかどうか
(2) メジャーバージョンが一致したかどうか
(3) 機種名が一致したかどうか
(4) マイナーバージョンが一致したかどうか
(5) タイムスタンプと OS バージョンの前後関係に矛盾がないか
(6) IP アドレス第一オクテットが一致したかどうか
(7) OS が一致したかどうか
(8) メンテナンスバージョンが一致したかどうか
(9) ISP 名が一致したかどうか
(10) メジャーバージョンの値

最も重要な変数は，OS バージョンが Web 由来のデータとアプリ由来のデータで完全に一致していることだった．

このことから，クロス App-Web トラッキングは複数のデバイスを跨ってトラッキングをするクロスデバイストラッキングとは重要な特徴点が異なる可能性があることがわかった．

6.5 実験 E（深層学習・サンプル数 4 種）について

この実験では，データ数を 3 万〜1,000 万と 4 パターンで精度を求めた．いずれも良好であった．

しかし，これは，元々の母集団が，Web 由来データのサンプル数は 22M 個，また，アプリ由来データのサンプル数は 354M 個の組み合わせである．その組み合わせからランダム抽出してデータセットを作成している．ランダムサンプルが統計的に有効で，少ないデータセットでも母集団の性質を検証できたと言える．

6.6 実験 F（パッシブフィンガープリンティング）について

この実験は，パッシブフィンガープリンティングを用いたブラウザのトラッキングの試みである．

Web 由来データのみを組み合わせブラウザのトラッキングを試みた．結果として F_1 などが 0.9 以上と良好であった．この結果から UserAgent の情報，IP アドレス，タイムスタンプという少ない情報でもパッシブフィンガープリンティングができ得ると言える．この結果は，高橋ら [10] の実験結果とも符合する．

クロス App-Web トラッキングと同じ特徴点を使い同じように実現できることから，クロス App-Web トラッキングを行う際，副次的に，ブラウザフィンガープリントによる Web トラッキングも実施できると言える．

6.7 実験データの採取期間についての考察

実験 D のランダムフォレストを用いた実験により，OS バージョンやメジャーバージョンが重要な意味を持つことがわかった．

このことは，長期的なデータの取得をする場合，精度への影響が予想される．本論文のデータセットでは，15 日間の採取という短い期間であったので，OS バージョンやメジャーバージョンのアップデートはほぼなかったと言える．よって，長期間でのトラッキングにおいてはまた違う結果になる可能性がある．

7. まとめ

本論文では，クロスアプリケーショントラッキングの一種であるクロス App-Web トラッキングを，深層学習とランダムフォレストを用いて行った．

利用したフィンガープリントは，IP アドレス，UserAgent の情報，タイムスタンプ．また，サンプル総数は，約 376M 個であり，Web 由来データのサンプル数は 22M 個，アプリ由来データのサンプル数は 354M 個である．実験結果として，Precision，Recall，Accurarcy，F_1，F_2，および，$F_{0.5}$ の値は，いずれも 0.9 以上であり良好であった．この結果は，クロスデバイストラッキングについての調査論文 [11] においても，他の手法と比較して，良い結果であると結論づけられている．

実験 F では，パッシブフィンガープリンティングを行った．結果として F_1 などが 0.9 以上だった．このことより，UserAgent の情報，IP アドレス，タイムスタンプという少ない情報でもパッシブフィンガープリンティングが高精度で実現できると言える．

参考文献

[1] Cross-Device Consumer Graph, https://go.drawbridge.com/rs/454-ORY-155/images/Drawbridge-Cross-Device-Consumer-Graph.pdf.

[2] GeoIP2-python, https://github.com/maxmind/GeoIP2-python.

[3] pyisp, https://github.com/ActivisionGameScience/pyisp.

[4] Smartphone users worldwide 2014-2020, https://www.statista.com/.

[5] The State of Cross-Device Commerce, https://www.criteo.com/wp-content/uploads/2017/07/Report-criteo-state-of-cross-device-commerce-2016-h2-UK.pdf.

[6] 世界 40 カ国、主要 OS・機種シェア状況 【2018 年 3 月】, https://www.globalmarketingchannel.com.

[7] Boda, K., Földes, A. M., Gulyás, G. G. and Imre, S. User Tracking on the Web via Cross-browser Fingerprinting, Proceedings of the 16th Nordic Conference on Information Security Technology for Applications, NordSec'11, Berlin, Heidelberg (2012), Springer-Verlag.

[8] Brookman, J., Rouge, P., Alva, A. and Yeung, C. Cross-device tracking: Measurement and disclosures, *Proceedings on Privacy Enhancing Technologies*, **2017**, 2 (2017), 133–148.

[9] Zimmeck, S., Li, J. S., Kim, H., Bellovin, S. M. and Jebara, T. A privacy analysis of cross-device tracking, 26th USENIX Security Symposium (USENIX Security 2017) (2017).

[10] 高橋和司, 安田昂樹, 種岡優幸, 田邉一寿, 細谷竜平, 野田隆文, 齋藤祐太, 小芝力太, 齋藤孝道, HTTP ヘッダのみを用いた Browser Fingerprinting の考察, 暗号と情報セキュリティシンポジウム (SCIS) 2018 (2018).

[11] 齋藤孝道, 細谷竜平, 森達哉, クロスデバイストラッキング技術に関する調査：2018 年版, 2018 Computer Security Symposium (2018).