From Grim Reality to Practical Solution: Malware Classification in Real-World Noise

S&P '23

Labels in Malware

- Key assumption in DL technique: Sufficient Training data with correct labels

- Incorrectly labeled Malware sample -> prevalent

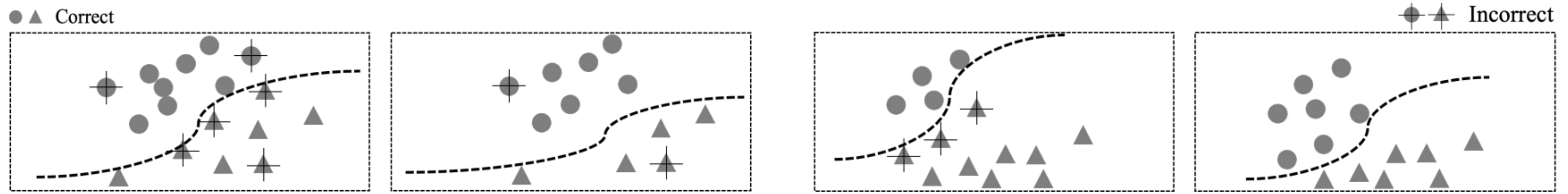
Overview

- Propose MORSE (**M**alware classificati**O**n f**R**om noi**S**y lab**E**ls)
- Analyze previous Noise Learning Solution’s limitation in the context of Malware
- Noise Learning Solution tailored for Malware

Noise Learning

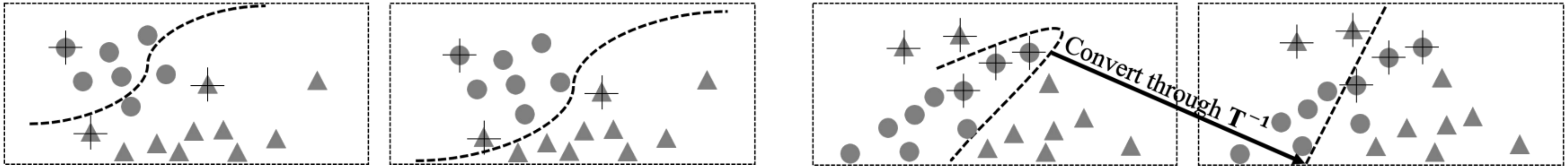
- Two Major Types
  - #1 Use all noisy data
  - #2 Use some noisy data

- Previous Works use the following 4 techniques
- Use all
  - Label Sanitization
  - Loss Robustification
  - Noise Matrix Estimation
- Use some
  - Sample Selection



(a) Sample selection selects samples possibly correctly labeled and then learns the decision boundary.

(b) Label sanitization corrects mistakenly labeled samples and then learns the decision boundary.



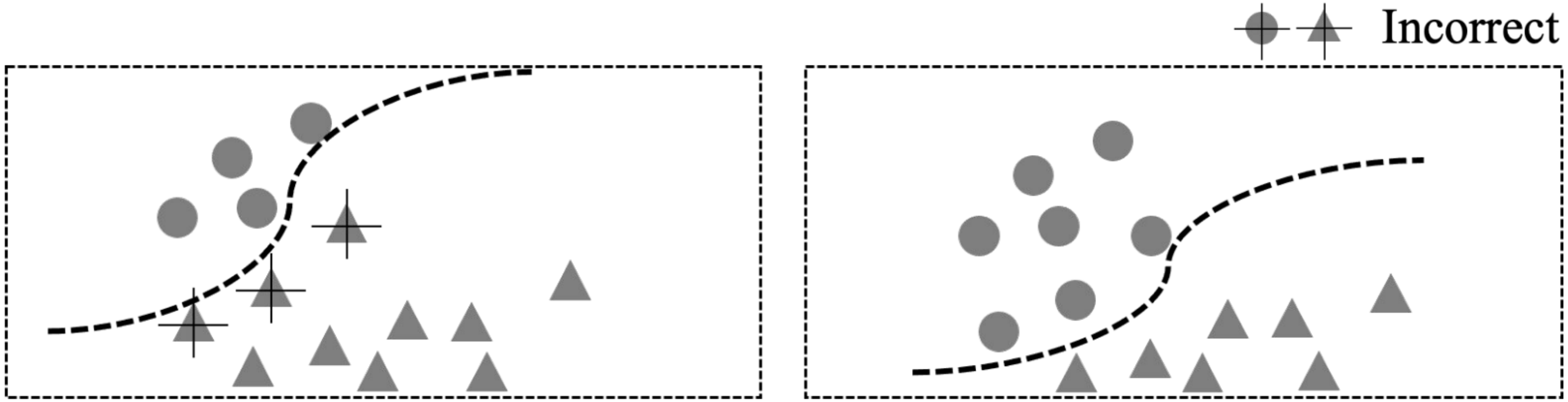
(c) Loss robustification uses a new, robust loss function to learn the decision boundary.

(d) Noise matrix estimation learns a transition matrix ("**T**") and then uses it to correct the inaccurate decision boundary.

Figure 1: The demonstration of existing noise learning methods in binary classification. Each geometry pattern represents a sample. The circle and triangle pattern indicates the different label of the sample. The geometry with a cross indicates the sample is incorrectly labeled. For example, circle with a cross denotes the sample is mistakenly labeled as "circle" and its true label should be "triangle". In each subfigure, the left shows the decision boundary learned directly from the noisy training dataset, whereas the right depicts the decision boundary learned by the corresponding noise learning method.

Label Sanitization

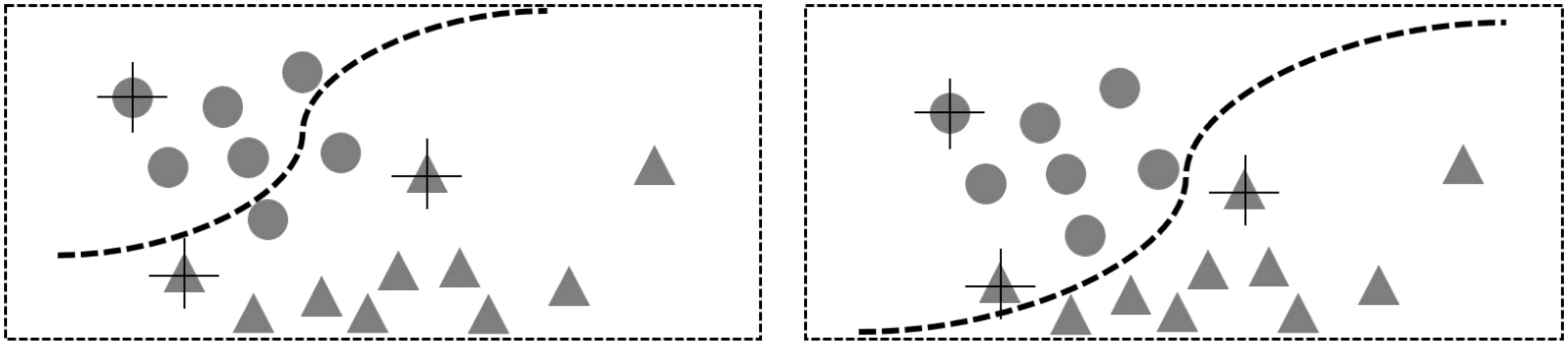
- Uses all entire noisy dataset for training
- Corrects Labels to offset their negative impact
- Research show incorrectly labeled data impact loss function differently



(b) Label sanitization corrects mistakenly labeled samples and then learns the decision boundary.

Loss Robustification

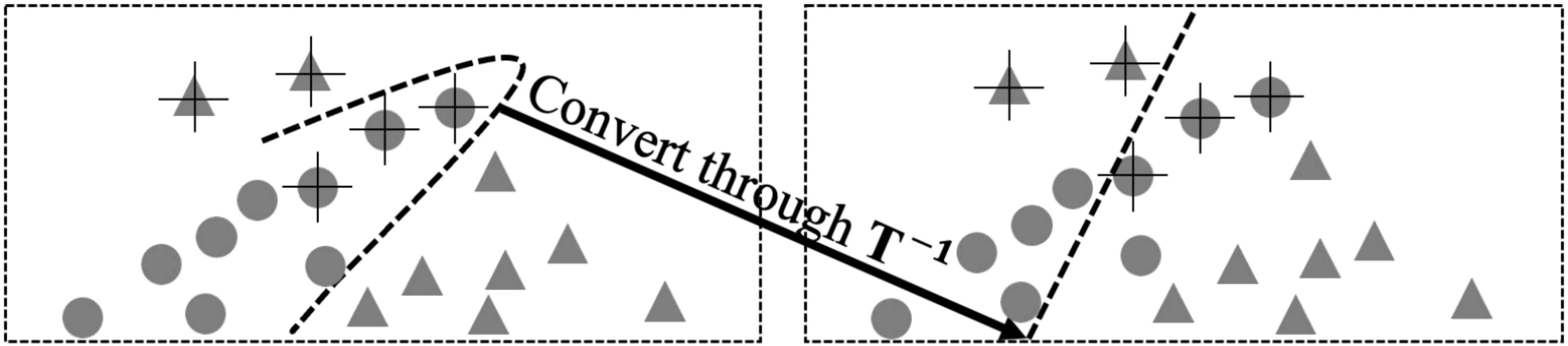
- Uses the entire noisy dataset
- Incorrect labels impacts loss function differently
- Propose a new loss function robust against noisy labels



(c) Loss robustification uses a new, robust loss function to learn the decision boundary.

Noise Matrix Estimation

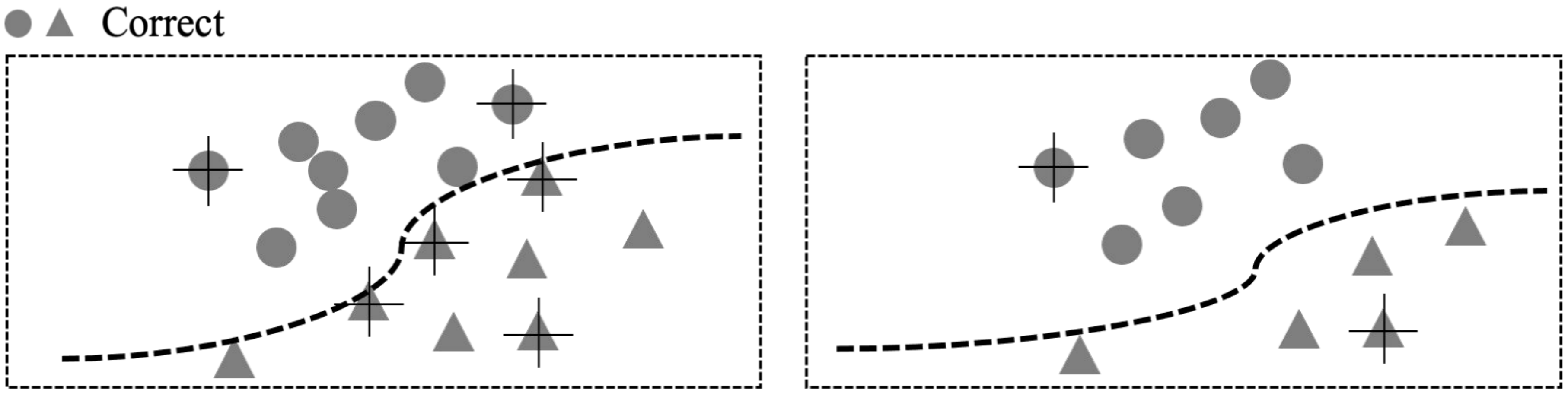
- Learn a transition matrix from entire noisy dataset
- Transition Matrix could flip incorrect prediction results made by the model trained on noisy training datasets.



(d) Noise matrix estimation learns a transition matrix ("**T**") and then uses it to correct the inaccurate decision boundary.

Sample Selection

- Minimize the impact of noisy labels
  - Identify incorrectly labeled samples
  - Eliminate or downplay the noisy samples from model training procedure



(a) Sample selection selects samples possibly correctly labeled and then learns the decision boundary.

Evaluation of Existing Methods

Dataset

- Windows PE dataset
- Android Dataset

- How do we know noise rate?

TABLE 1: Statistics of the Windows PE malware dataset.

ID	Family	# of Samples	# of Training samples	Noise rate in training set
0	Benign	500	400	0.000
1	VirLock	900	800	0.005
2	WannaCry	920	820	0.002
3	Upatre	440	340	0.032
4	Cerber	1044	944	0.156
5	Urelas	572	472	0.011
6	WinActivator	166	66	0.106
7	Pykspa	744	644	0.019
8	Ramnit	324	224	0.295
9	Gamarue	608	508	0.750
10	InstallMonster	299	199	0.472
11	Locky	157	57	0.544

TABLE 2: The statistics of the Android malware dataset.

ID	Family	# of Samples	# of Training samples	Noise rate in training set
0	Smserg	2687	2587	0.040
1	Benign	4683	4583	0.699
2	Autoins	200	100	0.150
3	Jiagu	678	578	0.235
4	Shedun	10867	10767	0.548
5	Wapron	857	757	0.136
6	Dnotua	136	36	0.583
7	Hiddad	185	85	0.000
8	Secneo	203	103	0.223
9	Triada	299	199	0.005
10	Secapk	155	55	0.055
11	Smspay	281	181	0.028
12	Qappusin	158	58	0.000

Windows PE Dataset

- We assume the labels are 100% accurate (500 benign, 6174 malicious)
  - Obtained from security lab
  - Carefully analyzed by at least three security analyst with 5+ years of experience
- Generation of noisy labels
  - Used VirusTotal
  - Upload all executables to VirusTotal (discovered 11.38% of PE files provided had at least one wrong label by a vendor)
  - Randomly change the label to what the vendor provided (noisy label)

TABLE 1: Statistics of the Windows PE malware dataset.

ID	Family	# of Samples	# of Training samples	Noise rate in training set
0	Benign	500	400	0.000
1	VirLock	900	800	0.005
2	WannaCry	920	820	0.002
3	Upatre	440	340	0.032
4	Cerber	1044	944	0.156
5	Urelas	572	472	0.011
6	WinActivator	166	66	0.106
7	Pykspa	744	644	0.019
8	Ramnit	324	224	0.295
9	Gamarue	608	508	0.750
10	InstallMonster	299	199	0.472
11	Locky	157	57	0.544

Android Dataset

- VirusShare2018, 4683 benign 16706 android malware
- Difficult to check label correctness in this case

- Generation of noisy labels
  - Automatically obtained noisy labels through Virus Total
  - Upload to VT, correct label: majority vote between vendors
  - Noisy Label: from one vendor (Ikarus)

TABLE 2: The statistics of the Android malware dataset.

ID	Family	# of Samples	# of Training samples	Noise rate in training set
0	Smserg	2687	2587	0.040
1	Benign	4683	4583	0.699
2	Autoins	200	100	0.150
3	Jiagu	678	578	0.235
4	Shedun	10867	10767	0.548
5	Wapron	857	757	0.136
6	Dnotua	136	36	0.583
7	Hiddad	185	85	0.000
8	Secneo	203	103	0.223
9	Triada	299	199	0.005
10	Secapk	155	55	0.055
11	Smspay	281	181	0.028
12	Qappusin	158	58	0.000

Models to Evaluate

- SOTA, Representative Method (via citation)

TABLE 3: Summary of our selected noise learning methods.

Category	Representative Method	State-of-the-art Method
Sample selection	Coteaching+[12]	Mentormix[6]
Label sanitization	Bootstrap[27]	LRT[28]
Loss robustification	GCE[8]	ELR[9]
Noise matrix estimation	Noise-adaption[10]	LIO[13]

Windows PE Evaluation Results

Methods	Average (%)			Class-11 Locky (%)		
	Accuracy	Precision	F_1	Accuracy	Precision	F_1
Vanilla DNN	93.08/0.25	93.65/0.51	92.57/0.36	44.33/3.78	96.34/4.22	60.48/3.22
Coteaching+	87.39/0.65 $p = 0.999$	92.24/0.39 $p = 0.998$	87.90/1.13 $p = 0.999$	2.00/2.89 $p = 0.999$	99.44/1.84 $p = 0.055$	3.92/3.16 $p = 0.996$
Mentormix	92.34/0.10 $p = 0.997$	93.40/0.14 $p = 0.812$	92.32/0.28 $p = 0.322$	41.17/4.13 $p = 0.767$	95.24/1.34 $p = 0.505$	57.82/2.14 $p = 0.434$
Bootstrap	92.92/0.30 $p = 0.876$	93.33/0.23 $p = 0.829$	92.27/1.91 $p = 0.503$	46.33/5.15 $p = 0.260$	92.71/2.69 $p = 0.858$	61.03/2.49 $p = 0.191$
LRT	92.52/0.21 $p = 0.995$	93.38/0.24 $p = 0.815$	92.15/0.18 $p = 0.817$	42.50/1.50 $p = 0.887$	93.85/4.67 $p = 0.753$	59.34/2.47 $p = 0.557$
Noise-adaption	92.65/0.24 $p = 0.978$	93.25/0.30 $p = 0.861$	92.18/0.22 $p = 0.488$	40.83/1.46 $p = 0.968$	90.04/3.68 $p = 0.968$	56.18/2.47 $p = 0.687$
LIO	92.30/0.35 $p = 0.990$	93.21/0.25 $p = 0.879$	92.01/0.15 $p = 0.435$	38.00/4.04 $p = 0.910$	90.12/4.23 $p = 0.958$	53.42/1.45 $p = 0.956$
GCE	92.15/0.27 $p = 0.998$	93.40/0.32 $p = 0.809$	91.74/0.65 $p = 0.962$	36.33/7.40 $p = 0.926$	94.85/3.11 $p = 0.705$	52.38/5.95 $p = 0.991$
ELR	91.84/0.18 $p = 0.999$	93.27/0.24 $p = 0.882$	91.20/0.27 $p = 0.998$	34.50/2.75 $p = 0.999$	96.39/3.37 $p = 0.493$	50.01/2.70 $p = 0.997$

Android Evaluation Results

Methods	Average (%)			Class-6 Dnotua (%)		
	Accuracy	Precision	F_1	Accuracy	Precision	F_1
Vanilla DNN	73.00/0.35	78.65/2.21	69.96/0.54	2.67/5.53	19.79/34.31	4.63/9.52
Coteaching+	72.73/0.61 $p = 0.895$	75.83/1.76 $p = 0.949$	69.44/0.41 $p = 0.963$	0.00/0.00 $p = 0.835$	0.00/0.00 $p = 0.873$	0.00/0.00 $p = 0.837$
Mentormix	75.52/0.24 $p = 0.001$	78.20/3.12 $p = 0.645$	72.15/0.88 $p = 0.003$	3.75/6.49 $p = 0.767$	20.24/31.24 $p = 0.563$	6.30/10.18 $p = 0.620$
Bootstrap	72.24/0.42 $p = 0.993$	77.41/3.94 $p = 0.738$	68.97/0.70 $p = 0.999$	0.33/0.47 $p = 0.817$	16.67/23.57 $p = 0.591$	0.65/0.92 $p = 0.816$
LRT	72.07/0.49 $p = 0.991$	79.02/3.15 $p = 0.416$	69.15/0.87 $p = 0.933$	2.50/5.59 $p = 0.517$	15.62/34.94 $p = 0.574$	4.31/9.64 $p = 0.519$
Noise-adaption	74.73/0.80 $p = 0.003$	78.33/3.01 $p = 0.605$	71.62/0.82 $p = 0.004$	5.50/7.46 $p = 0.132$	39.17/41.87 $p = 0.114$	9.40/12.63 $p = 0.138$
LIO	72.56/0.41 $p = 0.838$	80.02/2.83 $p = 0.233$	72.78/1.00 $p = 0.001$	3.00/5.86 $p = 0.468$	25.00/38.19 $p = 0.417$	5.24/10.09 $p = 0.466$
GCE	72.67/0.95 $p = 0.710$	78.90/3.98 $p = 0.459$	69.90/0.87 $p = 0.539$	5.00/7.00 $p = 0.314$	32.29/45.70 $p = 0.333$	8.66/12.24 $p = 0.314$
ELR	70.99/0.12 $p = 0.999$	76.46/0.77 $p = 0.967$	67.62/0.25 $p = 0.999$	0.00/0.00 $p = 0.835$	0.00/0.00 $p = 0.873$	0.00/0.00 $p = 0.837$

Hypothesis

- #1 Performance degradation is due to highly skewed dataset.
  (Locky Dnotua, both are smallest class in the dataset)

- #2 High noise rate reduces the number of clean samples useful for classifier training
  (Locky Dnotua have high noise rate 54%, 58.3%)

Hypothesis Test

- Synthetic Dataset form BODMAS (57293 samples 581 families)
- #1 Skewed
  - Remove proportion of families obtain two dataset with different imbalance (20x and 100x)
- #2 High Noise Rate
  - Randomly change the label for each dataset (30% and 60%)

Methods	Noise rate 0.3 and imbalance ratio 20x						Noise rate 0.3 and imbalance ratio 100x					
	Overall cls (%)			Rare cls (%)			Overall cls (%)			Rare cls (%)		
	Accuracy	Precision	F_1	Accuracy	Precision	F_1	Accuracy	Precision	F_1	Accuracy	Precision	F_1
Vanilla DNN	75.55/1.38	82.94/3.78	73.46/0.90	57.89/3.69	85.74/6.65	63.03/1.20	70.83/0.87	76.29/0.85	67.80/1.19	48.45/1.44	79.47/0.62	56.77/0.24
Coteaching+	74.49/2.56	78.23/1.45	72.79/1.25	56.71/3.08	77.92/2.50	62.09/1.03	69.25/0.92	74.63/3.38	67.02/1.95	45.57/3.68	75.96/7.39	53.85/4.53
	$p = 0.738$	$p = 0.951$	$p = 0.683$	$p = 0.715$	$p = 0.952$	$p = 0.871$	$p = 0.977$	$p = 0.863$	$p = 0.817$	$p = 0.891$	$p = 0.834$	$p = 0.898$
MentorMix	77.58/0.70	80.80/1.05	74.15/1.13	62.77/1.14	79.52/1.02	63.85/1.02	72.58/0.56	75.25/0.24	67.51/0.10	48.54/1.06	78.86/0.83	56.73/1.02
	$p = 0.043$	$p = 0.715$	$p = 0.052$	$p = 0.058$	$p = 0.893$	$p = 0.082$	$p = 0.020$	$p = 0.998$	$p = 0.738$	$p = 0.430$	$p = 0.975$	$p = 0.481$
Bootstrap	75.17/1.43	78.80/1.05	72.78/1.13	58.58/3.00	77.55/1.79	62.11/2.10	70.30/0.66	76.45/0.74	67.55/1.24	49.24/1.18	79.65/0.56	56.51/1.46
	$p = 0.624$	$p = 0.934$	$p = 0.826$	$p = 0.398$	$p = 0.940$	$p = 0.992$	$p = 0.970$	$p = 0.362$	$p = 0.637$	$p = 0.947$	$p = 0.082$	$p = 0.662$
LRT	73.17/2.16	78.41/0.61	72.30/1.11	58.30/3.66	78.25/1.10	62.33/0.88	65.17/5.24	74.18/3.94	64.42/4.93	40.01/8.61	77.63/4.04	51.71/6.40
	$p = 0.927$	$p = 0.951$	$p = 0.928$	$p = 0.398$	$p = 0.934$	$p = 0.776$	$p = 0.992$	$p = 0.871$	$p = 0.923$	$p = 0.994$	$p = 0.809$	$p = 0.935$
Noise-adaption	77.75/0.66	80.44/1.06	74.13/1.33	62.50/0.98	79.02/2.68	63.73/1.56	72.56/1.41	75.59/0.81	67.30/1.51	50.81/2.90	79.21/0.57	56.76/0.79
	$p = 0.016$	$p = 0.846$	$p = 0.066$	$p = 0.028$	$p = 0.943$	$p = 0.096$	$p = 0.006$	$p = 0.846$	$p = 0.691$	$p = 0.075$	$p = 0.975$	$p = 0.504$
LIO	75.72/1.78	81.19/2.74	72.73/0.75	60.19/1.04	84.40/6.52	63.61/2.44	70.06/2.34	75.12/2.35	65.80/1.12	46.25/5.68	79.65/0.30	52.72/0.28
	$p = 0.825$	$p = 0.642$	$p = 0.883$	$p = 0.114$	$p = 0.479$	$p = 0.238$	$p = 1.000$	$p = 0.852$	$p = 0.992$	$p = 0.798$	$p = 0.306$	$p = 0.987$
GCE	76.98/1.49	81.52/3.26	73.68/1.74	59.08/4.09	82.89/5.96	62.67/2.38	70.83/2.21	73.25/7.41	66.94/3.21	47.58/3.32	72.31/14.53	53.49/6.24
	$p = 0.113$	$p = 0.594$	$p = 0.287$	$p = 0.350$	$p = 0.614$	$p = 0.540$	$p = 0.504$	$p = 0.793$	$p = 0.704$	$p = 0.728$	$p = 0.836$	$p = 0.848$
ELR	74.28/0.69	84.60/1.20	72.05/1.38	53.58/2.10	92.02/1.57	60.82/1.97	66.84/1.66	65.70/0.13	61.00/0.40	42.33/3.37	59.46/0.57	44.90/0.77
	$p = 0.953$	$p = 0.077$	$p = 0.899$	$p = 0.982$	$p = 0.027$	$p = 0.944$	$p = 0.998$	$p = 1.000$	$p = 0.999$	$p = 0.987$	$p = 1.000$	$p = 1.000$

Methods	Noise rate 0.6 and imbalance ratio 20x						Noise rate 0.6 and imbalance ratio 100x					
	Overall cls (%)			Rare cls (%)			Overall cls (%)			Rare cls (%)		
	Accuracy	Precision	F_1	Accuracy	Precision	F_1	Accuracy	Precision	F_1	Accuracy	Precision	F_1
Vanilla DNN	68.04/1.77	70.38/6.31	63.68/4.12	48.93/4.03	73.16/9.31	53.39/7.05	65.57/0.87	71.67/3.30	61.11/0.89	44.39/1.94	76.67/7.45	51.02/1.56
Coteaching+	61.20/4.81	58.92/5.03	54.73/4.60	33.64/6.49	55.50/9.97	39.84/8.42	50.85/7.03	37.47/4.82	41.93/4.07	10.37/14.67	16.38/7.34	15.33/6.87
	$p = 0.993$	$p = 0.997$	$p = 0.998$	$p = 0.998$	$p = 0.997$	$p = 0.997$	$p = 0.997$	$p = 1.000$	$p = 0.999$	$p = 0.998$	$p = 1.000$	$p = 1.000$
MentorMix	69.11/0.74	75.05/1.05	63.28/1.03	51.40/2.28	77.55/3.79	58.11/2.10	55.28/1.50	76.45/0.74	57.55/1.24	19.96/2.34	77.58/4.01	30.51/3.46
	$p = 0.129$	$p = 0.934$	$p = 0.826$	$p = 0.376$	$p = 0.151$	$p = 0.124$	$p = 0.998$	$p = 0.362$	$p = 0.999$	$p = 0.999$	$p = 0.082$	$p = 0.999$
Bootstrap	68.94/1.40	71.12/4.69	64.81/2.92	50.03/5.04	71.35/8.21	55.56/7.42	65.83/1.50	72.73/0.67	60.89/1.10	45.66/2.09	79.95/0.07	50.30/1.21
	$p = 0.174$	$p = 0.431$	$p = 0.328$	$p = 0.370$	$p = 0.602$	$p = 0.346$	$p = 0.316$	$p = 0.253$	$p = 0.669$	$p = 0.182$	$p = 0.184$	$p = 0.784$
LRT	64.97/2.70	70.83/2.85	61.18/4.68	48.40/3.06	78.15/1.91	54.61/2.64	55.51/5.83	57.23/6.02	47.04/5.76	26.78/11.11	59.15/10.55	29.60/10.03
	$p = 0.982$	$p = 0.453$	$p = 0.759$	$p = 0.643$	$p = 0.168$	$p = 0.380$	$p = 0.992$	$p = 0.993$	$p = 0.998$	$p = 0.994$	$p = 0.962$	$p = 0.997$
Noise-adaption	70.62/3.54	77.15/2.84	66.75/2.11	53.65/7.79	83.70/4.08	58.07/3.34	66.20/1.79	72.04/3.48	62.01/1.18	44.82/2.55	75.80/7.18	53.60/2.67
	$p = 0.130$	$p = 0.021$	$p = 0.141$	$p = 0.155$	$p = 0.010$	$p = 0.146$	$p = 0.196$	$p = 0.443$	$p = 0.120$	$p = 0.380$	$p = 0.563$	$p = 0.101$
LIO	68.54/2.97	73.43/2.67	64.63/2.40	48.73/6.64	77.28/3.51	57.39/3.04	52.54/5.26	56.90/7.90	45.92/6.74	24.53/9.29	57.12/15.47	27.23/14.41
	$p = 0.390$	$p = 0.167$	$p = 0.311$	$p = 0.517$	$p = 0.171$	$p = 0.133$	$p = 1.000$	$p = 0.997$	$p = 0.998$	$p = 0.997$	$p = 0.980$	$p = 0.993$
GCE	57.38/4.32	53.55/6.63	50.76/6.05	29.01/7.37	45.59/10.18	32.45/7.67	47.15/0.35	27.53/1.75	33.92/1.02	0/0	0/0	0/0
	$p = 0.996$	$p = 0.993$	$p = 0.992$	$p = 0.997$	$p = 0.994$	$p = 0.995$	$p = 1.000$	$p = 1.000$	$p = 1.000$	$p = 1.000$	$p = 1.000$	$p = 1.000$
ELR	68.19/1.93	68.62/5.42	62.58/3.58	43.55/4.02	66.42/9.61	48.51/5.95	63.65/0.53	60.18/5.11	58.69/1.41	36.13/1.07	50.00/10.00	40.00/2.46
	$p = 0.430$	$p = 0.727$	$p = 0.719$	$p = 0.985$	$p = 0.914$	$p = 0.917$	$p = 0.994$	$p = 0.998$	$p = 0.999$	$p = 1.000$	$p = 0.999$	$p = 1.000$

Malware != Images

- Highly Skewed and High Noise Rate in Malware dataset require different noise learning approach

- MORSE
- Semi Supervised learning method + Sample re-weighting Mechanism

Semi Supervised Learning: FixMatch

- Tailored for Image Recognition
- Partition labeled and unlabeled datasets
- Weakly augment (flip, shift) unlabeled data
- Predict labels for unlabeled data (pseudo labels)
- Calculate Both supervised, unsupervised loss
- Calculate Final Loss and update model's weights

Algorithm 1: FixMatch – the state-of-the-art semi-supervised learning algorithm.

Input: labeled dataset $\mathcal{X}$, unlabeled dataset $\mathcal{U}$, number of total training epochs K, confidence threshold τ, unsupervised loss weight λ.

Initialization: Initialize the weights $\boldsymbol{\Theta}$ for the model $f(\cdot)$ randomly.

for $k = 0, 1, 2, ..., K$ **do**

  for $iter = 1, 2, ..., num\_batches$ **do**

    From $\mathcal{X}$, draw a mini-batch $\{(\mathbf{x}_b, y_b) : b \in (1, ..., B)\}$

    From $\mathcal{U}$, draw a mini-batch $\{\mathbf{u}_b : b \in (1, ..., B_\mu)\}$

    for $b = 1, 2, ..., B_\mu$ **do**

      $\mathbf{q}_b = f(g(\mathbf{u}_b); \boldsymbol{\Theta})$

      $\widehat{q_b} = \arg\max(\mathbf{q}_b)$

    end

    $\mathcal{L}_s = \frac{1}{B} \sum_{b=1}^{B} H(y_b, f(g(\mathbf{x}_b)))$

    $\mathcal{L}_u = \frac{1}{B_\mu} \sum_{b=1}^{B_\mu} \mathbf{I}(\max(\mathbf{q}_b) > \tau) H(\widehat{q_b}, f(h(\mathbf{u}_b)))$

    $\mathcal{L} = \mathcal{L}_s + \lambda \mathcal{L}_u$

    Update the model's weights $\boldsymbol{\Theta}$ by minimizing the loss function $\mathcal{L}$

  end

end

Output: the well trained model $f(\cdot; \boldsymbol{\Theta})$.

MORSE

- FixMatch -> MORSE
- Change design to suit Malware Classification (Augmentation)
- Add Sample Re-weighting

Algorithm 2: Proposed learning algorithm. The customized and extended parts are highlighted.

Input: imbalanced noisy training dataset D, number of total training epochs K, labeled data's proportion d, starting re-weighting epoch T_d, confidence threshold τ, unsupervised loss weight λ, learning rate α.
Initialization: Initialize the weights $\boldsymbol{\Theta}$ for the model $f(\cdot)$ by using entire dataset D with only a few epochs.
for $k = 0, 1, 2, ..., K$ **do**
    Partitioning the training dataset D into labeled dataset $\mathcal{X}$ and unlabeled dataset $\mathcal{U}$: select the top $d\%$ examples with the least loss values from each given class and treat them as labeled data and the rest as unlabeled data.
    for $iter = 1, 2, ..., num\_batches$ **do**
        From $\mathcal{X}$, draw a mini-batch $\{(\mathbf{x}_b, y_b) : b \in (1, ..., B)\}$
        From $\mathcal{U}$, draw a mini-batch $\{\mathbf{u}_b : b \in (1, ..., B_\mu)\}$
        for $b = 1, 2, ..., B_\mu$ **do**
            // Use Eqn. (3) to perform weak augmentation against $\mathbf{u}_b$
            $\mathbf{q}_b = f(g(\mathbf{u}_b); \boldsymbol{\Theta})$
            $\widehat{q}_b = \arg\max(\mathbf{q}_b)$
        end
        if $k < T_d$ **then**
            // Use Eqn. (3) to perform weak augmentation against $\mathbf{x}_b$
            $\mathcal{L}_s = \frac{1}{B} \sum_{b=1}^{B} H(y_b, f(g(\mathbf{x}_b)))$
            // Use Eqn. (3) to perform strong augmentation against $\mathbf{u}_b$
            $\mathcal{L}_u = \frac{1}{B_\mu} \sum_{b=1}^{B_\mu} \mathbf{I}(\max(\mathbf{q}_b) > \tau) H(\widehat{q}_b, f(h(\mathbf{u}_b)))$
        else
            // Calculate the weight of each training sample using Eqn. (4)
            // Use Eqn. (3) to perform weak augmentation against $\mathbf{x}_b$
            $\mathcal{L}_s = \frac{1}{B} \sum_{b=1}^{B} w_b H(y_b, f(g(\mathbf{x}_b)))$
            // Use Eqn. (3) to perform strong augmentation against $\mathbf{u}_b$
            $\mathcal{L}_{u1} = \frac{1}{B_\mu} \sum_{b=1}^{B_\mu} \mathbf{I}(\max(\mathbf{q}_b) > \tau)$
            $\mathcal{L}_u = \mathcal{L}_{u1} + w_b H(\widehat{q}_b, f(h(\mathbf{u}_b)))$
        end
        $\mathcal{L} = \mathcal{L}_s + \lambda \mathcal{L}_u$
        Update the model's weights $\boldsymbol{\Theta}$ by minimizing the loss function $\mathcal{L}$
        Optional: decay the learning rate α
    end
end
Output: the well trained model $f(\cdot; \boldsymbol{\Theta})$.

Selecting a Label

- Assume a pretrained model (At initialization)
- Each Epoch we determine labeled and unlabeled data
    - Calculate loss, top d% samples with least loss -> labeled dataset

Weak & Strong augmentation

- Mask vector $[m_1, ..., m_d]^T \in \mathbb{R}^d$ Bernoulli distribution
- Replace features with from $\bar{\mathbf{x}}$
- Where, $\bar{\mathbf{x}}$ is sampled from the entire dataset

$$\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{m} + (1 - \mathbf{m}) \odot \bar{\mathbf{x}}.$$

Sample Re-Weighting

- Handle Class imbalance (weigh the loss differently depending on whether they belong to majority or minority class)

- Samples associated with minor class is assigned higher weight

$$E_{n_{y_b}} = \frac{1-\beta^{n_{y_b}}}{1-\beta}. \tag{4}$$

n_{y_b} is the number of samples in class y_b, and $\beta \in [0,1)$ is a hyperparameter. Using its inverse (*i.e.*, $w_b = \frac{1}{E_{n_{y_b}}}$), we can extend the loss shown in Equation (1) and (2) as

$$\begin{aligned} \mathcal{L}_s &= \frac{1}{B}\sum_{b=1}^{B} w_b H(y_b, f(g(\mathbf{x}_b))); \\ \mathcal{L}_u &= \frac{1}{B_\mu}\sum_{b=1}^{B_\mu} \mathbf{I}(\max(\mathbf{q}_b) > \tau) w_b H(\widehat{q}_b, f(h(\mathbf{u}_b))). \end{aligned} \tag{5}$$

Synthetic Dataset

TABLE 7: The testing performance of the models learned from MORSE and Vanilla method on the synthetic dataset.

Methods	**Noise rate 0.3 and imbalance ratio 20x**						**Noise rate 0.3 and imbalance ratio 100x**					
	Overall cls (%)			Rare cls (%)			Overall cls (%)			Rare cls (%)		
	Accuracy	Precision	F_1	Accuracy	Precision	F_1	Accuracy	Precision	F_1	Accuracy	Precision	F_1
Vanilla DNN	75.55/1.38	82.94/3.78	73.46/0.90	57.89/3.69	85.74/6.65	63.03/1.20	70.83/0.87	76.29/0.85	67.80/1.19	48.45/1.44	79.47/0.62	56.77/0.24
Vanilla DNN w re-weighting	78.87/0.30 $p = 0.000$	82.64/0.88 $p = 0.335$	77.71/0.44 $p = 0.000$	68.59/2.29 $p = 0.000$	80.75/2.44 $p = 0.878$	69.63/1.03 $p = 0.000$	72.85/1.79 $p = 0.083$	75.97/0.98 $p = 0.656$	69.86/1.74 $p = 0.049$	56.81/4.63 $p = 0.009$	76.44/1.89 $p = 0.988$	60.88/2.64 $p = 0.008$
Our method w/o re-weighting	79.64/0.83 $p = 0.000$	87.81/1.68 $p = 0.011$	77.61/1.01 $p = 0.000$	64.85/2.62 $p = 0.001$	94.46/2.31 $p = 0.009$	70.00/2.70 $p = 0.000$	71.73/2.24 $p = 0.138$	73.38/4.94 $p = 0.900$	65.87/3.16 $p = 0.891$	46.84/3.98 $p = 0.752$	72.99/9.19 $p = 0.917$	50.54/5.68 $p = 0.972$
Our method	79.73/1.85 $p = 0.008$	87.11/1.82 $p = 0.039$	79.11/1.30 $p = 0.000$	67.47/3.17 $p = 0.006$	88.45/4.63 $p = 0.182$	70.76/3.42 $p = 0.002$	73.47/0.91 $p = 0.001$	78.97/2.11 $p = 0.027$	70.42/1.02 $p = 0.007$	54.40/7.97 $p = 0.092$	82.59/3.85 $p = 0.071$	62.30/5.55 $p = 0.040$

Methods	**Noise rate 0.6 and imbalance ratio 20x**						**Noise rate 0.6 and imbalance ratio 100x**					
	Overall cls (%)			Rare cls (%)			Overall cls (%)			Rare cls (%)		
	Accuracy	Precision	F_1	Accuracy	Precision	F_1	Accuracy	Precision	F_1	Accuracy	Precision	F_1
Vanilla DNN	68.04/1.77	70.38/6.31	63.68/4.12	48.93/4.03	73.16/9.31	53.39/7.05	65.57/0.87	71.67/3.30	61.11/0.89	44.39/1.94	76.67/7.45	51.02/1.56
Vanilla DNN w re-weighting	68.01/1.52 $p = 0.464$	70.65/2.95 $p = 0.448$	65.52/2.68 $p = 0.184$	58.50/2.59 $p = 0.011$	67.70/5.18 $p = 0.960$	57.18/3.31 $p = 0.106$	66.84/2.36 $p = 0.137$	67.22/2.28 $p = 0.979$	64.34/2.33 $p = 0.021$	55.85/4.36 $p = 0.001$	61.15/4.13 $p = 0.996$	56.36/4.29 $p = 0.037$
Our method w/o re-weighting	72.82/2.86 $p = 0.012$	77.24/1.69 $p = 0.050$	69.42/2.61 $p = 0.038$	54.09/3.96 $p = 0.051$	81.00/1.34 $p = 0.061$	60.26/2.85 $p = 0.041$	64.85/4.83 $p = 0.624$	65.85/5.03 $p = 0.927$	58.93/2.81 $p = 0.988$	38.65/9.12 $p = 0.895$	66.67/8.43 $p = 0.898$	42.38/6.12 $p = 0.987$
Our method	76.20/0.59 $p = 0.000$	79.45/1.69 $p = 0.016$	72.90/1.17 $p = 0.003$	61.21/1.00 $p = 0.000$	80.45/0.99 $p = 0.066$	64.25/0.20 $p = 0.009$	72.05/1.94 $p = 0.000$	74.97/3.41 $p = 0.005$	68.03/1.01 $p = 0.000$	53.82/4.98 $p = 0.005$	76.60/7.48 $p = 0.891$	58.00/3.80 $p = 0.003$

PE and Android Dataset

TABLE 8: MORSE vs. Vanilla method on the PE dataset.

Methods	Average (%)			Class-11 (%)		
	Accuracy	Precision	F_1	Accuracy	Precision	F_1
Vanilla DNN	93.08/0.25	93.65/0.51	92.57/0.36	44.33/3.78	96.34/4.22	60.48/3.22
Vanilla DNN w re-weighting	93.82/0.29 $p = 0.003$	94.01/0.30 $p = 0.102$	93.50/0.34 $p = 0.001$	55.67/2.98 $p = 0.001$	89.71/2.30 $p = 0.974$	68.61/1.70 $p = 0.001$
Our method w/o re-weighting	92.50/0.18 $p = 0.980$	93.35/0.28 $p = 0.924$	91.33/0.58 $p = 0.992$	36.50/2.75 $p = 0.995$	99.51/1.10 $p = 0.089$	51.07/5.05 $p = 0.979$
Our method	94.15/0.43 $p = 0.003$	94.14/0.57 $p = 0.061$	93.91/0.67 $p = 0.000$	65.33/4.20 $p = 0.000$	90.18/2.08 $p = 0.984$	76.74/1.22 $p = 0.000$

TABLE 9: MORSE vs. Vanilla on the Android dataset.

Methods	Average (%)			Class-6 (%)		
	Accuracy	Precision	F_1	Accuracy	Precision	F_1
Vanilla DNN	73.00/0.35	78.65/2.21	69.96/0.54	2.67/5.53	19.79/34.31	4.63/9.52
Vanilla DNN w re-weighting	77.32/1.72 $p = 0.002$	82.20/3.73 $p = 0.049$	75.68/2.58 $p = 0.003$	20.83/7.99 $p = 0.007$	80.39/9.40 $p = 0.003$	32.03/8.99 $p = 0.005$
Our method w/o re-weighting	74.58/0.86 $p = 0.004$	77.33/4.47 $p = 0.691$	71.00/0.96 $p = 0.015$	5.17/7.31 $p = 0.309$	32.35/45.79 $p = 0.347$	8.91/12.60 $p = 0.311$
Our method	79.76/1.02 $p = 0.000$	80.37/1.99 $p = 0.088$	78.72/1.22 $p = 0.000$	27.67/8.01 $p = 0.000$	58.54/18.90 $p = 0.039$	35.80/7.88 $p = 0.000$