The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

転移学習による機械語命令列分類における学習の効率化

Efficient learning in machine language instruction sequence classification by transfer learning

大坪 雄平 *1*2 Yuhei Otsubo　大塚 玲 *2 Akira Otsuka　岩田 吉弘 *1 Yoshihiro Iwata　三村 守 *3 Mamoru Mimura　榊 剛史 *4 Takeshi Sakaki

*1警察庁 National Police Agency　*2情報セキュリティ大学院大学 Institute of Information Security　*3防衛大学校 National Deffense Academy　*4東京大学 University of Tokyo

In this paper, we propose a learning method for a machine language instruction sequence which is difficult to prepare a large-scale data set, e.g., packer classification. We used the learning result of the compiler classification which is easy to prepare a large-scale data set. Our examination result shows that it is possible to classify with accuracy of 92% even if only 10 samples of each class are gathered in 30 class classification.

1. はじめに

深層学習において，高精度な分類の実現には大規模なデータセットが重要な要因の 1 つとされている．過去の我々の研究で，ファイルの断片から実行コードを 99%以上の精度で認識できる手法（o-glasses）を提案した [Otsubo 18]．この手法は，入力データを x86/x86-64 実行コードとみなし，これを固定長命令に変換したものを 1d-CNN に入力している．CNN の局所受容野と重み共有の仕組みを固定長命令に適用させることで 99%という精度が実現されている．過去の研究においては，オープンソースから機械語命令列を生成することにより比較的容易に大規模なデータセットを準備することができた．しかしながら，サンプルを大量に集めることが困難であったり，集められたとしても正確なラベル付けが困難であったり，現実には容易に大規模なデータセットを準備できない問題も数多く存在する．

そこで我々は，機械語命令列の分類問題の中で大規模なデータセットを準備することが困難な分野の問題について，転移学習を活用することにより効率的に学習する手法を提案する．本論文では，データセットの大規模化が困難な分類問題として，パッカー推定を例に実験を行った．その結果，各クラスで 10 サンプルしか準備できない場合においても，比較的容易に大規模なデータセットが準備できるコンパイラ推定問題のデータセットの学習結果を転移学習することで 30 クラス分類で 92.0%の Accuracy で分類できることが明らかとなった．これは，転移学習を用いない場合と比較して 1.7%の精度向上であった．

提案手法は，既存手法と同程度のデータセットを準備できる場合は同程度の性能が得られる．加えて，十分にデータセットを準備できない場合は既存手法より良い性能が得られる可能性がある．そのため，提案手法は既存手法より多様な領域に適用可能であると考えられる．

2. 予備知識

2.1 パッカー

パッカーとは，プログラムをパッキング（暗号化／圧縮）するソフトウェアツールの総称である．マルウェアの多くは，静的解析 *1 を妨害するために，パッキングという技術を用いている．そのため，解析をする際は，解析者はパッキングツールの種類を特定してから，パッキングを解除するための手順を考え，マルウェア本来のコード（以下，オリジナルコード）を抽出しなければならない．さらに，UPX[Markus 96] や Themida[Oreans Technologies] などのパッカーを使用するだけでマルウェアをパッキングすることができる．そのため，攻撃者の労力はほぼない．加えて，多くの種類のパッカーが流通しているため，あるマルウェアに対してどのパッカーが使用されたかはマルウェア取得時点では解析者側には分からない．そのため，取得のたびに使用されているパッカーの特定やアンパッキングをしなければならない．そこで，パッカーの種類を自動的に特定することができれば，オリジナルコードを取り出すまでにかかる手間が削減でき，解析の効率化につながる．また，オリジナルコードを自動で抽出するツールの作成にもつながる可能性がある．

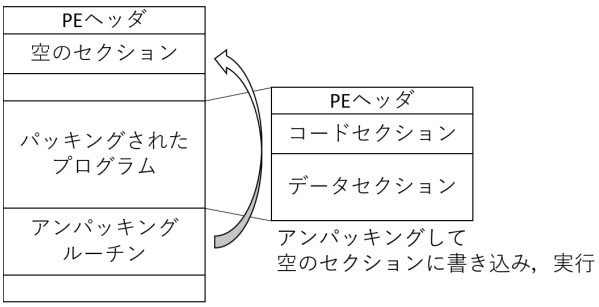

図 1: UPX でパッキングされたプログラムのファイル構造と挙動

パッカーは対象のプログラムをパッキングし，それにアンパッキングルーチンを付加する．このアンパッキングルーチンにより，自己解凍及び実行が可能となっている．例として，UPX でパッキングされたプログラムのファイル構造を図 1 に示す．UPX でパッキングされたプログラムは，PE ヘッダと空のセクション，パッキングされたオリジナルプログラム，アンパッキングルーチンで構成されている．プログラムが実行されると，アンパッキングルーチンが先に実行され，パッキングされたプログラムをアンパッキングし，メモリ上に展開する．アンパッキングされたオリジナルプログラムは空のセクションに書き込まれた後，実行される．

UPX は非常に単純なパッカーだが，Themida や

連絡先: 氏名，所属，住所，電話番号，Fax 番号，電子メールアドレスなど

*1 マルウェアを動作させることなく，プログラミングコードから情報を得る解析

ASProtect[StarForce Technologies Ltd.] などは複雑なパッキングアルゴリズムを有しており，パッキングに使用されたパッカーによりプログラムの動きは異なってくる．

3. 関連研究

パッカー推定によく使われるツールとして PEiD [*2] があげられる．PEiD は非常に高い精度でパッカーを検知できるものの，パターンマッチングを用いていることから，誤検知率を抑えたシグネチャの作成に手間がかかる上に，シグネチャが複雑になる傾向がある．また，新しいパッカーに対応するためには，新しいパッカー用のシグネチャの作成だけでなく，既存のシグネチャを修正する必要もあり，性能の維持にコストがかかる．

上記の課題を解決するため機械学習を用いたパッカー推定について，様々な研究がされている．この中で本論文と最も関連の深いものは伊沢らの手法 [Isawa 14] である．伊沢らは，プログラムのエントリポイント部分（EP）にパッカーの特徴が最も現れると考え，EP から 15 バイト分のバイナリデータを SVM で学習させた．26 種類のパッカーと 3 種類のパッキングされていないプログラムの計 29 クラスを含む約 4,000 のサンプルを用いた実験では，99.46 %という非常に高い Accuracy で分類できることが確認されている．

4. o-glasses

この節では，過去に我々が提案した o-glasses[Otsubo 18] について述べる．この手法は，入力データを x86/x86-64 実行コードとみなし，これを固定長命令に変換したものを畳み込みニューラルネットワーク（CNN [LeCun 89]）に入力している．CNN の局所受容野と重み共有の仕組みを固定長命令に適用させることで，実行コードの分類が 99%という精度で実現されている．

入力データは，2048bit 値配列で，128bit 固定長命令に変換された実行コードを 16 命令分としている．1 層目は 1d-CNN で，フィルタサイズを 128，ストライドを 128 とし，1 命令分の局所受容野を形成している．出力されるチャンネルの深さは 96 である．2 層目も 1d-CNN で，フィルタサイズを 2，ストライドを 1，チャンネルの深さを 256 としている．この層により，2 つの命令間の関係の特徴を得ることを期待している．3〜5 層目は全結合で，ノード数は 400,400，n としている．ここで，n は分類するクラス数である．また，3〜4 層目の入力部分には，ネットワークの学習プロセスをより安定化させ高速化するため，Batch Nomalization[Ioffe 15] を取り入れている．中間層の活性化関数は ReLU を，出力層の活性化関数は Softmax 関数を使用している．

この手法は，実行コードか否かの認識に留まらず，様々な分野に適用できる可能性がある．例えば，この手法を用いて実行コードを生成したコンパイラの推定できるか実験を行ったところ，コンパイラの種類，対象アーキテクチャ，最適化レベル等の 15 クラス分類について，16 命令というコード断片から 95%以上の精度で分類できることが確認されている [大坪 18]．

5. 提案手法

本論文では，パッカーの推定手法として，伊沢らの手法 [Isawa 14] と同様にプログラムの EP に格納されている機械語命令列に着目する．具体的には，EP を起点として L 個の命令を入力データとして o-glasses に入力し，パッカー推定を行う．この EP の機械語命令列に着目したパッカー推定は，前述のコンパイラ推定と比較して学習用データセットの作成について課題がある．コンパイラ推定の場合，1 つのプログラムに含まれる機械語命令すべてをデータセットに組み込めるため，データセットの大規模化は容易である．一方，EP の機械語命令列に着目したパッカー推定の場合，1 つのプログラムから作成できるサンプルは 1 つだけである．加えて，攻撃者が使用しているパッカーの入手が困難な場合がある．その場合，実際の攻撃で観測されたマルウェアからサンプルを作成することとなる．一般的なプログラムと比較してマルウェアの入手はより困難であることから，データセットの大規模化は更に困難なものとなる．

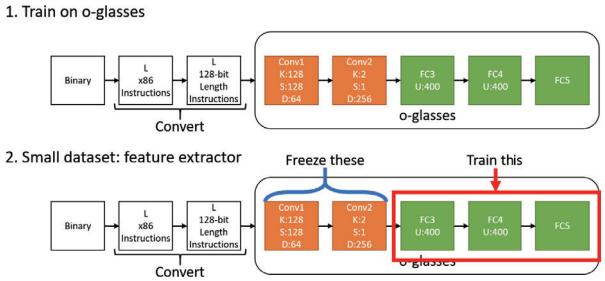

図 2: 提案手法の概要

そこで我々は o-glasses の学習フェーズで転移学習を行う．転移学習は，ある領域で学習させたモデルを別の領域に適応させる技術のことであり，少ないデータで精度の高い学習結果を得ることができるというメリットがある．提案手法の概要を図 2 に示す．

具体的には以下の手順で学習を行う．まずは，o-glasses のネットワーク全体でコンパイラ推定用のデータセットを用いて学習を行う．我々はこの学習済みネットワークに，以下の特徴を有していることを期待している．

- 1 層目の CNN で 1 命令分の局所受容野の形成
- 2 層目の CNN で 2 つの命令間の特徴を得る
- プログラムとして成立する命令列か否かの判別

この学習済みネットワークの第 1 層と第 2 層の CNN における重み（パラメータ）を固定し，パッカー推定用のデータセットの学習を行う．パッカー推定用のデータセットはコンパイラ推定用のデータセットと比較して非常に小規模なものであるが，転移学習を用いることで効率的に学習を行うことが期待される．

6. 評価実験

提案手法の有効性を評価するため，以下の手順で実験を行った．

6.1 コンパイラ推定用のデータセットによる事前学習

コンパイラ推定用のデータセットの概要を表 1 に示す．このデータセットは，オープンソースから誰でも収集することができ，大規模なデータセットを容易に作成できる．具体的には以下の手順で作成した．GitHub[*3] 上でオープンソースソフト

*2 最新のバージョン 0.95 は次の Web サイトからダウンロード可能：https://www.softpedia.com/

*3 https://github.com/

表 1: Number of elements in each of our datasets

Label	Compiler	Arch.	Opt.	File	Code
Program	VC2017	32bit	None	1,170	369,605
			Max	1,147	255,143
		64bit	None	1,456	540,568
			Max	1,242	542,020
	VC2003	32bit	None	1,350	292,277
			Max	1,306	270,743
		64bit	None	-	-
			Max	-	-
	GCC	32bit	None	2,111	227,004
			Max	1,844	239,821
		64bit	None	1,582	283,276
			Max	1,580	287,775
	Clang	32bit	None	1,205	101,024
			Max	1,196	86,521
		64bit	None	1,892	332,278
			Max	1,883	246,500
	ICC	32bit	None	1,761	1,494,677
			Max	1,724	1,161,499
		64bit	None	1,796	1,419,705
			Max	1,728	1,046,958
Others				130	912,958
Total				28,103	10,110,352

ウェアを収集し，複数のコンパイラ（VC[*4], GCC[*5], Clang[*6], ICC[*7]）で最適化レベルを「なし」と「最大」に変えながらコンパイルを行った．VC2003 は 32bit のみでコンパイルし，それ以外は 64bit および 32bit の両方でコンパイルした．コンパイルに成功したオブジェクトファイルのヘッダ情報を元に実行コード部分のみを取り出し，先頭から順番に L 個の命令ずつ切り出すことでデータセットを作成した．表 1 は $L = 16$ のときのデータセットの概要を示している．また，入力データが実行コードとみなせない機械語命令列にも対応するため，「Others」とラベル付けしたデータセットも作成した．これは，文書ファイルを元に作成した．文書ファイルは，文章，画像，メタデータなど様々な種類のデータが含まれている．その一方で，この文書ファイルがマルウェアでなければ，実行コードが含まれている可能性は極めて少ないと思われる．そこで，我々は検索エンジンを用いて「rtf」,「doc」,「docx」および「pdf」の 4 種類のファイルを収集し，VirusTotal[*8] を利用し，ウイルス対策ソフトでそのファイルがマルウェアとして検知されないことを確認した．その上で，文書ファイルを機械語命令列の集合とみなして，先頭から順番に強制的に逆アセンブルして L 個の命令ずつデータを切り出すことで，「Others」とラベル付けしたデータセットを作成した．

$L = 16$ の場合．19 クラスで延べ 28,103 個のファイルから合計 10,110,352 個の訓練データができた．このデータセットに含まれる命令の長さの平均を調べたところ，「Program」カテゴリで 3.69 バイト，「Others」カテゴリで 2.38 バイトであった．各クラス毎のサンプル数に偏りがあるため，学習に使用する各クラス毎のサンプル数の上限を 10 万 $\times L \div 16$ に設定し，$L = 16$ のときは、19 クラスで約 190 万のサンプルで事前学習を行った．なお，事前学習の epoch 数は 50 を選択した．

∗4 Microsoft Visual C++: 2003 and 2017
∗5 the GNU Compiler Collection: 6.3.0
∗6 a C language family frontend for LLVM: 5.0.2
∗7 Intel C++ Compiler: 19.0.0.117 Build 20180804
∗8 https://www.virustotal.com/

表 2: クラス分類結果

分類器	Accuracy	入力データ	クラス数
PEiD	66.53	シグネチャベース	29
伊沢ら SVM	99.46	EP から 15 バイト	29
提案手法	98.89	EP から 16 命令	30
提案手法	98.94	EP から 8 命令	30
提案手法	99.21	EP から 4 命令	30

6.2 既存手法と提案手法の性能比較

コンパイラ推定用のデータセットで学習した o-glasses のネットワークの 1 層目と 2 層目の重みを固定し，パッカー推定用データセットによる転移学習を行った．なお，転移学習の epoch 数は 500 を選択した．

データセットおよび既存手法との比較結果の概要を表 2 および表 3 に示す．

パッカー推定用データセットには 27 種類のパッカーでパッキングされたプログラムとパックする前のオリジナルのプログラム 3 種類（32bit，64bit および.NET）の計 30 のクラスが含まれる．表 3 の「Num.(1)」は準備したプログラムの数であり，「Num.(2)」は伊沢らが正常に動作（アンパッキングおよび実行）することを確認できたプログラムの数である．「PEiD」はパターンマッチングを用いたパッカー等を検知するツールであり，「伊沢ら SVM」は関連研究で取り上げた手法 [Isawa 14] である．この 2 つの既存手法の実験は「Num.(2)」に該当するデータを 10 分割交差検証で実験したものであり，論文 [Isawa 14] から実験結果を引用した．「Num.(2)」以外のプログラムについて，我々が確認したところ，全てではないが複数のプログラムで正常動作を確認することができた．そこで，提案手法の実験については，データセットの規模を大きくするため，正常に動作しなかったものも含む「Num.(1)」に該当するデータを用いた．このデータで伊沢らと同様に 10 分割交差検証で各クラスの Precision（P）および Recall（R）を求めた．

実験の結果，同規模のデータセットを使用した場合，提案手法は既存の伊沢らの手法とほぼ同等の性能を獲得していることが明らかとなった．

6.3 小規模データセットを用いた実験

より小規模データセットにおける提案手法の性能を評価するため，10 分割交差検証に使用する各クラスのサンプル数の上限を S とし，S を変化させた場合の Accuracy の変化を求めた．実験の結果を表 4 に示す．実験の結果，$S = 10, 20$ のような非常に小規模なデータセットにおいて，転移学習を用いる方が高い Accuracy でパッカーを分類できることが明らかとなった．

7. おわりに

本論文では，機械語命令列の分類問題について，容易に大規模なデータセットを準備できる分類問題の学習結果を転移学習することにより，大規模なデータセットの準備が困難な分類問題を効率的に学習する手法を提案した．データセットの準備が困難な分類問題の例としてパッカー推定の実験を行った結果，転移学習を活用することにより，各クラスのサンプル数が 10 個しか集まらない場合でも 92%という精度で分類できることを示した．

提案手法は，既存手法と同程度のデータセットを準備できる場合は同程度の性能が得られる．加えて，十分にデータセット

表 3: クラス分類結果. (Num.): サンプル数, (P): precision, (R): recall, (NaN): PEiD が未対応のもの.

		Num.		PEiD		伊沢ら SVM		提案手法 (L=16)		提案手法 (L=8)		提案手法 (L=4)	
No.	Class	(1)	(2)	P	R	P	R	P	R	P	R	P	R
1	Unpacked(64bit)	475	432	NaN	0.0	100.0	99.8	99.8	100.0	100.0	99.6	100.0	99.8
2	Armadillo 4.00.0053	203	203	100.0	99.0	98.5	100.0	99.0	100.0	98.2	100.0	99.0	100.0
3	MoleboxPro 2.6.4	203	195	100.0	99.0	100.0	100.0	99.1	100.0	99.5	100.0	100.0	100.0
4	Themida 1.8.5.5	195	194	100.0	99.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
5	PESpin 1.330	194	191	100.0	99.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
6	obsidium 1.3.5.4	191	188	NaN	0.0	100.0	100.0	98.0	100.0	99.5	100.0	99.0	100.0
7	ASProtect 2.100	188	185	100.0	98.9	100.0	100.0	99.5	100.0	99.5	100.0	100.0	100.0
8	yoda's protector 1.020	194	170	100.0	98.9	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
9	Unpacked(32bit)	212	165	98.1	89.4	95.3	94.7	96.8	93.8	95.7	94.3	96.3	95.3
10	obsidium 1.4.5	165	143	NaN	0.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
11	nPack 1.1.300	201	141	100.0	11.2	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
12	eXPressor 1.5.0.1	182	139	NaN	0.0	100.0	100.0	100.0	100.0	100.0	100.0	99.5	100.0
13	ASPack 2.12	194	127	100.0	98.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
14	PECompact 2.64	201	126	100.0	98.4	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
15	Unpacked(32bit&.net)	144	122	100.0	95.2	100.0	99.2	100.0	98.6	100.0	98.6	100.0	98.6
16	NsPack 3.7	203	116	100.0	98.4	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
17	RLPack 1.2	200	116	NaN	0.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
18	FSG 2.0	203	111	100.0	99.1	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
19	UPX 3.08	194	99	97.4	99.1	98.2	99.1	99.5	100.0	99.5	100.0	98.5	100.0
20	ASPack 2.11	147	99	100.0	98.0	100.0	100.0	99.3	100.0	96.9	100.0	100.0	100.0
21	Mew11SE 1.2	189	88	100.0	98.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
22	ACProtect pro 1.32	88	75	98.7	87.5	95.6	98.9	87.1	92.1	88.8	92.2	92.8	97.8
23	WWPack32 1.2	108	64	100.0	98.7	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
24	AntiCrack 1.32Pro	64	64	NaN	0.0	100.0	95.3	91.0	77.9	90.1	81.4	98.6	90.7
25	PETITE 2.2	90	63	100.0	95.2	96.8	96.8	98.2	97.8	99.0	97.8	99.0	97.8
26	exe32pack 1.4.2	97	62	100.0	96.8	100.0	98.4	100.0	98.9	100.0	98.9	100.0	98.9
27	tElock 0.98	74	55	100.0	98.2	100.0	100.0	98.9	100.0	98.9	100.0	100.0	100.0
28	PKLITE 32	20	20	100.0	85.0	90.0	90.0	93.3	90.0	91.7	90.0	93.3	90.0
29	Upack 0.39	201	10	100.0	90.0	100.0	100.0	96.7	97.0	97.1	97.0	97.2	96.5
30	WinUpack 0.31beta	197	-	-	-	-	-	97.1	95.3	97.0	95.4	96.6	95.9

表 4: o-glasses における転移学習の評価実験結果

	Accuracy				
S	10	20	40	80	制限なし
転移学習あり	92.0	95.8	97.3	98.2	99.2
転移学習なし	90.3	94.3	98.0	98.2	99.1

を準備できない場合は既存手法より良い性能が得られる可能性がある．そのため，提案手法は既存手法より多様な領域に適用可能であると考えられる．

謝辞

本研究では，パッカーのデータセットを国立研究開発法人情報通信機構サイバーセキュリティ研究所の伊沢亮一博士に提供いただいた．ここに感謝する．

参考文献

[Ioffe 15] Ioffe, S. and Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift, *arXiv preprint arXiv:1502.03167* (2015)

[Isawa 14] Isawa, R., Ban, T., Guo, S., Inoue, D., and Nakao, K.: An Accurate Packer Identification Method Using Support Vector Machine, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 97, No. 1, pp. 253–263 (2014)

[LeCun 89] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D.: Backpropagation applied to handwritten zip code recognition, *Neural Computation*, Vol. 1, No. 4, pp. 541–551 (1989)

[Markus 96] Markus, F. O., Lszl, M., and John, F. R.: the Ultimate Packer for eXecutables, `https://upx.github.io/` (1996)

[Oreans Technologies] Oreans Technologies, : Themida, `https://www.oreans.com/themida.php`

[Otsubo 18] Otsubo, Y., Otsuka, A., Mimura, M., Sakaki, T., and Goto, A.: o-glasses: Visualizing x86 Code from Binary Using a 1d-CNN, *ArXiv e-prints*, p. arXiv:1806.05328 (2018)

[StarForce Technologies Ltd.] StarForce Technologies Ltd., : ASPack Software, `http://www.aspack.com/asprotect32.html`

[大坪 18] 大坪 雄平, 大塚 玲, 三村 守, 榊 剛史, 受川 弘, 岩田 吉弘 F コード断片からのコンパイラ推定手法, コンピュータセキュリティシンポジウム 2018(CSS2018), pp. 4C2–1 (2018)