机器学习和未知样本检测

——云中的反病毒引擎

毛大鹏

从反病毒引擎说起：

- 从技术上讲：“反病毒引擎”是一套判断特定程序是否为恶意程序或可疑程序的技术机制。
- 反病毒引擎大致有三代：特征码引擎、云引擎和人工智能引擎。

特征码引擎（单机时代）：

- 扫描特征码式反病毒引擎
- 启发式反病毒引擎
- 主动防御式反病毒引擎
- 本地模拟器式反病毒引擎

云引擎（互联网时代）：

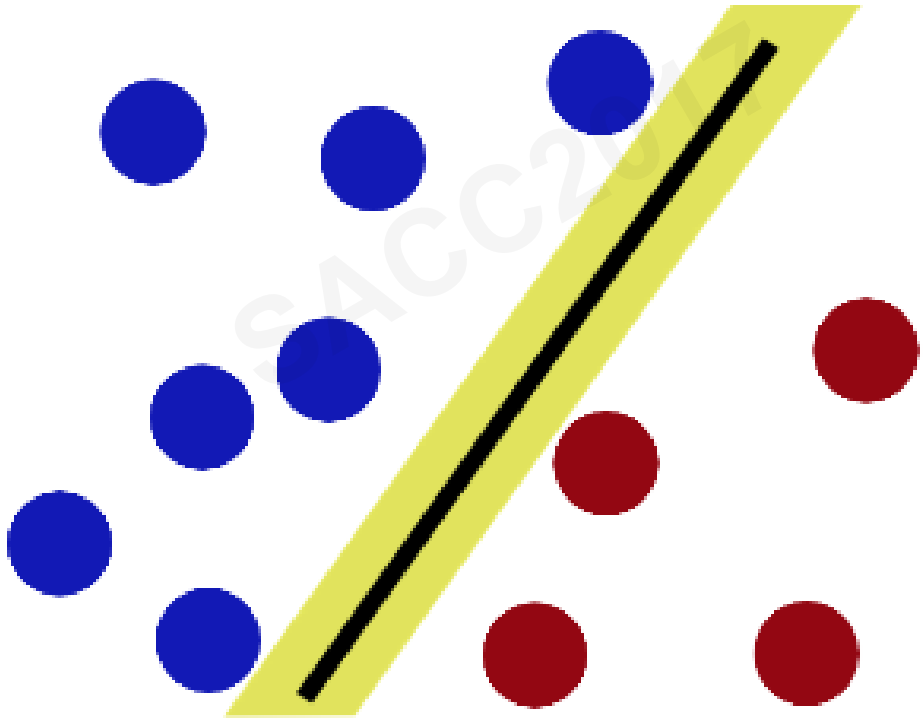
- 云查杀
- 某个客户端发现可疑样本时，将样本发送到云端样本分析集群里进行分析跑测，然后将分析的结果形成特征库再下放到全网客户端。形成一个互联网病毒样本自动处理中心。

人工智能引擎（大数据时代）：

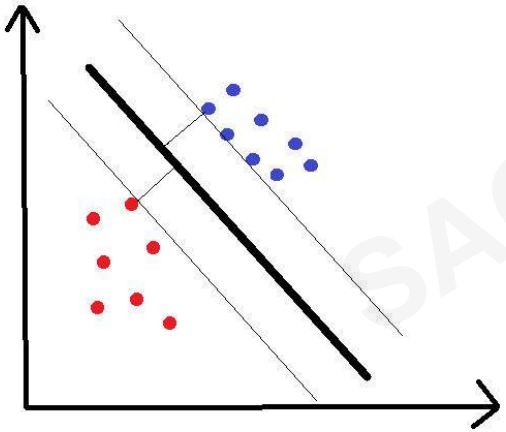
- QVM，它是在Vapnik著作的机器学习经典《Statistical Learning Theory》中的理论基础上进行了创新，首次将机器学习的理论用于未知病毒识别。
- 它的技术原理是先通过对病毒样本的分析和分类形成样本向量和向量机，然后建立一个机器学习的决策机模型，利用决策树和向量机，对大量样本进行学习，从而识别恶意程序。

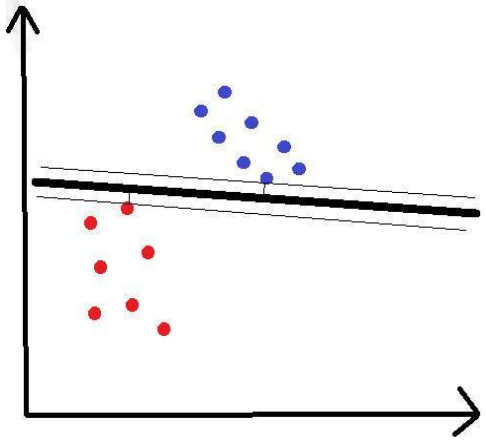
SVM（支持向量机）

SVM就是试图把棍放在最佳位置，好让在棍的两边有尽可能大的间隙。

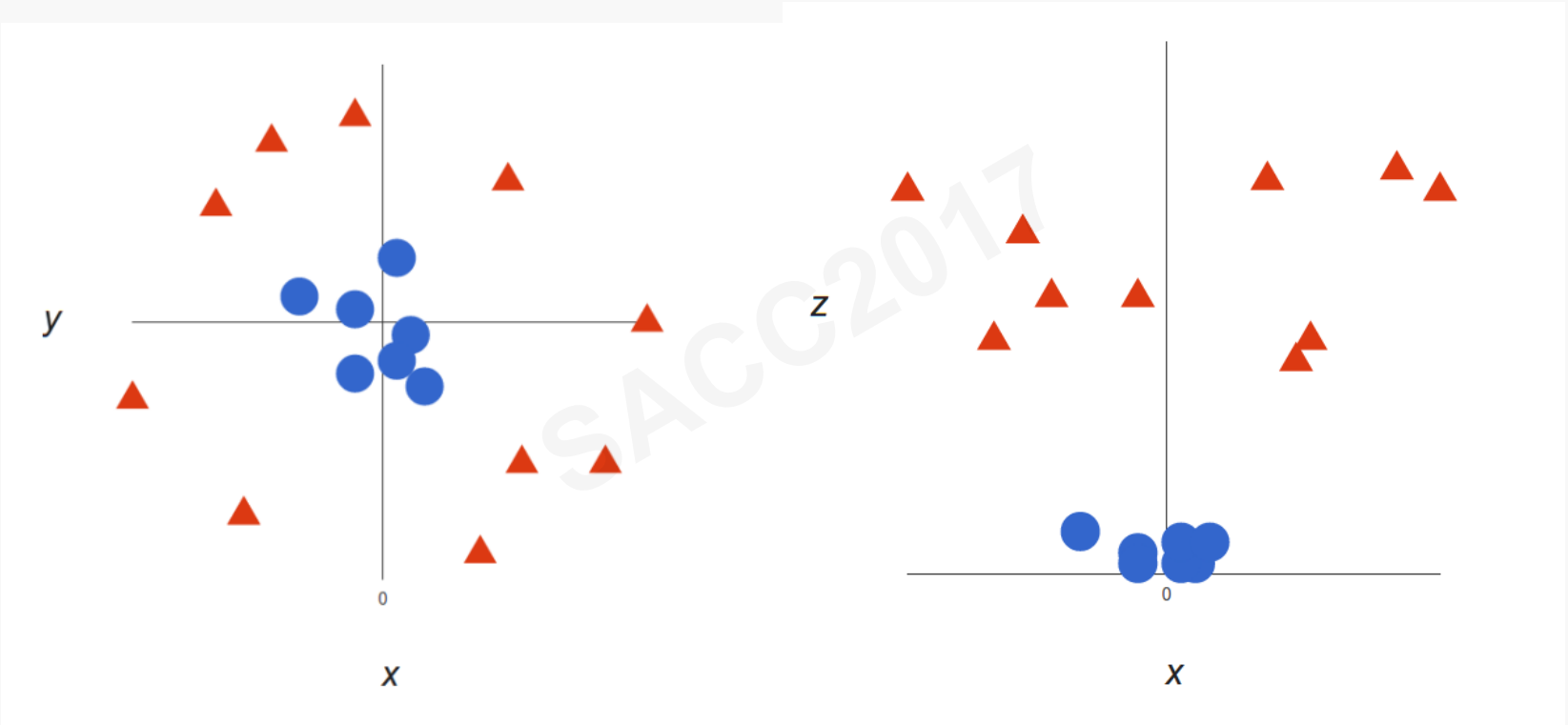


SVM（支持向量机）





SVM（支持向量机）



第三个维度：$z = x^2 + y^2$

人工智能引擎（大数据时代）：

- 支持向量机的核心思想是将特征向量映射到一个高纬空间中，该空间中存在一个最大间隔超平面，空间中的样本点被两个互相平行的超平面隔开，分隔超平面使得两个平行超平面之间的距离最大。平行超平面之间的距离越大，分类器的总误差越小，分类的准确性越高。
- 对于未知样本，以支持向量机为基础的二分类划分方法，有极高的检出率。

新时代（云+移动端）：

- 云服务普及，服务概念得到认可，场景发生了变化：

✓云端：资源数据高度复合体，安全共担模型。

✓移动端：厂商控制紧密，程序上架审核。

云端威胁：

- 系统漏洞攻击
- 内部流量攻击
- 虚拟化攻击
- APT

。 。 。

安全云：

- 提供网络安全保障、数据安全保障、恶意程序查杀、威胁情报等安全功能的云。
- 用户不应该是安全的买单者。云服务商应提供安全保障。
- 在恶意程序查杀上，基于云端大数据和机器深度学习技术可以轻松构建一个云中的反病毒引擎。

云中的反病毒引擎：

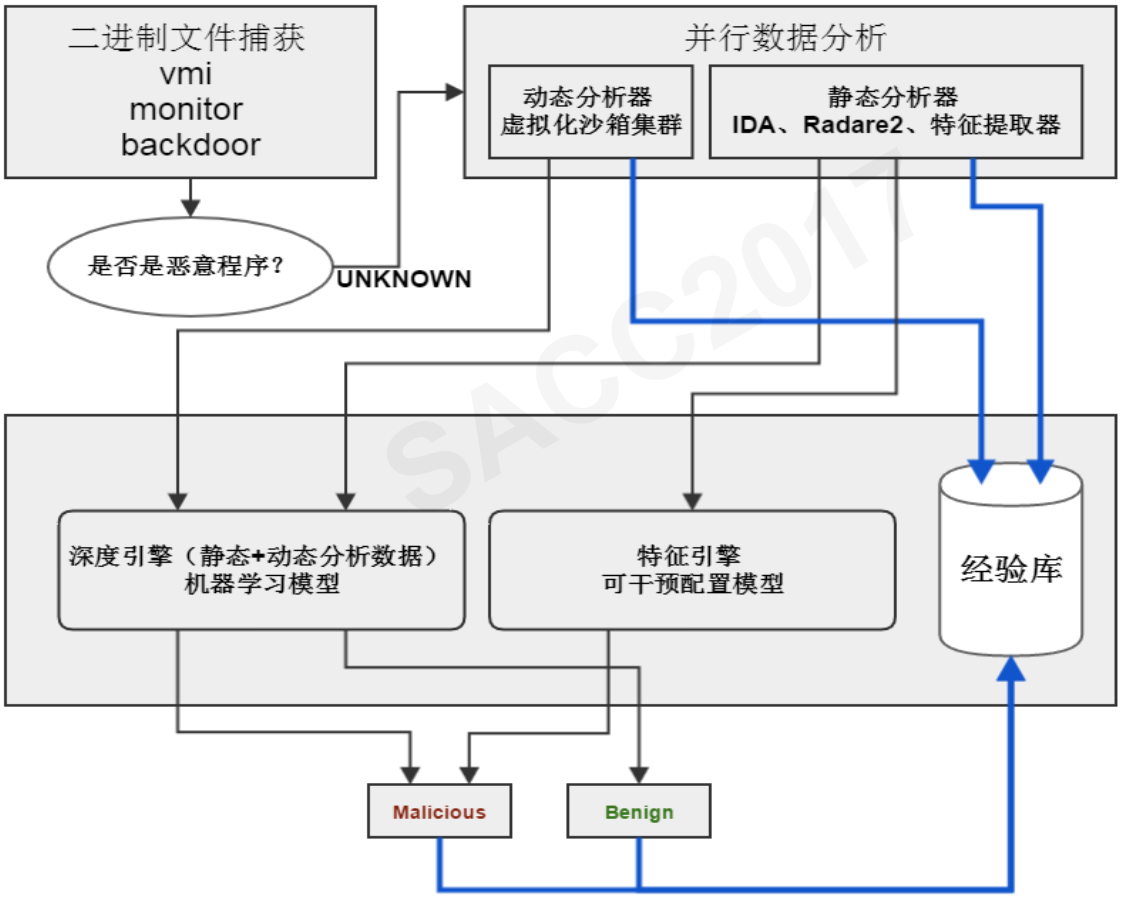
- 云是计算机资源的集中体，基于云的强大计算能力和信息收集能力，可以将自身数据转化成威胁情报信息，再对这些信息进行数据挖掘分析，然后利用特征引擎和深度机器学习技术，可以将反病毒能力提升到一个前所未有的级别。
- 这种与云密切结合的引擎我们定义为安全云引擎。

安全云引擎：

宏观流程图：


数据源
http://www.
恶意程序识别与发现
分析
特征
学习
计算
Hadoop
APACHE
Spark
NVIDIA

大数据+深度学习=安全云引擎

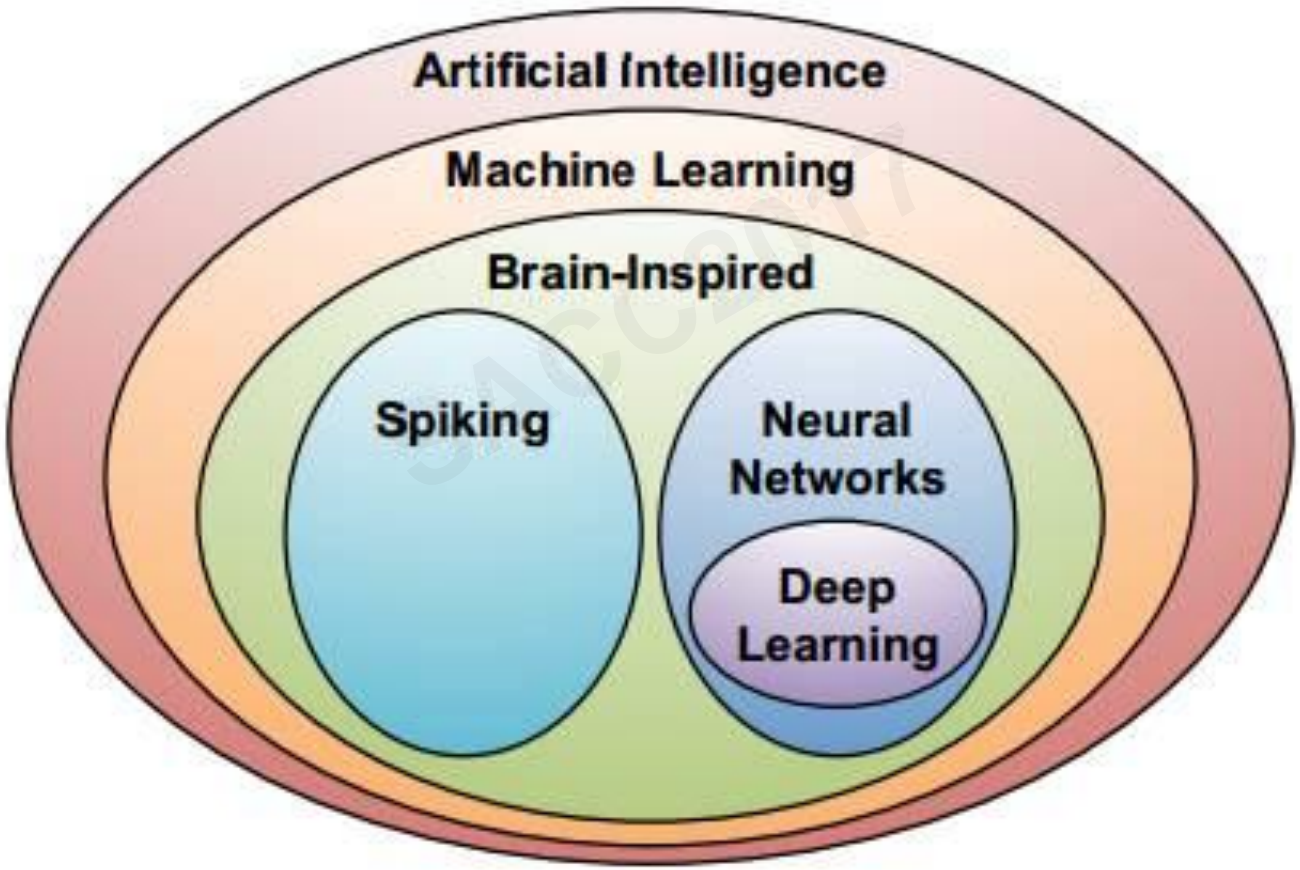


机器学习能做什么

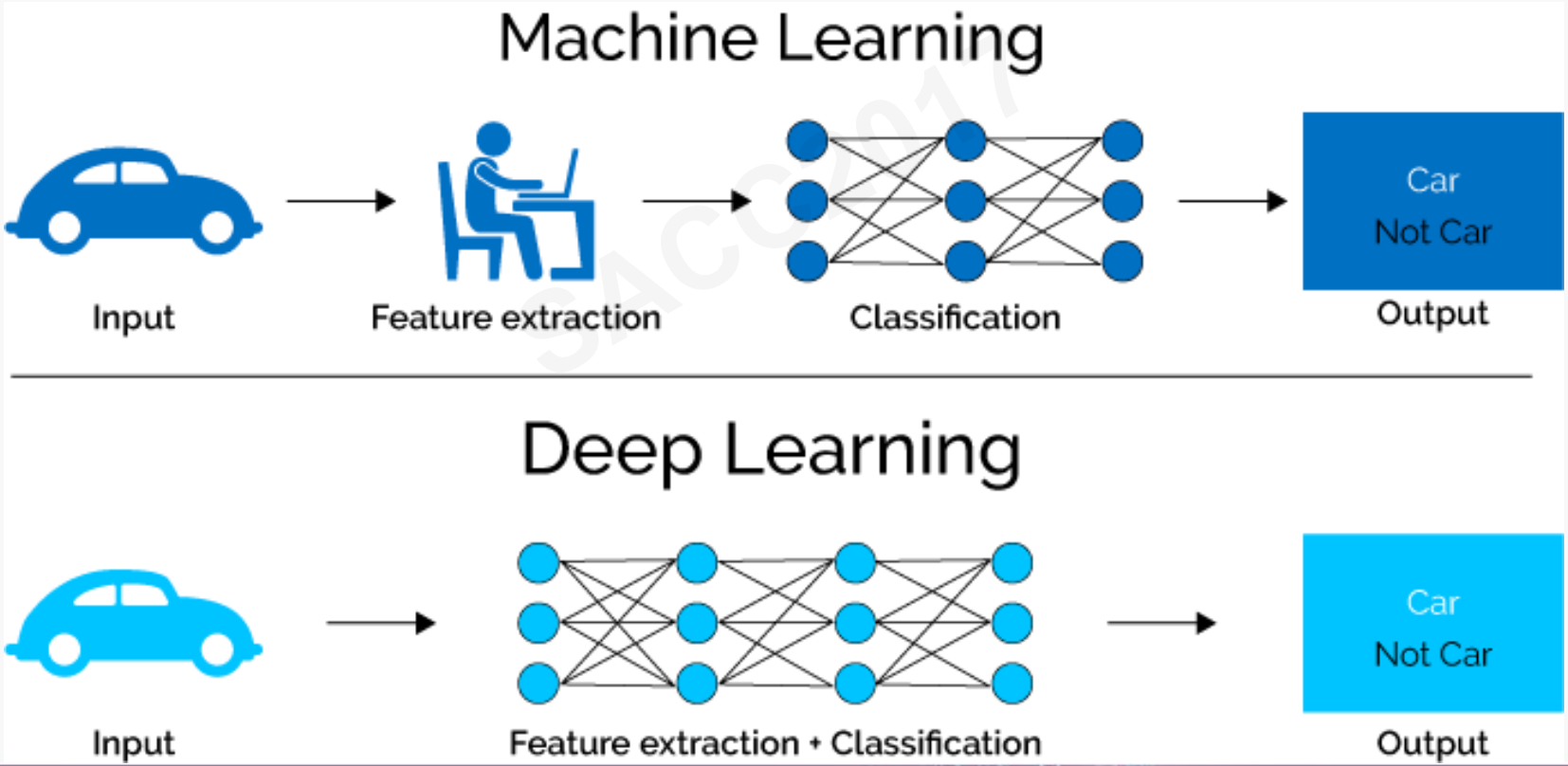
- 图像识别
- 语音识别
- 机械控制
- 安全分析



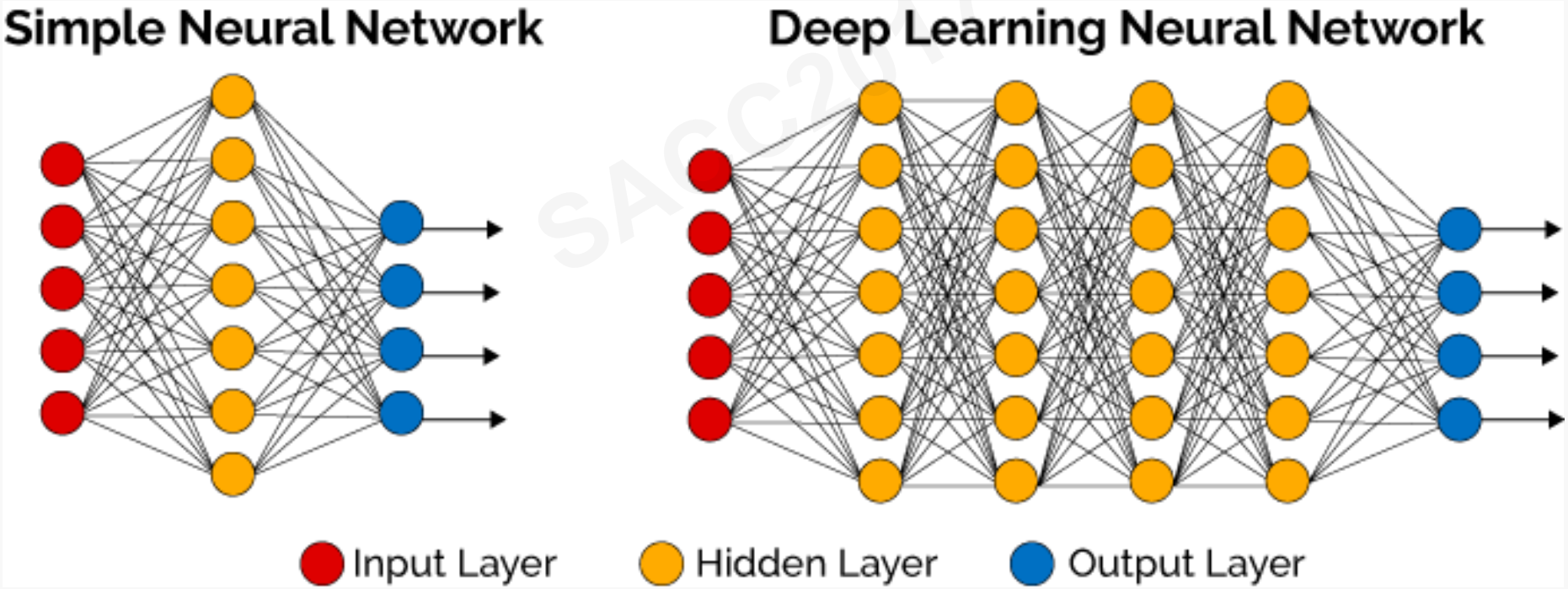

人工智能领域


Artificial Intelligence
Machine Learning
Brain-Inspired
Spiking
Neural Networks
Deep Learning

深度学习 VS 机器学习：


Machine Learning
Input
Feature extraction
Classification
Car
Not Car
Output
Deep Learning
Input
Feature extraction + Classification
Car
Not Car
Output

深度学习 VS 神经网络：


Simple Neural Network
Deep Learning Neural Network
Input Layer
Hidden Layer
Output Layer

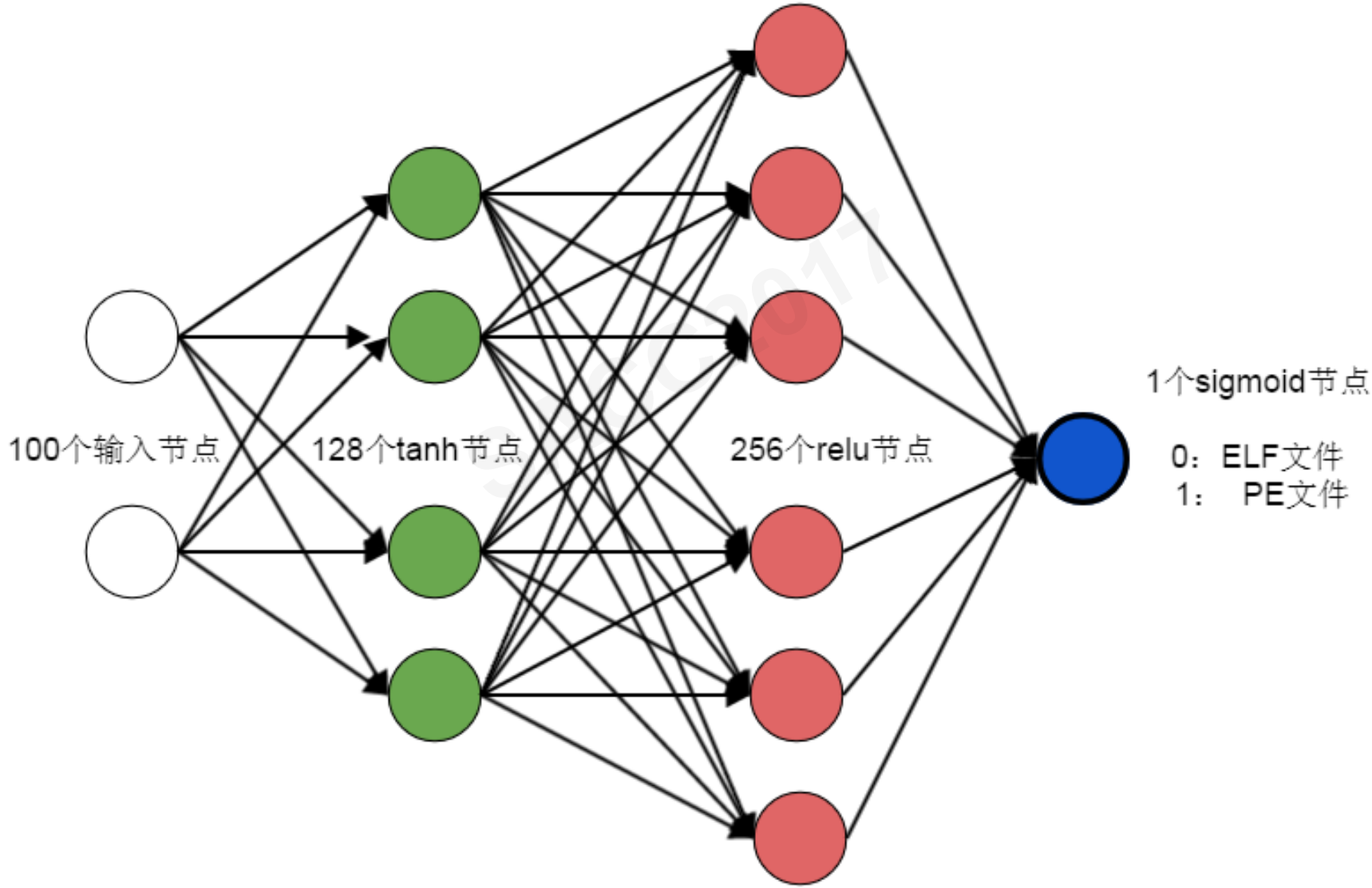
为什么深度学习这么火：

- 仿人的大脑神经感知外部世界的算法
- 解决了很多复杂问题
- 开源了很多能“上天”的框架
- 云计算普及化为深度学习提供了土壤。

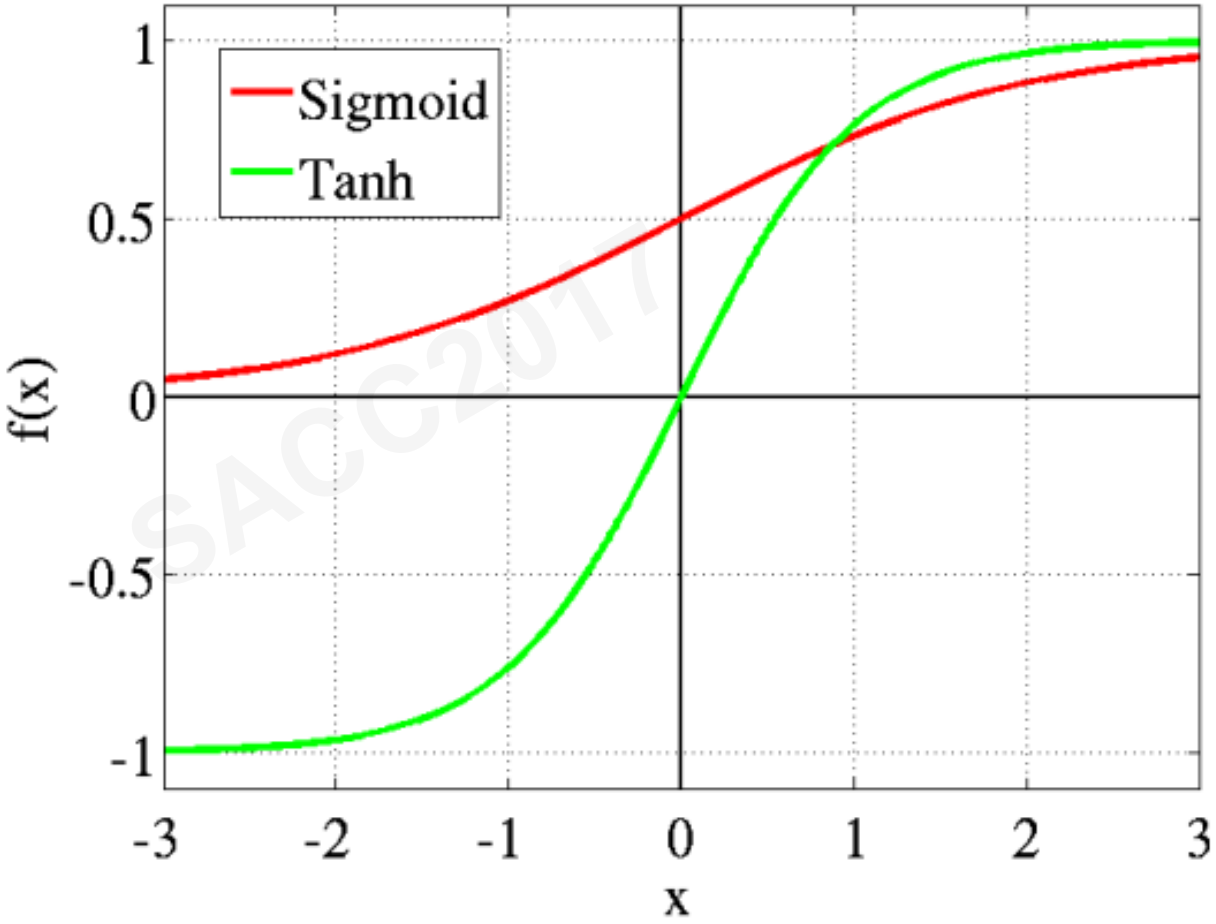
举个例子：

对文件是WINPE格式还是ELF格式进行分类。

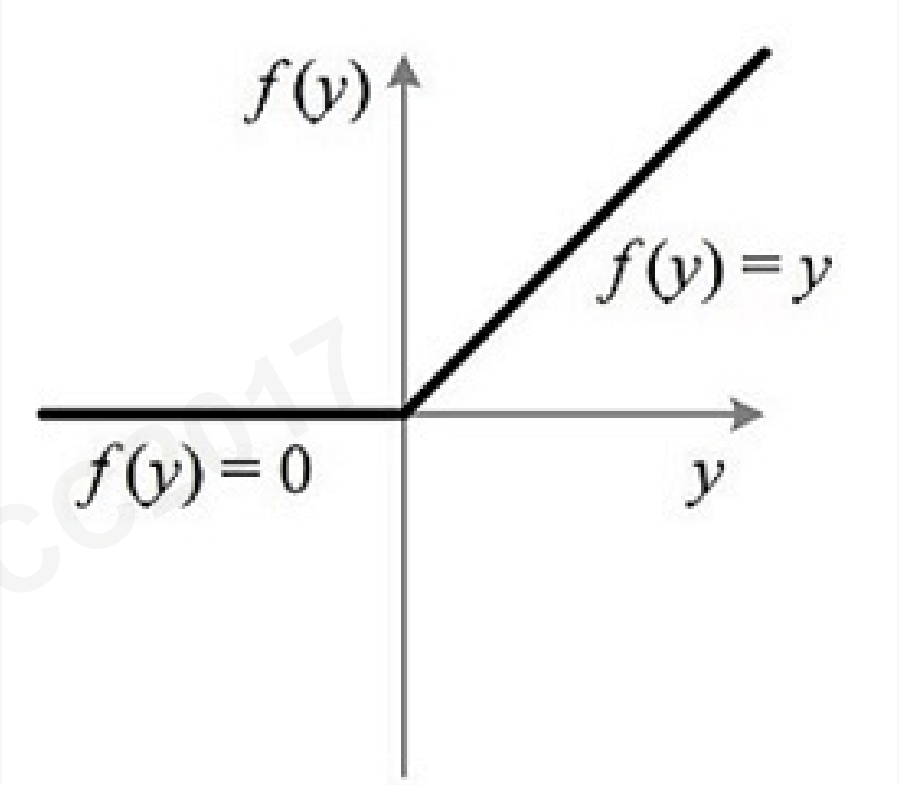
网络结构图：


100个输入节点
128个tanh节点
256个relu节点
1个sigmoid节点
0：ELF文件
1： PE文件

激活函数



ReLU
激活函数：



$$ReLU(x) = \begin{cases} x & if\, x > 0 \\ 0 & if\, x \le 0 \end{cases}$$

代码：

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import numpy as np
import cPickle as pickle
from keras.models import Sequential
from keras.layers import Dense, Dropout
#Load Data
f2 = file('data.pkl', 'rb')
x_train = np.array(pickle.load(f2))
y_train = np.array(pickle.load(f2))
f2.close()
#Create Model
model = Sequential()
model.add(Dense(128, input_dim=100, activation='tanh'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
#train
model.fit(x_train, y_train, epochs=20, batch_size=100)
#Load Data
f2 = file('data_test.pkl', 'rb')
x_test = np.array(pickle.load(f2))
y_test = np.array(pickle.load(f2))
f2.close()
score = model.predict(x_test, batch_size=30, verbose=0)
score = score.astype(int)
print score
score = model.evaluate(x_test, y_test, batch_size=30, verbose=1)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

```

运行：

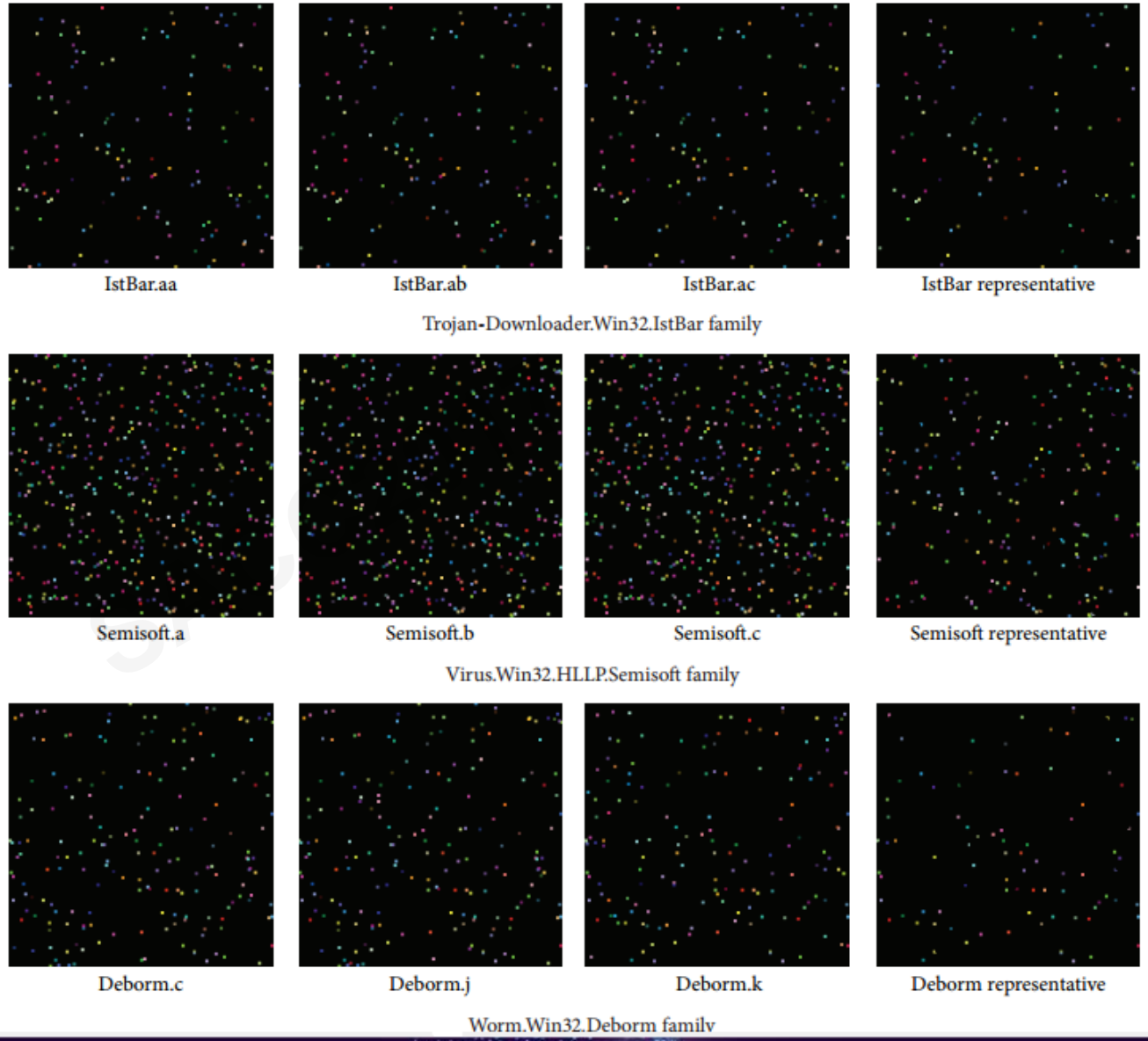
训练结果

```
Epoch 10/20
4000/4000 [==============================] - 0s - loss: 9.3012e-04 - acc: 0.9995
Epoch 11/20
4000/4000 [==============================] - 0s - loss: 0.0011 - acc: 0.9995
Epoch 12/20
4000/4000 [==============================] - 0s - loss: 0.0017 - acc: 0.9995
Epoch 13/20
4000/4000 [==============================] - 0s - loss: 8.8657e-04 - acc: 0.9998
Epoch 14/20
4000/4000 [==============================] - 0s - loss: 0.0010 - acc: 0.9995
Epoch 15/20
4000/4000 [==============================] - 0s - loss: 3.8162e-05 - acc: 1.0000
Epoch 16/20
4000/4000 [==============================] - 0s - loss: 1.7466e-04 - acc: 1.0000
Epoch 17/20
4000/4000 [==============================] - 0s - loss: 9.1167e-04 - acc: 0.9998
Epoch 18/20
4000/4000 [==============================] - 0s - loss: 9.5357e-06 - acc: 1.0000
Epoch 19/20
4000/4000 [==============================] - 0s - loss: 5.6347e-06 - acc: 1.0000
Epoch 20/20
4000/4000 [==============================] - 0s - loss: 5.0951e-05 - acc: 1.0000
[[0]
 [0]
 [0]
 [0]
 [1]
 [1]]
30/30 [==============================] - 0s
('Test loss:', 1.0640312098075810e-07)
('Test accuracy:', 1.0)
mao@KERAS:~/ml$
```

稍微复杂点儿的例子：

图像->决策->控制

文件可视化处理


IstBar.aa
IstBar.ab
IstBar.ac
IstBar representative
Trojan-Downloader.Win32.IstBar family
Semisoft.a
Semisoft.b
Semisoft.c
Semisoft representative
Virus.Win32.HLLP.Semisoft family
Deborm.c
Deborm.j
Deborm.k
Deborm representative
Worm.Win32.Deborm family

结构图：

网络结构：

将图像处理为4x80x80的图像矩阵输入。

第一层卷积层，有32卷积核，尺寸8x8，使用ReLU激活函数。

第二层卷积层，有64卷积核，尺寸4x4，使用ReLU激活函数。

第三层卷积层，有64卷积核，尺寸3x3，使用ReLU激活函数。

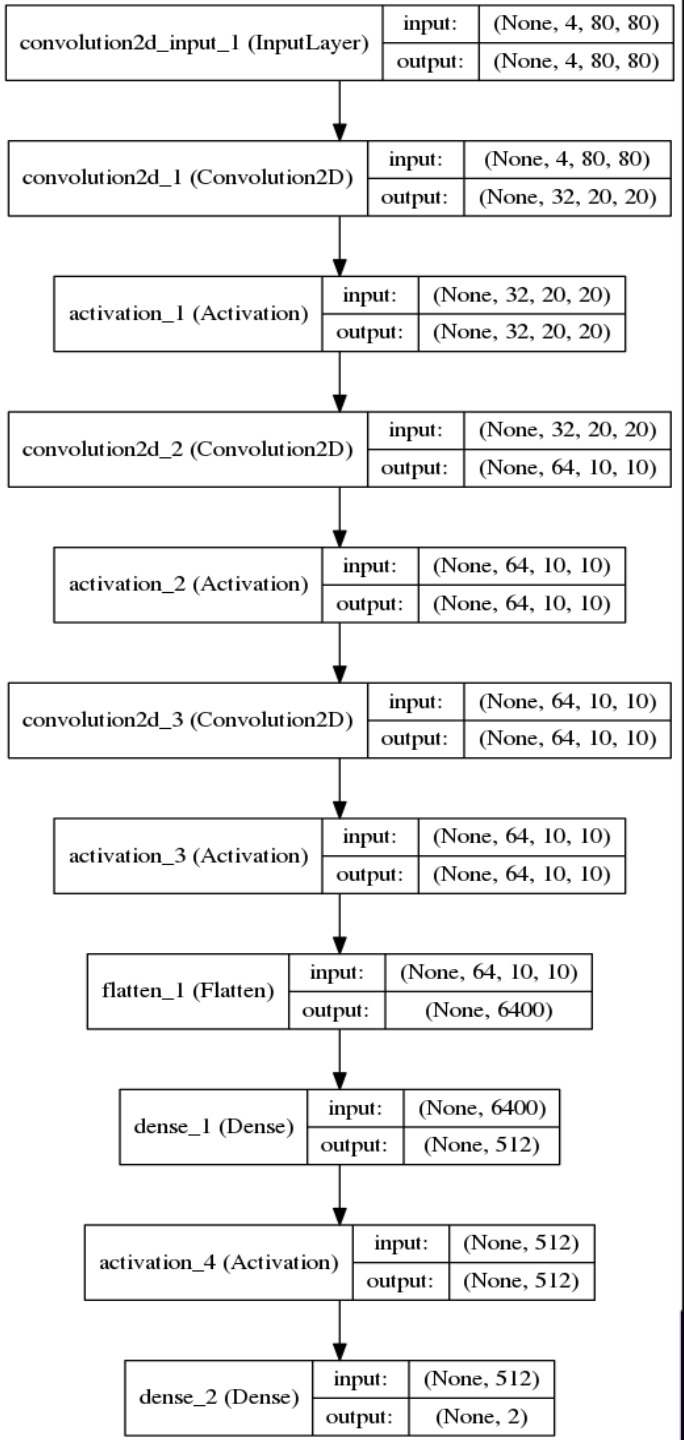
全连接为一维512个神经元的隐藏层，使用ReLU激活函数。

输出全连接线性层，输出对应动作：

0：什么也不做，1：跳一下。

决策结果：

+0.1表示存活，+1表示通过管道，-1表示死亡



心得：

1. PE ELF 都是可以处理的，做好训练集区分。
2. 两个学习侧重方向：
   - 识别程序的意图。（自然语言处理）
   - 二进制数据可视化。（图像处理）
3. 训练集临界点：恶意样本：10w，白样本50w。
   - 深度学习对样本数量要求还是蛮高的。
4. 恶意样本质量估算：正态分布。

目前成果


Samples
首页
资源监控
沙箱管理
人工分析
模型设置
系统设置
实时监控
最近14天
最近7个月
样本分析总数
5211246 个
2.4% 周期分析效率同比
样本分析进度
82%
样本增长率
10,000
8,000
6,000
4,000
2,000
0
8.16 8.17 8.18 8.19 8.20 8.21 8.22 8.23 8.24 8.25 8.26 8.27 8.28 8.29
沙箱集群
3.42 TB
CPU使用 85.4%
磁盘I/O 12.7 MB/s
存活节点 50个
MySQL集群
23 TB
CPU使用 13.9%
磁盘I/O 21.5 MB/s
存活节点 9个
MongoDB集群
295 TB
CPU使用 4%
磁盘I/O 13.1 MB/s
存活节点 24个
HBase集群
713 TB
CPU使用 2.1%
磁盘I/O 4.5 MB/s
存活节点 40个
人工分析
毛大鹏
领取任务 提交报告 报告审核 成果入库
累计任务 7
何斌斌
领取任务 提交报告 报告审核 成果入库
累计任务 68
应鑫磊
领取任务 提交报告 报告审核 成果入库
累计任务 45
事件监控
存储增速异常,超过阈值
2017-05-15 18:26:31
MongoDB 未能找到对象
2017-04-25
离线，受影响服务: HBase-DB3
2017-04-23 14:35:25
离线，受影响服务: HBase-DB2
2017-04-23 10:05:03
系统日志
save 2017-08-30 16:45:01 (24.6 MB/s) - '0b95a881a3a6bd2458495dcb5353fac1972015ed'
save 2017-08-30 16:36:25 (1.45 MB/s) - '0b95a9502f5e8945b553718a23de037ec4cb4cb8'
drop 2017-08-30 16:23:27 (1.81 MB/s) - '001b618d60495ad983bade61ced33ef17d508c0a'
save 2017-08-30 16:15:32 (1.05 MB/s) - '001aa79578fc941de8e4a104556fc8e103884226'
drop 2017-08-30 15:43:48 (363 MB/s) - '001b6195ee37f368488249ea6026b02072c024ca'
drop 2017-08-30 15:28:05 (17.0 MB/s) - '0b95a3f640b65b4399755635c3a7db489234e730'

第九届中国系统架构师大会

SYSTEM ARCHITECT CONFERENCE CHINA 2017

谢谢大家！