MALWARE ANALYSIS USING MACHINE LEARNING

Anshuman Singh

Associate Professor

Department of Information Systems and Technology

University of Missouri St Louis

CAE TechTalk, October 17, 2019

Outline

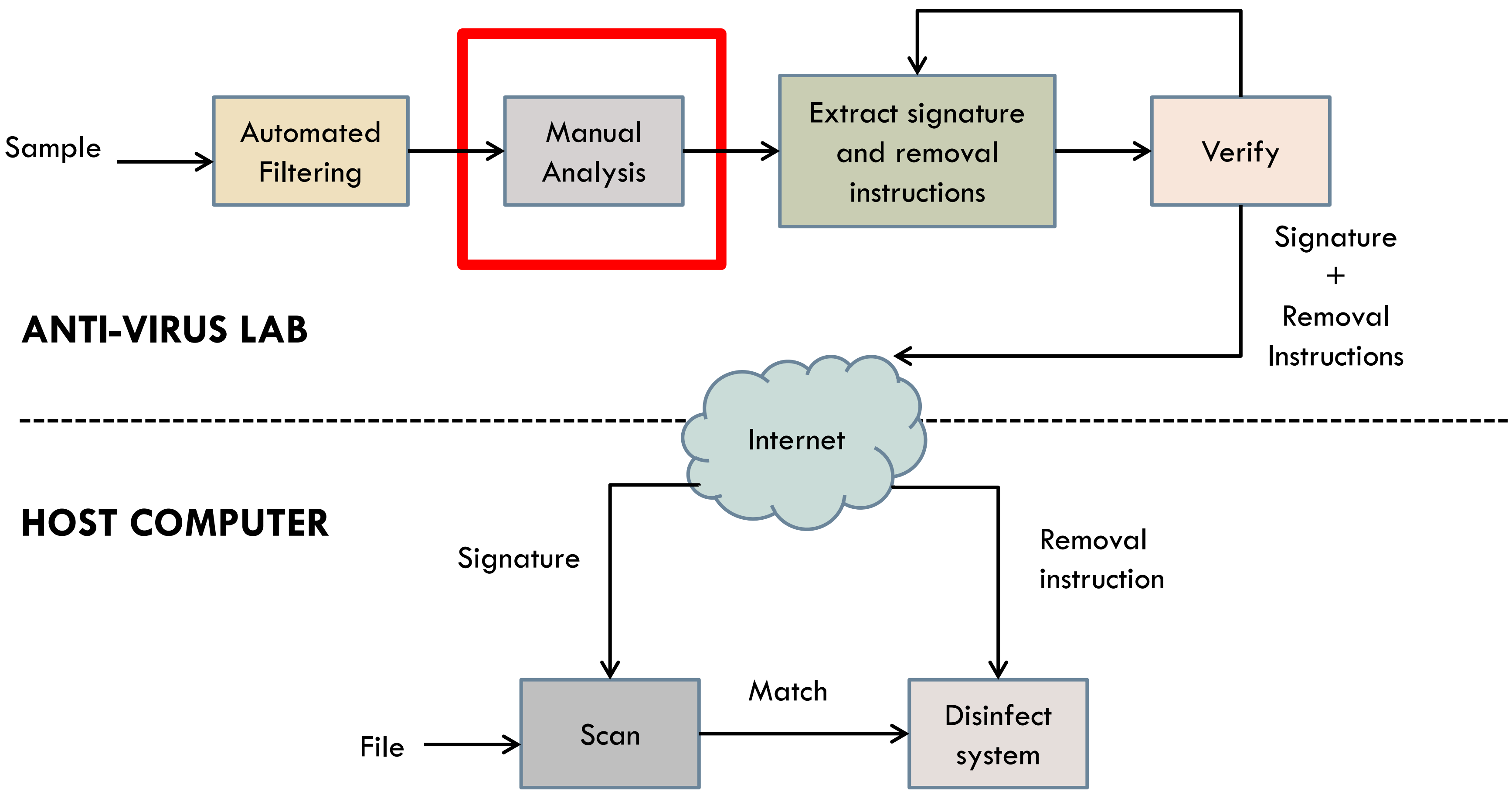
- Manual malware analysis
- Traditional machine learning in malware analysis
  - Applications
  - Features
  - Models
  - Challenges
- Deep learning in malware analysis
  - CNN, RNN, Stacked autoencoders
  - Familial classification
  - Signature generation

Malware analysis

- Definition:

  Examining an executable program (binary) to determine if it is malicious and identifying unique attributes of its malicious behavior

Signature-based detection



Manual malware analysis

Preliminary analysis

- Header analysis
- Section analysis
- Packed/Unpacked

Static analysis

- Disassembly
- Look for strings, API calls, encryption, etc.

Dynamic analysis

- Binary debugging
- Examine stack and registers at breakpoints

Manual malware analysis: Tools

Preliminary analysis

- PE explorer
- PE view
- Objdump

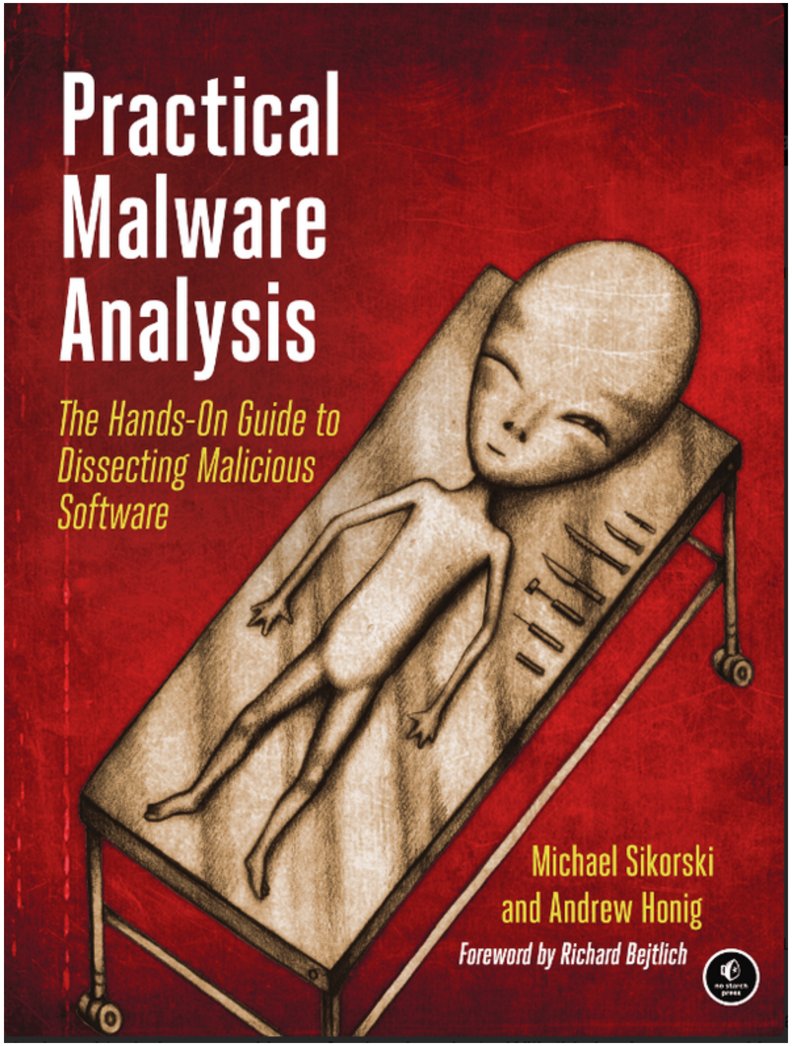
Static analysis

- IDA Pro
- Ghidra
- Radare

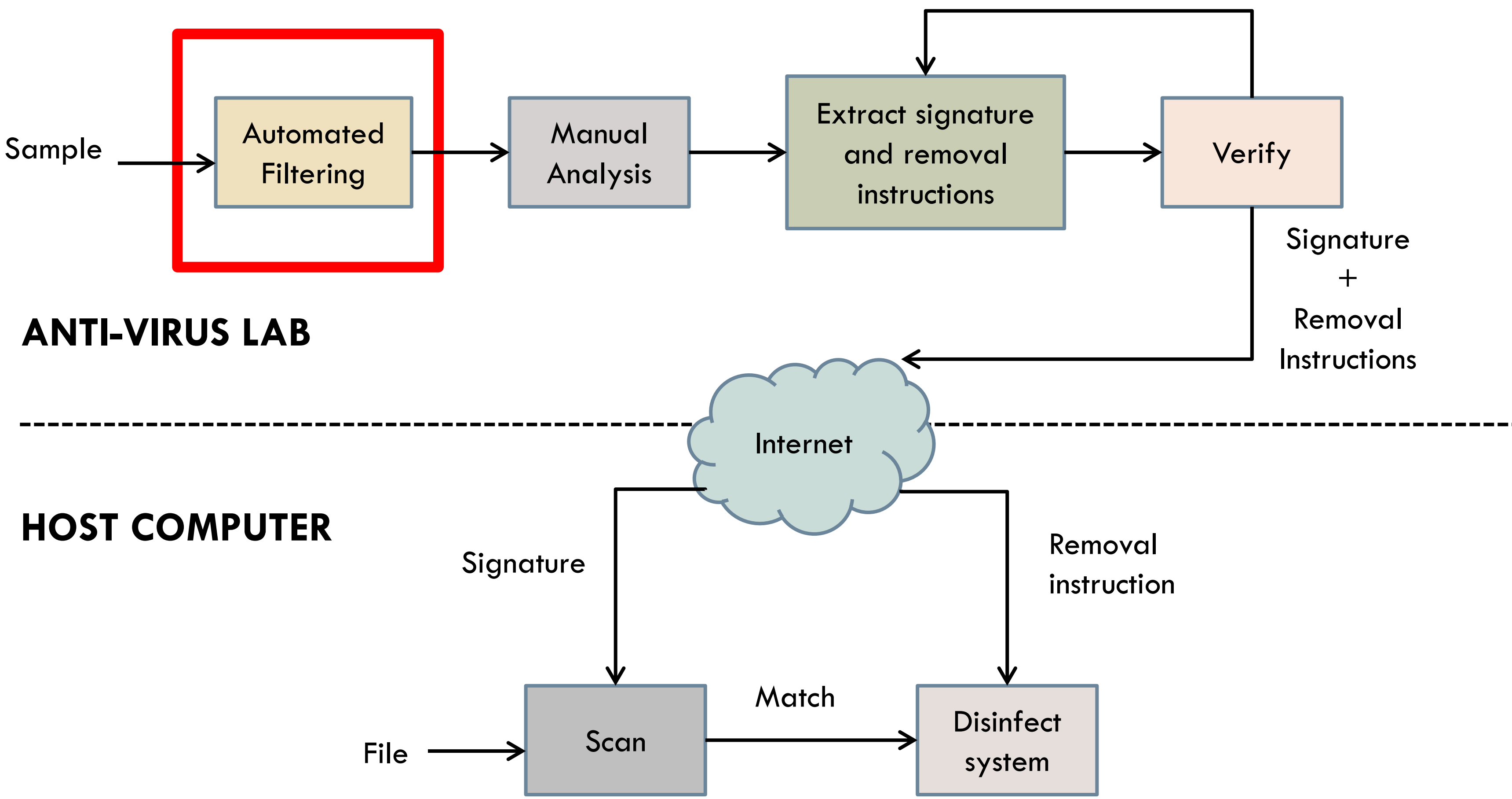
Dynamic analysis

- Ollydbg
- WinDbg
- Immunity

Manual malware analysis



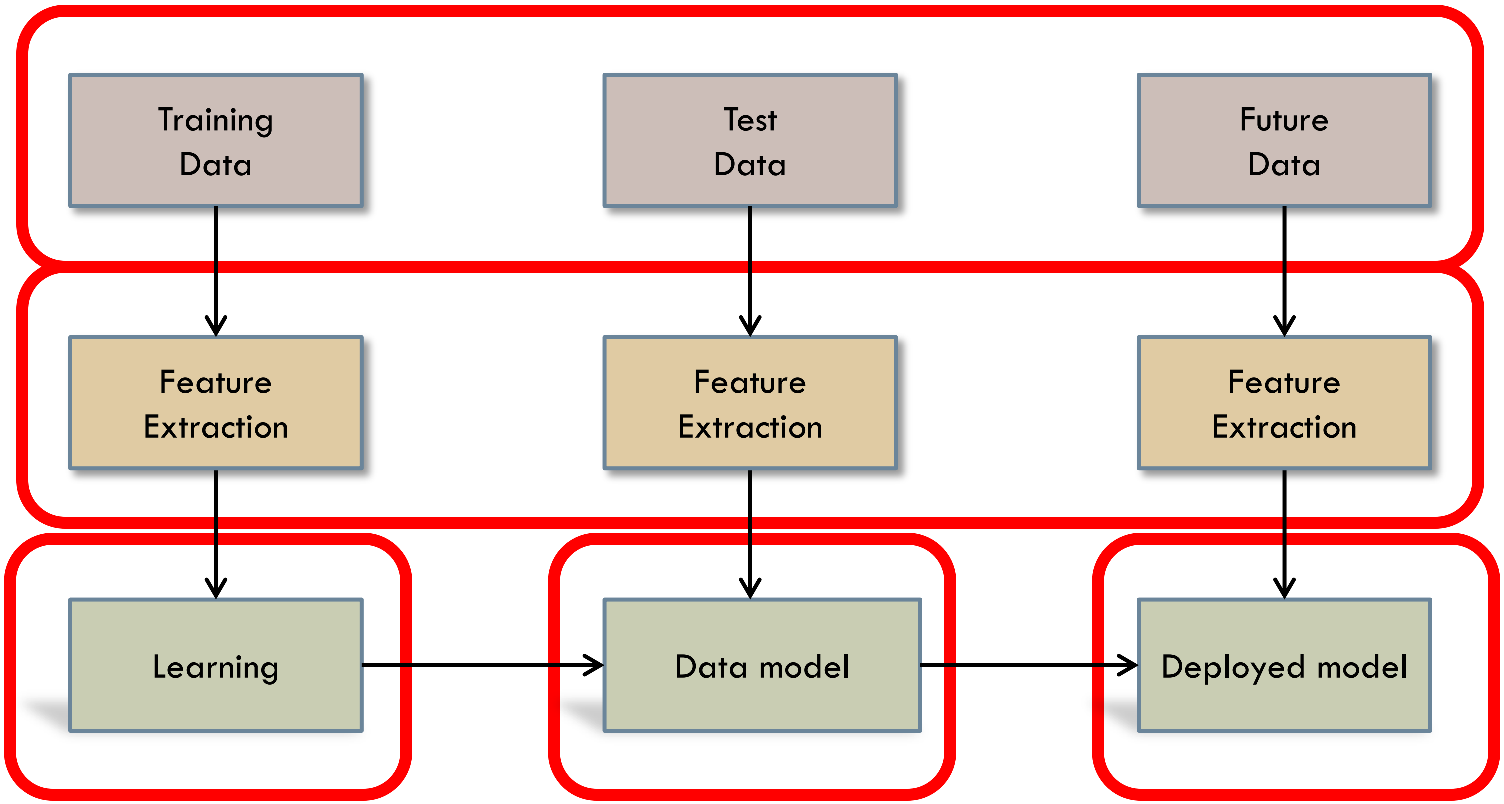
Signature-based detection



Machine learning applications

- **Malware Triage**: Prioritize incoming samples for manual analysis
- **Familial classification**: Classify samples into known malware families
- **Functional classification**: Classify samples based on their primary function (e.g., ransomware, bot, trojan, rootkit, etc.)
- **Packed/Unpacked**: Classify samples as packed or unpacked

Machine learning process



Malware features

- Static: Features obtained from the raw binary file, disassembly, or decompiled source code
  - Byte n-grams
  - Opcode n-grams
  - PE header data
- Behavioral: features obtained by running the sample
  - API call sequence
  - File activity
  - Network activity

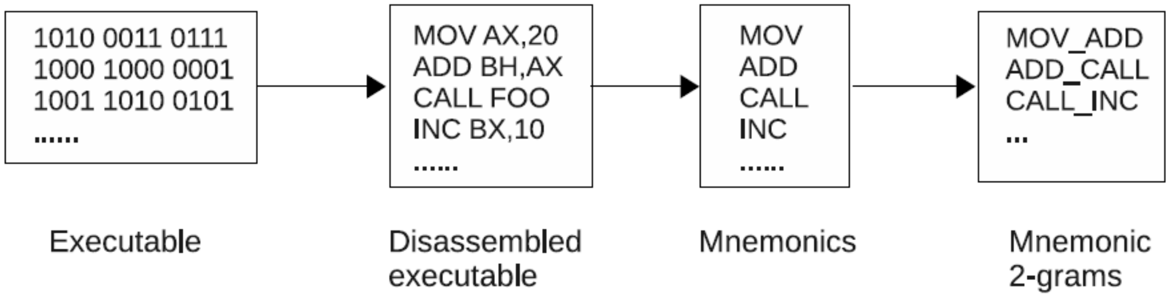
Static features

Byte 2-grams

- Extracted by sliding a 2-byte window along the executable
- Feature vector of each sample: Count of each byte 2-gram
- Total 65,536 byte 2-grams

Mnemonic 2-grams

- Feature vector of each sample: TFIDF of each mnemonic 2-gram
- TFIDF = TF*IDF
- IDF weighs down more commonly occurring features



Extracting malware features

- Static features
  - Objdump
  - Sliding window over binary, opcodes

```
[root@linux-server etc]# objdump -S userid | more

userid:     file format elf32-i386

Disassembly of section .init:

08048330 <_init>:
 8048330:       55                      push   %ebp
 8048331:       89 e5                   mov    %esp,%ebp
 8048333:       83 ec 08                sub    $0x8,%esp
 8048336:       e8 b9 00 00 00          call   80483f4 <call_gmon_start>
```

- Dynamic
  - Run sample in Cuckoo sandbox
  - Process JSON logs from the sandbox

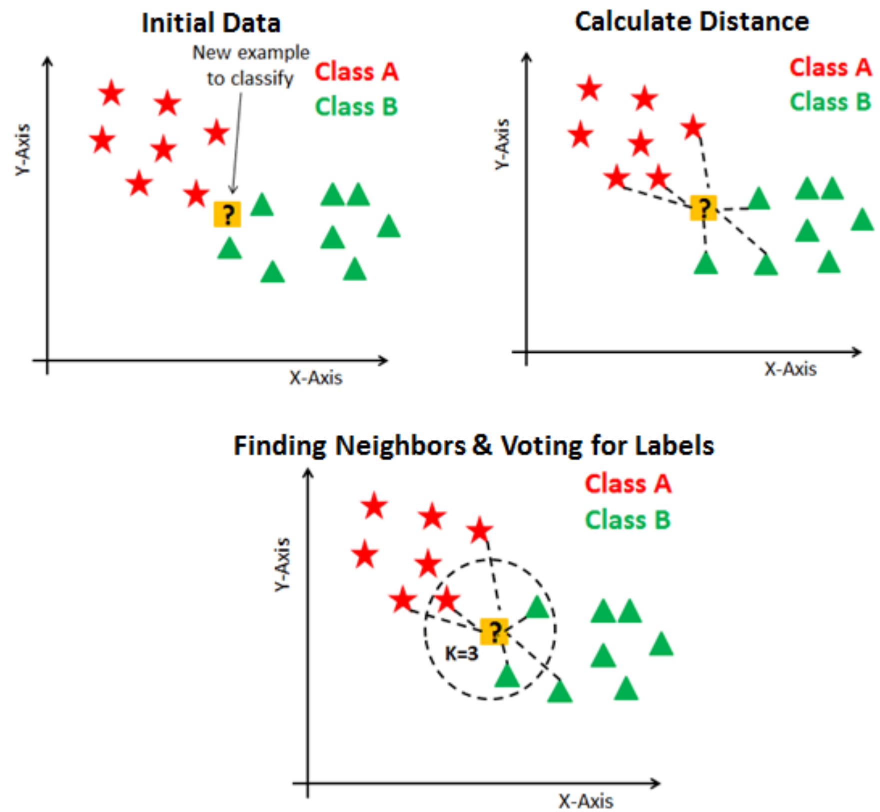
cuckoo Automated Malware Analysis

JSON log from Cuckoo sandbox


```
...
"hosts": ["0.0.0.0", "255.255.255.255",
"10.0.2.2", "10.0.2.15", "239.255.255.250",
"224.0.0.22", "10.0.2.255"], "dns": [],
"tcp": []}, "behavior": {"processes":
[{"parent_id": "428", "process_name":
"0a1cc307ed378bc79bc524497282c4d9c535cc3014d
8e2a9e72c0baad681b3e9", "process_id": "700",
"first_seen": "20140831184558.308", "calls":
[{"category": "filesystem", "status":
"SUCCESS", "return": "0x00000024",
"timestamp": "20140831184558.308",
"repeated": 0, "api": "CreateFileW",
"arguments": [{"name": "lpFileName",
"value": "C:\\WINDOWS\\system32
\\duser.dll"}, {"name": "dwDesiredAccess",
"value": "GENERIC_READ"}]}, {"category":
"filesystem", "status": "SUCCESS", "return":
"", "timestamp": "20140831184558.308",
...
```

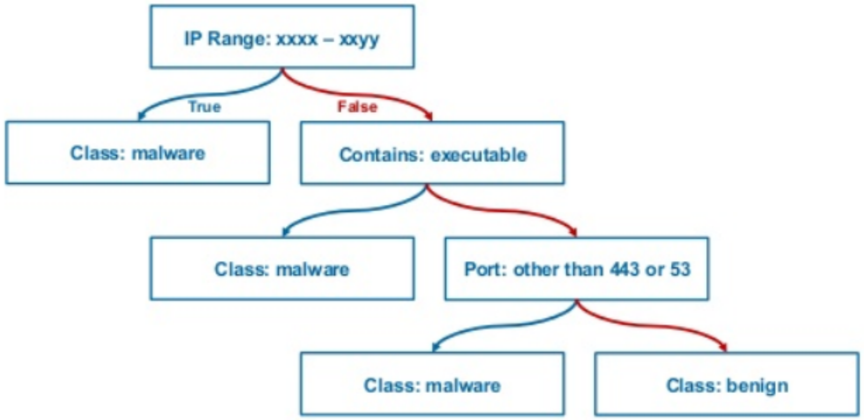
Learning models for malware

- **K-Nearest Neighbor:** Samples in the training dataset are mapped to an n-dimensional space. If the majority of the K nearest neighbors of an incoming samples are malicious, the incoming sample is labeled malicious.
  - No model construction needed
  - Minimal structural assumptions about the dataset
  - Best suited for malware triage



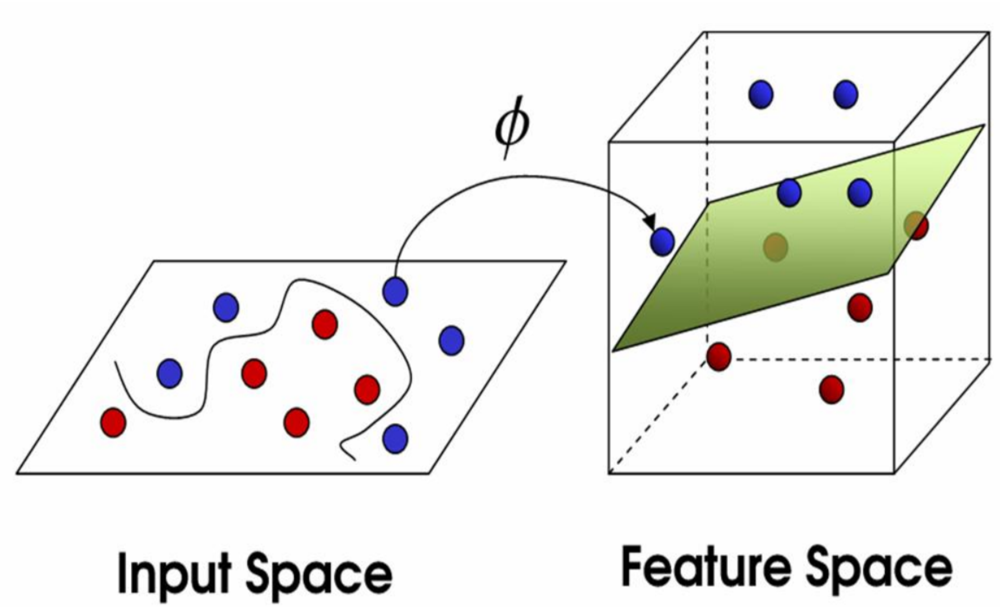
Learning models for malware

- **Decision trees:** Tree nodes consists of features that split the samples based on values that give most homogeneous samples in each subtree. Only most discriminatory features used for generating the tree.
  - Works better with behavioral features
  - Models can be easily explained
  - No need to keep all samples after tree is constructed



Learning models for malware

- **Support vector machines:** produce nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space
  - Used when classes are not linearly separable
  - High accuracy with behavioral features
  - Most successful before deep learning



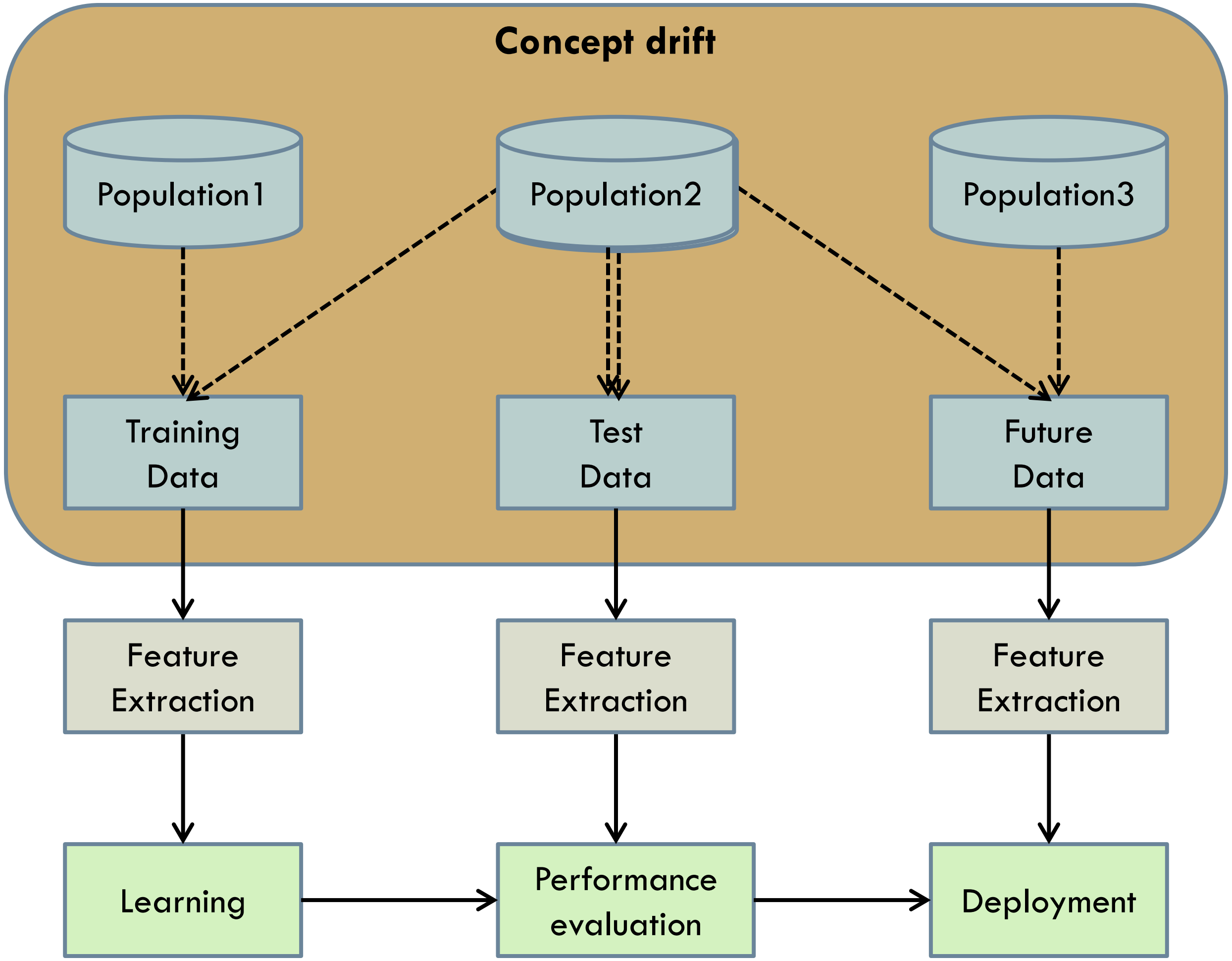
Challenges

- **Concept drift:** i.i.d (independent and identically distributed) assumption of traditional machine learning may not hold for malware
- **High FP:** Difficult to keep false positives under a threshold. Nobody will use an anti virus if it starts flagging non-malicious files as malicious
- **Feature engineering:** Feature construction still requires human expertise and is error prone
- **Poisoning attacks:** machine learning techniques are prone to training data poisoning leading to incorrect model construction

Causes of change in malware

Natural evolution	Environmental evolution	Polymorphic evolution
• Adding functionalities • Making bug fixes • Porting to a new environment	• Evolution in the compiler • Using different compiler switches • Using a different compiler itself • Changes in the libraries linked to the malware	• Encrypted code • Obfuscated code

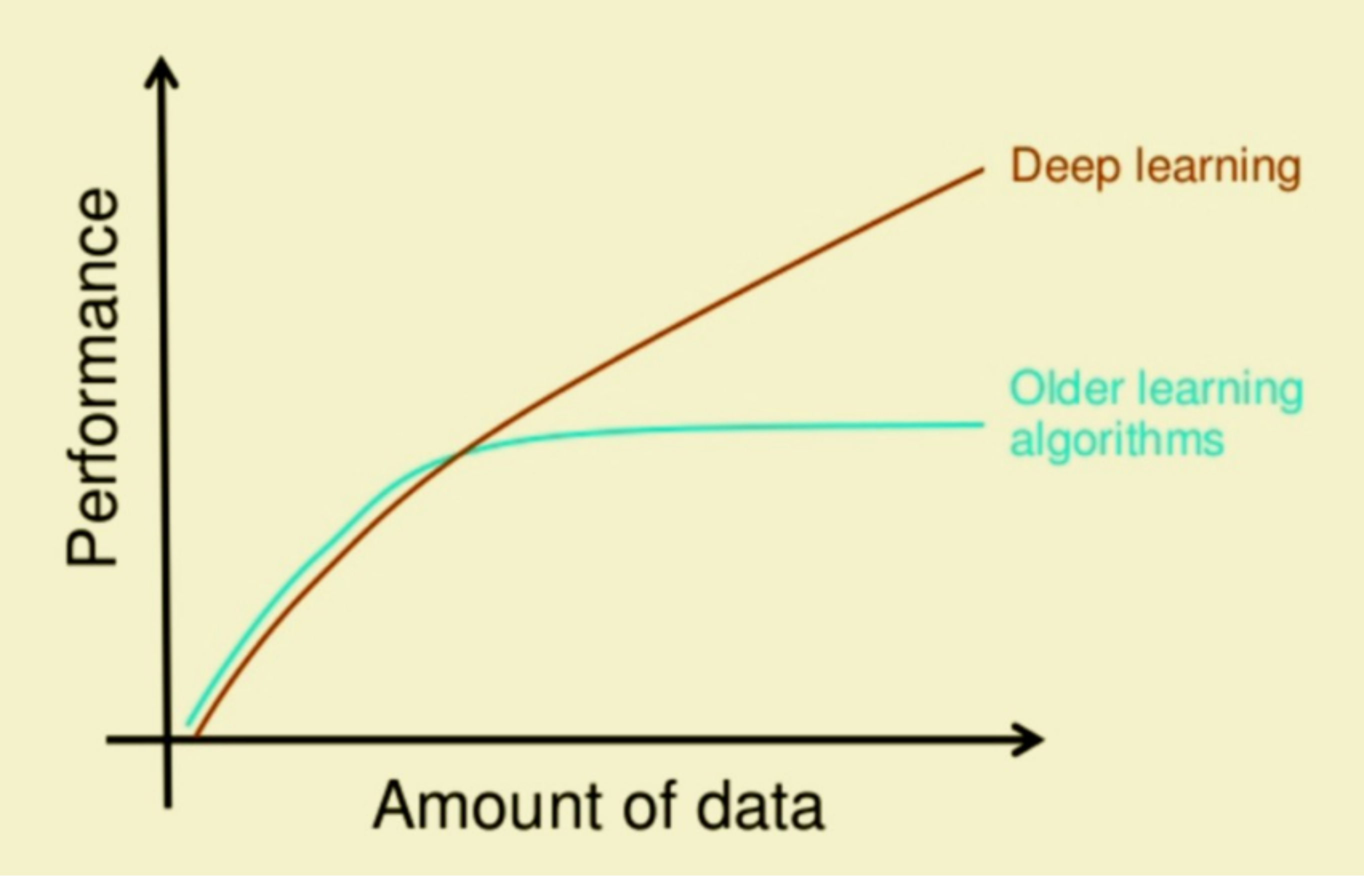
Concept drift



Deep Learning Approach

- **No feature engineering:** No need to determine the right features. Deep neural networks discover interesting features.
- **No concept drift:** Deep neural networks continue to learn and adapt with new data
- **Very high accuracy:** Usually greater than 99%
- **Low False Positives:** No more regular files getting labeled as malicious

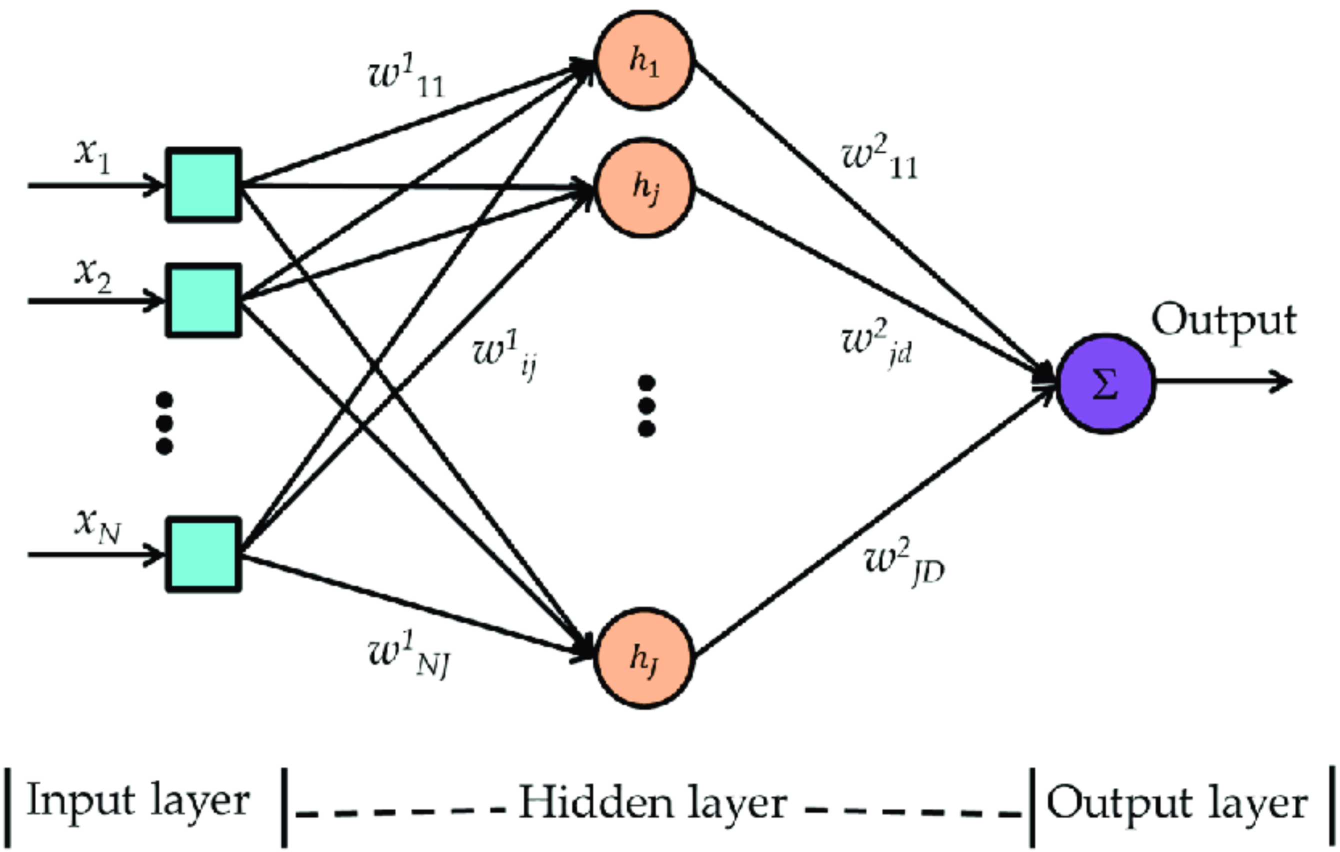
Why deep learning?



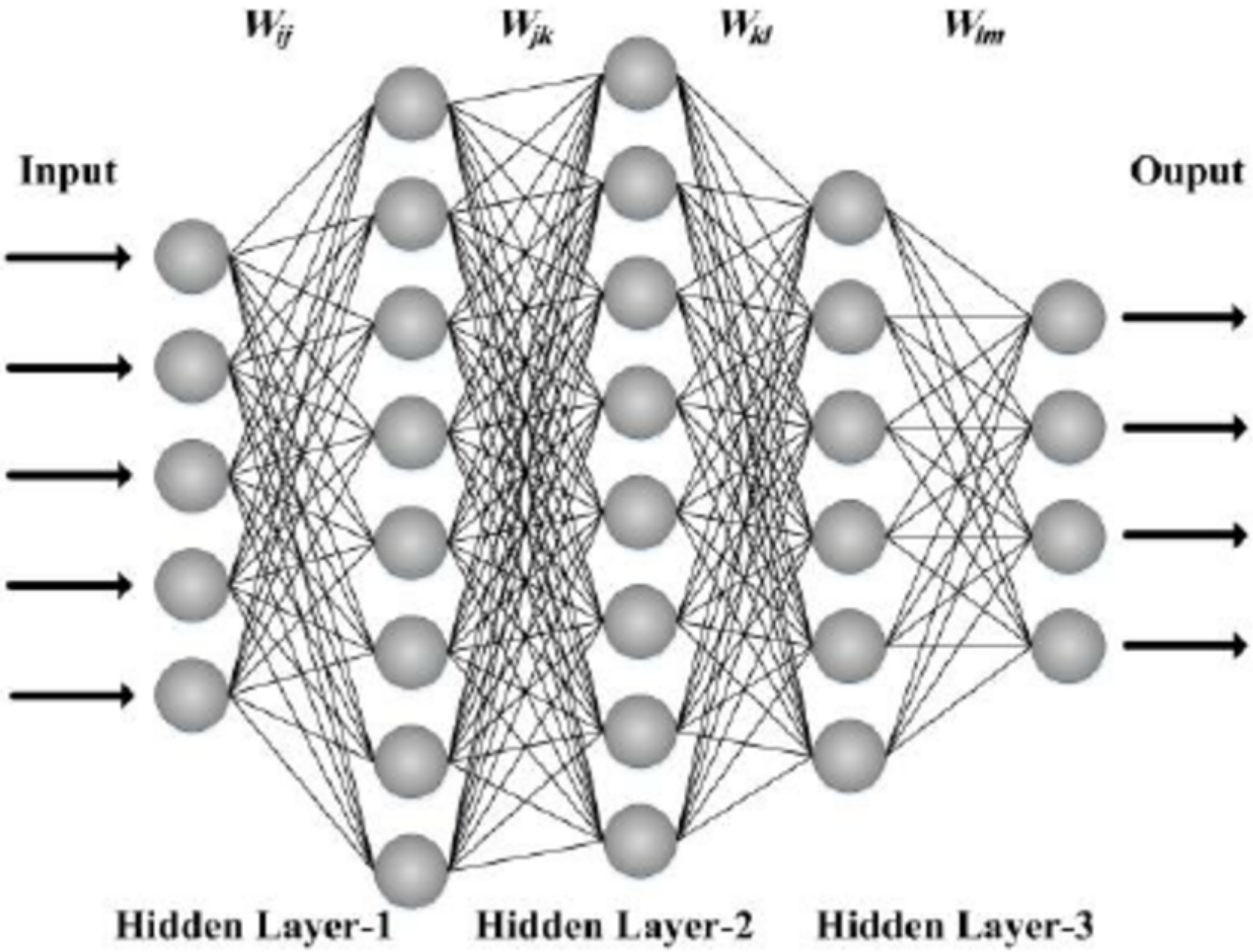
DL approach

- CNN: Convolutional neural networks
  - Convolve inputs (weighted map) to a lower dimensional feature space to extract more prominent features
- RNN: Recurrent neural networks
  - LSTM: Long short-term memory, a type of RNN
  - Suitable for sequential inputs
- Stacked Autoencoders
  - Suitable for unsupervised deep learning

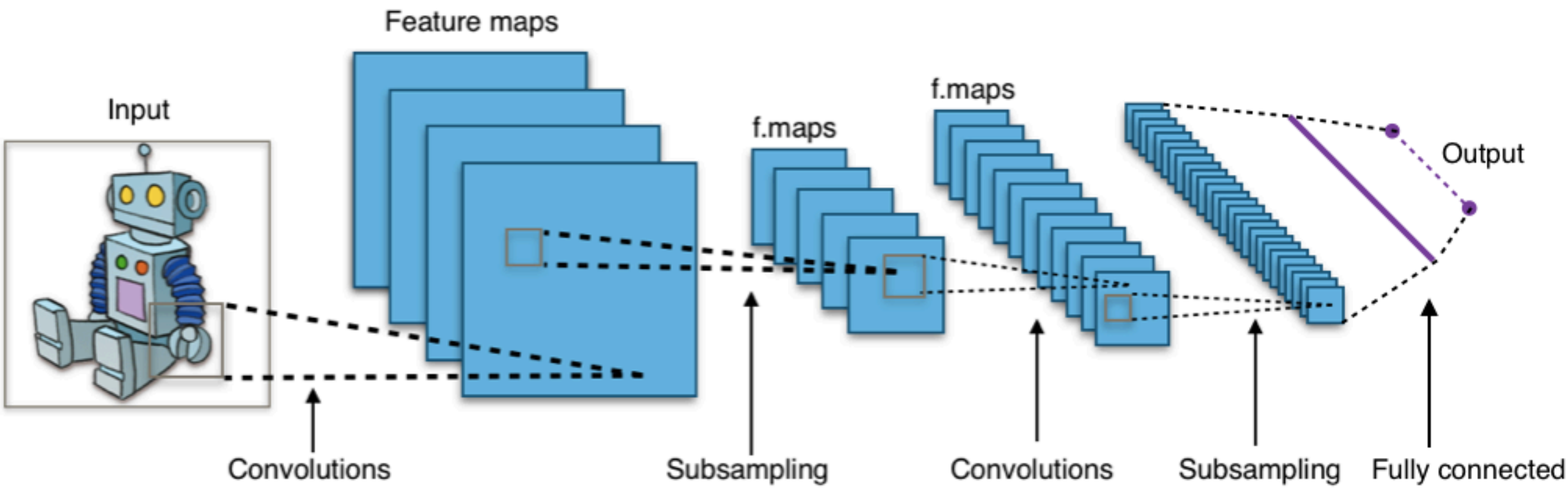
Multilayer perceptron



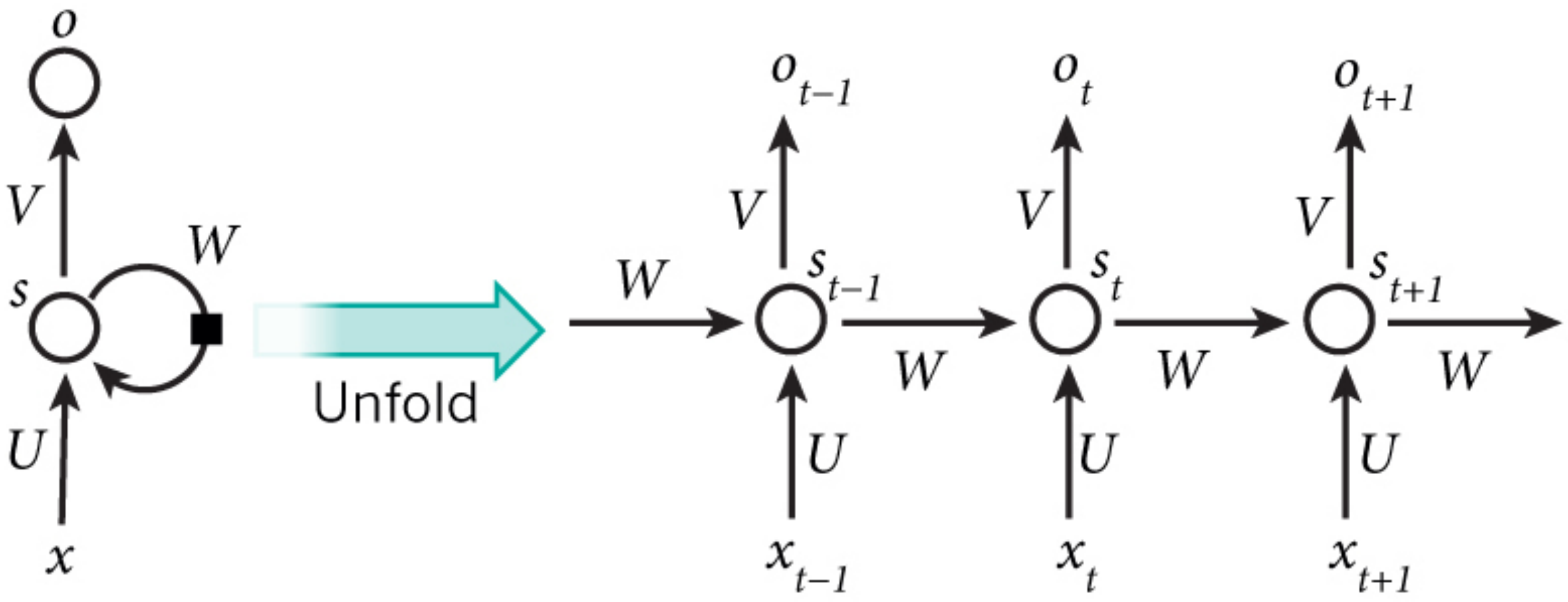
Deep neural network



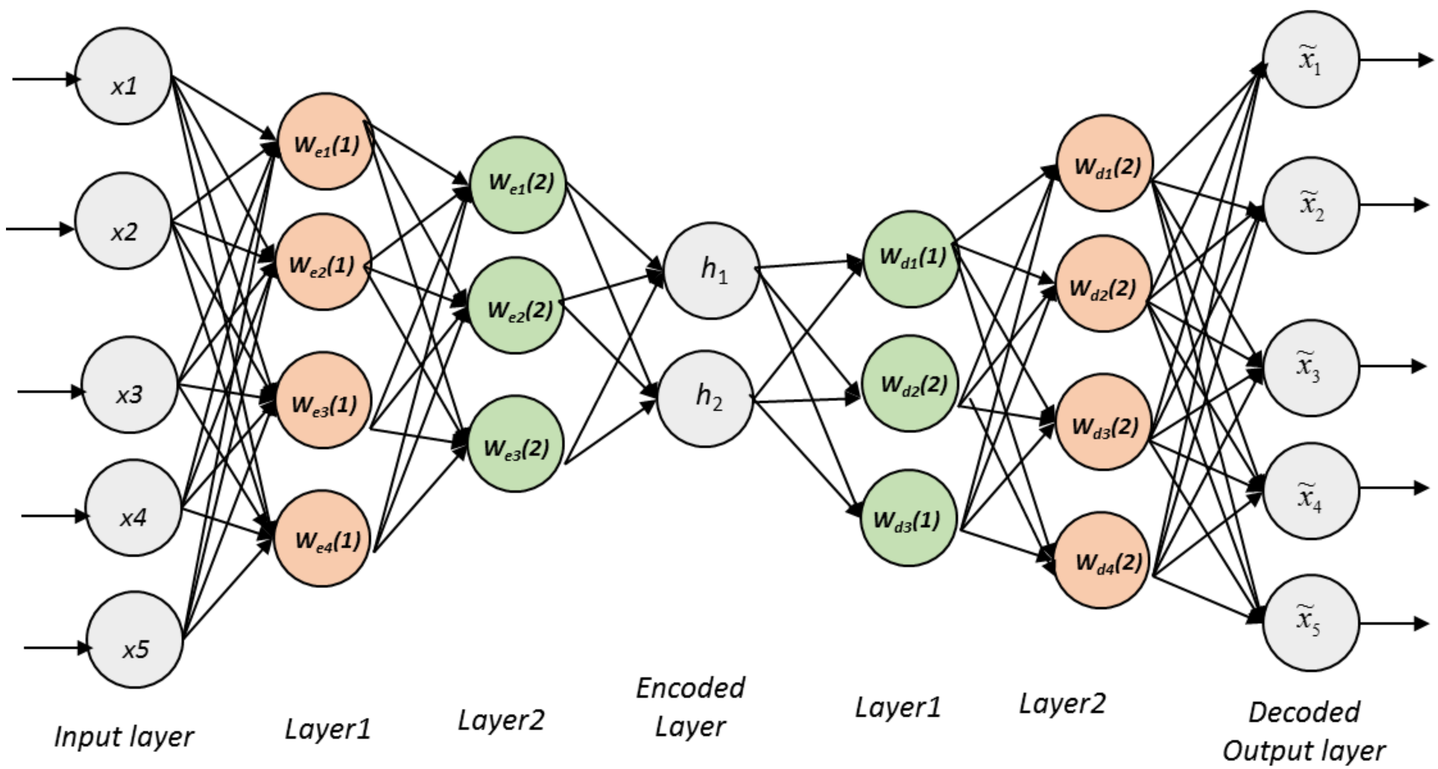
Convolutional neural network



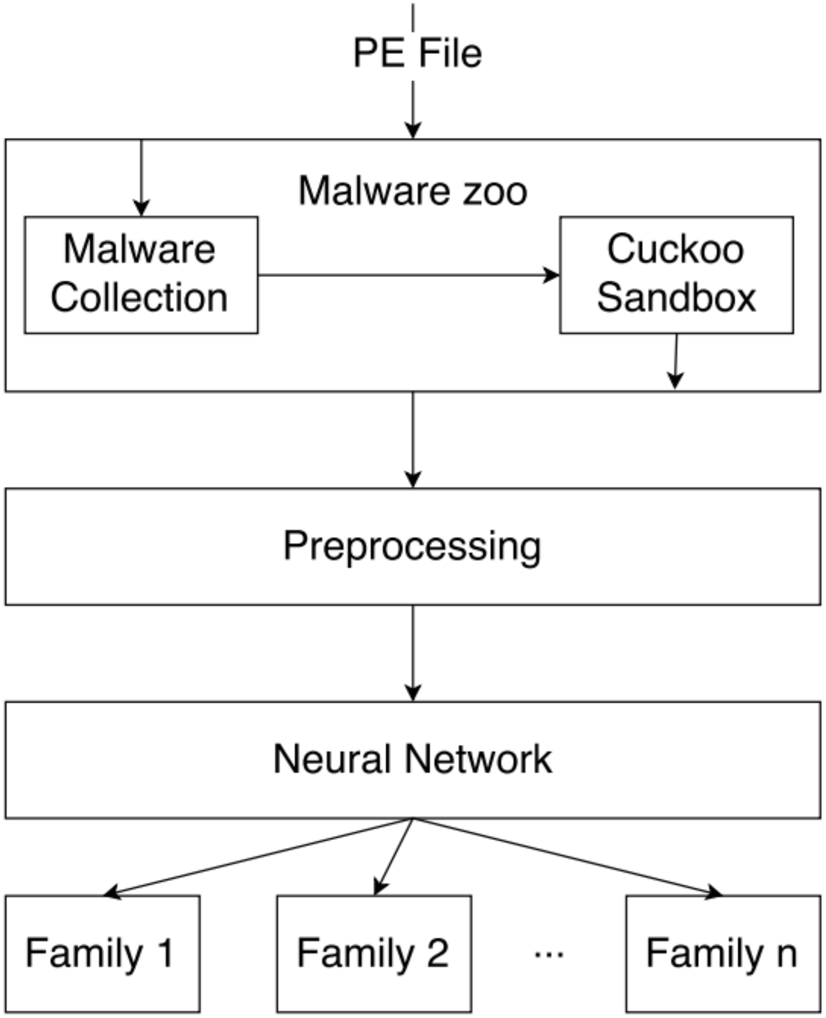
Recurrent neural networks



Stacked autoencoders

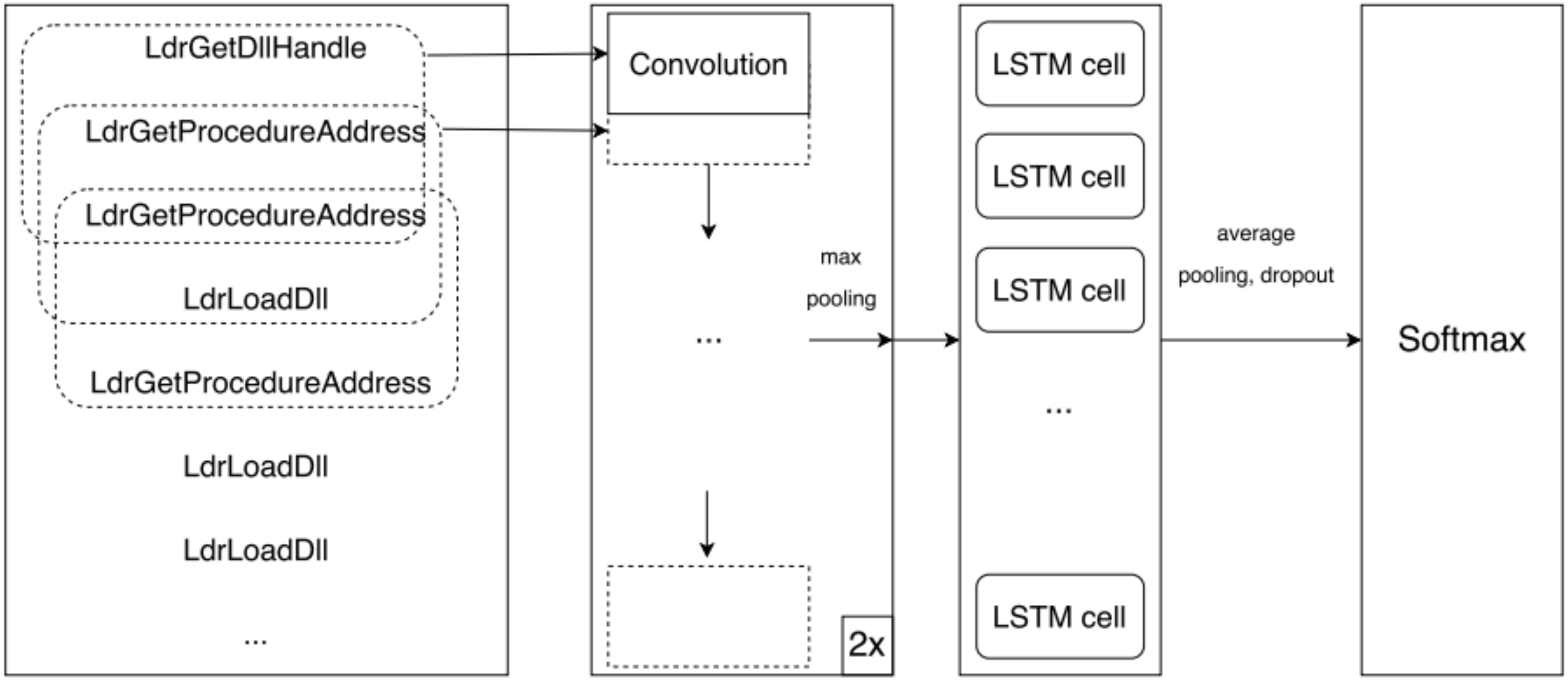


Familial classification using DL



TensorFlow

Deep learning architecture

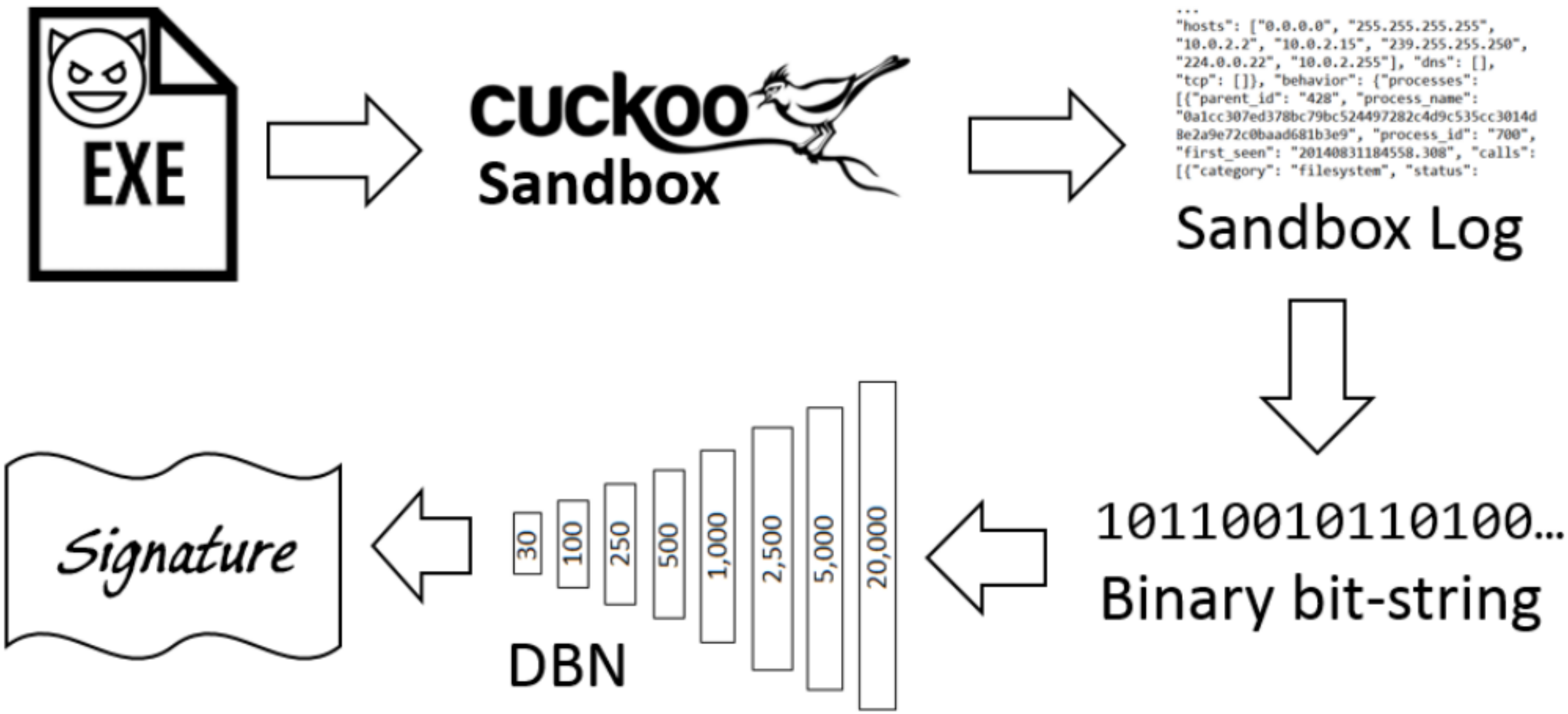


DL vs Traditional ML: Results

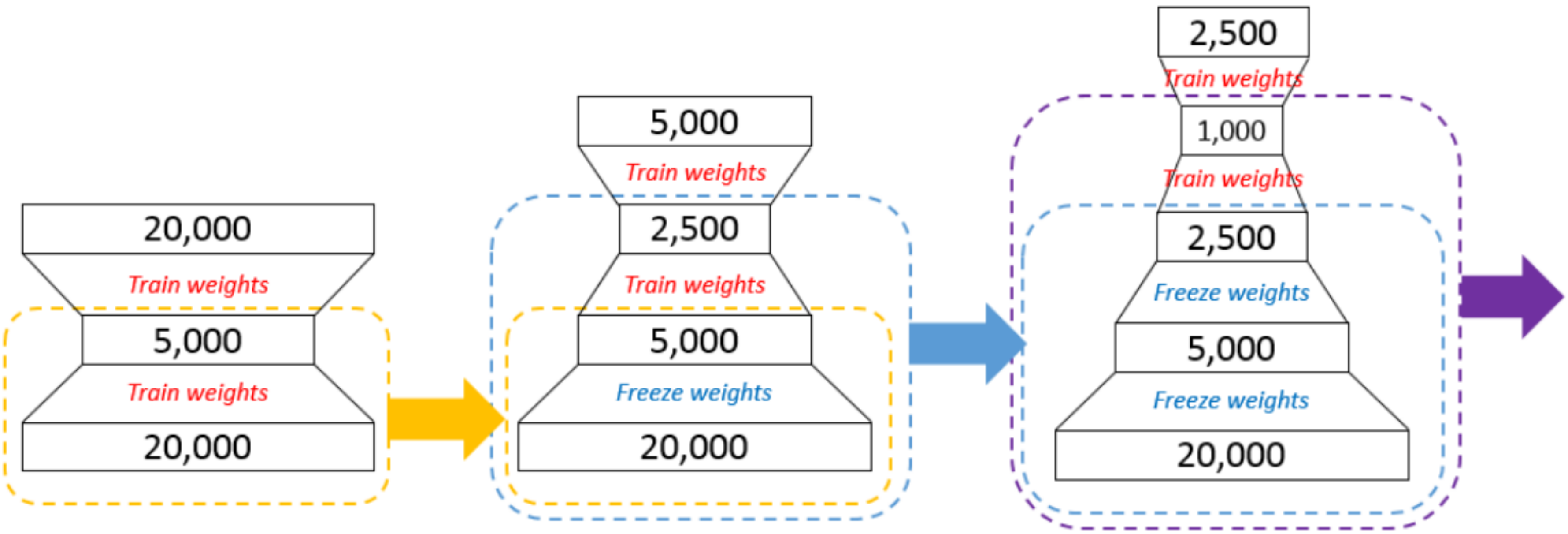
Family	Deep Neural Network			Hidden Markov Model			Support Vector Machine		
	ACC	PR	RC	ACC	PR	RC	ACC	PR	RC
Multiplug	98.9	99.8	99.0	91.5	74.5	91.5	99.3	99.9	99.3
Kazy	100.0	99.9	100.0	73.1	95.1	73.1	96.6	93.1	96.6
Morstar	100.0	99.9	100.0	80.0	63.7	80.0	82.3	91.0	82.3
Zusy	100.0	57.5	100.0	65.4	45.1	65.4	100.0	58.4	100.0
SoftPulse	100.0	99.1	100.0	51.1	100.0	51.1	99.9	99.6	99.9
Somoto	100.0	100.0	100.0	50.0	37.6	50.0	99.8	100.0	99.8
Mikey	0.0	0.0	0.0	5.7	20.0	5.7	0.0	0.0	0.0
Amonetize	99.1	100.0	99.6	29.4	100.0	29.4	99.3	100.0	99.3
Eldorado	99.4	100.0	99.5	20.0	80.4	20.0	100.0	100.0	100.0
Kryptik	96.6	100.0	96.2	10.0	100.0	10.0	97.1	100.0	97.1
Average	89.4	85.6	89.4	47.5	71.6	47.6	87.4	84.2	87.4

Source: Kolosnjaji, B., Zarras, A., Webster, G., & Eckert, C. (2016, December). Deep learning for classification of malware system call sequences. In *Australasian Joint Conference on Artificial Intelligence* (pp. 137-149). Springer.

Signature generation using DL



Stacked autoencoders in DBN



TensorFlow

References

- Kolter, J. Z., & Maloof, M. A. (2006). Learning to detect and classify malicious executables in the wild. *Journal of Machine Learning Research*, *7*(Dec), 2721-2744.
- Lakhotia, A., Walenstein, A., Miles, C., & Singh, A. (2013). Vilo: a rapid learning nearest-neighbor classifier for malware triage. *Journal of Computer Virology and Hacking Techniques*, *9*(3), 109-123.
- Singh, A., Walenstein, A., & Lakhotia, A. (2012, October). Tracking concept drift in malware families. In *Proceedings of the 5th ACM workshop on Security and artificial intelligence* (pp. 81-92). ACM.
- Kolosnjaji, B., Zarras, A., Webster, G., & Eckert, C. (2016, December). Deep learning for classification of malware system call sequences. In *Australasian Joint Conference on Artificial Intelligence* (pp. 137-149). Springer, Cham.
- David, O. E., & Netanyahu, N. S. (2015, July). Deepsign: Deep learning for automatic malware signature generation and classification. In *2015 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). IEEE.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112, p. 18). New York: springer.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Questions

?