SpacSec 2026 · International Conference on Cyber Warfare, Security & Space Computing



Behavior-based Malware Detection in CAPEv2 Sandbox Using Deep Learning

Diana Levshun (Sr. Res., Cand. Sci. Tech.), Dmitry Levshun (Sr. Res., Cand. Sci. Tech., PhD)

St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS)

Saint Petersburg · October 26–28, 2026

Contents



1

Introduction

Relevance and our contribution



2

Proposed approach

Pipeline and report representation



3

Experiments

Evaluation of the quality



4

Conclusion

Key findings and future work

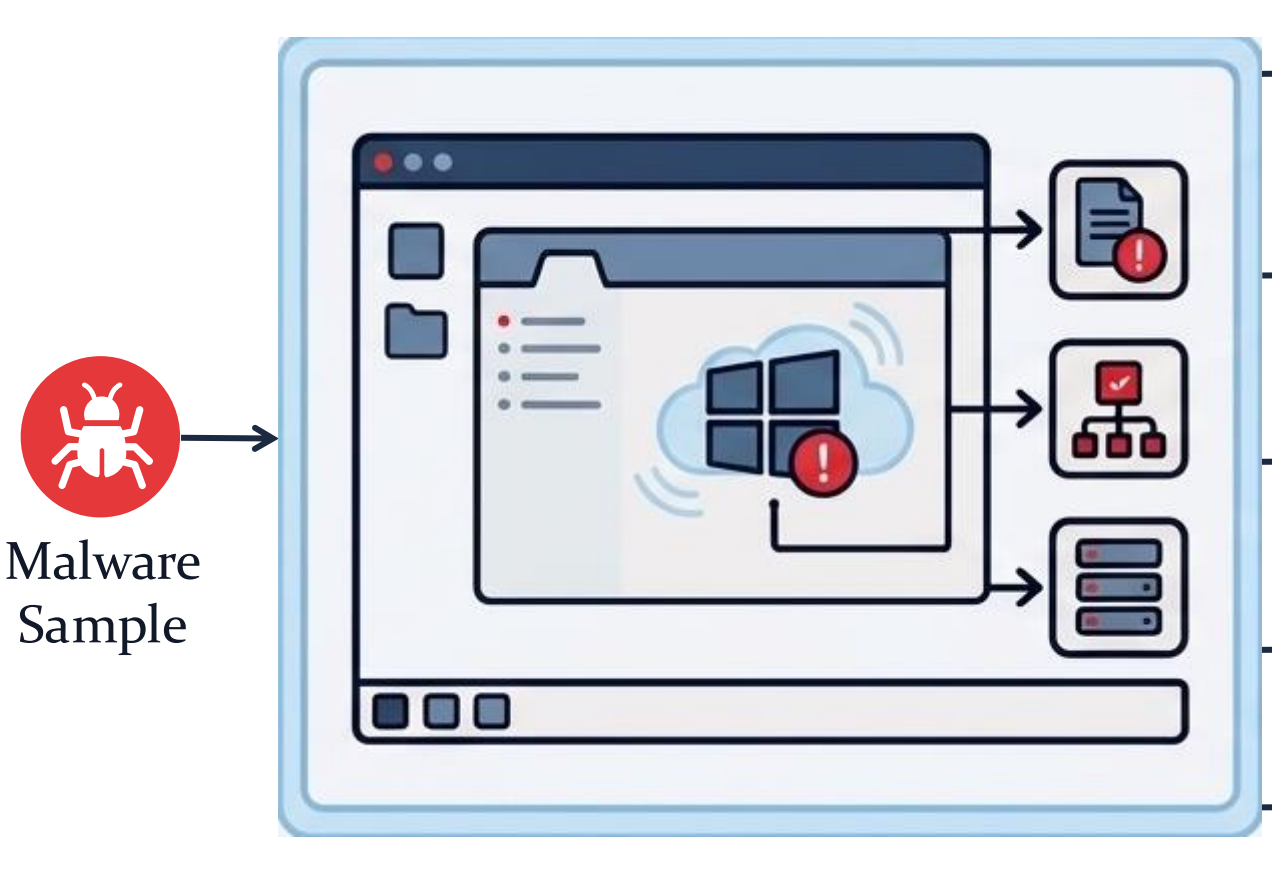


5

Contacts

Authors and acknowledgement

Dynamic sandbox analysis



File System Activity
New files, modifications, deletions

API Calls
Interaction with os libraries

Network Connections
Domains, IPs, data transfer

Executed Commands
Shell instructions, processes

and others...
(registry changes, process tree)

Why behavior-based detection?

- **Polymorphism, packing, obfuscation, and execution staging** reduce the effectiveness of signature testing and static analysis.
- **In open-source software**, malicious code hides in legitimate packages, loading components dynamically only at runtime.
- **Dynamic sandboxes (like CAPEv2)** produce detailed execution reports covering file systems, registry, APIs, network, and commands.
- **These reports are semi-structured and heterogeneous**, mixing natural-language fragments with identifiers, paths, and domains.
- **Research gap**: how the choice of a pre-trained language model affects behavior-based malware detection on sandbox reports remains understudied.

CAPEv2 report · Behavior → Summary

```
"read_files": [
  "C:\\Windows\\Fonts\\staticcache.dat",
  ... ],
"write_files": [
  "C:\\ProgramData\\Mozilla\\ysokqrk.exe"
 ],
"keys": [
  "HKEY_LOCAL_MACHINE\\Software\\...",
  ... ],
"resolved_apis": [
  "kernelbase.dll.LoadLibraryExA",
  "kernel32.dll.GetProcAddress", ... ],
"executed_commands": [...],
"mutexes": [...]
```

Rich behavioral evidence — but hard to process automatically

Our contributions

Six-component representation
of dynamic analysis reports capturing the key behavioral aspects of program execution.

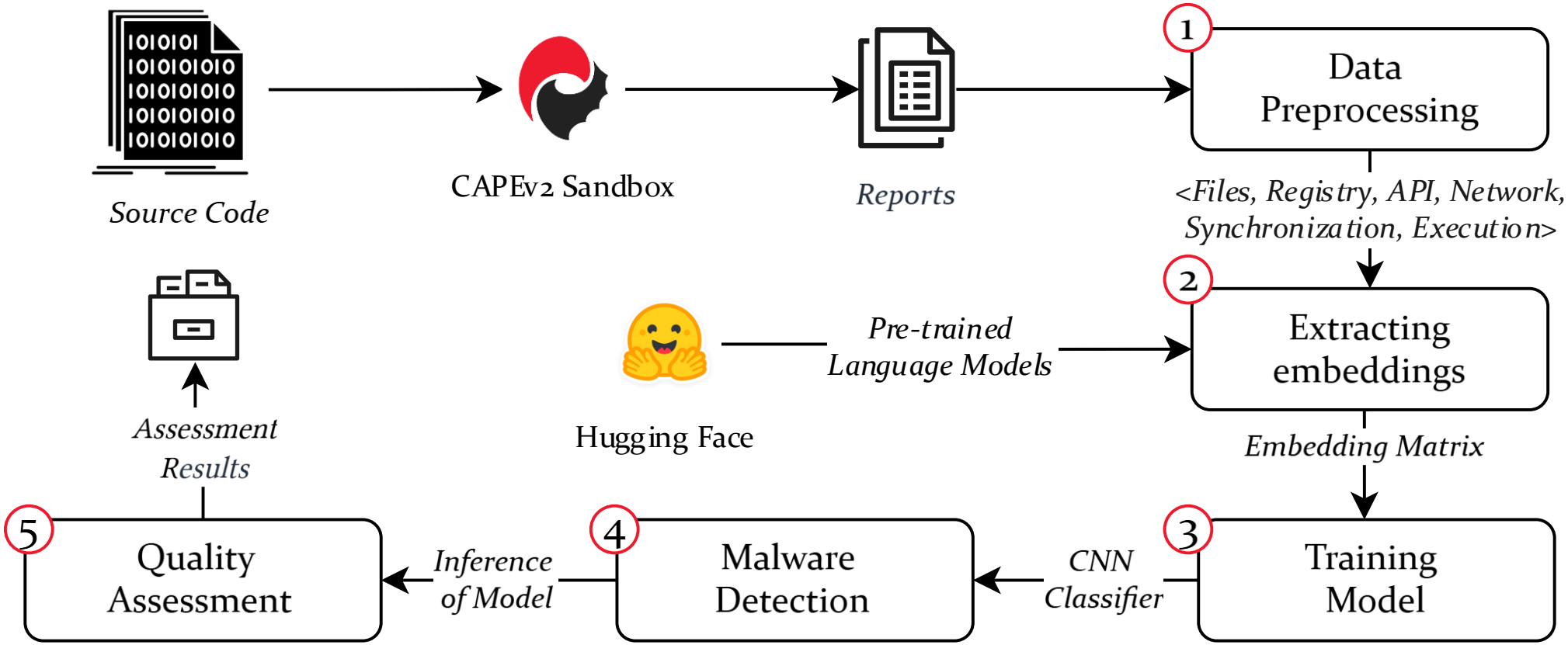
Cross-Category Relation (CCR) block
placed before the convolutional layers to explicitly model interdependencies between components.

Multi-scale convolutional analysis
with three parallel kernels extracting local and cross-component behavioral patterns.

Comparative evaluation of language models
MicroBERT, TinyBERT and CodeBERTa as embedding models for malware classification.

Evaluation on public datasets
showing high quality of multi-class malware classification on MGDAR and Avast-CTU.

Five-stage pipeline



From sandbox report to embedding matrix

Files
`files, read_files, write_files, delete_files`

Registry
`keys, read_keys, write_keys`

API
`resolved_apis`

Network
`unique IP addresses and domains`

Synchronization
`mutexes`

Execution
`executed_commands`

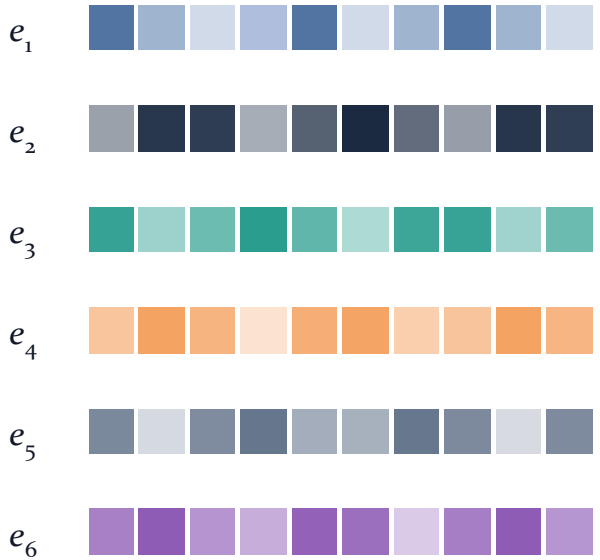
Pre-trained LM

$e_k = \Omega(R_k)$

MicroBERT d = 128
TinyBERT d = 312
CodeBERTa d = 768

[CLS] embedding
L = 256 tokens

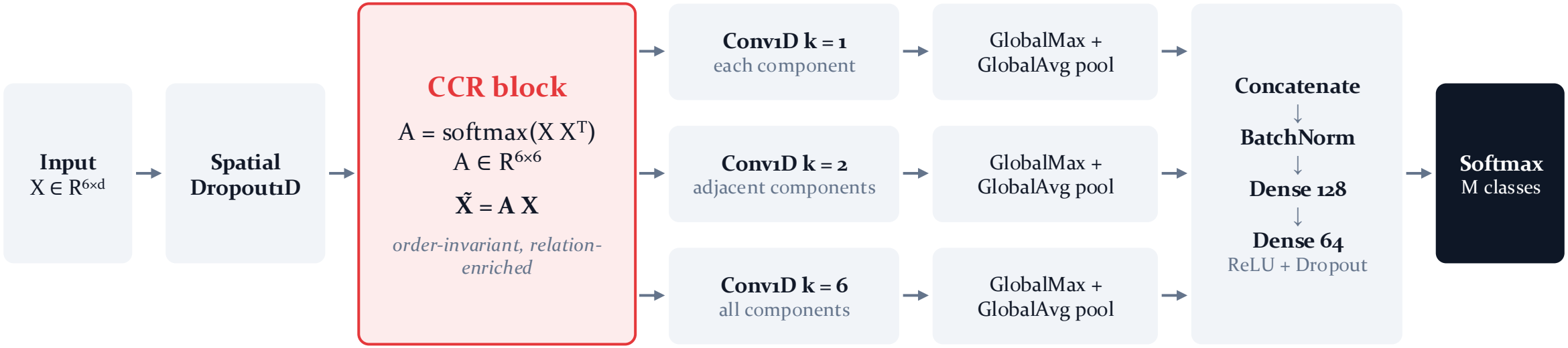
Embedding matrix



$$X = [e_1, \ldots, e_6] \in R^{6 \times d}$$

Each row is a semantic profile of one kind of program activity — the matrix is treated as a low-resolution "image".

CCR-CNN classifier



Why CCR?

The affinity matrix A captures pairwise alignment of behavioral domains (e.g., Network ↔ Registry). Each row aggregates all others, removing the CNN's sensitivity to component order.

Multi-scale kernels

Kernel width equals d, so every embedding is an indivisible unit; heights 1, 2 and 6 cover single, adjacent and all components (128 filters each).

Training

Batch size 32, up to 100 epochs with early stopping (patience 20). Prototype in Python 3.11 with Keras 3.14 / TensorFlow 2.21.

Datasets and experimental setup

26,200

reports

MGDAR — Malware & Goodware Dynamic Analysis Reports

Goodware 8,600 (33%, Chocolatey) and malware 17,675 (67%, VirusTotal); 3 families: Virlock, Shipup, Vobfus

48,976

malicious reports

Avast-CTU Public CAPEv2

10 families: Qakbot, Swisyn, Emotet, Lokibot, Zeus, Trickbot, Ursnif, HarHar, njRAT, Adload; goodware added from MGDAR

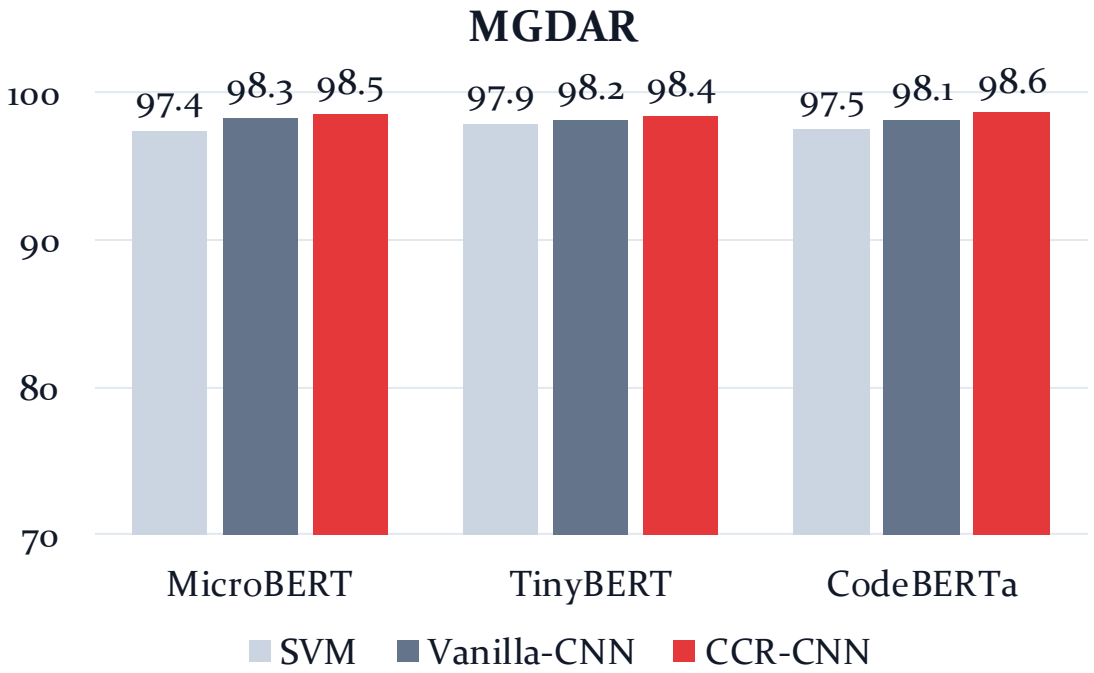
Embedding models

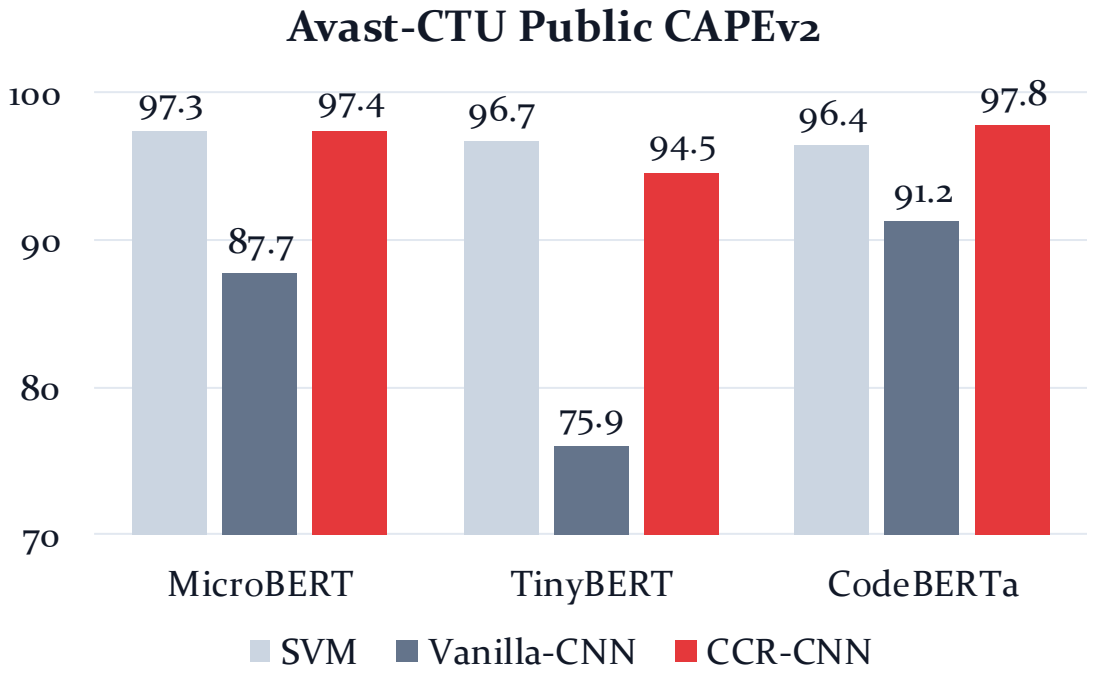
Model	Architecture	Params	d
MicroBERT	2-layer BERT	4.42M	128
TinyBERT	4-layer BERT	4.5M	312
CodeBERTa	6-layer RoBERTa	84M	768

Protocol

- Split **60 : 20 : 20**; L = 256 tokens; subword tokenization (WordPiece / Byte Pair Encoding)
- Baselines: **Vanilla-CNN** (2 × Conv1D) and **linear SVM** (C = 1.0)
- **Bootstrap**: 100 iterations on a fixed test set of 2,500 samples
- **Friedman test**, $\alpha = 0.01$; feature ablation per component

Overall detection quality (F1, %)



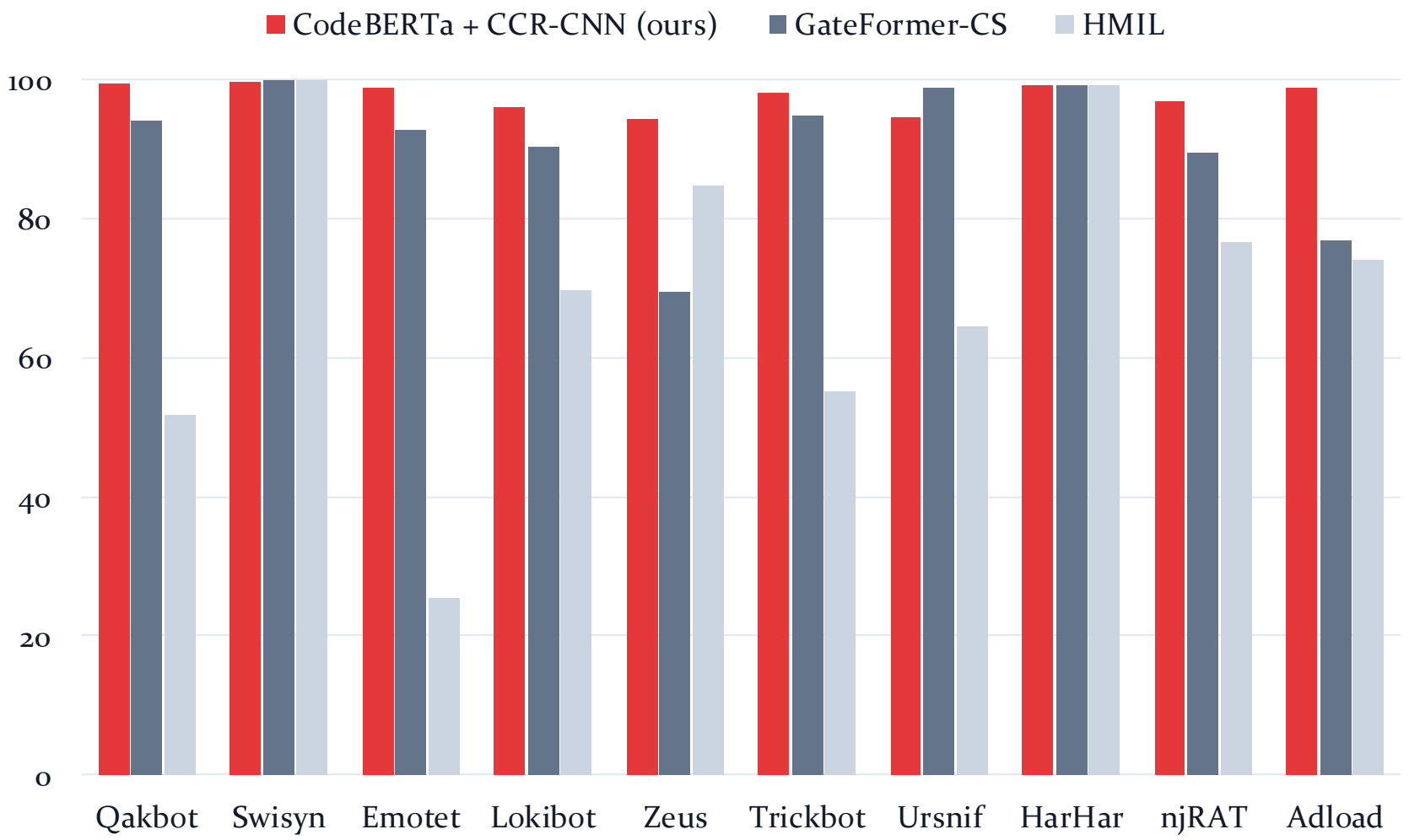


98.6% **F1 on MGDAR**
CodeBERTa + CCR-CNN

97.8% **F1 on Avast-CTU**
CodeBERTa + CCR-CNN

p < 0.01 **Friedman test**
Q = 431.7 (MGDAR)
Q = 762.5 (Avast-CTU)

Per-family F1 vs. state of the art (Avast-CTU)



7 / 10

families with the best F1 among the three models

+24.8 pp

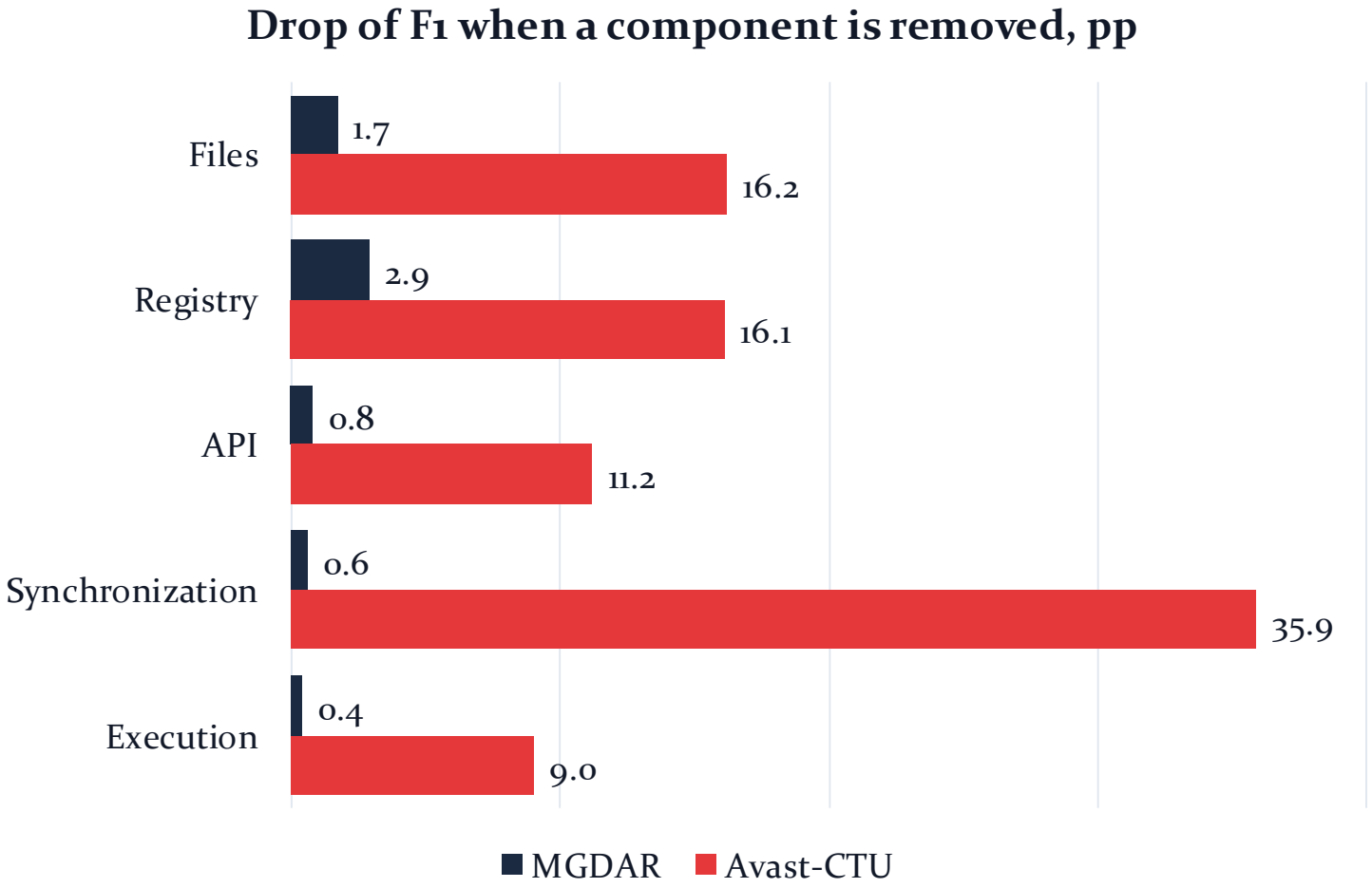
on Zeus vs. GateFormer-CS (94.3% vs 69.5%)

+22.0 pp

on Adload vs. GateFormer-CS (98.9% vs 76.9%)

GateFormer-CS keeps a small edge on Swisyn, Ursnif and HarHar. External results exist for malware families only (no goodware class).

Feature ablation: which components matter?



Registry → MGDAR
ΔF1 = 2.9 pp — the largest drop; reliance on system persistence artifacts.

Synchronization → Avast-CTU
ΔF1 = 35.9 pp — sophisticated malware relies on unique multi-instance locking (mutex) signatures.

Network
No drop on either dataset.

Computational cost of training

Language model	Classifier	MGDAR			Avast-CTU		
		Time, s	Per-epoch, s	RAM, MB	Time, s	Per-epoch, s	RAM, MB
MicroBERT	SVM	40.37	–	0.72	54.52	–	1.55
	Vanilla-CNN	147.9	1.78	91.5	112.21	5.61	156.0
	CCR-CNN	351.19	4.39	93.21	660.7	6.61	176.0
TinyBERT	SVM	134.03	–	0.84	109.53	–	3.92
	Vanilla-CNN	204.33	2.65	154.32	133.34	6.67	157.37
	CCR-CNN	579.67	7.94	169.32	949.67	10.44	197.0
CodeBERTa	SVM	174.58	–	1.5	73.22	–	4.41
	Vanilla-CNN	273.94	5.17	349.29	162.37	8.12	179.88
	CCR-CNN	1,795.87	22.45	214.4	679.14	8.18	200.0

SVM is the lightest
Fast convergence and < 5 MB of RAM, but below CCR-CNN in quality.

CCR-CNN costs more
Extra runtime comes from the matrix multiplications inside the CCR block.

MicroBERT + CCR-CNN
Best quality / cost trade-off: 351 s vs 1,796 s for CodeBERTa on MGDAR.

CONCLUSION

Key findings

- We proposed a **behavior-based malware detection** method for CAPEv2 reports: **six behavioral components → language-model embeddings → CCR-CNN**.
- **CCR-CNN** gives the best results on both datasets; the differences between classifiers are **statistically significant** (Friedman test, $p < 0.01$).
- **CodeBERTa + CCR-CNN** is the best: F1 = **98.6%** on MGDAR and **97.8%** on Avast-CTU — code-oriented embeddings match the syntax of sandbox reports.
- **MicroBERT + CCR-CNN** is a compact and efficient alternative; **TinyBERT** consistently underperforms.
- **Registry** (MGDAR) and **Synchronization** (Avast-CTU) are the most informative behavioral components.



Future work

- Extend the set of **language models** and study the impact of their architectural features
- Compare embeddings with **full fine-tuning** of the language model on sandbox reports
- Move to **hybrid classifiers** combining convolutional, recurrent and attention mechanisms

CONTACTS

Thank you for your attention!

Laboratory of Computer Security Problems, St. Petersburg Federal Research Center of the Russian Academy of Sciences

39, 14th Liniya, St. Petersburg, 199178, Russia

++7 (812) 508-33-11

+7 (812) 328-44-50 (fax)

https://spcras.ru/

Authors

Diana Levshun
diana.levshun@comsec.spb.ru

Dmitry Levshun
levshun@comsec.spb.ru

Acknowledgement

The study was supported by the grant of the Russian Science Foundation No. 24-71-10095, https://rscf.ru/en/project/24-71-10095/