Adaptive and Semantics-aware Machine Learning-based Malware Detection

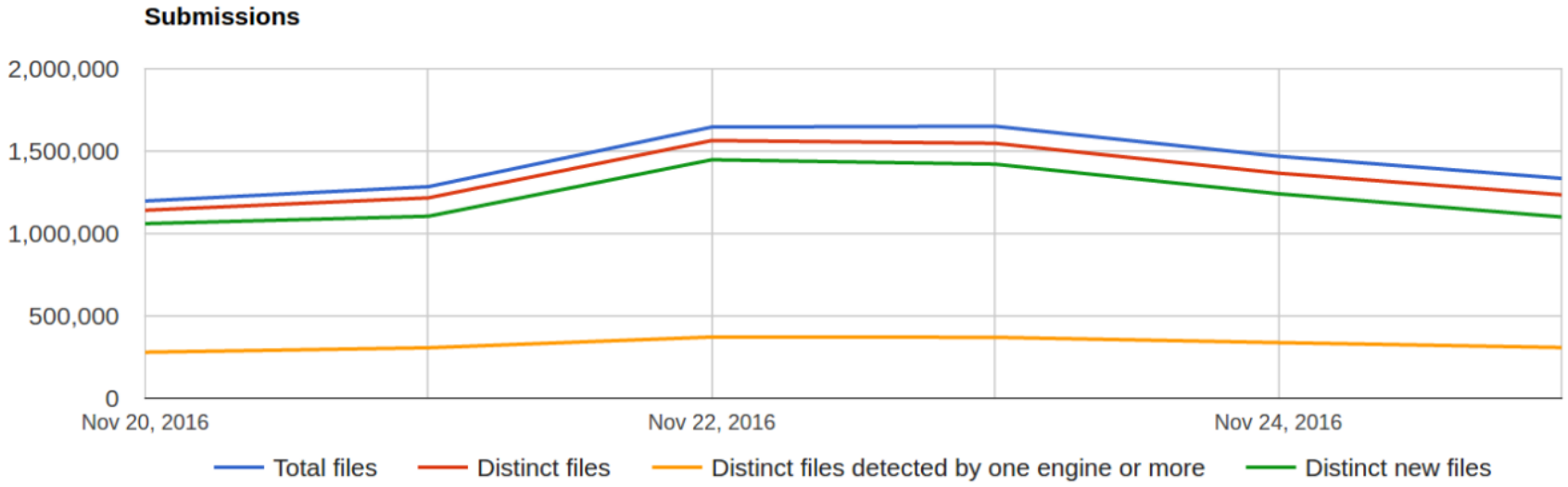
Bojan Kolosnjaji
Chair of IT Security
Technical University of Munich
Germany

Seminar talk, Data61

Sydney, Australia, 13.12.2016.

Problem statement

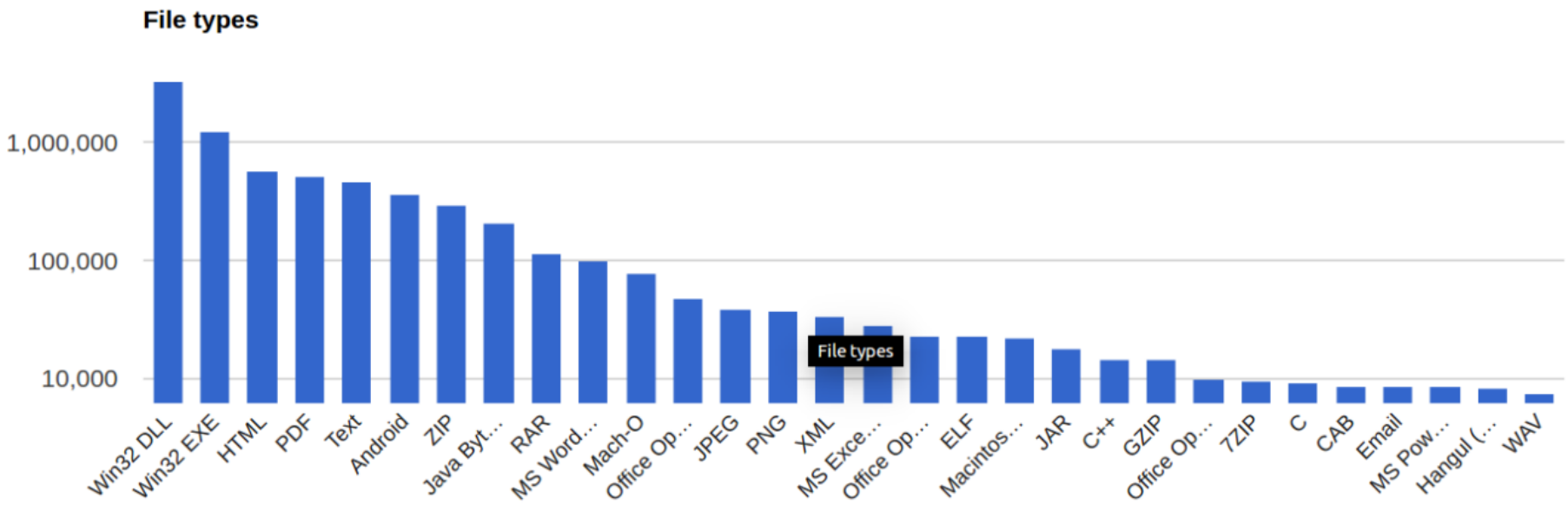
- Increase in **number** and **variety** of newly detected samples
- How to scale up the analysis?
- Use knowledge about similar samples, malware families, code reuse



www.virustotal.com/r/statistics

Problem statement

- Mostly windows PE (*.exe) files and DLL



Problem statement

Malware detection and triage process

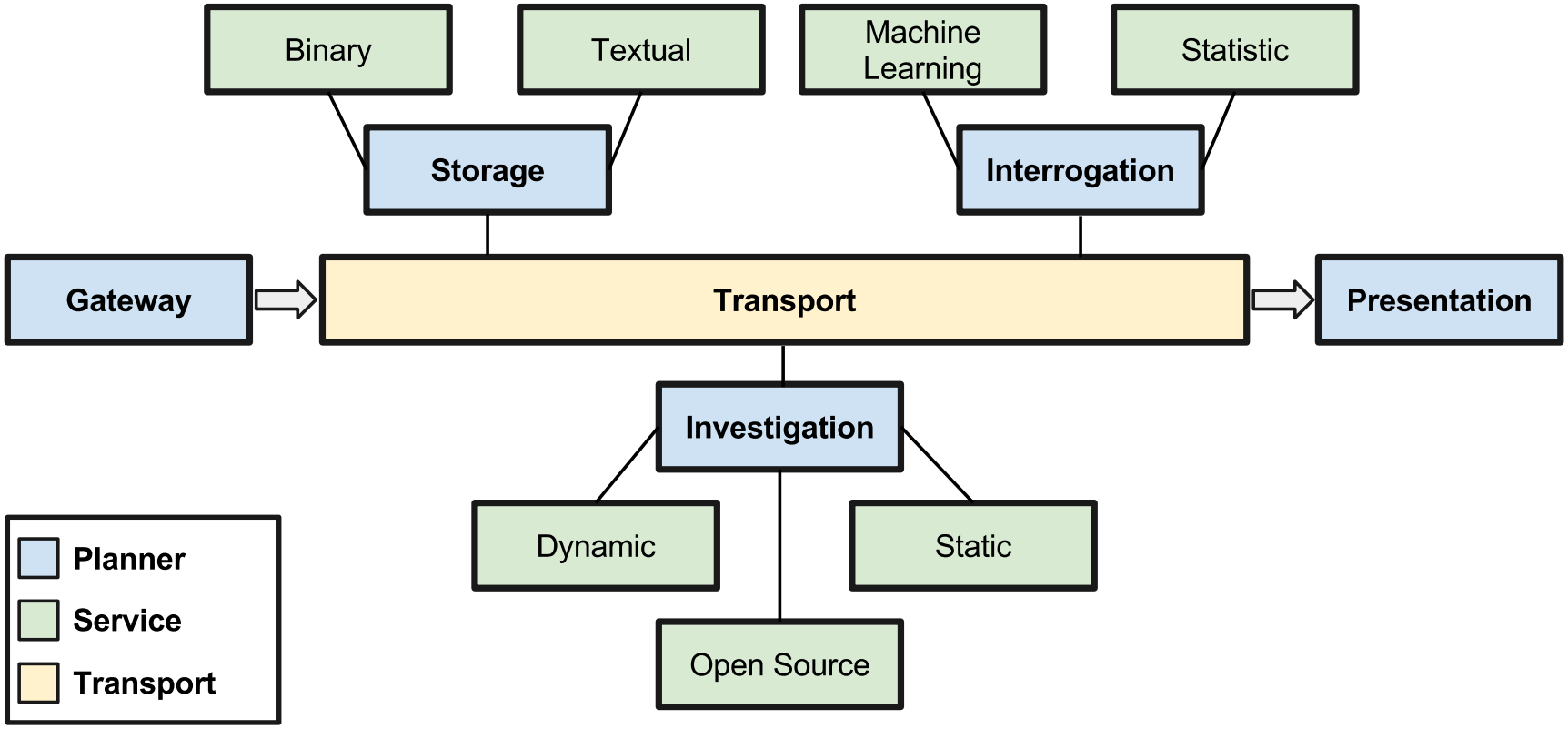
1. **Malware collection** - retrieve and store a large-scale sample set

Problem statement

Malware detection and triage process

1. **Malware collection** - retrieve and store a large-scale sample set

2. **Data collection** - static and dynamic analysis tools
    a. **Static** analysis - code features, PE header, easy to obfuscate
    b. **Dynamic** analysis - trace malware execution (kernel API calls)



https://holmesprocessing.github.io/

Problem statement

Malware detection and triage process

1. **Malware collection** - retrieve and store a large-scale sample set

2. **Data collection** - static and dynamic analysis tools
    a. **Static** analysis - code features, PE header, easy to obfuscate
    b. **Dynamic** analysis - trace malware execution (kernel API calls)

3. **Data analytics** - analyze the gathered data
    - Usually signature- or heuristics-based
    - Very time consuming if done manually
    - Machine Learning - one approach for effective automation

Research Goal

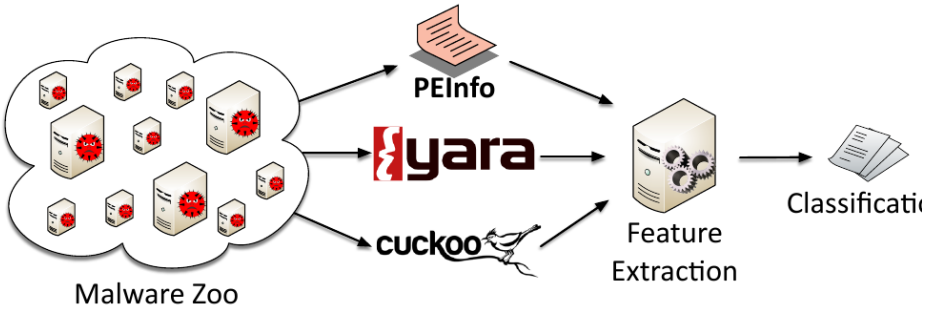
- Investigate and improve automatic **feature extraction** approaches
  - Key step in detection/classification

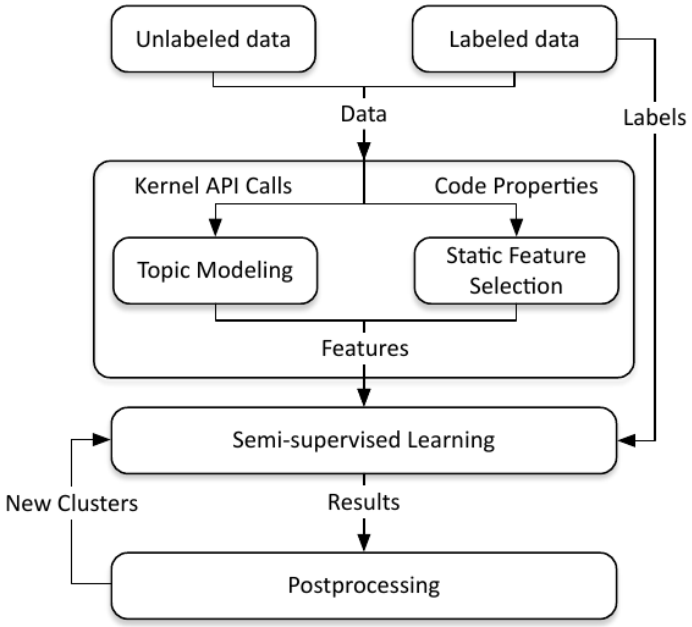
- Make the malware detection and decisions **semantics-aware**, **explainable**
  - Discover semantics from behavioral traces
  - Model interpretability

- Make our classifiers **adaptive** and **robust**
  - Maintain the model during high influx of samples
  - Robust to outliers (open world)

Integrating Topic Modeling

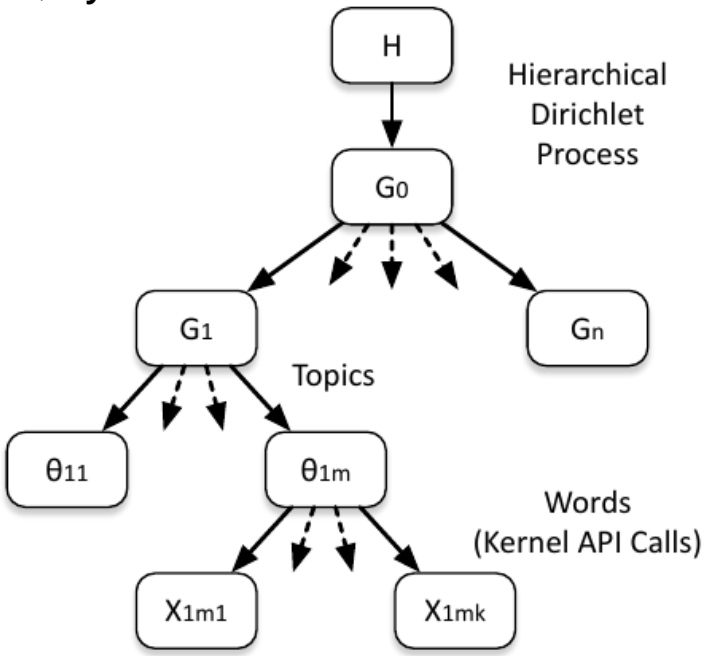
- Topic Modeling + semi-supervised learning





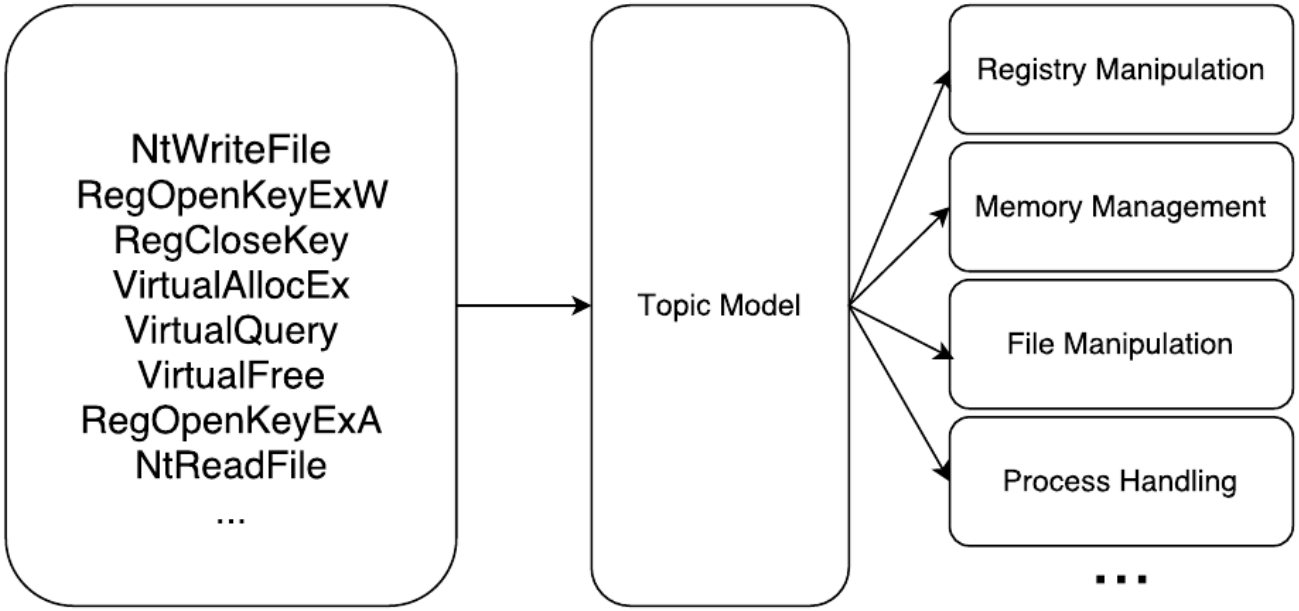
Integrating Topic Modeling

- Hierarchical Dirichlet Process
  - Model syscall traces as documents, syscalls as words
  - Topics change with dataset



Integrating Topic Modeling

- 1 Topic model per class (malware or benign)



Integrating Topic Modeling

- Topics and words example

Registry manipulation	Memory management	File manipulation	Process Handling
NtWriteFile	VirtualAllocEx	NtReadFile	OpenProcess
RegOpenKeyExW	VirtualQueryEx	NtWriteFile	ReadProcessMemory
RegCloseKey	VirtualQuery	NtDelayExecution	WriteProcessMemory
RegEnumValueW	VirtualFreeEx	LdrGetProcedureAddress	CloseHandle
RegQueryValueExW	VirtualFree	NtSetInformationFile	LocalAlloc
LdrGetProcedureAddress	LdrGetProcedureAddress	NtCreateFile	LocalFree
RegOpenKeyExA		NtQueryDirectoryFile	

Integrating Topic Modeling :: Results

- Results

Family	LDA for a different number of topics						HDP
	1(%)	5(%)	10(%)	20(%)	40(%)	80(%)	
Amonetize	0.0	0.0	10.0	100.0	100.0	100.0	100.0
Somoto	0.0	0.0	0.1	30.3	20.4	30.0	99.8
Kryptik	0.0	18.0	30.0	70.0	60.0	30.5	91.5
Multiplug	0.0	57.4	80.0	30.0	40.0	69.4	80.0
Bladabindi	0.0	1.7	5.7	4.0	7.0	10.3	93.0
Eldorado	0.0	0.0	0.0	0.0	0.0	0.0	54.4
Morstar	0.0	0.0	0.0	0.0	0.0	0.0	100.0
Preloader	0.0	0.0	7.5	71.0	50.0	60.0	100.0
SProtector	100.0	100.0	100.0	100.0	100.0	100.0	100.0
SoftPulse	0.0	4.2	4.1	6.7	5.0	6.9	86.2

Family	Supervised(%)			Semi-supervised(%)					
				1^{st} Experiment			2^{nd} Experiment		
	ACC	PR	RC	ACC	PR	RC	ACC	PR	RC
Amonetize	100.0	100.0	100.0	100.0	88.3	100.0	100.0	98.4	100.0
Somoto	100.0	100.0	100.0	93.6	72.2	93.3	100.0	96.8	100.0
Kryptik	100.0	100.0	100.0	100.0	86.4	100.0	100.0	100.0	100.0
Multiplug	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Bladabindi	99.4	98.1	96.0	83.5	95.4	82.9	96.6	95.8	96.6
Eldorado	75.6	26.3	86.0	31.4	98.1	31.6	86.8	98.9	86.8
Morstar	100.0	98.5	100.0	99.2	97.5	99.2	100.0	99.0	100.0
Preloader	100.0	100.0	100.0	57.1	100.0	55.4	100.0	100.0	100.0
SProtector	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
SoftPulse	64.4	75.4	87.0	49.5	51.1	50.8	88.9	86.5	88.9
Average	93.9	89.8	96.9	81.4	88.9	81.3	**97.2**	**97.5**	**97.2**
Rieck et al. [27]	**88.0**	-	-	-	-	-	-	-	-
Dahl et al. [9]	**90.5**	-	-	-	-	-	-	-	-

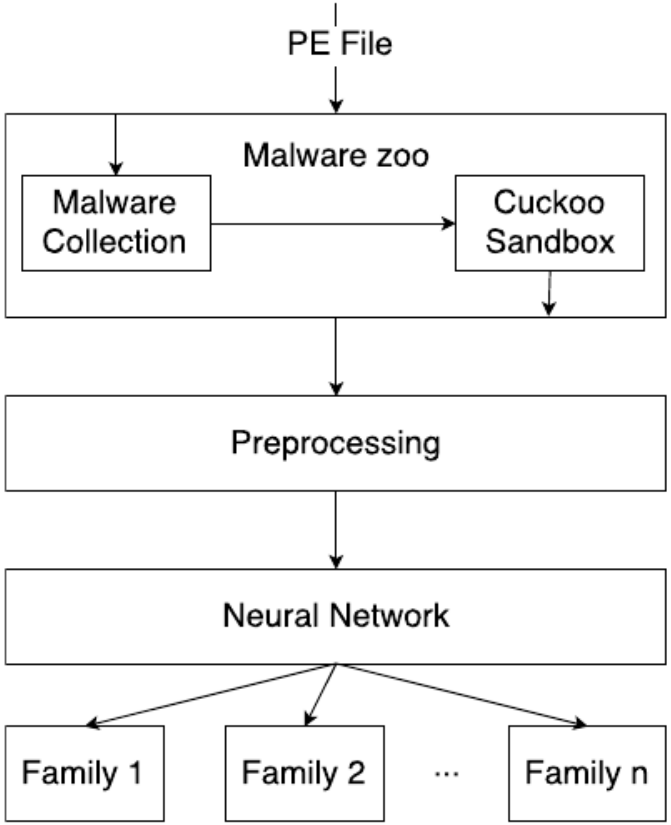
Neural Network approach

- Many previous approaches: n-grams, SVM with string kernels, hidden markov models, topic modeling...
- However, application of neural networks underexplored
  - Static:
    - Saxe et al. - deep feedforward networks (FFNN) for malware code (MALWARE 2015)
  - Dynamic:
    - Dahl et al. (ICASSP 2013) - random projection + FFNN
    - Pascanu et al. (ICASSP 2015) - RNN, malware language modeling

Our Goal

- We investigate possibilities of leveraging **deep learning** principles and methods for the malware system call sequences classification
- Motivated by applications of convolutional networks for classifying short texts (Yoon Kim, 2014)
- We combine **convolutional** and **recurrent** approaches to feature extraction
- We investigate neural network **feature extraction** and try to interpret results

System Overview

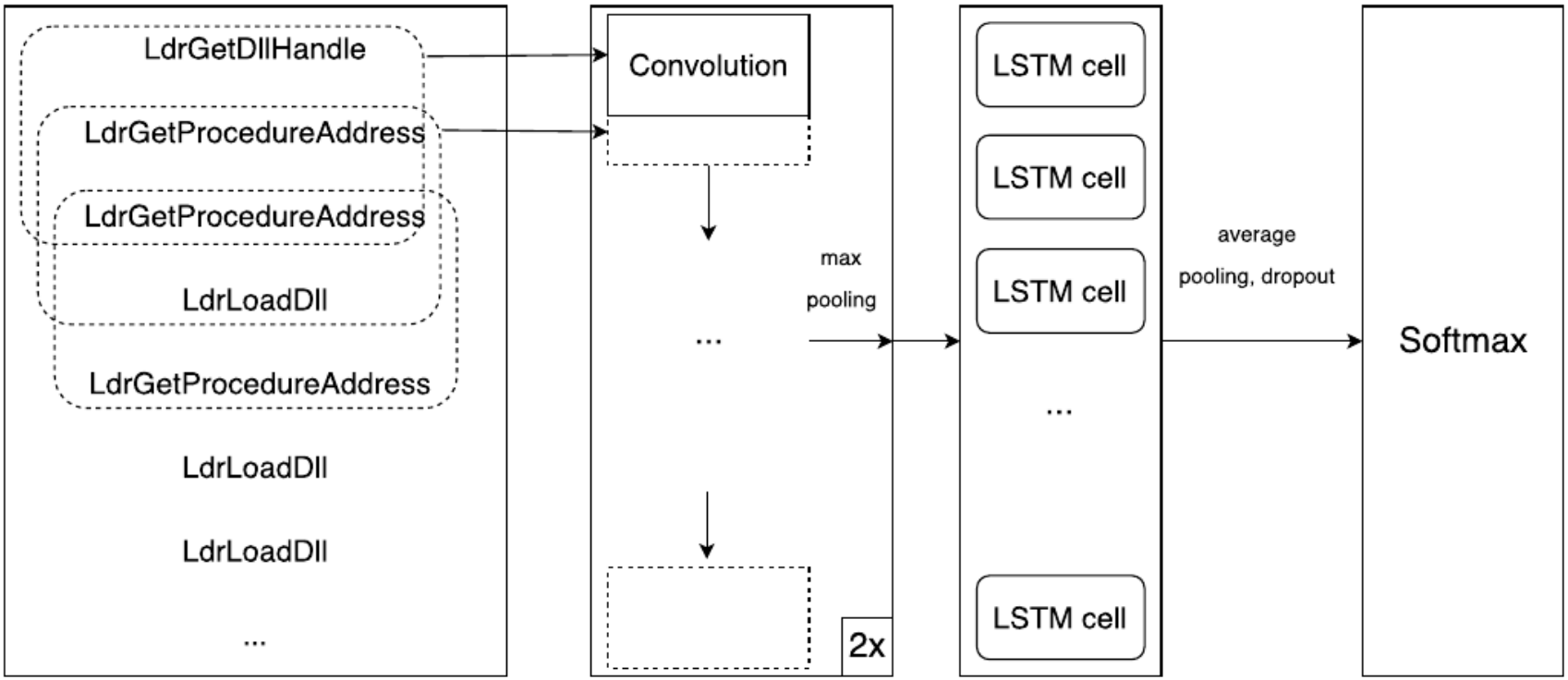


Data Collection and Preprocessing

- **Malware** sources: VirusShare, Maltrieve, private collections (diversity)
- Cuckoo Sandbox for **malware execution traces**
- Virustotal API for **ground truth** labels
  - Create binary vectors from AV signatures
  - Label clustering to retrieve malware families
  - Extract 10 most populous families for ground truth, covers 95% of the dataset
- Remove long subsequences with repeating API calls - malware stuck
- **One-hot encoding** for API calls (dictionary of 60 calls)
- Prune the API call dictionary

Neural network architecture

- Nx60 filter matrix, best results for N=3,4,5



Evaluation

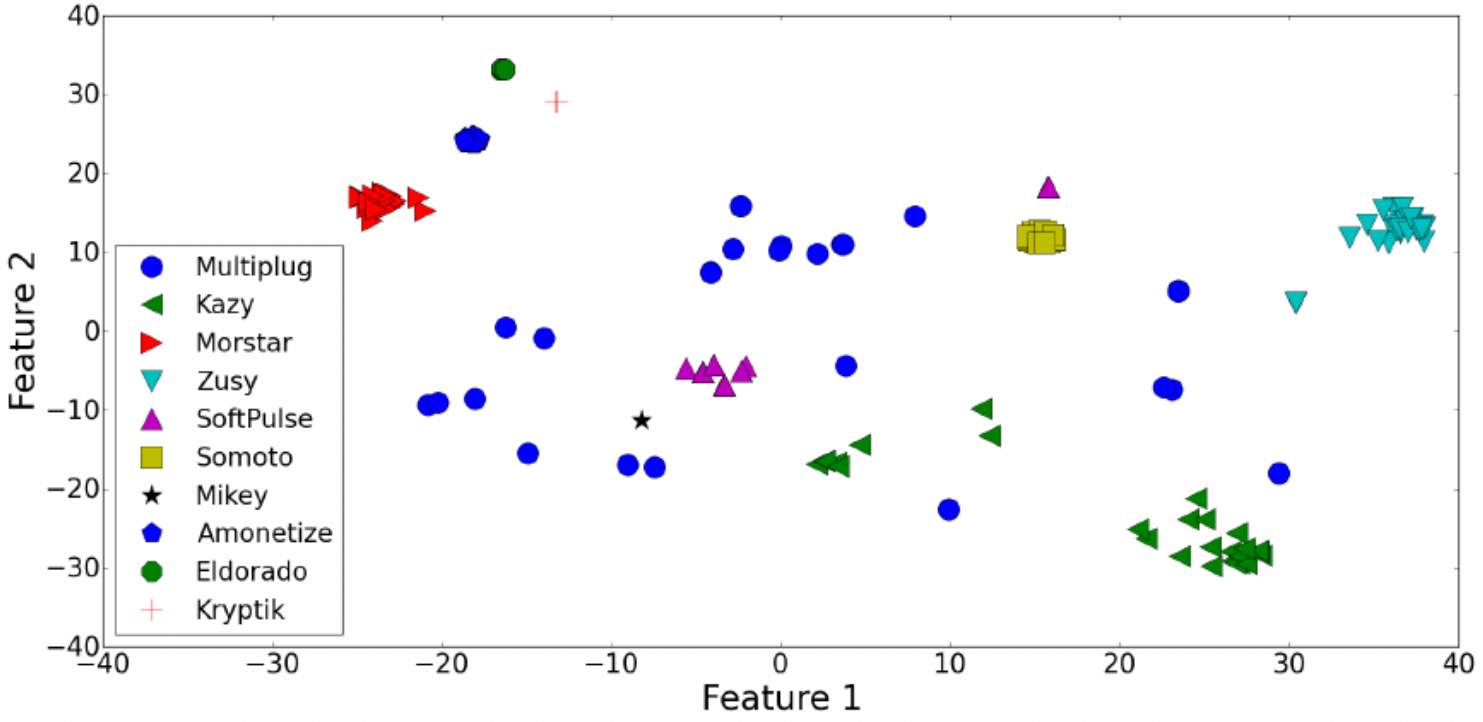
Family	Deep Neural Network			Hidden Markov Model			Support Vector Machine		
	ACC	PR	RC	ACC	PR	RC	ACC	PR	RC
Multiplug	98.9	99.8	99.0	91.5	74.5	91.5	99.3	99.9	99.3
Kazy	100.0	99.9	100.0	73.1	95.1	73.1	96.6	93.1	96.6
Morstar	100.0	99.9	100.0	80.0	63.7	80.0	82.3	91.0	82.3
Zusy	100.0	57.5	100.0	65.4	45.1	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

Evaluation

- Significant **improvements** using our architecture w.r.t. baseline methods
    - HMM (over 30% on precision, over 10% on recall)
    - SVM (around 2% on precision, 1% on recall)
- Approach also better than using only FFNN or CNN
- Final results: PR:85.6, RC: 89.4
- Performance varies in breakdown by families

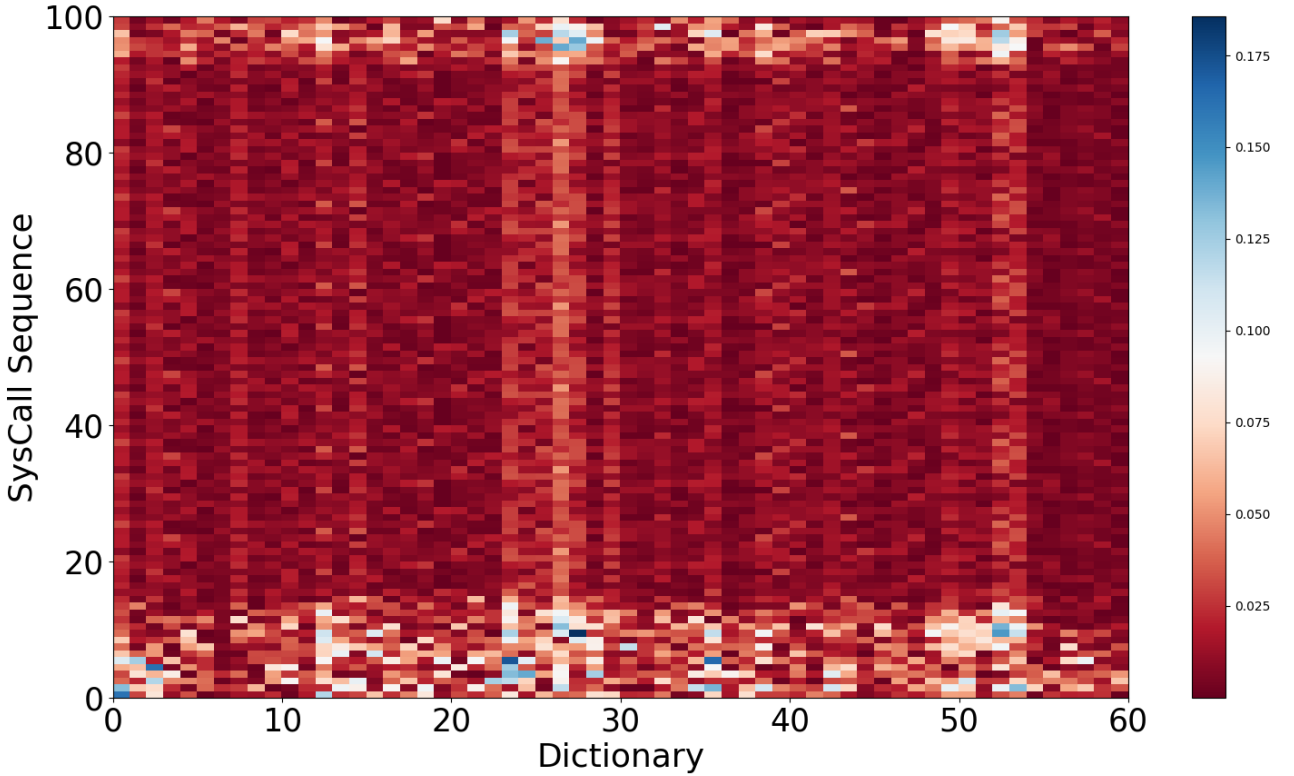
Evaluation

- Malware family separation



Evaluation

- Prediction Heatmap, constructed using gradients w.r.t inputs*

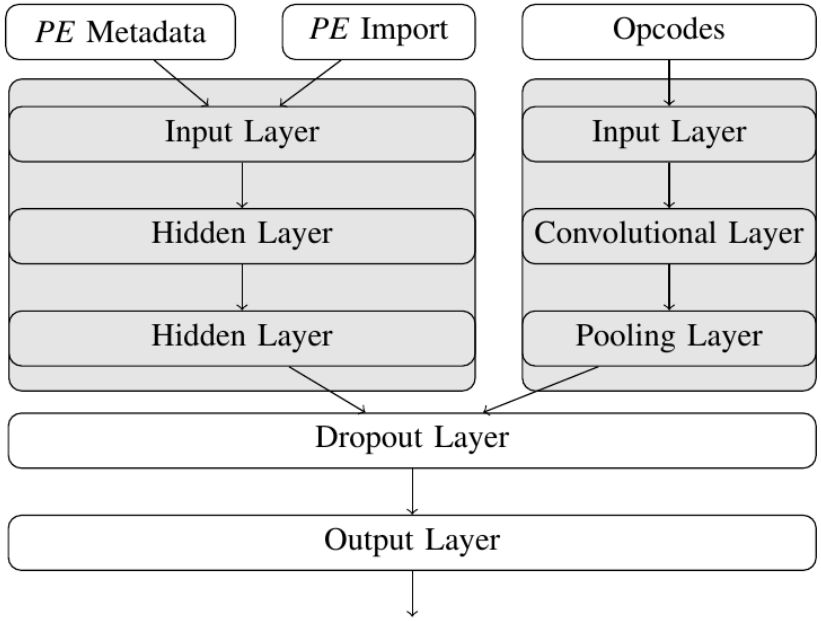


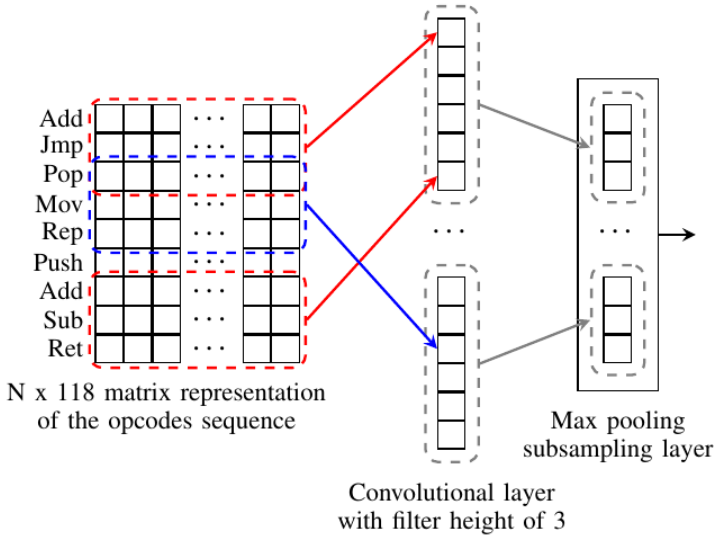
$$w(e) = \frac{\partial(S_c)}{\partial e} \mid_e$$

$$S(e) = |w(e)|$$

*Based on Li, Jiwei, et al. "Visualizing and understanding neural models in NLP

Neural Network approach - objdump



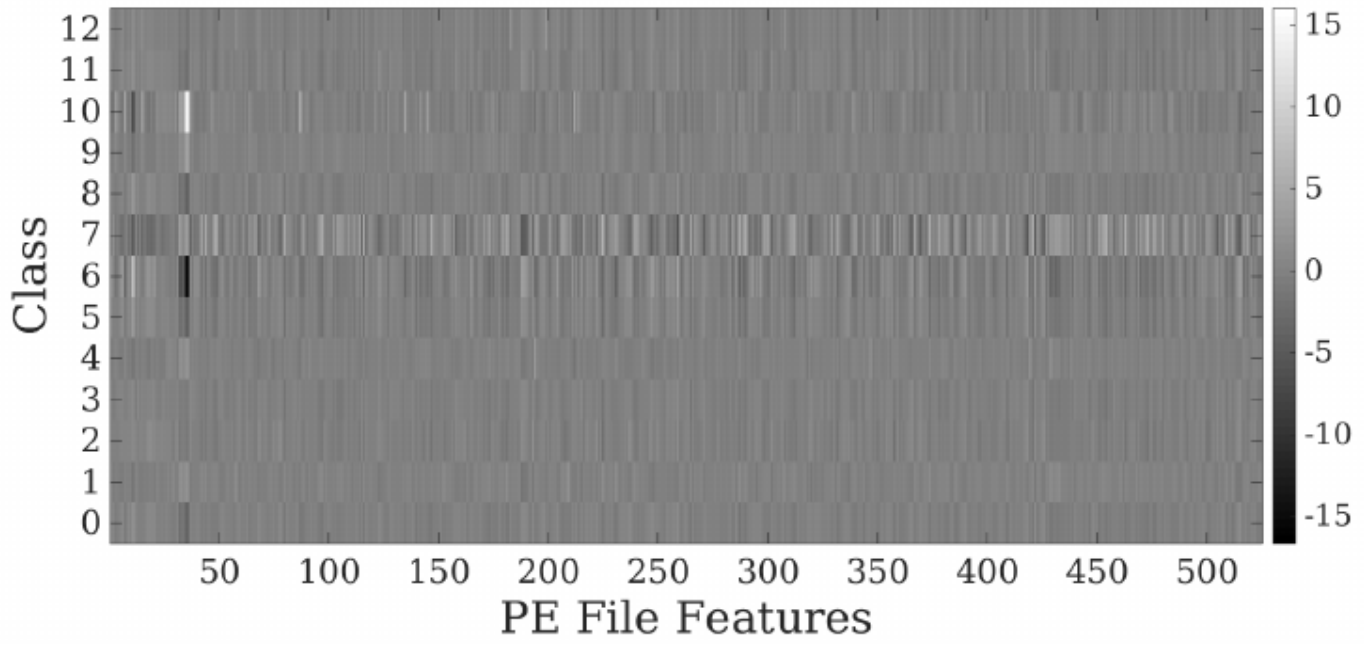


Neural Network approach – objdump results

- 93% on precision, 92% on recall
- Immunity to small perturbations in code: instruction shuffling, adding nop instructions
- Better performance than simple FFNN network
- Combining PE header and objdump features works well

Neural Network approach - objdump

- Saliency map – which feature contributes to classification to a certain class



Future work

- Combine **neural** and **topic model** approaches in a computationally-efficient framework
  - Neural network – nonlinear feature extraction powerful
  - Topic model – interpretability, convenient for analysts

- Investigate **robustness** of used methods in an adversarial environment by executing:
  - Exploratory attacks
  - Causative attacks

- Investigate other types of data: rich header, gadgets, control-flow graphs