A Novel Network Traffic Classification Approach via Discriminative Feature Learning

Lixin Zhao, Lijun Cai, Aimin Yu, Zhen Xu, and Dan Meng

Institute of Information Engineering, Chinese Academy of Sciences

Traffic Classification

Traffic Classification: Automatically associating network traffic with the generating application.

QoS control

Network Monitoring

Bandwidth Optimization

Network Security

.....

.....

Existing methods

- Port-based method
  - Simple and fast, but can not deal with NAT, port obfuscation, random ports assignments……
- Payload-based method (Deep Packet Inspection)
  - Accurate, but can not deal with encryption and may have privacy issues.
- Statistical-based method
  - Encryption independent, but feature design depends on expert knowledge and the handcrafted features are not discriminative in traffic classification for different specific applications.
- Deep learning based method
  - Automated feature learning and classification accuracy have improved, but ……..

Problem & challenge

- The within-class diversity and between-class similarity are still two big challenges for application-level traffic classification.
  - E.g. an application may use multiple protocols to achieve different functions such as login, file transfer, update etc. Meanwhile, different applications with similar functionality often generate flows with similar patterns.

Contributions

- We combine metric learning techniques with deep learning models to learn more discriminative features for network traffic classification.
- We provide a thorough evaluation and comparison with state-of-the-art methods on three datasets.

Approach Overview

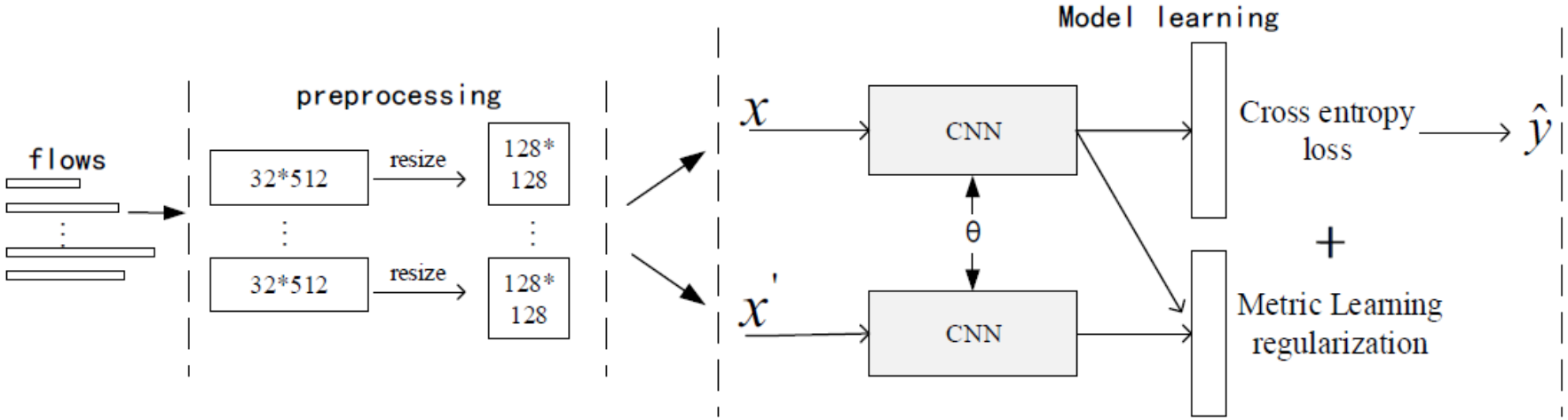


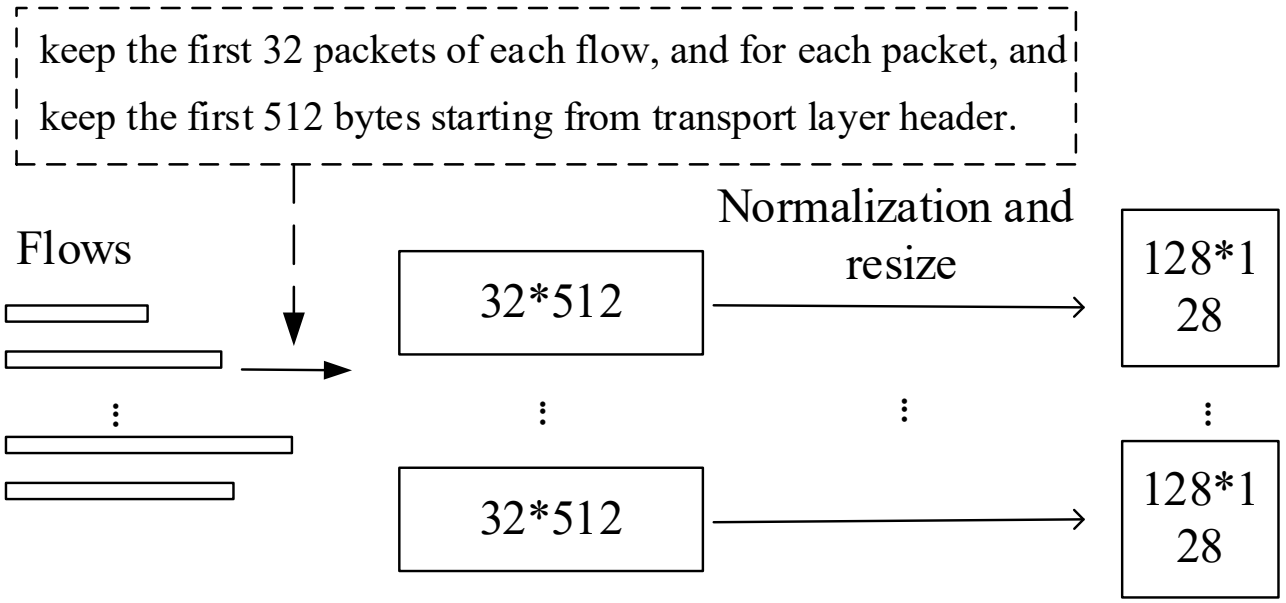
Illustration of the overall architecture of the proposed approach.

Two core modules:

- Preprocessing
- Model Learning

Proposed Approach

- **Preprocessing**



Proposed Approach

Model Learning

- New Objective function:

$$J = \min(J_1(X, Y, \theta_{ce}) + \lambda J_2(X, \theta_{ml}))$$

(1) Traditional cross entropy loss term:

$$J_1(X, Y, \theta_{ce}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{k}\mathbf{y}_i^n \log\frac{e^{\mathbf{W}_i^T f(\mathbf{x}^n)+\mathbf{b}_i}}{\sum_{j=1}^{k} e^{\mathbf{W}_j^T f(\mathbf{x}^n)+\mathbf{b}_j}}$$

(2) Metric Learning Regularization Term (contrastive embedding):

$$J_2(X, Y, \theta_{ml}) = \sum_{i,j}\overline{y}D^2(\mathbf{x}^i, \mathbf{x}^j) + (1-\overline{y})h(D(\mathbf{x}^i, \mathbf{x}^j) - m + 1)^2$$

Proposed Approach

- Contrastive Pairs Generation
  - Given N training samples, we can produce C_N^2 sample pairs.

Algorithm 1 Contrastive Pairs Generation In One Epoch

Input:

Training samples: $X = \{x^1, x^2, ..., x^n\}$;

Training sample labels: $Y = \{y^1, y^2, ..., y^n\}$;

Output:

Contrastive pairs: $\{(a^1, b^1, label_1), ..., (a^m, b^m, label_m)\}$

Get the label set of Y: $label\_set \leftarrow set(Y)$

Initialization: $class\_center \leftarrow \{\}$ and $pairs \leftarrow \{\}$

for l in $label\_set$ **do**

  $M \leftarrow$ Number of samples with label l

  $class\_center[l] \leftarrow \frac{1}{M} \sum_{i=1}^{n} I(y^i == l) x^i$

end for

for each x^i in X **do**

  $label' \leftarrow$ randomly select from $\{0, 1\}$

  if $label' == 1$ **then**

    $a \leftarrow x^i$

    $b \leftarrow class\_center[y^i]$

  else

    $y' \leftarrow$ randomly select a label that is not equal to y^i

    $a \leftarrow$ randomly select a sample with label y'

    $b \leftarrow class\_center[y^i]$

  end if

  put $(a, b, label')$ into $pairs$

By replacing the contrastive sample $\mathbf{x}^j$ in pair $(\mathbf{x}^i, \mathbf{x}^j)$ to $\text{center}(\mathbf{x}^j)$, the sample pair space drops from C_N^2 to $k \times N$

Datasets

- Organization Dataset (OD)
  - Collected traffic data from the machines of a working group with 20 people in an organization using a process-oriented monitor tool.
- Malware Dataset (MD)
  - Public Malware traffic data maintained by Malware Capture Facility Project (https://www.stratosphereips.org/datasets-malware)
- Synthetic Dataset (SD)
  - The combination of the above two datasets.

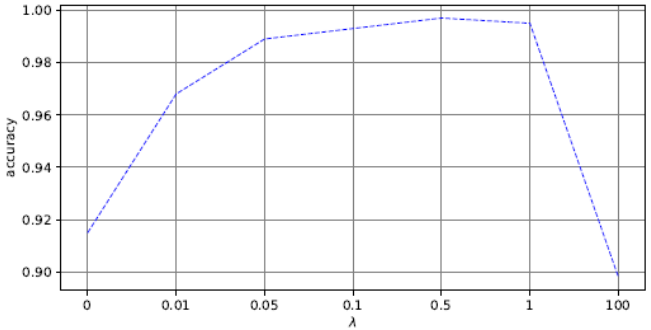
OD	Flows	MD	Flows
Browser	14026	Artemis	12654
360 Internet Security	10356	Sennoma	10003
Tencent PC Manager	10965	Dynamer	11777
Sogou pinyin	10084	Tinba	12862
Wechat	12168	Ursnif	11000
Utorrent	9794	CCleaner	11538
Kugou Music	11769	Miner	9864
PPTV	12846	Downloader	10239
Kuwo Music	10319	CoreBot	9892
Iqiyi	12999	Dridex	11319
Total	115326	Total	111148

Datasets

- Dataset settings
  - Organization Dataset (OD)
    - 70% for training, 10% for validation and 20% for testing.
  - Malware Dataset (MD)
    - 80% for training and 20% for testing.
  - Synthetic Dataset (SD)
    - 80% for training and 20% for testing.

Implementation details

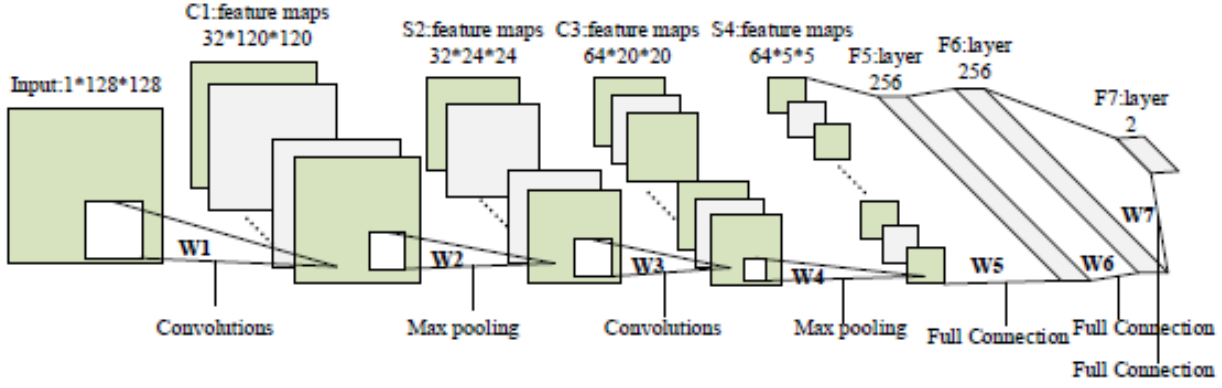
Parameters determination



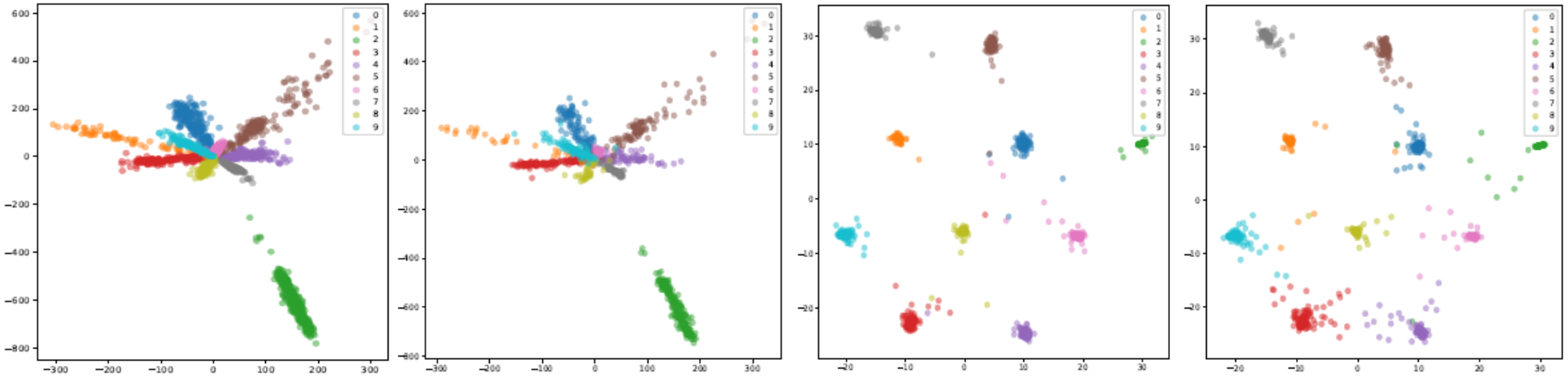
Overall accuracy on the validation set of OD. The metric learning regularization parameter λ varies from the set of {0.01, 0.05, 0.1, 0.5, 1, 100} with 200 iterations.

- Set λ to 0.5 and keep it fixed for the rest two datasets.
- For CNN model training:
  - the learning rate is set to 0.001 for all the layers
  - the L2 weight decay parameter is set to 0.0005 for preventing over-fitting.

CNN architecture



Evaluation Results and Comparisons



Comparison of the distribution of deeply learned features in (a) training set (b) testing set of OD, both under the supervision of cross entropy loss only, and (c) training set (d) testing set of OD, both under the joint supervision of cross entropy loss and metric learning regularization term. We use 70%/20% training/testing splits. The points with different colors denote features from different classes.

Evaluation Results and Comparisons

	$\lambda = 0$			Deep Packet [12]			$\lambda = 0.5$		
	pre.	rec.	acc.	pre.	rec.	acc.	pre.	rec.	acc.
Brow.	0.893	0.917		0.863	0.902		**0.995**	**0.996**	
360	0.944	0.879		0.966	0.901		**0.996**	**0.996**	
Tenc.	0.907	0.915		0.920	0.922		**0.996**	**0.997**	
Sogo.	0.908	0.903		0.958	0.894		**0.996**	**0.995**	
Wech.	0.919	0.934	0.908	0.926	0.945	0.912	**0.998**	**0.996**	**0.997**
Utor.	0.961	0.967		0.970	0.972		**0.998**	**0.999**	
Kugo.	0.842	0.817		0.873	0.869		**0.996**	**0.998**	
PPTV	0.890	0.904		0.851	0.899		**0.998**	**0.997**	
Kuwo	0.826	0.845		0.838	0.818		**0.995**	**0.996**	
Iqiyi	0.992	0.989		0.983	0.991		**0.999**	**0.999**	

(a)

	$\lambda = 0$			Deep Packet [12]			$\lambda = 0.5$		
	pre.	rec.	acc.	pre.	rec.	acc.	pre.	rec.	acc.
Arte.	0.957	0.901		0.905	0.911		**1**	**0.999**	
Senn.	0.922	0.949		0.967	0.981		**0.999**	**0.999**	
Dyna.	0.944	0.967		0.934	0.954		**0.999**	**1**	
Tinba	0.949	0.902		0.957	0.898		**0.999**	**0.999**	
Ursn.	0.931	0.973	0.917	0.971	0.945	0.919	**0.999**	**0.999**	**0.992**
CCle.	0.948	0.948		0.962	0.912		**1**	**0.999**	
Miner	0.942	0.958		0.888	0.969		**0.999**	**0.999**	
Down.	0.806	0.797		0.811	0.819		**0.961**	**0.958**	
Core.	0.950	0.989		0.982	0.986		**0.999**	**1**	
Drid.	0.811	0.799		0.822	0.835		**0.957**	**0.963**	

(b)

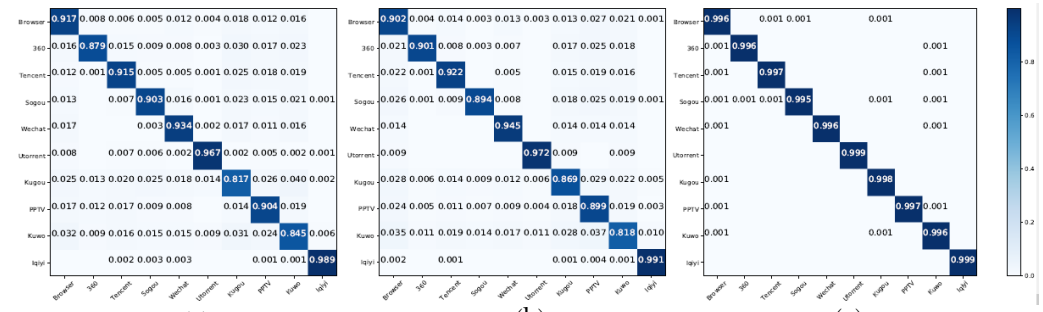
	$\lambda = 0$			Deep Packet [12]			$\lambda = 0.5$		
	pre.	rec.	acc.	pre.	rec.	acc.	pre.	rec.	acc.
Brow.	0.824	0.862		0.837	0.844		**0.992**	**0.987**	
360	0.955	0.876		0.960	0.898		**0.990**	**0.985**	
Tenc.	0.906	0.916		0.907	0.913		**0.989**	**0.994**	
Sogo.	0.906	0.899		0.927	0.884		**0.990**	**0.993**	
Wech.	0.927	0.888		0.930	0.944		**0.992**	**0.982**	
Utor.	0.960	0.966		0.958	0.958		**0.994**	**0.996**	
Kugo.	0.826	0.813		0.837	0.860		**0.990**	**0.997**	
PPTV	0.890	0.898		0.841	0.880		**0.992**	**0.985**	
Kuwo	0.820	0.839		0.826	0.831		**0.991**	**0.984**	
Iqiyi	0.986	0.987	0.901	0.978	0.988	0.897	**0.996**	**0.999**	**0.987**
Arte.	0.972	0.900		0.894	0.893		**0.995**	**0.989**	
Senn.	0.921	0.943		0.956	0.951		**0.992**	**0.995**	
Dyna.	0.936	0.965		0.915	0.933		**0.994**	**0.992**	
Tinba	0.936	0.874		0.955	0.891		**0.993**	**0.987**	
Ursn.	0.917	0.964		0.923	0.915		**0.992**	**0.996**	
CCle.	0.943	0.947		0.949	0.908		**0.992**	**0.996**	
Miner	0.930	0.968		0.874	0.929		**0.991**	**0.996**	
Down.	0.765	0.778		0.785	0.759		**0.939**	**0.953**	
Core.	0.980	0.981		0.960	0.972		**0.993**	**1**	
Drid.	0.758	0.788		0.762	0.798		**0.949**	**0.946**	

(c)

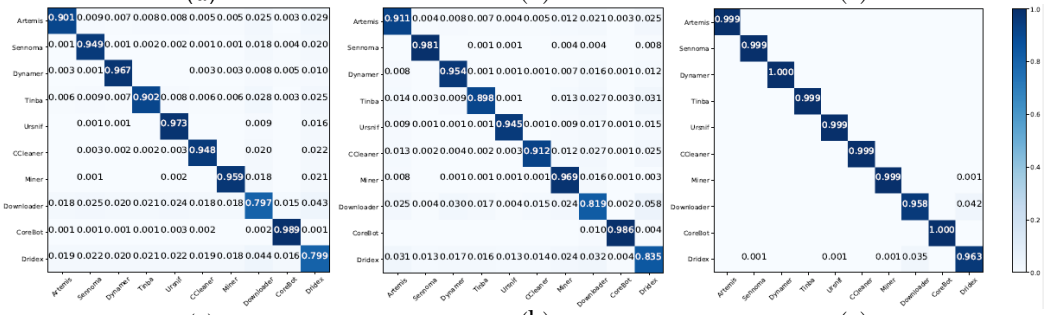
Classification results comparison on (a) OD (b) MD (c) SD. $\lambda = 0.5$ denotes our proposed approach, $\lambda = 0$ denotes that features are learned using only cross entropy loss term, and 'Deep Packet' is the method proposed in related work.

Evaluation Results and Comparisons

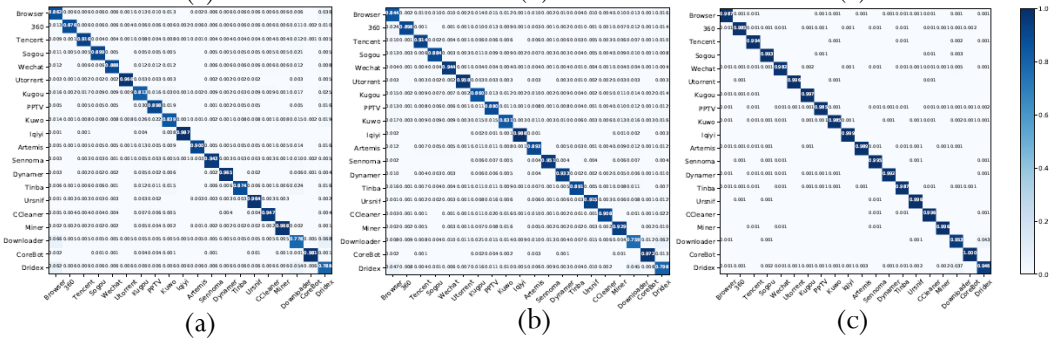
OD:



MD:



SD:



Confusion matrices of three datasets using the following methods: (a) $\lambda = 0$: cross entropy loss only (b) Deep Packet (c) $\lambda = 0.5$: combination of cross entropy loss and metric learning regularization term.

Conclusion and Future work

- Conclusion
  - We explore the possibility of integrating metric learning into deep learning models to deal with the problems of within-class diversity and between-class similarity when performing application-level traffic classification.
  - Extensive experiments have been conducted on three traffic datasets to evaluate the effectiveness of our method.
- Future work
  - The problem of network traffic classification in an *open world* setting, where traffic from previously unknown class (e.g. novel application) may exist in the testing phase.

Thanks !