基于融合技术的恶意程序检测方法的研究

叶　何[1]　赵国梁[1]　胡浩然[1]　文伟平[1]
（1. 北京大学软件与微电子学院，　北京　102600　中国）

摘要：反恶意软件目前面临的一个主要挑战就是有大量的数据文件需要被检测和评估是否有潜在的恶意企图。目前常见的分类方法预测的各个指标都比较低，主要原因之一就是恶意程序的作者为了逃避检测，将属于相同恶意家族和相同恶意行为的程序，经常性的修改或者混淆，使得他们看上去像不同的文件。本文设计一种基于深度学习的分类框架。该框架从恶意程序文件中构造和提取多种特征比如：n-gram、byte-frequency、asm-image 等。然后，尝试了两种特征融合的方式，构建深度神经网络的模型训练出 Concatenate、Multi-input 两个模型。最后，尝试了两种模型融合的方式，一种是对 Concatenate 和 Multi-input 模型进行模型堆叠（stacking）融合得到 Stacking 模型。另外一种是对 Concatenate、Multi-input、Stacking 三种模型进行加权平均数的融合，最后构成恶意程序分类的整体框架。实验结果表明，同时采用多特征和多模型融合技术检测恶意 Windows 程序时，在多个评价指标上都要优于单个特征和单个模型的预测。

关键词：深度学习；特征融合；模型融合；恶意程序
中图分类号：TP 520.2060

Design and Implementation of a Ensemble Method Based Windows Malware Classification Framework

YE He[1]　*ZHAO Guoliang*[1]　*HU Haoran*[1]　*WEN Weiping*[1]
（1. 2. 3. 4. School of software & microelectronics, Peking University, Beijing 102600 China）

Abstract: One of the main challenges facing anti-malware is that there are a large number of files that need to be tested and evaluated for potential malicious intent. The current classification method predicts that each indicator is relatively low. One of the main reasons is that the author of the malicious programs will modify or confuse the programs to the same malicious family and

作者简介：叶何（1998－），男（汉），安徽省，硕士。
通信作者：文伟平（1976－），男（汉），湖南省，教授，E-mail：weipingwen@ss.pku.edu.cn

the same malicious behavior in order to evade detection, making them look like different files. This paper designs a classification framework based on deep learning. The framework constructs and extracts various features such as: n-gram, byte-frequency, asm-image, etc. from malicious program files. Then, the two methods of feature fusion were tried. The model of deep neural network was constructed to train two models, Concatenate and Multi-input. Finally, two ways of model fusion are tried. One is to perform stacking fusion of Concatenate and Multi-input models to get Stacking model. The other is the fusion of the weighted average of Concatenate, Multi-input and Stacking models, and finally constitutes the overall framework of malicious program classification. The experimental results show that when using multi-feature and multi-model fusion technology to detect malicious Windows programs, it is better than single feature and single model prediction in multiple evaluation indicators.

Key Words: Deep learning; Feature fusion; Model ensemble; Malware

引 言

大约在上世纪 80 年代,最早一批的恶意代码开始出现,随之也就诞生了反恶意代码软件。早期恶意代码检测技术主要采用单一特征匹配方法,通过提取恶意代码中的特征串来进行检测。为了逃避这种检测,恶意程序的设计者们对恶意代码采用了各种变形技术使其在传播过程中避免被检测到,各种变形技术的使用也导致了恶意代码总体数量的急剧增加。在这样的情况之下广谱特征码技术开始出现,通过对特征码进行分段,掩码字节能够对需要进行比较的区段和不需要进行比较的区段做出划分。当然,这种技术都是首先要对恶意代码样本进行特征提取之后才可以进行检测,这样的技术有明显的滞后性也就是当恶意程序被发现之后,才能进行分析检测。为了能够对未知病毒和病毒变种也能做出检测,启发式扫描的技术开始出现。启发式扫描的技术会将已有的知识和经验结合到一起来对未知的二进制代码进行检测。恶意程序代码具有一些普通二进制文件所不具有的恶意行为能够被这种技术抓住,例如终结自身、非常规读写文件等。

随着恶意代码设计技术的进步,反恶意代码程序技术也随之在发展,越来越多的安全厂商也开始使用云查杀、主动防御等技术,但很多时候静态的检测方法仍然很高效率,所以,依然是被运用最为广泛的恶意代码查杀技术。

反恶意软件目前面临的一个主要挑战就是有大量的数据文件需要被检测和评估是否有潜在的恶意企图。例如,微软就需要实时的监测全球超过 1.6 亿台的计算机,每月的检测量更是超过了 7 亿台。拥有如此众多的不同文件需要被检测和评估,主要原因之一就是恶意程序设计者为了逃避检测,将属于相同恶意家族和相同恶意行为的程序,经常性的修改或者混淆,使得他们看上去像不同的文件。

从而影响现有的分类程序的准确性。

1 相关工作

Matthew G. Schultz 和 Eleazar Eskin[1] 等人第一个提出了将机器学习的方法运用到恶意程序检测中，并且提出了一个数据挖掘框架，依然是采用静态分析的方法提取出可执行文件中的特征用于分类，实验是在一组恶意和良性的可执行文件样本上训练了多个分类器，用训练好的分类器检测样本外数据，并且与传统基于特征的方法进行比较，实验结果表明机器学习的方法在恶意软件检测的准确率上提升了一倍。

Tony Abou-Assaleh 和 Nick Cercone[2] 等人首次将统计语言模型 n 元语法模型(n-gram)用来提取出恶意程序可执行文件中特定的字节序列，构建恶意程序样本集的字典，用来作为分类特征来表示原来的恶意程序文件。随后，Robert Moskovitch 等人使用了 n-gram 来表示可执行文件中的操作码，用来作为分类的特征。

之前的这些研究主要都是集中在静态分析的方法上。Andreas Moser[3] 等人指出了静态分析方法本身固有的缺陷，提出让程序运行起来对程序运行起来的行为特征进行分析，因为程序的静态特征可以使用混淆、加壳等方法进行各种的变化，但是只要程序运行起来了，恶意程序一定就会有一些本质的共性行为特征出现，而这些特征就可以作为实际分类检测的关键。

随后 J-Y. Xu[4] 等人指出应用程序在实际的运行中主要通过 API 来调用资源与系统进行交互，所以提出了使用程序在实际运行中 API 的调用序列来间接的反映程序的行为特征。随后，Faraz Ahmed[5] 等人对该思想进行了深入的研究，提出通过提取程序所调用的 API 序列信息，把这些序列信息作为模型的输入进行训练学习。

目前，在恶意程序检测上使用比较多的依然是 n-gram，Windows API 等文件，而且在特征上往往只会采用单个特征，在模型上依然是以传统的学习模型为主。本文提出了多特征融合和多模型融合相结合的方式，分类模型上使用了深度模型——CNN(Convolutional Neural Network)，并且与单个特征和单个模型的基准实验进行了对比，以此来说明本论文中提出的多特征和多模型融合分类框架的有效性。

2 基于模型融合的检测方法

本文设计的恶意程序分类框架结合了特征融合和模型融合的方法。其中，特

征融合部分采用了两种方式,一种是将 4-gram[6], byte-frequency, asm-image 三种特征直接拼接作为神经网络的输入进行训练;另一种是将三种特征分别作为网络的输入进行训练,三个输入分别经过卷积层、池化层自动提取完特征之后再次拼接到一起作为全连接层的输入进行训练。模型融合部分采用了两种方式,一种是对第一种特征融合训练得到的模型 Concatenate,和第二种特征融合训练得到的模型 Multi-input 进行堆叠(stacking)融合得到模型 Stacking,另一种是对 Concatenate, Multi-input, Stacking 三个模型进行加权平均数的融合,整个分类框架将同时包含以上多种不同的融合方法。

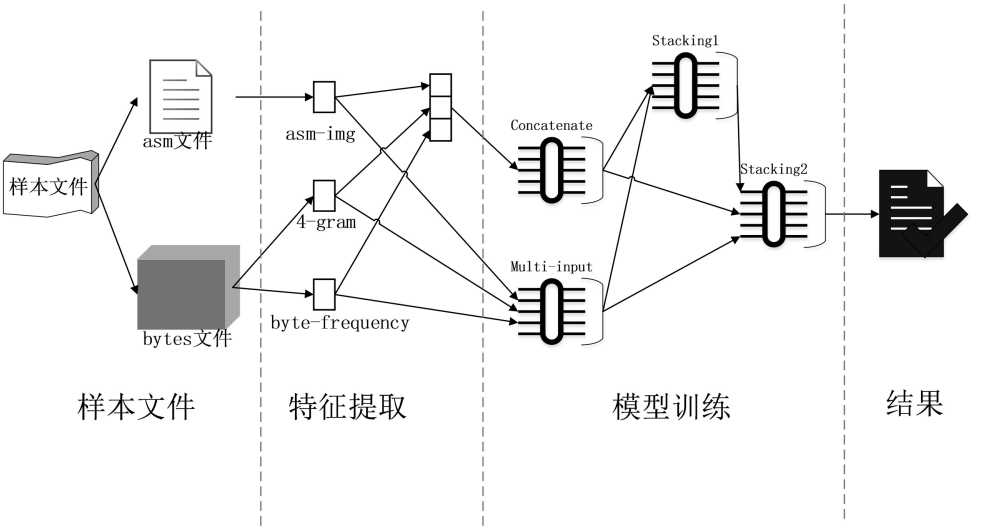

图 1　总体流程

2.1　特征提取

1)4-gram

传统的统计语言模型是对给定长度为 n 的句子,计算其概率分布 $p(w_1,w_2,\cdots,w_n)$,用来表示该句子存在的可能性。概率由以下的公式可以计算得到:

$$p(w_1,w_2,\cdots,w_n)=p(w_1)\times p(w_2|w_1)\times\cdots\times p(w_n|w_1,w_2,\cdots,w_{n-1}) \quad (式\ 1)$$

但是,在实际的计算中如果句子很长计算的难度将会很大,因此 n-gram 模型对以上的计算进行简化,假定第 i 个词的出现仅与其前的 n−1 个词有关,即:

$$p(w_i|w_1,w_2,\cdots,w_{i-1})\approx p(w_i|w_{i-n+1},\cdots,w_{i-1}) \quad (式\ 2)$$

实际计算中,通常采用长度为 n 短语在语料中出现的频率来估计其概率:

$$p(w_i \mid w_{i-n+1},\cdots,w_{i-1}) = \frac{count(w_{i-(n-1)},\cdots,w_{i-1},w_i)}{count(w_{i-(n-1)},\cdots,w_{i-1})} \quad (式\ 3)$$

其中，$count(w_{i-n+1},\cdots,w_{i-1})$表示第 i 个词前 n－1 个词在语料中出现的次数。为了保留句子原有的顺序信息我们希望 n 越大越好，但实际上如果 n 很大，n 元短语在语料中出现的频率也会越低，容易出现数据稀疏的问题。其次，n 如果越大实际的计算量和内存的消耗也会越大。所以，在实际问题当中 n 的选择是要在计算的效率和模型预测性能之间做平衡。

经过多次试验，我们决定取 $n=4$。即将 bytes 文件中每 4 个十六进制的字符作为一个字构造字典，用 4 元模型(4-gram)表示每个恶意程序样本的一种特征。

2)asm-img

通过将 asm 文件的每个字节解释成图像中的一个灰度值[7](即每个像素点是 0－255 之间的整数作为一个特征)，来表示一个恶意样本文件。生成的图像具有非常精细的纹理模式，可以用作每个恶意软件家族的视觉签名，并作为模型的输入。

3) byte-frequency

除了以上两种特征，还可以将 bytes 文件看做一篇文章来对词频进行统计。即对 bytes 文件分别统计 0x00-0xFF，256 个十六进制字符的频数，这样每个 bytes 文件都可以用一个 256 维的整数向量表示作为一种特征。

2.2 模型训练

本文中会设计多个深度神经网络的模型，但每个网路设计都会通过比较模型在训练集和验证集上评价指标的表现，来评估模型的优劣，此过程本身也是一个不断迭代和提升的过程。

1)Concatenate 模型

网络的设计分为三个部分：首先是输入层，Concatenate 网络是单输入，输入的个数和每个输入尺寸可以根据需要自定义；输出层的设计比较简单，可自行选取合适的损失函数(如 softmax)和优化器(如 adam)。最重要的部分是中间隐藏层的设计，隐藏层的设计也可以分为两个部分，浅层的网络部分主要由卷积层、池化层、dropout[9]层和 BatchNormalization[10]层构成，这些层的主要作用就是能够自动提取出有效特征，最后就是全连接层对从浅层网络中提取的特征进行拟合。

将三个特征拼接成一个特征[4-gram，asm-image，byte-frequency]，训练出第一个深度神经网络模型 Concatenate。

2)Multi-input 模型

Multi-input 模型的网络是对 asm-image 特征[11]进行卷积和池化自动提取特征的设计。将三个特征分别作为单独的输入，经过卷积层和池化层，最后将学习到的特征拉直成一维向量。

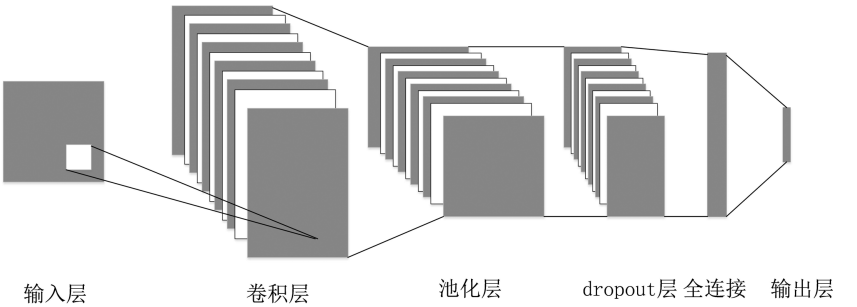

图 2　Concatenate 模型

2.3　模型融合

Stacking 模型的输入可以分为两个部分，分别是 Concatenate，Multi-input 两个模型的输出层，然后将两个模型的输出线性拼接在一起作为 Stacking 融合[12]模型的输入。因为是多分类的任务，新网络的输出层依然是 softmax 层。

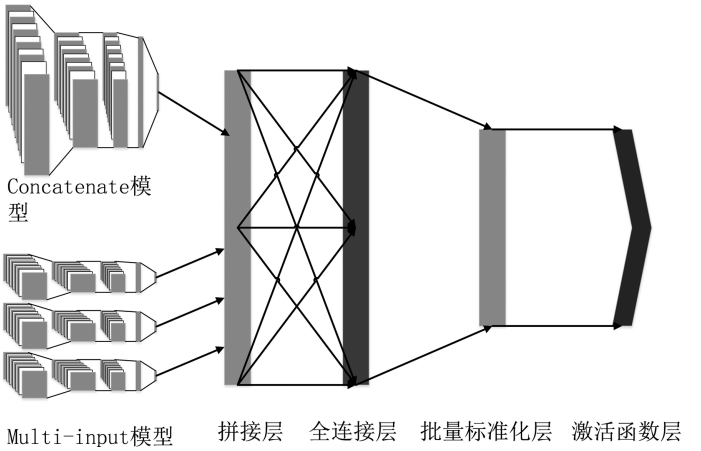

图 3　Stacking 模型

2.4　加权平均

完成以上的模型训练后，为了找到三种模型最好的权重比，我们进行了下面的加权平均处理，同时进行第二次三模型融合。

权重归一化处理：

$$w_i = \frac{\frac{\mu_i}{\sigma_i}}{\sum_{j=1}^{3}\frac{\mu_j}{\sigma_j}}，或者 w_i = \frac{\frac{\mu_i}{\sigma_i^2}}{\sum_{j}^{3}\frac{\mu_j}{\sigma_j^2}} (i = 1, 2, 3) \quad (式 4)$$

其中，j=1，2，3 分别对应模型 Concatenate，Multi-input，Stacking 在某个评价指标上的均值(μ)和标准差(σ)，i=1，2，3 分别对应每个模型归一化之后相应的权重。

模型预测结果融合公式：

$$p_{ij} = w_1 * p_{ij}^{(1)} + w_2 * p_{ij}^{(2)} + w_3 * p_{ij}^{(3)} \quad (式 5)$$

其中，w_i(i=1，2，3)分别是模型 Concatenate，Multi-input，Stacking 前的权重值；$p_{ij}^{(k)}$(k=1，2，3)分别是模型 k 把第 i 个样本预测为类 j 的概率值 p_{ij} 是加权平均以后第 i 个样本预测为类 j 的概率值。

权重构造尝试了两种方式：一种是用均值比上标准差；另一种是均值比上方差。由于，无论从直觉上还是从理论上都无法直接判断出究竟哪种权重构造的方式更为合理，所以，采用了实验的方式通过实际的实验验证来进行判断。因为权重经过了归一化，即 $\sum_{i=1}^{3} w_i = 1$，因此，只需要计算出其中两个模型的权重用 1 减去这两个模型的权重就可以得到另一个模型的权重。

方式一权重公式：

$$w_i = \frac{\mu_i}{\sigma_i} (i = 1, 2, 3) \quad (式 6)$$

方式二权重公式：

$$w_i = \frac{\mu_i}{\sigma_i^2} (i = 1, 2, 3) \quad (式 7)$$

权重归一化处理：

$$w_i = \frac{\frac{\mu_i}{\sigma_i}}{\sum_{j=1}^{3}\frac{\mu_j}{\sigma_j}}，或者 w_i = \frac{\frac{\mu_i}{\sigma_i^2}}{\sum_{j}^{3}\frac{\mu_j}{\sigma_j^2}} (i = 1, 2, 3) \quad (式 8)$$

其中，j=1，2，3 分别对应模型 Concatenate，Multi-input，Stacking 在某个评价指标上的均值(μ)和标准差(σ)，i=1，2，3 分别对应每个模型归一化之后相应的权重。

模型预测结果融合公式：

$$p_{ij} = w_1 \times p_{ij}^{(1)} + w_2 \times p_{ij}^{(2)} + w_3 \times p_{ij}^{(3)} \quad (式 9)$$

其中，w_1(i=1，2，3)分别是模型 Concatenate，Multi-input，Stacking 前的权重值；$p_{ij}^{(k)}$(k=1，2，3)分别是模型 k 把第 i 个样本预测为类 j 的概率值；p_{ij} 是加权平均以后第 i 个样本预测为类 j 的概率值。

3 结果分析

3.1 数据集

微软公司[13]为科学社区提供了前所未有的恶意软件数据集,并鼓励大家将恶意软件文件变体分组到各自的恶意软件族中。Microsoft 恶意软件分类挑战赛发布了近 0.5 TB 的大型数据集,包括超过 20K 恶意软件样本的反汇编和字节码。本文使用了两个部分的数据集:其中,训练集包含 10868 个恶意程序文件,测试集包含 10873 个恶意程序文件,每个恶意程序又对应包括 asm 和 bytes 两个文件。

3.2 评价指标

本论文的研究内容是一个多分类的任务,采用了准确率(accuracy)和对数损失(logloss)两个评价指标。

准确率(accuracy):正确分类的样本数与样本总数的比值,用来评价模型的预测性能。

$$\text{accuracy} = \frac{n_T}{n} \quad (\text{n: 样本总数 } n_T\text{: 正确分类的样本数}) \qquad (式\ 10)$$

对数损失(logloss):分类问题在给数据做标注的时候一般会有两种标注方式:一种是有序标注(1, 2, 3, …)每个数字对应着一个类,这样的类之间往往有大小或者序的关系;另一种十分常见的标注方式就是独热编码,每个样本用一个只包含 0/1 的向量进行表示,每个表示向量只含有一个 1 而 1 出现的位置就是对应的类。同时,分类器的输出也对应着两种情况:一种是类标签,另一种是每个类对应概率值的向量。而对数损失就是用来度量数据独热编码和预测概率值向量之间差异的。

$$\text{logloss} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{C} y_{ij} log\ (p_{ij}) \qquad (式\ 11)$$

n:样本总数 C:类别总数 y_{ij}:第 i 个样本对应的第 j 类为 1,其他为 0 p_{ij}:分类器预测输出的第 i 个样本对应的第 j 类的概率值。

3.3 基准实验效果

随机森林的超参数调节同时使用了网格搜索和随机化搜索,首先会确定一个比较大的超参数空间采用随机化的搜索方式,比如在整个超参空间中随机采集 10 个超参数的组合点。接着使用网格搜索的方式,从随机搜索中搜索到的最佳超参数附近,进行一次完整搜索。将随机搜索和网格搜索结合在一起的方式,可

以做到更好的平衡搜索效率和搜索到超参数的性能。

实验结果如表 1：

表 1　随机森林验证和测试结果

	Validation		Test
Feature	Accuracy	Logloss	Logloss
byte-frequency	0.97571	0.26654	0.15397
asm-image	0.97856	0.17984	0.18866
4-gram	0.99033	0.12090	0.13889

3.4　模型堆叠训练过程

1)将 4-gram，asm-image，byte-frequency 直接拼接在一起得到[4-gram，asm-image，byte-frequency]作为 Concatenate 模型的输入，调节模型的超参数并且训练，将最终训练好的模型保存；

2)将 4-gram，asm-image，byte-frequency 作为 Multi-input 模型的三个输入，调节模型的超参数并且训练，将最终训练好的模型保存；

3)将 Concatenate 模型和 Multi-input 模型的输出层，即 softmax 层拼接在一起作为全连接层的输入，设置网络输出层为 softmax 层，这就是 Stacking 模型的设计；

4)将数据集按照 3:1 的比例随机划分成训练集和验证集两个部分，在验证集上调节网络配置和调整最后全连接层的结构，但只改变全连接层部分，其他网络层不变并且在训练集上训练，将最终训练好的模型保存；

多个模型最终实验结果：

表 2　多个模型训练、验证和测试的结果

		Train		Validation		Test
Model	Epoch	Accuracy	Logloss	Accuracy	Logloss	Logloss
Concatenate	48	0.9995	0.0065	0.9996	0.0016	0.00653
Multi-input	51	1.0000	0.0034	0.9989	0.0025	0.00883
Stacking	83	0.9999	0.0015	0.9989	0.0038	0.00867

整个实验过程分为三个阶段：训练阶段、验证阶段和测试阶段。其中：epoch 指的是完整数据集通过神经网络的次数，accuracy 指的是分类的正确率，logloss 指的是分类交叉熵损失。

3.5 加权平均融合

1)在训练集上做 5-Fold 交叉验证,分别计算出 5 次验证得到地分类准确准率(accuracy)和对数损失(logloss)的均值和标准差。一般认为分类准确率(accuracy)越大、交叉熵损失(logloss)越小说明模型的预测能力越好,相应分配地权重也应该越大,而每次模型预测的标准差越大说明这个模型越不够稳定也越容易因为数据集的扰动而对最终的预测结果产生大的影响,所以相应分配的权重也应该越小。根据以上直觉的想法,来构造权重并且用实验的方式来验证所构造权重的有效性。实验效果如表 3。

表 3 模型交叉验证的结果

	Accuracy		Logloss	
Model	mean(μ)	standard deviation(σ)	mean(μ)	standard deviation(σ)
Concatenate	0.9977	0.000651	0.0108	0.006077
Multi-input	0.9954	0.002045	0.0249	0.013342
Stacking	0.9975	0.000991	0.0152	0.007078

2)权重构造

权重构造尝试了两种方式:一种是用均值比上标准差;另一种是均值比上方差。由于,无论从直觉上还是从理论上都无法直接判断出究竟哪种权重构造的方式更为合理,所以,采用了实验的方式通过实际的实验验证来进行判断。因为权重经过了归一化,即 $\sum_{i=1}^{3} w_i = 1$,因此,只需要计算出其中两个模型的权重用 1 减去这两个模型的权重就可以得到另一个模型的权重。

最终的权重构造结果:

表 4 模型对应权重

	Accuracy		Logloss	
Weight	Concatenate	Multi-input	Concatenate	Multi-input
$\frac{\mu}{\sigma}$	0.5064	0.1608	0.3079	0.3217
$\frac{\mu}{\sigma^2}$	0.6524	0.0659	0.3986	0.1897

3)模型加权平均融合

权重归一化处理:

通过对模型 Concatenate, Multi-input, Stacking 在 Accuracy 和 Logloss 两个指标上的均值(μ)和标准差(σ),来对权重进行归一化处理。然后计算不同模型

对同一个样本预测位同一类的概率作为该模型的权重，对三个模型进行加权平均。最终结果如表5。

表5　模型加权平均的结果

Model	Logloss	Accuracy_std	$Accuracy\_std^2$	Logloss_std	$Logloss\_std^2$
Concatenate	0.006538				
Multi-input	0.008829	0.006453	0.006496	0.006494	0.006540
Stacking	0.008678				

3.6　结果评价

到目前为止我们从每个恶意程序的两个文件 asm 和 bytes 中一共提取了三种特征：4-gram，asm-image，byte-frequency，训练了三个深度神经网络模型：Concatenate，Multi-input，Stacking，采用了两种特征融合的方式：直接将特征拼接在一起作为输入、将三个特征分别作为输入。两种模型融合的方法模型堆叠（stacking）、加权平均结果，总结提出完整的 Windows 恶意程序分类的框架。然后分别在 10873 个未标注的数据集上做测试，从实验的结果看融合后的模型相比较于融合前的单模型，在 logloss 这一评价指标上都有大幅度的提高。

4　结束语

反恶意软件目前面临的一个主要挑战就是有大量的数据文件需要被检测和评估是否有潜在的恶意企图。本文采用了深度卷积神经网络模型同时结合了特征融合和模型融合的技术。通过与基准实验以及在单特征、单模型上的实验比较发现，多特征和多模型融合相结合的技术要优于单特征和单模型的方法。以此来说明本论文中设计的融合框架在 Windows 恶意程序分类这一问题上是有效的。

将来我们会构建更庞大的恶意程序数据集用于模型训练，另一方面尝试构造更多不同和更具可解释性的特征，尝试更多不同的融合方式。同时，由于本框架使用的输入特征并没有太多涉及到恶意程序软件领域本身的相关知识，将来的实践中可以在特征构造上利用更多恶意程序软件相关知识，期待可以构造出更多有效的分类特征。

参 考 文 献(References)

[1] M. G. Schultz, E. Eskin, F. Zadok and S. J. Stolfo, Data mining methods for detection of new malicious executables[J], *Proceedings* 2001 *IEEE Symposium on Security and Privacy*. S&P 2001, Oakland, CA, USA, 2001, pp. 38-49.

[2] Abou-Assaleh T, Cercone N, Keselj V, et al. N-gram-based detection of new malicious code[C]//Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004. IEEE, 2004, 2: 41-42.

[3] Moser A, Kruegel C, Kirda E. Limits of static analysis for malware detection[C]//Twenty-Third AnnualComputer Security Applications Conference (ACSAC 2007). IEEE, 2007: 421-430.

[4] Xu J Y, Sung A H, Chavez P, et al. Polymorphic malicious executable scanner by API sequence analysis[C]//Fourth International Conference on Hybrid Intelligent Systems (HIS'04). IEEE, 2004: 378-383.

[5] Ahmed F, Hameed H, Shafiq M Z, et al. Using spatio-temporal information in API calls with machine learning algorithms for malware detection [C]//Proceedings of the 2nd ACM workshop on Security and artificial intelligence. ACM, 2009: 55-62.

[6] 来斯惟. 基于神经网络的词和文档语义向量表示方法研究[J]. 2016.

[7] Ahmadi M, Ulyanov D, Semenov S, et al. Novel feature extraction, selection and fusion for effectivemalware family classification[C]//Proceedings of the Sixth ACM Conference on Data andApplication Security and Privacy. ACM, 2016: 183-194.

[8] Andrew Ng. Machine Learning Yearning[M].

[9] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research, 2014, 15(1): 1929-1958.

[10] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[J]. 2015.

[11] Kabanga E K, Kim C H. Malware images classification using convolutional neural network[J]. Journal of Computer and Communications, 2017, 6(01): 153.

[12] Yan J, Qi Y, Rao Q. Detecting malware with an ensemble method based on deep neural network[J]. Security and Communication Networks, 2018, 2018.

[13] Ronen R, Radu M, Feuerstein C, et al. Microsoft malware classification challenge[J]. Arxiv preprint Arxiv:1802.10135, 2018.