MADMAX: Extreme Learning Machineを用いたブラウザベース悪性ドメイン検知アプリケーション

岩花 一輝[1] 竹村 達也[1] 鄭 儒謙[1] 芦澤 奈実[1] 梅田 直希[1] 佐藤 航大[1] 川上 遼太[1] 清水 嶺[1]
知念 祐一郎[1] 矢内 直人[1]

概要：高速かつ高精度の悪性ドメイン検知において, 未知の悪性ドメインを検知するために, 機械学習が注目されている. 本稿では, 単一の中間層を持つニューラルネットワークである extreme learning machine (ELM) を用いたブラウザベースの悪性ドメイン検知アプリケーションである *MADMAX (MAchine learning-baseD MAlicious domain eXhauster)* を提案する. 既存の ELM を用いた悪性ドメイン検知と比較して, MADMAX では 2 つの新たな手法を導入している. まず, 高精度かつ高速なドメイン検知を実現するため, permutation importance に基づいた最適な特徴量の選択を行う. さらに, 日々進化し続ける悪性ドメインの更新にも対応できるように, 学習モデルを再学習させるリアルタイム学習を行う. MADMAX では最適な特徴量の選択を通じて, 既存研究と比較して精度とスループットの改善に成功した. さらに, リアルタイム学習で学習したモデルは安定して未知の悪性ドメインを検知し続けたが, リアルタイム学習を行わないモデルは未知の悪性ドメインを検知できず, 精度が低下することを確認した. MADMAX のソースコードは GitHub を通じて, 公開利用可能である.

1. 背景

悪性ドメインによるサイバー犯罪の被害が年々増加しており, 例えば新型コロナウイルスが蔓延した後も COVID-19 に便乗したフィッシングサイトへの誘導などを目的とした多くの悪性ドメインが攻撃者によって生成されている. これら悪性ドメインに対する古典的な対策は悪性ドメインの悪性リストを事前に作成しておくことだった. しかし, 新しいドメインがドメイン生成アルゴリズム (DGA) により自動生成される昨今においては, 既知のドメインに対しての悪性リストを作成する対策は不十分であり, 未知の悪性ドメインに対する防御策が必須である. 上述した背景から機械学習を応用した悪性ドメイン検知が近年注目を集めている [60]. しかし, 被害を受けるユーザとのインターフェースとして最も近くに位置するブラウザ上で, 機械学習を利用している悪性ドメイン検知ツールは, 著者らの知る限り存在しない. 既存サービスとしては悪性ドメイン検知を行う VT4Browsers[*1] が Google Chrome のアドオンとして存在するが, これは既知の悪性ドメインに対するリストに基づいた検知であり, 急増する未知の悪性ドメインに有効とは言えない. また, ブラウザ非依存な機械学習を用いた悪性ドメイン検知の既存研究 [5,41,54,55,57] もあり, それらをブラウザ実装することも考えられる. しかし, 上述した既存研究では複雑かつ大規模なアーキテクチャを持ち, 悪性ドメイン検知において高い精度を実現するために多大な学習時間を必要とする. このため, ブラウザのようなユーザがリアルタイムに利用するサービスへの実装には適していない.

本稿では, ユーザのブラウジング時に平行して機械学習を用いることで悪性ドメインを検知するアプリケーション *MADMAX (MAchine learning-baseD MAlicious domain eXhauster)* を提案する. MADMAX はブラウザにアドオンとしてインストールすることで, ブラウザ上で自動的に悪性ドメインを検知する. つまり, MADMAX は悪性リストにない悪性ドメインをブラウザ上で自動的に検知できる点が, 既存サービスに対しての優位点である.

一般に, 機械学習をブラウジングのようなリアルタイム処理での環境に導入する際は, 主に 2 つの技術的問題がある. 1 つ目は, ドメインに関する特徴量の抽出など悪性ドメイン検知に付随する処理が多いため, 悪性リストを用いた方式と比較して, 悪性ドメイン検知に時間がかかることが想定される. 2 つ目は, 機械学習の処理時間と精度のトレードオフである. 一般には精度を上げるためには複雑な機械学習モデルが必要となるが, このとき機械学習の複雑

[1] 大阪大学

[*1] https://chrome.google.com/webstore/detail/vt4browsers/ efbjojhplkelaegfbieplglfidafgoka

な計算が原因でスループットも低下する．加えてサイバーセキュリティに機械学習を用いる際は, 新種の悪性ドメインの出現により, 重要な特徴量の変化（コンセプト・ドリフト）が起きやすい [38]．このため, 常に再学習, つまりリアルタイム学習も行う必要がある．リアルタイム学習において, 学習のスループットは極めて重要な要因である．上述した既存研究で扱われているような複雑かつ大規模なアーキテクチャは学習時間が長くなることから, むしろリアルタイム学習には適さない．機械学習の研究においては学習時間は一般には考慮されないが, ブラウザへの導入を考えた場合は必須といえる．このため, ブラウザでは機械学習の導入が進んでいないと著者は考えている．

本稿では, 上述した問題に対する潜在的な解決手法として extreme learning machine (ELM) [26] を用いた悪性ドメイン検知 [48] に着目した．これは負荷が重い誤差逆伝播がないまま特徴量を学習できる単層のニューラルネットワークである．つまり, ドメインの学習および悪性ドメインの検知いずれも高速に行えるため, 上述した技術的問題の解決が期待できる．

しかしながら, 既存研究 [48] では ELM で悪性ドメインが検知できるかを評価したのみである．すなわち, アプリケーションとしての実装は行っておらず, アプリケーションレベルでの悪性ドメイン検知精度及びスループットは不明である．具体的には, 実際にユーザがブラウザ上で機械学習を利用する際は, 検知に必要な特徴量をリアルタイムで取得する必要がある．すなわち, アプリケーションとして実装した際に, 特徴量抽出の時間までを含めた性能を評価する必要だと言える．加えて, 悪性ドメインの更新に対して, モデル自体の更新も必要となる．このとき, 高速な学習に加え, 更新されたモデルが新たに出現するであろう悪性ドメインを継続して検知できるかも確認する必要となる．つまり, 既存研究の結果だけでは, ブラウザ環境における実用性の可否の判断はできない．

以上の背景から, 本稿ではアプリケーションとして MADMAX を実装した．さらに既存研究 [15] が選択した特徴量を活用することで, 高速かつ高精度なモデルを構築できる特徴量を厳密に選択した．すなわちユーザにとって最適なアプリケーションを実現し, その結果として, 既存研究 [48] と比較した精度およびスループットの改善に成功している．加えて, リアルタイム学習によるモデルの更新も併せて行うことで, 新出する未知の悪性ドメインに対応できることも確認している．他のニューラルネットワークを用いた既存の悪性ドメイン検知手法 [1, 23, 53] と比較して, MADMAX は学習時間と推論時間の短縮が期待できる．本稿での貢献を要約すると,

- ELM を用いたブラウザベース悪性ドメイン検知アプリケーションとして, MADMAX を提案し, さらに, ブラウザアドオンとしてのプロトタイプを実装した．実装した成果物は GitHub[*2]で公開されている．
- モデルに用いる特徴量を選別することで, どの特徴量が ELM の検知精度とスループットにとって最適か明らかにした．この結果として, 悪性ドメインの検知精度に関して, MADMAX は既存研究 [48] を上回った．
- リアルタイムでの学習によりモデルの更新を行えること, つまり, 悪性ドメインの更新にも対応できることを確認した．これに対し, 更新のないモデルは悪性ドメインの傾向変化が原因で, ドメインの検知精度が劣化することも確認している．

1.1 論文構成

本稿の構成を以下に示す．2 節ではドメイン名や機械学習を用いた悪性ドメイン検知の説明, 及びその定式化について述べる．3 節では MADMAX のシステム要件と, 本稿における学術的問いを述べる．4 節では MADMAX の具体的な構成方法について説明する．5 節では実験結果から MADMAX の性能について評価する．また MADMAX に対する考察や制約条件については 6 節で述べる．7 節では関連研究について述べ, また, 本稿の結論と今後の展望については 8 節で述べる．

2. 準備

本節では, 本稿の背景知識となるドメインと機械学習ベースの悪性ドメイン検知について述べる．

2.1 ドメイン名

ドメイン名とは, ネットワーク上のホストについた名前で, 代表的には Domain Name System (DNS) によって運用される．ドメイン情報は, 例えば ウェブサービスにおいて, IP アドレスなどの物理的な配置と論理的に切り分けて運用される情報であり, 一般には, ドメイン名はゾーンと呼ばれる名前空間に従って階層的に管理がされている．最上位のドメインを一般にルート (root) と呼び, 代表的なドメインには.com や.jp などがある．このようなドメインは最もレベルの高いドメインとして Top Level Domain (TLD) と呼び, それぞれの TLD の下には, 各国別, 組織別のドメインが存在することで, ゾーンと呼ばれる階層的かつ分散的なドメイン情報を管理することが可能となる．

2.2 機械学習を用いた悪性ドメイン検知

機械学習を用いた悪性ドメイン検知は, 与えられたドメインに対して悪性かどうか推論する．大まかには, 機械学習モデルがドメインの特徴量とその良性, 悪性のクラスを学習することで, 目的のモデルが得られる．その後, 推論を

[*2] https://github.com/kzk-IS/MADMAX

行う際は, 推論対象となるドメインの特徴量をモデルの入力として, そのドメインが悪性かどうかを推論する. 近年では特にニューラルネットワークを用いた悪性ドメイン検知が注目されている.

問題の定式化: 機械学習を用いた悪性ドメイン検知において, 本稿でのアプローチを定式化する. $\mathcal{F} = \{f_1, \cdots, f_l\}$ を特徴量とする. 各ドメイン $d_i \in D$ は特徴量 $F_i = \{f_{i,1}, \cdots, f_{i,l}\}$ を持つ. ただし, D はドメインのデータセット, $l \in \mathbb{N}$ は特徴量の個数 F_i を表す. さらに, 各ドメイン $d_i \in D$ は ラベル $L_i \in \{0,1\} \subseteq L$ を持つ. ただし, 0 は良性ドメイン, 1 は悪性ドメインを表すラベルである. D の大きさ, つまりドメインの個数が与えられた条件で, $DFL = \{(d_1, F_1, L_1), \cdots (d_n, F_n, L_n)\}$ はドメインと特徴量, そしてドメインのラベルの対応を示す. $Model = M(DFL)$ を訓練済のモデルとする. ただし M は学習アルゴリズムである. 本稿における目的とは, 未学習のドメイン d_t に対し, 特徴量 $F_t = \{f_{t,1}, \cdots, f_{t,l}\}$ を抽出することによって, $L_t = Model(F_t)$ を得ることにより, 良性か悪性かを判定することである.

2.3 Extreme Learning Machine(ELM)

ELM [26] とは高速に動作する機械学習アルゴリズムであり, 単一の隠れ層を持つニューラルネットワークを訓練するアルゴリズムである. 大まかには, 効率的に汎化性能を求めることが可能であり, 例えば SVM(Support Vector Machine) [19] と同等, またはそれ以上の大局的な最適解を求めることができる [25]. このため, 生命科学やコンピュータビジョンなど様々な応用が検討されている. 一方, 悪性ドメイン検知の文脈では, Shi ら [48] を除いて, 使われていない. 1 節で述べたとおり, コンセプト・ドリフト の考慮が悪性ドメイン検知では特徴的であり, 本稿で検討するリアルタイム学習は, 上述した文献とは異なる課題といえる.

古典的な ELM [26] では教師あり学習の文脈で, 分類問題と回帰問題の両方で利用される. 本稿では古典的な ELM を利用するが, 不均衡なデータや推論誤差に取り組むための拡張 ELM も存在する. 例えば, 分類問題向けにクラス固有のコストを導入する CCR-ELM [56] は, 不均衡な分類データに対応できる. また, 入力と出力の間の特徴マッピングを構築するためのベースライン層と残差補償のための層を備えた多層構造を導入した RC-ELM [62] は, 非線形で確率的な性質によって予測誤差が出てくる回帰問題に対応できる. 同様に, ガウス誤差のみでは説明できないような非常に複雑なノイズにも対応するために, ガウスノイズと非ガウスノイズによるモデリング機能と頑健性を改善する robust ELM（R-ELM [61]）がある.

さらに, 株価予測など実際のアプリケーションのように, 事前にデータを用意するかわりに, 連続的に取得されるデータへの応用として online sequential ELM (OS-ELM [30]) が存在する. OS-ELM ではブロックをスライドさせることで再学習をかける手法が取られており, 本稿のリアルタイム学習と類似している. さらに, OS-ELM に時系列を考慮した拡張として FOS-ELM [65] があり, 忘却因子 (forgetting factor) を導入することによって, 古いデータよりも新しいデータに偏りを持たせて時系列情報を学習している. 一般にコンセプト・ドリフト は過去と未来の文脈を考慮すべきことから, 上述した手法を応用することは有益である.

3. MADMAX

MADMAX (MAchine learning-baseD MAlicious domain eXhauster) は機械学習に基づく悪性ドメイン検知用ブラウザ・アプリケーションである. 本節ではまず問題設定として, MADMAX のシステム要件について述べる. 次に, その潜在的な解決手法として ELM に基づく悪性ドメイン検知及び本稿における学術的問いについて述べる.

3.1 システム要件

MADMAX は機械学習を用いることで, ブラウザ上で高速かつ高精度で悪性ドメインを検知するアプリケーションである. 具体的には, 図 1 に示すように, ブラウザ拡張となるアドオンを介してアクセス先のドメインをサーバに送信することで, サーバ上で動作する機械学習モデルが悪性か良性かの判定を行うサーバ・クライアント型システムである. 一般にユーザはブラウザを用いてウェブサイトにアクセスすることから, ウェブブラウザ利用時の脅威を防ぐことが重要である. また, アドオンは様々なブラウザで利用されることから汎用性が高く, ユーザも気軽に導入できる. MADMAX の機能について, ユーザ側およびサーバ側それぞれの観点から以下に説明する.

3.1.1 ユーザ側が持つ機能

ユーザ側はアドオンとして MADMAX をブラウザに導入することで, 悪性サイトに接続するときに, 自動的に警告画面を出力する機能を得る. 具体的に, 接続先の URL からドメインを抽出し, そのドメインをサーバに送信する. 次に, サーバから受けとった検知結果が悪性なら, 警告画面を表示する. ドメインが良性なら, そのまま接続できる.

3.1.2 サーバ側が持つ機能

サーバ側は, ユーザ側からドメインを入力として受け取り, サーバが保有する学習済みモデルを用いて, ドメインが悪性か良性かを判定する. その際, 以下の 3 つの機能を通じて, サーバ上で悪性ドメイン検知を行う.

(1) ドメインの特徴量抽出: 一般にドメイン d_i そのものが持つ特徴量は少ない. すなわち, 高精度の推論には不十分であるため, 与えられたドメインに関して DNS レコードなどの特徴量を抽出することが一般的な方法である. しかし, この特徴量抽出は一般に時間がかか

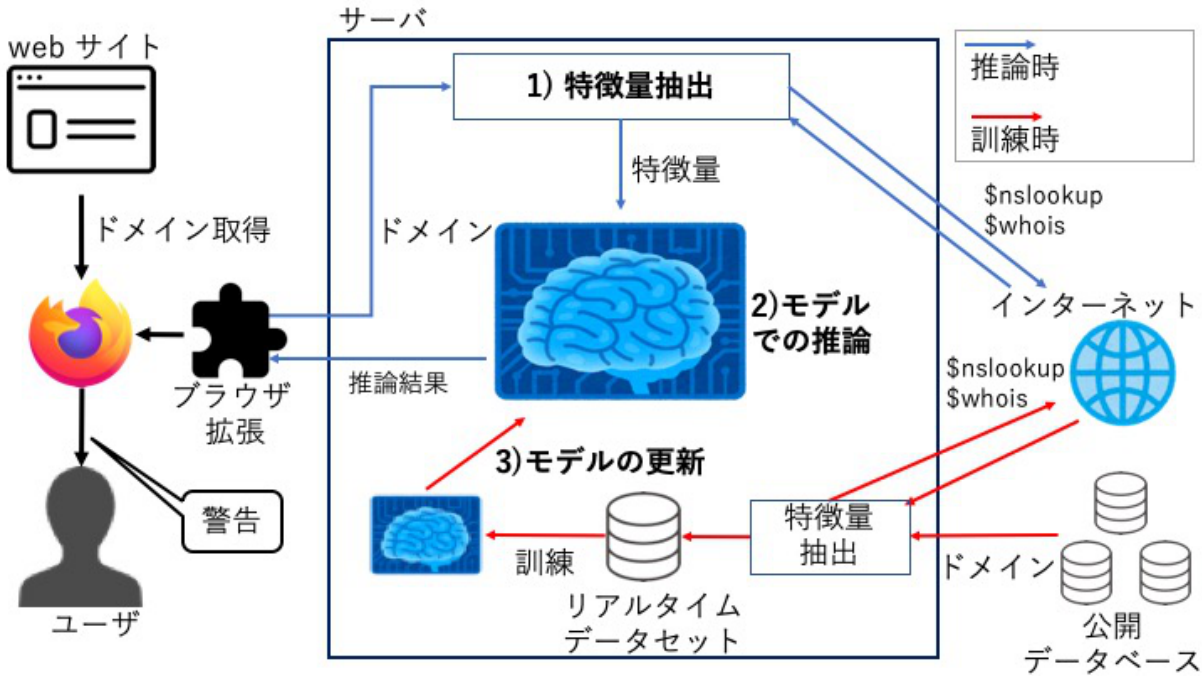


図 1: MADMAX の全体像

る処理であり, アドオンのようなリアルタイムで悪性ドメインを検知する環境では, その特徴量抽出時間も考慮する必要がある.

(2) モデルの推論処理: 推論処理では, ドメイン d_i の特徴量 F_i を入力として学習済みモデル $Model$ を用いて推論処理 ($Model(F_i)$) を行い, d_i が悪性 $L_i = 1$ か良性 $L_i = 0$ かを判定する. また, 特徴量抽出の時間だけでなくモデルの推論時間も含めユーザの待機時間となるため, 特徴量抽出に加え, モデルの推論時間も高速である必要がある.

(3) モデルの更新処理: 最新の悪性ドメインを用い, 現在利用しているモデルを再学習することで, 未知の悪性ドメインへの検知精度を継続して維持する必要がある. このとき, 再学習によるモデルの更新が長いと, 更新されるまでの間はユーザは新種の悪性ドメインの脅威にさらされる可能性がある. そのため, リアルタイム学習されるモデルは高い精度で悪性ドメインを検知し, かつ, モデルの更新が高速であることが必要となる.

3.2 ELM による悪性ドメイン検知

MADMAX では, 機械学習を用いた悪性ドメイン検知として, 学習が高速かつ検知精度が高い extreme learning machine (ELM) [26] を用いた手法 [48] に着目する.

通常のニューラルネットワークを用いた学習は入力層, 中間層, 出力層の大きく 3 層からなり, 中間層の数や各層に含まれるニューロンの数を増やすことで精度の向上が見込める. このとき, 中間層の層数に依存してモデルの学習時間, および入力層から出力層にかかる推論時間も増加する. これに対し, ELM は単一の中間層からなるニューラルネットワークであり, 重み計算に誤差逆伝播の代わりに擬似逆行列を用いる. これにより, 学習が高速に行える. 加えて, 推論も高速であることから, リアルタイム性が重視されるブラウザ上での悪性ドメイン検知にも適している.

ELM の推論処理は以下のとおり定式化される. ドメイン d_i の特徴量 $F_i = \{f_{i,1}, \cdots, f_{i,l}\}$ に関して, 以下のとおり計算する:

$$\sum_{j=1}^{N} \beta_j A(W_j \cdot F_i + b_j) = O_i. \tag{1}$$

ここで N は隠れ層におけるノードの数, $A(\cdot)$ は活性化関数, W_j は j 番目のノードに関する入力層の重みベクトル, b_j は j 番目のバイアス項, また, O_i は推論の出力をそれぞれ意味する. 加えて, β_j は ELM の学習によって得られるパラメータをそれぞれ表す.

ELM の学習処理として $||H\beta - L||$ の最小化問題を解く. ここで, $\beta = (\beta_1^T, \cdots, \beta_N^T)$ であり, H は以下のとおり定義される:

$$H = \begin{pmatrix} A(W_1 \cdot F^1 + b_1) & \cdots & A(W_N \cdot F^1 + b_N) \\ \vdots & \ddots & \vdots \\ A(W_1 \cdot F^n + b_1) & \cdots & A(W_N \cdot F^n + b_N) \end{pmatrix}. \tag{2}$$

上記を解くために, まず ELM では W_j と b_j をランダムに生成する. このとき, 最小化問題は $||H\beta - L|| = 0$ を満たすような β を計算する問題とみなせる. すなわち, ELM は $\beta = H^{\dagger}L$ を計算することで, β を決定可能である. ここで,

$H^{\dagger}$ は擬似逆行列 [26] である．ELM は逆行列計算を 1 度行うことで学習を終えるため, 誤差逆伝播を用いることで W, b, β を計算する深層学習よりも高速に学習を終えることが可能である．

3.3 学術的問い

MADMAX の設計の主な着想は, ユーザがウェブブラウザを利用しながら, アドオンを通じて悪性ドメインを検知できることである．これを実現するために, MADMAX は悪性ドメインの検知を高速に行えること, また, 常に未知の悪性ドメインに対応できることが望ましい．具体的に, MADMAX の設計にあたり, 主に以下の 2 つの問いを検討する．

1 つ目の観点は, 特徴量抽出を含めたサーバ上のスループットと構成されるモデルの精度のトレードオフである．一般に, 特徴量が多いほうが高い精度が期待できるが, 特徴量の数に比例して特徴量抽出にかかる時間が増加する．既存研究 [48] では悪性ドメイン検知に関して高いスループットが実験的に示されているが, 特徴量抽出時間を含むアプリケーションとしての性能は示されていない．つまり, 高い精度及び高いスループットを満たす特徴量集合 $\mathcal{F}' \subseteq \mathcal{F}$ は依然として非自明である．

2 つ目の観点は, ELM が常に高い精度で未知の悪性ドメインを検知できるかどうかである．具体的に, DGA により悪性ドメインは常に生成されていることから, モデルはリアルタイムに学習・更新されることが望ましい．既存研究 [48] により ELM は学習のスループットが高いことが示されているが, 新たに出現する未知の悪性ドメインに対応し続けることができるよう, 更新し続ける必要がある．既存研究では, 継続的に学習及び検知が可能かは示されていない．

本稿の目的は上記 2 つの問いを明らかにすることである．

4. 具体的な構成方法

本節では, MADMAX の具体的な構成方法について説明する．まず, 前節で述べた学術的問いへの解決アプローチを示した後, 具体的な手法として 最適特徴量選択とリアルタイム学習について説明する．

4.1 提案手法の概要

MADMAX を実現するにあたり, 3.3 節で述べた学術的問いに答えるべく, 2 つの手法を導入する．まず最初の学術的問いに答えるために, 最適特徴量選択による精度とスループットの間のトレードオフを明らかにする．次に, 最適化された特徴量を用いて, 連続的な悪性ドメイン検知が可能かどうかを明らかにする．

まず, 1 つ目の学術的問いへの答えとして, 最適特徴量選択を提案する．著者らの先行研究 [15] で示されている特徴量の集合に着目し, 各部分集合に関して, 特徴量の抽出含む処理時間と, ELM にその部分集合を入力した際の検知精度のトレードオフを明らかにする．より具体的には, 全体の特徴量の集合に対して permutation importance [9] を利用することで, どの特徴量が悪性ドメイン検知精度に影響しているか, すなわち特徴量の重要度のランキングを計算することが可能である．そのとき, 重要度のランキングから特徴量と精度の関係および, 検知時間の関係を明らかにすることで, 高いスループットかつ高精度な特徴量集合 $\mathcal{F}'$ の選別を行うことが可能である．

次に, 2 つ目の学術的問いへの答えとして, リアルタイム学習を提案する．リアルタイム学習は時刻が進むに連れて新たに出現する未知の悪性ドメインに対しても検知性能を維持するための手法である．具体的には, サーバが悪性ドメインデータセット DFL を更新することで, 一定時間置きにモデル $Model = M(DFL)$ を再学習する．直観的には, 傾向が変化する未知の悪性ドメインをモデルが迅速に学習することが期待できる．悪性ドメインデータセットの更新では, 著者らの先行研究 [15] と同じ公開データベースからリアルタイムで悪性ドメイン D を取得し続ける．このとき, 1 つ目の手法, 最適特徴量選択を用いて, 明らかにした最適な特徴量集合 $F' \subset F$ を抽出することでモデルを再学習する．

4.2 最適特徴量選択

ここでは MADMAX で用いる特徴量および, 特徴量の重要度に基づいた特徴量の選別について説明する．最適特徴量選択の流れは図 2 に示す．まず, あらかじめ決められた特徴量集合から permutation importance に基づき, 特徴量の重要度をランキング付けする (Rank features based on permutation importance) ．つぎに, 閾値 T(the threshold T) に基づいて特徴量集合を選び実験 (Experiment) を行う．最後に実験結果に基づいて, 最適な特徴量を選ぶ (Select optimized features).

4.2.1 特徴量集合

本稿では, 著者らの先行研究 [15] と同様の特徴量を用いる．特徴量は計 25 個で, 文字列ベース特徴量, DNS ベース特徴量, ウェブベース特徴量の 3 種類に分類されている．以下はその 3 種類の特徴量について説明する．

4.2.1.1 文字列ベース特徴量

文字列ベース特徴量はドメイン名の文字列から得られる情報を指しており, 悪性ドメインをドメイン名から判別できるかを議論する．具体的には, 母音数, 母音割合, 母音種類, 子音数, 子音種類, 母音子音の切替, 数字数, 数字割合, 数字アルファベット切替, 記号等の文字数, 最大繰返し数, エントロピー [47], 評判値 [64] の 14 種類である．

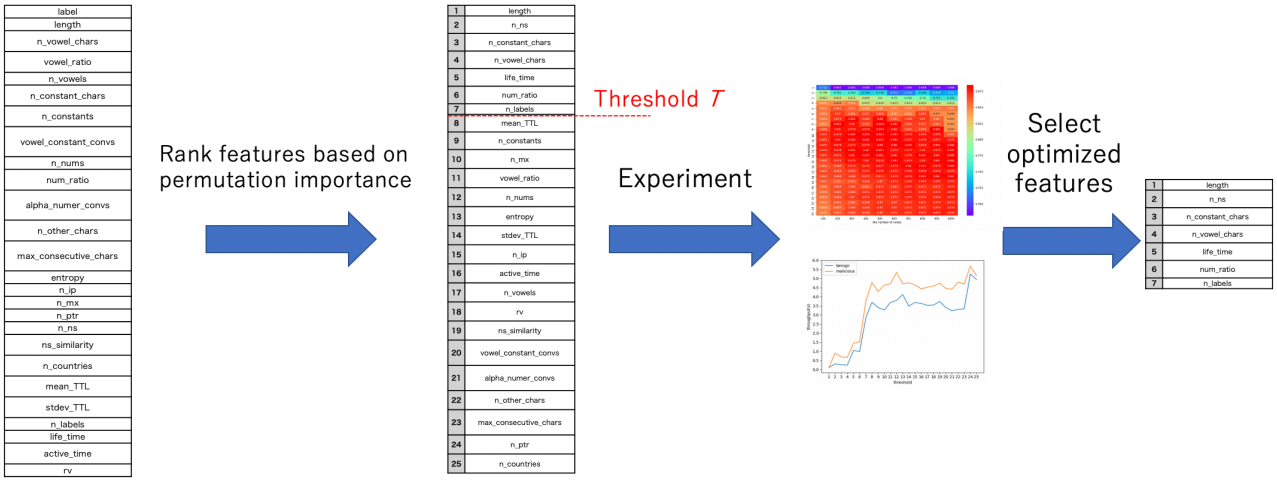


図 2: MADMAX における最適特徴量選択の流れ

4.2.1.2 DNS ベース特徴量

DNS ベース特徴量は, ドメイン名に関する DNS レコードを介して得られる情報であり, 悪性ドメインと良性ドメインの DNS レコードの違いを議論する. 具体的には IP 数, PTR 数, NS 数, MX 数, NS 間の文字列類似度, IP 所属国の数, TTL 平均, TTL 標準偏差の 8 種類である.

4.2.1.3 ウェブベース特徴量

ウェブベース特徴量はドメイン自体から付随して得られる情報を意味しており, 悪性ドメインが提供するコンテンツの傾向を議論する. 具体的には, 該当するウェブページコンテンツの HTML タグ数, WHOIS 情報のライフタイム (生成日と有効期限の差), WHOIS 情報のアクティブタイム (生成日と最終更新日の差) の 3 種類である.

4.2.2 特徴量の最適化

以下に特徴量の最適化について述べる. まず, 上記に述べた特徴量を含むデータセット [15] を用いて, 重要度の計算として permutation importance [3] を適用する. これにより, 特徴量集合 $\mathcal{F} = \{f_1, f_2, \cdots, f_l\}$ における各特徴量 f_j の重要度を計算する. その結果得られる特徴量をランキング化し, 特徴量ランキングと呼ぶ. Permutation importance とは大まかには, 特徴量集合 $\mathcal{F}$ に対して, 各特徴量 f_j ごとに値をランダムに入れ替えることで, ランダムに入れ替えた特徴量が学習済モデルに対する影響を評価する手法である. このとき, ランダムに入れ替えない場合, すなわち元々のデータセットを用いて作成された場合のモデルの出力結果との差が大きいほど重要な特徴量であり, 低いとあまり重要ではない特徴量と言える.

続いて, 重要度ランキングから用いる特徴量の数を決定する閾値 T を設定する. その後, ランキング上位 T 個の特徴量を利用し, その特徴量を用いたモデルの精度および, スループットを評価する. つまり, 全体の特徴量において permutation importance を用いて, 事前に特徴量の重要度ランキングを計算することで, 悪性ドメイン検知に有益な特徴量の組み合わせを選択できる.

4.3 リアルタイム学習

本節ではデータセットのリアルタイムな更新と, 更新済みデータセットを用いた再学習の方法について説明する. 図 3 にリアルタイム学習の流れを示す. 良性ドメインデータセット (benign dataset) とリアルタイムな悪性ドメインデータセット (real-time mallicious dataset) は公開データベース (open database) からリアルタイムに取得する (pull by real time) ドメインのそれぞれに特徴量を抽出する (extract features) ことにより更新 (update) される. モデルは n 個の良性ドメインと最新の n 個の悪性ドメインのデータを用いて再学習 (retrain) される. 赤色と緑色の箱で囲まれた部分はそれぞれ各時刻において更新されたデータセットを示している.

4.3.1 リアルタイムなデータ収集

リアルタイムで再学習を行うために利用するデータセットの更新方法について説明する. 大まかには, 訓練用データセットは著者らの先行研究で利用した公開データベースから最新の悪性ドメインデータを取得し続けることによって更新される. 良性ドメインは Tranco[*3]から上位にあるデータを取得し, 悪性ドメインについては URLhaus[*4], CyberCrime Tracker[*5], PhishTank[*6]からそれぞれリアルタイムで取得し続ける. 未知のドメイン d_i に関して 4.2 節で示した最適特徴量集合 $\mathcal{F}'$ をリアルタイムで抽出することによって訓練用データセット DFL を更新し続ける.

*3 https://tranco-list.eu/
*4 https://urlhaus.abuse.ch/
*5 https://cybercrime-tracker.net/
*6 https://www.phishtank.com/

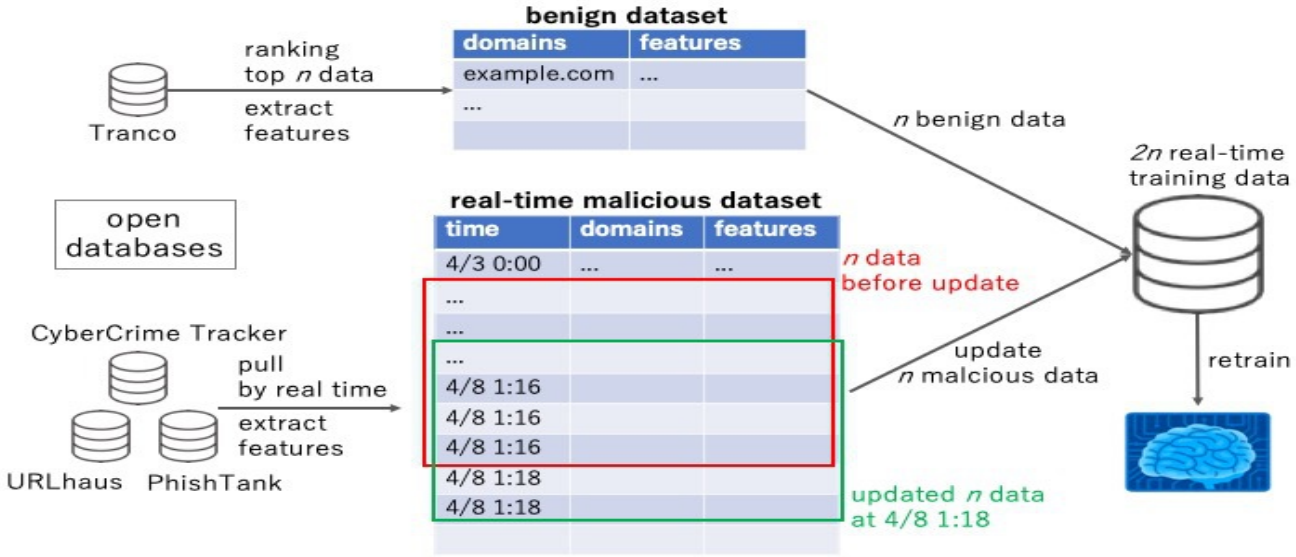


図 3: MADMAX におけるリアルタイム学習の流れ

4.3.2 再学習

更新済みデータセットを利用した再学習手法を提案する．リアルタイムで更新されるデータセットを用いた再学習によって，機械学習モデルは未来に新たに生成される未知の悪性ドメインに対してさえ可能な限り高速で高精度で検知することが可能となる．MADMAX における再学習の設計では，良性データと悪性データをそれぞれ n 件ずつ ELM モデルに入力として与える．より詳細には，n 件の良性ドメインは初期設定として 1 度だけ与えられ，それ以降は更新されない．一方で，悪性ドメインは常に更新され，ELM モデルは一定時間（例えば 30 秒など）ごとに更新済みデータセットの最新の n 件のドメインを入力として学習する．この際，高速学習可能な ELM を用いているため，毎度のモデルの更新に要する時間を抑えることができる．これにより，時間が進むにつれて新しく出現する未知の悪性ドメインをリアルタイムで検知し続けることができる．また，新種の悪性データが一定数，例えば n 件現れた地点で最新 n 件を利用することも再学習の手法として考えることができるが，そのような再学習に関しても ELM は高速で学習を行うことができる．

5. 実験

本節では，3.3 節で述べた学術的問いの観点から MADMAX の性能を評価するために 2 種類の実験を行った．特に，最適特徴量選択とリアルタイム学習について議論する．実験の目的と設定を MADMAX の実装を含めて説明して実験結果を示す．

5.1 実験目的

実験目的は以下の 2 つである．1 つ目に，MADMAX のスループットと検知精度とのトレードオフを確認し，それから検知が高速かつ高性能にできる特徴量集合 $\mathcal{F}'$ を探すことを目的とする．これを行うために，用いる特徴量の数 T と中間ノード数 N の変化に対し，検知精度の変化を確認する．さらに，それに付随して MADMAX 導入によるスループットを評価する．このとき，検知精度として今回，F1 score, G-mean, Accuracy, Precision, Recall を用いる．2 つ目に，4.3 節で示したリアルタイム学習によって未知の悪性ドメインを継続して検知できるかを確認する．具体的には，1 つ目の実験にて明らかになった特徴量集合 $\mathcal{F}'$ を利用した際の，再学習を行うモデルと再学習を行わないモデルの性能について上に示した指標を用いて比較評価する．これにより，リアルタイム学習が高速な学習に加えて未知の悪性ドメインの検知もできるということを確認する．以降では簡単のために，再学習を行わないモデルを *normal model* と，再学習を行うモデルを *retrained model* と表記する．

5.2 実験設定

5.2.1 実装

MADMAX の各機能を具体的にどう実装したのかについて説明する． まず，MADMAX のユーザ側の機能は全て Firefox (バージョン 81.0.2) のアドオン機能として JavaScript[*7]を用いて実装した． 一方で，MADMAX のサーバ側の機能は全て Amazon EC2 c4.8xlarge 上に Python[*8]とそのウェブアプリケーションフレームワークである Flask[*9]を用いて実装した．特に，ELM は数値計算ライブラリ NumPy[*10] を用いて実装を行った．また，特徴量の重要度を計算する permutation importance は機械学習

[*7] https://developer.mozilla.org/ja/docs/Web/JavaScript
[*8] https://www.python.org/
[*9] https://flask.palletsprojects.com/en/1.1.x/
[*10] https://numpy.org/

ライブラリ scikit-learn[*11]を用いて実装を行った.

5.2.2 評価指標

MADMAX を定量的に評価するために, 4 つの用語, TP(True Positive), TN(True Negative), FP(False Positive), FN(False Negative) の説明を述べる. TP は悪性のクラスに分類された悪性ドメインの数, TN は良性のクラスに分類された良性ドメインの数, FP は誤って悪性ドメインに分類された良性ドメインの, FN は良性ドメインに分類された悪性ドメインの数である.

上述した 4 つの観点から評価指標について述べる.

Accuracy: 全体のドメイン数に対して, 正しく検知されたドメインの割合である. Accuracy の定義は以下の通りである.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

Precision: 悪性ドメインとして検知されたドメインの数に対して, 実際に悪性ドメインだったドメイン数の割合である. Precision の定義は以下の通りである.

$$Precision = \frac{TP}{TP + FP}.$$

Recall: 全体の悪性ドメインの数に対して, 正しく悪性として検知されたドメイン数の割合である. Recall の定義は以下の通りである.

$$Recall = \frac{TP}{TP + FN}.$$

F1 score: prcision と recall の調和平均である. F1 score の定義は以下の通りである.

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}.$$

G-mean: precision と recall の幾何学的平均である. G-mean の定義は以下の通りである.

$$G - mean = \sqrt{Precision \times Recall}.$$

5.2.3 実験工程

最適特徴量選択とリアルタイム学習の実験の工程及び設定について説明する.

5.2.3.1 最適特徴量選択

最適特徴量選択の実験工程は 3 つある. まず, permutation importance に基づいて重要度が高い順に特徴量をランキング付けする. このとき, 著者らの先行研究 [15] で作成されたデータセットのうち, 文献で示されている同一の条件, すなわち良性ドメイン上位 24,126 個, 悪性ドメイン 24,126 個をそれぞれ MADMAX のデータセットとして用いる.

次に, 閾値 T の変化, すなわち特徴量の上位から用いる数を変化させたときと, 中間ノード数 N の変化, それぞれに対し, 検知精度の変化を測定する. このとき, 統計的に安定した評価が行えるよう, 5-fold 交差検証を用いて, 検知精度を計測する.

最後に, 最適な N と T に基づき, MADMAX のスループットを計測する. このとき, 良性 100 件, 悪性 100 件をそれぞれデータセットから抽出し, アドオンを通じて, ドメインをサーバに送ってから結果を受け取るまでのそれぞれの平均時間を測定する. これにより, ユーザ目線の MADMAX のスループットの評価が可能となる.

5.2.3.2 リアルタイム学習

normal model の検知精度と更新済みデータセットを用いてリアルタイム学習アルゴリズムによって学習される retrained model の検知精度を比較する. 本実験ではまず, 時間経過とともに悪性ドメインが更新され続け, 良性ドメインは更新されないデータセットを作成する. 実験の初期過程として, まず著者らの先行研究 [15] においても利用した公開データベースから時系列データを収集する. 具体的には, Tranco [39] から 2020 年 11 月 25 日時点での上位 25,000 個のドメインを良性ドメインとして収集し, それぞれのドメインにおいて 4.2.1 節で示した 25 個の特徴量を抽出して良性ドメインデータセットを作成した. 次に, 良性データセットをシャッフルし, 20,000 個のドメインを訓練用データ, 残りの 5,000 個のドメインをテストデータとしてそれぞれ利用する. 同様に, 3 つの悪性ドメイン公開データベース URLhaus, CyberCrime Tracker, そして PhishTank における 2021 年 1 月 29 日から同年 11 月 25 日までの期間の合計 35,000 個の悪性ドメインを出現時刻の情報を含めて収集した. 次に, これら 3 つのデータベースから収集したドメインを併合して時系列順に並び替えた. 最後に, 良性ドメインと同様に 25 個の特徴量を抽出して悪性ドメイン時系列データセットを生成した. 以降では簡単のために, 35,000 個の悪性ドメインを $d_1 - d_{35,000}$ と表記する. 上に示した時系列データを用いて retrained model と normal model の F1 score, G-mean, そして accuracy がそれぞれどれだけ異なるかを評価する.

より詳細には, normal model と retrained model はどちらも 20,000 個の良性ドメインと 20,000 個の悪性ドメイン (d_1-$d_{20,000}$, 具体的には 2021 年 1 月 29 日 22:48:07 から同年 8 月 25 日 15:10:52 までの期間に出現したドメインとなる) を訓練用データとして訓練される. そして, retrained model のみ 2021 年 8 月 25 日 15:15:05 から同年 10 月 28 日 12:01:15 までの期間にかけて観測された悪性ドメインについて時系列を追ってリアルタイム学習を行った. ここで, ある時点の時刻に該当する最新の悪性ドメインを d_i とする. 該当する悪性ドメインが複数個存在する時刻については, 最後の行に存在する悪性ドメインを d_i とする. normal model は実験が終わるまで変更が加えられることはなく, retrained model は 30 秒ごとに最新の 20,000 個の悪性ド

[*11] https://scikit-learn.org/stable/modules/generated/sklearn.inspection.permutation_importance.html

メイン ($d_{i-20,000}$-d_i) にスライドして繰り返し再学習する. そして, 5,000 個の良性ドメインと再学習した直後に出現した 5,000 個の悪性ドメイン (d_i-d_{i+5000}) , つまり retrained model にとって未来に出現する悪性ドメインをテストデータとして利用する. これらテストデータを利用して, 未知の悪性ドメインの検知精度評価として 30 秒ごとに 2 つのモデルの F1 score, G-mean, そして Accuracy を評価する. また, MADMAX のスループットの評価としてサーバ上におけるモデルの学習時間を計測する. 具体的には, 10,000 回分の学習時間の平均値をスループットとして計算する.

5.3 実験結果

最適特徴量選択および, リアルタイム学習の実験結果をそれぞれ以下に示す.

5.3.1 最適特徴量選択

まず中間層のノード数 N に関して作成したモデルをそれぞれ用いて, permutation importance を行った. 特徴量の重要度が高い順にソートした結果を表 1 に示す. 各モデルごとに多少の重要度ランキングの変化はあるものの, どのモデルにおいても重要となる特徴量は大きく変わらなかった. 例えば, $N = 100$ の場合を除いたどの中間ノード数に関しても上位 3 つの重要な特徴は共通であった.

続いて, 表 1 に示した特徴量の重要度ランキングに基づき, 上位 T 個の特徴量を用いた際の F1 score, G-mean, Accuracy, Precision, Recall の値を図 4 示す. まず, 全体的に Precision より Recall の値が大きい. すなわち, 悪性ドメインの見逃し率が少なく, 誤検知が多いことを意味している. 加えて, F1 score と G-mean の実験結果から, 数値的に安定していることを確認した. また, F1 score の最大値であるノード数 $N = 600$ のモデルを用いた際のランキング付けされた特徴量上位 10 個で, F1 score が 88.5%であること確認した. 加えて, F1 score, G-mean, Accuracy, Precision, Recall ともに, 特徴量 25 個全て用いた場合よりも重要度の高い特徴量を用いることで精度が改善される傾向が見られた. Cao ら [10] によると, ELM における中間層のノード数が多いと最善の精度を維持できない. すなわち, 本稿で示した図 4 の結果は Cao らによる発見と一致する.

最後に, 最善の F1 score を達成したノード数 $N = 600$ のモデルを用いた際の, 良性ドメインおよび悪性ドメインのそれぞれの閾値 T に対する MADMAX のスループットを図 5 に示す. 全体的に閾値 T が増える, すなわち用いる特徴量の数が増えるとスループットが低下する傾向が見られた. 例えば, $T = 6$ で, 悪性ドメイン約 1.5 秒, 良性ドメインでは約 1.0 秒であった. また, $T = 10$ の時で, 悪性ドメインでは約 4.6 秒, 良性ドメインでは約 3.3 秒, $T = 25$ のときで, 悪性ドメイン約 5.2 秒, 良性ドメイン約 5.0 秒であった.

結果として, F1 score に関して, 最適特徴量選択では $N = 600$ と $T = 10$ を選んだ. 一方で, ブラウザベースアプリケーションとして, MADMAX のスループットも考慮されるべきである. このとき, $N = 600$ で $5 \leq T \leq 10$ の範囲で, F1 score が高いと考えられる. これらの T に関して, 図 5 によると, $T = 5$ のとき, もっともスループットが高い. ゆえに, $N = 600$, $5 \leq T \leq 10$ に関して, リアルタイム学習の性能を評価した結果を次節で示す.

5.3.2 リアルタイム学習

normal model と retrained model の 2 つのモデルの F1 score, G-mean, そして Accuracy の結果をそれぞれ図 6, 7, 8 に示す. 青色と橙色の曲線はそれぞれ normal model と retrained model の各評価指標を示している. retrained model においてはリアルタイム学習の実験中に計 6,136 回の再学習が行われた. 最適特徴量選択の実験結果でも述べた様に, 実験では ELM の隠れ層のノード数 N は 600, 最適特徴量の個数 T として $5 \leq T \leq 10$ としている. これらの図より, 各モデルの F1 score, G-mean, そして Accuracy は全体的に時間経過とともに増加している. また, F1 score と G-mean の結果はほとんど一致している.

その一方で, 10 月上旬から normal model は retrained model よりも F1 score, G-mean, Accuracy が低くなっている. つまり, retrained model は normal model よりも高い検知精度を獲得していることが分かる.

また, 図 9 に MADMAX における ELM モデルの学習時間 (training time) を示す. 各閾値 T (the threshold) においてサーバ上での ELM モデルの学習時間を計測した. ここでは, ELM モデルのノード数 N は 600 としている. これより, 学習時間が全ての閾値 T において約 1.6 秒で安定していることが分かる.

6. 考察

本章では, 各実験の結果について考察し, MADMAX の性能を ELM ベースの悪性ドメイン検知に関する既存手法と比較する. 次に, より現実な世界への応用として不均衡なデータセットが利用された際の結果について考査する. 最後に MADMAX における制約事項について説明する.

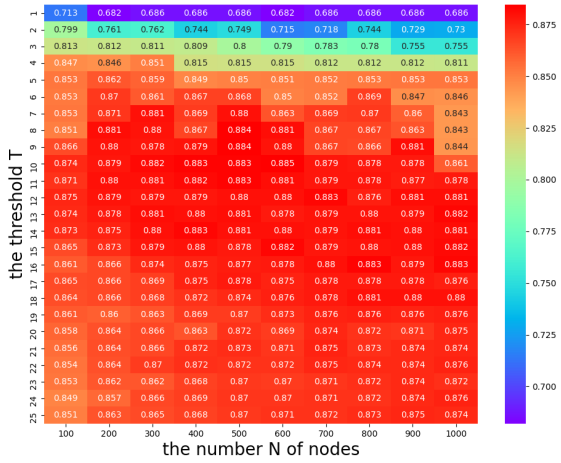
6.1 最適特徴量選択

6.1.1 特徴量の重要度

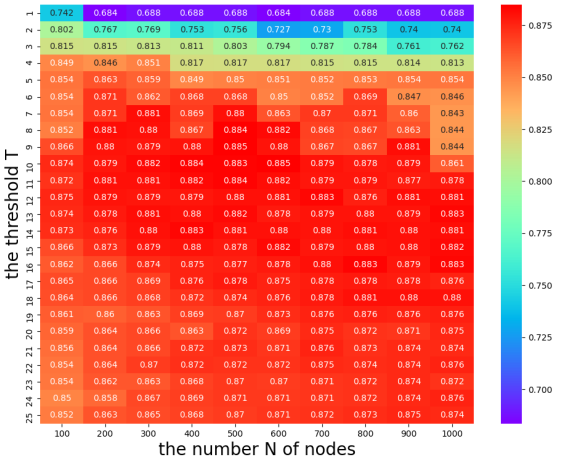
最初に, 特徴量の重要度について議論する. 前節における permutation importance を用いた特徴量の重要度順の結果と, 著者らの先行研究 [15] に示されている LGBM を用いた場合の重要度計算による結果が異なっていた. 具体的には, 著者らの先行研究 [15] による上位 13 個の特徴量と, permutation importance を用いた上位 13 個の特徴量を比較して, 共に重要であった特徴量は, ドメインの長さ, NS 数, ライフタイム, HTML のタグ数, TTL 平均, MX 数, エントロピーの計 7 個である. この違いについて, いくつかの考察すべき点を以下に述べる.

表 1: 特徴量の重要度ランキング

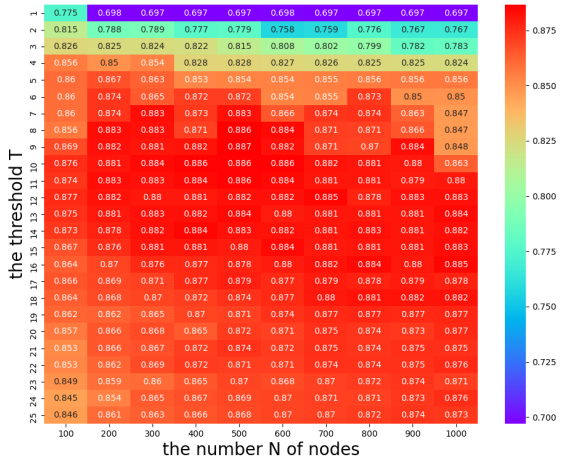
	100	200	300	400	500	600	700	800	900	1000
1	NS 数	ドメインの長さ	ドメインの長さ	ドメインの長さ	ドメインの長さ	ドメインの長さ	ドメインの長さ	ドメインの長さ	ドメインの長さ	ドメインの長さ
2	ドメインの長さ	NS 数	NS 数	NS 数	NS 数	NS 数	NS 数	NS 数	NS 数	NS 数
3	数字数	子音数	子音数	子音数	子音数	子音数	子音数	子音数	子音数	子音数
4	TTL 平均	ライフタイム	ライフタイム	母音数	母音数	母音数	母音数	母音数	母音数	母音数
5	MX 数	TTL 平均	HTML タグ数	ライフタイム	ライフタイム	ライフタイム	ライフタイム	ライフタイム	ライフタイム	ライフタイム
6	母音数	MX 数	母音数	TTL 平均	TTL 平均	数字割合	数字割合	TTL 平均	子音種類	子音種類
7	数字割合	数字割合	TTL 平均	数字割合	HTML タグ数	HTML タグ数	TTL 平均	数字割合	HTML タグ数	エントロピー
8	母音割合	HTML タグ数	数字割合	数字数	MX 数	TTL 平均	母音割合	子音種類	数字割合	母音割合
9	HTML タグ数	数字数	数字数	HTML タグ数	数字割合	子音種類	子音種類	母音割合	TTL 平均	数字割合
10	ライフタイム	母音割合	MX 数	MX 数	数字数	MX 数	HTML タグ数	HTML タグ数	エントロピー	HTML タグ数
11	子音数	TTL 標準偏差	子音種類	母音割合	アクティブタイム	母音割合	数字数	数字数	母音割合	TTL 平均
12	TTL 標準偏差	母音数	母音割合	子音種類	母音割合	数字数	MX 数	エントロピー	MX 数	MX 数
13	子音種類	IP 数	TTL 標準偏差	IP 数	TTL 標準偏差	エントロピー	エントロピー	MX 数	数字数	評判値
14	記号等の文字数	母音子音の切替	IP 数	TTL 標準偏差	子音種類	TTL 標準偏差	IP 数	IP 数	評判値	数字数
15	アクティブタイム	子音種類	エントロピー	エントロピー	エントロピー	IP 数	アクティブタイム	評判値	IP 数	IP 数
16	IP 数	アクティブタイム	NS 間の文字列類似度	アクティブタイム	IP 数	アクティブタイム	TTL 標準偏差	TTL 標準偏差	アクティブタイム	NS 間の文字列類似度
17	数字アルファベット切替	数字アルファベット切替	アクティブタイム	評判値	評判値	母音種類	NS 間の文字列類似度	母音種類	NS 間の文字列類似度	アクティブタイム
18	エントロピー	エントロピー	評判値	母音種類	母音種類	評判値	評判値	NS 間の文字列類似度	TTL 標準偏差	TTL 標準偏差
19	母音子音の切替	NS 間の文字列類似度	母音子音の切替	母音子音の切替	NS 間の文字列類似度	NS 間の文字列類似度	母音種類	アクティブタイム	母音種類	母音種類
20	母音種類	記号等の文字数	数字アルファベット切替	NS 間の文字列類似度	数字アルファベット切替	母音子音の切替	数字アルファベット切替	母音子音の切替	母音子音の切替	母音子音の切替
21	評判値	母音種類	母音種類	記号等の文字数	母音子音の切替	数字アルファベット切替	記号等の文字数	数字アルファベット切替	数字アルファベット切替	数字アルファベット切替
22	NS 間の文字列類似度	評判値	記号等の文字数	数字アルファベット切替	記号等の文字数	記号等の文字数	母音子音の切替	記号等の文字数	記号等の文字数	記号等の文字数
23	最大繰返し数	最大繰返し数	最大繰返し数	最大繰返し数	最大繰返し数	最大繰返し数	最大繰返し数	最大繰返し数	最大繰返し数	最大繰返し数
24	IP 所属国の数	PTR 数	PTR 数	PTR 数	PTR 数	PTR 数	PTR 数	PTR 数	PTR 数	PTR 数
25	PTR 数	IP 所属国の数	IP 所属国の数	IP 所属国の数	IP 所属国の数	IP 所属国の数	IP 所属国の数	IP 所属国の数	IP 所属国の数	IP 所属国の数



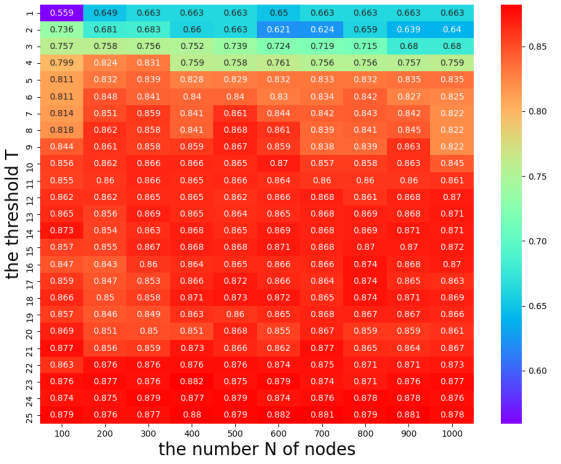
(a) F1 score



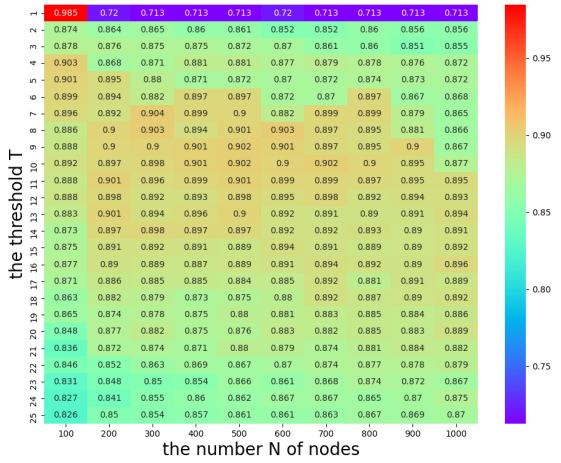
(b) G-mean



(c) Accuracy



(d) Recall



(e) Precision

図 4: 閾値 T(the threshold) と中間層のノード数 N(the number of nodes) それぞれに関して, 最適特徴量選択による F1 score, G-mean, Accuracy, Recall, Precision.

先行研究 [15] の結果を踏まえると, 文字列ベース特徴量においてドメインの長さと, エントロピーは重要である. なぜなら DGA で生成される悪性ドメインの名前の特徴を示しているからである. もう少し具体的には, 良性ドメインの名前は 1 単語か 2 単語程度の意味のある単語と SLD, TLD の組み合わせで構成される. 一方, DGA が生成するドメイ

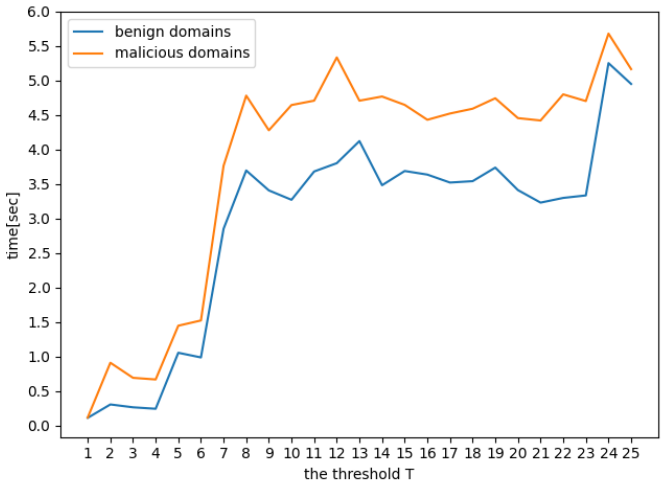


図 5: 各閾値 T(the threshold T) に対した検知時間

ンは, ドメイン名の衝突を防ぐため, より長く, よりランダムな文字を利用する傾向がある [49]. この原因はドメイン名が衝突すると, 登録されないためである. このため, 良性ドメインと比較して, 悪性ドメインはより長く, より高いエントロピーがある.

一方で, DNS ベース特徴量である NS 数と MX 数は, 悪性ドメインの方が良性ドメインよりも少ない. この理由として悪性ドメインは良性ドメインと比べて必要な機能が少ないことが考えられる. 詳細は倫理的配慮から伏せるが, 良性の上位は多国籍企業が多いことも理由の 1 つといえる.

さらに, ウェブベース特徴量について, 悪性ドメインの HTML タグ数が良性ドメインに比べて少ない. 実際に, 今回用いたデータセット, 良性 24,126 件, 悪性 24,126 件で調査したところ, 悪性ドメイン平均約 136 個, 良性ドメイン平均約 724 個であった. この違いはコードを配布するような悪性サイトはウェブページの構成を考慮しないからである.

6.1.2 検知精度

次に, 図 4 では重要度でランキング付けされた特徴量を何個か抽出することで, 全ての特徴量を用いる場合より, 高い検知精度がでている. これは不要な特徴量が, 本来の悪性ドメインの傾向を攪乱することを意味している. そのため, MADMAX は特徴量の選定を行うことで, 検知精度の改善に成功している.

6.1.3 スループット

図 5 の結果より, 悪性ドメインが良性ドメインよりもスループットが全体的に高い. これは, 悪性ドメインは登録されていない新規のドメインが用いられていることが多く, DNS レコードそのものを新たに取得しているためである. すなわち, DNS ベース特徴量の抽出に時間が要していると考えられる. 実際に, 図 5 のネームサーバの数を取得している $T = 1$ から $T = 2$ と, メールサーバの数を取得している $T = 9$ から $T = 10$ かけてのスループットの上昇傾向は, 悪性ドメインが良性ドメインよりも大きい.

また, 閾値 T が大きくなるにつれ, スループットが低下する傾向にある. 例えば, $T = 6$ から $T = 7$ にかけて悪性ドメイン, 良性ドメインともにスループットは大きく低下する. この理由は悪性ドメインと良性ドメインで若干異なる. 良性ドメインでは HTML のタグ数が多いため, 取得する時間を要している. 一方, 悪性ドメインは, 前述した通り, いくつかのドメインは HTML コンテンツがなく, HTML タグが得られないためである.

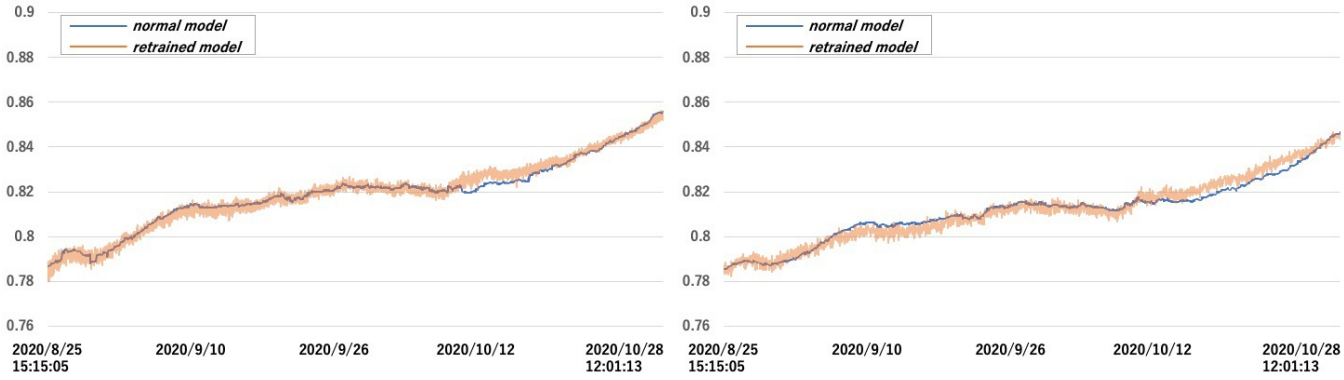
6.1.4 トレードオフ

最後に, MADMAX の検知精度とスループットとのトレードオフを考察する. 図 4 によると, ある程度の検知精度かつ速度を重視するなら, $N = 600$, $T = 6$ の特徴量集合, すなわちドメインの長さ, NS 数, 子音数, 母音数, ライフタイム, 数字割合の特徴量が最適である.

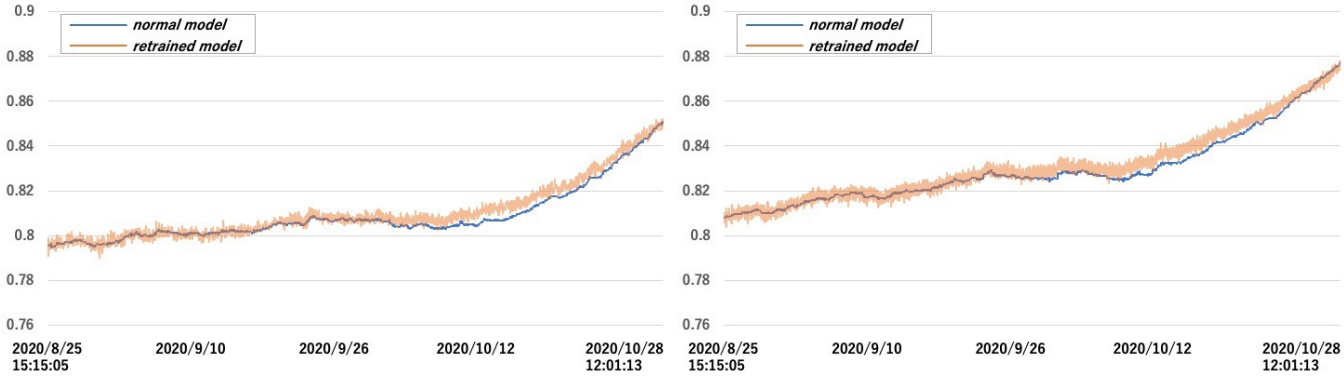
6.2 リアルタイム学習

未知の悪性ドメインを検知するという観点におけるリアルタイム学習の影響について考察する. 5.3.2 節で示した実験結果によると, 全ての実験パターンにおいて 10 月上旬以降に normal model の検知精度が低下しているので, この期間にいくつかのコンセプト・ドリフトが発生したと考えられる. 対照的に, retrained model は新しく出現した悪性ドメインの特徴を学習することができている.

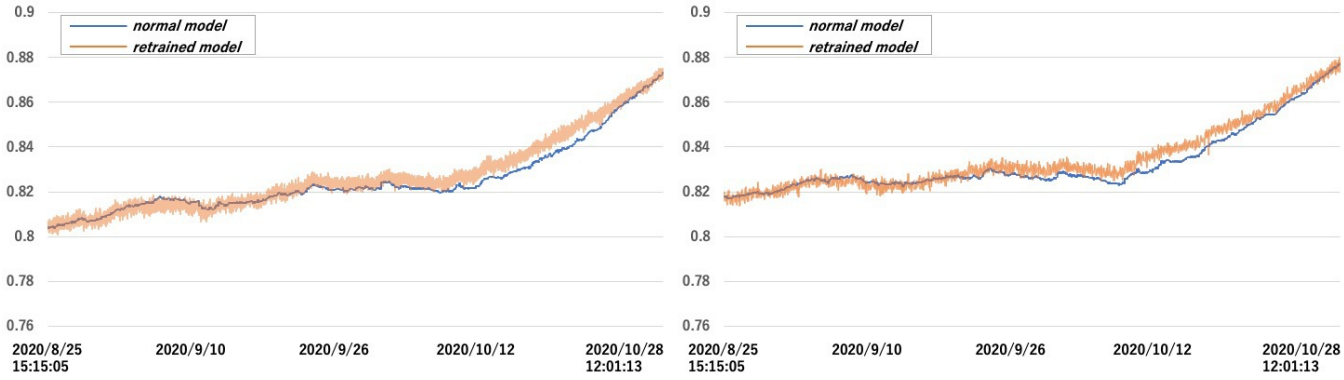
10 月上旬以降に normal model が retrained model よりも検知性能が低下した原因へのさらなる考察のために, 同実験において別の評価指標として precision と recall を計測し, 結果を図 10, 11 に示す. 図 6, 7, 8 と同様に青色と橙色の曲線はそれぞれ normal model と retrained model の各評価指標を示している. これらより, normal model は retrained model よりも Precision が高いが, Recall が低いことが分かる. この事実は normal model が retrained model よりも悪性ドメインについて FP が少ないが, FN が多いことを意味する. 言い換えると, normal model は悪性ドメインの誤検知は少ないものの未知の悪性ドメインの見逃しが多く, retrained model はそれらも検知できているといえる. retrained model の検知精度が優れていることへのより直感的な理解のために, 各閾値 T において normal model と retrained model の F1 score の値の差が最大となった時刻の 2 つのモデルの各評価指標を表 2-7 に示す. 任意の T においてもそれらの時刻はコンセプト・ドリフトが発生したと考えられる 10 月上旬以降の期間内に存在していた. F1 score と G-mean では retrained model の方が normal model よりも最大で 0.011 高くなっている. そのとき, 悪性ドメインの見逃し率として, Recall について retrained model の方が normal model よりも最大で 0.025 高くなっている. 実験ではテストデータとして 2,500 個の悪性ドメインが利用されたため, normal model と retrained model の recall の値の差は retrained model



(a) $T = 5$ (b) $T = 6$



(c) $T = 7$ (d) $T = 8$



(e) $T = 9$ (f) $T = 10$

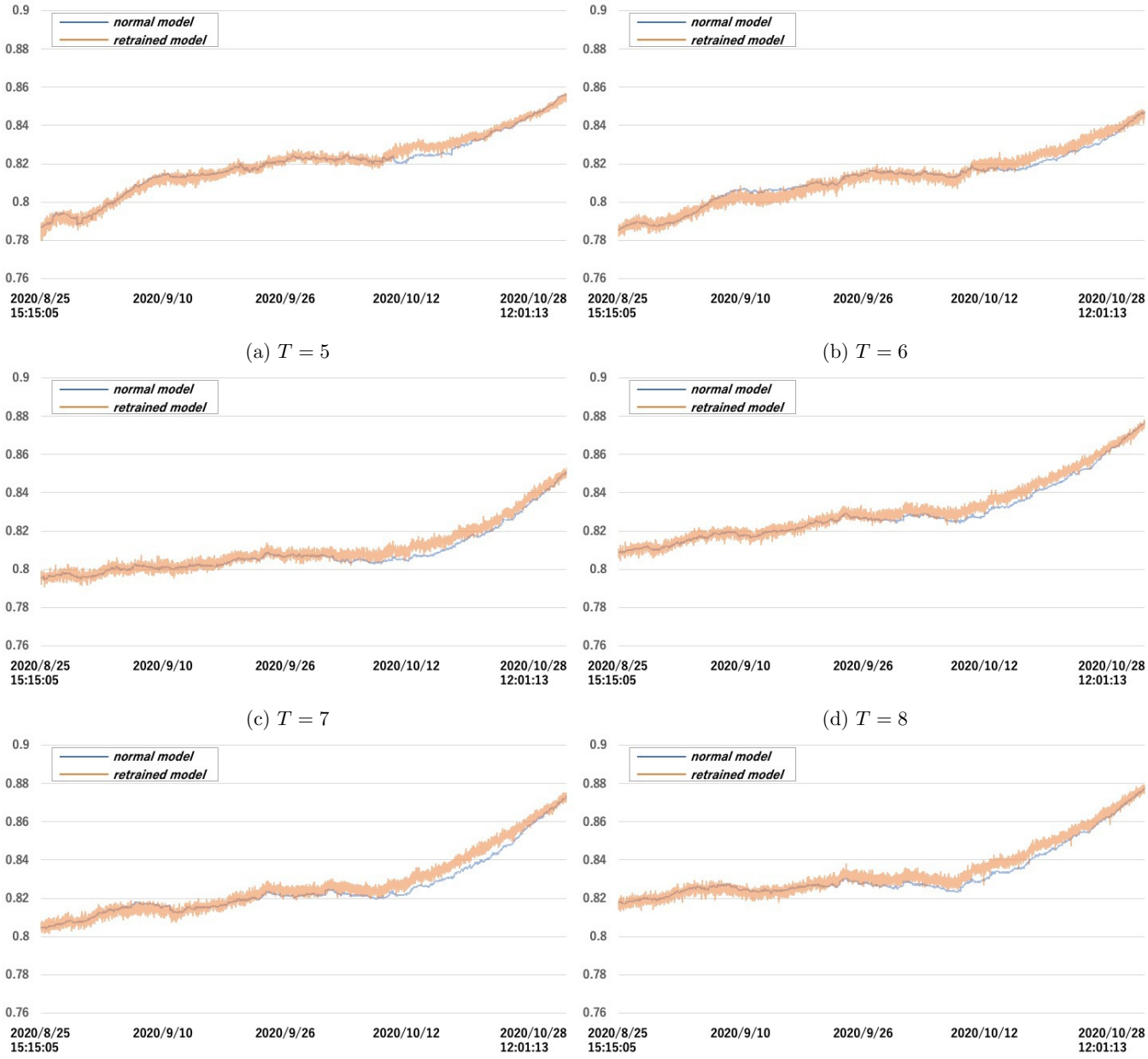
図 6: 各閾値 T における normal model と retrained model の F1 score

表 2: $T = 5$ における retrained model と normal model の各評価指標の差

評価指標	F1 score	G-mean	Accuracy	Precision	Recall
normal model	0.819	0.82	0.811	0.785	0.857
retrained model	0.829	0.831	0.818	0.781	0.884
差	0.00976	0.01057	0.0067	-0.00441	0.0272

表 3: $T = 6$ における retrained model と normal model の各評価指標の差

評価指標	F1 score	G-mean	Accuracy	Precision	Recall
normal model	0.828	0.817	0.82	0.792	0.868
retrained model	0.837	0.826	0.83	0.802	0.875
差	0.00855	0.00878	0.0095	0.01048	0.0062



(a) $T = 5$ (b) $T = 6$

(c) $T = 7$ (d) $T = 8$

(e) $T = 9$ (f) $T = 10$

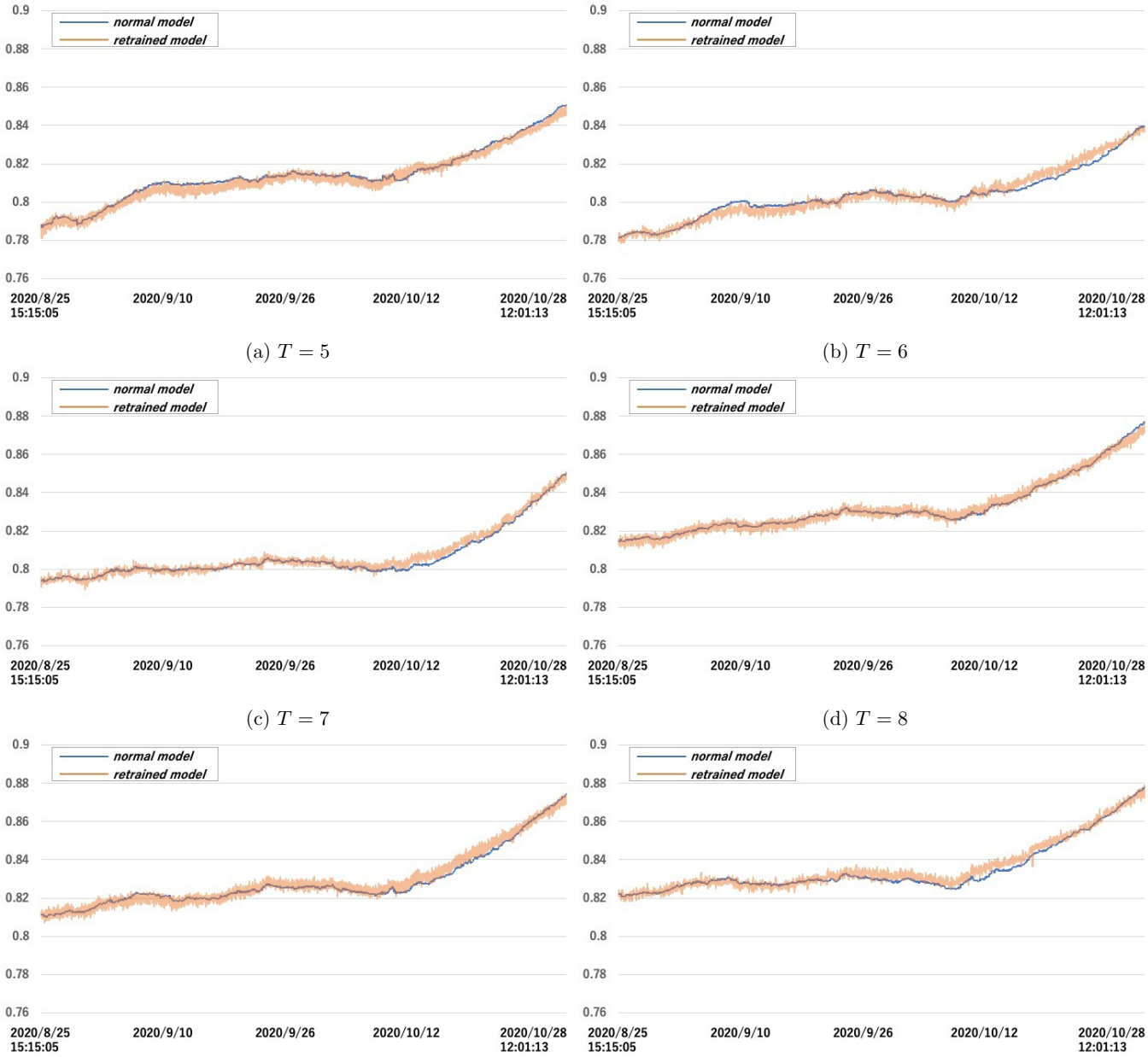
図 7: 各閾値 T における normal model と retrained model の G-mean

表 4: $T = 7$ における retrained model と normal model の各評価指標の差

評価指標	F1 score	G-mean	Accuracy	Precision	Recall
normal model	0.808	0.808	0.804	0.791	0.8256
retrained model	0.818	0.819	0.812	0.790	0.849
差	0.01058	0.0109	0.0082	-0.00016	0.0234

表 5: $T = 8$ における retrained model と normal model の各評価指標の差

評価指標	F1 score	G-mean	Accuracy	Precision	Recall
normal model	0.825	0.825	0.826	0.831	0.818
retrained model	0.834	0.834	0.832	0.823	0.845
差	0.00931	0.00935	0.0057	-0.0082	0.027



(a) $T = 5$ (b) $T = 6$

(c) $T = 7$ (d) $T = 8$

(e) $T = 9$ (f) $T = 10$

図 8: 各閾値 T における normal model と retrained model の Accuracy

表 6: $T = 9$ における retrained model と normal model の各評価指標の差

評価指標	F1 score	G-mean	Accuracy	Precision	Recall
normal model	0.838	0.838	0.841	0.853	0.824
retrained model	0.849	0.849	0.849	0.849	0.85
差	0.01104	0.01092	0.0082	-0.00421	0.0258

表 7: $T = 10$ における retrained model と normal model の各評価指標の差

評価指標	F1 score	G-mean	Accuracy	Precision	Recall
normal model	0.829	0.829	0.83	0.832	0.826
retrained model	0.839	0.839	0.838	0.835	0.844
差	0.01026	0.01027	0.0087	0.00236	0.0182

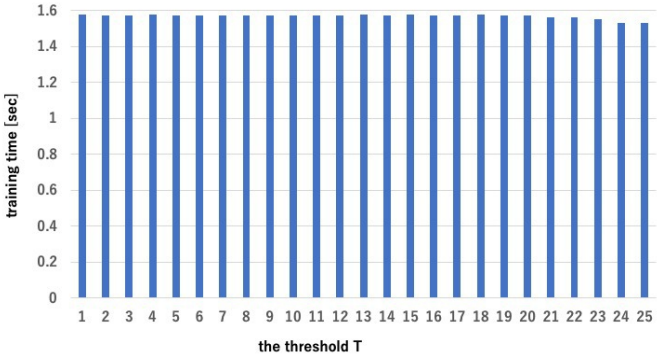


図 9: MADMAX における ELM モデルの学習時間

が normal model が見逃した 625 個の悪性ドメインをも検知可能であることを意味する.

次に, 良性ドメインの更新について考察する. 本稿におけるリアルタイム学習の実験では悪性ドメインのデータのみを 2020 年 8 月 25 日から同年 10 月 28 日までの期間で更新しており, 良性ドメインは Tranco [39] の 2020 年 11 月 25 日の上位データを固定して利用していたため, 24,126 個の良性ドメインの更新が行われていなかった. これが原因で 2 つのモデルの F1 score と Accuracy がどちらも全体的に 11 月 25 日に近づくにつれて増加していたと考えられる. この結果はリアルタイム学習の際に良性ドメインも更新されるべきであることを示唆している.

次に, リアルタイム学習のスループットについて考察する. 実験の中では retrained model は 30 秒ごとに再学習していたが, 実際の使用事例としては モデルは新たな悪性ドメインが悪性ドメインデータベースから取得されたときのみ再学習すれば良いと考えられる. 実際に, MADMAX が利用する 3 つの公開悪性ドメインデータベースでは平均して 12 分に 1 回のペースでドメインが更新される. その際に, 同時に 2 つ, または 3 つといった様に複数個の新しいドメインが出現することが多くあり, 最大で 64 個のドメインが 1 回の更新のみで出現することもあるなど大きな偏りがある. 図 12 に 1 回の更新での新たなドメインの出現個数 (The number of new domains in one update) の頻度 (The number of appearances) を示す. 縦軸は対数目盛をとっており, 横軸が 32, 34, 36, 37, 41, 42, 50, そして 64 のときは 1 つのみのドメインが新たに出現している. また, 横軸が 65 以上のときは 1 回の更新で新たに出現するドメインは存在しなかった.

retrained model を更新するためにデータベースから新たなドメインを取得するのに要する時間, つまり未知のドメインに対して脆弱な時間についてさらに詳細に考察する. 図 9 に示す様に, 600 個のノードを持つ ELM を訓練するための時間は概して 1.6 秒以内である. また, 典型的な誤差逆伝搬によって上述した実験において利用された ELM と同様のアーキテクチャである隠れ層に 600 個のノードを持つニューラルネットワークを訓練するための時間を同様の設定のサーバ環境で測定した. ELM モデルとニューラルネットワークモデルを更新するために要する時間を表 8 に示す. これらの時間にはデータセットのリアルタイムな更新 1 回のために取得したドメインの数だけそれらの特徴量抽出に要する時間とそれらの特徴量を使用した訓練に要する時間を含んでいる. 1 回の更新で新しく取得したドメインの個数として 1, 2, そして, 最大値である 64 の場合をそれぞれ示している. ELM(X) という表記は各 X において 2 回の更新で新しく出現する X 個のドメインに関して ELM を更新するのに要する時間を示している. 同様に, NN(X) という表記はニューラルネットワークを更新するのに要する時間を示している. この表によると, ELM を利用する MADMAX はニューラルネットワークと比較して非常に高いスループットで動作し, 未知のドメインに対して脆弱な時間, つまりユーザを危険に晒す時間を大幅に減らすことが可能である.

6.3 既存研究との比較

Shi ら [48] の特徴量および MADMAX において選別した特徴量を用いて, 各評価指標とスループットを比較した結果を表 9 に示す. また, MADMAX のデータセットを用いて公平な比較を行う.

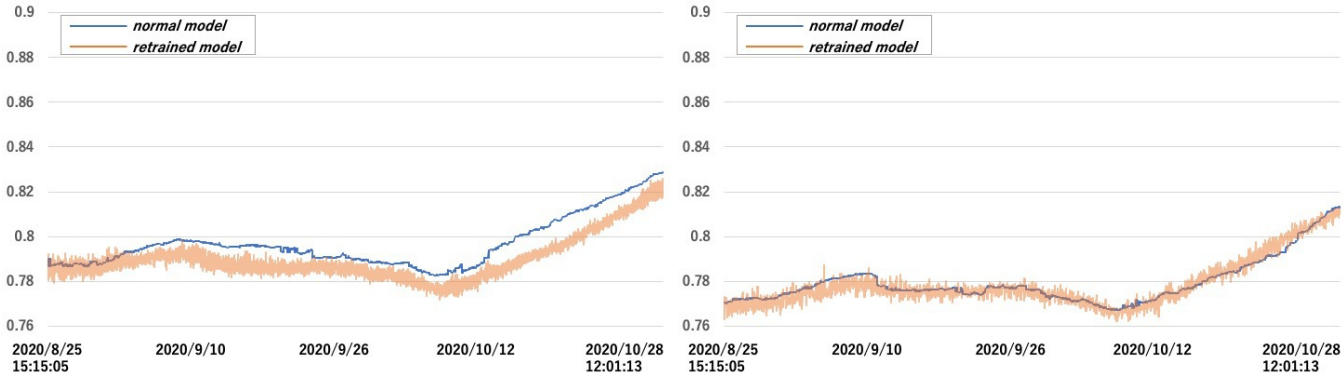
この結果から $N = 600$ のモデルにおいて $T = 6$ の特徴量を用いた MADMAX と, Shi らを比較して同等の精度で悪性ドメインの検知時間は, 0.7 秒高速である. また, $N = 500$ のモデルにおいて $T = 6$ の特徴量を持った MADMAX と, Shi らを比較して同等のスループットで 2.0%ほど上回っている.

これらより, MADMAX では Shi らの手法を単に導入するよりも, 著者らの最適特徴量選択によって, 高い性能を持つと言える.

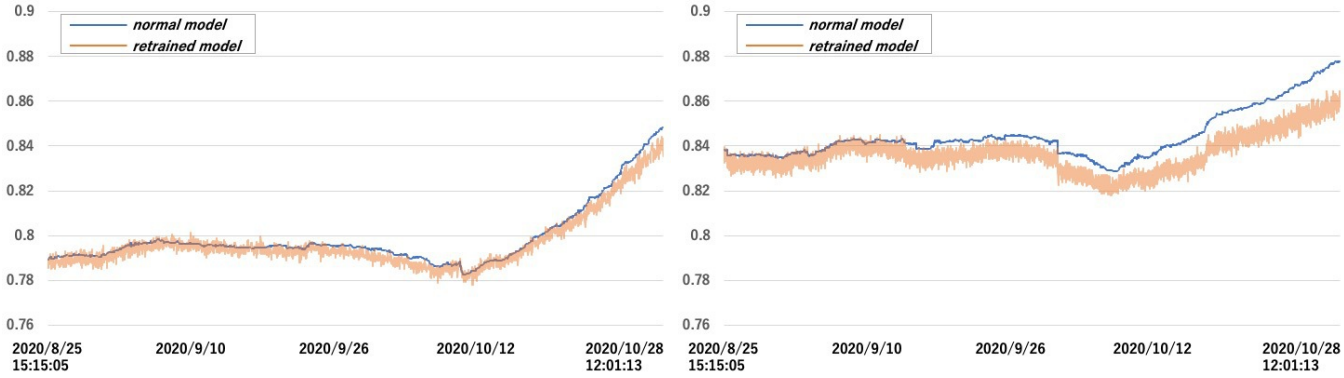
6.4 不均衡なデータセットを用いた評価

また, 均衡なデータセット, および良性と悪性ドメインの数に偏りがある不均衡なデータセットで, それぞれ学習した場合の各ノード N に対して, F1 score の比較を行った結果を表 10 に記載する. ここで, IBD (Imbalanced Benign Dataset) は 19,500 個の良性ドメインと 6,050 個の悪性ドメインの不均衡なデータセットで, IMD(Imbalanced Malicious Dataset) は 6,050 個の良性ドメインと, 19,500 個の悪性ドメインの不均衡なデータセット, BD(Benign Dataset) は良性, 悪性ドメインともに 24,126 個の均衡なデータセットを示している. また, 特徴量は 25 個全て用いている.

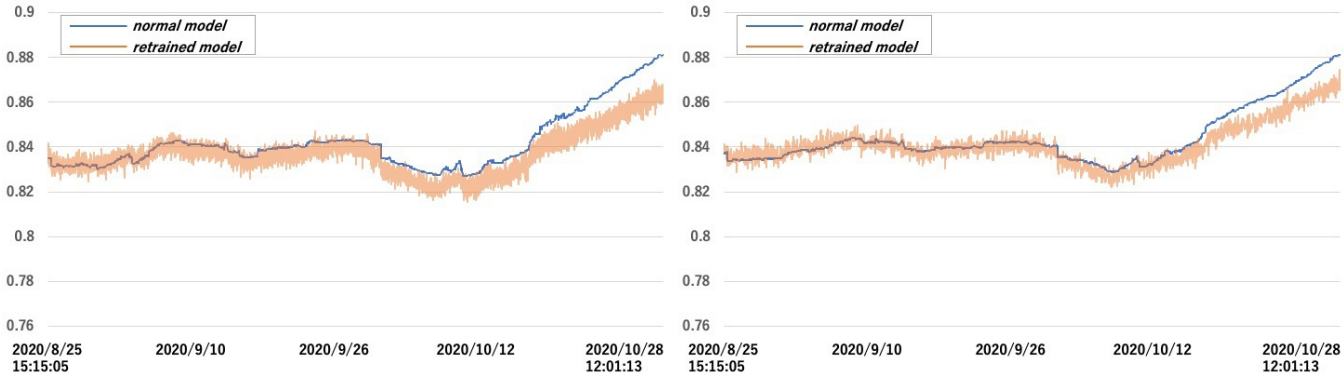
表 10 から, どのノード数においても悪性ドメインの方が良性ドメインより多い場合の方が均衡なデータセットおよび, 良性の方が多い場合より F1 score が高い. これより, 良性ドメインが少ないことにより, 悪性ドメインの傾向を



(a) $T = 5$ (b) $T = 6$



(c) $T = 7$ (d) $T = 8$



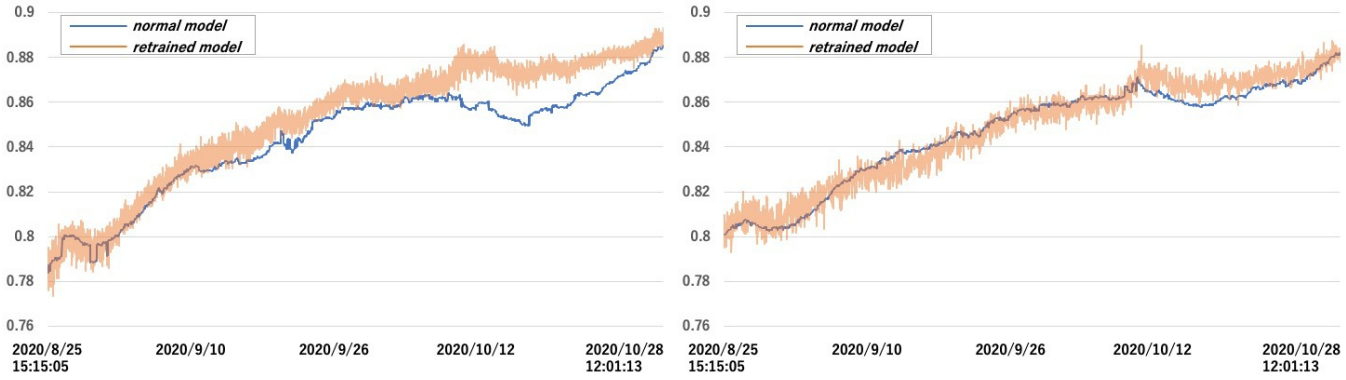
(e) $T = 9$ (f) $T = 10$

図 10: 各閾値 T における normal model と retrained model の Precision

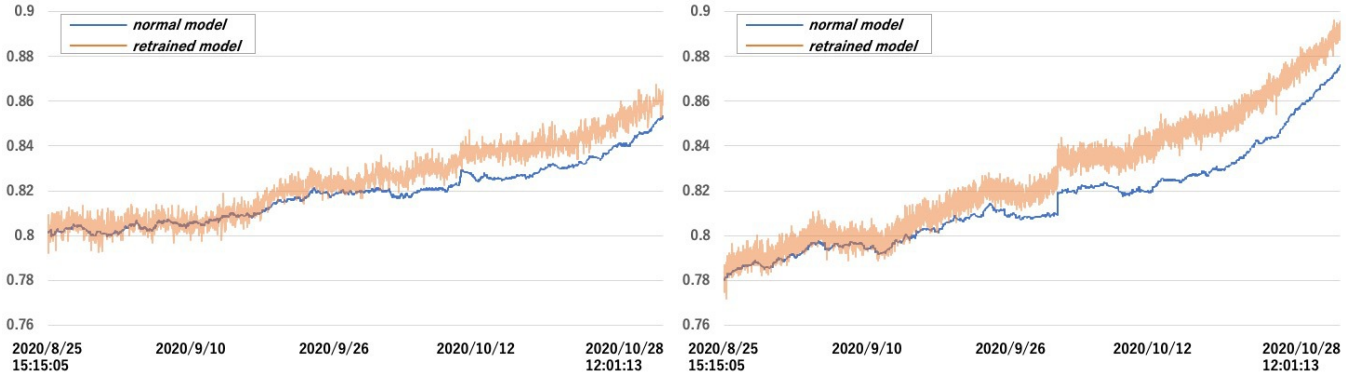
表 8: ELM モデルとニューラルネットワークモデルを更新するために要する時間 [秒]

モデル	$T = 5$	$T = 6$	$T = 7$	$T = 8$	$T = 9$	$T = 10$
ELM (1)	3.02	3.09	5.34	6.35	5.85	6.22
ELM (2)	4.47	4.61	9.11	11.13	10.13	10.87
ELM (3)	5.92	6.13	12.88	15.91	14.41	15.52
ELM (64)	94.37	98.85	242.85	307.49	275.49	299.17
NN (1)	101.45	90.52	90.77	171.78	199.28	283.65
NN (2)	102.9	92.04	94.54	176.56	203.56	288.3
NN (3)	104.35	93.56	98.31	181.34	207.84	292.95
NN (64)	192.8	186.28	328.28	472.92	468.92	576.6

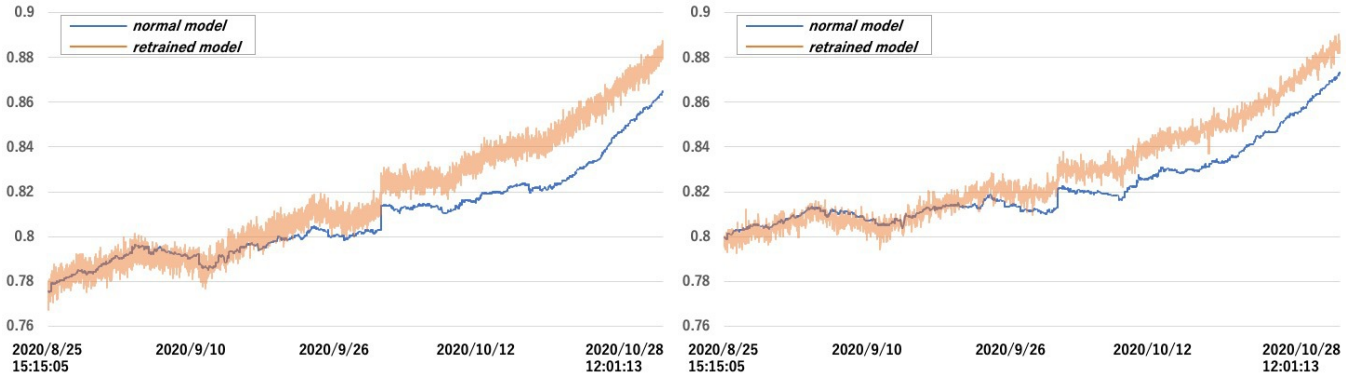
より正確に捉えているといえる．このことから，悪性ドメインの方をより多く集めたデータセットを構築することで，より精度の高い検知が可能となる．



(a) $T = 5$ (b) $T = 6$



(c) $T = 7$ (d) $T = 8$



(e) $T = 9$ (f) $T = 10$

図 11: 各閾値 T における normal model と retrained model の Recall

表 9: 既存研究 [48] との比較

手法	F1 score	Accuracy	Precision	Recall	良性ドメインの推論時間 (秒)	悪性ドメインの推論時間 (秒)
Shi ら [48]($N = 500$)	0.848	0.85	0.855	0.841	0.8	2.4
Shi ら [48]($N = 600$)	0.85	0.851	0.856	0.843	0.9	2.3
MADMAX($T = 6, N = 500$)	0.868	0.872	0.897	0.84	1.1	2.2
MADMAX($T = 10, N = 500$)	0.883	0.886	0.902	0.865	3.5	4.5
MADMAX($T = 6, N = 600$)	0.85	0.854	0.872	0.83	1.0	1.5
MADMAX($T = 10, N = 600$)	0.885	0.885	0.9	0.867	3.3	4.6

6.5 重み行列の初期値

ここでは重み行列の初期値について議論する．本稿では ELM の重み行列 の生成について, 単に一様分布からの生成を仮定しており, とくに構成については議論していなかった．一般に, ELM の重み行列は悪性ドメイン検知の不安定さに影響を与える．

上述した問題は既存技術 [11, 12] を応用することで解決できる．具体的には, ガウス分布で初期化すると一様分布

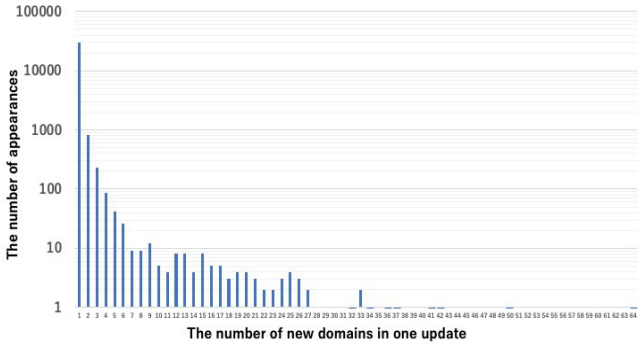


図 12: 1 回の更新での新たなドメインの出現個数の頻度

表 10: 3 つのデータセットを用いた F1 score の比較

N	UBD	UMD	BD
100	0.674	0.928	0.851
200	0.691	0.936	0.863
300	0.709	0.939	0.865
400	0.730	0.940	0.868
500	0.735	0.940	0.870
600	0.742	0.943	0.871
700	0.747	0.943	0.872
800	0.746	0.944	0.873
900	0.752	0.944	0.875
1000	0.753	0.945	0.874

より収束が速くなる. 加えて, 分散 が小さい分布になっている限り, 収束も速くなり, 精度もより良くなる. このため, MADMAX を実際にデプロイする際は, ELM の重み行列をガウス分布で初期化することで, 精度が安定する可能性がある.

6.6 制約

本節では MADMAX の制約事項について述べる.

6.6.1 特徴量の重要度

MADMAX は特徴量の重要度を決定するために permutation importance [9] を用いてる. しかし, permutation importance はアルゴリズムの性質上, 特徴量の値に偏りがあると特徴量の重要度が小さくなるという制限がある. 例えば, 著者らの先行研究 [15] によると, IP 所属国の数は重要な特徴量の 1 つである. それにもかかわらず, MADMAX では permutation importance に基づくと, IP 所属国の数の重要性が低いという結果を示した. この理由の 1 つに, ほとんどのドメインでは IP 所属国の数の値が 0 であるため, ランダムに値を入れ替えても, ほとんどの値が変化しない可能性がある. そのため, permutation importance に基づく重要度では低く算出されたと考えられる.

6.6.2 サーバの保守管理

MADMAX はクライアントサーバ型アプリケーションである. そのため, MADMAX を利用するためには悪性ドメイン検知を行うサーバを用意し, デプロイを事前に行う必要がある. さらに, アドオンを通じてサーバに対し, ドメインを送信するため, サーバ側の機能を継続して管理する必要がある. もし, ユーザ単体で利用したい場合はローカルホスト環境を用いて, サーバを立ち上げる必要がある.

6.6.3 欠損値

前節で述べた実験結果では, リアルタイム学習のためのデータセットを構築する際に, 値が取れない特徴量が複数存在する. 例えば, whois の取得に失敗する WHOIS ライフタイムや, ページがないことによる HTML コンテンツ, 証明書の言語に対応できない場合などがある. このような特徴量を便宜上, 欠損値と呼ぶ.

今回, 本実験において値が取れない場合は, 特徴量の値を 0 で置き換えた. 本稿では MADMAX における欠損値の影響について厳密には議論できてないが, 欠損値が増加するにつれて, 欠損値そのものの傾向を学習する可能性が高い. すなわち, MADMAX による悪性ドメイン検知は欠損値による予期せぬ影響を受けているかもしれない. 欠損値に対するもう 1 つの対処法としては, 欠損値が見つかった際に, そのようなデータを取り除くか, 0 以上の定数で置き換えるかである. この欠損値の対処法については今後の課題である.

6.6.4 非線形データによる検知精度の不安定さ

古典的な ELM の検知精度は, 非線形データに対して不安定になる傾向がある. 本実験で利用したデータは線形性を持っているため, MADMAX の検知精度は安定傾向にある. しかし, 様々なドメインを扱う際に, 非線形データが現れてくる可能性がある. それらの非線形データに対応するため, RC-ELM [62] や R-ELM [61] が利用できる可能性がある. これらも MADMAX の検知精度改善に向けて, 今後の課題である.

7. 関連研究

本節では, 悪性ドメイン検知の関連研究について述べる. 悪性ドメイン検知には, ドメイン名の文字列に着目するドメインベースの手法と, 文字列に加えて他の情報に着目する挙動ベースの手法の 2 種類がある. ドメインベースの手法では, ドメイン名のみを文字列として利用する. そのため, 複雑なディープニューラルネットワークを用いることで精度を向上させることができる. 一方, 挙動ベースの手法は入力データの種類を増やすことで精度を向上させることができる. この 2 つの手法について下記で詳細に説明する.

7.1 ドメインベースの手法

ドメインベースの手法において主要な方法は, 複雑で大規模なモデルをドメイン名に対してのみ利用することで, 悪性ドメインを検知する方法である. Woodbridge ら [55] は, 悪性リストを用いることの限界を考慮して, 悪性ドメイ

ンの検知に再帰型ニューラルネットワーク (RNN) の 1 つのアーキテクチャである長・短期記憶 (LSTM) を用いていた. 次に, Bin ら [58] は, 畳み込みニューラルネットワーク (CNN) と LSTM の両方を活用して, 実際の通信データを活用していた. また, Bin ら [59] は次の 5 つのモデルの精度を比較していた.

- Woodbridge ら [55] の単層 LSTM モデル
- Dhingra ら [20] の LSTM の順伝播層と逆伝播層の複合モデル
- Saxe ら [46] の並列 CNN モデル
- Zhang ら [63] の積層 CNN モデル
- Vosoughi ら [54] の積層 CNN と単層 LSTM の複合モデル

他には, Berman ら [5] は CapsNet をベースにした最初の研究を発表し, その結果, 従来の RNN や CNN よりも優れた性能を発揮することがわかった. Yanchen ら [41] は, ある条件下ではドメイン名の一部を無視するモデルを紹介している. さらに, Luhui ら [57] は並列 CNN と複数層の LSTM を組み合わせた深層学習の手法を提案している.

これらの研究は複雑なアーキテクチャを用いているため, 軽い処理が求められるブラウザ環境での利用には適していない. これらに対し, 本稿の MADMAX はブラウザへの導入を目的としているため, 単純なアーキテクチャである ELM を利用している.

7.2 挙動ベースの手法

挙動ベースの悪性ドメイン検知手法では, 主に次の 4 種類のデータが用いられる [60].

- DNS 情報
- 証明書情報
- ウェブページの構造情報
- ドメインから付随して得られる情報

本小節ではそれぞれのデータについて説明する.

7.2.1 DNS 情報

まず DNS 情報による悪性ドメイン検知について述べる. 文献 [17, 18] でボットネットによって管理されているホストは, クエリ内容と時系列パターンが類似していることが多いと示されている. そのため, 悪性ドメインでは, クライアントと DNS サーバ間の相互通信の挙動に特徴があることが知られている [16, 17, 32, 36, 40, 42, 43]. これらの情報は行列やグラフなど, データとして扱える形に変換して利用する研究も存在する. 例えば, Grill ら [22] は DGA に対して, 知識ベースの悪性ドメイン検知手法を提案している. 具体的には, 感染しているホストは通信している IP アドレス数と DNS クエリ数のバランスが正規のものと区別できることから, 統計的に検知を試みている.

また, Chiba ら [14] は, ドメインの登録日や, 良性リストと悪性リストの登録理由などの情報をもとに, ランダムフォレストで予測モデルを作成している. 他にも, DNS の通信情報を利用した機械学習モデルの研究が盛んに行われている [27, 33, 51, 52].

上述した既存研究を踏まえて, 著者らが知る限り, ブラウザアドオンとして機械学習をアプリケーションに組み込んだ研究は MADMAX が初めてである.

7.2.2 証明書情報

次に, 証明書情報を用いた悪性ドメイン検知を行う研究を紹介する [23, 53]. 文献 [53] ではデータとして, 発行者情報, 自己署名証明書か否か, 悪性サイトで用いられる証明書はいくつかの項目が欠落していることが多いことから証明書の各項目が記入されているかなどを用いることで, 高精度での検知に成功している. また, 文献 [4] では, 証明書情報を含む TLS 通信や HTTP 通信から特徴を抽出して, ロジスティック回帰に基づく分類モデルを構築している. 他には, 文献 [23, 53] では証明書情報みに着目し, DNN や SVM などを用いてマルウェア, フィッシングサイトの検知を行っている. しかし, 証明書による検知は HTTPS に対応しているサイトでない悪性サイトは検知できないため, 網羅的な検知ができないという問題を孕んでいる.

7.2.3 ウェブページの構造情報

DNS 情報や証明書情報に加えて, ウェブページの構造もフィッシングサイトなどの悪性ドメインの検知に利用できる. 文献 [24] はページのテキスト, フォント, 色などの応報から良性のサイトとの比較を行なったり, 文献 [34] では Cascading Style Sheets (CSS) の比較から悪性ドメインを検出しようと試みている. しかし, これらの手法では, コードの難読化技術に対して無力であることが指摘されている [21, 29]. そして, 難読化に有効な手法として, ページの画像情報から検知を試みる研究が行われている [2, 8, 13, 31, 44]. これらの研究は画像処理であることから, CNN で学習させたほうが精度が高くなると言われている [28, 45]. それゆえ, 文献 [1] では, 良性のサイトと悪性サイトの画像情報をもとに, CNN でフィッシングサイトを検出するモデルを構築している. しかしながら, CNN は一般に処理が重く, ブラウザへの導入に不向きである.

7.2.4 ドメインから付随して得られる情報

最後に, 悪性ドメインの検出によく用いられる情報として, 公開された外部情報について説明する. 特に URLhaus[*12] のような悪性リストや, Tranco [39] のような人気

[*12] https://urlhaus.abuse.ch/

サイトランキングのリストがよく利用されている.

さらに MaxMind Database[*13]を用いた IP アドレスから得られる地理情報を利用することもある [6,7,35,37,50]. 地理的な情報の特徴を検知に用いることで, MADMAX の性能をさらなる向上が期待できる.

8. 結論

本稿では extreme learning machine (ELM) [26] を用いたブラウザベース悪性ドメイン検知アプリケーション, MADMAX を提案した. とくに MADMAX の設計にあたり, 最適特徴量選択とリアルタイム学習の機能を, アプリケーションレベルとして著者らが知る限り初めて実装している. 実装した成果物は, GitHub で公開されている.

最適特徴量選択について得られた知見として, 全 25 種類の特徴量を全て用いるよりも, 重要な特徴量を選択することで悪性ドメイン検知の精度が向上することを示した. とくに, DNS レコードを抽出するため, 悪性ドメインと良性ドメインの検知の際に, MADMAX のスループットに差が生じることを確認した.

また, MADMAX は既存の ELM を用いた悪性ドメイン検知 [48] と比較して, 最適特徴量選択により, 精度が優れていることを実証した. また一方で, リアルタイム学習の側面においては, モデルを継続的に再学習させることで, 通常のモデルであればコンセプト・ドリフトにより精度が低下するような未知の悪性ドメインを検知できることを実証した.

さらに本稿では, MADMAX の設計及び実験を通して, 6.6 節で述べたように, permutation importance がうまく動作しない場合など, 悪性ドメイン検知における新たな問題を発見した. permutation importance による問題が悪性ドメイン検知においてどのような影響を与えるか, またその改善案についての調査は現在取り組み中である. また, 今回は古典的な ELM を一定時間ごとに再学習することによってリアルタイム学習を実現しているが, ストリームオンラインから時系列データとして学習をし続けることができる OS-ELM [30] を MADMAX に導入することもできる. これにより, 過去の情報と現在の情報の間の依存関係を捉えた上で再学習を行うリアルタイム学習を実現することが可能である. この拡張については今後の課題とする.

謝辞

本研究の一部は文部科学省 Society 5.0 実現化研究拠点支援事業により支援されている.

参考文献

[1] Abdelnabi, S., Krombholz, K., Fritz, M.: Visualphishnet: Zero-day phishing website detection by visual similarity. In: *Proc. of CCS 2020. pp. 1681–1698. ACM (2020)*

[2] *Afroz, S., Greenstadt, R.: Phishzoo: Detecting phishing websites by looking at them. In: Proc. of ICSC 2011. IEEE (2011)*

[3] *Altmann, A., Toloşi, L., Sander, O., Lengauer, T.: Permutation importance: a corrected feature importance measure. Bioinformatics* ***26****(10), 1340–1347 (2010)*

[4] *Anderson, B., McGrew, D.: Identifying encrypted malware traffic with contextual flow data. In: Proc. of AISec 2016. pp. 35–46. ACM (2016)*

[5] *Berman, D.S.: Dga capsnet: 1d application of capsule networks to dga detection. Information* ***10****(5), 157 (2019)*

[6] *Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: EXPOSURE: Finding malicious domains using passive DNS analysis. In: Proc. of NDSS 2011. The Internet Society (2011)*

[7] *Bilge, L., Sen, S., Balzarotti, D., Kirda, E., Kruegel, C.: Exposure: A passive DNS analysis service to detect and report malicious domains. ACM Transactions on Information and System Security* ***16****(4), 1–28 (2014)*

[8] *Bozkir, A.S., Sezer, E.A.: Use of hog descriptors in phishing detection. In: Proc. of ISDFS 2016. pp. 148–153. IEEE (2016).* DOI: 10.1109/ISDFS.2016.7473534

[9] Breiman, L.: Random forests. Machine learning **45**, 5–32 (2001)

[10] Cao, W., Gao, J., Ming, Z., Cai, S.: Some tricks in parameter selection for extreme learning machine. *IOP Conference Series: Materials Science and Engineering* ***261****, 012002 (2017)*

[11] *Cao, W., Gao, J., Ming, Z., Cai, S., Zheng, H.: Impact of probability distribution selection on rvfl performance. In: Proc. of SmartCom 2018. LNCS, vol. 10699, pp. 114–124. Springer (2018)*

[12] *Cao, W., Patwary, M.J.A., Yang, P., Wang, X., Ming, Z.: An initial study on the relationship between meta features of dataset and the initialization of nnrw. In: Proc. of IJCNN 2019. pp. 1–8. IEEE (2019)*

[13] *Chen, K.T., Chen, J.Y., Huang, C.R., Chen, C.S.: Fighting phishing with discriminative keypoint features. Internet Computing* ***13****(3), 56–63 (2009)*

[14] *Chiba, D., Yagi, T., Akiyama, M., Shibahara, T., Yada, T., Mori, T., Goto, S.: Domainprofiler: Discovering domain names abused in future. In: Proc. of DSN 2016. pp. 491–502. IEEE (2016)*

[15] *Chien, C.J., Yanai, N., Okamura, S.: Design of malicious domain detection dataset for network security (2021),* `http://www-infosec.ist.osaka-u.ac.jp/~yanai/dataset.pdf`

[16] *Choi, H., Lee, H., Lee, H., Kim, H.: Botnet detection by monitoring group activities in dns traffic. In: Proc. of ICCIT 2007. pp. 715–720. IEEE (2007)*

[17] *Choi, H., Lee, H.: Identifying botnets by capturing group activities in DNS traffic. Computer Networks* ***56****(1), 20–33 (2012)*

[18] *Choi, H., Lee, H., Kim, H.: Botgad: detecting botnets by capturing group activities in network traffic. In: Proc. of COMSWARE 2009. pp. 1–8. ACM (2009)*

[19] *Cortes, C., Vapnik, V.: Support vector machine. Machine learning* ***20****(3), 273–297 (1995)*

[20] *Dhingra, B., Zhou, Z., Fitzpatrick, D., Muehl, M., Cohen, W.W.: Tweet2vec: Character-based distributed representations for social media. arXiv preprint arXiv:1605.03481 (2016)*

[21] *Fu, A.Y., Wenyin, L., Deng, X.: Detecting phish-*

[*13] `https://www.maxmind.com/`

ing web pages with visual similarity assessment based on earth mover's distance (EMD). IEEE Transactions on Dependable and Secure Computing **3**(4), 301–311 (2006)

[22] Grill, M., Nikolaev, I., Valeros, V., Rehak, M.: Detecting DGA malware using NetFlow. In: Proc. of IM 2015. pp. 1304–1309. IEEE (2015)

[23] Herrald, D., Kovar, R.: The "hidden empires" of malware. In: SANS Cyber Threat Intelligence Summit 2018 Invited Talk (2018)

[24] Huang, C.Y., Ma, S.P., Yeh, W.L., Lin, C.Y., Liu, C.T.: Mitigate web phishing using site signatures. In: Proc. of TENCON 2010. pp. 803–808. IEEE (2010)

[25] Huang, G., Huang, G.B., Song, S., You, K.: Trends in extreme learning machines: A review. Neural Networks **61**, 32–48 (2015)

[26] Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: Theory and applications. Neurocomputing **70**(1), 489–501 (2006)

[27] Khalil, I., Yu, T., Guan, B.: Discovering malicious domains through passive dns data graph analysis. In: Proc. of ASIACCS 2016. pp. 663–674. ACM (2016)

[28] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proc. of NIPS 2012. vol. 1, pp. 1097–1105. Curran Associates Inc. (2012)

[29] Lam, I.F., Xiao, W.C., Wang, S.C., Chen, K.T.: Counteracting phishing page polymorphism: An image layout analysis approach. In: Proc. of ISA 2009. Lecture Notes in Computer Science, vol. 5576, pp. 270–279. Springer (2009)

[30] Liang, N.Y., Huang, G.B., Saratchandran, P., Sundararajan, N.: A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Transactions on neural networks **17**(6), 1411–1423 (2006)

[31] Malisa, L., Kostiainen, K., Capkun, S.: Detecting mobile application spoofing attacks by leveraging user visual similarity perception. In: Proc. of CODASPY 2017. pp. 289–300. ACM (2017)

[32] Manadhata, P.K., Yadav, S., Rao, P., Horne, W.: Detecting malicious domains via graph inference. In: Proc. of ESORICS 2014. Lecture Notes in Computer Science, vol. 8712, pp. 1–18. Springer (2014)

[33] Manadhata, P.K., Yadav, S., Rao, P., Horne, W.: Detecting malicious domains via graph inference. In: Proc. of ESORICS 2014. pp. 1–18. Springer (2014)

[34] Mao, J., Tian, W., Li, P., Wei, T., Liang, Z.: Phishing-alarm: Robust and efficient phishing detection via page component similarity. IEEE Access **5**, 17020–17030 (2017)

[35] Mishsky, I., Gal-Oz, N., Gudes, E.: A topology based flow model for computing domain reputation. In: Proc. of DBSec 2015. Lecture Notes in Computer Science, vol. 9149, pp. 277–292. Springer (2015)

[36] Oprea, A., Li, Z., Yen, T.F., Chin, S.H., Alrwais, S.: Detection of early-stage enterprise infection by mining large-scale log data. In: Proc. of DSN 2015. pp. 45–56. IEEE (2015)

[37] Passerini, E., Paleari, R., Martignoni, L., Bruschi, D.: Fluxor: Detecting and monitoring fast-flux service networks. In: Proc. of DIMVA 2006. Lecture Notes in Computer Science, vol. 5137, pp. 186–206. Springer (2008)

[38] Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L.: Tesseract: Eliminating experimental bias in malware classification across space and time. In: Proc. of USENIX Security 2019. pp. 729–746. USENIX Association (2019)

[39] Pochat, V.L., van Goethem, T., Tajalizadehkhoob, S., Korczynski, M., Joosen, W.: Tranco: A research-oriented top sites ranking hardened against manipulation. In: Proc. of NDSS 2019. Internet Society (2019)

[40] Prieto, I., Magaña, E., Morató, D., Izal, M.: Botnet detection based on DNS records and active probing. In: Proc. of SECRYPT 2011. pp. 307–316. IEEE (2011)

[41] Qiao, Y., Zhang, B., Zhang, W., Sangaiah, A.K., Wu, H.: Dga domain name classification method based on long short-term memory with attention mechanism. Applied Sciences **9**(20), 4205 (2019)

[42] Rahbarinia, B., Perdisci, R., Antonakakis, M.: Segugio: Efficient behavior-based tracking of malware-control domains in large isp networks. In: Proc. of DSN 2015. pp. 403–414. IEEE (2015)

[43] Rahbarinia, B., Perdisci, R., Antonakakis, M.: Efficient and accurate behavior-based tracking of malware-control domains in large isp networks. ACM Transactions on Privacy and Security **19**(2), 1–31 (2016)

[44] Rao, R.S., Ali, S.T.: A computer vision technique to detect phishing attacks. In: Proc. of CSNT 2015. pp. 596–601. IEEE (2015)

[45] Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: An astounding baseline for recognition. In: Proc. of CVF 2014. pp. 512–519. IEEE (2014)

[46] Saxe, J., Berlin, K.: expose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys. arXiv preprint arXiv:1702.08568 (2017)

[47] Shannon, C.E.: A mathematical theory of communication. ACM SIGMOBILE mobile computing and communications review **5**(1), 3–55 (2001)

[48] Shi, Y., Chen, G., Li, J.: Malicious domain name detection based on extreme machine learning. Neural Processing Letters **48**(3), 1347–1357 (2018)

[49] Sood, A.K., Zeadally, S.: A taxonomy of domain-generation algorithms. IEEE Security & Privacy **14**(4), 46–53 (2016)

[50] Stevanovic, M., Pedersen, J.M., D'Alconzo, A., Ruehrup, S., Berger, A.: On the ground truth problem of malicious dns traffic analysis. Computers & Security **55**, 142–158 (2015)

[51] Sun, X., Tong, M., Yang, J., Xinran, L., Heng, L.: Hindom: A robust malicious domain detection system based on heterogeneous information network with transductive classification. In: Proc. of RAID 2019. pp. 399–412. USENIX Association (2019)

[52] Sun, X., Yang, J., Wang, Z., Liu, H.: Hgdom: Heterogeneous graph convolutional networks for malicious domain detection. In: Proc. of NOMS 2020. pp. 1–9. IEEE (2020)

[53] Torroledo, I., Camacho, L.D., Bahnsen, A.C.: Hunting malicious TLS certificates with deep neural networks. In: Proc. of AISec 2018. pp. 64–73. ACM (2018)

[54] Vosoughi, S., Vijayaraghavan, P., Roy, D.: Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In: Proc. of SIGIR 2016. pp. 1041–1044. ACM (2016)

[55] Woodbridge, J., Anderson, H.S., Ahuja, A., Grant, D.: Predicting domain generation algorithms with

long short-term memory networks. arXiv preprint arXiv:1611.00791 (2016)

[56] Xiao, W., Zhang, J., Li, Y., Zhang, S., Yang, W.: Class-specific cost regulation extreme learning machine for imbalanced classification. Neurocomputing ***261****, 70–82 (2017)*

[57] Yang, L., Liu, G., Dai, Y., Wang, J., Zhai, J.: Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework. IEEE Access ***8****, 82876–82889 (2020)*

[58] Yu, B., Gray, D.L., Pan, J., De Cock, M., Nascimento, A.C.: Inline dga detection with deep networks. In: Proc. of ICDMW 2017. pp. 683–692. IEEE (2017)

[59] Yu, B., Pan, J., Hu, J., Nascimento, A., De Cock, M.: Character level based detection of dga domain names. In: Proc. of IJCNN 2018. pp. 1–8. IEEE (2018)

[60] Yu, T., Zhauniarovich, Y., Khalil, I., Dacier, M.: A survey on malicious domains detection through dns data analysis. ACM Computing Surveys ***51****(4) (2018)*

[61] Zhang, J., Li, Y., Xiao, W., Zhang, Z.: Robust extreme learning machine for modeling with unknown noise. Journal of the Franklin Institute ***357****(14), 9885–9908 (2020)*

[62] Zhang, J., Xiao, W., Li, Y., Zhang, S.: Residual compensation extreme learning machine for regression. Neurocomputing ***311****, 126–136 (2018)*

[63] Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. arXiv preprint arXiv:1509.01626 (2015)

[64] Zhao, H., Chang, Z., Bao, G., Zeng, X.: Malicious domain names detection algorithm based on N*-gram. Journal of Computer Networks and Communications* ***2019****, 1–9 (2019)*

[65] Zhao, J., Wang, Z., Park, D.S.: Online sequential extreme learning machine with forgetting mechanism. Neurocomputing ***87****, 79–89 (2012)*