SATTVA: SpArsiTy inspired classificaTion of malware VAriants

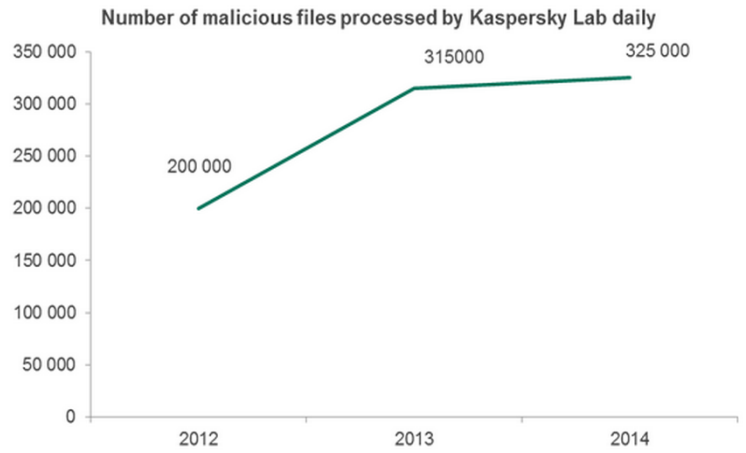
Lakshmanan Nataraj, S. Karthikeyan, B.S. Manjunath
Vision Research Lab
University of California, Santa Barbara

Sattva (सत्त्व) means Purity

Introduction

- The number of malware is increasing!
- In 2014, Kaspersky Lab reported they process on average 325,000 malware per day
- The main reason for such a deluge is:

  malware mutation: the process of creating new malware from existing ones



http://usa.kaspersky.com/about-us/press-center/press-releases/kaspersky-lab-detecting-325000-new-malicious-files-every-day

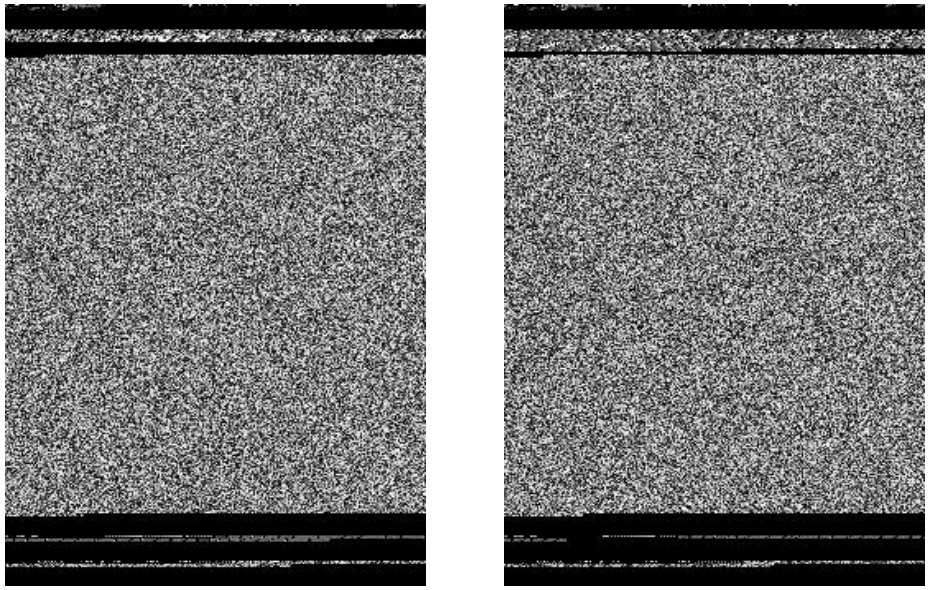
Introduction

- Variants are created either by making small changes to the malware code or by changing the structure of the code using executable packers

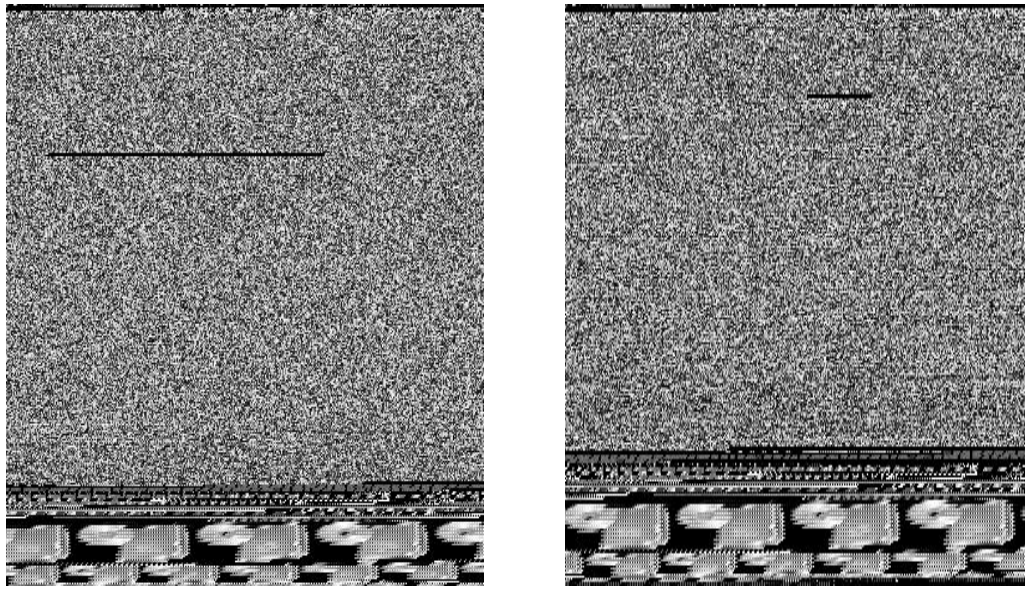
- Based on their function, variants are classified into different *malware families*

- Identifying the family of a malware plays an important role in understanding and thwarting new attacks

Examples of malware variants



Variants of Family *Alueron.gen!J*



Variants of Family *Fakerean*

Problem Statement

- Consider a Malware Dataset comprising of:
  - N labelled malware
  - L malware families
  - P malware per family
- Problem is to *identify the family of an unknown malware* $\mathbf{u}$

Related Work

- Static Code analysis based features
  - Disassembles the executable code and studies its control flow
  - Suffers from obfuscation (packing)
- Dynamic analysis based features
  - Executes malware in a virtual environment and studies its behavior
  - Time consuming and many recent aware are VM aware
- Statistical and Content based features
  - Analyzes statistical patterns based on the malware content
  - n-grams, fuzzy hashing, Image similarity based features

Statistical and Content based Features

- n-grams
  - n-grams are computed either on raw bytes or instructions
  - n > 1 which makes this computationally expensive
- Fuzzy hashing (ssdeep, pehash)
  - Fuzzy hashes are computed on raw bytes or PE parsed data
  - Does not work well on packed malware
- Image similarity
  - Malware binaries are converted to digital images
  - Image Similarity features (GIST) are computed on the malware

Image Similarity based Features

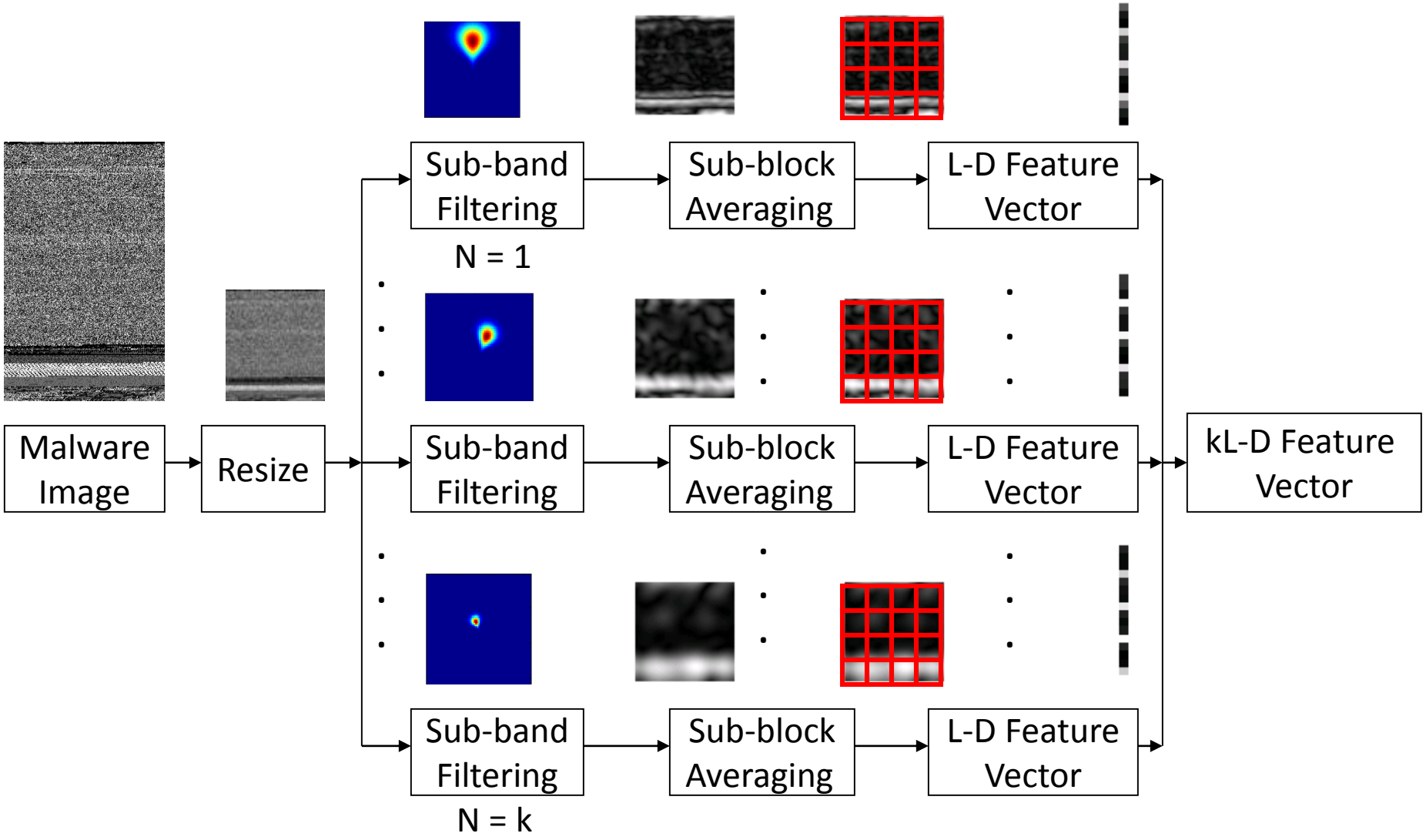
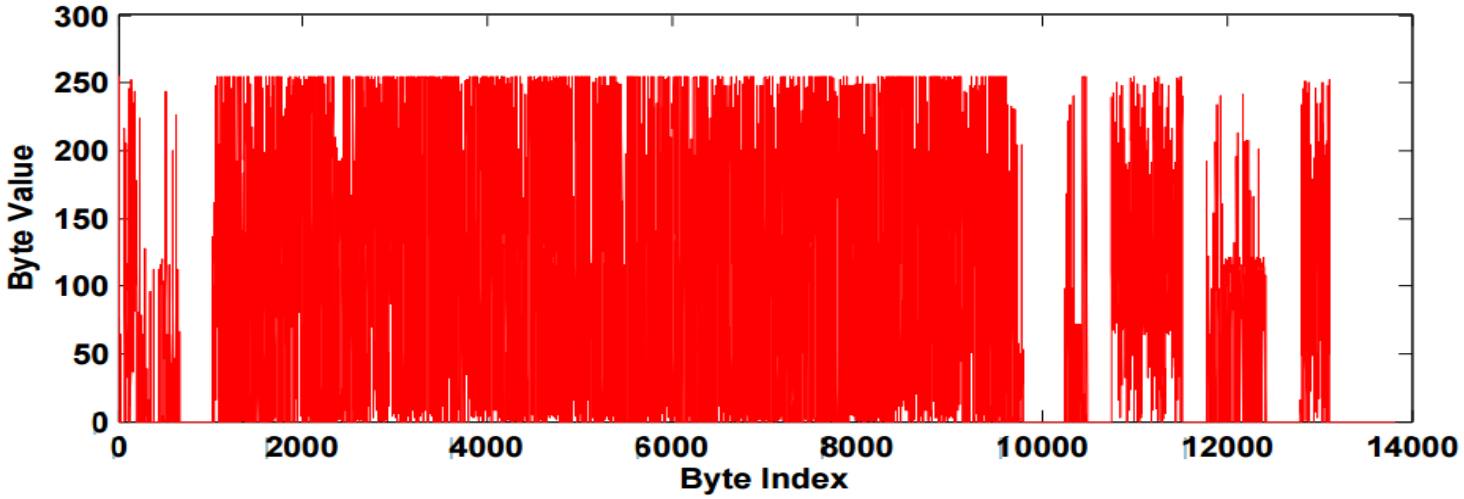


Image Similarity based Features

- Pros
  - Fast and compact
  - Better than static code based analysis (works on both packed and unpacked malware)
  - Comparable with dynamic analysis
- Cons
  - Arbitrary column cutting and reshaping
  - Images are resized to a small size for normalization which introduces interpolation artifacts
  - A large malware image, on resizing, lose lots of information

Approach – Signal Representation

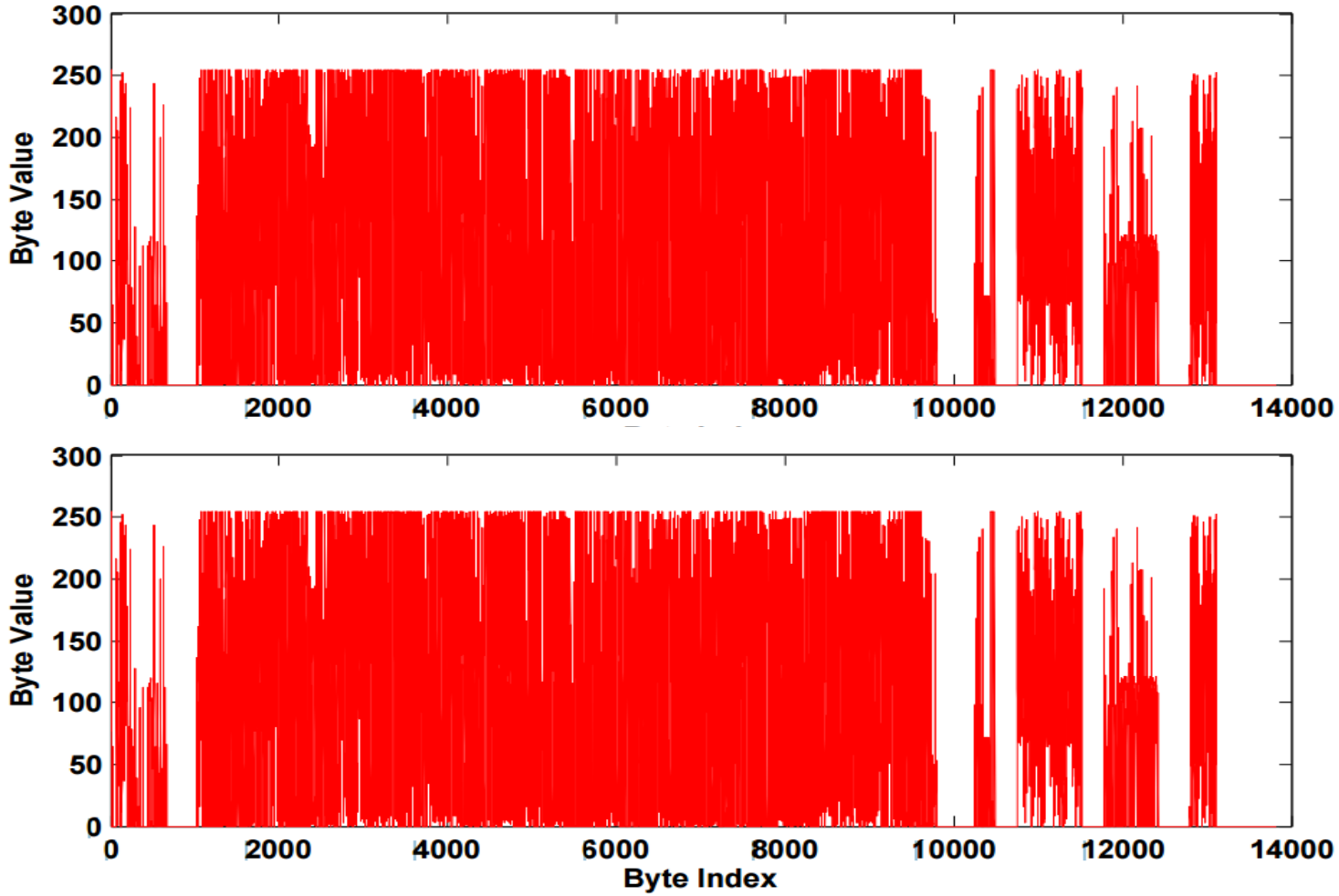
- Let $\mathbf{x}$ be the signal representation of a malware sample
- Every entry of $\mathbf{x}$ is a byte value of the sample in the range [0,255]



Variants in Signal Representation

Variant 1

Variant 2



Variants of recently exposed Regin malware. Differ only in 7 out of 13,284 (0.0527%)

Approach – Dataset as a Matrix

- Since malware are of different sizes, the vectors are zero padded such that all vectors are of length *M*, the number of bytes in the largest malware.

- We now represent the dataset as an $M \times N$ matrix **A**, where every column of **A** is a malware sample

Approach – Dataset as a Matrix

- Further, for every family *k*, (*k* = 1,2,…,L), we define an *M* x *P* block matrix A_k:

$$\mathbf{A}_k = [\mathbf{x}_{k1}, \mathbf{x}_{k2}, \ldots, \mathbf{x}_{kP}]$$

- $\mathbf{A}$ can now be represented as a concatenation of block matrices:

$$\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_L]$$

Approach – Sparse Linear Combination

- Let $\mathbf{u} \in \mathrm{R}^M$ be an unknown malware test sample whose family is to be determined.

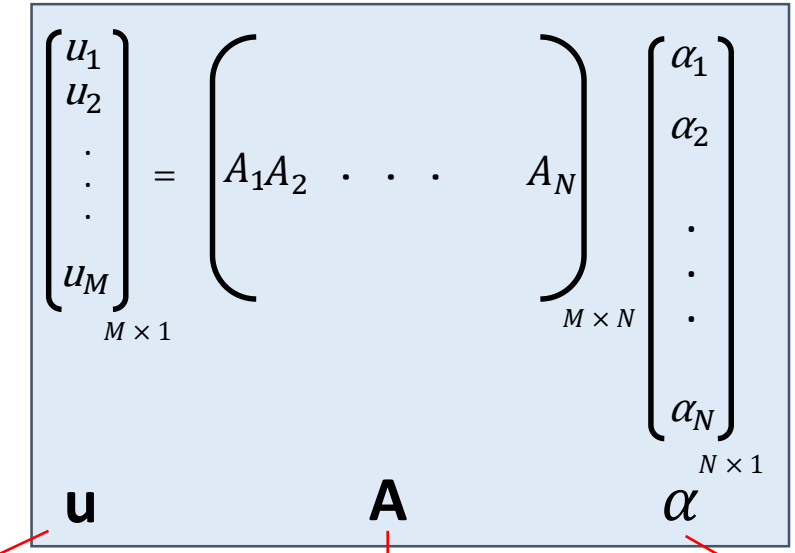
- Then $\mathbf{u}$ can be represented as a sparse linear combination of the training samples:

$$\mathbf{u} = \sum_{i=1}^{L} \sum_{j=1}^{P} \alpha_{ij} \boldsymbol{x}_{ij} = \mathbf{A}\boldsymbol{\alpha}$$

where $\boldsymbol{\alpha} = [\alpha_{11}, \alpha_{12}, \ldots, \alpha_{ij}, \ldots, \alpha_{LP}]^T$ is the coefficient vector

Approach – Sparse Linear Combination

$$\mathbf{u} = \mathbf{A}\boldsymbol{\alpha}$$



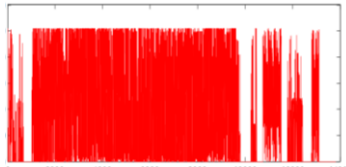
Unknown test sample

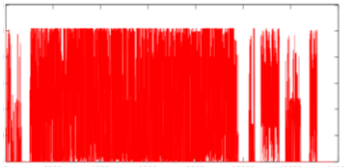
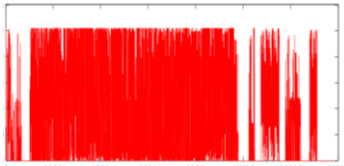
Matrix of training samples

Sparse Coefficient Vector

Illustration

- Let the unknown malware belong to family 2



$= \alpha_{21}$  $+ \alpha_{22}$ 

$$\boldsymbol{\alpha} = [0,0 \ldots, \alpha_{21}, \alpha_{22}, \ldots, 0,0]^T$$

Approach – Sparse Solution

- Sparsest solution can be obtained by Basis Pursuit by solving the l_1-*norm* minimization problem:

$$\hat{\boldsymbol{\alpha}} = \underset{\alpha' \in \mathrm{R}^N}{\operatorname{argmin}} ||\boldsymbol{\alpha}'||_{\mathbf{1}} \quad subject\ to\ \mathbf{u} = \mathbf{A}\boldsymbol{\alpha}'$$

where $||.||_{\mathbf{1}}$ represents the l_1-*norm*

Approach – Minimal Residue

- To estimate the family of $\mathbf{u}$, we compute residues for every family in the training set and then choose the family with minimal residue:

$$r_k(\mathbf{u}) = ||\mathbf{u} - \mathbf{A}\prod\nolimits_{\boldsymbol{k}}(\hat{\boldsymbol{\alpha}})\,||_2$$

$$\mathbf{c} = \underset{k}{\operatorname{argmin}}\, r_k(\mathbf{u})$$

where $\prod_{\boldsymbol{k}}(\hat{\boldsymbol{\alpha}})$ is the characteristic function that selects coefficients from $\hat{\boldsymbol{\alpha}}$ that are associated with family *k* and zeros out the rest, $\mathbf{c}$ is the index of the estimated family

Random Projections

- Dimensionality of malware *M* can be high

- We project all the malware to lower dimensions using Random Projections:

$$\mathbf{w} = \mathbf{Ru} = \mathbf{RA}\boldsymbol{\alpha}$$

where $\mathbf{R}$ is a $D \times M$ pseudo random matrix ($D \ll M$) and $\mathbf{w}$ is a $D \times 1$ lower dimensional vector

Sparse Solution

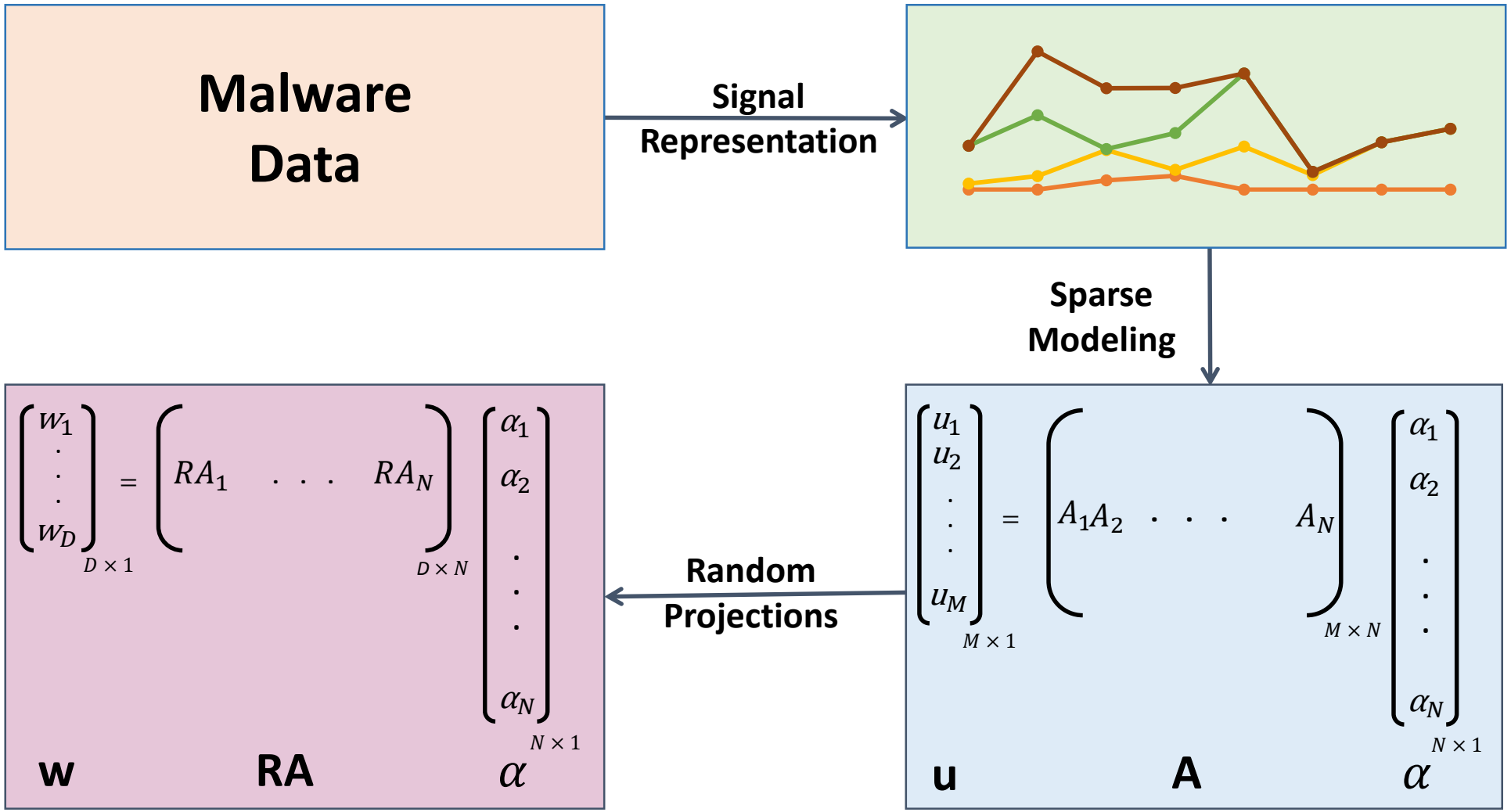
- The system of equations are underdetermined and can be solved using l_1-*norm* minimization:

$$\hat{\boldsymbol{\alpha}} = \underset{\alpha' \in \mathrm{R}^N}{\operatorname{argmin}} ||\boldsymbol{\alpha}'||_1 \quad subject\ to\ \mathbf{w} = \mathbf{RA}\boldsymbol{\alpha}'$$

$$\underset{D \times 1}{\begin{pmatrix} w_1 \\ \cdot \\ \cdot \\ \cdot \\ w_D \end{pmatrix}} = \underset{D \times N}{\begin{pmatrix} RA_1 & \cdots & RA_N \end{pmatrix}} \underset{N \times 1}{\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \cdot \\ \cdot \\ \cdot \\ \alpha_N \end{pmatrix}}$$

w **RA** α

Complete Approach



Modeling Malware Variants

- New variants are created from existing malware samples by making small changes and both variants share code

- We model a malware variant as:

$$\mathbf{u}' = \mathbf{u} + \mathbf{e_u} = \mathbf{A}\boldsymbol{\alpha} + \mathbf{e_u}$$

where $\mathbf{u}'$ is the vector representing malware variant and $\mathbf{e_u}$ is the error vector

Modeling Malware Variants

- This can be expressed in matrix form as:

$$\mathbf{u}' = [\mathbf{A} \quad \mathbf{I}_M] \begin{bmatrix} \boldsymbol{\alpha} \\ \mathbf{e_u} \end{bmatrix} = \mathbf{B_u s_u}$$

where $\mathbf{B_u} = [\mathbf{A} \quad \mathbf{I}_M]$ is an $M \times (N + M)$ matrix, $\mathbf{I}_M$ is an $M \times M$ Identity matrix, and $\mathbf{s_u} = [\boldsymbol{\alpha} \quad \mathbf{e_u}]^T$

- This ensures that the above system of equations is always underdetermined and spare solutions can be obtained

Sparse Solutions in Lower Dimensions

$$\hat{\boldsymbol{\alpha}} = \underset{\alpha' \in \mathrm{R}^N}{\operatorname{argmin}} ||\boldsymbol{\alpha}'||_{\mathbf{1}} \quad subject\ to\ \mathbf{w}' = \mathbf{B_w s_w}$$

$$r_k(\mathbf{w}') = ||\mathbf{w}' - \mathbf{B_w s_w} \prod\nolimits_{\boldsymbol{k}} (\hat{\boldsymbol{\alpha}}) \,||_{\mathbf{2}}$$

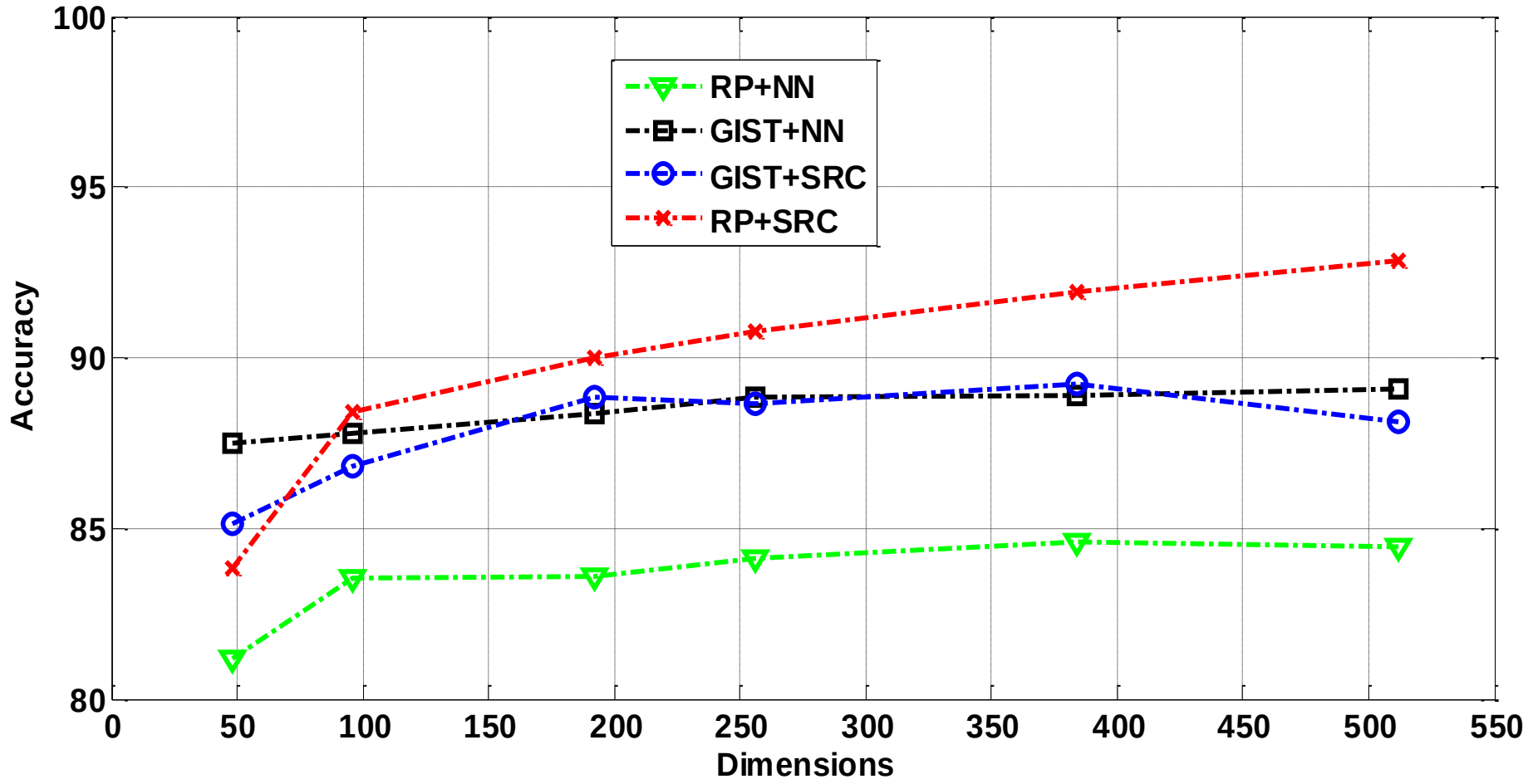
$$\mathbf{c} = \underset{k}{\operatorname{argmin}}\, r_k(\mathbf{w}')$$

where $\mathbf{w}' = \mathbf{w} + \boldsymbol{e}_{\mathbf{w}} = \mathbf{Ru} + \boldsymbol{e}_{\mathbf{w}}$, $\mathbf{B_w} = [\mathbf{RA}\boldsymbol{\alpha} \quad \mathbf{I_D}]$ is a $D \times (N + D)$ matrix, $\mathbf{I_D}$ is a $D \times D$ Identity matrix and $\mathbf{s_w} = [\boldsymbol{\alpha} \quad \mathbf{e_w}]^{\boldsymbol{T}}$.

Experiments

- Two datasets: Malimg and Malheur
- Malimg Dataset: 25 families, 80 samples per family, M = 840,960.
- Malheur Dataset: 23 families, 20 samples per family, M = 3,364,864.
- Vary Randomly projected dimensions D in {48,96,128,256,512}
- We compare with GIST features of same dimensions
- Two Classification methods: Sparse Representation based Classification (SRC) and Nearest Neighbor (NN) Classifier
- 80% Training and 20% Testing

Results on Malimg Dataset



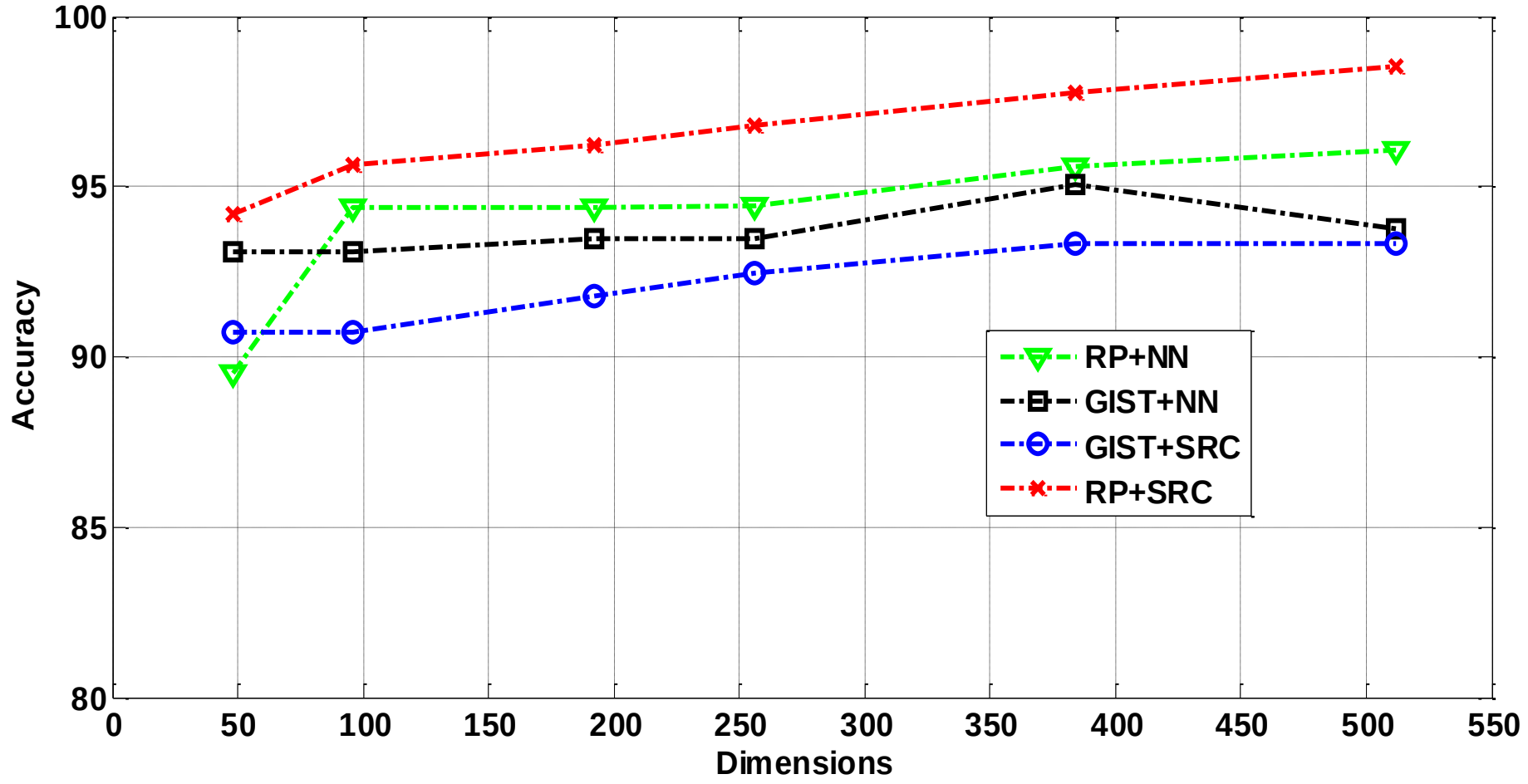
Results on Malimg Dataset

- Best classification accuracy of **92.83%** for combination of Random Projections (RP) + Sparse Representation based Classification (SRC) at D = 512

- Accuracies of GIST features for both classifiers almost the same in the range 88% - 90%

- Lowest accuracy for RP + Nearest Neighbor (NN) classifier

Results on Malheur Dataset



Results on Malheur Dataset

- Again, best classification accuracy of **98.66%** for combination of Random Projections (RP) + Sparse Representation based Classification (SRC) at D = 512

- Accuracies of GIST features for both classifiers almost the same at around 93%.

- However, the combination of RP + Nearest Neighbor (NN) classifier also had high accuracy of 96.06% - Projections Closely Packed

Comparison with Other Features

- Compare with 3 content based features:
  - *ssdeep* (fuzzy hash based feature)
  - GIST
  - 2-grams (2^16 dimensions)

Dataset	ssdeep	GIST	2-grams	RP
Malimg Dataset	67.63	89.08	91.75	**92.83**
Malheur Dataset	81.6	94.21	94.26	**98.55**

AV Labeling and Low Confidence Samples

- Ground Truth generated by Anti Virus (AV) software labels are not consistent
- Often, there are singletons or outliers in a family
- Using Sparse modeling, we show how singletons can be rejected

Low Confidence Samples

- Sparsity Coefficient Index (SCI) of a coefficient vector $\boldsymbol{\alpha}$:

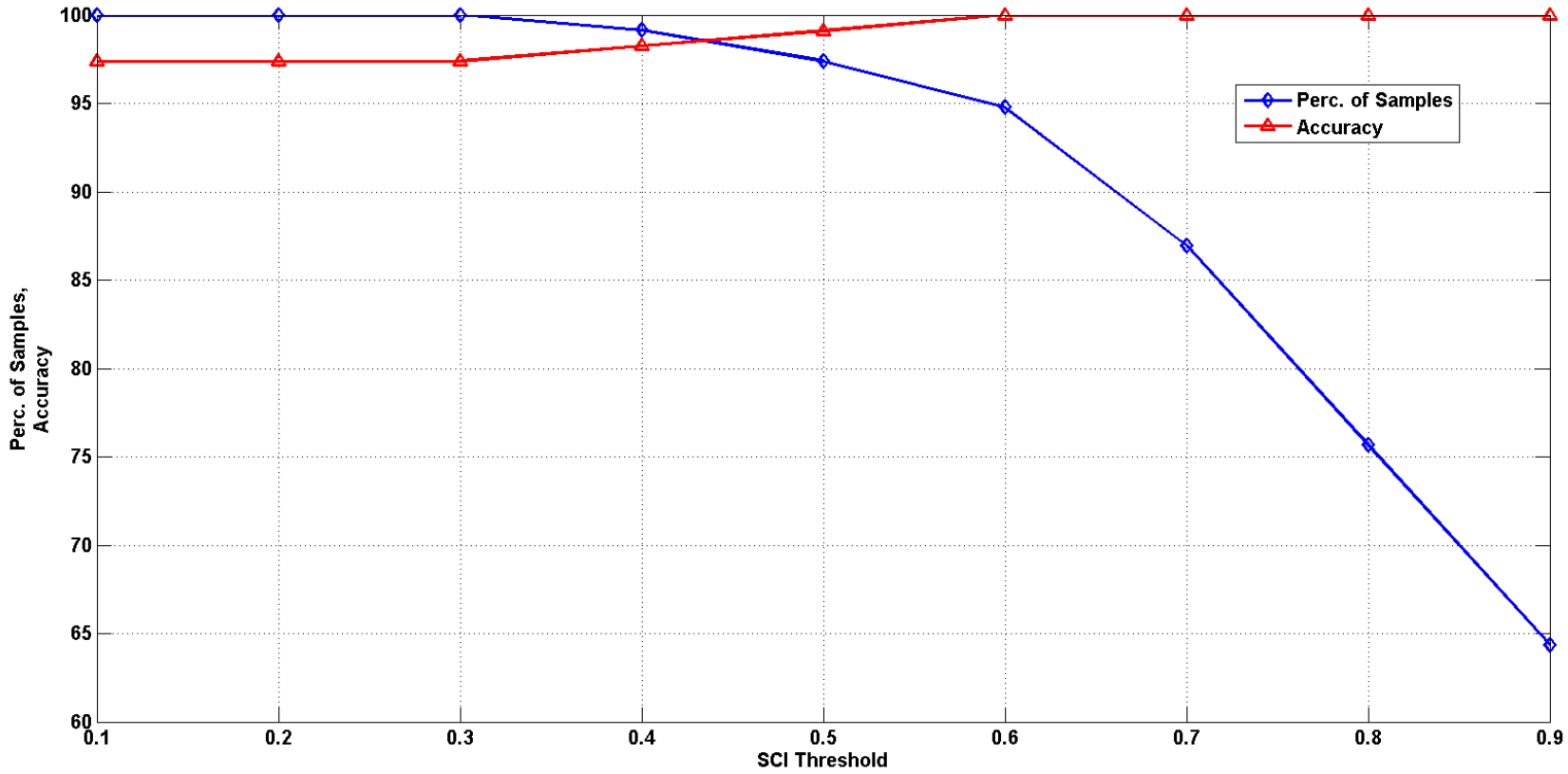
$$SCI(\boldsymbol{\alpha}) = \frac{\frac{L.max \,||\, \prod_i(\boldsymbol{\alpha})||_1}{||\boldsymbol{\alpha}||_1} - 1}{L-1}$$

- SCI = 1 → Test sample is linear combination of one family
- SCI = 0 → Test sample spread across all families
- SCI is a confidence measure and a threshold $\tau \in (0,1)$ can be used to reject potential low confidence samples.

Low Confidence Samples

- For both datasets, we fix D = 512 and vary τ
- For the Malimg Dataset, “accuracy” of **100%** is achieved at $\boldsymbol{\tau = 0.5}$, at which 25% of samples are rejected
- For the Malheur Dataset, “accuracy” of **100%** is achieved at $\boldsymbol{\tau = 0.6}$, with only 5% samples rejected

SCI Threshold for Malheur Dataset



Orthogonal Matching Pursuit (OMP)

- Basis Pursuit (BP) is computationally expensive
- Orthogonal Matching Pursuit (OMP) is a greedy method which does approximate l_1-*norm* minimization
- Iteratively selects a subset from the training set that are almost orthogonal

Basis Pursuit (BP) vs Orthogonal Matching Pursuit (OMP)

- OMP several times faster than BP (18 times for Malimg and 30 times for Malheur)

- But Accuracy slightly lesser for both datasets (Tradeoff)

Dataset	BP Accuracy	OMP Accuracy	BP Comp Time	OMP Comp Time
Malimg Dataset	92.83	89.25	420	**24**
Malheur Dataset	98.55	97.39	180	**6**

Large Scale Experiments

- Two diverse large scale datasets (no results reported on these)

- Used OMP on both with 80% training and 20% testing

- **Offensive Computing Dataset**:
  - 2,124 families, 20 samples per family, N = 42,480 and M = 9.3 Mb
  - Many families and fewer samples per family

- **Anubis Dataset**:
  - 209 behavioral clusters, 176 samples per cluster, N = 36,784, M = 8.1 Mb
  - Fewer clusters and more samples per cluster

Results on Offensive Computing Dataset

- Average Classification Accuracy with 2,124 families = **66.34%**
- **927** families had **100%** accuracy with SCI value of **0.97**
- At an SCI threshold of 0.6, accuracy = **77.08%** with 24.78% samples rejected
- Overall computation time was 4 hours on a standard desktop without parallelization

Results on Anubis Dataset

- Average Classification Accuracy with 209 clusters = **57.36%**
- **27** clusters had **100%** accuracy and 50 clusters had > 90% accuracy with SCI value of **0.97**
- At an SCI threshold of 0.6, accuracy = **77.12%** with 34.64% samples rejected
- Overall computation time was 3 hours on a standard desktop without parallelization

Discussion

- Accuracies for both datasets are similar (**77%**) at an SCI threshold of 0.6

- Computation time depends on both the total number of samples and number of classes

Future Work

- Use Random Projections as Malware Signatures
  - Project the full malware and individual sections to lower dimensions and represent the malware as bag of randomly projected features

- Finding the exact source of malware variants
  - Use the error model to find the commonalities between variants and also the exact positions where they vary

Conclusion

- We presented a novel method for identifying malware families using a combination of Sparse Representation based Classification and Random Projections
- We represented the malware binaries as signals, thus opening avenues for applying signal processing techniques to analyze malware
- We showed the efficacy and scalability of our method on real large malware datasets

Thank you

Questions?