FARE: Enabling Fine-grained Attack Categorization under Low-quality Labeled Data

JunJie Liang*, Wenbo Guo*, Tongbo Luo, Vasant Honavar, Gang Wang, Xinyu Xing
Pennsylvania State University & UIUC

* Equal contribution.
Background.

- Deep learning techniques have been broadly adopted in cybersecurity.
  - Malware Analysis.
    - Transcend, USENIX Security’17.
    - Drebin, NDSS’14.
  - Intrusion Detection.
    - Deeplog, CCS’17.
    - Log2vec, CCS’19
  - Binary Analysis.
    - Function start identification, USENIX Security’15.
    - DEEPVSA, USENIX Security’19.
  - Etc.
Background.

• The success of DL heavily relies on **accurate and sufficient labeled training data**.

• This requirement can be easily broken in security applications – low-quality labels.
  • E.g., malware detection.
  • Labeling malware requires tremendous efforts from domain experts.
    • Short of domain experts/efforts – large volume of unlabeled data and malware classes.
  • A malware family could evolve into thousands of subfamilies in a short period of time.
    • Missing knowledge of these subfamilies - coarse-grained labels.

Professional analysts: malware?
Time: hours or days.
Highly likely to make an error.

Dev a thousands of subfamilies.
Outline

• Problem Scope & Definition.
• Key technique: FARE - Fine-grained Attack Categorization through Representation Ensemble.
• Evaluation.
• Discussion & Conclusion.
Outline

• Problem Scope & Definition.
  • Key technique: FARE - Fine-grained Attack Categorization through Representation Ensemble.
  • Evaluation.
  • Discussion & Conclusion.
Problem Scope.

• Missing classes – labeled data cannot cover all classes.
  • E.g., malware classification.
    • Unrealistic to assume the knowledge of all malware families.
    • Leave the unseen classes as unlabeled data.

• Coarse grained labels – mistakenly group several classes into one group.
  • E.g., malware classification.
    • Only aware of the parent malware class.

• A small proportion of labeled data in each known class.

• Noisy/corrupted labels – random label errors/poison labels in the known classes.
Problem Definition.

Given a dataset with $n$ true classes.

• Missing classes.
  • Labels of $n_c$ classes are completely missing.
  • The other $(n - n_c)$ classes only have 1% of labeled samples.

• Coarse-grained labels.
  • Original $n_g$ classes are labeled as one union class.
  • These $(n - n_g + 1)$ classes only have 1% of labeled samples.

• Goal: recover the true clustering structure of the input data.
  • Identify there are $n$ clusters.
  • Correctly assign all the data to the $n$ clusters.
Outline

• Problem Scope & Definition.

• **Key technique: FARE - Fine-grained Attack Categorization through Representation Ensemble.**

• Evaluation.

• Discussion & Conclusion.
Technical Overview.

- Utilize various unsupervised learning methods to cluster the entire dataset.
  - Extract useful/reliable categorization information from the input data.
  - *Ensemble the clustering results with the given labels*.

- **Contrastive learning** - Use the fused labels to train an input transformation net.
  - Transform the high dimensional data into a *lower latent space*.
  - Distance is well defined – *Euclidean distance*.
  - Similar samples have a *shorter* distance.

- Final clustering – perform clustering at the latent space.
  - *K-means* clustering with *Euclidean distance* as the distance measure.
## Technical Detail

### Table A: Neighborhood models.

<table>
<thead>
<tr>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>x₄</th>
<th>x₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>G. labels</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>K-means</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DEC</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

### Table B: Neighborhood relationships \( \{y_{ij}\} \)

<table>
<thead>
<tr>
<th>(1, 2)</th>
<th>(1, 3)</th>
<th>(1, 4)</th>
<th>(1, 5)</th>
<th>(2, 3)</th>
<th>(2, 4)</th>
<th>(2, 5)</th>
<th>(3, 4)</th>
<th>(3, 5)</th>
<th>(4, 5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G. labels</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K-means</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DEC</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Contrastive Learning

\[
\mathcal{L}(x_i, x_j) = \sum_{m \in \mathcal{M}} \pi_m \delta_{ij}^m \tilde{L}(x_i, x_j | m) \\
= \sum_{m \in \mathcal{M}} \pi_m \delta_{ij}^m \left[ y_{ij}^m d_{ij}^2 + (1 - y_{ij}^m)(\alpha - d_{ij})^2_+ \right] \\
d_{ij} = d(x_i, x_j) = \| h_i - h_j \|_2
\]
Table A: Neighborhood models.

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>G. labels</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K-means</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>DEC</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table B: Neighborhood relationships $\{y_{ij}\}$

<table>
<thead>
<tr>
<th></th>
<th>(1, 2)</th>
<th>(1, 3)</th>
<th>(1, 4)</th>
<th>(1, 5)</th>
<th>(2, 3)</th>
<th>(2, 4)</th>
<th>(2, 5)</th>
<th>(3, 4)</th>
<th>(3, 5)</th>
<th>(4, 5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G. labels</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>K-means</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DEC</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Contrastive learning

Input Trans. Net

K-means Clustering

Human analyst

Clusters

Classes
Outline

• Problem Scope & Definition.
• FARE - Fine-grained Attack Categorization through Representation Ensemble.
• Evaluation.
• Conclusion.
Experiment Setup and Design.

- Datasets – randomly split datasets into training/validation/testing set.
  - Android Malware: one benign class and five malicious classes; 270,000 samples.
    - Features: 100 dimensions, encoding of sand-box behaviors.
  - Network intrusion (KDDCUP’99): 9 classes; 493,346 samples; imbalance.
    - Features: 120 dimensions, network behaviors.

- Evaluation metric – AMI (widely used unsupervised metric) and Accuracy.

- Experiment design:
  - FARE vs. baselines approach in *no label settings*.
  - FARE vs. baselines approach in *missing class settings*.
  - FARE vs. baselines approach in *coarse-grained label settings*.
Experiment Results (in no label setting).

- FARE is more effective than existing clustering and ensemble clustering methods.
- FARE’s computational cost depends on the base clustering methods (Kmean, DBSCAN, DEC); significantly less computationally intensive than ensemble approaches (CSPA, HGPA).

### Table: Experiment Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric</th>
<th>MALWARE</th>
<th></th>
<th>Network Intrusion</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AMI</td>
<td>Accuracy</td>
<td>Run (s)</td>
<td>AMI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.74 ± 0</td>
<td>0.81 ± 0.01</td>
<td>432.12</td>
<td>0.78 ± 0</td>
</tr>
<tr>
<td>Full Training set</td>
<td>Kmeans</td>
<td>0.47 ± 0.12</td>
<td>0.51 ± 0.04</td>
<td>26.99</td>
<td>0.39 ± 0.18</td>
</tr>
<tr>
<td></td>
<td>DBSCAN</td>
<td>0.69 ± 0.03</td>
<td>0.77 ± 0.02</td>
<td>174.63</td>
<td>0.38 ± 0.1</td>
</tr>
<tr>
<td></td>
<td>DEC</td>
<td>0.37 ± 0.09</td>
<td>0.47 ± 0.07</td>
<td>342.42</td>
<td>0.64 ± 0.12</td>
</tr>
<tr>
<td></td>
<td>Unsup. FARE</td>
<td>0.72 ± 0.01</td>
<td>0.80 ± 0.01</td>
<td>77.33</td>
<td>0.76 ± 0</td>
</tr>
<tr>
<td></td>
<td>CSPA</td>
<td>0.5 ± 0.04</td>
<td>0.61 ± 0.06</td>
<td>176.29</td>
<td>0.36 ± 0.11</td>
</tr>
<tr>
<td></td>
<td>HGPA</td>
<td>0.57 ± 0.03</td>
<td>0.69 ± 0.05</td>
<td>90.1</td>
<td>0.4 ± 0.09</td>
</tr>
</tbody>
</table>
Experiment Results (in missing class setting).

- Supervised DNN performs extremely poor.
- FARE is more effective than baselines.
  - Recovering the correct classes.
  - Assigning samples correctly (Higher AMI).
Experiment Results (in coarse-grained label setting).

- Supervised DNN performs extremely poor.
- FARE is more effective than baselines.
  - Recovering the correct classes.
  - Assigning samples correctly (Higher AMI).
Real-world Application.

• FARE - fraudulent accounts identification for an e-commerce service company.
  • Dataset – 200,000 active users; 264-dimensional feature vectors; 0.5% fraudulent/0.1% trustworthy users.
  • A/B test experiment – verify the FARE results.
    • Group-A: Fraudulent accounts identified by FARE; Group-B: Labeled trustworthy accounts.
    • Force a two-step authentication and monitor login attempt rate (LAR)/authentication pass rate (APR).
• Experiment results.

<table>
<thead>
<tr>
<th>Group</th>
<th>1-day</th>
<th>1-week</th>
<th>1-month</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(LAR, APR)</td>
<td>(LAR, APR)</td>
<td>(LAR, APR)</td>
</tr>
<tr>
<td>A: FARE-detected</td>
<td>(20.9%, 0.0%)</td>
<td>(25.3%, 0.0%)</td>
<td>(39.3%, 0.0%)</td>
</tr>
<tr>
<td>B: Confirmed-legit</td>
<td>(22.1%, 100%)</td>
<td>(27.9%, 100%)</td>
<td>(30.9%, 100%)</td>
</tr>
</tbody>
</table>

• None of the FARE-detected fraudulent accounts pass the two-step authentication.
• A manual analysis of the identified fraudulent clusters – discover unseen behaviors.
  • Deal-hunter: over-Heavy coupon usages.
  • Click-farm: regularly buy products from certain retailers and leave positive reviews, then return and get a refund.
Outline

• Problem Scope & Definition.
• FARE - Fine-grained Attack Categorization through Representation Ensemble.
• Evaluation.

• Conclusion.
Conclusion.

• Low-quality labels pose a crucial challenge to deploy supervised DNNs in security applications.

• Contrastive Learning with ensemble clustering enables fine-grained attack categorization.

• FARE can serve as an effective tool for attack categorization in real-world security applications.
Thank you very much!

Code and data can be found @ https://github.com/junjieliang672/FARE

Wenbo Guo

✉️ wzg13@ist.psu.edu
🐦 @WenboGuo4
🏠 http://www.personal.psu.edu/wzg13/
Discussion.

- FARE vs. semi-supervised learning (SSL) & few-shot learning.
  - SSL – a small proportion of labeled data from each class.
  - FARE – similar with SSL setup but with missing classes/coarse-grained labels.
  - Few-shot learning - transfer learning (little knowledge about the second task), require side information.

- Computational complexity – quadratic to the batch size.
  - Depend on the cost of base clustering methods (DBSCAN could be slow).

- Hyper-parameters – selecting via a validation set.

- Adversarial resistance (Poisoning labels) – FARE’s performance slightly drops as more labels are corrupted.