# Convolutional Neural Networks for Classification of Malware Assembly Code

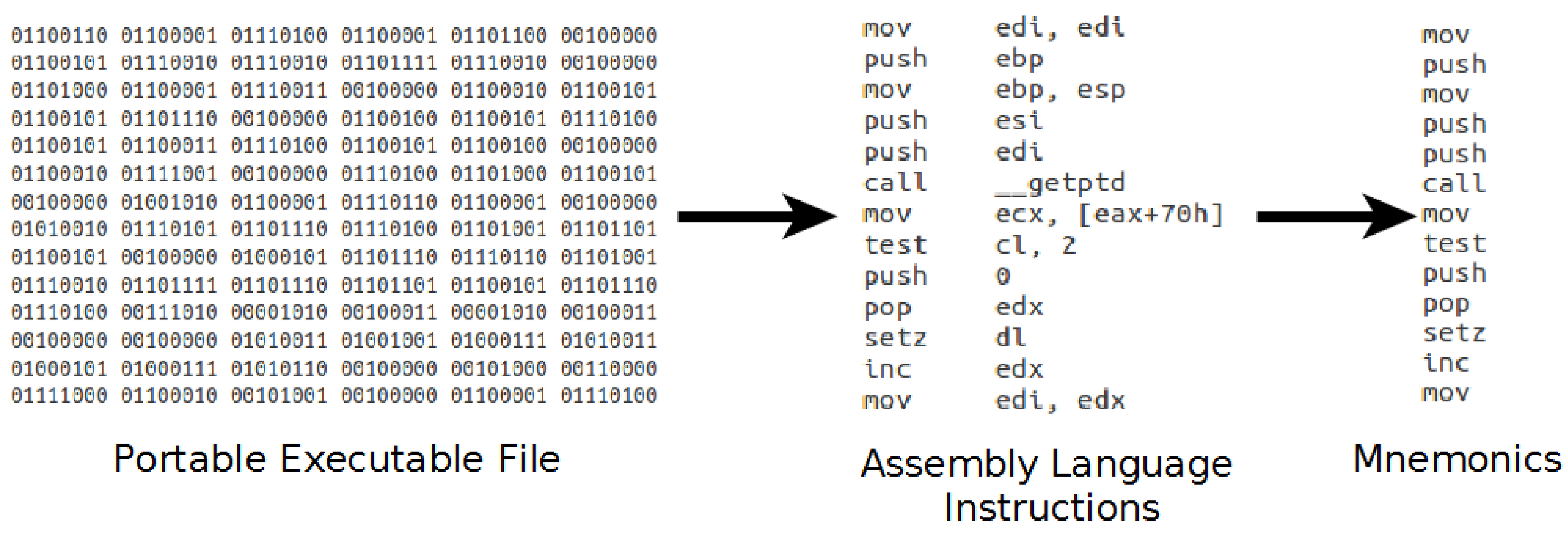
Daniel Gibert, Javier Béjar, Carles Mateu, Jordi Planes,
Daniel Solis, Ramon Vicens

Universitat Politècnica de Catalunya BarcelonaTech
Universitat de Lleida, Escola Politècnica Superior
Blueliv.

## Objectives

- Build a static classifier without relying on hand-crafted features defined by experts.
- Group malware into families based on their assembly language source code.
- Extract N-Gram like signatures with convolutional neural networks from malware's machine instructions.

## Data Transformation




## CNN Layers Description

- **Input**

  An assembly program is represented as a concatenation of mnemonics

  $$x_{1:n} = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

  where $n$ is the length of the program and $x_i \in \mathbb{R}^k$ corresponds to the i-th mnemonic in the program.

- **Embedding**

  Every mnemonic is represented as a low-dimensional vector of real values (word embedding).

- **Convolution**

  A convolution operation involves a filter $w \in \mathbb{R}^{hk}$ where $h$ is the number of mnemonics to which is applied and $k$ is the size of the word embedding. In particular, filters are applied to sequences containing from 2 to 7 mnemonics.

  A feature $c_i$ is generated from a window of mnemonics $x_{i:i+h-1}$ (it comprises all mnemonics between position $i$ and $i + h - 1$) and is defined as

  $c_i = f(w \cdot x_{i:i+h-1} + b)$,

  where $f$ is a rectifier linear unit (ReLU) function and $b$ the bias term.

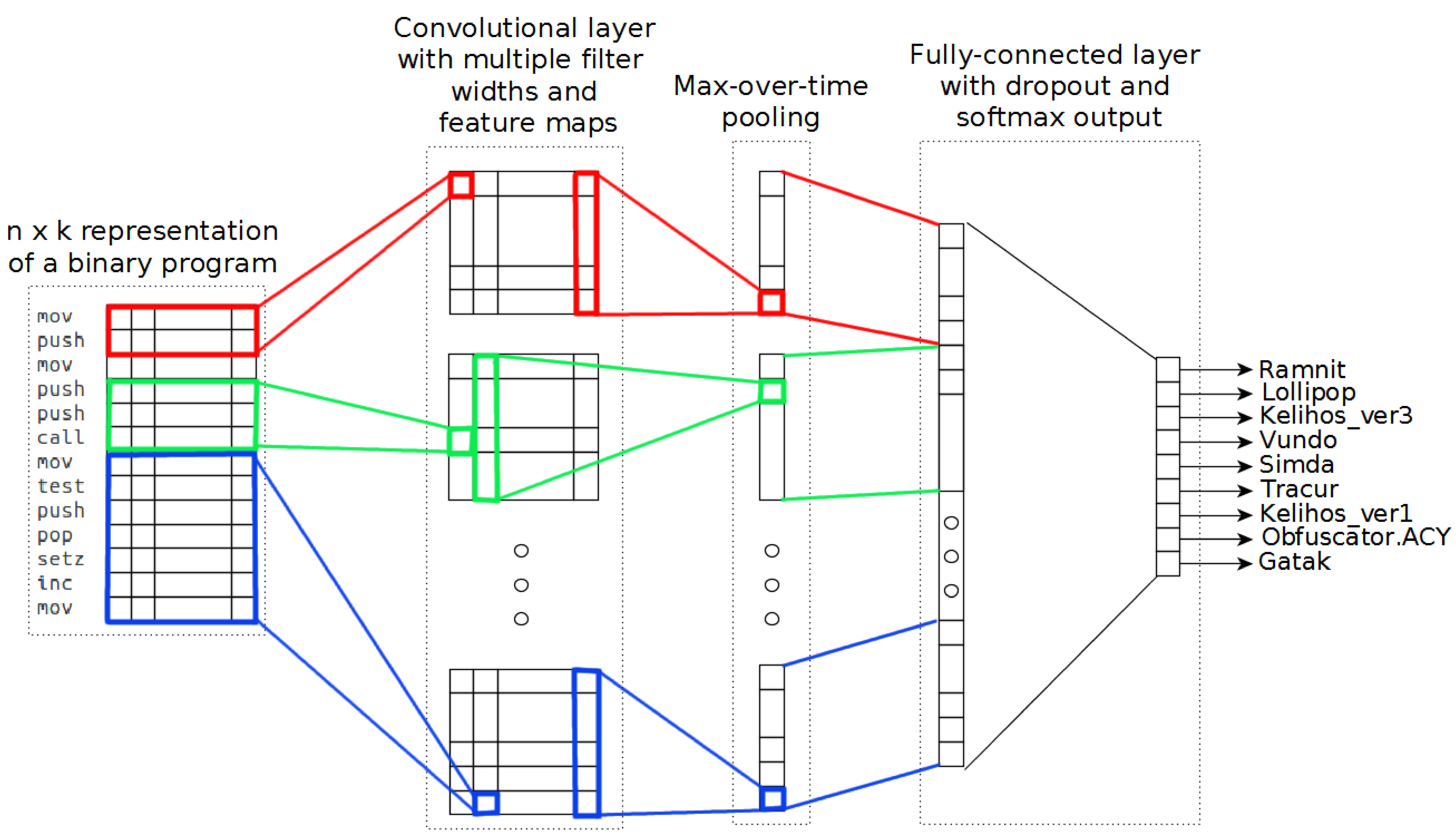
- **Max-Pooling**

  The maximum value $\hat{c} = \max\{c\}$ is taken as the feature corresponding to the filter by applying the max pooling operator over the feature map.

- **Softmax layer**

  The extracted features are passed to a fully-connected softmax layer whose output is the probability distribution over families.

## Architecture




## N-Gram Comparison

- An N-Gram is a contiguous sequence of N items from a given sequence of text.
- N-Gram like signatures have proved useful in classifying malware.
- The main limitation of standard N-Gram based methods is the exponential increase in the number of unique n-grams as n is increased.

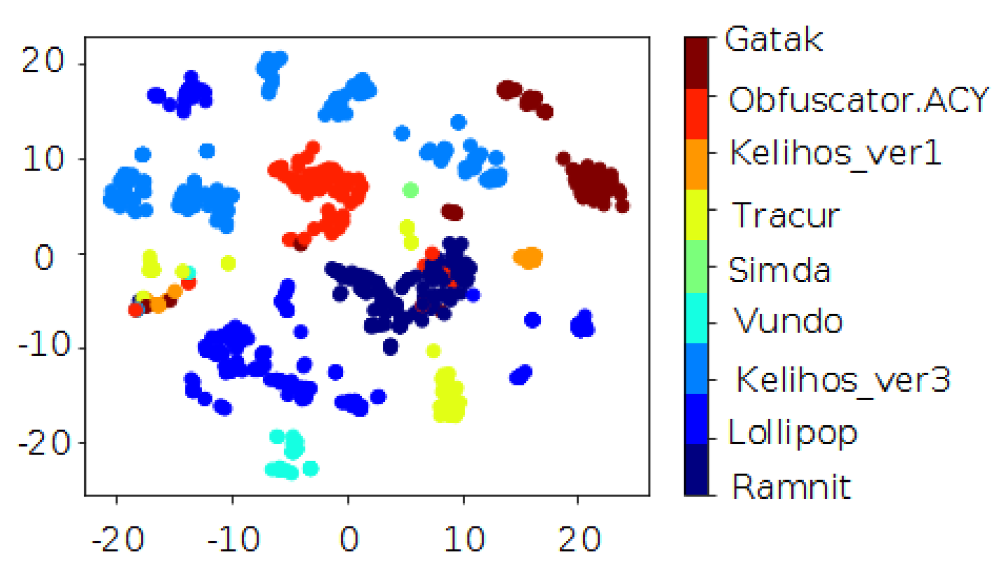
| Method | #features | RAM Usage | Extraction Time (in sec.) | | |
|---|---|---|---|---|---|
| | | (in GB) | Avg | Max | Min |
| 1-Gram | 977 | $1.39 \times 10^{-6}$ | 0.47 | 3.55 | 0.02 |
| 2-Gram | 485809 | $9.72 \times 10^{-4}$ | 0.48 | 3.74 | 0.03 |
| 3-Gram | 338608873 | 0.68 | 23.36 | 31.68 | 9.42 |
| 4-Gram | 236010384481 | 420.02 | - | - | - |
| CNN | 384 | $1.54 \times 10^{-6}$ | 0.49 | 3.57 | 0.04 |

**Table 1:** RAM requirements and feature extraction time considering a subset of 977 mnemonics.

## Conclusion

- End-to-end deep learning framework to automatically extract N-Gram like features and classify malicious software into families based on their assembly language source code.
- Efficient alternative to N-Grams.
- The N-Gram like features learned are highly discriminant and useful for clustering malware into groups.
- Greater predictive power in comparison to opcode-based approaches in the literature.
- Resilient to common obfuscation techniques such as code transposition and function reordering.

## T-SNE Visualization




## Results

| Model | Training accuracy | Test Score |
|---|---|---|
| CNN | 0.9964 | 0.0244 |
| Winner's solution | 0.9986 | 0.0028 |
| NFESF | 1.0000 | 0.0063 |
| SMCMCF (4-Gram+VT) | 0.9980 | 0.0259 |
| SMCMCF (4-Gram) | 0.9930 | 0.0546 |
| STRAND | 0.9859 | 0.0479 |

**Table 2:** Comparison with state-of-the-art methods.

## Acknowledgements

This work has been partially funded by the Spanish MICINN Projects TIN2014-53234-C2-2-R, TIN2015-71799-C2-2-P and ENE2015-64117-C5-1-R, and by AGAUR DI-2016-091.

DOCTORATS INDUSTRIALS