Vol.2021-CSEC-95 No.10
Vol.2021-SPT-45 No.10
Vol.2021-EIP-94 No.10
2021/11/8

APIコール情報を用いた注意機構付きLSTMによるマルウェアの特徴抽出と分類

大江 弘晃[1,a)]　毛利 公一[1]　鄭 俊俊[1]

概要：機械学習を用いた様々なマルウェア分類手法が提案されている．しかし，これらの分類手法のほとんどは，分類精度のみに着目したものであり，高い分類性能を実現するための効果的な特徴抽出についてはあまり着目されていない．そこで，本稿では画像やテキストの分類において，注目点の可視化に利用されるニューラルネットワーク技術の一つである注意機構（Attention mechanism）を用いて，マルウェア分類精度の向上に寄与する特徴抽出の可能性について述べる．具体的には，重要な特徴を明らかにし，それにより高精度な分類を実現する注意機構付き LSTM (Long Short-Term Memory) を提案する．本稿では，特に，Windows API の関数名および引数の時系列データを用いた分類を行い，注意機構の有無による分類精度の比較を行う．また，注意機構から抽出した重み値から重要度が高いと判断された特徴が，対象のファミリに由来する特徴かを確認する．

キーワード：マルウェア分類， LSTM (Long short-term memory)， 注意機構， API コール，特徴抽出

Malware Classification and Feature Extraction Using Attention-Based LSTM and API Call Information

Hiroaki OE[1,a)]　Koichi MOURI[1]　Junjun ZHENG[1]

Abstract: Various malware classification methods using machine learning have been proposed. However, Almost all the proposed classification methods focused on only the classification accuracy, and didn't pay much attention to the effective feature extraction achieving high classification performance. Therefore, in this paper, we attempt to identify important features contributing more to the malware classification accuracy using attention mechanism, which is one of the neural network technologies widely adopted to visualize the points of interest in image and text classification. Specifically, we propose an attention-based long short-term memory (LSTM) method to reveal important features and thereby achieve a high-accuracy classification. In this paper, in particular, we use the time series data of Windows API calls, containing both function names and arguments, and compare the classification accuracies with and without the attention mechanism. In addition, the weight values extracted from the attention mechanism confirm whether the features identified as being of high importance originate from the target family.

Keywords: Malware classification, LSTM (Long short-term memory), attention mechanism, API Call, feature extraction

1. はじめに

近年，マルウェアが急増しており，脅威の拡大が深刻な社会問題となっている．McAfee 社によると，2021 年の第 1 四半期までに検出された新種のマルウェアは 8.7 千万件，マルウェアの総数は 15 億件を超えることが報告されている [1]．このようなマルウェア増加の主な要因として，マルウェアの亜種の増加が挙げられる．マルウェアの亜種は，オリジナルとなったマルウェアの改変，または機能の追

[1] 立命館大学
Ritsumeikan University
[a)] hohe@asl.cs.ritsumei.ac.jp

加・削除を行った検体であるため，僅かな変更で作成できる．また，2007 年頃から出現したクライムウェアキットである ZeuS [2] や，Android 向け TDK アプリ [3] などの出現によるマルウェア作成の容易化も，マルウェアの亜種の増加に拍車をかけている．マルウェアの亜種の多くは，シグネチャとの照合によるパターンマッチング検出を回避する．これはマルウェアの亜種が作成済みのシグネチャとは異なるパターンを有するためである．検出が回避されたマルウェアは解析が必要となるため，亜種の増加はマルウェア解析者の負担増加につながる．

これらの問題に対しては，機能の類似性に基づいたマルウェアの分類が有効である．分類で得られた結果を用いることで，解析すべき検体の削減や，解析前の挙動推測による解析の効率化に貢献することが期待できる．特に近年では，機械学習によるマルウェア対策が数多く提案され，データやモデルの最適化による分類精度の向上が試みられており，API コールやシステムコールなどの動的解析情報を用いた分類 [4][5] や，逆アセンブルによる機械語命令列やマルウェアバイト列などの静的解析による情報を用いた分類 [6][7][8] が行われている．しかし，分類の結果に寄与した特徴の抽出にはあまり着目されていない．特徴抽出によって分類先ファミリに由来する特徴が明らかになれば，分類根拠の明確化やより分類に適した特徴の選出による高精度な分類が実現できることが考えられる．

このような特徴抽出と分類根拠の明確化を目的としたニューラルネットワーク技術の一つに，注意機構（Attention mechanism）という仕組みが存在しており，これをマルウェアの検知・分類に応用した研究もいくつか存在する．例えば，矢倉ら [9] は，マルウェアバイナリを 64×64 サイズのグレイスケール画像に変換し，注意機構付き CNN (Convolutional Neural Network) による分類と注意度マップの生成を行った．これにより，既存手法より高い精度で分類できることを示した．また，注意度マップの重要領域を解析することで，その領域がファミリ特有の情報を有することを示した．Xie ら [10] は，逆アセンブルによって取得した機械語命令列を CBOW と skip-gram を用いた特徴量に変換し，注意機構付きの 2 段 LSTM による分類を行い，注意機構によって精度が向上することを示した．また，Sunoh ら [11] は，注意機構付きモデルによって，ファイルの悪性判定をする上で重要な特徴を含む部分シーケンスを API コール列から特定し，それらを用いた 2 値分類を行うことで，従来のモデルより高い検知精度が得られることを示した．

本稿では，マルウェアの API コール情報を特徴とした分類のために注意機構を用いた実験を行い，特に以下の内容について検討する．

- 注意機構が分類精度向上に有効に働くかどうか．
- 注意機構から抽出した入力情報の重要度がマルウェアファミリごとの特徴的な動作を表すかどうか．

具体的には，分類に寄与した特徴を明らかにし，それらを学習に用いることで高精度な分類を実現する注意機構付き LSTM (Long Short-Term Memory) 手法を提案する．特に，分類には Windows API の関数名および引数の時系列データを用いて，注意機構の有無による分類精度の比較を行う．また，注意機構から抽出した重み値に基づき，重要度が高いとみなされた特徴が，分類先のファミリに由来した特徴となっているかを確認する．その結果，（1）注意機構および引数情報を用いることによる分類精度の向上および（2）抽出した特徴からファミリに特有だと推測される情報を取得するという結果が得られた．

以下，本稿では，2 章で提案手法について述べ，3 章で検証実験に使用するデータとその結果について述べ，4 章で今後の展望について述べ，5 章でまとめる．

2. 注意機構付き LSTM によるマルウェア分類

本章では， ニューラルネットワークの関連技術である LSTM と注意機構を紹介した後，提案手法について述べる．

2.1 Long Short-term Memory

Hochreiter と Schmidhuber [12] により導入された Long Short-term Memory (LSTM) は，長期的な依存関係を学習可能なリカレントニューラルネットワーク（RNN: Recurrent Neural Network）の一種である．LSTM では，従来の RNN の学習時に生じる勾配爆発および消失問題を，中間層の複雑化によって改善している．LSTM の中間層は，値を記憶するセルと，直前時刻のデータの流れを制御する入力・忘却・出力の 3 つのゲートから構成される．図 1 に LSTM のブロック構造を示す. 具体的に，入力 C_{t-1}〜出力 C_t 間で表すのが長期に渡って値を記憶するセル（Memory）の情報伝搬で，入力 h_{t-1}〜出力 h_t 間で示されるのが短期の値を記憶するセル（Reccurent）の情報伝搬である．また，図 1 の σ で表す部分が左から順に入力・忘却・出力ゲートであり，0〜1 の範囲の出力を行うシグモイド関数によって，情報伝達の重みを調整する．これらによって，現在の入力 x_t と，前回の出力 x_{t-1} を用いた計算を行い，出力 h_t を決定する．この出力 h_t は次の時刻 $t+1$ の入力となる．また，前述した単方向の LSTM に加え，双方向に発展させた Bidirectional-LSTM (BLSTM) も用いる．このモデルは未来から過去に流れるデータも考慮するように，順方向と逆方向の 2 つの LSTM の中間層が連結した構造となっており，学習対象によっては通常の LSTM より高い性能を発揮するとされている．

2.2 注意機構

注意機構は，入力情報の要素ごとの関係性や重要箇所を

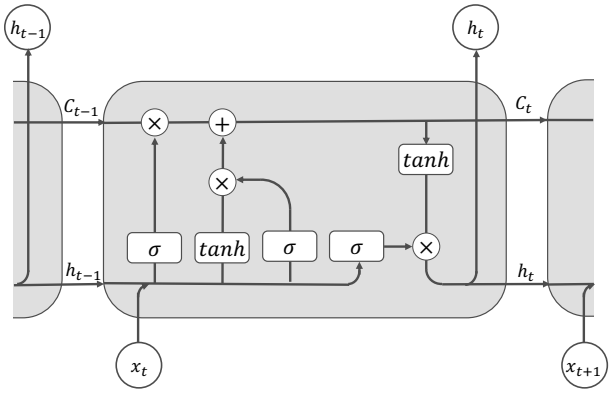

図 1 LSTM のブロック構造

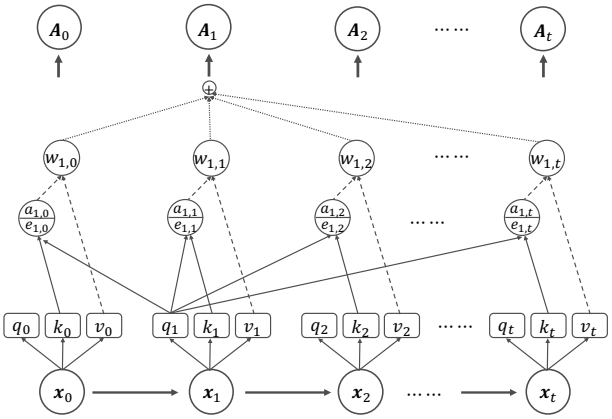

図 2 注意機構の仕組み

学習する機構のことであり，自然言語処理や画像処理の問題における精度向上や，ニューラルネットワーク内部の可視化に貢献している [13][14]．特に，単一の時系列データに対して適用するものは，自己注意機構（Self-attention）と呼ばれる．図 2 に注意機構の仕組みを示す．注意機構の入力には LSTM の隠れ層の値が用いられ，各時点の入力と無作為に生成した重みセットを用いて，Query, Key, Value の３つの値を生成する．次に重み値を求めたい入力の Query と各入力の Key でそれぞれの内積を求め，それを正規化して注意度スコアを求める．最後に求めたすべての注意度スコアを加算することで，入力に対する注意機構からの出力が求められる．図 2 の例では，入力 x_1 に対する出力を求めるために，q_1 と $k_n(n = 0, 1, \cdots, t)$ のそれぞれの内積から注意度スコア $a_{1,n}$ を求め，$a_{1,n}$ と v_n の乗算で求めたそれぞれの重み値 $w_{1,n}$ をすべて加算して，入力 x_1 に対応した注意機構の出力 A_1 を求める．

2.3 提案手法

提案手法では，マルウェアの動的解析によって取得した Windows API の関数名および引数を時系列順に並べたものを特徴として，注意機構付き LSTM によるマルウェアの分類を行う．分類では，後述する 2 種類の LSTM に注意機構を加えた 4 通りでの精度の比較を行うことで，注意機構が分類精度の向上に貢献するかを示す．また，注意機構によって求めた入力の重みを抽出し，重要度が高いとされた特徴がいずれであるか，その特徴がファミリ分類に寄与した特徴であるかを確認する．API の引数情報を含む分類は関数名のみの場合と比べ，特徴が圧倒的に増加するために学習コストや特徴選択が困難となるが，マルウェアファミリごとの詳細な特徴を把握するには，引数の情報こそ重要だと考えられる．そのため，分類および重要特徴の確認では，引数を含む API コール情報を用いることで，より詳細な特徴の検出が可能だと考えられる．また，これらによって抽出した特徴の中から，重要度の高い特徴のみを選出したデータによる，更に高精度な分類の実現可能性について考察する．

2.3.1 特徴量の生成

モデルの学習には，API 関数名のみのデータセット（データセット１）と API 関数名と引数を組み合わせたデータセット（データセット２）の２つを用いる．特徴量生成の流れを以下に示す．なお，以下では引数を含むデータセットを扱う場合について述べる．含まない場合は手順 (3)(4) は行わない．

(1) Windows API 呼び出し情報を時系列順に取得
(2) 類似機能の API を統合
(3) 特定の引数値を固定文字列に置換する
(4) フィルタ値以下の出現割合の特徴を削除
(5) 時系列の重複を削除後，長さを 2,000 に揃える（2000 に満たないものはゼロパディングする）

引数情報を特徴に含む場合，API 関数名のみの場合と比較して大幅に特徴の数が増加する．これはハンドル値などの検体や実行環境によって値が変化する特徴も含まれているためであり，これらの特徴を取り除くために，引数を取得する API や引数を限定する．(2) では，類似した機能の API を統合する．API の中には，実行される環境が可変長なマルチバイト文字と Unicode のいずれを扱うかで名称が異なる API が存在する．例として，ファイルを複製する API である CopyFile の末尾には，A か W のいずれかの文字列がつく．これは実行環境に対応するよう使い分けられており，実際の機能はほぼ同じものである．また，末尾に Ex などがつくものも存在し，これらは元の API を拡張したものであるため，ベースの機能は共通している．したがって，これらを一つの特徴とみなして統合する．また，(3) では，特定の引数値を固定文字列に置換する．引数に含まれる情報のうち，「ユーザ名，HASH 値 (実行された検体名)，CLSID」は汎用的に出現する文字列であるため，固定の文字列に置き換えることで同じ特徴とみなす．生成された特徴は以下のようなフォーマットで表される：APIname{arg_key1：arg_value1,arg_key2：arg_value2,...}.

また，(4) では，閾値（フィルタ）以下の特徴を取り除く．生成した特徴の中には，出現回数が極端に少ないために分

表 1 引数取得対象の API

Category	API
crypto (2)	CryptCreateHash, CryptGenkey
file (16)	CopyFileW(A), CreateDirectoryW, DeleteFileW, GetFileAttributesExW(W), GetFileVersionInfoW, MoveFIleWithProgressW, NtCreateFile, NtDeleteFile, NtOpenDirectoryObject, NtOpenFile, NtQueryAttributesFile, NtQueryDirectoryFile, NtQueryFullAttributesFile, RemoveDirectoryW(A), SearchPathW, SetFileAttributesW
misc (1)	SHGetFolderPathW
network (9)	DeleteUrlCacheEntryW(A), HttpOpenRequestW(A), InternetConnectW(A), InternetCrackUrlW(A), InternetOpenW(A), InternetOpenUrlW(A),URLDownloadToFileW, WSAConnect, WSASocketW(A)
process (13)	CreateProcessInternalW, NtAllocateVirtualMemory, NtCreateProcessEx, NtCreateSection, NtCreateThreadEx, NtCreateUserProcess, NtMapViewOfSsection, NtOpenProcess, NtOpenSection, NtOpenThread, NtProtectVirtualMemory, ShellExeuteExW
registory (15)	NtCrateKey, NtDeleteKey, NtDeleteValueKey, NtOpenKeyEx, NtQueryKey, NtQueryMultipleValueKey, NtQueryValueKey, NtSetValueKey, RegCrateKeyExW(ExA), RegDeleteKeyW(A), RegDeleteValueW(A), RegOpenKeyExW(A), RegQueryInfoKeyW(A), RegQueryValueExW(ExA), RegSetValueExW(ExA)
system (5)	LdrGetDllHandle, LdrLoadDll, LdrUnloadDll, LookupPrivilegeValueW, SetErrorMode

類に影響しない特徴も存在することが推測される．そのため，各ファミリで一定割合以上出現する特徴のみを分類に使用するため，フィルタの値に基づく特徴の選出を行う．今回の分類では，適切な値が把握できていないため，フィルターの値は $[0.1, 0.3, 0.5, 0.7, 0.9]$ の 5 種類とした．さらに，(5) で，生成される時系列データを実行した検体と同名のプロセス（メインプロセス）内に出現するものを対象として，後述する時系列圧縮のアルゴリズムを適用した上で，時系列長を 2,000 に揃える．ただし，圧縮処理適用後の時系列が 50 未満の検体は，分類データから取り除く．

2.3.2 時系列データの圧縮

マルウェアには，同じ API コールが繰り返し出現する検体が一定数存在している．このような動作を行う理由として，動的解析により情報が取得されることを避けるために，特定環境下で解析終了まで意図的に同じ API コールを発行し続けることが考えられる．また，先に述べたような意図がなくとも，マルウェアには特定の API コールが頻繁にに呼び出される場合がある．特に，今回の時系列長の上限を定めた分類では，分類精度が低下する要因となる可能性があるため，冗長な部分シーケンスを削除する．提案手法では，時系列を一定サイズのブロックに分割し，隣り合うブロック内の要素が完全に一致した場合に片方を削除する．本実験では，ブロックの初期値を５とし，１つづつ下げながらアルゴリズムを適用することで冗長性を取り除いたデータを生成する．

3. 検証実験

本章では，分類に使用する学習データの作成と，分類結果および重み抽出によって得られた知見について述べる．

3.1 データセット

データセットには，マルウェア対策研究人材育成ワークショップ（MWS）の研究用データセットの一部として提供される FFRI Dataset 2016, 2017 [15] を用いる．このデータセットは，FFRI セキュリティ社が独自収集した，PE フォーマットかつ Windows プラットフォーム上で実行可能なマルウェア検体を，Cuckoo Sandbox [16] によって動的解析したログで構成されている．2017 年度は Windows 7，2016 年度は Windows 8.1 と Windows 10 の２つのゲスト環境で解析された結果となっており，2016 年度は Windows 10 のデータを使用する．また，動的解析されたログには，マルウェアが発行した Windows API 情報や，VirusTotal によるアンチウイルソフトの検知結果が含まれる．そのため，分類には Windows API の関数名および表 1 の 61 種類の API から取得した引数の時系列データを，判定ラベルには Microsoft の検知結果を使用する．[17] によると，Microsoft の命名規則は [Type:Platform/Family.Variant!Suffixes] となっている．提案手法ではファミリ単位の分類を行うため，命名規則の先頭から “Family” までをラベルとし，100 検体以上を有する表 2 の 27 ファミリ 5,958 検体を使用する．

3.2 実験環境

ニューラルネットワークの実装に Keras [18]，注意機構の実装に keras-self-attention [19] を使用した．分類モデルには，ユニット数を 128 とした LSTM と BLSTM の注意機構の有無による４種類を使用する．それぞれのモデルパラメータ設定を表 3 に示す．データセットは学習，検証，評価でそれぞれ 6:2:2 の割合で分割し，エポックの最大回数を 100（3 エポック連続で loss の変化がなければ終了），バッチサイズを 16 として分類を行った．

3.3 性能評価

分類精度の評価にはマクロ平均を用いる．マクロ平均はマルチクラス分類の適用時に使われる評価指標で，各クラスの評価指標をクラス数で割ったものである．このとき，各クラスの評価については，True Positive (TP)，True Negative (TN)，False Positive (FP)，False Negative (FN) の４つの要素で構成された混合行列をもとにした評価指標が用いられる．それぞれが示す意味は以下の通りである．

- TP: 正解ファミリの検体を正しく分類した数
- TN: 別ファミリの検体を正しく分類した数
- FP: 別ファミリの検体を誤って分類した数

表 2 学習に用いる FFRI Dataset の検体

	Type	Family	Samples
0	Backdoor	Fynloski	588
1		Simda	103
2		Vawtrak	161
3	PWS	Fareit	209
4		Zbot	126
5	Ransom	Cerber	1,006
6		Crowti	100
7		Haperlock	111
8		Locky	164
9		Teerac	155
10		Tescrypt	251
11	Trojan	Gupboot	119
12		Kovter	315
13		Mooqkel	172
14		Pariham	163
15	TrojanDownloader	Upatre	133
16	TrojanDropper	Bunitu	130
17	TrojanSpy	Banker	117
18		Ursnif	547
19	Virus	Parite	137
20		Ursnif	166
21	Worm	Dorkbot	126
22		Drolnux	101
23		Gamarue	279
24		Ludbaruma	103
25		Pykspa	257
26		Rebhip	119
	Total		5,958

表 3 モデルの構成

層	出力サイズ	活性化関数
Input	時系列長*1	-
Embedding	時系列長*128	-
LSTM (BLSTM)	時系列長*128(*2)*	-
(Self-attention)	128(*2)*	-
MaxPooling	128(*2)*	-
Affine	128(*2)*	ReLU
Dropout	128(*2)*	-
Output	ファミリ数	Softmax

* 括弧内は Bidirectional LSTM の場合

- FN: 正解ファミリの検体を誤って分類した数

また，上記の混合行列によって，正解率 (Accuracy)，適合率 (Precision)，再現率 (Recall)，F 値 (F-measure) を求める．正解率は予測が正しい検体の割合，適合率は誤検知の少なさ，再現率は検知漏れの少なさを表し，F 値は適合率と再現率の調和平均である．各評価指標は以下の計算式で表される．

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

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

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

$$\text{F-measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.3.1 分類結果

モデルおよびデータセットごとの分類結果を表 4 に示す．注意機構の適用により，データセット 1 の BLSTM では約 0.4%，LSTM では約 3.6%精度が向上した．また，データセット 2 を用いた実験では，各フィルター値ごとに BLSTM で最大約 2.6%，LSTM で最大約 5.1%精度が向上し，すべてのパターンで同等以上の精度となることが確認できた．次にデータセットごとの精度を確認すると，注意機構付きの LSTM および BLSTM では約 0.6〜2.8%精度が向上することが確認できた．ただし，通常の LSTM の結果では関数名のみのほうが精度が高い場合がある．また，出現頻度を閾値としたフィルターによって引数を含む特徴の数を制限したが，分類精度への影響は確認できなかった．

次にファミリ別の分類結果について，最も精度が高かったデータセット 2 による注意機構付き BLSTM（filter=0.3）の結果を確認した．表 5 に示すファミリ別精度の F 値は，最高で 1.0, 最低で 0.19 となることが確認できた．各ファミリを確認すると 23/27 ファミリで分類精度がおよそ 90%に近い精度となっており，他の 4 ファミリは精度が著しく低いことが確認できた．精度の低いファミリとその精度は PWS:Zbot が 0.19, PWS:Fareit が 0.63, Trojan:Kovter が 0.78, Worm:Rebhip が 0.43 である．この中で Trojan:Kovter は recall が 1.0 となっているため，誤分類されることによって精度が下がっていることが確認できる．また，PWS:Fareit, PWS:Zbot, Worm:Rebhip の誤分類された検体は，Backdoor:Fynloski，Trojan:Kovter のいずれかへの誤分類が大半を占めていることが確認できている．

3.3.2 考察

注意機構の有無によるモデルごとの精度比較では，すべてのケースで分類精度が向上することが確認できた．このことから注意機構は，API コール情報を特徴とした分類精度の向上に有効といえる．また，データセットごとの精度比較では，注意機構付きのモデルではデータセット 2 を用いた場合の精度が高いことから，引数情報はマルウェアの分類精度の向上に有効だといえる．ただし，注意機構なしモデルでは，データセット 1 を用いた精度の方が高いことがあるため，分類モデルによって精度の向上に有効かは変わると考えられる．また，今回 API 引数を含む場合の特徴が多くなるため，フィルターによって特徴数を制限したが，分類精度に大きな影響は与えなかった．これは，デー

表 4 分類結果

Dataset	Model	Attention	Filter	Accuracy	Precision	Recall	F-measure
データセット 1: (API 関数名のみ)	BLSTM	○	–	0.891	0.891	0.860	0.866
		–		0.887	0.872	0.853	0.854
	LSTM	○		0.897	0.914	0.861	0.868
		–		0.860	0.872	0.824	0.831
データセット 2: (API 関数名+引数)	BLSTM	○	0.1	0.913	0.916	0.898	0.901
			0.3	0.919	0.925	0.894	0.899
			0.5	0.914	0.904	0.890	0.893
			0.7	0.906	0.905	0.882	0.887
			0.9	0.901	0.890	0.891	0.886
		–	0.1	0.887	0.876	0.862	0.862
			0.3	0.907	0.892	0.876	0.877
			0.5	0.895	0.905	0.860	0.865
			0.7	0.896	0.896	0.871	0.876
			0.9	0.895	0.886	0.882	0.881
	LSTM	○	0.1	0.910	0.919	0.889	0.892
			0.3	0.918	0.920	0.901	0.904
			0.5	0.912	0.909	0.888	0.890
			0.7	0.903	0.903	0.889	0.887
			0.9	0.904	0.896	0.879	0.879
		–	0.1	0.859	0.844	0.832	0.830
			0.3	0.876	0.860	0.844	0.840
			0.5	0.874	0.836	0.837	0.829
			0.7	0.859	0.844	0.831	0.833
			0.9	0.854	0.845	0.818	0.819

表 5 ファミリごとの分類精度評価

Family	Precision	Recall	F-measure	Samples
00:Backdoor:Fynloski	0.840	0.948	0.890	116
01:Backdoor:Simda	1	1	1	20
02:Backdoor:Vawtrak	1	0.969	0.984	32
03:PWS:Fareit	0.905	0.500	0.644	38
04:PWS:Zbot	0.375	0.130	0.194	23
05:Ransom:Cerber	0.976	0.976	0.976	166
06:Ransom:Crowti	0.938	0.833	0.882	18
07:Ransom:Haperlock	1	1	1	22
08:Ransom:Locky	1	0.781	0.877	32
09:Ransom:Teerac	0.963	0.929	0.945	28
10:Ransom:Tescrypt	0.964	1	0.982	27
11:Trojan:Gupboot	1	1	1	24
12:Trojan:Kovter	0.642	1	0.782	61
13:Trojan:Mooqkel	0.971	1	0.986	34
14:Trojan:Pariham	1	1	1	33
15:TrojanDownloader:Upatre	0.964	1	0.982	27
16:TrojanDropper:Bunitu	1	0.920	0.958	25
17:TrojanSpy:Banker	0.846	0.957	0.898	23
18:TrojanSpy:Ursnif	0.962	0.944	0.953	107
19:Virus:Parite	1	1	1	27
20:Virus:Ursnif	1	1	1	33
21:Worm:Dorkbot	1	1	1	22
22:Worm:Drolnux	1	1	1	20
23:Worm:Gamarue	0.855	0.959	0.904	49
24:Worm:Ludbaruma	1	1	1	21
25:Worm:Pykspa	1	1	1	51
26:Worm:Rebhip	0.778	0.304	0.438	23
Average accuracy			0.919	1102
Macro avg	0.925	0.894	0.899	1,102

タの時系列長を固定していたために，固定長以降の特徴の変化が影響しないことが要因の一つとして考えられる．

また，ファミリ別の分類精度は，PWS:Zbot, PWS:Fareit, Trojan:Kovter, Worm:Rebhip の 4 ファミリの精度が特に低いことが確認できた．この中で Trojan:kovter は recall の値が 1.0 となっている．これにより，自ファミリの検体を 100%分類しながら，他ファミリの誤分類先ともなっていることで精度が低下していることが分かる．また，残りの 3 ファミリの誤分類された検体は大半が Backdoor:Fynloski, Trojan:Kovter のいずれかに分類されていることが確認できた．この理由は 2 つ考えられる．一つは学習データのばらつきまたはラベルが不適切だったために，望ましい分類結果が得られなかったことである．もう一つは，異なるファミリ同士に共通する性質によって誤分類が多くなってしまったことが考えられる．

3.4 注意機構の重み値に基づくファミリの特徴

注意機構から抽出した重みは各入力ごとの重要度を表しており，重要度の高い特徴を確認することで，どのような特徴が分類に寄与したか，またその特徴がファミリを象徴する特徴であるかを把握する．

3.4.1 分類精度の高いファミリの特徴

Backdoor:Fynloski（別名：DarkComet）は，正規ツールを悪用して侵入先の端末から情報窃取を行う RAT (Remote Access Trojan) である．このファミリでは表 6 に示すように，特権に対応した LUID を取得する “LookupPrivirageValueW” という API の情報を重要としており，実際の API コールでは “LookupPrivirageValueW” と “NtClose”

表 6 Backdoor:Fynloski の Top 5 の特徴

順位	API
1	NtClose
2	LookupPrivilegeValueW{'privilege_name':'SeManageVolumePrivilege'}
3	LookupPrivilegeValueW{'privilege_name':'SeSystemEnvironmentPrivilege'}
4	LookupPrivilegeValueW{'privilege_name':'SeUndockPrivilege'}
5	LookupPrivilegeValueW{'privilege_name':'SeRemoteShutdownPrivilege'}

表 7 Ransom:Crowti の Top 5 の特徴

順位	API
1	CryptAcquireContextW
2	NtAllocateVirtualMemory{'protection':4,'allocation_type':12288}
3	NtFreeVirtualMemory
4	CryptHashData
5	CryptCreateHash{'algorithm_identifier':'0x00008003'}

表 8 TrojanSpy:Ursnif の Top 5 の特徴

順位	API
1	NtResumeThread
2	NtSuspendThread
3	NtGetContextThread
4	NtDelayExecution
5	NtAllocateVirtualMemory{'protection':64,'allocation_type':12288}

が交互に出現するシーケンス傾向が確認できた．このような特徴は他のファミリでは確認できないため，分類に有効な特徴だと考えられる．

また，Ransom:Crowti（別名:CryptWall）は，バージョンアップを重ねながら多くの被害を出したランサムウェアである．このファミリでは，表 7 に示すように，暗号化に関する API が重要としていることが確認できる．特に，“CryptAcquireContextW”という API の重要度が高いファミリはこのファミリのみであり，分類に有効な特徴群であると考えられる．

また，TrojanSpy:Ursnif（別名:Gozi/Papras）は，オンラインバンキングマルウェアである．このファミリでは，表 8 に示すように，インジェクション攻撃に使用されるスレッドに関する API を重要としていることが確認できる．この特徴は今回使用するデータセットに含まれる他のファミリでは確認できず，分類に有効な特徴だと考えられる．

3.4.2 分類精度の低いファミリの特徴

分類精度の低いファミリとして, PWS:Fareit, PWS:Zbot, Worm:Rebhip の重み傾向について調査すると，検体ごとの重要度の高い特徴に共通点が少ないために，ファミリとしての特徴は把握できなかった．一方で，各ファミリの誤分類された検体について重みの傾向を確認すると，一部の検体間で分類ラベルに関係なく，表 9 に示すような共通した特徴があることが確認できた．これらの検体について調査すると，重み値だけでなく，時系列そのものが Trojan:Kovter に非常に類似した検体が 30 検体近く存在することが確認できた．今回分類精度の低かったファミリには，これらの特徴を持つ検体が特に多かったために精度が下がっていると考えられる．したがって，これらの共通する特徴を持つ検体およびのラベルが適切かは，検討の必要がある．

3.4.3 考察

重み値に基づくファミリの特徴を確認したことで，一部のファミリでは特有だと考えられる特徴が重要とされていることが確認できた．このことから，注意機構の適用による分類に有効な特徴抽出の可能性が考えられる．一方で，分類精度が高いファミリ同士で重要視される特徴が類似しているものも確認できた．今回の特徴確認では時系列での位置を考慮せず，重要度が上位の特徴のみを確認している．そのため，上位の特徴が必ずしもファミリ特有の特徴ではなく，時系列における出現位置やその周辺のサブシーケンスなども踏まえた特徴抽出を行うことが望ましい．また，誤分類検体の重み抽出を行った結果，一部の検体ではファミリのラベルに関わらずに類似した特徴を有することが確認できた．そのため，特徴抽出は誤分類の要因や傾向を調べるためにも有効となることが考えられる．

4. 議論

本稿では，分類に用いたデータはメインプロセスから取り出した時系列データであり，複数のプロセスがある検体からも同様の手法を取っている．マルウェアは複数のプロセスを用いて悪性挙動を行うものも存在しており，メインプロセスのみのデータでは，特徴的な挙動情報を損失していることが考えられる．そのため，メインプロセス以外のプロセス情報を含めたデータを用いることで，より詳細な挙動を明らかにすることが可能だと考えられる．また，引数を含む特徴選択では対象となる API を選択した上で出現割合にフィルターをかけることで，特徴削減を行った．しかし，削除された中にも分類に有効と考えられる特徴が存在していると考えられる．例として，今回分類に使用した Ransom:Cerber ファミリでは，身代金を要求するためにウィンドウに警告を促す文字列を表示する．その文字列の中には “_HELP_HELP_HELP_{random}”や，“_READ_THIS_FILE_{random}”と行ったファミリ特有の文字列が含まれる．これらはログ内の情報でも確認でき，文字列を統一しなければそれぞれ別の特徴として扱われる．一方で，一見無作為に作成されたと考えられる文字列でも，ファミリとして共通する特徴である可能性も考えられる．そのため，これらの文字列情報を適切に処理することができれば，ファミリ特有な特徴抽出に活かせると考えられる．

また，分類では，注意機構を用いることによる分類精度の変化と抽出できる特徴の傾向を確認した．その結果，分類精度の向上と一部のファミリで分類に有効らしき特徴の把握が可能なことが確認できた．したがって注意機構の適用が今回の 2 つの目的を実現するのに有効だと考えられる．一方で，今後の課題として注意機構から抽出したこれらの

表 9 共通する特徴を持つ誤分類検体の Top 5 の特徴

順位	API
1	NtOpenKeyEx{'desired_access':'0x00000001','regkey':'HKEY_CURRENT_USER\Software\Policies\Microsoft\IME\Shared'}
2	NtOpenKeyEx{'desired_access':'0x00000001','regkey':'HKEY_CURRENT_USER\Software\Policies\Microsoft\IME\IMEJP'}
3	NtOpenKeyEx{'desired_access':'0x00000001','regkey':'HKEY_CURRENT_USER\Software\Microsoft\IME\15.0\IMEJP\MSIME'}
4	NtQueryValueKey{'reg_type': 4, 'information_class': 2,'regkey': 'HKEY_CURRENT_USER\SOFTWARE\Microsoft\IME\15.0\IMEJP\MSIME\option1'}
5	NtOpenKeyEx{'desired_access':'0x02000000', 'regkey':'HKEY_CURRENT_USER\WOW6432Node\CLSID\{{CLSID}}\InprocServer32'}

特徴をどのように活用するかが重要となる．一例には，時系列の形式を維持して重要な特徴によるシーケンスを用いた分類などに活用することが考えられる．今回特徴抽出した中で，重みの上位の特徴が共通していても分類精度が高いファミリも存在していることから，これらによるシーケンスを形成することでより高い精度での分類可能性が考えられる．

5. おわりに

本稿では，API の関数名および引数を含む２つのデータセットを用いて，注意機構付きモデルによる分類精度の比較，および注意機構の重み値に基づく特徴抽出を行った．分類性能では，注意機構を導入したモデルがより高い性能を発揮することが確認でき，API 引数の使用による精度についても，一定の性能向上の可能性があることが確認できた．また，重み値に基づく特徴抽出では，一部のファミリで抽出された特徴がファミリ分類に寄与した特徴である可能性を示唆することができた．

今後は，重み値に基づいた特徴選出を行い，それらを用いることで分類性能の向上に貢献できるかを検証する．具体的には，抽出した特徴のみをデータセットとし，別の機械学習モデルによる分類を行うことで分類が可能であるか，また性能向上に貢献できるかについて実験を行うことを考えている．

参考文献

[1] McAfee: 2021 年 6 月 McAfee Labs 脅威レポート, (オンライン)，入手先 ⟨https://www.mcafee.com/enterprise/ja-jp/assets/reports/rp-threats-jun-2021.pdf⟩ (参照 2021-09-15).
[2] T.J. O'Leary: Threat Spotlight: ZeuS (aka Zbot) Infostealer Trojan, (online), available from ⟨https://blogs.blackberry.com/en/2020/04/threat-spotlight-zeus-infostealer-trojan⟩ (accessed 2021-09-15).
[3] マイナビニュース: Android デバイスをロックするランサムウェア、アプリで手軽に作成, (オンライン)，入手先 ⟨https://news.mynavi.jp/article/20170828-a228/⟩ (参照 2020-09-15).
[4] Kwon, I. and Im, E. G.: Extracting the Representative API Call Patterns of Malware Families Using Recurrent Neural Network, pp. 202–207 (online), DOI: 10.1145/3129676.3129712 (2017).
[5] Alsulami, B. and Mancoridis, S.: Behavioral Malware Classification using Convolutional Recurrent Neural Networks, pp. 103–111 (online), DOI: 10.1109/MALWARE.2018.8659358 (2018).
[6] Safa, H., Nassar, M. and Al Orabi, W.: Benchmarking Convolutional and Recurrent Neural Networks for Malware Classification, pp. 561–566 (online), DOI: 10.1109/IWCMC.2019.8766515 (2019).
[7] Nataraj, L., Karthikeyan, S., Jacob, G. and Manjunath, B. S.: Malware Images: Visualization and Automatic Classification, *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, VizSec '11, New York, NY, USA, Association for Computing Machinery, (online), DOI: 10.1145/2016904.2016908 (2011).
[8] Nataraj, L., Yegneswaran, V., Porras, P. and Zhang, J.: A comparative assessment of malware classification using binary texture analysis and dynamic analysis, (online), DOI: 10.1145/2046684.2046689 (2011).
[9] 矢倉 大夢, 他：CNN と注意機構による画像化されたマルウェアの解析手法，コンピュータセキュリティシンポジウム 2017 論文集，pp. 1381–1388 (2017).
[10] Qi Xie, Yongjun Wang, Z. Q.: Malware Family Classification using LSTM with Attention, *International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)* (2020).
[11] Choi, S., Bae, J., Lee, C., Kim, Y. and Kim, J.: Attention-Based Automated Feature Extraction for Malware Analysis, *Sensors*, Vol. 20, No. 10 (online), DOI: 10.3390/s20102893 (2020).
[12] Hochreiter, S. and Schmidhuber, J.: Long Short-term Memory, *Neural computation*, Vol. 9, pp. 1735–80 (online), DOI: 10.1162/neco.1997.9.8.1735 (1997).
[13] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I.: Attention Is All You Need, *CoRR*, Vol. abs/1706.03762 (2017).
[14] Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B. and Bengio, Y.: A Structured Self-attentive Sentence Embedding, *CoRR*, Vol. abs/1703.03130 (online), available from ⟨http://arxiv.org/abs/1703.03130⟩ (2017).
[15] 寺田真敏, 他：マルウェア対策のための研究用データセット MWS Datasets 〜コミュニティへの貢献とその課題〜, 情報処理学会， Vol. 2020-IFAT-139, No. 8 (2020).
[16] Foundation, C.: Cuckoo Sandbox - Automated Malware Analysis, `https://cuckoosandbox.org`.
[17] Dansimp: マルウェアの名前 - Windows security, (オンライン)，入手先 ⟨`https://docs.microsoft.com/ja-jp/windows/security/threat-protection/intelligence/malware-naming`⟩ (参照 2021-06-23).
[18] Chollet, F. et al.: Keras, https://keras.io (2015).
[19] CyberZHG: keras-self-attention, `https://pypi.org/project/keras-self-attention/`.