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

矢倉 大夢[1]　篠崎 慎之介[1]　西村 礼恩[1]　大山 恵弘[1]　佐久間 淳[1]

概要：本研究では，画像化されたバイナリデータにCNNを適用することによって，マルウェアからそのファミリに特有の領域を検出する分類手法を提案する．この手法では，CNNに注意機構と呼ばれる仕組みを組み合わせることによって，画像の中で分類に重要な領域を示す「注意度マップ」を出力する．これにより，該当するファミリに特有の領域を検出することでき，人手でその動作を解析する際のヒントとなる．評価実験では，提案手法が既存手法より高い分類精度を誇ることを示すと共に，注意度マップを元にマルウェアを解析することによって，提案手法から得られる情報が，パックされたマルウェアを解析する場合を含め，有用であることを確かめた．

キーワード：マルウェア解析，Convolutional Neural Network，注意機構

1. はじめに

マルウェアの蔓延は，コンピュータセキュリティにおける大きな脅威となっており，多額の経済的損失を生んでいる [1]．様々なマルウェアの，とどまるところを知らない進化に対応するには，マルウェアの挙動に関する情報を簡単に解析できることが大切である．特に，静的解析においては，マルウェアはリバースエンジニアリングされ，逆アセンブルによって得られた命令列を1つずつに解析していくことで，挙動に関する情報を得る．

マルウェアの駆除には，その挙動に関する情報が欠かせないことから，新しい亜種のマルウェアが見つかったときには，専門家が解析を行うのが一般的である [2]．しかし，こうした人手による解析には，マルウェアの複雑さに応じて，数時間から数日を要する [3]．そして，静的解析が時間を要する理由の1つに，マルウェアのバイナリデータのうちで，どの領域がそのマルウェアに特有の挙動を関わっているのかを特定するのが難しいという点がある．逆に，マルウェアのバイナリデータから，そういった領域を自動的に検出することができれば，マルウェア解析を効率化することができるといえる．

近年，深層学習の発達に伴い，画像分類技術が飛躍的な発展を遂げている．特に，畳み込みニューラルネットワーク（CNN）を用いた手法については，人間を超える分類精度を発揮したという報告 [4] もある．こうしたCNNの精度向上には，画像のどの領域が分類に重要な情報を持っているかを示す特徴量が，ネットワークの中間層において自動的に獲得されることが寄与している．さらに，この，画像のどの領域が分類に重要かという情報は，「注意機構」を組み合わせることによって，可視化できる [5]．これは，CNNに注意機構を組み合わせることによって得られる「注意度マップ」が，分類の対象に特有の領域を示すことによる．例えば，**図1**は，それぞれの画像に対応する見出し文のうち，特定の単語に対する注意度マップを示している．この注意度マップは，画像から見出し文を生成するネットワーク [5] から抽出されたものである．

図1　[5] によって得られる，画像内の特定の領域と見出し文内の下線部の単語の対応の例（ [5] より引用）

本研究では，CNNに注意機構を組み合わせることによって，マルウェアの静的解析を効率化する手法を提案する．提案手法は，マルウェアのバイナリデータを画像へと変換し，注意機構を持つCNNに適用することによって，既知のどのマルウェアの亜種（ファミリ）であるかを分類する．同時に，注意度マップを得ることによって，バイナリデータ内の重要な領域を自動的に検出することを可能とする．そうした情報をヒントとして活用することで，マルウェア

[1] 筑波大学

表 1　マルウェアのファイルサイズと，変換によって得られる画像の幅の対応（ [6] より引用）

ファイルサイズ	画像の幅
< 10KB	32
10KB ∼ 30KB	64
30KB ∼ 60KB	128
60KB ∼ 100KB	256
100KB ∼ 200KB	384
200KB ∼ 500KB	512
500KB ∼ 1000KB	768
100KB ≤	1024

解析の作業量が軽減されると考えられる．

1.1　関連研究

本節では，関連研究として，バイナリデータの画像化に基づくマルウェア分類手法と，深層学習を用いたマルウェア分類手法，そして，マルウェア同士の比較による類似領域検出手法について述べる．

1.1.1　バイナリデータの画像化に基づくマルウェア分類手法

マルウェアのバイナリデータを画像化して分類を行う手法は，Nataraj らによって提案されている [6,7]．この手法では，以下のようにしてマルウェア分類を行う．

(1) マルウェアのバイナリデータを 1 バイトずつ読み込み，0 ∼ 255 の整数列に変換する．
(2) マルウェアのファイルサイズに基づいて，**表 1** のように幅を決め，整数列を 2 次元のグレースケールの画像に変換する．
(3) 得られた画像を 64×64 の解像度に伸縮する．
(4) 伸縮させた画像から，320 次元の GIST 特徴量 [8] を抽出し，特徴ベクトルとする．

以上の手順によって，既知のマルウェアから特徴ベクトルを抽出しておき，k-近傍法によって新たなマルウェアを分類する．つまり，既知のマルウェアとそのファミリの情報を教師データとして，画像化し，GIST 特徴量を抽出することでマルウェア分類を行っている．

Nataraj らは，[7] において，様々なデータセットを用いて評価を行った結果，既存手法よりも高い分類精度を示したと報告している．特に，「パッカ」と呼ばれるプログラムを用いて難読化されている，パックされたマルウェアに対しても，有効であったと述べている．また，Kirat らは，[9] において，画像から得られる特徴量に最近傍法を適用した別のマルウェア分類手法を提案しており，パックされたマルウェアを含むデータセットに対して，バイナリデータの n-gram を用いた手法や，制御フローグラフを用いた手法よりも，高い精度を示したと結論づけている．

一般に，パッカによって難読化されたバイナリデータは，ランダムなバイト列となることが期待されている．そのため，パックされたマルウェアのバイナリデータをそのまま分類器に適用したり，画像化して適用したりしても，適切に分類できないように思われる．しかし，同じファミリに属するマルウェアを，UPX[*1]のような辞書式圧縮法を用いたパッカで難読化すると，大抵の場合，得られるファイルは共通のバイト列を多く含んでいることが知られている [10]．これは，パッカが似た辞書を使ってしまうことに起因しており，これによって，パックされたマルウェアであっても，ファミリの特定につながる特徴が生まれる．

ただし，GIST 特徴量に基づく k-近傍法による，これらの手法では，バイナリデータのどの領域が分類に影響を与えているのかを明確にできない．そのため，人手による解析を効率化するという目的では，用いることができない．

1.1.2　深層学習を用いたマルウェア分類手法

深層学習をマルウェア検知やマルウェア分類に適用する手法はいくつか提案されている [11–15]．これらの手法は，静的解析によるものと動的解析によるものの 2 つに大別することができる．ここで，静的解析によるものとは，マルウェアを実行せずに得られる情報のみを用いた手法を指し，このうちでは，バイナリデータのバイト列のヒストグラムとヘッダの情報を用いた [11] のみが該当する．一方で，動的解析によるものとは，マルウェアを実行することによって得られる情報も用いた手法を指し，API コール列を特徴量としている，他の 4 つの手法 [12–15] が当てはまる．

動的解析は静的解析に対し，より多くの情報を利用することができるが，解析を妨害するために，仮想マシンモニタなどの存在を検知して挙動を変えるマルウェアが多く存在することも指摘されており [16,17]，そういった場合に正しく検知，分類できない可能性がある．また，静的解析による [11] は，マルウェアかどうかの判別を目的としており，あるファミリに特有の領域を検出し，人手による解析を効率化するという目的では利用できない．

1.1.3　マルウェア同士の比較による類似領域検出手法

同じファミリのマルウェアに対する詳細な解析結果が利用できる場合に，解析作業を効率化することを目的として，マルウェア同士の類似領域を検出する手法が研究されている [18–22]．このうち，[18] は，逆アセンブルによって得られる命令列を正規化して特徴量ベクトルを生成し，それを比較することによって検出を行っているが，[19–22] は，逆アセンブルによって得られた命令列から制御フローグラフを生成し，グラフの比較によって検出を行っている．

ただし，いずれもコードセクションに含まれる命令列の情報のみから比較を行っており，データセクションの情報を活用することはできない．そのため，マルウェアがパックされており，挙動に関する情報がデータセクションに置かれている場合には，対応することができない．

[*1] https://upx.github.io/

もちろん，データセクションまでを含めた比較のために，バイナリデータ全体を比較するツール [23] を使用することはできる．ただし，こうしたツールは，コンパイラの差によって生まれる細かなバイナリデータの違いなども差分として出力するため，マルウェアの比較には適していない．

1.2 貢献

本研究の貢献は 3 点ある．

まず，マルウェアのバイナリデータから，そのファミリに特有の領域を自動的に検出する手法を提案したという点である．これにより，人手によるマルウェア解析を効率化することができる．これは，1.1.1 節や 1.1.2 節で述べた既存手法では実現することはできないが，本研究では，CNN に注意機構を組み合わせ，マルウェア分類に適用することで可能とした．

次に，1.1.3 節で述べた既存手法は，コードセクションの情報しか活用できないのに対し，データセクションも含めたバイナリデータ全体から，解析に活用できる情報を得る手法を提案したという点である．本研究は，バイナリデータを画像化し，CNN に適用することによって，コードセクションとデータセクションを等しく扱うことを可能とした．これにより，難読化されたコードがデータセクションに置かれている，パックされたマルウェアにも対応できる．

最後に，提案手法が，既存手法より高いマルウェア分類精度を誇り，かつ，注意度マップによって得られる情報がマルウェア解析に有用であることを示したという点である．例えば，4.4.1 節で述べる *Backdoor.Win32.Agobot.lt* の場合では，注意度マップが最も重要な情報を持つとした領域は，リモートサーバから IRC で命令を受け取る関数を指していた．これは，該当するファミリ *Worm:Win32/Gaobot* の，ボットネットを構築するために，IRC で送られた命令を実行するという挙動を特徴づけるものだといえる．

2. 背景

2.1 畳み込みニューラルネットワーク

CNN は，深層学習手法の 1 つで，画像認識分野にて特に成果を挙げている．CNN は，他の深層学習手法と同様に多層ニューラルネットワークからなるが，その中で，一般的なニューラルネットワークで用いられる全結合層に加え，畳み込み層と呼ばれる特定の構造を持つ点に特徴がある．

畳み込み層は，**図 2** のように，入力に対して，一定のサイズのフィルタを適用し，畳み込み演算を行う．これにより，出力の 1 つの要素が入力のすべての要素を元に計算される全結合層に対し，畳み込み層では，出力の 1 つの要素が入力の限られた領域の要素を元に計算されるという違いを生む．この，結合のスパース性と呼ばれる性質に加え，同じフィルタが入力のすべての要素に適用されることによって生まれる，結合重みの共有という性質によって，

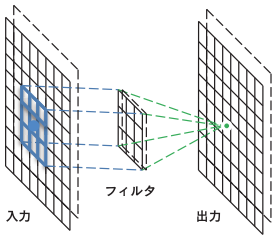

図 2 畳み込み層の模式図（ [24] より引用）

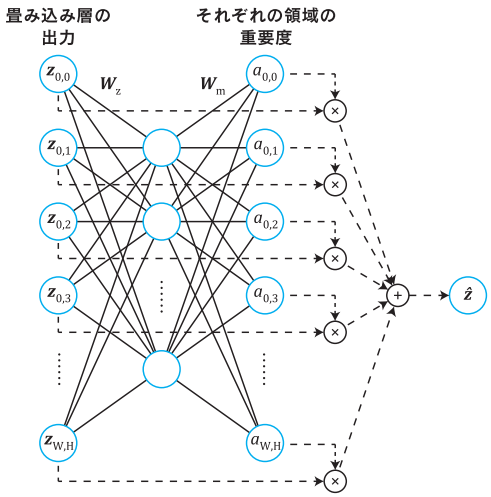

図 3 注意機構の模式図

CNN では，画像内に含まれる局所的な特徴が自動的に獲得できるようになっていることが知られている [25]．

2.2 注意機構

注意機構は，深層学習において用いられる手法の 1 つである．これは，自然言語処理の研究で提案された [26] もので，重要な特徴を動的に選び出し，それらを直接的に用いることで，翻訳精度が高まったとしている．これを CNN に組み合わせることで，図 1 に示すように，画像における重要な領域を可視化することができる [5]．

図 3 は，注意機構の概要を示したものである．まず，$\boldsymbol{z}_{i,j} \in \mathbb{R}^N$ $(0 \leq i < W, 0 \leq j < H)$ を，CNN の畳み込み層の出力とする．通常の CNN では，$\boldsymbol{z}_{i,j}$ をそのまま，次の層の入力として伝播させる．しかし，注意機構は $\boldsymbol{z}_{i,j}$ の重み付き平均 $\hat{\boldsymbol{z}} \in \mathbb{R}^N$ を求め，$\hat{\boldsymbol{z}}$ を次の層へと伝播させる．そして，$\hat{\boldsymbol{z}}$ を計算するために，$\boldsymbol{z}_{i,j}$ のそれぞれの重みを示す $a_{i,j} \in \mathbb{R}$ を，以下のように求める．

$$\boldsymbol{A} = \mathrm{softmax}(\boldsymbol{W}_m \ \mathrm{LeRU}(\boldsymbol{W}_z \boldsymbol{Z}^{\mathrm{T}})) \tag{1}$$

ここで，$\boldsymbol{A} = (a_{i,j})$ かつ $\boldsymbol{Z} = (\boldsymbol{z}_{i,j})$ とし，softmax 及び LeRU は活性化関数とする．このとき，全結合層は行列の積で表せることから，式 1 は $\boldsymbol{W}_m$，$\boldsymbol{W}_z$ を重みとする，2 層のニューラルネットワークとして考えることができる [27]．さらに，$a_{i,j}$ は softmax で $\sum_i \sum_j a_{i,j} = 1$ を満たすよう計算されることから，$\hat{\boldsymbol{z}}$ は以下のように求められる．

$$\hat{\boldsymbol{z}} = \sum_{i=1}^{W}\sum_{j=1}^{H} a_{i,j}\boldsymbol{z}_{i,j} \tag{2}$$

逆に言うと，$a_{i,j}$ が $\boldsymbol{z}_{i,j}$ の重みを表すように，$\boldsymbol{W}_m$，$\boldsymbol{W}_z$ が学習されると考えることもできる．

ここで，$\boldsymbol{z}_{i,j}$ は畳み込み層の出力の各要素であり，図 2 に示されるように，入力の一部の領域から得られた特徴を表しているため，その重みである $a_{i,j}$ は，入力画像の特定の領域の重要度を表していると考えることができる．そのため，$\boldsymbol{z}_{i,j}$ が CNN の浅い層から抽出された場合は，$a_{i,j}$ は入力画像の局所的な特徴に基づいて計算され，それぞれの重みが指す領域は小さくなる．逆に，$\boldsymbol{z}_{i,j}$ が CNN の深い層から抽出された場合は，$a_{i,j}$ は入力画像の大域的な特徴も含めて計算され，それぞれの重みが指す領域は大きくなる．

3. 提案手法

本章では，提案手法の概要と，マルウェアを入力として与えてから，解析者が提案手法によって得られた情報に基づいて，解析を行うまでの具体的な流れについて述べる．

3.1 概要

本研究は，マルウェアから，そのファミリに特有の領域を自動的に検出し，解析を効率化する手法を提案する．提案手法の概要は，**図 4** に示す通りである．まず，マルウェアを入力として受け取り，そのファミリに特有の領域を検出し，出力する（図 4, ①–③）．それらの領域を含む関数がどのように動作するか（図 4, ④），あるいは，それらの領域を含むリソースがどのように用いられるか（図 4, ⑤）を解析することによって，解析者はマルウェアの挙動を知ることができる．

小さな変更を加えることによってアンチウイルスソフトウェアの検知を逃れようとする攻撃者の傾向に対応するためには，提案手法は，検出すべき領域の位置が変わっても対応できる必要がある．また，人手による解析作業の負担を軽減するために，出力される領域は，大域的にではなく，ピンポイントに検出されることが望ましい．

3.2 解析の流れ

3.2.1 マルウェアのバイナリデータの画像化

まず，入力として与えられたマルウェアは，画像に変換される．提案手法では，1.1.1 項で述べた，Nataraj ら [6,7] と同様の手法を用いる．具体的には，表 1 の基準に従って，マルウェアのバイナリデータは 2 次元の画像に変換され，64×64 の解像度に伸縮される．

3.2.2 注意機構を持つ CNN への適用

次に，変換によって得られた画像は，注意機構を持つ CNN に適用される．なお，この CNN は，既知のマルウェアを画像化したものと，そのマルウェアがどのファミリに

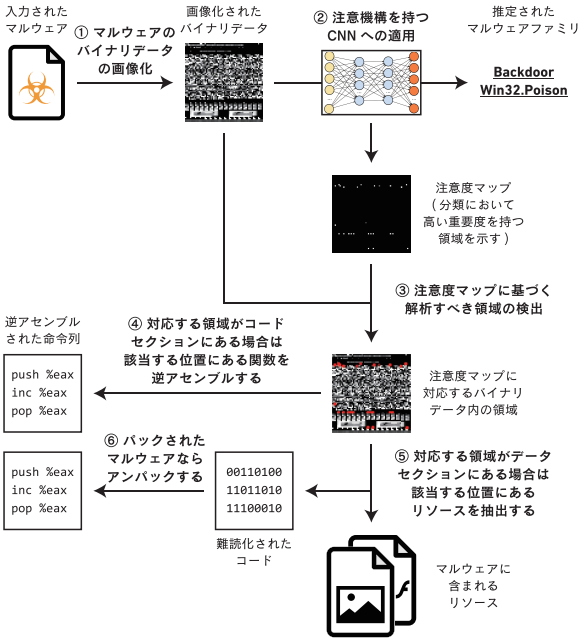

図 4 提案手法の概要図

表 2 提案手法における CNN の構造

層	フィルタ	出力マップサイズ	活性化関数
input	-	$64 \times 64 \times 1$	-
conv1	3×3	$62 \times 62 \times 128$	ReLU
conv2	3×3	$60 \times 60 \times 256$	ReLU
attention	-	256	ReLU
fc1	-	512	ReLU
fc2	-	出力クラス数	softmax

属するかを示すラベルのペアを用いて，事前に学習しておく．そして，入力として与えられたマルウェアのファミリを推定するとともに，注意度マップを出力する．

CNN の構造は**表 2** の通りである．ここで，attention とは，2.2 節で述べた注意機構を指している．つまり，式 2 における $a_{i,j}$ が，入力画像のそれぞれの領域の重要度を示す，注意度マップとして用いられる．これにより，入力として与えられたマルウェアのバイナリデータのうち，そのファミリに特有の領域を検出することが可能になる．

ここで，CNN を用いた理由は，[6,7] における GIST 特徴量のように，人間が過去の経験から設計したものよりも適した特徴量を自動的に獲得できることが知られているためである．例えば，Razavian らによる実験 [28] では，一般画像分類において，CNN が GIST 特徴量を用いた分類器よりも高い分類精度を示したと報告されている．

また，[4] のような，何十層からなる深いネットワークに対し，比較的浅い構造のネットワークを用いた理由は，3.1 節で述べた要件による．つまり，マルウェアのバイナリデータから得られる画像には，大域的な特徴があまり現れないと考えられるのに対し，注意度マップによって示される領域は局所的になることが望ましいためである．そのため，2.2 節で述べたように，浅い層からの出力を注意機

構に与えることで，提案手法の目的が達成される．

3.2.3 注意度マップに基づく解析すべき領域の検出

注意度マップからは，入力として与えられたマルウェアのバイナリデータのうちで，そのファミリに特有の領域がどこかという情報が得られる．より詳しく述べると，注意度マップが示すのは，CNN への入力画像において，高い重要度を持つ領域の位置であるが，バイナリデータから画像への変換処理では，その位置の対応関係が保持されるので，注意度マップに対応するバイナリデータ内の領域を検出することができる．

提案手法が検出した領域が，マルウェアのコードセクションにある場合は，該当する位置にある関数を逆アセンブルする．また，検出した領域が，データセクションにある場合は，該当する位置にあるリソースを抽出する．

入力として与えられたマルウェアがパックされている場合には，データセクションにあるリソースが，難読化されたコードである可能性があり，さらなる解析を進めるためには，アンパックする必要がある．そうした場合でも，アンパックによって得られるバイナリデータから，注意度マップに対応する領域を検出することが可能である．これは，1.1.1 項で述べたように，多くのパッカは典型的な圧縮アルゴリズムを用いていることによる．

3.2.4 得られた情報に基づく人手による解析

以上の手順によって，該当するファミリに特有の関数やリソースを検出することが可能となる．さらに，解析者が，それらの関数がどのように動作するか，あるいは，それらのリソースが用いられるかを解析することで，マルウェアの挙動を知ることができる．

さらに，同じファミリのマルウェアに対する詳細な解析結果が利用できる場合は，バイナリデータと注意度マップ同士を比較することによって，重要な関数やリソースの対応関係を見つけることができ，マルウェア解析を更に効率化することができる．

4. 評価

提案手法の有効性を確かめるため，評価実験を行った．具体的には，まず，注意度マップから得られる情報が信頼に足るものかを確かめるために，マルウェア分類の精度を評価した．そして，注意度マップから得られる情報が解析に有用かを確かめるために，注意度マップが検出した関数やリソースの解析を行った．

本章では，それぞれの詳細な手順と結果を述べる．

4.1 データセット

評価実験では，マルウェアデータセットとして，VX Heaven[*2]を用いた．これは，公開されているマルウェアデータセットの中でも，含まれるマルウェアの数が特に大きく，[7] を含む，多くの研究で用いられているためである．また，データセットに含まれるそれぞれのバイナリデータが，どのファミリに属するのかを示す正解データとして，[7] と同様に，Microsoft Security Essentials[*3]による検知結果を利用した．そして，[7] と同様にして，ファイル数が少ないファミリ（ここでは閾値を 60 とした）を削除した結果，542 ファミリ，147,803 ファイルが得られた．

[*2] http://vxheaven.org/

表 3 既存手法と提案手法の分類精度の比較

手法	Top-1 Error	Top-5 Error
提案手法	**50.97%**	**31.34%**
既存手法 [7]	53.53%	41.78%

表 4 既存手法と提案手法の分類結果の比較

		既存手法 [7] 正解	既存手法 [7] 不正解	合計
提案手法	正解	53,977	18,498	72,475
	不正解	14,712	60,616	75,328
合計		68,689	79,114	147,803

4.2 実装

提案手法の実装には，TensowFlow[*4]を用いた．また，比較のためのベースラインとして，scikit-learn[*5]を用いて，GIST 特徴量に基づく k-近傍法による [7] の手法を実装した．なお，k-近傍法における k の値は，[7] に従って 3 とした．そして，どちらについても，5-fold cross-validation によって分類精度を計測した．

4.3 分類精度

評価実験によって得られた分類精度は，**表 3** の通りとなった．また，提案手法と既存手法による分類結果の相違は，**表 4** の通りとなった．以上より，同じマルウェアデータセットを用いた場合でも，GIST 特徴量に基づく k-近傍法による既存手法より，CNN を利用した提案手法の方が，高い分類精度を誇るということが確かめられた．これより，提案手法によって得られる注意度マップが，該当するファミリに特有の領域を検出できていると言える．

4.4 分析

本節では，提案手法で得られる注意度マップの示す情報が，解析を行う上で有用かどうかを確かめるため，3.2.4 項のように，いくつかのマルウェアを人手で解析した結果を示す．なお，マルウェアの解析には，IDA Pro[*6]を用いた．

4.4.1 Worm:Win32/Gaobot の解析

Worm:Win32/Gaobot（Agobot とも呼ばれる）は，2004

[*3] https://www.microsoft.com/security/portal/mmpc/
[*4] https://www.tensorflow.org/
[*5] http://scikit-learn.org/
[*6] https://www.hex-rays.com/products/ida/

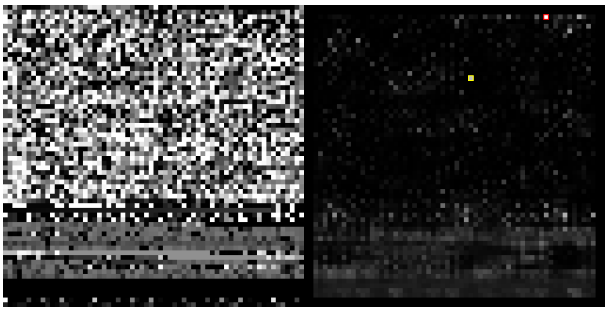

図 5　Backdoor.Win32.Agobot.lt を画像化したバイナリデータ（左半分）と，それに対する注意度マップ（右半分）

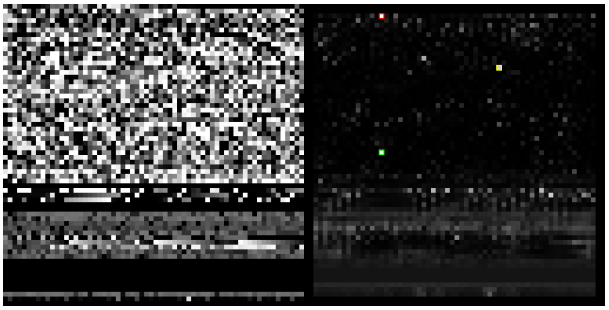

図 6　Backdoor.Win32.Agobot.on を画像化したバイナリデータ（左半分）と，それに対する注意度マップ（右半分）

年ごろに爆発的に広まったボットネットであり [29]，IRC を用いてリモートサーバからのコマンドを実行する他，HTTP/FTP 通信を傍受し，ログイン情報などを取得する機能があることが知られている [30].

図 5 は，Worm:Win32/Gaobot ファミリに含まれるマルウェアの 1 つである，Backdoor.Win32.Agobot.lt を画像化したものと，それに対して提案手法が出力した注意度マップを示している．ここで，このマルウェアを解析すると，注意度マップで最も重要度が高いとされた領域（赤枠）は，リモートサーバからのコマンドを受け取るために，IRC サーバに接続し，チャットルームに入る処理を行う関数 `sub_401356` を指していた．さらに，2 番目に重要度が高いとされた領域（黄枠）は，傍受した HTTP 通信の内容を受け取り，PAYPAL や paypal.com という文字列を含むような，保存すべきデータがあるかをチェックする関数 `sub_410F80` を指していた．

また，**図 6** は，同じく Worm:Win32/Gaobot ファミリに含まれる Backdoor.Win32.Agobot.on を画像化したものと，その注意度マップである．この注意度マップでは，最も重要度が高いとされた領域（赤枠）は，同じく IRC サーバへ接続し，チャットルームへ入室する関数 `sub_401356` を指しており，3 番目に重要度が高いとされた領域（黄枠）は，同じく HTTP 通信の内容に PAYPAL や paypal.com という文字列を含むかをチェックする関数 `sub_4108AC` を指していた．さらに，2 番目に重要度が高いとされた領域（緑枠）は，DDoS 攻撃のために，TCP や GRE 通信を指定の宛先へリダイレクトする関数 `sub_4258B7` を指していた．

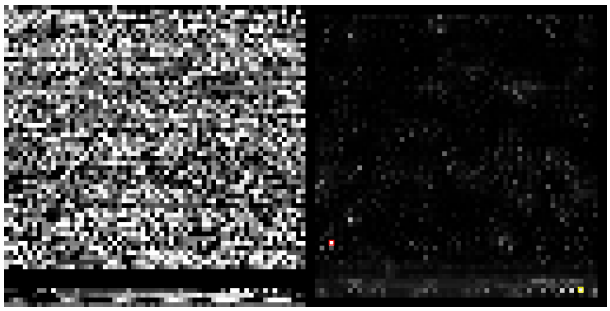

図 7　Trojan-Banker.Win32.Banbra.r を画像化したバイナリデータ（左半分）と，それに対する注意度マップ（右半分）

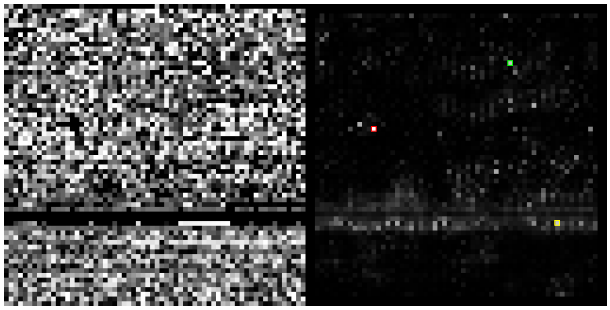

図 8　Trojan-Banker.Win32.Banbra.as を画像化したバイナリデータ（左半分）と，それに対する注意度マップ（右半分）

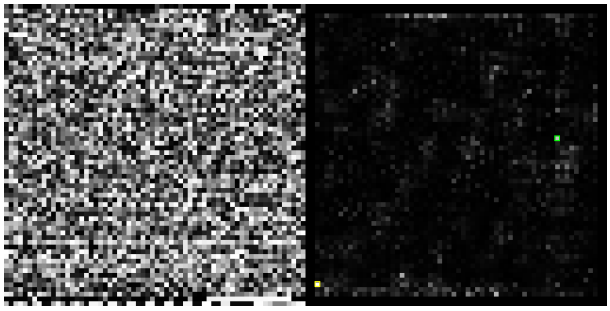

図 9　Trojan-Banker.Win32.Banbra.ghf を画像化したバイナリデータ（左半分）と，それに対する注意度マップ（右半分）

以上より，提案手法によって得られる注意度マップが，該当するファミリに特有の領域を指しており，それらはマルウェアの挙動を解析する上で有用であることが確かめられた．さらに，マルウェアによって，それらの領域の位置が変化していても，提案手法は対応できていることが確かめられた．また，検出された領域には，異なるマルウェア間で対応関係が見られることが確かめられた．

4.4.2　TrojanSpy:Win32/Banker の解析

TrojanSpy:Win32/Banker は，キーボードやマウスの入力を記録し，銀行口座情報を盗みだすことを目的としたマルウェアで [31]，特にブラジルでは，亜種が 2006–2007 年ごろに流行したことが知られている [32].

図 7 は，TrojanSpy:Win32/Banker ファミリに含まれるマルウェアの 1 つである，Trojan-Banker.Win32.Banbra.r を画像化したものと，それに対して提案手法が出力した注意度マップを示している．ここで，このマルウェアを解析すると，注意度マップで最も重要度が高いとされた領域

（赤枠）は，実行されている OS の情報や入力されたキーの履歴，キャプチャしたスクリーンショットなどをメールを通して攻撃者に送信するための関数 sub_480A84 を指していた．さらに，2 番目に重要度が高いとされた領域（黄枠）と，その付近で重要度が高くなっている領域は，Delphi で作成されたバイナリデータに含まれている，ボタン用の「✓」や「✗」といったビットマップを指していた．これは，アンチウイルスソフトウェアを開発している ESET が，ブラジルで流行した銀行口座情報を盗み出すタイプのマルウェアの傾向についてまとめた報告書 [33] で，「ほとんどのサンプルが Delphi を用いていた」と述べられていたことから，TrojanSpy:Win32/Banker ファミリとの関連性が高く，分類において高い重要度を持つ情報となったのではないかと考えられる．

また，**図 8** は，同じく TrojanSpy:Win32/Banker ファミリに含まれる Trojan-Banker.Win32.Banbra.as を画像化したものと，その注意度マップである．この注意度マップでは，最も重要度が高いとされた領域（赤枠）は，同じく傍受した情報をメールを通して攻撃者に送信するための関数 sub_46596C を指しており，2 番目に重要度が高いとされた領域（黄枠）は，同じく Delphi によるボタン用のビットマップを指していた．さらに，3 番目に重要度が高いとされた領域（緑枠）は，入力された情報を記録するために，GetKeyState を用いて特定のキーが押下されているかを取得する関数 sub_42F778 を指していた．

そして，**図 9** は，同じく TrojanSpy:Win32/Banker ファミリに含まれる Trojan-Banker.Win32.Banbra.ghf を画像化したものと，その注意度マップである．このマルウェアは UPX を用いてパックされているが，アンパックすることによって，Delphi で実装されたマルウェアの本体を得ることができる．ここで，注意度マップで最も重要度が高いとされた領域（黄枠）に対応する，アンパック後のマルウェア本体の領域を求めると，同じく Delphi によるボタン用のビットマップを指していた．さらに，2 番目に重要度が高いとされた領域（緑枠）に対応するアンパック後の領域は，同じく GetKeyState を用いて特定のキーが押下されているかを取得する関数 sub_4755F8 を指していた．

以上より，4.4.1 項と同様に，提案手法によって得られる注意度マップが，該当するファミリに特有の領域を指しており，それらはマルウェアの挙動を解析する上で有用であることが確かめられた．さらに，UPX でパックされている場合でも，提案手法は解析に重要な領域を検出でき，また，それらの領域に対応関係が見られることが確かめられた．

4.4.3 TrojanSpy:Win32/BZub.IX の解析

TrojanSpy:Win32/BZub.IX は，ブラウザ内での入力内容を記録し，銀行口座情報や保存されたパスワードなどを盗み取ることを目的としたマルウェアで，ダイナミックリンクライブラリ（DLL）を Internet Explorer の Browser Helper Object（BHO）としてインストールすることで，入力内容を読み取ることが知られている [34]．

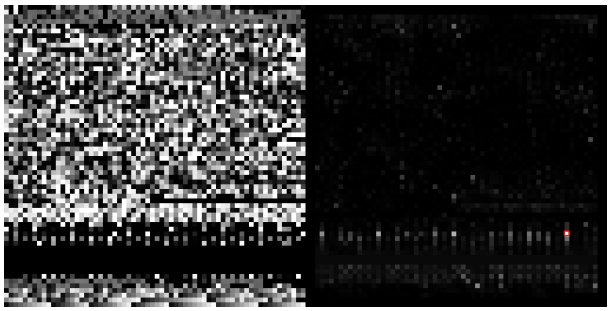

図 10 Trojan-Spy.Win32.BZub.by を画像化したバイナリデータ（左半分）と，それに対する注意度マップ（右半分）

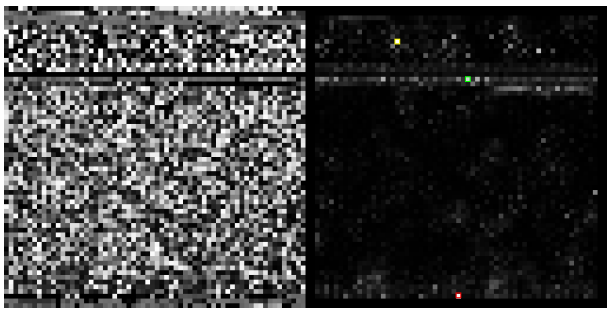

図 11 Trojan-Spy.Win32.BZub.ca を画像化したバイナリデータ（左半分）と，それに対する注意度マップ（右半分）

図 10 は，TrojanSpy:Win32/BZub.IX ファミリに含まれるマルウェアの 1 つである，Trojan-Spy.Win32.BZub.by を画像化したものと，それに対して提案手法が出力した注意度マップを示している．なお，Trojan-Spy.Win32.BZub.by のバイナリデータは，実行ファイルではなく，BHO としてインストールされる DLL 本体となっていた．ここで，このマルウェアを解析すると，注意度マップで最も重要度が高いとされた領域（赤枠）は，傍受した情報を HTTP で攻撃者に送信し，条件に応じて C ドライブ直下のファイルをすべて削除するコマンドを実行する関数 sub_1000F98E の例外処理を行うための構造体 stru_1001CCA8 を指していた．

また，**図 11** は，同じく Trojan:Win32/BZub.IX ファミリに含まれる Trojan-Spy.Win32.BZub.ca を画像化したものと，その注意度マップである．このバイナリデータは，実行ファイルであり，データセクションに含まれる UPX でパックされた DLL をシステムにインストールするという処理を行う．ここで，注意度マップで最も重要度が高いとされた領域（枠）は，DLL が BHO として読み込まれるようにレジストリを変更する関数 sub_402C48 を指しており，3 番目に重要度が高いとされた領域（緑枠）は，データセクションにある，インストールされる DLL のヘッダを指していた．そして，2 番目に重要度の高いとされた領域（赤枠）は，DLL のデータセクションを指しており，バイナリデータから DLL を抽出してアンパック後の対応する領域を求めると，図 10 と同様に，（ファイルを削除する

処理はないものの）傍受した情報を HTTP で攻撃者に送信する関数 sub_10010329 の例外処理を行うための構造体 stru_1001D490 を指していた．

以上より，4.4.1 項，4.4.2 項と同様に，提案手法によって得られる注意度マップが，該当するファミリに特有の領域を指しており，それらはマルウェアの挙動を解析する上で有用であることが確かめられた．さらに，UPX でパックされた DLL を展開するマルウェアという特殊な場合でも，提案手法は解析に重要な領域を検出でき，また，それらの領域に対応関係が見られることが確かめられた．

5. おわりに

本研究では，CNN に注意機構を組み合わせることによって，マルウェアのバイナリデータのうちで，そのファミリに特有の領域を検出し，人手による解析を効率化する手法を提案した．そして，評価実験により，提案手法が，GIST 特徴量に基づく k-近傍法によるマルウェア分類手法 [7] よりも，高い分類精度を示すことを確かめた．さらに，提案手法が出力する注意度マップから得られる情報は，マルウェアの挙動を解析する上で有用であることを確かめた．また，注意度マップは，同じファミリに属する複数のマルウェア間で対応関係が見られたことから，同じファミリのマルウェアに対する詳細な解析結果が利用できる場合は，バイナリデータと注意度マップ同士を比較することによって，重要な関数やリソースの対応関係を見つけることができ，マルウェア解析を更に効率化することができる．

今後は，提案手法の利便性を確かめるために，専門家を対象としたユーザスタディの実施を考えている．

参考文献

[1] Alazab, M. et al.: Cybercrime: The Case of Obfuscated Malware, *ICDS3*, pp. 204–211 (2011).

[2] Moser, A. et al.: Exploring Multiple Execution Paths for Malware Analysis, *S&P*, pp. 231–245 (2007).

[3] Anderson, B. et al.: Automating Reverse Engineering with Machine Learning Techniques, *AISec*, pp. 103–112 (2014).

[4] He, K. et al.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, *ICCV*, pp. 1026–1034 (2015).

[5] Xu, K. et al.: Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, *ICML*, pp. 2048–2057 (2015).

[6] Nataraj, L. et al.: Malware images: visualization and automatic classification, *VizSec*, p. 4 (2011).

[7] Nataraj, L. et al.: A comparative assessment of malware classification using binary texture analysis and dynamic analysis, *AISec*, pp. 21–30 (2011).

[8] Oliva, A. and Torralba, A.: Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope, *Int'l Journal of Computer Vision*, Vol. 42, No. 3, pp. 145–175 (2001).

[9] Kirat, D. et al.: SigMal: a static signal processing based malware triage, *ACSAC*, pp. 89–98 (2013).

[10] Jacob, G. et al.: A Static, Packer-Agnostic Filter to Detect Similar Malware Samples, *DIMVA*, pp. 102–122 (2012).

[11] Saxe, J. and Berlin, K.: Deep neural network based malware detection using two dimensional binary program features, *MALWARE*, pp. 11–20 (2015).

[12] Pascanu, R. et al.: Malware classification with recurrent networks, *ICASSP*, pp. 1916–1920 (2015).

[13] 飛山　駿ほか：Deep Neural Network 多段化によるプロセスの挙動に着目したマルウェア推定手法，*CSS*, pp. 310–317 (2016).

[14] Huang, W. and Stokes, J. W.: MtNet: A Multi-Task Neural Network for Dynamic Malware Classification, *DIMVA*, pp. 399–418 (2016).

[15] 佐藤拓未ほか：Paragraph Vector を用いたマルウェアの亜種推定法，*CSS*, pp. 298–304 (2016).

[16] Chen, X. et al.: Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware, *DSN*, pp. 177–186 (2008).

[17] Oyama, Y.: Trends of anti-analysis operations of malwares observed in API call logs, *Journal of Computer Virology and Hacking Techniques*, Vol. 14, pp. 1–17 (2017).

[18] Farhadi, M. R. et al.: BinClone: Detecting Code Clones in Malware, *SERE*, pp. 78–87 (2014).

[19] Dullein, T. and Rolles, R.: Graph-based comparison of Executable Objects, *SSTIC* (2005).

[20] Gao, D. et al.: BinHunt: Automatically Finding Semantic Differences in Binary Programs, *ICICS*, pp. 238–255 (2008).

[21] Alrabaee, S. et al.: SIGMA: A Semantic Integrated Graph Matching Approach for identifying reused functions in binary code, *Digital Investigation*, Vol. 12, No. Supplement-1, pp. S61–S71 (2015).

[22] 羽田大樹，後藤厚宏：BinGrep: 制御フローグラフの比較を用いた関数の検索によるマルウェア解析の効率化の提案，*CSS*, pp. 676–683 (2016).

[23] Percival, C.: Matching with mismatches and assorted applications, PhD Thesis, University of Oxford, UK (2006).

[24] Guo, Y. et al.: Deep learning for visual understanding: A review, *Neurocomputing*, Vol. 187, pp. 27–48 (2016).

[25] Goodfellow, I. et al.: *Deep Learning*, MIT Press (2016).

[26] Bahdanau, D. et al.: Neural Machine Translation by Jointly Learning to Align and Translate, *ICLR* (2015).

[27] Lin, Z. et al.: A structured self-attentive sentence embedding, *ICLR* (2017).

[28] Razavian, A. S. et al.: CNN Features Off-the-Shelf: An Astounding Baseline for Recognition, *CVPR*, pp. 512–519 (2014).

[29] Liang, Z. et al.: Component similarity based methods for automatic analysis of malicious executables, *VB*, pp. 283–299 (2007).

[30] Goel, S. et al.: Botnets: the anatomy of a case, *Journal of Information Systems Security*, Vol. 1, No. 3, pp. 1–12 (2006).

[31] García-Cervigón, M. and Llinàs, M. M.: Browser function calls modeling for banking malware detection, *CRiSIS*, pp. 1–7 (2012).

[32] Ståhlberg, M.: The Trojan Money Spinner, *VB*, pp. 234–241 (2007).

[33] Porolli, M. and Ramos, P.: CPL Malware in Brazil: Somewhere Between Banking Trojans and Malicious Emails (2015).

[34] Williamson, S. A. et al.: Active malware analysis using stochastic games, *AAMAS*, pp. 29–36 (2012).