恶意软件的图像：可视化和自动识别

引言

我们提出了一种简单但是有效的方法，通过图像处理技术，去可视化和识别恶意软件。恶意软件在二进制层面可以被可视化为灰度图像，通过观察到很多恶意软件的家族，他们具有的图像在布局和纹路上非常相似。通过这种视觉上的相似性，一个使用标准图像特征的识别方法被提出了。在这种识别方法中反汇编和代码运行都是不需要的。初步的实验结果表明十分有效，在 25 个不同的恶意软件家族中的 9458 个样本在实验中具有 98%的正确率。

1. 介绍

传统的分析恶意软件的方法包括找出恶意软件的二进制特征。由于恶意软件的快速发展，他们的新特征每年都会有指数级的增长（和 2008 年的 169323 相比，2009 年报道了 2895802 个)。

其他的识别恶意软件的方法包括静态代码分析和动态代码分析。静态分析工作通过反汇编代码并探索可执行的控制流去找出恶意的方式。另一方面，动态分析以在虚拟环境中运行代码并查看基于运行轨迹的行为报告来工作。这两种技术都有他们的优势和劣势。静态分析提供了最完整的覆盖面，但是它经常遭受代码混淆的干扰。在分析前必须解压和解密软件，尽管如此，分析中的困难依然有很高的复杂度。动态分析回更加高效并且不需要去解压或者解密代码。但是他需要大量的时间和资源消耗，因此提高了可扩展风险。并且，一些恶意的行为可能不会被发现因为环境不满足特定的条件。

在这篇论文中，我们带来了一种完全不同并且中立的方式去识别和分析恶意软件。在

最底层，恶意软件可以被二进制用零和一表示。它可以被重塑为矩阵并用图像表示。我们发现特定的视觉相似性在属于相同家族的恶意软件中。这可能可以被解释为相同的在不同软件中被重复利用的代码。在第三节中我们讨论代表性的用二进制表示的恶意软件的图像。在第四节中，我们认为恶意软件识别问题是图像识别问题。现存的识别技巧需要拆卸或识别，但是我们的方式不需要这两种技巧并仍然有显著的表现的提升。此外，它同样对流行的混淆技术具有弹性比如部分加密。这种自动识别技术对于每天收到成百上千恶意软件的反病毒公司和安全员来讲很有价值。

文章剩余的结构如下：在第二节，我们讨论相关的恶意软件可视化和识别工作。在第三节和第四届中，我们描述我们的房发去识别恶意软件和通过图像自动识别他们。在第五节中描述了实验细节。我们在第六节中讨论了我们的方法的限制并在第七节中进行总结。

2. 相关工作

一些工具比如文本编辑器和二进制编辑器都可以可视化和控制二进制数据。最新的，有一些 GUI-based 工具便于文件的比较。然而，他们对可视化恶意软件的研究帮助有限。在 Yoo 里，它使用自组织映射在可执行位置中查找和可视化恶意的代码。Quist 和 Liebrock 为逆向工程开发了一种可视化框架，他们可以识别工作区域和去混淆通过节点-链接可视化，用节点代表地址，链接代表地址间的状态转化。Trinius 等人用树状图和线程图来展示操作分布。Goodall 等人开发了一种视觉分析环境，可以让软件还发着去更好地理解代码。这也展示了如何在软件的环境中展示他们的漏洞。

但他们没有在用数字图像识别恶意软件的方面做更多的工作。Conti 等人 将原始的二进制片段如 c++数据结构，图片的数据，音频数据可视化为图像。他们展示了如何通

过统计特征自动化识别不同的二进制片段。然而他们的分析仅关注于识别基础的二进制片段和非恶意软件。这个工作展现了相似的发现通过用灰度图像展示恶意软件。

一些技巧目的是分类和发现恶意软件。他们包括静态分析和动态分析。我们将回顾具体的解决恶意软件识别的论文。Rieck 等人使用基于根据恶意软件家族的行为分析的特征去识别恶意软件。他们用 10072 个被杀毒软件标记且被分为 14 个恶意软件家族的恶意软件数据集，然后他们在沙箱环境中监视这些恶意软件的行为并生成行为报告。在这份报告中，他们为每个恶意软件生成了基于某些特定字符串的特征向量。一个支持向量机被用来训练和测试这些 14 个家族的特征并且他们的报告中平均正确率为 88%。与之相反的 Tian 等人使用了非常简单的特征，项目的长度去识别其中不同的木马，最后的正确率是 88%。然而，他们的分析仅基于 721 个文件。同一个作者通过使用恶意软件的可打印的字符串信息改良了他们的技术。他们用来自 13 个家族的 1521 个恶意软件测试这个方法并且准确率带到 89.8%。Park 等人基于检测图中最大公共子图去识别恶意软件。他们通过来自 6 个家族的 300 个恶意软件验证了自己的结果。

对于相关工作，我们的识别不需要拆解或运行具体的恶意软件代码。并且，用于分类的图像质感在模糊处理技术特别是加密方面提供了更有弹性的特征。最终，我们苹果我们的方法通过一个更大的数据集包括来自 25 个家族的 9458 个恶意软件。测试结果展示了我们的方式在更少运算小号的同时提供了相似的运算精度。

3.可视化

一个特定的恶意软件二进制会被当作一个 8bit 的无符号数向量并被组织为二维数组来阅读。它可以被可视化为范围在[0,255]之间的灰度图像。图像的宽度是固定的，高度

由文件大小决定，我在表格一中给出不同的文件大小对应的宽度。

图像二展示了一个常见木马下载者的图像：Dontovo A 会下载并执行任意文件。它在很多情况下都很有趣，正如图像二中的，很多不同的恶意软件的区域（二进制片段）展示出独特的条纹。更多基本的二进制片段的分类和他们的灰度可视化图像可以在引用 9 中查看。

.text 片段包含可执行代码，从图中可以看出，它的第一个部分很有质地，剩下的部分被 0 填充（黑色）直到结尾。接下来的.data 片段既包括未初始化的代码（黑色片段）也有已初始化的数据（有纹理的片段）。最后的片段是.rsrc，它包括所有的资源，也许也包括应用会使用的图标。

4.恶意软件分类

图像三展示了来自两个不同家族的恶意软件，可以得出不同家族的不同的恶意软件样本出现视觉上的相似和区别。像前面提到的，他们可能在制作新恶意软件时复用了就的恶意软件的二进制串。这种恶意软件图像视觉上的相似性激励我们去关注经过充分学习的机器视觉进行恶意软件识别。特定的恶意软件家族的图像可以在图像七里看见，各种恶意软件家族具有可分辨的特征。

4.1 图像纹理

什么是视觉纹理没有统一的定义，但它一般被认为是如图像四中那样具有重复的模式。纹理研究的三个主要领域是纹理识别，纹理分析和纹理合成。纹理识别被认为要识别各种均匀的纹理区域在图像中。纹理分割的主要不敌是识别各种纹理区域的边界。纹理合成被用作合成纹理图案，这杯广泛应用于计算机图形学。

纹理分析是计算机视觉的一个重要学习领域。大多是表面可以发现大量的纹理，纹理

分析有广泛的应用包括医学图像分析，远程感知和文档图像处理。之前在图像二和三中展示的恶意软件图像虽然不是完全相同的模式，但是显示出的纹理足够用来自动分类。

4.2 特征向量和分类

很多特征被用来分析纹理，一个最常用的分析方式是分析纹理块的频率成分。标准的方法是把频率分为环（刻度）和楔（方向）并且区域内的特征会被计算。生理学的结果显示人的眼睛通过把图像分为频率和方向向量来分析图像。现在的主流计算机方法去纹理分析是用 Gabor 滤波器。二维 Gabor 函数由具有特定频率和方向的正弦平面组成，该平面由高斯包络调制。Gabor 滤波器具有频率和方向选择器。通过不同的频率和方向，我们得到一个滤波器组。图像通过这个滤波器组，得到基于特征被提取出纹理的被过滤的图像。一个这样的特征是通过计算滤波图像的变换值与小窗口内的平均值的绝对平均偏差来获得的。纹理特征使用过滤器去纹理分割和分类中是成功的。我们使用相似的特征在这篇文件中去描述和分类恶意软件。去计算纹理特征，我们使用 GIST，这种方式在图像中使用小波分解。这个功能在场景分类和对象分类方面取得了成功。每个图像位置由调整到不同方向和比例的滤波器的输出组成。我们使用可操控的八个方向和四个比例的金字塔去表示图像。这个图像的局部表现由公式给出，N=20 是子带的数量。为了捕获全局的图像属性时保留局部信息，我们计算在大空间区域上平均的局部特征幅度的平均值。

公式

W(x)是平均窗口，得到的结果表现是 MxM 分辨率（合理分辨率取为 4）的下采样因此，m 为 MxMxN=320，这是我们使用的 GIST 特征维度的大小。更多的细节解释对

于 GIST 特征可以在引用 12 中被找到。

我们使用具有欧几里得距离的 k 临近分类器进行分类。对于我们所有的测试，我们做了 10 倍的交叉验证，在每次测试中，一个类的随机子集被用于训练和测试。对于每次迭代，从一个类中随机选择 90%的数据用于训练，10%用于测试。因此，给定的测试数据被分类为 k 临近分类器的模式。

KNN 与 CNN 比较实验结果

设计思路

Knn 设计思路：

1. 将灰度图导入并裁切为相同分辨率，根据论文，提取图像的 GIST 特征，然后转化为一维二进制向量。
2. 设置 knn 实验 k=3，交叉验证次数为 10，使用 shuffle=True 泛化打乱数据，设置随机数种子为 42。
3. 进入交叉验证并测试准确率。

Cnn 设计思路：

1. 将灰度图导入并裁切为相同分辨率。
2. 构建卷积神经网络，设置三层卷积层和对应的三层池化层，随后将得到的数组扁平化，导入全连接层进一步处理，最后输出。
3. 在执行中进行 10 次迭代，并在验证集中验证神经网络。

比较不同

1. Knn 是传统的机器学习算法，通过计算样本距离来进行分类，Cnn 是深度学习算法，通过多层的卷积和池化来提取特征。
2. Knn 需要通过人为设置更多的特征才可以提高预测的准确率，Cnn 自动从数据中提取。
3. Knn 处理图像时需要将图像转化为向量进行处理，Cnn 可以直接处理图像信息。
4. Knn 训练速度较快，但预测速度较慢，与 Cnn 相反。
5. Knn 使用交叉验证，每次选择一个折叠作为验证集，其余的作为训练集，Cnn 提前划分出训练集，验证集的比例，随后进行随机分配。

实验结果

Cnn 实验结果：

```
234/234 ━━━━━━━━━━━━━━━━━━━━ 12s 49ms/step - accuracy: 0.7708 - loss: 0.9138 - val_accuracy: 0.9497 - val_loss: 0.1730
Epoch 2/10
234/234 ━━━━━━━━━━━━━━━━━━━━ 11s 48ms/step - accuracy: 0.9526 - loss: 0.1396 - val_accuracy: 0.9673 - val_loss: 0.1031
Epoch 3/10
234/234 ━━━━━━━━━━━━━━━━━━━━ 11s 48ms/step - accuracy: 0.9723 - loss: 0.0837 - val_accuracy: 0.9796 - val_loss: 0.0790
Epoch 4/10
234/234 ━━━━━━━━━━━━━━━━━━━━ 11s 48ms/step - accuracy: 0.9861 - loss: 0.0476 - val_accuracy: 0.9829 - val_loss: 0.0747
Epoch 5/10
234/234 ━━━━━━━━━━━━━━━━━━━━ 11s 48ms/step - accuracy: 0.9892 - loss: 0.0282 - val_accuracy: 0.9823 - val_loss: 0.0581
Epoch 6/10
234/234 ━━━━━━━━━━━━━━━━━━━━ 11s 48ms/step - accuracy: 0.9942 - loss: 0.0157 - val_accuracy: 0.9818 - val_loss: 0.0671
Epoch 7/10
234/234 ━━━━━━━━━━━━━━━━━━━━ 12s 50ms/step - accuracy: 0.9958 - loss: 0.0185 - val_accuracy: 0.9791 - val_loss: 0.0898
Epoch 8/10
234/234 ━━━━━━━━━━━━━━━━━━━━ 12s 50ms/step - accuracy: 0.9967 - loss: 0.0082 - val_accuracy: 0.9839 - val_loss: 0.0773
Epoch 9/10
234/234 ━━━━━━━━━━━━━━━━━━━━ 12s 50ms/step - accuracy: 0.9946 - loss: 0.0193 - val_accuracy: 0.9834 - val_loss: 0.0727
Epoch 10/10
234/234 ━━━━━━━━━━━━━━━━━━━━ 12s 50ms/step - accuracy: 0.9973 - loss: 0.0126 - val_accuracy: 0.9861 - val_loss: 0.0790
59/59 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.9863 - loss: 0.0785
验证集损失: 0.0790, 验证集准确率: 0.9861
```

Knn 实验结果：

```
数据加载完成
加载的图像数量: 9339

第 1 折交叉验证:
模型训练完成!
验证集准确率: 0.987

第 2 折交叉验证:
模型训练完成!
验证集准确率: 0.993

第 3 折交叉验证:
模型训练完成!
验证集准确率: 0.986

第 4 折交叉验证:
模型训练完成!
验证集准确率: 0.984

第 5 折交叉验证:
模型训练完成!
验证集准确率: 0.980

第 6 折交叉验证:
模型训练完成!
验证集准确率: 0.984

第 7 折交叉验证:
模型训练完成!
验证集准确率: 0.986

第 8 折交叉验证:
模型训练完成!
验证集准确率: 0.986

第 9 折交叉验证:
模型训练完成!
验证集准确率: 0.983

第 10 折交叉验证:
模型训练完成!
验证集准确率: 0.981

平均准确率: 0.985
```