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

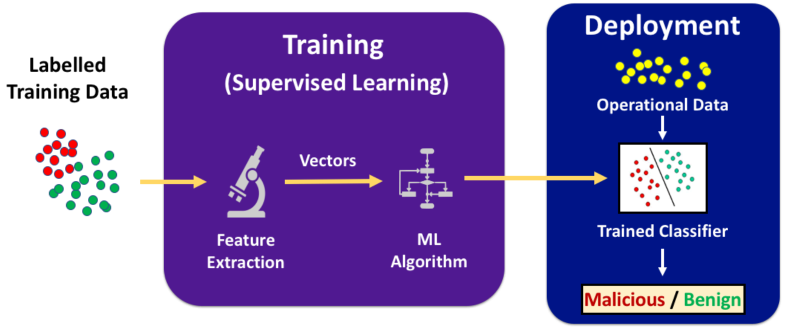
JunJie Liang*, **Wenbo Guo***, Tongbo Luo, Vasant Honavar, Gang Wang, Xinyu Xing

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** - **F**ine-grained **A**ttack Categorization through **R**epresentation **E**nsemble.
- Evaluation.
- Discussion & Conclusion.

Outline

- **Problem Scope & Definition.**
- Key technique**: FARE** - **F**ine-grained **A**ttack Categorization through **R**epresentation **E**nsemble.
- Evaluation.
- Discussion & Conclusion.

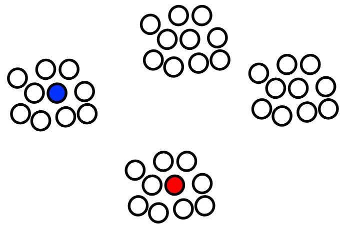
Problem Scope.

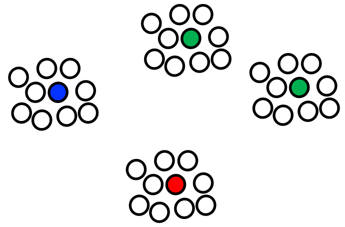
- Missing classes – labeled data cannot cover all classes.
  - E.g., malware classification.
    - Unrealistic to assume the knowledge of all malware families.
    - Leave the unseen classes as unlabeled data.
- Coarse grained labels – mistakenly group several classes into one group.
  - E.g., malware 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.

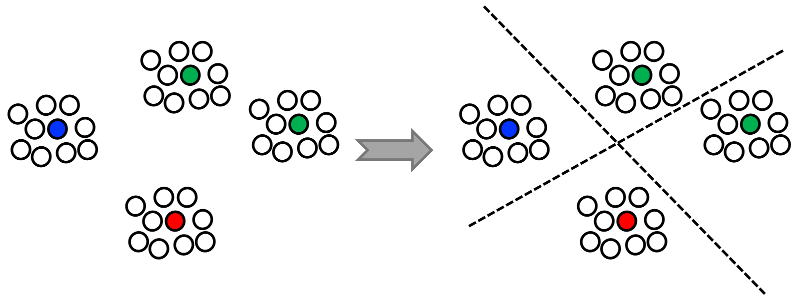
Given a dataset with n true classes.

- 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.

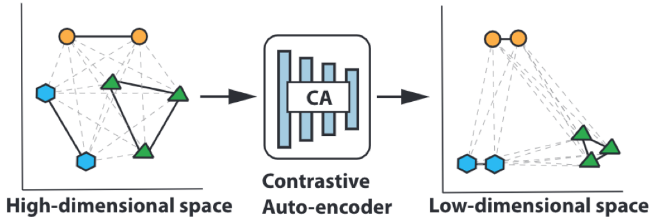


Outline

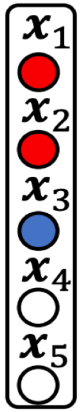
- Problem Scope & Definition.
- **Key technique: FARE - Fine-grained Attack Categorization through Representation Ensemble.**
- Evaluation.
- Discussion & Conclusion.

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



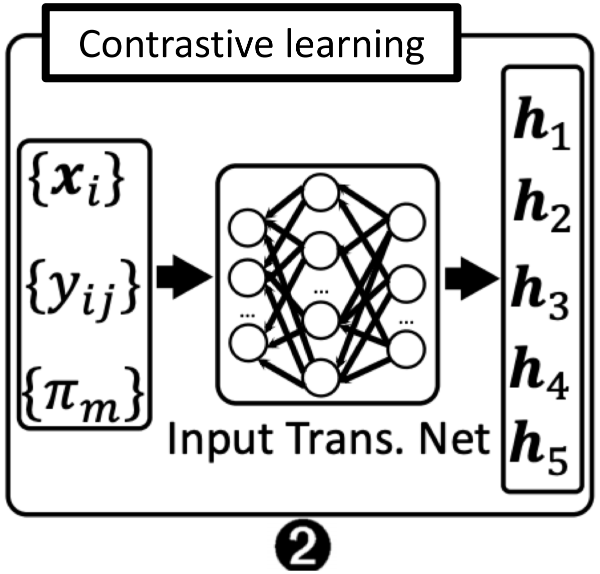
	x_1	x_2	x_3	x_4	x_5
G. labels	1	1	2	-	-
K-means	3	2	1	4	1
DBSCAN	2	1	1	1	3
DEC	2	3	2	2	1

Table A: Neighborhood models.

	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(2, 5)	(3, 4)	(3, 5)	(4, 5)
G. labels	1	0	-	-	0	-	-	-	-	-
K-means	0	0	0	0	0	0	0	0	1	0
DBSCAN	0	0	0	0	1	1	0	1	0	0
DEC	0	1	1	0	0	0	0	1	0	0

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

❶

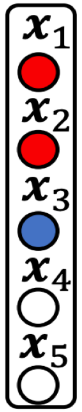


$$\mathcal{L}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m \in \mathcal{M}} \pi_m \delta_{ij}^m \tilde{\mathcal{L}}(\boldsymbol{x}_i, \boldsymbol{x}_j | m)$$

$$= \sum_{m \in \mathcal{M}} \pi_m \delta_{ij}^m \left[y_{ij}^m d_{ij}^2 + (1 - y_{ij}^m)(\alpha - d_{ij})_+^2 \right]$$

$$d_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{h}_i - \boldsymbol{h}_j\|_2$$

Technical Detail



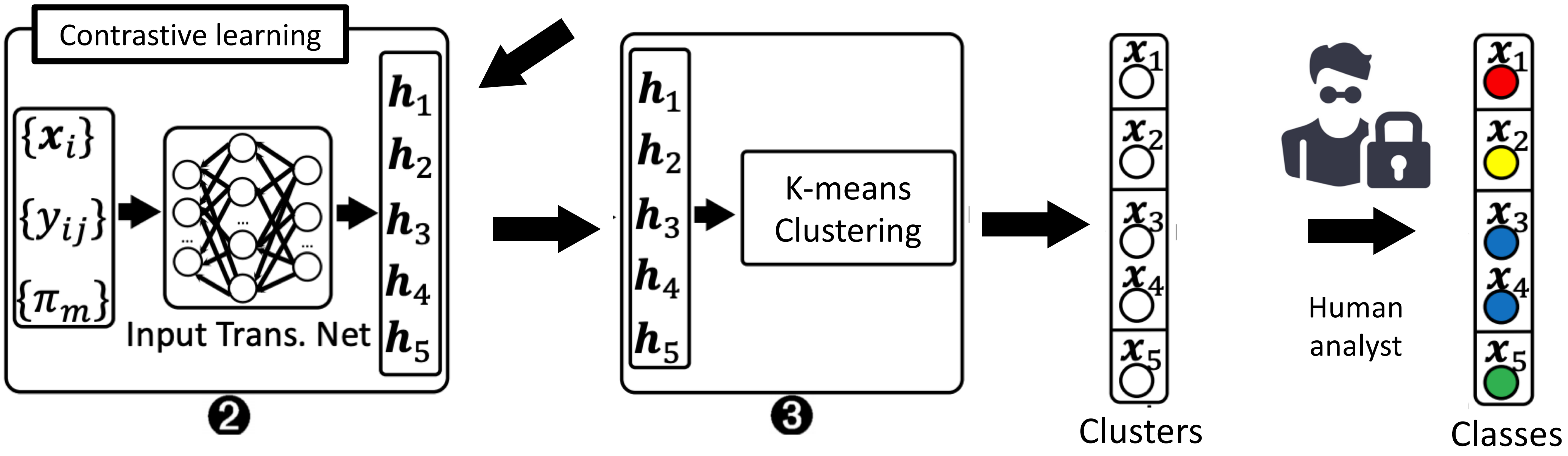
	x_1	x_2	x_3	x_4	x_5
G. labels	1	1	2	-	-
K-means	3	2	1	4	1
DBSCAN	2	1	1	1	3
DEC	2	3	2	2	1

Table A: Neighborhood models.

	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 3)	(2, 4)	(2, 5)	(3, 4)	(3, 5)	(4, 5)
G. labels	1	0	-	-	0	-	-	-	-	-
K-means	0	0	0	0	0	0	0	0	1	0
DBSCAN	0	0	0	0	1	1	0	1	0	0
DEC	0	1	1	0	0	0	0	1	0	0

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

❶



Outline

- Problem Scope & Definition.
- **FARE** - **F**ine-grained **A**ttack Categorization through **R**epresentation **E**nsemble.
- **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*.

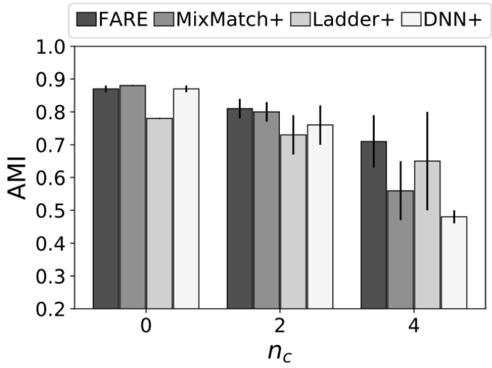
Experiment Results (in no label setting).

Dataset		MALWARE			Network Intrusion		
Metric		AMI	Accuracy	Runtime (s)	AMI	Accuracy	Runtime (s)
Full Training set	Unsup. FARE	0.74 ± 0	0.81 ± 0.01	432.12	0.78 ± 0	0.99 ± 0	$8,942.52$
	Kmeans	0.47 ± 0.12	0.51 ± 0.04	26.99	0.39 ± 0.18	0.64 ± 0.12	16.30
	DBSCAN	0.69 ± 0.03	0.77 ± 0.02	174.63	0.38 ± 0.1	0.66 ± 0.04	$8,918.36$
	DEC	0.37 ± 0.09	0.47 ± 0.07	342.42	0.64 ± 0.12	0.85 ± 0.04	725.58
10% Training set	Unsup. FARE	0.72 ± 0.01	0.80 ± 0.01	77.33	0.76 ± 0	0.98 ± 0	$1,801.74$
	CSPA	0.5 ± 0.04	0.61 ± 0.06	176.29	0.36 ± 0.11	0.64 ± 0.08	$2,013.77$
	HGPA	0.57 ± 0.03	0.69 ± 0.05	90.1	0.4 ± 0.09	0.79 ± 0.06	$1,804.82$

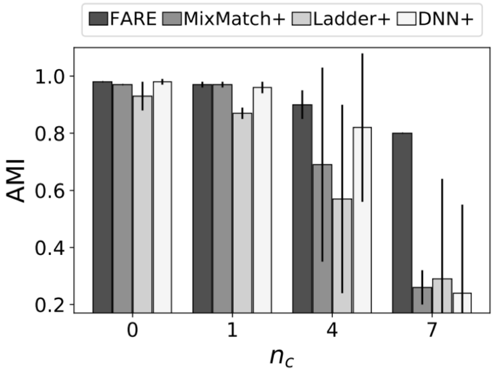
- FARE is more effective than existing clustering and ensemble clustering methods.
- FARE's computational cost depends on the base clustering methods (Kmean, DBSCAN, DEC); significantly less computationally intensive than ensemble approaches (CSPA, HGPA).

Experiment Results (in missing class setting).

Methods	Num. of missing classes (n_c)						
	Malware ($N = 6$)			Intrusion ($N = 9$)			
	0	2	4	0	1	4	7
FARE	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{8 \pm 1.25}$	$\mathbf{10 \pm 1.89}$
MixMatch	$\mathbf{6 \pm 0}$	4 ± 0	4 ± 0	5 ± 0.47	4 ± 0	6 ± 1.69	5 ± 1.41
Ladder	4 ± 0	4 ± 0	5 ± 0	5 ± 0	6 ± 2.44	6 ± 0	7 ± 2.36
DNN+	$\mathbf{6 \pm 0}$	5 ± 0.82	4 ± 0	5 ± 0.92	$\mathbf{6 \pm 0}$	6 ± 0	4 ± 0



(a) Malware categorization.

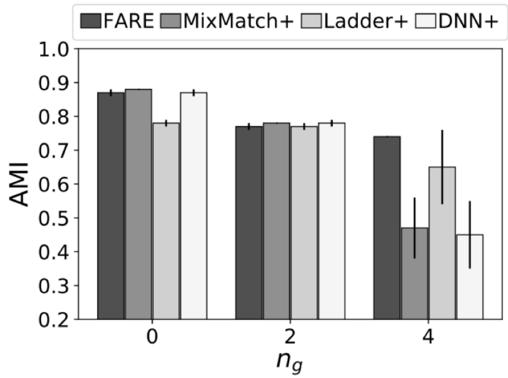


(b) Intrusion detection.

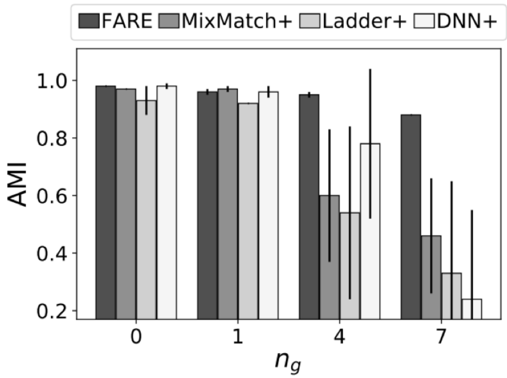
- Supervised DNN performs extremely poor.
- FARE is more effective than baselines.
    - Recovering the correct classes.
    - Assigning samples correctly (Higher AMI).

Experiment Results (in coarse-grained label setting).

Methods	Num. of mistaken grouped classes (n_g)				
	Malware ($N = 6$)		Intrusion ($N = 9$)		
	2	4	1	4	7
FARE	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{5 \pm 0}$	5 ± 0.47	4 ± 0
MixMatch	5 ± 0	4 ± 0	5 ± 0.47	5 ± 0.47	$\mathbf{7 \pm 1.25}$
Ladder	4 ± 0	5 ± 0.47	5 ± 0.47	7 ± 2.05	16 ± 3.77
DNN+	5 ± 0	5 ± 5.44	5 ± 0.47	$\mathbf{6 \pm 1.7}$	15 ± 2.49



(a) Malware categorization.



(b) Intrusion detection.

- 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.

Group	1-day (LAR, APR)	1-week (LAR, APR)	1-month (LAR, APR)
A: FARE-detected	(**20.9%**, **0.0%**)	(**25.3%**, **0.0%**)	(**39.3%**, **0.0%**)
B: Confirmed-legit.	(22.1%, 100%)	(27.9%, 100%)	(30.9%, 100%)

- *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** - **F**ine-grained **A**ttack Categorization through **R**epresentation **E**nsemble.
- 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.