情報処理学会論文誌 Vol.63 No.4 1052–1071 (Apr. 2022)

n-gram抽出と機械学習を用いた亜種マルウェア分類手法の提案と評価

瀧口 翔貴[1,a)] 宇田 隆哉[1,b)]

受付日 2021年4月19日, 採録日 2022年1月11日

概要：マルウェア検出に機械学習を用いる研究はあるが，攻撃者に手法が既知である場合に検出を回避されるものがある．バイナリの n-gram に対して情報利得を求め，その値の高いもののみを利用して機械学習を行う方式にも問題はあり，すべてのマルウェアに有効な対策手法を考案することは困難である．そこで，本論文では，対象を単純にパターンマッチングできない亜種マルウェアに限定することで，n-gram を用いて亜種マルウェアからコードを抽出し，機械学習を用いてこれを検出する手法を提案する．マルウェアのバイナリすべてを画像化して機械学習により検出する手法は存在するが，本研究の手法では，機械学習に入力する前に検体のサイズを小さくできる．サイズ評価では，16gram を用いた場合の平均で約 40 分の 1〜110 分の 1 程度に縮小させることに成功した．検出評価では，それぞれ 670 から 1,500 個の亜種マルウェアファミリの検体と 1,500 個の良性ソフトウェアを使用し，数個の誤分類が生じたのみであった．

キーワード：亜種マルウェア，マルウェア検出，n-gram

Proposal and Evaluation of Malware Species Classification Method by n-gram Extraction and Machine Learning

Shoki Takiguchi[1,a)] Ryuya Uda[1,b)]

Received: April 19, 2021, Accepted: January 11, 2022

Abstract: Machine learning has been used for detecting malware, but some of the methods turn useless when the methods are known by attackers. One of the best methods is a method with binary n-grams of whole files and information gain of the n-grams. Especially, selecting top k n-grams of high information gain prevents attackers from adding common n-grams in benignware. However, the method still has problems and it is difficult to find an effective method against all malwares. Therefore, in this paper, we propose a method with n-grams extraction from malware and with machine learning by limiting targets to only malware subspecies. Of course, there is an existing method which uses whole malware binaries and machine learning. Compared to that, our method can downsize both malware and benignware before inputting machine learning network. In evaluation of file size, we succeeded to downsize from 1/40 to 1/110 in average by 16grams. Also, in evaluation of malware detection, we only got some misclassifications when comparing from 670 to 1,500 samples of each malware subspecies family with 1,500 benignware samples.

Keywords: malware species, malware detection, n-gram

1. はじめに

従来，マルウェア検出方法として最も一般的なものはパターンマッチングであった．この方式は多くのアンチウイルスソフトウェアで採用されているが，感染するたびにパターンを変更したり，異なるバイナリを作り出して大量に配布するワームのような亜種マルウェアには対応できない．

そこで，機械学習を使用してマルウェアを検知する様々な手法が提案されている．最も多く研究されているものは，バイナリの n-gram を用いてマルウェアを検出する手

[1] 東京工科大学
Tokyo University of Technology, Hachioji, Tokyo 192–0982, Japan
[a)] c011520728@edu.teu.ac.jp
[b)] uda@stf.teu.ac.jp

法で，OS や命令セットの知識を必要とすることなく利用可能な点が優れている．また，仮想環境上に構築されたサンドボックス内で解析した結果から命令セットを抽出し，その n-gram を用いる手法もある．その他には，マルウェアを画像として扱うことで，画像識別に置き換えてマルウェアを検出する手法も存在する．

まず，これらに共通していえるのは，バイナリの n-gram を用いる方法が，OS や命令セットの知識を必要とすることなく，その他の問題点を考慮しても，最も優れているということである．そこで，本研究においても，マルウェア検出にバイナリの n-gram を用いている．一方，既存研究においては，攻撃者に検出手法が既知であることを想定していないものもあり，攻撃者が対策を行えてしまうものがある．また，バイナリの n-gram を用いた一部の手法は，一般的なソフトウェアで使用される API のみを使用した攻撃方法に弱いという問題がある．

本論文では，出現順序，回数，共起度が残るようにバイナリの n-gram を使用することと，検出する対象を 1 つの亜種マルウェアファミリに限定することで，上記の問題点を解決した亜種マルウェア検出する手法を提案する．

2. 関連研究

2.1 バイナリの n-gram を使用したマルウェア検出手法

2.1.1 バイナリの n-gram と出現頻度を使用したマルウェア検出手法

バイナリの n-gram を使用してマルウェアを検出する手法として最も古いものは，著者らの知る限り Kephart らによるものである [1]．Kephart らは，150 個の 512 byte からなるウイルスのブートセクタから，76,500 個の trigram を作成してトレーニングセットに使用し，重複しないものが 25,000 個であるとしている．その中から，一般的なコードへの出現頻度が 1/200,000 の trigram を選択して，シングルレイヤーの順伝播型ニューラルネットワークで機械学習している．Kephart らの手法では，ブートセクタ以外を使用しないため，ブートセクタに特徴が現れないマルウェアを検出することはできない．

Abou-Assaleh らの手法も，バイナリの n-gram を使用してマルウェアを検出する古い手法の 1 つである [2]．彼らの手法では，最も頻繁に出現する L 個の n-gram を，2 クラスの出現頻度の差を出現頻度の和で割ったものの 2 乗を合計することで，分類を行っている．

まず，マルウェアや良性ソフトウェアから取り出したバイナリの n-gram をそのまますべて使用すると膨大な量となってしまうため，何らかの選択が必要である．Abou-Assaleh らのように，マルウェアに最も頻繁に出現する L 個の n-gram を使用する方法は，攻撃者に手法が既知である場合に無効化されてしまう．たとえば，良性ソフトウェアに多く含まれる n-gram をマルウェアに大量に追加すればよい．一方，Kephart らは一般的なソフトウェアにおける出現頻度が低い n-gram を使用しており，攻撃者に手法が既知である場合にも耐性がある．

2.1.2 バイナリの n-gram とデータセットとの一致を使用したマルウェア検出手法

Santos らは，バイナリの n-gram を使ってマルウェアを検出する手法を提案している [3]．彼らの手法では，ある n-gram が悪性もしくは良性 n-gram のセットにどれだけ一致しているかで分類を行っており，悪性インスタンスの量と良性インスタンスの量の差を，分類に使用している．

彼らの手法が攻撃者に既知の場合，良性ソフトウェアに多く含まれる n-gram をマルウェアに故意に多く含むことで，検出率を下げることができてしまう．

2.1.3 バイナリの n-gram と情報利得を使用したマルウェア検出手法

Reddy らは，バイナリの n-gram の情報利得を用いてマルウェアを検出する手法を提案している [4]．具体的には，n-gram を情報利得の降順に並べ，上位の L 個の n-gram を使用している．この方法であれば，良性ソフトウェアに多く含まれる n-gram を追加することで，攻撃者がマルウェアの検出率を下げることはできない．

Boujnouni らも，バイナリの n-gram と改良された Support Vector Domain Description を使って，新規の悪性プログラムを検出する手法を提案している [5]．彼らが実際の評価において何 gram を使ったかは論文中に明記されていないが，n-gram の全セットは非常に大きいので，情報利得に基づいて，最も関連するものを使うことを推奨すると彼らが論文中で述べている．そして，実際にトップ k の n-gram を選択して使用している．

Kolter らも，バイナリの n-gram を使ってマルウェアを検出する手法を提案している [6]．実際に，1,971 個の良性実行ファイルと 1,651 個の悪性実行ファイルから，2 億 5 千 5 百万の n-gram を集めて実験を行っている．Kolter らは，n-gram の情報利得を求め，トップ 500 の n-gram を選択している．そのトップ 500 の n-gram をどのように各サンプルに適用するのか論文中の手法の説明には明記されていないが，決定木の説明より，それぞれの n-gram が含まれているかいないかを各サンプルの情報としているようである．

Fuyong らは，バイナリの n-gram 属性の類似性に基づいてマルウェアを検出する手法を提案している [7]．マルウェアと良性ソフトウェアの属性の平均値を個別に計算しており，個々の n-gram の情報利得を求めて降順にソートしている．そして，これらのソートされた個々の n-gram がその検体に含まれていれば 1，いなければ 0 として特徴ベクトルを作成している．なお，情報利得の値で降順にソートした n-gram のすべてを使用しており，マルウェアと良性ソフトウェアそれぞれの平均ベクトルに検体の n-gram が一致した数を数えている点に問題がある．これは，マルウェアと

良性ソフトウェアそれぞれの平均ベクトルに対して，0.5 より大きいものを 1，それ以外を 0 として作成したベクトルであるため，良性ソフトウェアの平均ベクトルが 0.5 より大きくなる程度に，特定の n-gram をマルウェアに追加していけば，攻撃者がマルウェアの特徴を消すことができる．

2.1.4 バイナリの n-gram を使用したその他のマルウェア検出手法

Raff らは，バイナリの n-gram を使った手法について，様々な観点かつ方法で評価している [8]．まず，カウント情報として 32 ビットか 64 ビットの整数を付けて，単純に 6-gram を保存すると，それぞれ 503 か 791 GB の RAM を使うと Raff らは述べている．これは，合計 40 万個のトレーニングセットのファイルから，ユニークな 4-gram が 4,289,759,510 個，ユニークな 6-gram が 35,953,973,975 個あったことについてである．つまり，バイナリの n-gram はデータ量として膨大であり，これをこのままの形で扱うことは困難であるということである．

Raff らは，評価のために情報利得の高い最初の 20 万個の n-gram を選択している．また，悪性スコア（Malice Score）と良性スコア（Benign Score）というものを定義して評価を行っている．これらは，マルウェアと良性ソフトウェアのクラスに含まれる n-gram の出現率の差であり，悪性 – 良性が悪性スコア，良性 – 悪性が良性スコアとなっている．このほかに，絶対悪性スコア（Absolute Malice Score）として，悪性スコアの絶対値をとったものと，ルート悪性スコア（Root Malice Score）として，悪性の平方根 – 良性の平方根をとったものも用いている．それ以外には，ジニ係数（Gini coefficient）とカルバック・ライブラー情報量（KL-divergence）での評価も行っている．

情報利得の高いものを使用する以外の評価方法では，攻撃者に手法が既知である場合，良性ソフトウェアに多く含まれる n-gram をマルウェアに追加されることで，検出率を下げられてしまう．

Raff らは，Hash-Grams という，n-gram を用いたマルウェア検出手法も提案している [9]．Raff らはこの論文中で，サンプル数が多いときに，バイナリ n-gram のトップ k を選択する方法はパフォーマンスのボトルネックになっていると述べている．そこで，彼らは n-gram に対してハッシュ値を求め，それをある数（彼らの実装では $2^{31} - 18$）で割った余りを使用している．彼らは Java の環境の都合で 32 bit の変数を用いたいためにこのようにしたと思われるが，たとえば Kolter らの方法では，もともとが 4-gram で 4 バイトであるため，これではまったく小さくならない．もちろん，8-gram や 16-gram では有効である．

2.2 命令セットの n-gram を使用したマルウェア検出手法

O'Kane らは，動的解析を通して得られた n-gram 密度ヒストグラムを使って，マルウェアの検出を行っている [10]．彼らの方式では，命令セットをオプコードとオペランドに分け，重要ではないオペランドを破棄し，オプコードのみの n-gram を作成している．彼らは pop や ret といったオプコードの密度を特徴としてマッピングしている．

O'Kane らが分類に用いているものは，オプコードの密度であるため，攻撃者に対策が既知である場合に，攻撃者は攻撃処理とは関係ないコードを大量に挿入することで，検出を逃れることができてしまう．

Mira らは，RLE と n-gram を用いてマルウェアを検出する手法の改善案を提示している [11]．RLE（Run Length Encoding）圧縮技術は，ロスなしの圧縮技術である．Mira らは，マルウェアの API コールシーケンスをリストにし，それを RLE を用いて 2-gram（2 つの API シーケンス）で圧縮している．良性ソフトウェアについても同様の圧縮を行う．なお，API コールシーケンスについては，API モニターツールによって生成されるとしか書かれていないため詳細は不明だが，何らかの方法でファイルを実行して取得していると思われる．なお，Mira らはどのようにしてマルウェアを検出するかについては言及しておらず，API コールシーケンスを可逆圧縮できるので，検出に要する時間を短縮できるとのみ結論づけている．

Lee らは，マルウェアの変異体を自動的に検出する手法を提案している [12]．Cuckoo Sandbox を用いて，マルウェアが実行された際の API データを収集している．これらの API データは n-gram を用いて API シーケンスにグループ化され，それらの API シーケンスの頻度から，悪性コードの類似性を算出している．

2.3 バイナリを画像化したマルウェア検出手法

Yakura らは，バイナリデータを画像に変換し，CNN（Convolutional Neural Network）に入力してマルウェアを検出する研究を行っている [13]．CNN は画像の分類によく用いられており，彼らの CNN におけるネットワーク構造はマルウェアに特化したものであるとは述べられていないため，画像のネットワークをそのまま利用したと考えられるが，CNN は 1 次元のデータでも扱うことは可能であり，1 次元のバイナリ列であるソフトウェアを 2 次元の画像に変換することは適切ではない．これは，ソフトウェアのバイナリ列は，Y 軸方向に相関がないためである．実際に，2 つの畳み込み層でも 3×3 の 2 次元で畳み込みを行っているが，Y 軸方向の畳み込みは不適切である．この点については著者らも論文中で言及しており，バイナリデータの挿入や削除があると縦方向の位置関係が壊れると述べている．また，nop 命令の挿入によって画像が完全に変わってしまう可能性がある一方，n-gram 方式もこの影響を受けることにも言及している．なお，彼らの方式では，100 KB を超えるソフトウェアは 1,024 ピクセル幅の画像に変換し

ているが，CNN では入力するサンプルのサイズを揃える必要があり，1 つでも大きいソフトウェアのサンプルがあると，入力するすべての画像サイズが大きくなってしまい，計算量が大きくなる．

Yakura らの研究には続きがある [14]．手法は Yakura らの先述の論文と同様だが，1 次元 CNN での実験が追加されている．これは，2017 年に発表された McLaughlin らによる論文における実験の再現である [15]．結果としては，1 次元 CNN より Yakura らの 2 次元 CNN のほうが良いが，その理由は，1 次元 CNN ではコードセクションしか用いていないが，2 次元 CNN ではデータセクションも用いているためであると Yakura らが述べている．つまり，分類精度を上げるにはデータセクションの使用も必須であるが，それでは機械学習の計算量が削減できない．

3. 提案手法

3.1 関連研究の問題点と 5 つの特徴による解決

提案手法について説明する前に，提案手法の 5 つの特徴をあげ，それによって 2 章であげた関連研究の問題点がどのように解決されるのか説明する．

本手法の特徴は次のとおりである．

特徴 1 良性ソフトウェアが使用する API コールのみでマルウェアが構成されていても検出可能なよう，特定のバイナリの出現順序，回数，共起度を使用する．

特徴 2 精度を落とさないよう，コードセクションのみではなくデータ全体を使用する．

特徴 3 処理コストを削減するため，機械学習に入力する際のデータ量を少なくする．

特徴 4 良性ソフトウェアに多く含まれるバイナリをマルウェアに追加されても検出精度が下がらないようにする．

特徴 5 マルウェアを実行して命令セットや API コールを抽出する手法は問題があるため，マルウェアを実行しない．

まず，特徴 1 が解決する問題点について説明する．

2.1 節であげた，バイナリの n-gram を使用したマルウェア検出手法は，OS や命令セットの知識を必要とすることなくマルウェア検出が行えるため，最も優れているといえる．しかし，2.1.1 項で述べた，バイナリの n-gram と出現頻度を使用したマルウェア検出手法には問題がある．まず，Abou-Assaleh らの手法のように出現頻度が高いものを使用すると，攻撃者に手法が既知である場合に，良性ソフトウェアに多く含まれるバイナリを大量に追加された場合，そちらの n-gram の出現頻度が高くなってしまい検出率が下がる．また，Kephart らの手法では出現頻度が低い n-gram を使用しているためこの問題は起きないが，亜種マルウェアで共通の命令セットを大量に使用して攻撃していても，これらは出現頻度が高いため特徴量として使用できなくなってしまう．

2.1.3 項で述べたバイナリの n-gram と情報利得を使用したマルウェア検出手法にもいくつかの問題点がある．Fuyong らの手法の問題点については 2.1.3 項ですでに述べたため本節では省略するが，Reddy ら，Boujnouni ら，Kolter らの手法では，情報利得の高い n-gram の上位から一定個数を使用するという点で共通点があり，それらの n-gram は，ある検体中に存在するかしないかという使用法になっているところに問題がある．たとえば，キーロガーのマルウェアがあるとして，ソフトウェアキーボードを使用する良性ソフトウェアと同様の API コールを使用した場合，これらは情報利得の高い n-gram にはならないか，該当する良性ソフトウェアが高い確率でマルウェアに誤分類されることになる．ユーザの了解を得て特定のファイルにアクセスする良性ソフトウェアも存在するが，ユーザの了解を得ずにそのファイルにアクセスするマルウェアと，使用している命令セットだけで区別することは困難といえる．なお，マルウェアは，盗み取った情報をどこかのサーバに送信するかもしれないが，通信する IP アドレスをマルウェア検出の決め手とするのは危険である．当然，いつまでも同じサーバに情報を送信していると気づかれてしまうため，一定期間が経てば通信先の IP アドレスが変化する．その際に，新しい IP アドレスと通信するマルウェアを検出できなくなる．

2.1.4 項で述べたように，Raff らはバイナリの n-gram はデータ量として膨大であることを指摘している．Raff らが提案する Hash-Grams は，ハッシュによって n-gram の種類を少なくできるだけであり，その後の処理過程で発生する問題点については，Reddy ら，Boujnouni ら，Kolter らの手法によるものと同様である．

以上のように，関連研究では，特定の n-gram の出現頻度か，特定の n-gram が出現するかどうかに基づいて分類を行っている．これに対して，本論文の提案手法では，特定のバイナリ n-gram の出現順序，回数，共起度が判断基準となるようになっており，同一の n-gram が近い割合で含まれていても，マルウェアを識別可能とする点に特徴がある．研究段階においては，既存のマルウェアに対して検知手法の有効性を評価するが，実際の環境では検知手法が確立されている状態で新たなマルウェアが作成される．攻撃者に検知手法が既知である場合，データセクションを用いれば特定の n-gram の出現頻度は変更が容易である．また，プログラムによっては，良性ソフトウェアに出現する n-gram のみでマルウェアを作成可能な場合もある．これに対して，ある目的を持ったマルウェアを作成するにあたり，特定のバイナリ n-gram の出現順序，回数，共起度の特徴を消して作成することは困難であると考えている．

次に，特徴 2 が解決する問題点について説明する．

自由にファイルサイズを決められるマルウェアの場合，

機械学習においてすべてのバイナリを扱うことは非常に処理コストが掛かる．2.3 節で述べたように，McLaughlin らはコードセクションしか用いないことでデータ量を削減しているが，データセクションも用いた Yakura らの手法のほうが結果が良いことから，データ全体を用いたほうが検出精度が上がることが分かる．本論文の提案手法においても，コードセクションだけでなくデータセクションも用いることでこの問題を解決している．Reddy ら，Boujnouni ら，Kolter らの手法も特徴 2 を満たすが，同時に特徴 1 も満たすところが彼らの手法との違いである．

さらに，特徴 3 が解決する問題点について説明する．

2.3 節であげた，Yakura らのバイナリを画像化したマルウェア検出手法は，バイナリの n-gram がデータ量として膨大であるという問題を解決した，1 つの手法である．n-gram に分解することなく，そのままのデータを用いるため扱うデータサイズは小さくなる．データ全体を扱うことで，2.1.3 項で述べたバイナリの n-gram と情報利得を使用したマルウェア検出手法に見られる問題点も解決できる．良性ソフトウェアが使用する API コールのみでマルウェアが構成されていても，その出現順序，回数，共起度を分類に使用できるため，これらが区別できる．

しかし，Yakura らの手法では，機械学習に入力するマルウェアのサンプルサイズがオリジナルの状態からまったく減っておらず，訓練のコストが膨大となる．これに対して本論文の提案手法では，機械学習に入力する際のデータ量を削減できる．

また，特徴 4 が解決する問題点について説明する．

2.1.2 項で述べた Santos らの手法は，ある n-gram が悪性もしくは良性 n-gram のセットにどれだけ一致しているかで分類を行うものであるが，手法が攻撃者に既知の場合，良性ソフトウェアに多く含まれる n-gram をマルウェアに故意に多く含むことで，検出率を下げることができてしまう．このような任意の n-gram を簡単にマルウェアに追加できる場所はデータセクションであるため，データセクションを分類に用いないという考え方もできるが，それでは特徴 2 が解決する問題が残ってしまう．本論文の提案手法では，出現順序，回数，共起度を分類に用いることで，この問題を解決している．

最後に，特徴 5 が解決する問題点について説明する．

2.2 節であげた，命令セットの n-gram を使用したマルウェア検出手法は，命令セットを取得するために仮想環境か防御された実環境でマルウェアを実行する点に問題がある．たとえば，一定の時期まで悪意のある行動を起こさないワームの場合，命令セットを観測していても，悪意のあるものは見つからない．また，何時間観測を行うかは論文中に書かれていないが，攻撃者に手法が既知である場合，観測される時間が経過した後で悪意のある行動を起こすマルウェアを作成すればよい．

これに対して，本論文の提案手法ではマルウェアを実行しないため，この問題は発生しない．

本論文の提案手法は，特徴 1〜5 を同時に満たすことで，関連研究の個々の問題をそれぞれ解決するものである．

3.2 提案概要

3.1 節で述べた問題点を受けて，本研究では，マルウェアおよび良性ソフトウェアのファイルから部分的な抽出を行うことで，効率的に機械学習を用いて 1 つのファミリの亜種マルウェアを検出する手法を提案する．関連研究は必ずしも亜種マルウェアを対象としたものではないが，特徴 1〜5 を満たすためには，1 つのファミリの亜種マルウェア検出に限定する必要がある．もちろん，1 つの亜種マルウェアファミリを検出するフィルタを直列に複数重ねることにより，亜種マルウェアファミリであれば同時に複数ファミリを検出できる．

なお，本論文は，著者らが電子情報通信学会マルチメディア情報ハイディング・エンリッチメント研究会で発表を行った論文（以下 EMM 論文）[16] をまとめたものであるが，EMM 論文で述べた手法と実際に評価した手法に違いがあったため，本論文では実際に評価に使用した手法を提案手法とする．

提案手法では，マルウェアおよび良性ソフトウェアから，バイナリの一部を n-gram を用いて抽出し，それをサンプルとして機械学習を用いてマルウェア検出を行う．機械学習には，機械学習の 1 つである CNN を用いて分類を行ったが，関連研究と同程度以上の値で適切に分類できることを示すための一例として用いており，分類精度が最善になるものとして選択しているわけではない．また，CNN のパラメータも固定で事前に決定しており，最適なハイパーパラメータを求めることを目的としてはいない．なお，Yakura らも CNN を用いているように，CNN は回転，平行移動，変形などに強く，画像分類に適していることから AlexNet や GoogLeNet などでも用いられている．亜種マルウェアも，これらと同様の特徴があると考え，我々も CNN を選択した．

本手法における訓練の流れを以下に示す．なお，本論文におけるサンプルとは，機械学習のネットワークに入力される個々のデータのことを指す．

(1) すべての訓練用サンプル（特定のマルウェアファミリの検体および良性ソフトウェアの検体）のバイナリコードを取得する．
(2) 訓練用サンプルのうち最小サイズのものを ‘比較用検体’ とし，すべての訓練用サンプルにおいて，比較用検体に含まれるバイナリ n-gram と一致する部分のみを抽出する．
(3) 抽出済みのすべての訓練用サンプルを入力して機械学習を行い，訓練済みモデルを作成する．

本手法におけるテストの流れを以下に示す．

(1) 分類対象となるテストサンプル（検体）のバイナリコードを取得する．
(2) 分類に使用する訓練済みモデル（特定の亜種マルウェアファミリ）の比較用検体を使用し，テストサンプルにおいて比較用検体に含まれるバイナリ n-gram と一致する部分のみを抽出する．
(3) 抽出済みのテストサンプルを訓練済みモデルに入力し，その亜種マルウェアファミリか良性ソフトウェアかの 2 クラス分類を行う．

なお，訓練済みモデルは亜種マルウェアのファミリごとに作成され，そのファミリかどうかの判定に使用される．つまり，2 クラス分類ではその亜種マルウェアファミリか良性ソフトウェアかで分類されるが，実際にはその亜種マルウェアファミリかそうでないかとなり，良性ソフトウェアかどうかの保証はない．

3.3 n-gram 抽出手法

本節の特徴 3 を満たすため，マルウェアや良性ソフトウェアのバイナリ全体をそのまま使用するのではなく，n-gram 抽出を行ってサイズを小さくしてから機械学習のネットワークに入力している．本論文で n-gram 抽出（圧縮）を説明する際に，使用する用語についてあらかじめ決めておく．先に述べたように，訓練用サンプルのうち最小サイズのものを ‘比較用検体’ と呼称している．また，比較用検体によって n-gram を抽出されるサンプルを ‘目標ファイル’ と呼称する．

図 1 に，比較用検体から n-gram リストを作成する方法を示す．

図は 4-gram の例であり，4 バイトずつの組をリストに登録していく．n-gram であれば n バイトずつの組となる．このとき，図のように 1 バイトずつずらしながらリストを作成するため，リストの次の項目に登録されるバイト列の先頭の $n-1$ バイトは，その項目の最後の $n-1$ バイトと同一となる．

次に，データの抽出方法について説明する．**図 2** に n-gram リストによる，目標ファイルからのデータの抽出方法を示す．

図のように，作成した n-gram リストにあるいずれか 1 つの項目と，目標ファイル中のバイト列が一致した場合には，そのバイト列の位置にマークを付け，最終的にマークされなかった部分を削除することで圧縮を行うと EMM 論文で述べた．しかし，プログラムを再度確認したところ，実際の実装はこのとおりに行われていなかった．図中の Byte Array of Target File において，3 カ所に下線が付けられている．このうち，「5C BD 36 FA」と「BD 36 FA 90」の箇所にマークを付けるというのは正しいが，先に「BD 36 FA 90」の箇所にマークが付けられていた場合，実際の実装においては「36 FA 90 DA」にはマークが付けられない．実際の実装においてマークを付けていた箇所は，その時点でどのバイトにもマークが付けられていない箇所のみであった．たとえば，16-gram を使用する場合，目標ファイルの中で，作成した 16-gram リストのある項目と一致する箇所を発見した場合，その 16 バイトすべてが未マークの状態でなければ，その 16 バイトのうちどのバイトにもマークを付けない．

本論文では，EMM 論文中で説明した方法を n-gram 圧縮（n-gram Compression），実際の実装に合わせた提案手法を n-gram 抽出（n-gram Extraction）と呼称する．n-gram 圧縮では理想的なファイルサイズにならなかった点，n-gram 抽出で実用的な亜種マルウェア検出精度を示せる点から，本論文の亜種マルウェア検出精度についての評価においては n-gram 抽出を使用する．

3.4 機械学習による亜種マルウェア検出

本提案の n-gram 抽出を用いた後，機械学習により亜種マルウェアを検出する．

亜種マルウェアは，決められたアルゴリズムに従って一度に大量生成され，それがメールなどを介して様々な宛先に流布されるか，感染して動作しているマルウェアが，決められたアルゴリズムに従って異なるバイナリコードを持つ複製を作成し，別所に感染させるかの方法で拡散する．よって，流通している同一ファミリの亜種マルウェアでバイナリコードが異なる検体を一定数確保し，機械学習を用いて訓練を行えば，検出されずに流通中のそのファミリの亜種マルウェアを，訓練済みモデルを用いて検出可能になる．

我々は，評価に亜種マルウェアのデータセットを用いており，同一ファミリの亜種マルウェアを無制限に生成でき

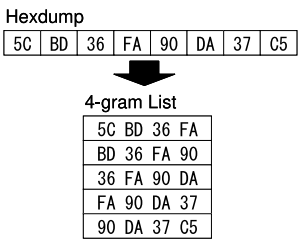

図 1 n-gram リストの作成方法
Fig. 1 n-gram list creation method.

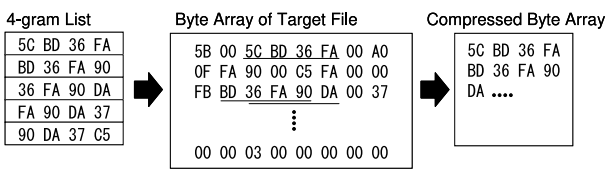

図 2 n-gram リストによる目標ファイルからのデータ抽出方法
Fig. 2 Data extraction method from target file by n-gram list.

るわけではないため，k-分割交差検証を用いて各ファミリの亜種マルウェアが正しく検出できるかどうか評価を行った．本来，機械学習におけるサンプル（ネットワークに入力される個々のデータ）のうち，検証用サンプルとテストサンプルは別のものであるが，前述の理由から，同一アルゴリズムで生成された亜種マルウェアが同一ファミリに区分されるため，同一データセットにある同一ファミリの一部データを訓練用サンプルに，残りをテスト用サンプルに使用して差し支えないものとした．訓練用サンプルとテストサンプルの選択には k-分割交差検証を用いるが，これは検証ではなくテストであるため，訓練におけるいかなるパラメータの決定にもテストサンプルが影響しないようにした．

機械学習のアルゴリズムには CNN を用いた．機械学習のアルゴリズムを変更することで亜種マルウェアの検出率がどの程度変化するかや，パラメータのチューニングによってどこまで精度を上げられるかなどは，本論文の対象外とする．

機械学習のネットワークにサンプルを入力する際，ファイルサイズを揃える必要がある．そこで，n-gram 抽出後のファイルの中で，ファミリごとに最大サイズのサンプルと同一のファイルサイズになるように，その他のサンプルをゼロパディングする．

亜種マルウェア検出の評価指標には，Accuracy，Precision，Recall，F 値を用いた．情報セキュリティ分野の攻撃検知においては，攻撃を検知することが目的であるため，攻撃が陽性（positive）であり，攻撃ではないものが陰性（negative）である．本論文においても同様に，マルウェアをマルウェアと判断した場合を TP（True Positive），良性ソフトウェアを良性ソフトウェアとして判断した場合を TN（True Negative），マルウェアを良性ソフトウェアとして判断した場合を FN（False Negative），良性ソフトウェアをマルウェアと判断した場合を FP（False Positive）とした．前述の Accuracy，Precision，Recall，F 値は，これらの TP，TN，FN，FP から算出している．

4. 実装

実行環境は，CPU Intel Core i5-8400，RAM 32 GB，OS Ubuntu 18.04.2 LTS であり，使用した言語およびライブラリのバージョンは，Python 3.6.8，NumPy 1.16.1 である．

CNN は Keras 2.2.4，Tensorflow 1.12.0 を用いて実装した．入力層と出力層の間のネットワークの構造は，Conv1D，Conv1D，MaxPooling1D，BatchNormalization，Conv1D，Conv1D，MaxPooling1D，BatchNormalization，Flatten，Dense，Dropout，Dense の順で層をつないだものである．畳み込み層の活性化関数は Rectified Linear Unit（ReLu），カーネルサイズは 3 とした．また，2 クラス分類であるためバッチサイズは最低の 4，エポック数は 3 とした．ステップ数は None にしたためデフォルト値が適用される．最終的な活性化関数はソフトマックス関数である．サンプルの読み込みには NumPy を用いた．

5. 評価

4 つのファミリの亜種マルウェアを用いて評価を行った．亜種マルウェアに関しては，MWS データセットの CCC Dataset 2012 および 2013 に含まれる 4 種のファミリのものである．マルウェアのファミリ名には Microsoft のものを採用した [17]．このマルウェアのサンプル内にハッシュ値が同一の検体は 1 つも含まれていない．良性ソフトウェアのサンプルは，Windows にプリインストールされている拡張子が exe のファイルおよび，Download.com Powered by Cnet からダウンロードされた実行ファイル 1,500 個である [18]．この良性ソフトウェアのサンプル 1,500 個内にハッシュ値が同一の検体は 1 つも含まれていない．

本論文中における亜種マルウェアファミリの表記は，Net-Worm.Win32.Allaple.a がファミリ A，Net-Worm.Win32.Allaple.b がファミリ B，Net-Worm.Win32.Allaple.e がファミリ E，Net-Worm.Win32.Kido.ih がファミリ IH である．

データセットにおける亜種マルウェアの個々の検体には，その検体のハッシュ値がファイル名として付けられている．そこで，本論文において個々の検体を指す場合には，「0a1b...8e9f」のように，ファイル名の前後 4 文字ずつを使用して参照するものとする．これは，検体のハッシュ値すべてを公開してはいけないという，MWS データセットの取り決めによるものである．

3.4 節で述べたが，k-分割交差検証を用いて各ファミリの亜種マルウェアが正しく検出できるかどうか評価を行う．その際，提案手法のとおりに比較用検体を全体のサンプルに混ぜてしまうと，k-分割により k 回に 1 回は比較用検体がテストサンプルに入ってしまうことになり不適切である．なぜなら，テストサンプルは訓練におけるいかなるパラメータの決定にも影響を与えてはいけないからである．そこで，本評価においては，比較用検体が目標ファイルとならないよう，各ファミリのすべての検体から，比較用検体として選んだものを 1 つ除外し，残りの検体を訓練用およびテスト用として使用した．

5.1 n-gram 抽出に用いる比較用検体の決定

3 章で述べた提案手法に従い，各ファミリの中で最小サイズの検体を比較用検体として n-gram 抽出を行い，評価を行った．その際，まず，ファミリごとに検体のファイルサイズを調査した．結果を**表 1**，**表 2**，**表 3**，**表 4** に示す．表中の検体数合計が（+1）という書き方をしているのは，訓練およびテストに使用する検体とは別に，比較用検体を 1 つ用意しているためである．なお，比較用検体は表中の個数に含まれていない．

表 1 Net-Worm.Win32.Allaple.a のファイルサイズと検体数
Table 1 File size and number of samples of Net-Worm.Win32.Allaple.a.

サイズ	個数
57,856 バイト	401 個
65,024 バイト	292 個
93,696 バイト	207 個
検体数合計	900（+1）

表 2 Net-Worm.Win32.Allaple.b のファイルサイズと検体数
Table 2 File size and number of samples of Net-Worm.Win32.Allaple.b.

サイズ	個数
57,856 バイト	868 個
62,985 バイト	110 個
63,488 バイト	522 個
検体数合計	1.500（+1）

表 3 Net-Worm.Win32.Allaple.e のファイルサイズと検体数
Table 3 File size and number of samples of Net-Worm.Win32.Allaple.e.

サイズ	個数
50,176 バイト	1 個
59,904 バイト	1 個
60,690 バイト	187 個
62,976 バイト	3 個
62,985 バイト	1 個
63,488 バイト	1 個
64,512 バイト	2 個
65,024 バイト	1 個
67,584 バイト	2 個
77,312 バイト	2 個
78,336 バイト	177 個
85,504 バイト	410 個
93,696 バイト	1 個
94,890 バイト	576 個
96,256 バイト	1 個
114,176 バイト	133 個
検体数合計	1,499（+1）

表 4 Net-Worm.Win32.Kido.ih のファイルサイズと検体数
Table 4 File size and number of samples of Net-Worm.Win32.Kido.ih.

サイズ	個数
80,300 バイト（最小）	1 個
それ以外	668 個
1,510,560 バイト（最大）	1 個
検体数合計	670（+1）

比較用検体を 1 検体のみ用いて評価を行うと，その結果は偶然偏ったものになる可能性がある．そこで，ファミリごとにもう 1 検体を比較用検体として評価を行った．その際，提案手法とは異なるが，最小サイズの検体ではないものを用いた．これは，実際の環境で，訓練のために捕獲されたマルウェアのうち，最小サイズのものが，本論文で使用したデータセットの最小サイズと等しくなるとは限らない場合を想定している．比較用検体を変更する際には，それまで比較用検体として使用していたものを代わりに訓練およびテスト用の検体とし，訓練およびテスト用の検体の総数が変わらないようにした．

ファミリ A の結果を表 1 に示す．表より，ファミリ A の n-gram 抽出の比較用の検体としては，最小サイズである 57,856 バイトの 0d36...6ab0 を使用した．また，最小サイズ以外の検体を比較用の検体とした場合の評価として，65,024 バイトの 0017...b3ab を用いた．

ファミリ B の結果を表 2 に示す．表より，ファミリ B の n-gram 抽出の比較用の検体としては，最小サイズである 57,856 バイトの 0a33...a22f を使用した．また，最小サイズ以外の検体を比較用の検体とした場合の評価として，62,985 バイトの 03e0...b227 を用いた．

ファミリ E の結果を表 3 に示す．表より，ファミリ E の n-gram 抽出の比較用の検体としては，最小サイズである 50,176 バイトの 63c8..da99 を使用した．また，最小サイズ以外の検体を比較用の検体とした場合の評価として，60,690 バイトの 30e6...59db を用いた．なお，EMM 論文では，最小サイズのものを比較用の検体としたと書かれていたが，実際には 60,690 バイトの 30e6...59db が用いられていたため，ここで訂正する．また，EMM 論文では，検体数が 1,500（+1）と書かれていたが，60,690 バイトの 30e6...59db が訓練用およびテスト用の検体にも混在していたため，これを除去した．よって，10 分割交差検証の際には，10 分割された最後のものだけが，サンプル数が 1 つ少なくなる．

ファミリ IH の結果を表 4 に示す．ファミリ IH は，数個同じファイルサイズのものがあるが，ほとんどサイズが異なっていたため記載を省略する．表より，ファミリ IH の n-gram 抽出の比較用の検体としては，最小サイズである 57,856 バイトの 6ede...7842 を使用した．また，最小サイズ以外の検体を比較用の検体とした場合の評価として，80,300 バイトの 3ab3...1baf を用いた．

参考のため，良性ソフトウェアのファイルサイズについても述べる．評価に使用した良性ソフトウェア 1,500 個の平均ファイルサイズは 552,375.2 バイト，標準偏差は 992,813.9 バイトであった．

5.2 n-gram 抽出によるファイルサイズの評価

n-gram 抽出および圧縮によるファイルサイズの評価を**表 5**，**表 6**，**表 7**，**表 8** に示す．比較用の検体は 5.1 節で決定したものである．n-gram 圧縮による評価も行ったのは，n-gram 抽出との比較のためである．

表の Name は亜種マルウェアのファミリ名，Original は

表 5 最小サイズの検体で n-gram 抽出した場合のファイルサイズ
Table 5 Extracted file size by n-gram with smallest sample.

Name	Original	4gram	8gram	16gram	S4gram	S8gram	S16gram	R4gram	R8gram	R16gram
famA 良性 Ave	552,375.2	2,694.8	1,867.4	1,485.7	551,027.7	551,441.5	551,632.3	1.73	1.36	1.09
famA 良性 Std	992,813.9	1,052.3	542.7	431.8	992,564.9	992,740.0	992,745.2	1.96	1.63	1.34
famA 悪性 Ave	6,8424.8	11,445.8	5,746.8	1,652.9	62,701.9	65,551.4	67,598.4	8.67	4.36	1.26
famA 悪性 Std	14,156.5	1,112.9	1,106.5	63.1	14,189.6	14,181.1	14,175.1	1.73	1.14	0.24
famB 良性 Ave	552,375.2	2,814.3	1,924.6	1,440.2	550,968.0	551,412.9	551,655.1	1.79	1.38	1.06
famB 良性 Std	992,813.9	1,111.4	569.8	417.6	992,543.3	992,728.9	992,746.5	2.02	1.65	1.30
famB 悪性 Ave	60,192.1	10,462.2	5,352.2	1,687.8	54,961.0	57,516.0	59,348.1	8.71	4.46	1.41
famB 悪性 Std	2,740.5	965.9	944.2	37.3	2,800.9	2,792.6	2,745.4	0.91	0.82	0.08
famE 良性 Ave	552,375.2	5,363.3	3,880.6	3,259.3	549,693.5	550,434.9	550,745.5	3.42	2.76	2.32
famE 良性 Std	992,813.9	2,491.0	1,439.9	1,139.0	992,375.5	992,686.7	992,650.3	3.90	3.28	2.81
famE 悪性 Ave	87,526.8	6,126.1	3,834.1	2,286.6	84,463.8	85,609.8	86,383.6	3.64	2.29	1.38
famE 悪性 Std	13,799.6	3,457.3	3,536.9	3,573.7	14,004.5	14,019.8	14,014.4	3.53	3.55	3.57
famIH 良性 Ave	552,375.2	6,280.1	2,649.8	2,142.9	549,235.1	551,050.2	551,303.7	3.02	1.91	1.57
famIH 良性 Std	992,813.9	4,170.8	822.2	672.7	991,751.8	992,703.0	992,708.0	3.04	2.30	1.91
famIH 悪性 Ave	200,543.5	9,485.9	3,069.6	2,753.7	195,800.6	199,008.7	199,166.6	3.01	0.97	0.87
famIH 悪性 Std	200,423.5	4,307.1	412.5	537.6	200,566.2	200,450.7	200,478.3	1.69	0.31	0.31

表 6 最小サイズではない検体で n-gram 抽出した場合のファイルサイズ
Table 6 Extracted file size by n-gram with another sample.

Name	Original	4gram	8gram	16gram	S4gram	S8gram	S16gram	R4gram	R8gram	R16gram
famA 良性 Ave	552,375.2	2,813.1	1,897.4	1,401.3	550,968.6	551,426.5	551,674.5	1.78	1.35	1.04
famA 良性 Std	992,813.9	1,117.7	573.4	400.3	992,535.2	992,724.4	992,755.9	2.00	1.61	1.29
famA 悪性 Ave	68,416.9	11,231.9	5,601.3	1,671.4	62,800.9	65,616.2	67,581.2	8.50	4.23	1.27
famA 悪性 Std	14,160.4	1,071.2	1,054.5	58.6	14,135.6	14,145.5	14,174.2	1.60	1.03	0.24
famB 良性 Ave	552,375.2	2,911.9	1,710.3	1,244.3	550,919.2	551,520.0	551,753.0	1.82	1.21	0.91
famB 良性 Std	992,813.9	1,163.7	523.7	363.2	992,512.5	992,730.0	992,759.9	2.02	1.42	1.12
famB 悪性 Ave	60,188.6	11,138.8	5,946.4	2,325.0	54,619.3	57,215.4	59,026.1	9.24	4.92	1.91
famB 悪性 Std	2,740.2	2,797.5	2,801.6	2,685.4	2,710.1	2,715.3	2,678.6	2.16	2.19	2.12
famE 良性 Ave	552,375.2	4,353.7	3,350.4	2,826.5	550,198.3	550,700.0	550,961.9	2.87	2.43	2.06
famE 良性 Std	992,813.9	1,949.7	1,205.6	930.2	992,497.0	992,712.6	992,695.1	3.34	2.91	2.50
famE 悪性 Ave	87,533.3	5,964.8	3,731.5	2,217.3	84,550.9	85,667.6	86,424.7	3.54	2.23	1.33
famE 悪性 Std	13,784.3	3,435.3	3,514.0	3,548.5	13,996.4	14,006.3	13,996.5	3.25	3.26	3.28
famIH 良性 Ave	552,375.2	6,070.9	2,113.4	1,518.8	549,339.7	551,318.5	551,615.7	2.62	1.49	1.12
famIH 良性 Std	992,813.9	4,557.2	633.2	431.3	991,572.2	992,708.7	992,746.0	2.47	1.76	1.37
famIH 悪性 Ave	200,510.0	17,533.0	10,599.0	10,287.7	191,743.5	195,210.5	195,366.2	5.44	3.22	3.13
famIH 悪性 Std	200,445.4	32,789.1	34,000.6	34,094.7	201,080.6	200,990.8	201,020.2	10.32	10.62	10.65

元のファイルサイズ，4gram，8gram，16gram は，それぞれの n-gram で抽出または圧縮した後のファイルサイズ（バイト），S4gram，S8gram，S16gram は，それぞれの n-gram で抽出または圧縮した際のファイルサイズ減少量（バイト），R4gram，R8gram，R16gram は，それぞれの n-gram で抽出または圧縮した際のファイルサイズ比率（元のファイルサイズに対するパーセンテージ）である．

表の famA 良性 Ave は，ファミリ A の亜種マルウェアで良性ソフトウェアを処理した際の平均値を示す．famA 良性 Std は，ファミリ A の亜種マルウェアで良性ソフトウェアを処理した際の標準偏差を示す．famA 悪性 Ave は，ファミリ A の亜種マルウェアでファミリ A の亜種マルウェアを処理した際の平均値を示す．famA 悪性 Std は，ファミリ A の亜種マルウェアでファミリ A の亜種マルウェアを処理した際の標準偏差を示す．famB〜famIH も同様である．

まず，提案手法のとおり，最小サイズの検体で n-gram 抽出した場合にファイルサイズがどのようになるのかの結果を表 5 に示す．

次に，最小サイズではない検体で n-gram 抽出した場合のファイルサイズがどのようになるのかの結果を表 6 に示す．ある亜種マルウェアファミリの検体を集める際に，データセットによっては必ずしも最小サイズの検体が同一のものとはならない．よって，異なる検体で n-gram 抽出した場合にどの程度結果に差があるのか確認した．ファイル圧縮率で顕著な差があったのは，famIH 悪性であった．

表 7 最小サイズの検体で n-gram 圧縮した場合のファイルサイズ
Table 7 Compressed file size by n-gram with smallest sample.

Name	Original	4gram	8gram	16gram	S4gram	S8gram	S16gram	R4gram	R8gram	R16gram
famA 良性 Ave	552,375.2	46,584.9	29,898.7	25,246.4	505,790.2	522,476.5	527,128.7	14.59	9.46	7.87
famA 良性 Std	992,813.9	128,178.6	106,191.0	100,348.0	943,784.2	960,735.2	965,730.1	11.19	9.83	9.36
famA 悪性 Ave	68,424.8	6,758.7	3,773.3	1,191.9	61,666.1	64,651.5	67,232.9	10.24	5.72	1.81
famA 悪性 Std	14,156.5	499.9	534.3	33.8	14,163.7	14,175.1	14,176.3	1.90	1.28	0.34
famB 良性 Ave	552,375.2	46,374.4	30,374.1	25,221.7	506,000.8	522,001.1	527,153.5	14.53	9.61	7.85
famB 良性 Std	992,813.9	127,669.9	106,597.0	100,362.6	944,148.0	960,235.9	965,733.9	11.18	9.84	9.36
famB 悪性 Ave	60,192.1	6,138.6	3,506.5	1,218.7	54,053.4	56,685.6	58,973.4	10.22	5.84	2.03
famB 悪性 Std	2,740.5	459.0	465.0	107.1	2,768.0	2,782.7	2,740.1	0.88	0.82	0.19
famE 良性 Ave	552,375.2	43,788.1	29,871.8	25,118.2	508,587.1	522,503.3	527,257.0	13.74	9.41	7.75
famE 良性 Std	992,813.9	124,426.1	106,233.1	100,346.5	946,495.3	960,695.3	965,778.8	11.08	9.82	9.33
famE 悪性 Ave	87,533.3	7,193.4	4,585.8	2,847.0	80,339.9	82,947.5	84,686.3	7.97	4.91	2.87
famE 悪性 Std	13,784.3	5,834.4	5,809.0	5,790.7	11,662.5	11,681.9	11,674.4	5.51	5.65	5.77
famIH 良性 Ave	552,375.2	58,588.7	30,020.5	25,300.8	493,786.4	522,354.6	527,074.4	16.12	9.43	7.88
famIH 良性 Std	992,813.9	175,723.7	106,540.6	100,516.5	931,905.0	960,517.9	965,671.5	12.49	9.82	9.38
famIH 悪性 Ave	200,543.5	19,741.1	12,795.3	11,884.1	180,802.5	187,748.2	188,659.4	5.50	2.48	2.27
famIH 悪性 Std	200,423.5	90,547.1	82,433.3	81,224.2	158,943.9	170,034.4	173,432.9	8.50	8.06	7.99

表 8 最小サイズではない検体で n-gram 圧縮した場合のファイルサイズ
Table 8 Compressed file size by n-gram with another sample.

Name	Original	4gram	8gram	16gram	S4gram	S8gram	S16gram	R4gram	R8gram	R16gram
famA 良性 Ave	552,375.2	46,930.7	30,014.1	25,119.6	505,444.5	522,361.1	527,255.6	14.87	9.49	7.75
famA 良性 Std	992,813.9	128,190.3	106,248.5	100,357.4	943,834.2	960,634.5	965,776.8	11.30	9.83	9.32
famA 悪性 Ave	68,416.9	6,682.4	3,737.3	1,196.0	61,734.4	64,679.5	67,220.9	10.12	5.66	1.81
famA 悪性 Std	14,160.4	484.1	497.7	29.2	1,4152.1	14,149.8	14,173.4	1.84	1.18	0.33
famB 良性 Ave	552,375.2	48,806.1	30,109.9	25,111.3	503,569.1	522,265.2	527,263.8	15.13	9.47	7.73
famB 良性 Std	992,813.9	132,471.6	106,366.7	100,331.2	941,019.7	960,415.4	965,769.8	11.39	9.81	9.31
famB 悪性 Ave	60,188.6	6,536.4	3,867.8	1,596.7	53,652.3	56,320.9	58,591.9	10.85	6.41	2.63
famB 悪性 Std	2,740.2	1,355.2	1,355.2	1,302.5	2,682.9	2,690.7	2,649.1	2.08	2.10	2.04
famE 良性 Ave	552,375.2	47,350.6	29,810.5	25,103.6	505,024.5	522,564.7	527,271.5	14.62	9.40	7.74
famE 良性 Std	992,813.9	130,718.9	106,198.4	100,332.5	942,193.7	960,746.0	965,786.6	11.30	9.83	9.32
famE 悪性 Ave	87,526.8	7,230.5	4,583.7	2,834.7	80,296.4	82,943.2	84,692.2	8.02	4.92	2.87
famE 悪性 Std	13,799.6	5,840.7	5,811.2	5,791.0	11,698.7	11,724.2	11,719.6	5.67	5.82	5.93
famIH 良性 Ave	552,375.2	60,030.2	30,223.1	25,327.3	492,345.0	522,152.1	527,047.8	16.43	9.48	7.88
famIH 良性 Std	992,813.9	177,508.2	106,868.9	100,608.2	930,392.8	960,274.4	965,639.1	12.49	9.83	9.38
famIH 悪性 Ave	200,510.0	23,615.0	16,756.2	15,869.3	176,895.0	183,753.8	184,640.7	7.85	4.86	4.66
famIH 悪性 Std	200,445.4	92,213.7	8,4831.1	8,3678.2	159,797.8	170,948.7	174,384.2	13.07	13.29	13.28

表 5 では R4gram～R16gram の Std が 0.31～1.69 の範囲であるが，表 6 ではそれらの値が 10.32～10.65 となっている．つまり，最小サイズの検体によってはファイル圧縮率のばらつきが大きくなり，つねに同じ割合で圧縮可能なことが保証されているわけではないことが分かった．

また，表 7 および表 8 では，n-gram 圧縮の場合についての結果を示した．表 5 および表 6 の R4gram では，Ave の値が 10 を超えるものはない．一方，表 7 および表 8 では，Ave の値が 10 を超えるものが 6 個ずつ出現している．機械学習においては，ファイルサイズは訓練のコストに大きく影響する．とくに，我々の環境では，n-gram 圧縮ではファイルサイズが大きすぎて訓練を行うことができなかった．

5.3 機械学習による分類精度評価

3.4 節で述べた手法に従い，CNN を用いて評価を行った．実装に関しては 4 章で述べたとおりである．

本提案手法では，4gram，8gram，16gram の 3 パターンの n-gram を用いて評価実験を行った．これらは，それぞれ 4 バイト，8 バイト，16 バイト単位で n-gram を行うことに相当する．Windows アプリケーションは，64 bit 命令と 32 bit 命令のものが存在し，現在の主流である．本手法では，64 bit であれば 16 バイト，32 bit であれば 8 バイトであるため，これらの n-gram を採用した．なお，4gram も用いた理由は，32 bit 命令の前半 16 bit が演算内容を表し，後半 16 bit がアドレスを表すことがあるためである．

なお，ファミリ E のみ圧縮後のファイルの中で最大サイ

ズのものに合わせてゼロパディングすると，我々の環境では Numpy のメモリ不足でプログラムが動作しなかった．4gram，8gram，16gram とも，極端に大きいファイルが 2 つあり，これが原因であった．そこで，ファミリ E のみ，3 番目に大きいファイルサイズを最大値とし，それより大きい 2 つのファイルは最大値を超えるバイトを切り捨てて使用した．

最小サイズの比較用の検体を用いて n-gram 抽出を行った場合は次のとおりである．

4gram のファミリ E の抽出後最大ファイルサイズおよびそれに近いものは以下のとおり（悪性の 2 サンプル）である．

「63c8...da99」49,391 バイト

「a5df...69d5」49,386 バイト

次に小さいのは，以下の良性の 1 サンプルであるため，これを用いた．

「migrate.exe」8,427 バイト

8gram のファミリ E の抽出後最大ファイルサイズおよびそれに近いものは以下のとおり（悪性の 2 サンプル）である．

「63c8...da99」49,420 バイト

「a5df...69d5」49,410 バイト

次に小さいのは，以下の良性の 1 サンプルであるため，これを用いた．

「nltest.exe」9,574 バイト

16gram のファミリ E の抽出後最大ファイルサイズおよびそれに近いものは以下のとおり（悪性の 2 サンプル）である．

「a5df...69d5」49,470 バイト

「63c8...da99」49,466 バイト

次に小さいのは，以下の良性の 1 サンプルであるため，これを用いた．

「Ge2Gpx.exe」4,465 バイト

最小サイズではない比較用の検体を用いて n-gram 抽出を行った場合は次のとおりである．

4gram のファミリ E の抽出後最大ファイルサイズおよびそれに近いものは以下のとおり（悪性の 2 サンプル）である．

「7173...fe90」49,217 バイト

「a5df...69d5」49,212 バイト

次に小さいのは，以下の良性の 1 サンプルであるため，これを用いた．

「migrate.exe」6,932 バイト

8gram のファミリ E の抽出後最大ファイルサイズおよびそれに近いものは以下のとおり（悪性の 2 サンプル）である．

「7173...fe90」49,265 バイト

「a5df...69d5」49,252 バイト

次に小さいのは，以下の良性の 1 サンプルであるため，これを用いた．

「nltest.exe」8,683 バイト

16gram のファミリ E の抽出後最大ファイルサイズおよびそれに近いものは以下のとおり（悪性の 2 サンプル）である．

「7173...fe90」49,314 バイト

「a5df...69d5」49,312 バイト

次に小さいのは，以下の良性の 1 サンプルであるため，これを用いた．

「WmiApSrv.exe」3,566 バイト

CNN による分類結果を**表 9**，**表 10**，**表 11**，**表 12**，**表 13**，**表 14** に示す．10 分割交差検証の偏差に関しては，誤分類が各試行ごと数個にとどまり，その個数とファイル名もすべて列挙しているため，表中への記載を省略した．なお，Accuracy などの値を小数点第 4 位まで掲示しているのは，小数点第 3 位以内で四捨五入してしまうと 1.0 になってしまう場合があり，誤分類があったかどうか分からなくなってしまうためである．

表 9 では，ファミリ E の誤分類は，FP が 1 つのみで，

表 9 最小サイズの検体で n-gram 抽出した場合の CNN 結果（4gram）

Table 9 CNN result by n-gram extraction with smallest sample (4gram).

名前	Accuracy	Precision	Recall	F 値
ファミリ A	1.0000	1.0000	1.0000	1.0000
ファミリ B	1.0000	1.0000	1.0000	1.0000
ファミリ E	0.9997	0.9993	1.0000	0.9997
ファミリ IH	1.0000	1.0000	1.0000	1.0000

表 10 最小サイズの検体で n-gram 抽出した場合の CNN 結果（8gram）

Table 10 CNN result by n-gram extraction with smallest sample (8gram).

名前	Accuracy	Precision	Recall	F 値
ファミリ A	1.0000	1.0000	1.0000	1.0000
ファミリ B	1.0000	1.0000	1.0000	1.0000
ファミリ E	1.0000	1.0000	1.0000	1.0000
ファミリ IH	0.9972	0.9913	1.0000	0.9956

表 11 最小サイズの検体で n-gram 抽出した場合の CNN 結果（16gram）

Table 11 CNN result by n-gram extraction with smallest sample (16gram).

名前	Accuracy	Precision	Recall	F 値
ファミリ A	1.0000	1.0000	1.0000	1.0000
ファミリ B	1.0000	1.0000	1.0000	1.0000
ファミリ E	0.9990	0.9993	0.9987	0.9990
ファミリ IH	0.9982	0.9956	0.9985	0.9970

表 12 最小サイズではない検体で n-gram 抽出した場合の CNN 結果（4gram）
Table 12 CNN result by n-gram extraction with another sample (4gram).

名前	Accuracy	Precision	Recall	F 値
ファミリ A	1.0000	1.0000	1.0000	1.0000
ファミリ B	—	—	—	—
ファミリ E	1.0000	1.0000	1.0000	1.0000
ファミリ IH	0.9977	0.9971	0.9955	0.9963

表 13 最小サイズではない検体で n-gram 抽出した場合の CNN 結果（8gram）
Table 13 CNN result by n-gram extraction with another sample (8gram).

名前	Accuracy	Precision	Recall	F 値
ファミリ A	1.0000	1.0000	1.0000	1.0000
ファミリ B	1.0000	1.0000	1.0000	1.0000
ファミリ E	1.0000	1.0000	1.0000	1.0000
ファミリ IH	0.9981	0.9956	0.9985	0.9970

表 14 最小サイズではない検体で n-gram 抽出した場合の CNN 結果（16gram）
Table 14 CNN result by n-gram extraction with another sample (16gram).

名前	Accuracy	Precision	Recall	F 値
ファミリ A	1.0000	1.0000	1.0000	1.0000
ファミリ B	0.9997	0.9993	1.0000	0.9997
ファミリ E	0.9990	0.9980	1.0000	0.9990
ファミリ IH	0.9982	0.9956	0.9985	0.9970

その検体は vsinstr.exe であった．

表 10 では，ファミリ IH の誤分類に関しては，10 分割交差検証の 10 回の試行で，FP が 1 個，2 個，3 個出現している箇所があった．それらの検体は GTA_5_Mod_Menu.exe，FreeVK.exe，iview442_setup.exe，GPSmapview.exe，Install_Flash_Cookie_Cleaner.exe，FreeClipViewer.exe であった．

表 11 では，ファミリ E の誤分類に関しては，10 分割交差検証の 10 回の試行で，FP が 1 個，FN が 1 個と 1 個出現している箇所があった．FP となった検体は TlbExp.ni.exe であり，FN となった検体は ‘bba4...15b4’ と ‘d5c3...bf6d’ であった．ファミリ IH の誤分類に関しては，10 分割交差検証の 10 回の試行で，FP が 1 個と 1 個と 1 個，FN が 1 個出現している箇所があった．FP となった検体は GTA_5_Mod_Menu.exe，PresentationFontCache.ni.exe，FreeVK.exe であり，FN となった検体は ‘ff10...8c8c’ であった．

表 12 では，ファミリ B は，n-gram 抽出後のファイルサイズが大きく，我々の環境では 10 分割交差検証の 10 回とも Memory Error となって試行できなかった．ファミリ IH の誤分類に関しては，10 分割交差検証の 10 回の試行で，FP が 2 個，FN が 1 個と 1 個と 1 個出現している箇所があった．FP となった検体は FreeClipViewer.exe と iview442_setup.exe であり，FN となった検体は ‘510c...52a2’，‘c13a...9461’，‘b9b5...4f47’ であった．

表 13 では，ファミリ IH の誤分類に関しては，10 分割交差検証の 10 回の試行で，FP が 1 個と 1 個と 1 個，FN が 1 個出現している箇所があった．FP となった検体は FreeVK.exe，Install_Flash_Cookie_Cleaner.exe，iview442_setup.exe であり，FN となった検体は ‘ee72...0256’ であった．

表 14 では，ファミリ B の誤分類は，FP が 1 つのみで，その検体は x64launcher.exe であった．ファミリ E の誤分類に関しては，10 分割交差検証の 10 回の試行で，FP が 3 個出現している箇所があった．FP となった検体は FEP.exe，INTUNE.exe，SCEP.exe であった．ファミリ IH の誤分類に関しては，10 分割交差検証の 10 回の試行で，FP が 1 個と 2 個，FN が 1 個出現している箇所があった．FP となった検体は PresentationFontCache.ni.exe，GTA_5_Mod_menu.exe，LicenseManagerShellext.exe であり，FN となった検体は ‘ad87...b0b3’ であった．

6. 考察

6.1 提案手法の特徴に関する考察

3.2 節で，提案手法の 5 つの特徴について述べた．これらの特徴について考察する．

まず，特徴 1 であるが，本手法では，比較用検体に含まれるすべての n-gram に一致する目標ファイルの n-gram を抽出するため，同一の n-gram が複数回登場した場合でも，その出現順序と回数が保存される．共起度に関しては，ある n-gram と別のある n-gram が同時に出現する可能性という意味では，Reddy らや Boujnouni らの手法でも保存されている．しかし，ある n-gram と別のある n-gram が近い範囲に同時に出現する可能性という点においては，Reddy らや Boujnouni らの手法では出現順序や位置が考慮されないため保存されない一方，本手法では保存される．これは，良性ソフトウェアに含まれる命令セットや API コールのみを使用してマルウェアが作成されているような場合に有効である．

特徴 2 について，本手法はマルウェアについても良性ソフトウェアについても，データ全体を使用している．n-gram 抽出が行われるため，データのすべてが保存されるわけではないが，特定のセクションのデータのみが欠落するということはなく，そこにマルウェアの強い特徴があればそのデータを使用できる．

特徴 3 について，EMM 論文で提案した n-gram 圧縮では，我々の環境で動作させられる程度に，機械学習のネットワークに入力するデータ量を削減できなかったが，提案手法の n-gram 抽出では同様の環境で機械学習可能な程度にデータ量を削減できた．なお，Yakura らは全データを使

用して機械学習が行えており，機械学習を行うためにデータ量の削減は絶対条件ではないが，訓練コストを下げられるため望ましい．

特徴 4 について，本手法では，特定の n-gram の出現頻度が n-gram 抽出結果に影響しないため問題はない．機械学習においては，マルウェアにのみ出現する n-gram がより大きい影響力を持つように訓練されていく．

特徴 5 について，本手法は動的解析を必要としていないため，特定の時刻にならないと悪意のある振舞いをしないようなマルウェアがあっても問題はない．

6.2 評価結果に関する考察

表 9〜表 11 より，提案手法に従って，最小サイズの検体を比較用検体として n-gram 抽出を行った場合，4gram と 8gram の場合には FN はなかった．この結果だけを見れば，4gram と 8gram を使用すれば，提案手法では亜種マルウェアを検出し逃すことがなく，FP も 4gram では 1 個であるため，ほぼ完璧といえる．

しかし，実際には，捕獲された亜種マルウェアのうち，どれが最小サイズになるかは分からない．そこで，比較用検体を最小サイズの検体ではないものにした場合の評価を行った．もちろん，捕獲されたものの中では最小サイズの検体を選択するため，2 番目のサイズを選択した．結果としては，表 12〜表 14 より，4gram のファミリ IH に FN が計 3 個出現している．つまり，表 9 の結果がほぼ完璧であるのは偶然といえる．

また，表 9〜表 14 より，FP や FN となった検体はつねに同じではなかった．iview442_setup.exe や FreeClipViewer.exe のように，複数回 FP となったものもあるが，必ず FP とはなっていない．FN となった亜種マルウェアも異なる検体である．

5.3 節で，ファミリ E のみ，n-gram 抽出後に他の検体と比べてファイルサイズが 10 倍以上大きい 2 検体があった．当初は，データセットに誤りがあり，この 2 検体のみ他のアルゴリズムにより生成された別ファミリの亜種マルウェアではないかと考えたが，最小サイズの比較用検体を用いて n-gram 抽出を行った場合と最小サイズではない検体の場合とで「a5df...69d5」は共通，「63c8...da99」と「7173...fe90」は異なる検体であるため，そうではないことが分かる．これも偶然，2 検体のみが他の検体より 10 倍以上大きいことになる．なお，これらの大きい検体は，CNN において誤分類されていないので，切り捨てた超過分のバイトがなくても，亜種マルウェアとしての特徴を保っていたことになる．

次に，n-gram 圧縮と n-gram 抽出の差について考察する．表 5 に，最小サイズの検体で n-gram 抽出した場合のファイルサイズがあるが，たとえばファミリ IH の 16gram であれば，悪性で平均 0.87%，良性で平均 1.57%までファイルサイズが縮小される．一方，表 7 より，最小サイズの検体で n-gram 圧縮した場合のファイルサイズに関しては，ファミリ IH の 16gram であれば，悪性で平均 2.27%，良性で平均 7.88%の圧縮率となる．良性のほうが悪性よりも差が顕著になるのは，良性ソフトウェアには比較用検体となっている亜種マルウェアと類似のコードが多くは含まれていないためと考えられる．なお，n-gram 圧縮したサンプルについては，ファイルサイズが大きすぎて我々の環境では CNN を実行することができなかったため，CNN の評価は行っていない．しかし，n-gram 圧縮のほうが n-gram 抽出よりも多くのコードが含まれるため，分類精度はよりよくなると考えるのが妥当であろう．

6.3 データセットの Correctness, Transparency, Realism, Safety

本節では，評価に使用したデータセットの Correctness, Transparency, Realism, Safety について，Rossow らが論文中で定義したもの [19] に基づいて考察する．

6.3.1 データセットの Correctness

1) Check if goodware samples should be removed from datasets.

投稿された検体の振舞いは，悪意があるというよりむしろ不明である部分が残っていることを示唆しており，良性サンプルが混在しないような手段を研究者が用いることを提唱している．VirusShare.com [*1] が提供する最新のマルウェアサンプルセットなどは，その中の検体を VirusTotal で調べると，多くのマルウェア検出エンジン（以下検出エンジン）で undetected と表示される．つまり，あるマルウェア検出手法に対して，最新のデータセットでも同様の結果が得られるかどうかという興味に基づき，たとえば，「2019 年以降に発見された検体で追加評価せよ」のような指示は危険であるということである．多くの検出エンジンがマルウェアであると判定したようなサンプル，つまり発見から一定程度時間が経過した著名なサンプルを使うことが望ましい．

我々が本論文で使用した，a，b，e，ih のサンプルは，実験に必要なサンプル数が確保されている点，発見から十分な時間が経過し，数多くの検出エンジンで亜種マルウェアであると判定されている点から，良性ソフトウェアが混在している可能性が非常に少ない「正しいデータセット」であるといえる．ただし，CCC Dataset はハニーポットでマルウェアを収集したものであるため，標的型攻撃で狙われ，一般的には知られていないマルウェアを偶然大量に収集した可能性がある．そこで，VirusShare.com の検索機能で，これらの検体がいくつ見つかるか調べた．VirusShare.com の検索機能の都合により，つねに 20 の検体が発見されると次を調べるようになっており，全部でいくつ発見される

[*1] https://virusshare.com/

のかを調べることはできない．しかし，a，b，e，ih のいずれも，200 以上の検体があることが分かった．

2) Balance datasets over malware families.

積極的にポリモーフィックマルウェアファミリを取り入れたデータセットは，バランスがとれていないことを示唆している．亜種マルウェアも含めてマルウェアを集めた場合，場合によっては同一ファミリの亜種マルウェアがデータセットの大部分を占めてしまい，それによってマルウェア全体を検出できる割合を評価することは不適切である．一方，我々の評価では，そもそもが特定の亜種マルウェアファミリを検出できるかどうかが目的であるため，この点に問題はない．

3) Check whether training and evaluation datasets should have distinct families.

1 つのマルウェアファミリが訓練と検証の両方に入る場合について言及している．なお，一方的に問題があるとしているわけではなく，ジェネリックのマルウェアを検出するような場合においては，このほうが望ましいとする一方，未知マルウェア検出を目的とする場合にはファミリごとに訓練と検証でセットを分けるべきだと述べている．本論文の目的は，同一ファミリの亜種マルウェアを検出するものであるため，訓練と検証の両方に同一ファミリの亜種マルウェアが入っているが問題はない．

4) Perform analysis with higher privileges than the malware's.

ルートキットの機能を持つマルウェアは，それを監視するしくみと干渉する場合があるため，マルウェアサンプルとモニタリング機構が衝突する範囲について著者らが報告すべきであると述べている．この問題は，マルウェアの命令セットや API を利用するタイプの検出手法においては当てはまるが，我々の提案手法のように，マルウェアのバイナリ列のみを扱うものには該当しない．

5) Discuss and if necessary mitigate analysis artifacts and biases.

ユーザ名や OS シリアルキーのような特定の文字列の存在や，解析環境のソフトウェアコンフィギュレーションがマルウェアの振舞いの記録に影響を与える点について言及されている．我々の提案手法はマルウェアの実行を行わないため，この問題には該当しない．

6) Use caution when blending malware activity traces into benign background activity.

マルウェアサンプルが動的解析環境で実行された場合と，実世界の犠牲者の機器で実行された場合とでは，振舞いに違いがある点が言及されている．我々の提案手法はマルウェアの実行を行わないため，この問題には該当しない．

6.3.2 Transparency

1) State family names of employed malware samples.

使用するマルウェアサンプルのファミリ名が明記されるべきであると述べられている．同一ではないマルウェアのサンプルが多数存在したとしても，バイナリレベルでは近いものが存在する可能性があるためである．なお，ラベリングされる際のマルウェアの名前については，読者がマルウェアの特性を特定するのに役立つとする一方，厄介な問題であるともしている．本論文の評価においては，亜種マルウェアのファミリ名を明記しており，他のファミリを混在させた評価を行わないため，この問題には該当しない．

2) List which malware was analyzed when.

実験を理解し再現するために，読者がマルウェアのリストを必要としていることが述べられている．マルウェアの中には短命なものもあり，実行されて観測された日にちを得られることは助けになるとされている．本論文で使用したマルウェアの検体は，CCC Dataset のものであり，読者が同じものを入手して実験を再現することが可能である．なお，MWS データセットを使用するにあたり，論文中で検体の完全なハッシュ値を公開してはいけないという取り決めがある．本論文には影響がないが，マルウェアを分類する手法によってはこの影響を受ける．

3) Explain the malware sample selection.

実験にはマルウェアサンプルのランダム選択のみが妥当であると述べられている．より最近の分析結果に注目したり，古いデータを無視したりすることは何らかの関連性を増加させてしまうとしている．ただし，Correctness の 2) で述べたファミリ間でバランスをとる必要があるため，ランダム選択さえすればバランスがとれるわけではないことも付記されている．本論文では，CCC Dataset から亜種マルウェアの 4 ファミリを使用する際，10 分割交差検証に区切りのよい数字となるよう，またシステムの都合で上限が 1,500 となるよう，検体を選択しているが，ランダム選択を行っており妥当であるといえる．なお，「2019 年以降に発見された検体」に固執するような指示は不適切であるということになる．

4) Mention the system used during execution.

マルウェアは様々なシステムやソフトウェアコンフィギュレーション，バージョンで実行されるので，実行するシステムを明記することが述べられている．たとえば，「Windows XP SP3 32 bit で追加のソフトウェアインストールなし」などと表記すると実験の透明性がより高まるとしている．本論文の提案手法では，マルウェアの実行を行わないためこの問題に該当しない．

5) Describe the network connectivity of the analysis environment.

マルウェアファミリは，システムの接続状態に応じて行動のロールを割り当てるとし，ネットワークへの接続状態が記録される振舞いに影響を与えると述べられている．本論文の提案手法では，マルウェアの実行を行わないためこの問題に該当しない．

6) Analyze the reasons for false positives and false negatives.

誤分類率単独では，システムのパフォーマンスに関してあまり明らかにならないとしている．何が誤分類を引き起こしたのかよく考えて調査することが推奨されており，そのような研究事例もあげられている．本論文では，同一亜種マルウェアファミリの検体を用いて評価を行っている．そのため，FN に関しては，誤分類を引き起こした原因は存在しない．つまり，機械学習における訓練が完璧ではなかったというだけのことである．一方，FP に関しては，その検体が何らかの特徴を持っているかもしれないため調査した．

その結果，VirusTotal の検出エンジンにおいて，FreeClip Viewer.exe，FreeVK.exe，GPSmapview.exe が SecureAge APEX で Malicious と，Install_Flash_Cookie_Cleaner.exe と iview442_setup.exe が 2 個の検出エンジンで悪性と，GTA_5_Mod_Menu.exe が 5 個の検出エンジンで悪性と診断された．つまり，誤分類された一部のファイルは，マルウェアに共通する何らかの要素を内包していると考えられる．

7) Analyze the nature/diversity of true positives.

True Positive 率単独では，手法の潜在性をしばしば適切に反映しないと述べられている．例として，何百もの感染したホストにフラグを立てるマルウェア検出器は有望であるようにみえるが，1 つのマルウェアファミリのみを検出していたり人工的な環境に依存したりしているのであればそうとはいえないとしている．正しい検出におけるマルウェアの多様性が求められているが，本論文では 1 つの訓練済みモデルにおいて検出対象とする亜種マルウェアファミリは 1 つであり，ここで述べられている多様性とは無縁である．

6.3.3 Realism

1) Evaluate relevant malware families.

非常に多くの一般的なマルウェアファミリを使うことは，実験結果をより強固なものにできると述べられている．マルウェアは日々進化しているので，古いもしくは欠陥のある検体を独占的に使用することで，関連性を弱められるとしている．本論文では 1 つの訓練済みモデルにおいて検出対象とする亜種マルウェアファミリは 1 つであるため，この問題は該当しない．

2) Perform real-world evaluations.

評価シナリオとしての実世界での実験は，著者のものよりもむしろ，他人によって実際に使われたかなり数のホストによるものだと定義されている．本論文で使用した CCC Dataset は，我々が設置した機器で集めたものではなく，他人のハニーポットによって集められたものであるため，この条件に合致している．ただし，いくつのハニーポットをどのように運用したかについては言及がなく不明である．

3) Exercise caution generalizing from a single OS version, such as Windows XP.

1 つの OS バージョンに限定して解析を行うと，他の OS で異なる振舞いをするマルウェアの場合には失敗すると述べられている．本論文の提案手法では，振舞い解析は行わないためこの問題は該当しない．

4) Choose appropriate malware stimuli.

キーロガーであれば，キーを押すという特定の刺激をトリガーにするように，一般的にはユーザのインタラクションが必要であると述べられている．著者らは，実験の解析期間が十分である理由を述べるべきであるとしている．本論文の提案手法では，振舞い解析は行わないためこの問題は該当しない．

5) Consider allowing Internet access to malware.

法律や倫理の問題を脇に置いて，マルウェアがインターネットにアクセスするのであれば，実験はより現実的である必要があることを議論している．本論文の提案手法では，振舞い解析は行わないためこの問題は該当しない．

6.3.4 Safety

1) Deploy and describe containment policies.

時間とともにマルウェアが他者に引き起こす潜在的な危害を緩和する一方，よくデザインされた封じ込めポリシは現実的な実験を促進すると述べられている．本論文の提案手法では，マルウェアの実行が必要ないためこの問題は該当しない．

6.4 新しいマルウェアデータセットでの確認

マルウェアは日々進化するため，本論文の提案手法が同様の有効性を保てなくなる可能性がある．そこで，新しいマルウェアでデータセットを作成し，追加実験を行った．

亜種マルウェアファミリのデータセットを作成するには，まず，亜種マルウェアファミリ名を取得する必要がある．そこで，岸波らの研究 [20] を参考にした．岸波らによると，Kaspersky はマルウェアの命名規則を “[Prefix:] Behaviour.Platform.Name[.Variant]” と定めているとのことである．具体的には，Prefix が検体が検知されたサブシステム名，Behaviour がマルウェアの種類，Platform が検体の実行環境，Name がマルウェア名，Variant が亜種名を表している．そこで，マルウェア検体を提供しているサイトから，この命名規則に従う新しい亜種マルウェアファミリを取得し，追加実験を行うこととした．岸波らは，マルウェア検体の MD5 ハッシュ値をオンラインマルウェア解析サービス VirusTotal [*2] に送信し，Kaspersky が定めた分類名を取得していたため，同様の作業を行った．

6.4.1 VirusShare からの亜種マルウェアファミリ取得

VirusShare は，マルウェアのアーカイブを提供している．

[*2] https://virustotal.com

アーカイブは 0 から 389 まで連番で存在する．VirusShare はマルウェア名による検索機能を備えているが，名前で検索した場合，60 秒以内に 20 個以上の検索候補が見つからないと，「Timeout: Search exceeded 60 seconds.」と表示されて 1 つも結果を得ることができないという特性がある．20 個以上見つかった場合，最初の 20 個が見つかった時点でそれらを表示し，「More Results」ボタンでそれ以降の候補を検索する．「More Results」ボタンを押した後も，60 秒以内に 20 個以上の次の検索候補が見つからないと，「Timeout: Search exceeded 60 seconds.」になる．

まず，アーカイブ 0 の先頭から 300 個のマルウェアについて手動で調べた．VirusTotal の Kaspersky の表記が，亜種マルウェアになっているものは，上記の 300 個中 216 個であった．なお，この 216 個の中に，Kaspersky の表記規則に従わない，「Virus.Win32.HLLW.Tefuss.d」と「Virus.DOS.SillyOC.186.b」があった．また，表記規則には従うが，亜種マルウェアのファミリ名ではないものが末尾に付記されていると思われる「Exploit.Win32.Nuker.Xobobus」があった．

この 216 個の中で，同一亜種マルウェアファミリとして最も多いものは「Backdoor.PHP.C99Shell.gm」の 13 個であった．ただし，これは PHP である．

4 個だったものは「Virus.Win32.Expiro.w」である．

2 個だったものは「Net-Worm.Win32.Allaple.e」，「Net-Worm.Win32.Padobot.bi」，「Trojan.NSIS.StartPage.ag」，「Trojan-Dropper.Win32.Agent.emlq」，「Trojan-PSW.Win32.Kykymber.leh」である．

それ以外はすべて 1 つずつしか存在しなかった．つまり，この方法で，なるべく多く見つかった亜種マルウェアファミリを，VirusShare で再度ファミリ名で検索し，データセットとして使用可能な検体を得るしかない．

なお，アーカイブ 0 は VirusShare の記述によると 2011 年 7 月頃から始まるものであった．VirusShare のアーカイブが古い順に連番となっていることに気付いたため，最新のアーカイブ 389 の先頭から 50 個のマルウェアについて手動で調べた．しかし，Kaspersky では 50 個中 33 個が Undetected となった．また，「Not-a-virus:」で始まるものも 4 個あった．亜種マルウェアの表記に従うものは 5 個しかなく，その中にはシェルスクリプトが 1 つ，VB スクリプトが 1 つあった．つまり，バイナリの亜種マルウェアは 3 つしかなく，すべて別のファミリである．なお，この 3 つに関しては，VirusShare の検索では，60 秒以内に 20 個以上見つからないため，「Timeout: Search exceeded 60 seconds.」となって 1 つも表示されない．

アーカイブ 360 あたりから 2019 年のものになるため，アーカイブ 360 の先頭から 50 個のマルウェアについて手動で調べた．50 個の内訳は，「Undetected」が 13 個，VB や JS などのスクリプトが 37 個であり，バイナリのマルウェアが存在しなかった．つまり，新しいマルウェアで同一亜種マルウェアファミリの検体を大量に用意することは非常に困難である．

調べた結果，一番新しそうなものとして，2018 年 1 月頃から出現している「Trojan.Win32.Agentb.bvrg」が見つかった．「TrID Win64 Executable (generic) (61.7%)」となっており，比較対象の候補としても適切である．

なお，「kaspersky:Trojan.Win32.Agentb.bvrg」では最初の 20 個しか表示されず，「More Results」後で時間切れになってしまう．やむを得ず，「kaspersky:Trojan.Win32.Agentb」で検索し，結果から「Trojan.Win32.Agentb.bvrg」を手動で抜き出した．それでも「Displaying results 141 to 160」が限界で，それ以上は候補が見つからず，検索できなくなった．ここまでに見つかったファイルは 30 個である．

なお，「kaspersky:Trojan.Win32.Agent」で検索すると 1,500 以上ヒットする．ただし，20 件ずつしか表示できない仕様は変わらず，20 件ずつ「Displaying results 1481 to 1500」まで手動で調べた．その結果，「kaspersky:Trojan.Win32.Agentb.bvrg」で調査したのと同じ検体の 30 個しか見つからなかった．そこで，この 30 個を追加実験の検体とした．以降，このファミリを本文中で bvrg と呼称する．bvrg 以外のファミリは，1,500 個までの検索で，検体が数個程度しか見つからなかった．

なお，bvrg の検体を VirusTotal に入力すると，「2020-08-22 04:04:39 UTC 1 year ago」のように表示される．VirusTotal のほうが VirusShare.com よりも登録が遅いようで，これであれば 2020 年に発見されたことになる．つまり，bvrg は 2018 年 1 月頃から 2020 年 8 月頃に現れたマルウェアと表現できる．余談だが，本論文で評価に用いたファミリ e も，VirusTotal で「2020-01-09 18:04:47 UTC 1 year ago」と表示された検体があったことを付記しておく．

6.4.2 MalShare からの亜種マルウェアファミリ取得

MalShare [*3] は，マルウェアサンプルの公開と検索機能を提供している．

VirusShare.com で見つけた「cfca...7b06」の MD5 ハッシュ値を持つ検体を，MalShare で検索してみた．表示された項目は，SHA256 Hash, File type（PE32 など），Added（日付と時刻），Source（URL），Yara Hits であった．Yara Hits には，「YRP/contentis_base64」のような値が並んでいる．つまり，この検体が何のマルウェアであるかは MalShare では分からない．MalShare に検索機能はあるが，Recent Searches に並ぶ項目は「yrp/shellbot_pl」のようになっており，MalShare の検索結果に表示される項目でしか検索できないことが分かった．実際に，各種マルウェア検出エンジンが検出名として表示する「Trojan」のような検索語を入力しても何もヒットしない．結論として，MalShare

[*3] https://malshare.com/

から特定の亜種マルウェアファミリの検体を得ることはできない．

6.4.3 Vx Underground からの亜種マルウェアファミリ取得

Vx Underground [*4]は，マルウェアの名前で分類されたサンプルを公開している．

Mydoom のような古いものから Mirai のような新しいものまで存在するが，1 マルウェア名あたりの検体数が非常に少ない．たとえば，2018 年頃の DanaBot は検体が 1 つしかなく，2019 年頃の Gafgyt も 17 個であり，他のものも数個〜十数個程度である．AgentTesla が 46 個あったが，2014 年のマルウェアである．多いものでは，Mirai が 86 個であった．しかし，ファイルサイズが 0 バイトの検体があることに気づき，調べてみることにした．

0 バイトの検体のハッシュ値を VirusTotal に入力したところ，Kaspersky で「HEUR:Trojan.Win32.Bingoml.gen」と表示された．0 バイトであるのもおかしいが，Mirai マルウェアなので Linux 用のものが Win32 のマルウェアとして検出されるのもおかしい．なお，VirusTotal の他のエンジンでも，Mirai を名前の文字列に含むものは存在しなかった．

同様に，別の検体のハッシュ値を VirusTotal に入力したところ，Kaspersky で「HEUR:Backdoor.Linux.Mirai.b」と表示されるものもあった．ただし，さらに別の検体は VirusTotal で「HEUR:Backdoor.Linux.Mirai.do」と表示され，同一亜種マルウェアファミリではない．

結論として，Vx Underground のマルウェアは，検体数が非常に少なく，分類も亜種マルウェアファミリごとになっておらず，本研究の評価には使用できない．また，ラベルが誤っているものも含まれていて信頼性にも欠ける．

6.4.4 新しい亜種マルウェアファミリによる評価

6.4.1 項の bvrg を用いて評価を行った．5.1 節で述べたように，比較用検体がテストサンプルの中に入るのは好ましくない．しかし，bvrg は全検体が 30 個しかなく，適切な機械学習のために 1 つでも訓練用サンプルを増やしたかったので，比較用検体も 10 分割交差検証に含めた．影響があるのは比較用検体がテストサンプルとなった 1 カ所のみであるため，精度を論じる際にはそこだけ除外すればよい．**表 15** に，bvrg を提案手法の 4-gram で分類した結果を示す．

i = 4 で正しく分類されたマルウェアが 1 つあるが「bfd6...78de」のため比較用検体である．i = 9 で正しく分類されたマルウェアが 1 つあり「94b2...93b8」であった．それ以外のマルウェアはすべて良性と判定された．

表 16 に，bvrg を提案手法の 8-gram で分類した結果を示す．

[*4] https://vx-underground.org/

表 15 CNN による評価結果 4gram（ファミリ bvrg）
Table 15 Evaluation results with 4gram by CNN (Family bvrg).

i	TP	FP	TN	FN
0	0	0	150	3
1	0	0	150	3
2	0	0	150	3
3	0	0	150	3
4	1	0	150	2
5	0	0	150	3
6	0	0	150	3
7	0	0	150	3
8	0	0	150	3
9	1	0	150	2

表 16 CNN による評価結果 8gram（ファミリ bvrg）
Table 16 Evaluation results with 8gram by CNN (Family bvrg).

i	TP	FP	TN	FN
0	0	0	150	3
1	0	0	150	3
2	1	0	150	2
3	1	0	150	2
4	0	0	150	3
5	0	0	150	3
6	0	0	150	3
7	0	0	150	3
8	0	0	150	3
9	0	0	150	3

表 17 CNN による評価結果 16gram（ファミリ bvrg）
Table 17 Evaluation results with 16gram by CNN (Family bvrg).

i	TP	FP	TN	FN
0	0	0	150	3
1	3	116	34	0
2	0	0	150	3
3	0	0	150	3
4	0	0	150	3
5	3	113	37	0
6	3	119	31	0
7	3	109	41	0
8	1	0	150	2
9	0	0	150	3

i = 2 で正しく分類されたマルウェアが 1 つあるが「bfd6...78de」のため比較用検体である．i = 3 で正しく分類されたマルウェアが 1 つあり「94b2...93b8」であった．それ以外のマルウェアはすべて良性と判定された．

表 17 に，bvrg を提案手法の 16-gram で分類した結果を示す．

i = 8 で正しく分類されたマルウェアが 1 つあり「94b2...93b8」であった．i = 1，5，6，7 の回を除き，それ

以外のマルウェアはすべて良性と判定された．

n-gram 抽出後のファイルサイズを調べてみると，bvrg 30 個のマルウェア全検体中，4-gram に 97 バイトのものが 25 個，8-gram に 60 バイトのものが 26 個，16-gram に 0 バイトのものが 26 個見つかった．同一亜種マルウェアファミリであるのに 0 バイトとなるのは明らかにおかしい．また，前述の 4-gram の 25 個は，8-gram と 16-gram の 26 個に完全に含まれる．

そこで bvrg 30 個のオリジナルのファイルサイズを調査した．結果を**表 18** に示す．

表 18 より，「bfd6...78de」と「94b2...93b8」は明らかに他の検体とはファイルサイズが異なり，この 2 つは類似している．この 2 つのみが同一亜種マルウェアファミリだとすると，表 15 や表 16 の結果は完全に正しいことになる．そこで，表 18 の「その他 24 個すべて」に含まれる「42ed...3521」を比較用検体として 4-gram のみやり直して確認した．なお，n-gram 抽出後の最大サイズの検体に合わせて 0 パディングすると，5.3 節の実験同様，我々の環境では機械学習が行えないため，最大サイズを 10,000 バイトにした．

結果を**表 19** に示す．

表 19 で，誤分類したのは「bf3e...5963」，「94b2...93b8」，「e166...c5cb」，「bfd6...78de」，「6e18...fc47」の 5 個である．これらはすべて表 18 で，ファイルサイズが極端に大きいか小さいものの 5 個であり，そもそも同一亜種マルウェアファミリではないと思われる．極端に小さいものに関しては，その検体で n-gram 抽出を行うと 0 バイトになる検体が多数出現することから同一亜種マルウェアファミリではないことは確定であり，極端に大きいものも，10 倍程度ファイルサイズに差があるというのも不自然である．ラベルが誤っているのあれば，本手法は新しい亜種マルウェアファミリに対しても，4-gram によって完全に分類可能であることが示されたことになる．

表 18 ファミリ bvrg のファイルサイズと検体名
Table 18 File size and name of specimen in family bvrg.

サイズ（byte）	検体名
52,408	bfd6...78de
52,416	94b2...93b8
106,496	その他 24 個すべて
124,748	7432...c859
1,018,368	e166...c5cb
1,018,368	bf3e...5963
1,327,104	6e18...fc47

表 19 CNN による評価結果 4gram（ファミリ bvrg，確認）
Table 19 Evaluation results with 4gram by CNN (Family bvrg, confirmation).

i	TP	FP	TN	FN
0	2	0	150	1
1	3	0	150	0
2	3	0	150	0
3	3	0	150	0
4	1	0	150	2
5	3	0	150	0
6	3	0	150	0
7	1	0	150	2
8	3	0	150	0
9	3	0	150	0

なお，表 17 で FP も FN も多数になったのは，n-gram 抽出後の良性ソフトウェアに，0 バイトとなったサンプルが 1,151 個あったためである．0 バイトのサンプルが悪性と良性に多数存在したため，機械学習が適切に行えなかったのは当然といえる．

大手のマルウェア検出エンジンでラベルを誤る理由に関しては，新しいマルウェアの場合，解析のための情報が十分に集まっていないことが考えられる．実際に，6.4.3 項で述べたように，Vx Underground の Mirai マルウェアの中に「HEUR:Trojan.Win32.Bingoml.gen」が紛れていたこともあり，検出エンジンが更新されることによって，新しいマルウェアを徐々に正確に分類できるようになっていくものと思われる．

追加実験のデータセットの Correctness，Transparency，Realism，Safety については，基本的に a，b，e，ih のファミリによる実験のものと同様であるため，差分のみ考察する．Correctness の 1) Check if goodware samples should be removed from datasets に関しては，VirusTotal の結果が undetected ではないため，良性ソフトウェアの混在はないと思われる．Transparency の 2) List which malware was analyzed when に関しては，VirusShare から取得した同一亜種ファミリ名のすべての検体を使用しており，読者が完全に同一の検体を取得できる．Transparency の 3) Explain the malware sample selection に関しては，古い a，b，e，ih のファミリだけでなく，新しい bvrg も追加で評価したため，より適切になったといえる．Transparency の 6) Analyze the reasons for false positives and false negatives に関しては，誤分類が起きた理由を考察し，検出エンジンに誤りがなければ完全に分類ができることを示せたので，適切といえる．Realism の 2) Perform real-world evaluations に関しては，他人が集めたものを VirusShare から取得したため適切といえる．

7. まとめ

本論文では，関連研究の問題点を指摘し，対象を亜種マルウェアファミリに限定することにより，n-gram 抽出と機械学習を用いて，関連研究の問題点を解決した亜種マルウェア検出手法を提案した．

4 つのファミリからなる亜種マルウェアの実際の検体を

用いて評価した結果，最小サイズの検体で n-gram 抽出した場合のファイルサイズは，16gram を用いた場合の平均で，ファミリ A 良性が 1.09%，ファミリ A 悪性が 1.26%，ファミリ B 良性が 1.06%，ファミリ B 悪性が 1.41%，ファミリ E 良性が 2.32%，ファミリ E 悪性が 1.38%，ファミリ IH 良性が 1.57%，ファミリ IH 悪性が 0.87%と約 40 分の 1〜110 分の 1 程度に縮小させることに成功した．しかも，その縮小したファイルサイズの状態で CNN による機械学習を用いれば，4gram の場合にはファミリ E に FP が 1 つのみ，16gram でもファミリ E に FP が 1 つと FN が 2 つ，ファミリ IH に FP が 3 つと FN が 1 つ現れるのみであった．比較用検体をファイルサイズの異なるものに変更した場合についても評価したが，誤分類の違いは数個程度にとどまった．

追加実験を行ったファミリ bvrg に関しても，一部の検体に亜種マルウェアファミリ名が誤ってラベリングされていることを発見しただけでなく，適切なラベルに基づくと完全に分類が行えることを示した．

本論文の目的は，亜種マルウェアを最高精度で検出することではないため，CNN より優れたアルゴリズムを試したり，最適なネットワークやパラメータを探したりはしていない．n-gram の情報利得を使用しない本手法は，情報利得が低い n-gram のみでマルウェアを作成されたような場合でも，n-gram の出現順序，回数，共起度を分類に使用できる点が特徴である．また，マルウェアの実行も必要なく，良性ソフトウェアに多く含まれるバイナリの追加にも耐性がある．さらに，コードセクションのような特定のセクションに依存しないため，どのセクションのデータからも特徴を得られる．

謝辞 本研究は JSPS 科研費 JP18K11248 の助成を受けたものです．また，本研究にはマルウェア対策のための研究用データセットである MWS データセットを使用しています．

参考文献

[1] Kephart, J.O, Sorkin, G.B., Arnold, W.C., Chess, D.M., Tesauro, G.J. and White, S.R.: Biologically Inspired Defenses Against Computer Viruses, *Proc. 14th International Joint Conference on Artificial Intelligence* (*IJCAI*), pp.985–996 (1995).

[2] Abou-Assaleh, T., Cercone, N., Keselj, V. and Sweidan, R.: N-gram-based Detection of New Malicious Code, *Proc. 28th Annual International Computer Software and Applications Conference* (*COMPSAC*), Vol.2, pp.41–42 (2004).

[3] Santos, I., Penya, Y.K., Devesa, J. and Bringas, P.G.: N-grams-based File Signatures for Malware Detection, *Proc. 11th International Conference on Enterprise Information Systems* (2009).

[4] Reddy, D.K.S. and Pujari, A.K.: N-gram Analysis for Computer Virus Detection, *Journal in Computer Virology*, Vol.2, No.3, pp.231–239, Springer (2006).

[5] Boujnouni, M.E., Jedra, M. and Zahid, N.: New Malware Detection Framework Based on N-grams and Support Vector Domain Description, *Proc. 11th International Conference on Information Assurance and Security* (*IAS*), pp.123–128 (2015).

[6] Kolter, J.Z. and Maloof, M.A.: Learning to Detect and Classify Malicious Executables in the Wild, *Journal of Machine Learning Research*, Vol.7, pp.2721–2744 (2006).

[7] Fuyong, Z. and Tiezhu, Z.: Malware Detection and Classification Based on n-grams Attribute Similarity, *Proc. 2017 IEEE International Conference on Computational Science and Engineering* (*CSE*) *and IEEE International Conference on Embedded and Ubiquitous Computing* (*EUC*), pp.793–796 (2017).

[8] Raff, E., Zak, R., Cox, R., Sylvester, J., Yacci, P., Ward, R., Tracy, A., McLean, M. and Nicholas, C.: An Investigation of Byte N-gram Features for Malware Classification, *Journal of Computer Virology and Hacking Techniques*, Vol.14, pp.1–20 (2018).

[9] Raff, E. and Nicholas, C.: Hash-Grams: Faster N-Gram Features for Classification and Malware Detection, *Proc. ACM Symposium on Document Engineering* (*DocEng*) *2018* (2018).

[10] O'Kane, P., Sezer, S. and McLaughlin, K.: N-gram Density Based Malware Detection, *Proc. World Symposium on Computer Applications & Research* (*WSCAR*) (2014).

[11] Mira, F., Huang, W. and Brown, A.: Improving Malware Detection Time by Using RLE and N-gram, *Proc. 2017 23rd International Conference on Automation and Computing* (*ICAC*), pp.1–5 (2017).

[12] Lee, T., Choi, B., Shin, Y. and Kwak, J.: Automatic Malware Mutant Detection and Group Classification Based on The n-gram and Clustering Coefficient, *The Journal of Supercomputing*, Vol.74, pp.3489–3503 (2018).

[13] Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y. and Sakuma, J.: Malware Analysis of Imaged Binary Samples by Convolutional Neural Network with Attention Mechanism, *Proc. 8th ACM Conference on Data and Application Security and Privacy* (*CODASPY*), pp.127–134 (2018).

[14] Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y. and Sakuma, J.: Neural Malware Analysis with Attention Mechanism, *Computers & Security*, Vol.87 (2019).

[15] McLaughlin, N., del Rincon, J.M., Kang, B., Yerima, S., Miller, P., Sezer, S., Safaei, Y., Trickel, E., Zhao, Z., Doupe, A. and Ahn, G.J.: Deep Android Malware Detection, *Proc. 7th ACM Conference on Data and Application Security and Privacy* (*CODASPY*), pp.301–308 (2017).

[16] 瀧口翔貴，宇田隆哉：N-gram 圧縮と深層学習を用いたマルウェア分類手法の提案，電子情報通信学会技術研究報告，Vol.118, No.494, EMM2018-112, pp.111–116 (2019).

[17] 高田雄太，寺田真敏，村上純一，笠間貴弘，吉岡克成，畑田充弘：マルウェア対策のための研究用データセット—MWS Datasets 2016，情報処理学会研究報告，Vol.2016-CSEC-74, No.17, pp.1–8 (2016).

[18] Windows PC Software - Free Downloads and Reviews, available from ⟨http://download.cnet.com/windows/⟩ (accessed 2017-08).

[19] Rossow, C., Dietrich, C.J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., Bos, H. and van Steen, M.: Prudent Practices for Designing Malware Experiments:

Status Quo and Outlook, *Proc. 2012 IEEE Symposium on Security and Privacy*, pp.65–79 (2012).
[20] 岸波敬介，梅澤 猛，大澤範高：ネットワークの分散表現学習に基づく亜種マルウェアの活動検知，情報処理学会研究報告，Vol.2020-CSEC-91, No.24, pp.1–8 (2020).

瀧口 翔貴

2019 年東京工科大学コンピュータサイエンス学部コンピュータサイエンス学科卒業．現在，SB テクノロジー勤務．

宇田 隆哉 （正会員）

1998 年慶應義塾大学理工学部計測工学科卒業．2000 年同大学大学院理工学研究科計測工学専攻前期博士課程修了．2002 年同大学院理工学研究科開放環境科学専攻後期博士課程修了．博士（工学）．現在，東京工科大学コンピュータサイエンス学部講師．ネットワークセキュリティの研究に従事．2002 年 IFIP/SEC 2002 Best Student Paper Award 受賞．電子情報通信学会会員．