



Malware Characterization using Compiler-based Graphs



Tristan Vanderbruggen

Dept of Computer & Information Sciences

University of Delaware

CISC 850: Cyber Analytics



Malware Characterization

Malware can:

- Send spam
- Steal private information
- Ransom data
- Be used for warfare

Current technologies inadequate

- Automation => millions of variants
- Zero day exploits

Discovering new malware

- Manual analyses

```

[0x0004638] 3k 185 .\i32ae].pd $r @ main+2488 # 0x0004638
: JMP XREF from 0x000464b (unk)
0x0004638 885acc21fff mov eax, dword [ebp - 0x3d54]
0x0004639 8942408 mov dword [esp + 8], eax
0x0004642 8b45dc mov eax, dword [ebp - 0x24]
0x0004645 8942404 mov dword [esp + 4], eax
0x0004648 8485beeffff lea eax, [ebp - 0xc42]
0x000464f 890424 mov dword [esp], eax
0x0004652 e878af7fff call sym.writefile ;[1] :sym.writefile(x)
0x0004657 85c0 test eax, eax
0x0004659 7421 je 0x000467c ;[2]
0x000465b 8485beeffff lea eax, [ebp - 0xc42]
0x000465c 890424 mov dword [esp], eax
0x0004664 ea135a7fff call sym.randmd5 ;[3] :sym.randmd5()
0x0004669 8485beeffff lea eax, [ebp - 0xc42]
0x000466f 890424 mov dword [esp], eax
0x0004672 ea37887fff call sym.LinuxExec
0x0004677 e801000000 jmp 0x000467c
: JMP XREF from 0x0004679 (unk)
0x0004677 885acc21fff mov eax, dword [ebp - 0x3d54]
0x0004682 8942408 mov dword [esp + 8], eax
0x0004685 8b45dc mov eax, dword [ebp - 0x24]
0x0004688 8942404 mov dword [esp + 4], eax
0x000468d 8485beeffff lea eax, [ebp - 0x1042]
0x0004693 890424 mov dword [esp], eax
0x0004696 e878af7fff call sym.writefile ;[1] :sym.writefile(x)
0x000469b 85c0 test eax, eax
0x000469d 741e je 0x00046bd ;[6]
0x000469f 8485beeffff lea eax, [ebp - 0x1042]
0x00046a5 890424 mov dword [esp], eax
0x00046a8 ea37887fff call sym.randmd5 ;[3] :sym.randmd5()
0x00046ad 8485beeffff lea eax, [ebp - 0x1042]
0x00046b3 890424 mov dword [esp], eax
0x00046b6 ea37887fff call sym.LinuxExec
0x00046bb e8f1 jmp 0x000467c
: JMP XREF from 0x0004693 (unk)
0x00046b4 885acc21fff mov eax, dword [ebp - 0x3d54]
0x00046b7 8942408 mov dword [esp + 8], eax
0x00046ba 8b45dc mov eax, dword [ebp - 0x24]
0x00046bd 8942404 mov dword [esp + 4], eax
0x00046c2 8485beeffff lea eax, [ebp - 0x1442]
0x00046c9 890424 mov dword [esp], eax
0x00046d7 e878af7fff call sym.writefile ;:sym.writefile(x)
0x00046dc 85c0 test eax, eax
0x00046de 741c je 0x000467c
0x00046e0 8485beeffff lea eax, [ebp - 0x1442]
0x00046e7 890424 mov dword [esp], eax
0x00046ea ea37887fff call sym.randmd5 ;:sym.randmd5()
0x00046ef 8485beeffff lea eax, [ebp - 0x1442]
0x00046f5 890424 mov dword [esp], eax
: XREFS: JMP 0x0004677 JMP 0x00046bb JMP 0x0004633
: XREFS: JMP 0x0004677 JMP 0x00046bb JMP 0x000467c
0x00046fc 8b45dc mov eax, dword [ebp - 0x3d]
0x00046ff 8942404 mov dword [esp + 4], eax ;:1 checks if it runs.
0x0004703 c70424020000 mov dword [esp], 2
0x000470a e80c7fff call sym.HidePdrPort ;:sym.HidePdrPort()
0x000470d e84c2010 call sym.getpid ;:sym.getpid() + sym.__getpid
0x0004714 8942404 mov dword [esp + 4], eax
0x000471b c70424020000 mov dword [esp], 2
0x0004721 e84c2010 call sym.HidePdrPort ;:sym.HidePdrPort()
0x0004724 8485c87ffff lea eax, [ebp - 0x38]
0x000472a 890424 mov dword [esp], eax
0x000472d e84c2010 call sym.remove ;:sym.remove()
0x0004732 c78544c2ffff mov dword [ebp - 0x3bc], 0
0x000473c e878af0000 jmp 0x00046bb ;cleaning up , retn & goto next process

```



Malware Characterization

Bytes

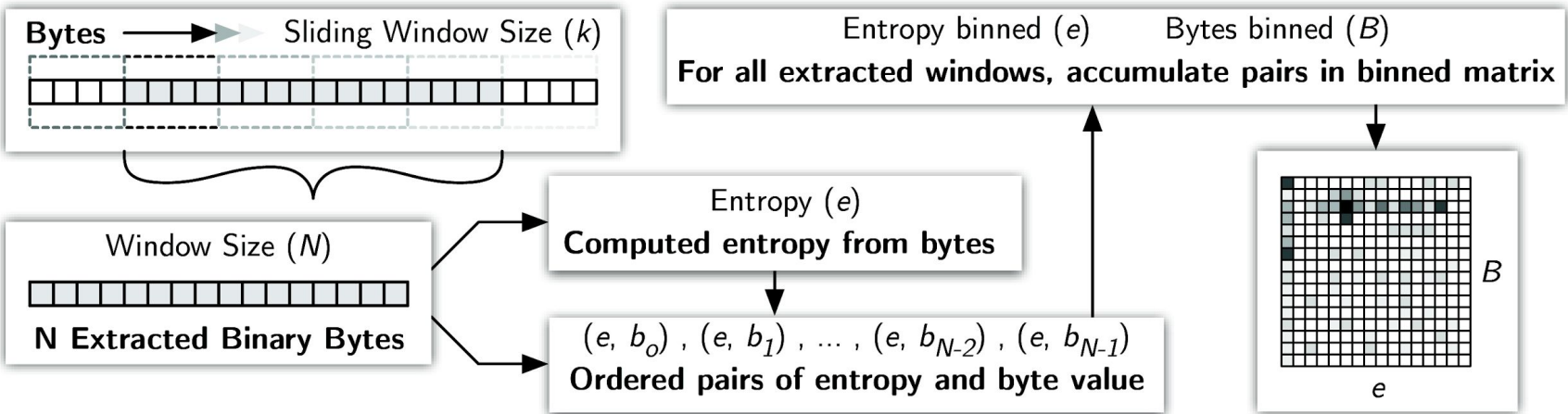
- Shannon Entropy
- Bytes N-grams
- Strings

Codes

- Instruction N-grams
- Statistics
 - Function, Blocks
 - Calls, Branches



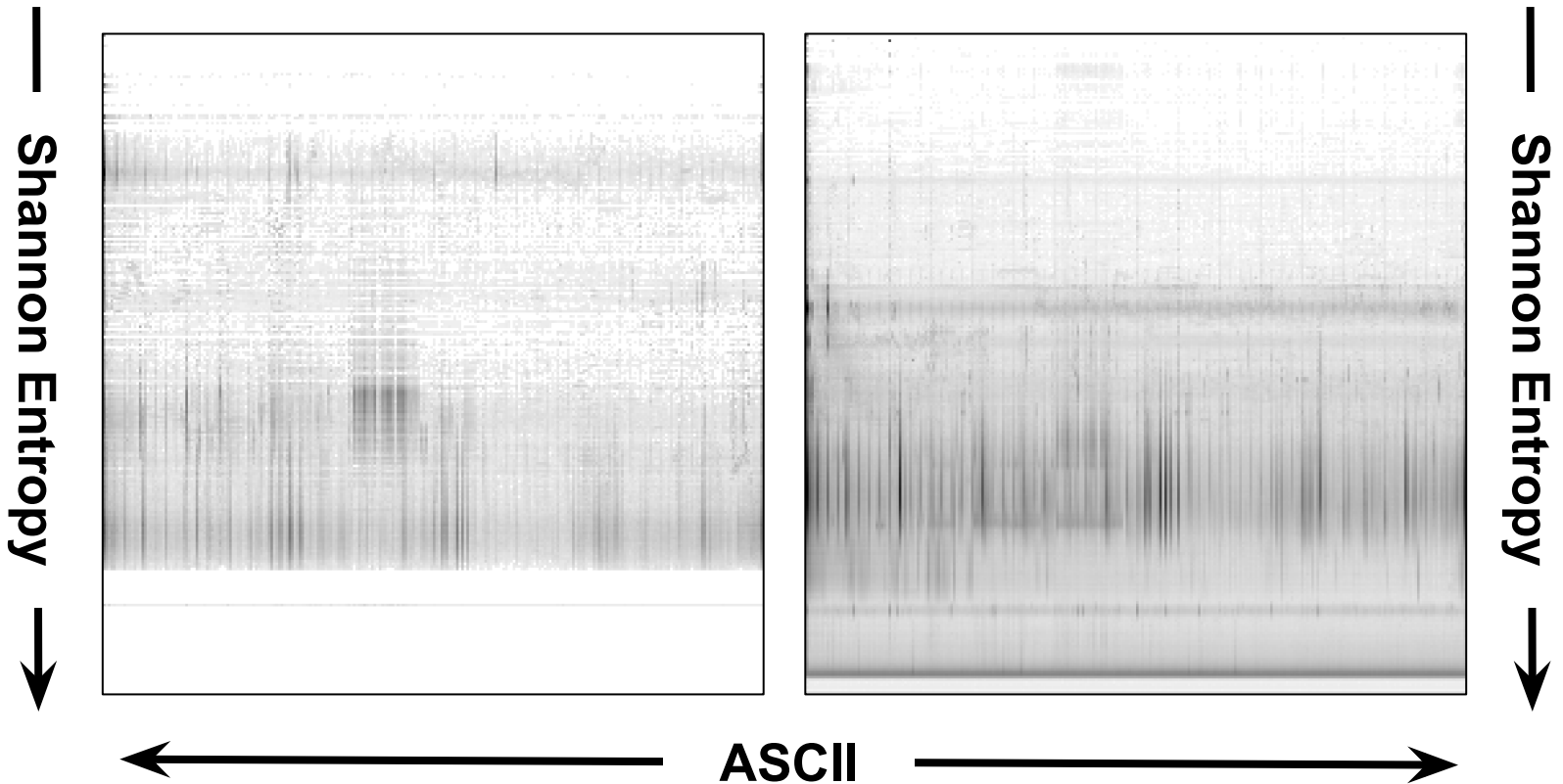
State-of-the-art: Bytes Analysis



[Saxe and Berlin, 2015] Deep neural network based malware detection using two dimensional binary program features.



State-of-the-art: Bytes Analysis





Our Approach: Code Analysis

Program ASM

```

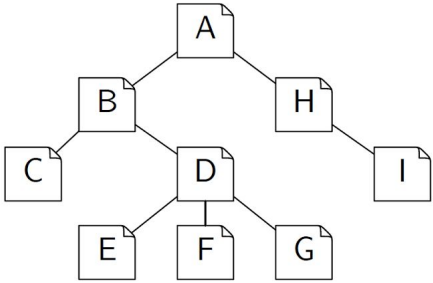
foo:
  push  %ebp
  mov   %esp,%ebp
  mov   %eax,-0x4c(%ebp,%eax)
  call  0x1234(%rip,%eax)
  mov   %eax,%eax
  add   %eax,%eax
  jmp   %eax,%eax
  retq

main:
  push  %ebp
  mov   %esp,%ebp
  push  %ebx
  mov   %eax,%eax
  sub   %eax,%eax
  movl  $0x0,-0xc(%rip,%eax)
  jmp   0x00000000-0x1234(%rip,%eax)
  mov   %eax,%eax
  mov   -0xc(%rip,%eax),%eax
  mov   -0xc(%rip,%eax),%eax
  movd  0x481000(%rip,%eax),%eax
  call  0x1234(%rip,%eax)
  mov   %eax,%eax
  mov   %eax,%eax
  call  0x00000000-0x0000(%rip,%eax)
  movd  %eax,0x00000000(%rip,%eax)
  addl  $0x1,-0xc(%rip,%eax)
  cmpl  $0x3,-0xc(%rip,%eax)
  jle   0x00000000-0x1234(%rip,%eax)
  add  $0x1,%esp
  jmp   %ebx
  retq

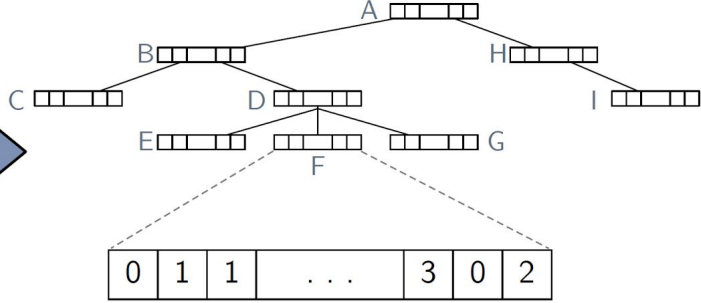
```



Call Graph



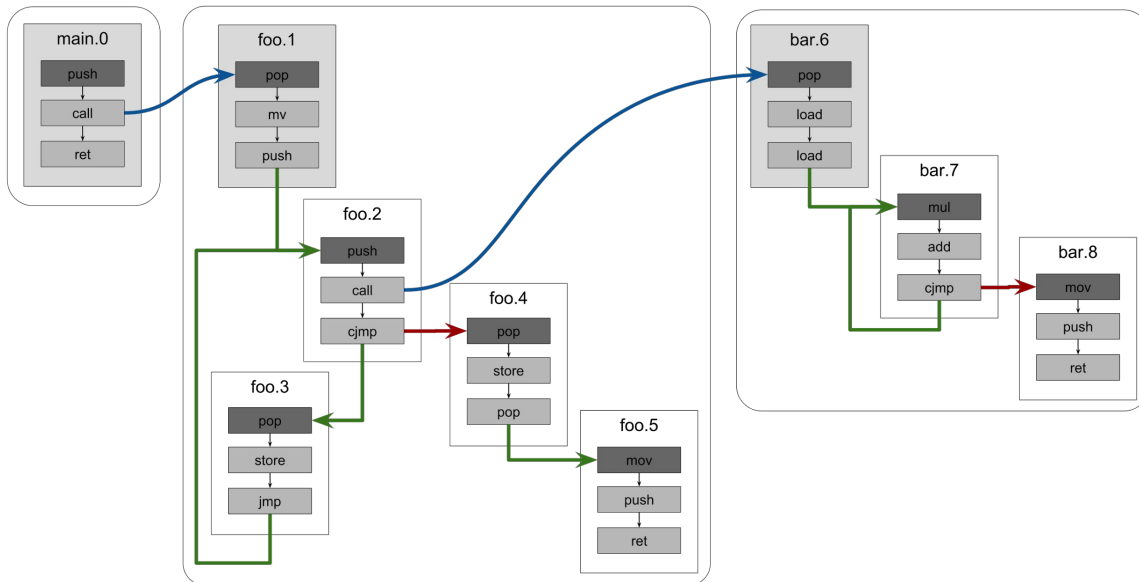
Extracted Features



Reverse engineering: Radare2
 Extract code from: x86, JAR, JS, etc



Compiler Internal Representation



Instruction Flow Graph

Control Flow Graph

Call Graph

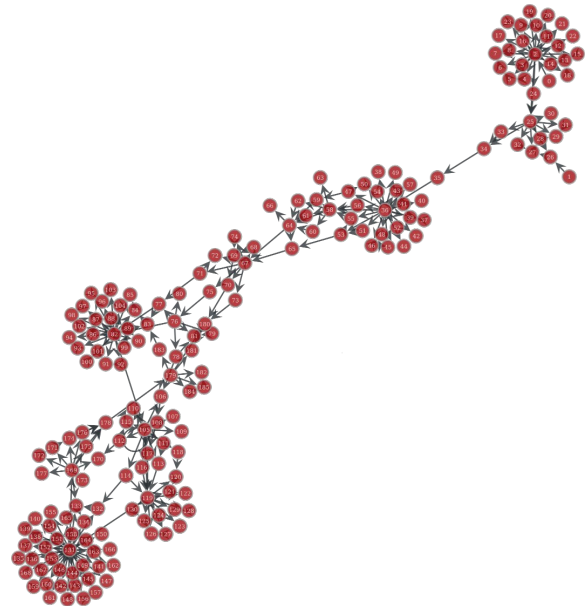
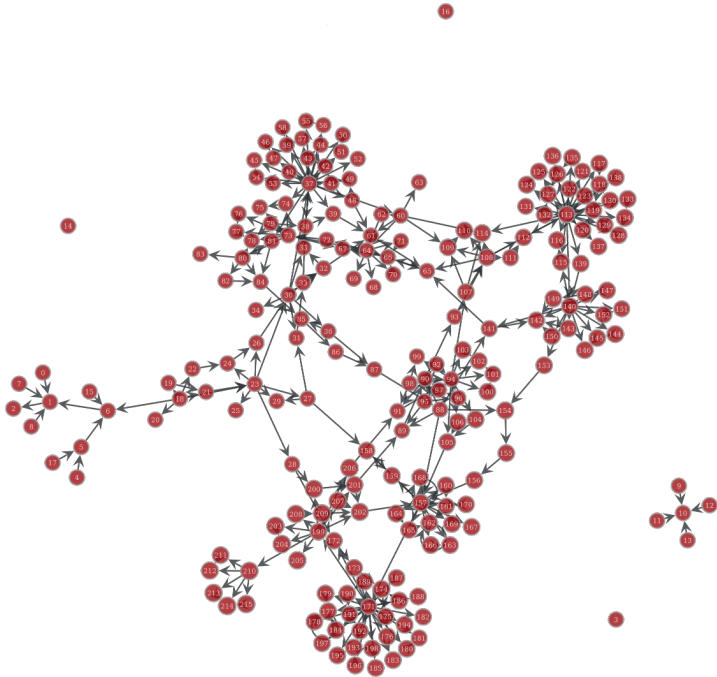
Statistics

Instructions 1-grams

Instructions 2-grams



Compiler-based Graphs

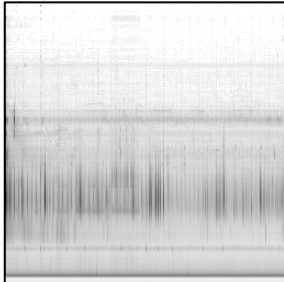




Malware Characterization

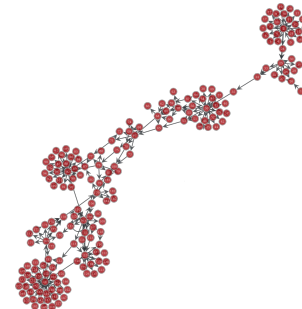
Bytes-Entropy Histogram

- Signature of the binary
- 256 x 256 positive integers
- Fast to extract (~ms)



Disassembled Code

- Multiple graphs
- No fixed size
- Long reverse engineering (~s)





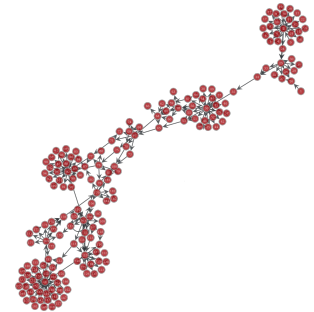
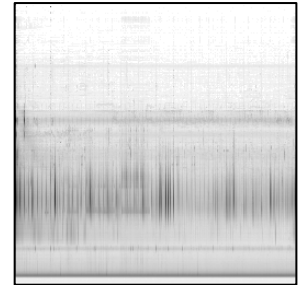
ML + Compiler-based Graphs for Malware

Malware Classification

Millions to classified

- Reversing Labs
- Virus Total

Models trained often



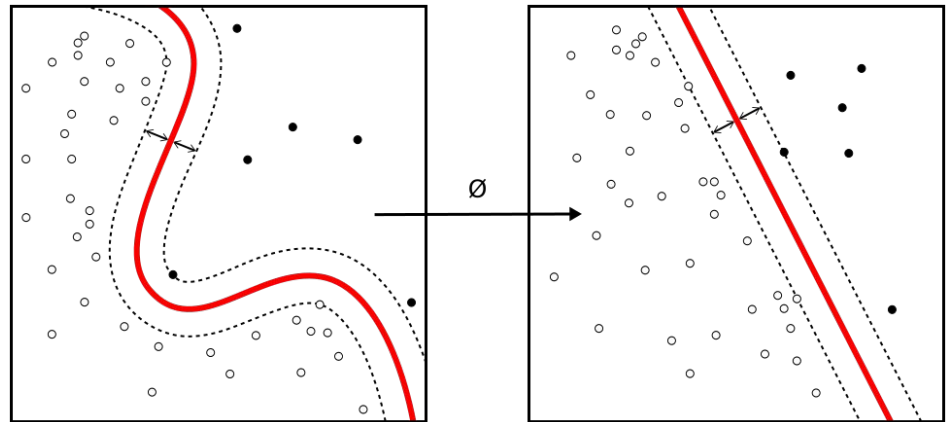


SVM and Graph Kernel

Support Vector Machine

- Linear Separator
- Kernel Trick
- Graph Kernels

Problem: **Poor Scaling**

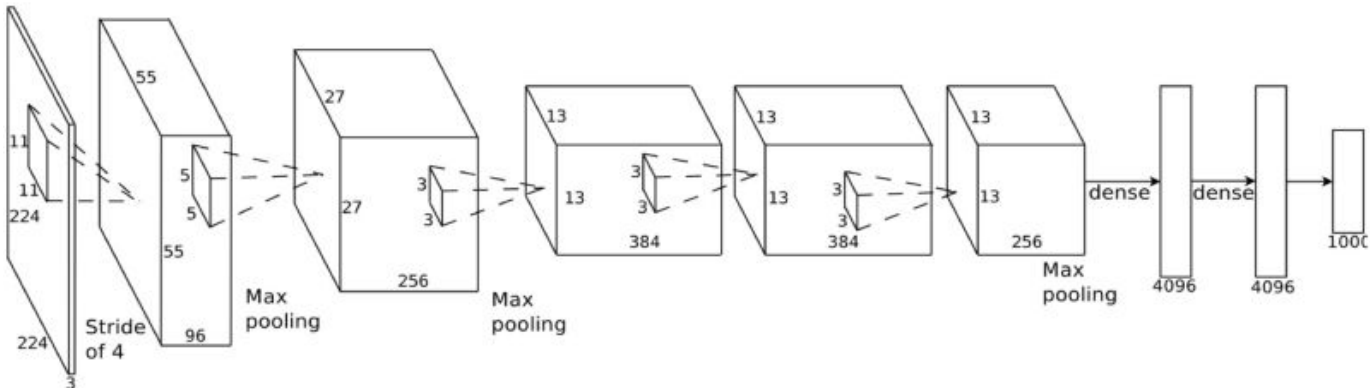
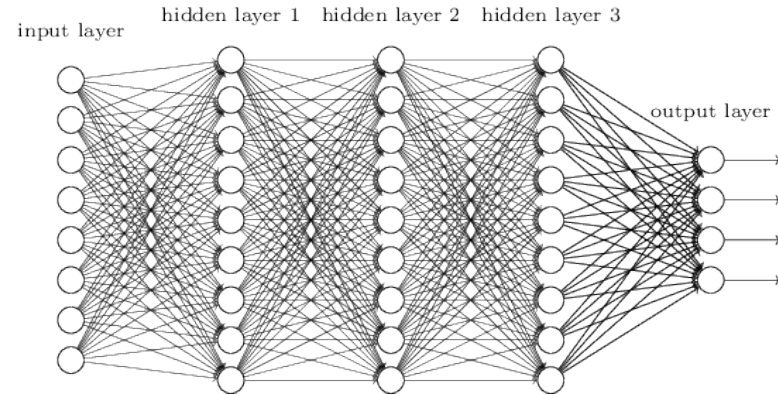




Deep Learning

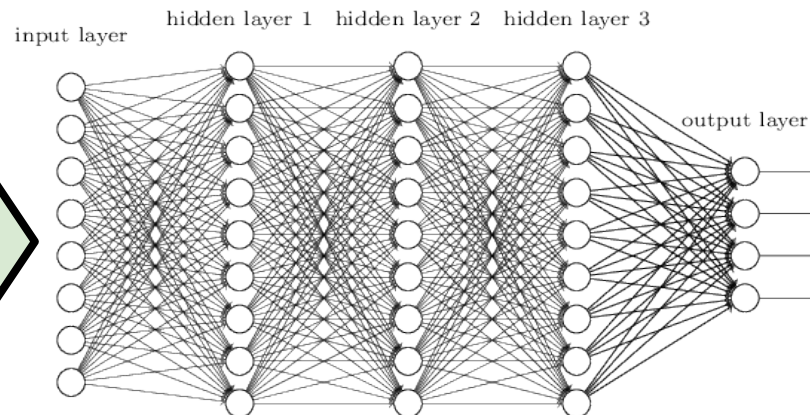
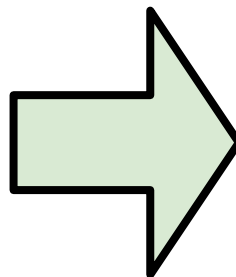
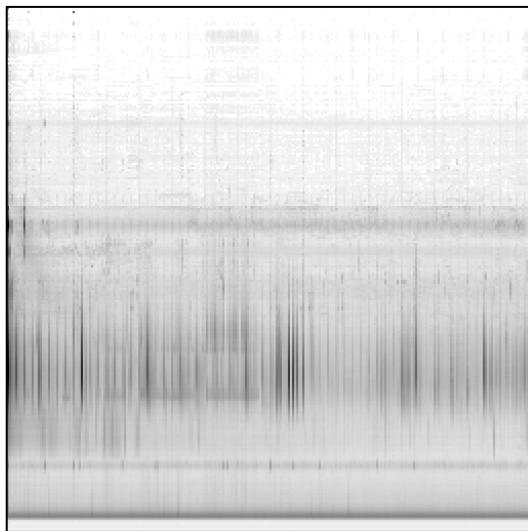
Old concept

- Large model expensive
- Scale with dataset size
- Convolutional neural networks



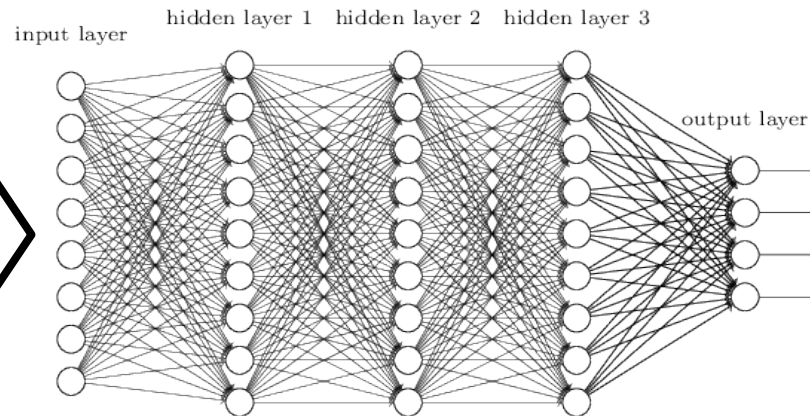
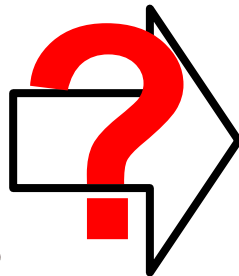
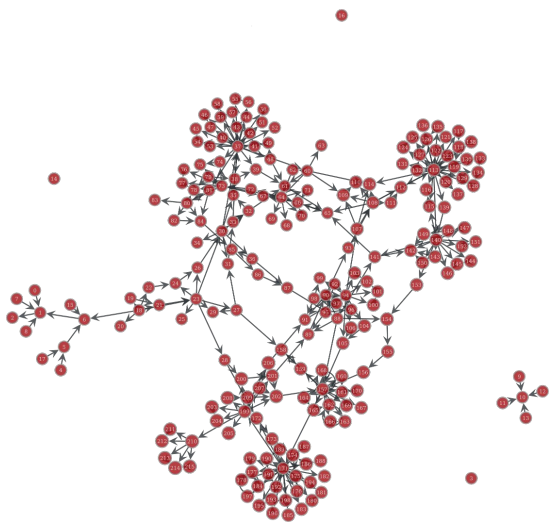


Deep Learning on Byte-Entropy Histogram





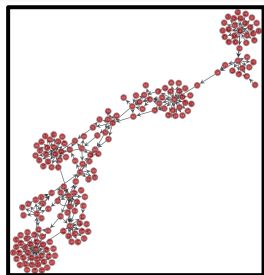
Deep Learning on Graphs



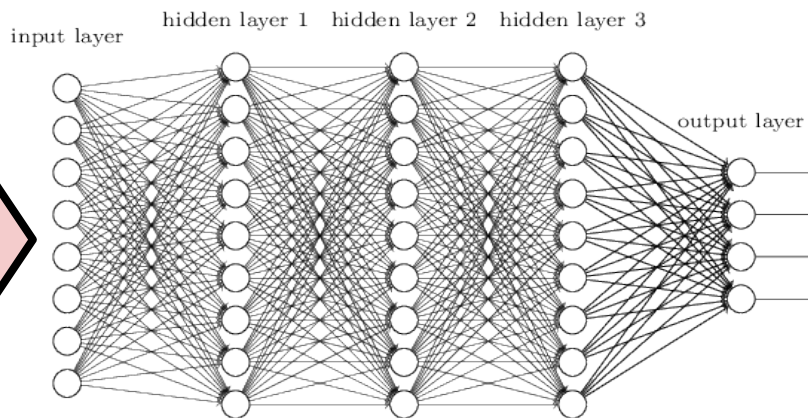
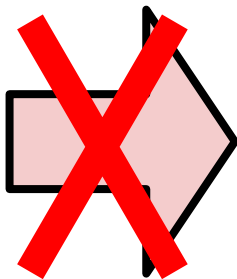
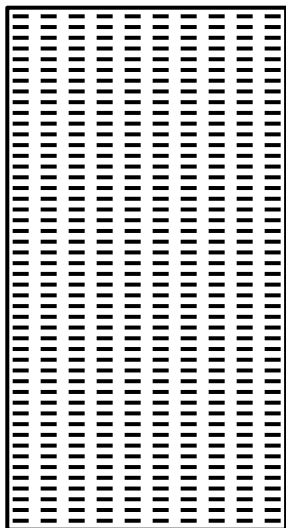


Deep Learning on Graphs

Adjacency Matrix



Node Features

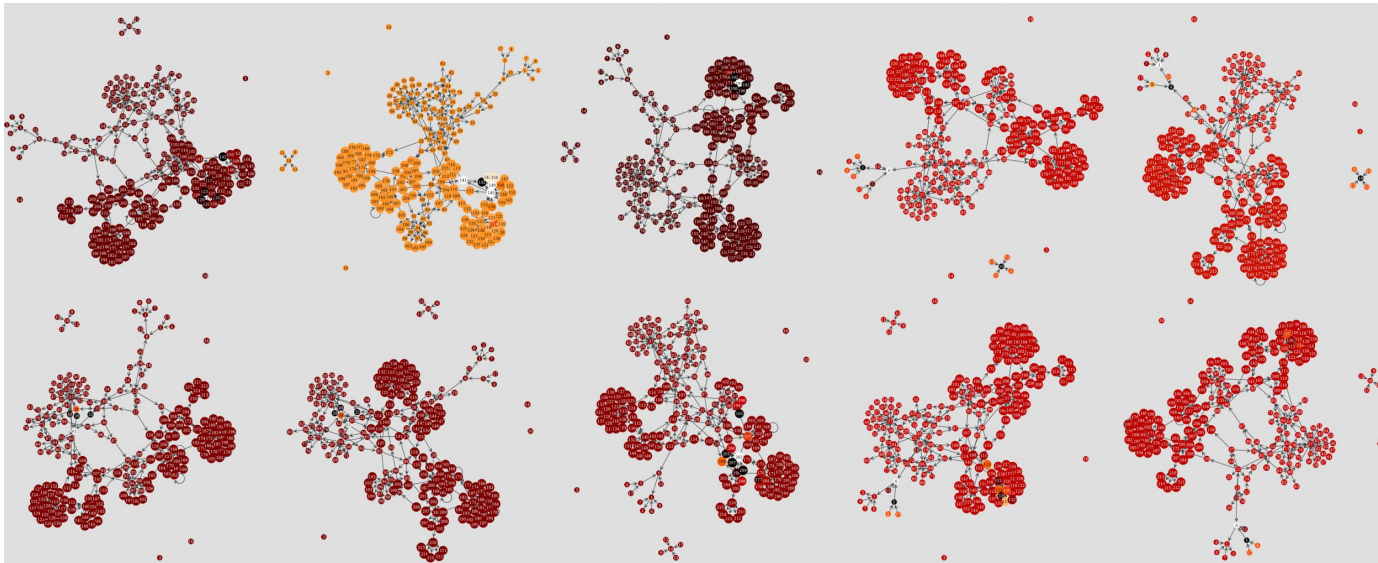


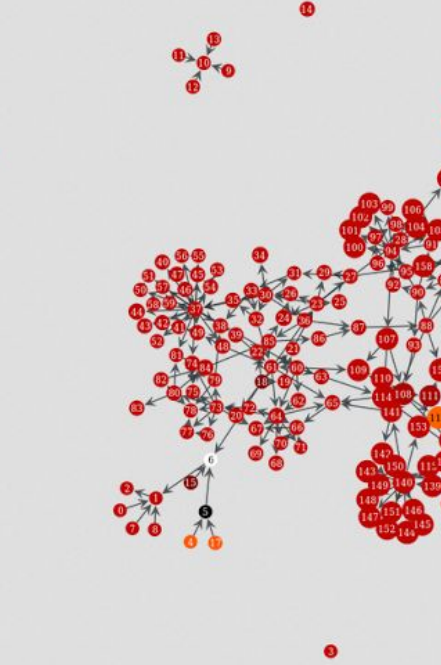
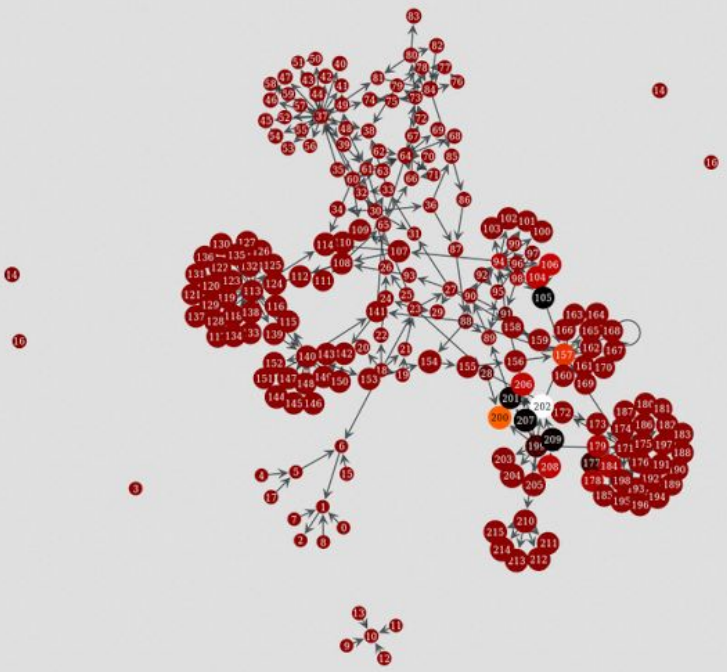
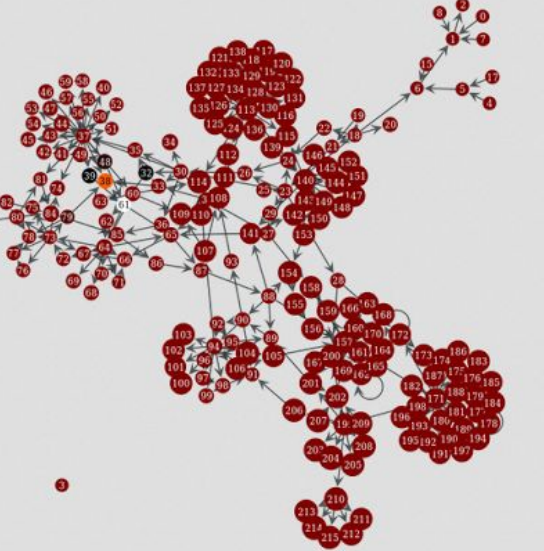
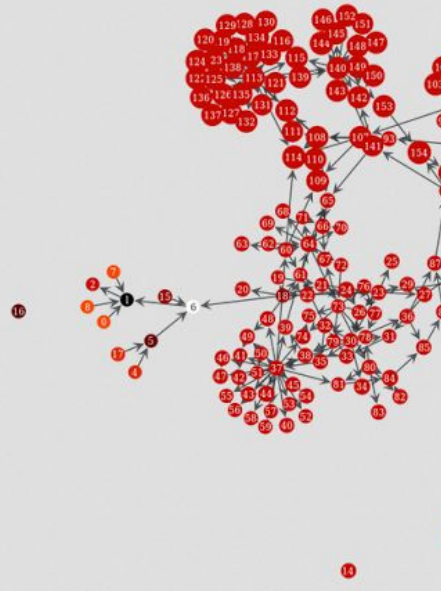
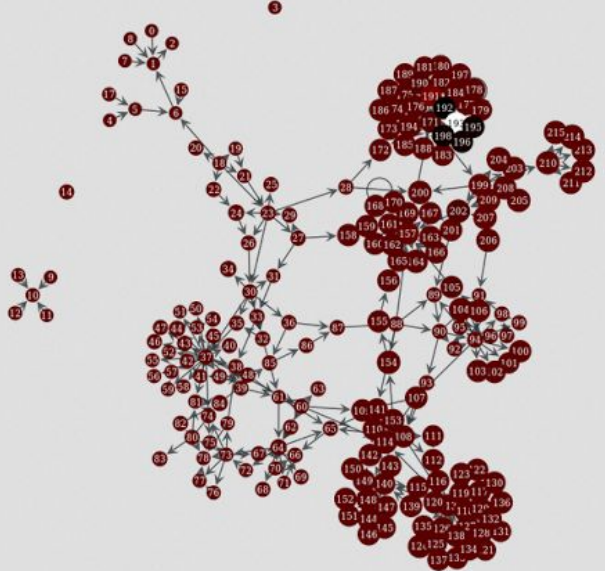
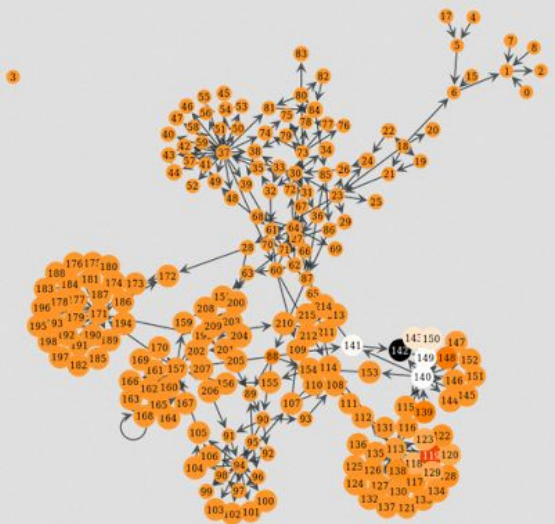


Graph Spectral Analysis

Eigenvectors of the graph-Laplacian

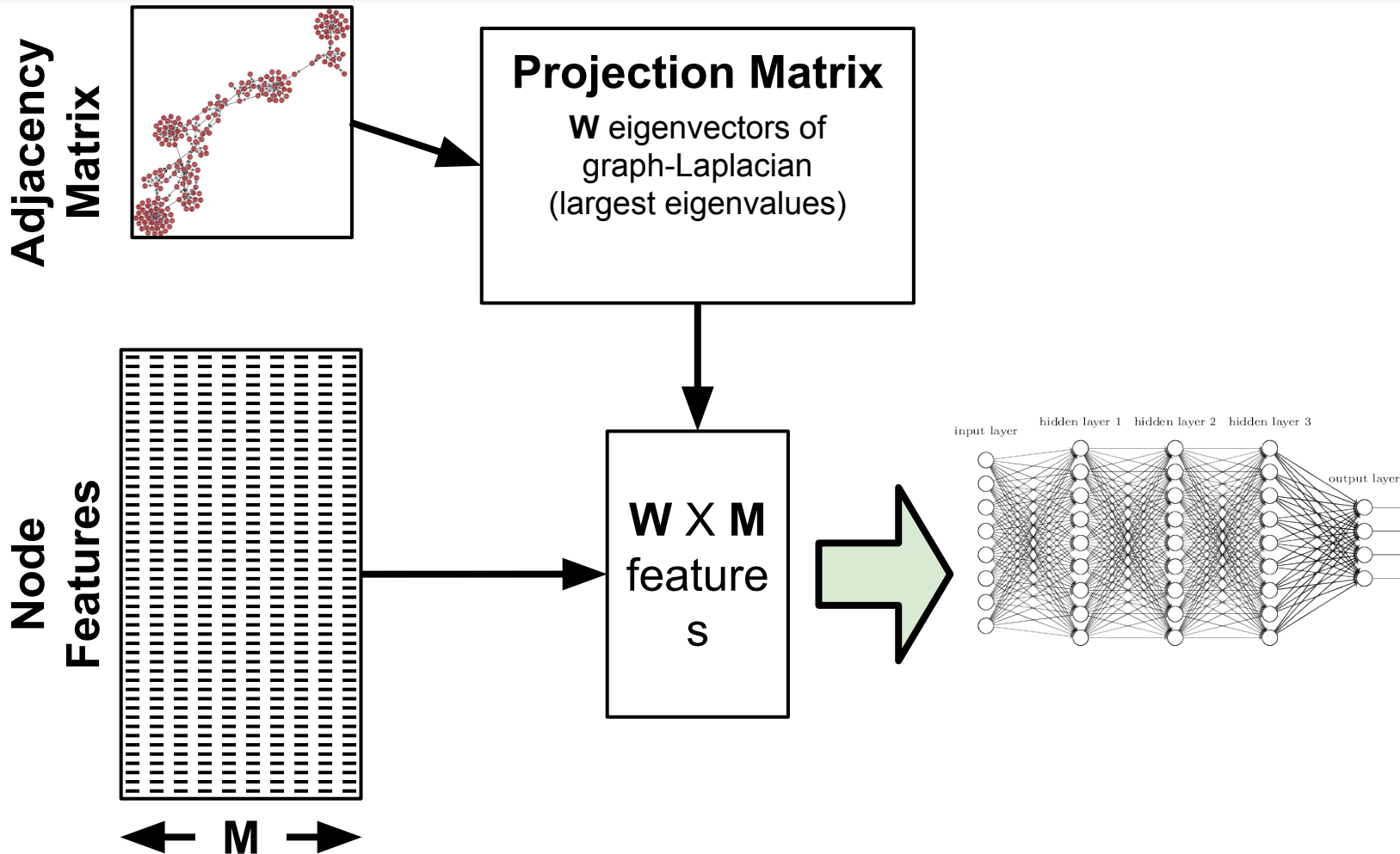
[Bronstein et al., 2016] *Geometric deep learning: going beyond euclidean data*





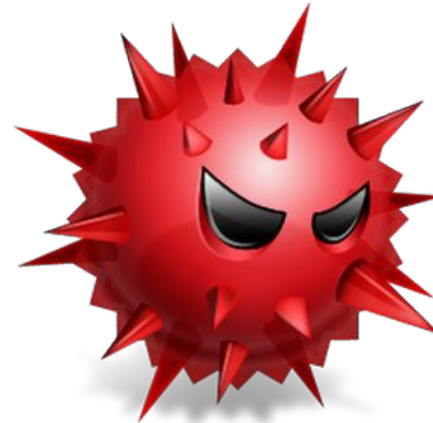


Graph Spectral Features





Malware Classification





Malware Stream

Reversing Labs

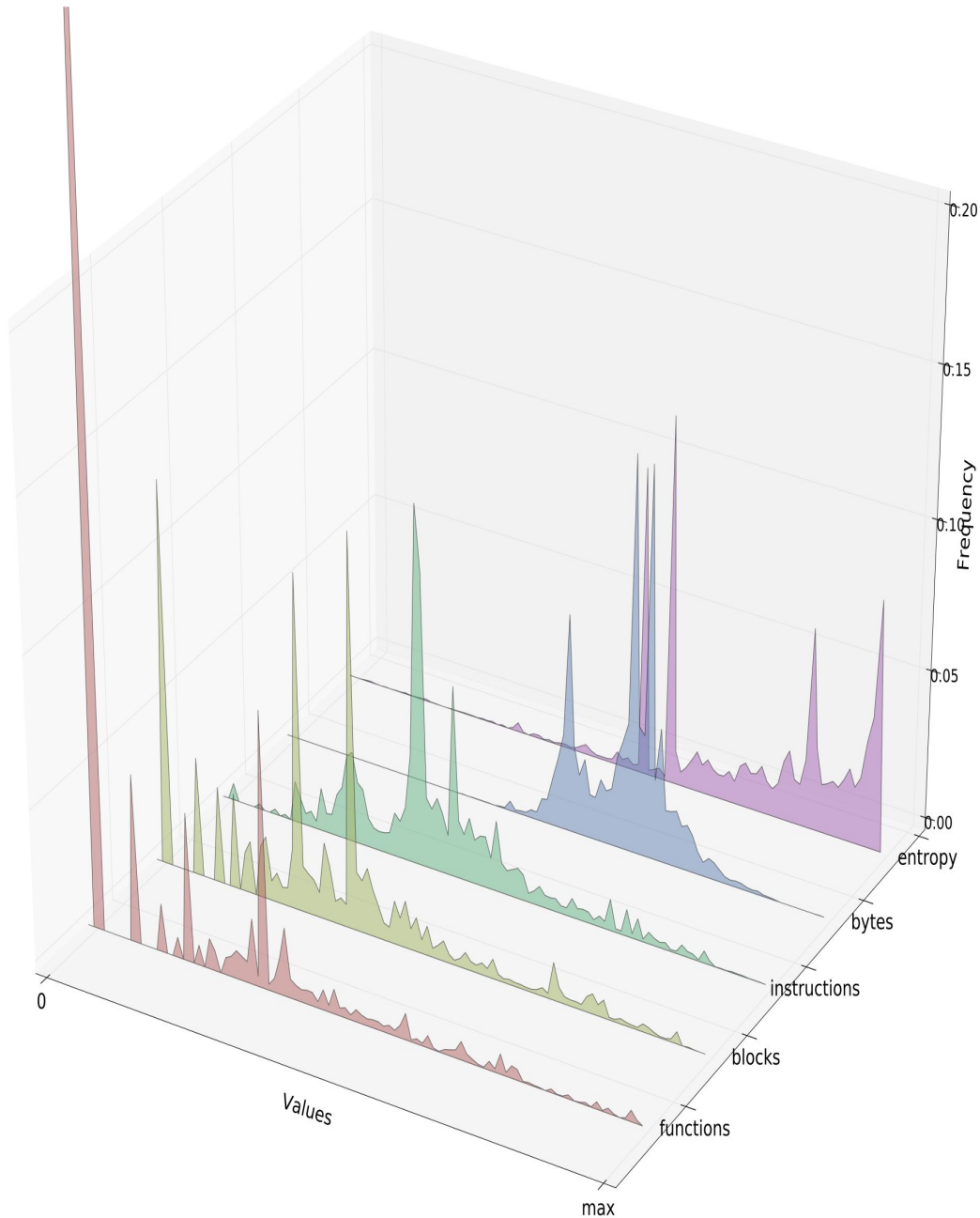
- Billions of malware
- Curated streams

Financial stream

- 1.2 millions
- 40+ families

Selected

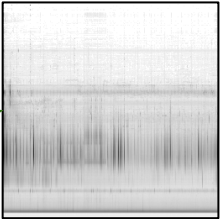
- families with more than 1,000 malware




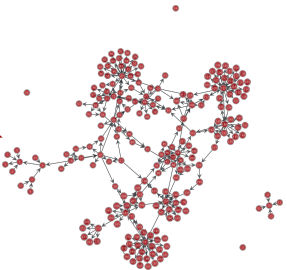


Malware Features for Deep Learning

Characterization	Format	Size
bytes-entropy histogram	matrix	256 x 256
Global	statistics	43
	1-grams	53
	2-grams	2809
Function	statistics	20 x 23
	1-grams	20 x 53
	2-grams	20 x 2809
Block	statistics	20 x 10
	1-grams	20 x 53
	2-grams	20 x 2809
Operations	statistic	20 x 2




Customer Feature Vector





Characterization		Format	Size	DNN layers	DNN shape
bytes-entropy histogram		matrix	256 x 256	12	geometric
Global	statistics	vector	43	5	arithmetic
	1-grams		53	6	arithmetic
	2-grams		2809	9	geometric
Function	statistics	matrix	20 x 23	7	geometric
	1-grams		20 x 53	8	geometric
	2-grams		20 x 2809	12	geometric
Block	statistics	matrix	20 x 10	5	geometric
	1-grams		20 x 53	8	geometric
	2-grams		20 x 2809	12	geometric
Operations	statistic	matrix	20 x 2	5	arithmetic



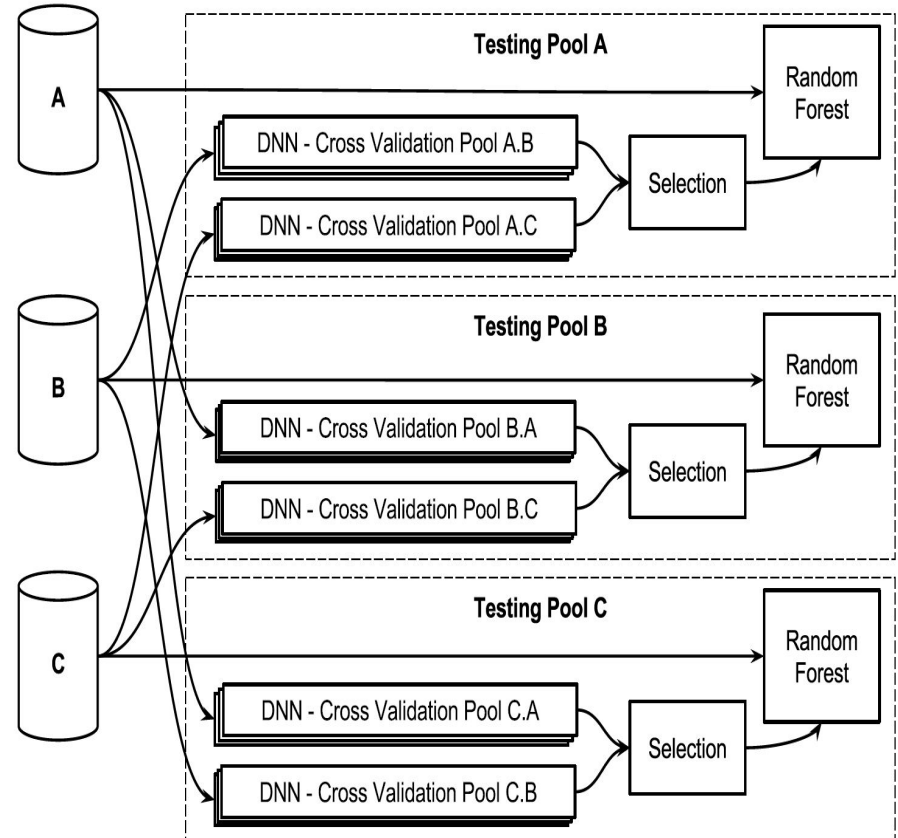
Characterization		Format	Size	DNN layers	DNN shape
bytes-entropy histogram		matrix	256 x 256	12	geometric
Global	statistics	vector	43	5	arithmetic
	1-grams		53	6	arithmetic
	2-grams		2809	9	geometric
Function	statistics	matrix	20 x 23	7	geometric
	1-grams		20 x 53	8	geometric
	2-grams		20 x 2809	12	geometric
Block	statistics	matrix	20 x 10	5	geometric
	1-grams		20 x 53	8	geometric
	2-grams		20 x 2809	12	geometric
Operations	statistic	matrix	20 x 2	5	arithmetic



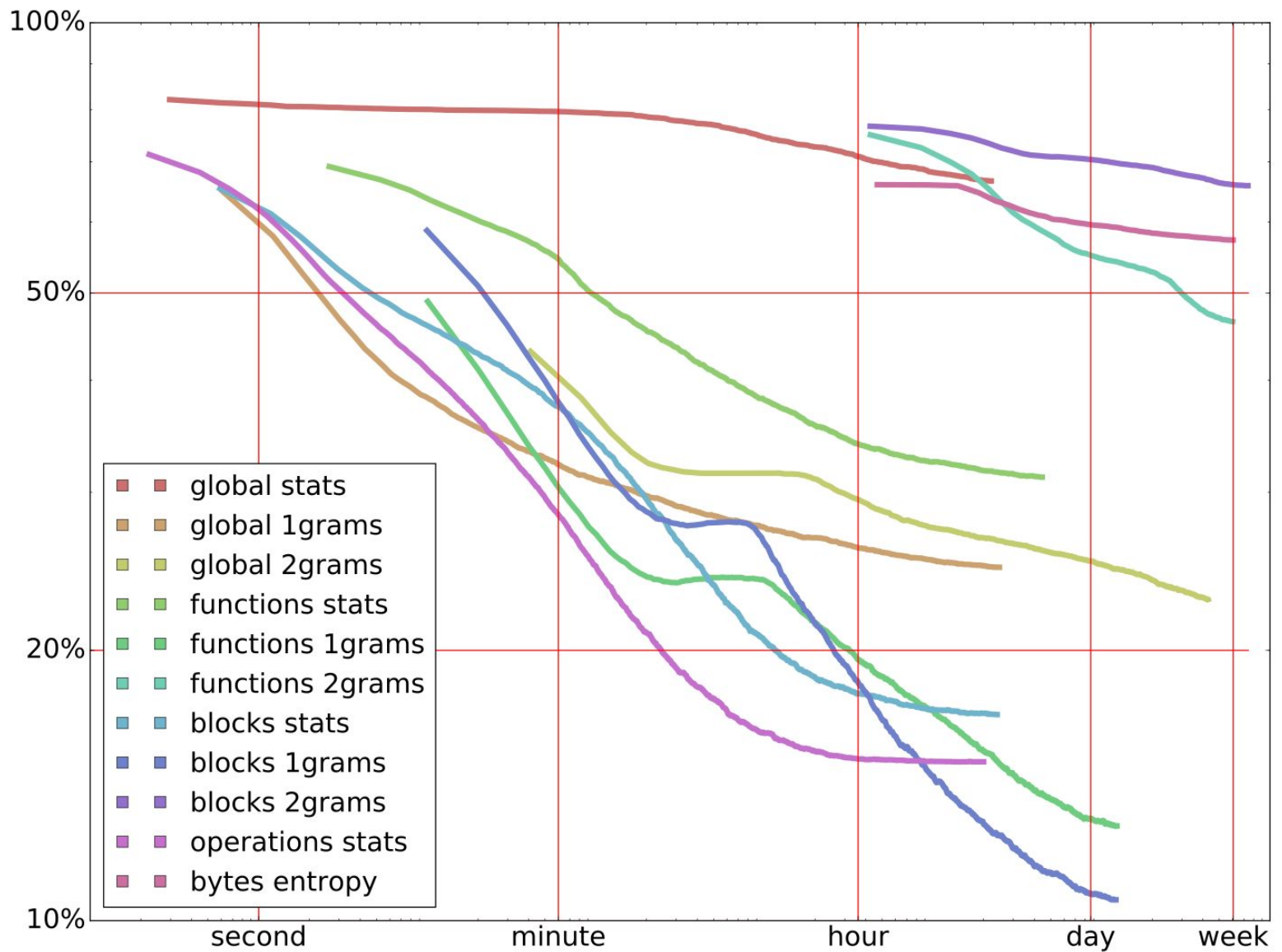
Cross-validation and Consensus

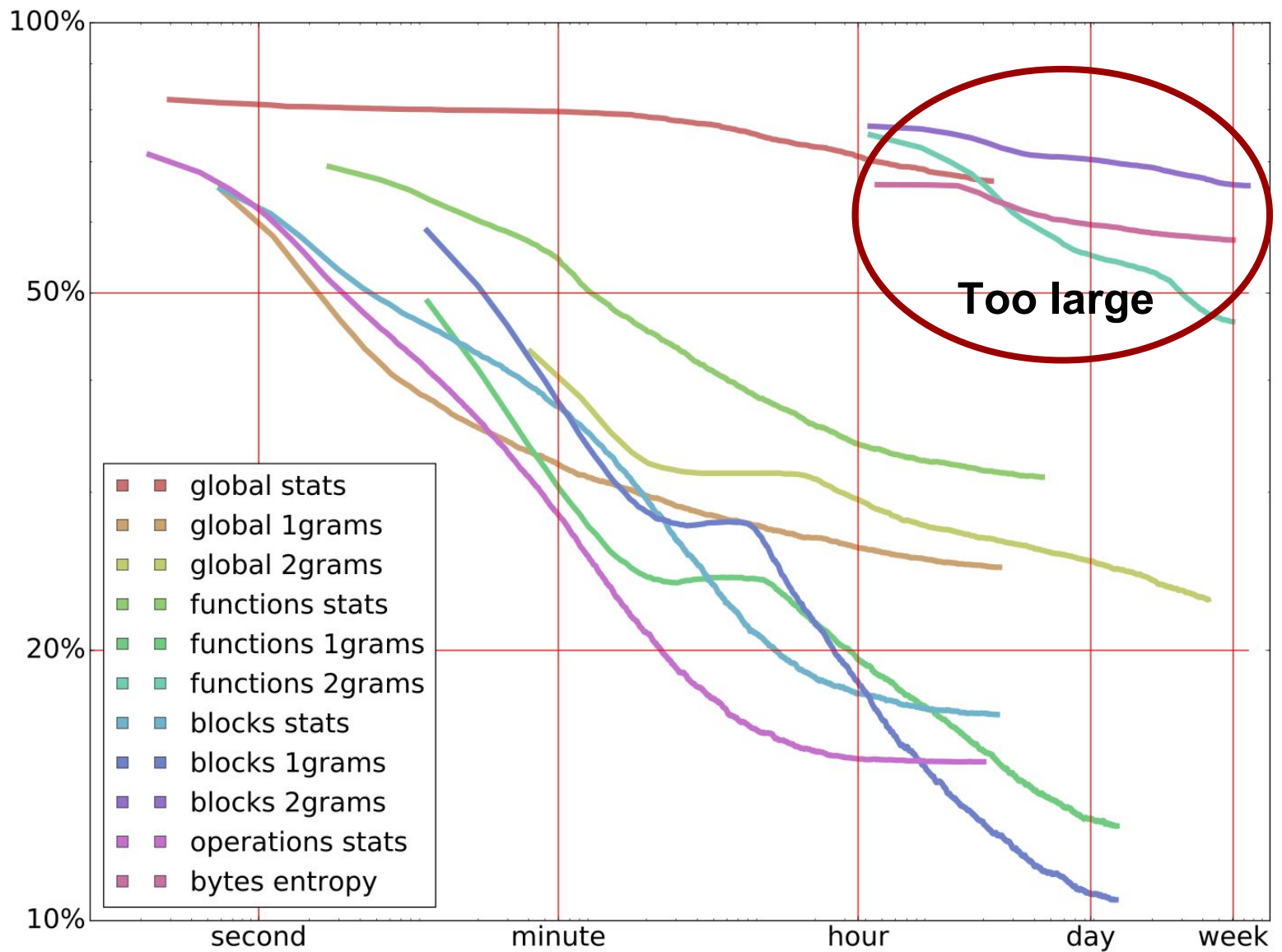
5-fold cross-validation

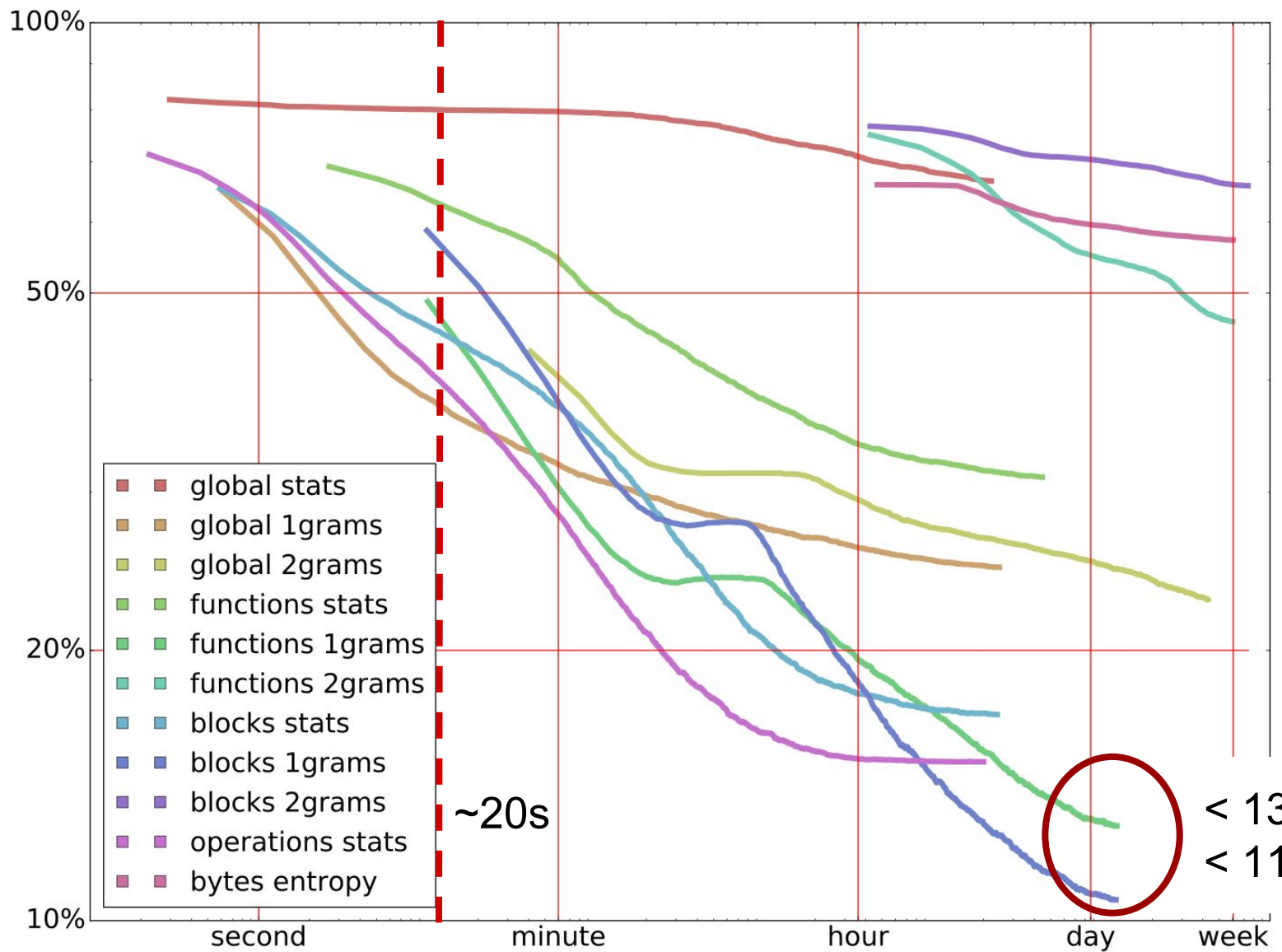
- Testing pools
- Cross-validation pool
 - Many DNNs
- Selection
 - K best DNNs
- DNN ensemble
 - Consensus



5 testing pools * 4 CV pools * 10 DNNs = **200 DNNs**









Models		1 best	2 best	5 best
bytes-entropy histogram		39.2%	32.8%	27.3%
Global	statistics	60.5%	53.2%	44.4%
	1-grams	22.9%	20.8%	19.1%
	2-grams	20.7%	19.2%	18.5%
Function	statistics	26.5%	23.8%	20.3%
	1-grams	12.3%	12.0%	10.8%
	2-grams	30.5%	27.2%	25.7%
Block	statistics	15.8%	14.8%	13.6%
	1-grams	10.3%	9.9%	9.9%
	2-grams	39.2%	35.1%	31.4%
Operations	statistic	13.6%	11.3%	10.4%



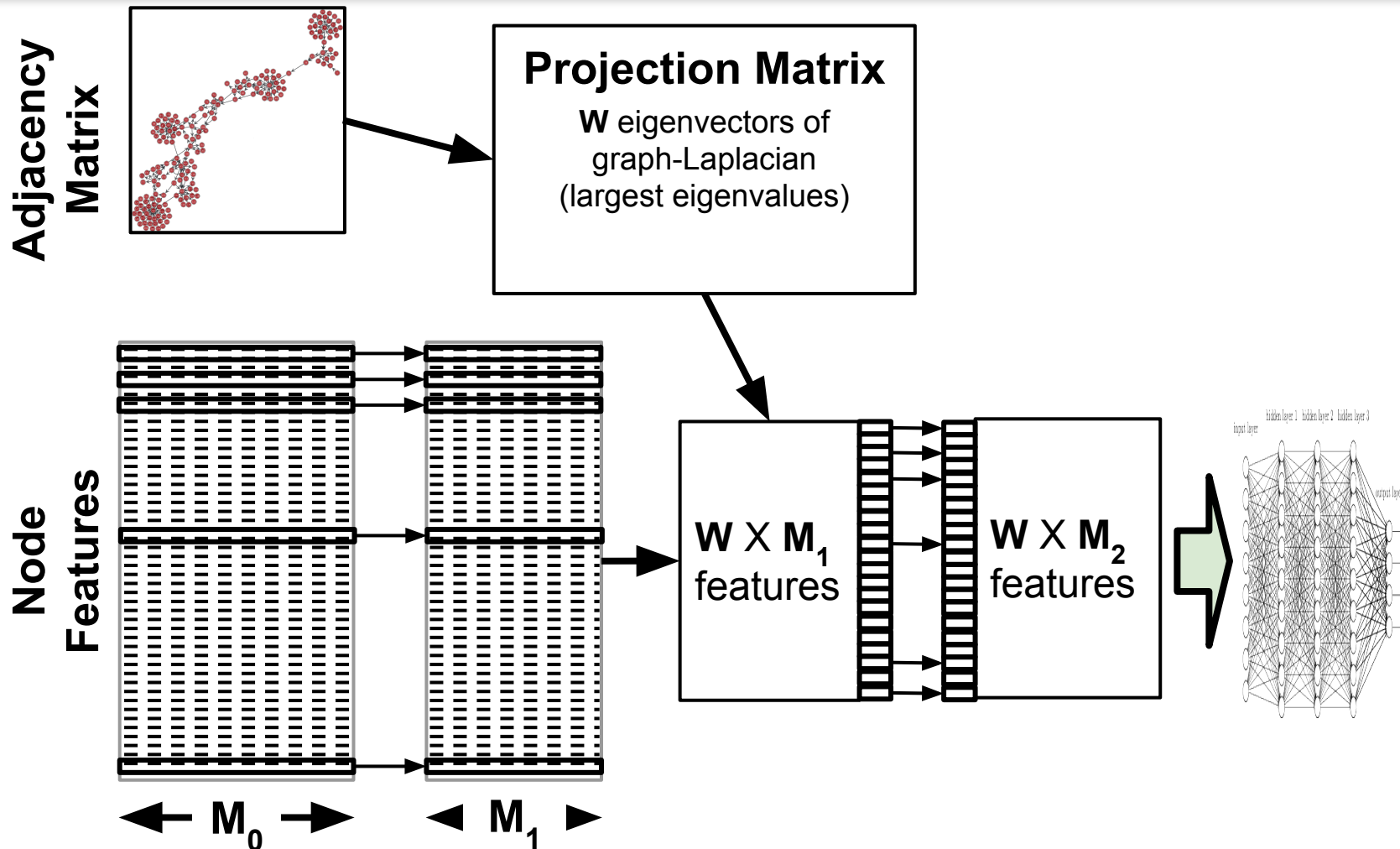
Best Results: DNN Ensemble

Models	1 best	2 best	5 best
BEH & Global level	16.0%	15.3%	13.8%
Compiler Graphs	8.4%	8.0%	8.0%
All Features	6.9%	6.5%	6.3%

More than twice as good when adding
Compiler-based Graphs



Improving Deep Learning Model

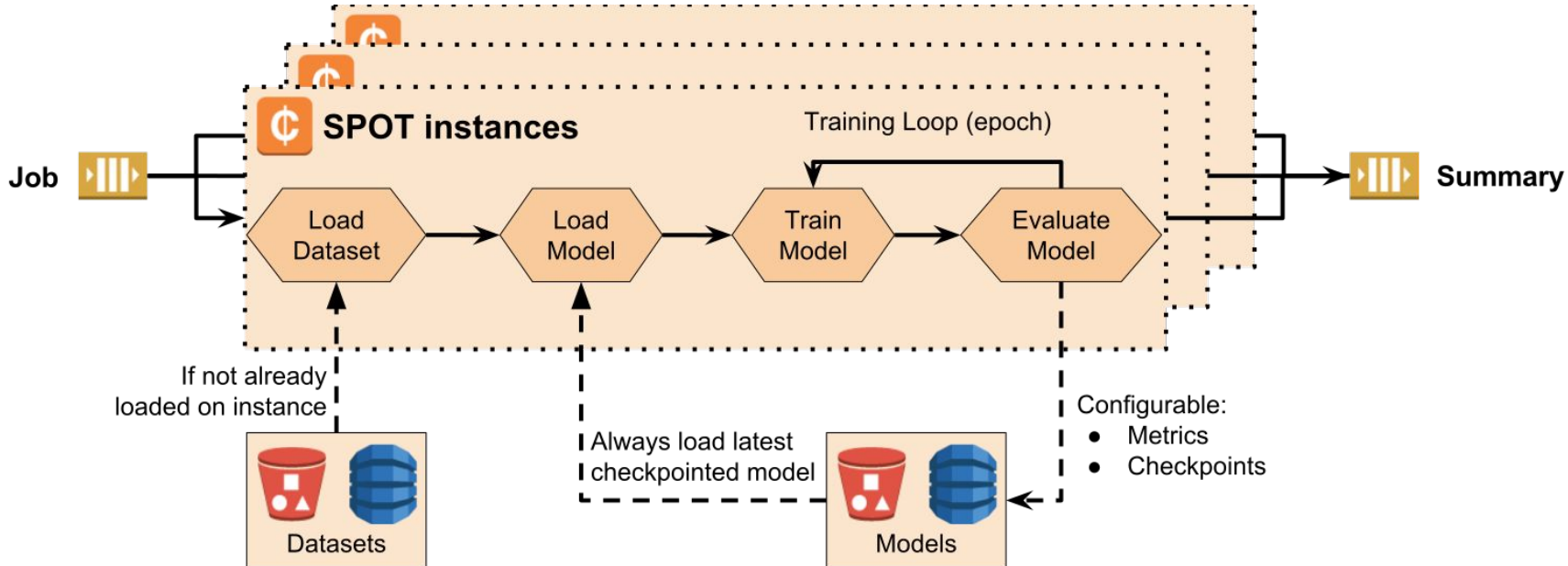




Classifying Millions of Malware

New DNN model

- Less parameters to learn
 - Train on full malware dataset





Dataset for the class

- Files
 - ***financial-1000***
 - ***All ?***
- New features
 - ASCII strings
 - More classifications sources (VirusTotal)
- Lessons learned
 - Bytes-Entropy Histograms
 - 16 entropy bins (16x256)
 - Different sliding windows (1024 by 256 **and** 2048 by 128)
 - Assembly Graphs
 - Removed 2-grams
 - 1-grams at operation level



QUESTIONS ?

Characterization

Deep-learning

Datasets

Results



CISC850: DevOps



Tristan Vanderbruggen

Dept of Computer & Information Sciences

University of Delaware



Tools

- Communication
 - Slack
- Version Control
 - GitHub
 - git
- Scrum
 - Waffle I/O
- Cloud
 - EC2
 - S3





Teams

- Analysis 1
 - Leonardo De La Rosa
 - Yang Yang
 - Yuhao Peng
- Analysis 2
 - Sean Kilgallon
 - Ashwag Altayyar
 - Peng Su
- Chatbot
 - Aman Sawhney
 - Abhijeet Srivastava
 - Anupam Basu
- Graphs
 - Ian Lantzy
 - Fan Li
 - Paul Soper
- Visual Analytics 1
 - Wanxin Li
 - Zicheng Liu
 - Ezeanaka Kingsley
 - Abdulrahaman Alshammari
- Visual Analytics 2
 - Yujun Zeng
 - Ruikai Zheng
 - Hancheng Zhao
 - Ruijie Xi
- Machine Learning
 - Vinit Singh
 - Abhilash Parthasarathy
 - Mingxing Gong



- Slack
 - <https://ud-cisc850.slack.com/signup>
 - Team lead create channel
 - Invite other members
 - Invite Dr Cavazos
 - Send me **private message** with:
 - name
 - project
 - **GitHub** username



Version Control & Scrum

- Version Control: GitHub
 - <https://github.com/cavazos-lab/>
 - Find repo for your group
 - spring-2017-CISC850-XXX
- Scrum: Waffle IO
 - <https://waffle.io/cavazos-lab/spring-2017-CISC850-XXX>
 - Create backlogs:
 - Documentations
 - Project design



- Start instance
- SSH in the instance
- git CLI
 - `sudo yum install git vim emacs`
 - `git clone https://github.com/cavazos-lab/XXX XXX`
 - `cd XXX`
 - `vim README.md`
 - **Make changes: each member fill a section**
 - `git status`
 - `git add README.md`
 - `git commit -m "update README with ..."`
 - `git push origin master`
 - **???**
- More on GIT:
 - GIT Book: <https://git-scm.com/book/en/v2>



QUESTIONS ?

AWS

GIT

Scrum