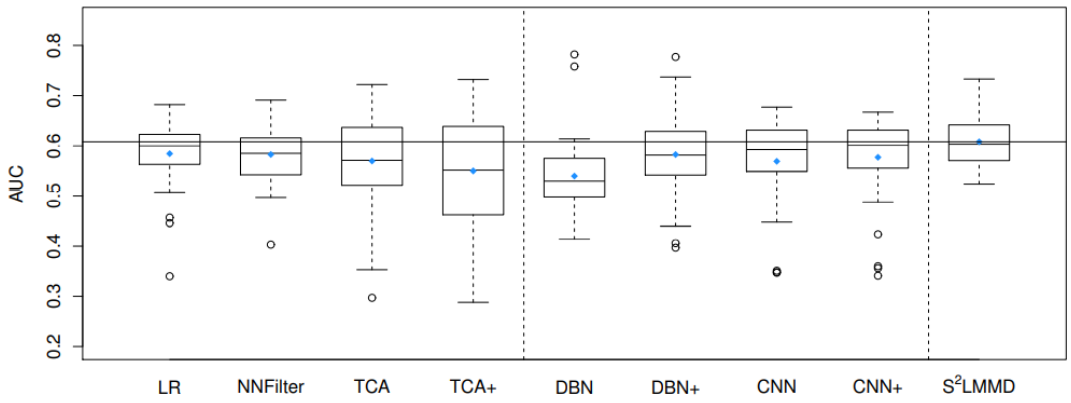
S^{2}LMMD: Cross-Project Software Defect Prediction via Statement Semantic Learning and Maximum Mean Discrepancy

Wangshu Liu[†‡*], Yongteng Zhu[†], Xiang Chen[§‡], Qing Gu[‡], Xingya Wang[†‡], Shenkai Gu[†]
[†]School of Computer Science and Technology, Nanjing Tech University, Nanjing, China
[‡]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[§]School of Information Science and Technology, Nantong University, Nantong, China

RQ1: Does our proposed method have a high and stable prediction performance when compared with the state-of-the-art CPDP methods?



To answer RQ1, we compare our proposed method with eight state-of-the-art baselines (in Section IV-B) to investigate whether S2LMMD can achieve better performance or not. Fig shows the box plot among these baseline methods and our proposed method in terms of AUC on ten projects. In this figure, the blue dot denotes their mean value and the solid line is drawn for a more intuitive display of our proposed method. From Fig, compared with traditional methods, our method can outperform these four baselines, especially for TCA and the improved version TCA+. Specifically, our proposed method can improve the AUC by more than 10.57% (TCA+), 6.78% (TCA), 4.39% (NNFilter), and 4.09% (LR) on average, and is individually ranked first in maximum, minimum, and median among 4×9 CPDP tasks. Besides, to verify the stability of these methods, we calculate the inter-quartile range (IQR) between each other and find our method can slightly outperform LR but better than other baselines. Although LR has the lowest IQR, its results among all trails contain the most outliers. Some similar outliers exist in NNFilter and TCA as well. These results show that our method can significantly improve the CPDP performance by exploiting SLT for encoding. On the contrary, the traditional methods only take advantage of static manual metrics, which may ignore the semantic and structure information of the source code itself, leads to poor performance on the CPDP task.

Compared with the deep learning-based methods, our proposed method S^{2}LMMD can still outperform these baselines, especially for DBN. Specifically, our proposed method can improve the AUC by more than 12.6% (DBN), 6.89% (CNN), 5.41% (CNN+), and 4.41% (DBN+) on average, and achieves the best once again in minimum and median among 4×9 CPDP tasks. Besides, only our proposed method has no outliers, and the IQR

reaches the smallest value, which revealed that our method appears to be more stable than the others. There are two reasons for the above results: (1) In the embedding layer, a finer semantic unit SLT has been used for encoding instead of the trivial AST, which can capture a refined semantic and structural representation. (2) For CPDP, unlike baselines, a useful transfer loss MMD is embedded in our loss function, which also has been proven to be effective to alleviate the data distribution difference in RQ2.

Group	Method	Avg.	Win/Tie/Loss	p-value
Traditional Methods	LR	0.584	24/0/12	0.02516
	NNFilter	0.583	24/1/11	0.01248
	TCA	0.569	25/2/9	0.02052
	TCA+	0.550	23/2/11	0.01317
Deep Learning	DBN	0.540	31/1/4	0.00013
	DBN+	0.582	22/0/14	0.03665
	CNN	0.570	22/1/13	0.03390
	CNN+	0.577	23/1/12	0.04142
Ours	S^2LMMD	**0.608**	-	-

Furthermore, Table provides statistical information of the comparison results, such as Win/Tie/Loss and p-value of the Wilcoxon test. From this table, consistent with Fig. 4, we can find that our proposed method can beat the other eight baselines at least 22 times in 36 trials (about a two-thirds probability). Moreover, our method is statistically significantly different from other methods, since all the p-values are less than 0.05.

Summary for RQ1: S2LMMD can achieve a better and stable performance compared with eight state-of-the-art base lines by combing semantic learning and transfer training. The Wilcoxon signed-rank test further illustrates the significant difference between our proposed method and the comparative baselines.

RQ2: Which component of our proposed method plays a more critical role in CPDP? Whether the joint effect of both semantic learning and transfer training is helpful for our proposed method?

	Full-AST	Bi-LSTM	-TF	-MMD	Ours
ant	0.671	0.673	0.673	0.633	**0.693**
camel[#]	0.560	0.566	0.568	0.530	**0.578**
jedit	0.639	**0.664**	0.617	0.633	0.658
log4j	0.513	0.528	0.539	0.557	**0.562**
lucene	**0.639**	0.609	0.608	0.559	0.621
poi[#]	0.569	0.589	0.584	0.569	**0.609**
synapse	0.604	0.605	0.603	0.597	**0.633**
velocity	0.552	0.575	0.557	0.545	**0.603**
xalan[#]	0.544	0.559	0.575	0.553	**0.578**
xerces[#]	0.526	0.524	0.537	**0.567**	0.541
Avg.	0.582	0.589	0.586	0.574	**0.608**
p-value	0.00032	0.00031	0.00033	0.00118	-

[#] the source projects do not contain themselves as training data

To answer RQ2, we decompose our proposed method into multiple components and aim to investigate which of them plays the key role in defect prediction. At the sematic

learning stage, we design two comparative methods “Full-AST” and “Bi-LSTM”. The former replaces SLT with full AST for encoding, and the latter adopts another popular recurrent neural network (i.e., Bi-LSTM) for sematic learning. At the transfer training stage, we implement another two comparative methods “-TF” and “-MMD”. The “-TF” method trains model without traditional static program features, while the “-MMD” method abandons the transfer loss $\mathcal{L}_{MMD}$ from Formula for training. Table shows the comparison results in terms of average AUC among these comparative methods and our proposed method on ten projects.

For semantic learning, from Table IV, our method outperforms “Full-AST”, “Bi-LSTM” by 4.0% and 2.7% on average. Specifically, among them, compared with “Full-AST”, our method has improved by more than 5% in four out of ten datasets (i.e., log4j, poi, velocity, and xalan), while for “Bi-LSTM”, it appears only one time on log4j dataset. These results imply that SLT, compared with full AST, has a higher impact on CPDP, since SLT produces a better representation of semantic and structural information for fine granularity, while the impact of replacing Bi-GRU with Bi-LSTM is relatively weak. Furthermore, we conduct the Wilcoxon signed-rank test. Both the p-values with “Full-AST” and “Bi-LSTM” are far less than 0.05, which indicates that our proposed method has a significant difference in changing the semantic learning components. Similarly, for transfer training, our method can still outperform “-TF”, “-MMD” by 3.2% and 5.4% on average as well. Specifically, among them, compared with “-MMD”, our method has improved by more than 5% in six out of ten datasets (except jedit, log4j, xalan, and xerces), while for the method “-TF”, it appears only two times on jedit and velocity datasets. These results demonstrate that the transfer loss function MMD plays an important role in alleviating the distribution difference between the source project and the target project. Furthermore, the p-values by the Wilcoxon signed-rank test are still far less than 0.05, which indicates that our proposed method has a statistically significant difference with “-TF” and “-MMD”.

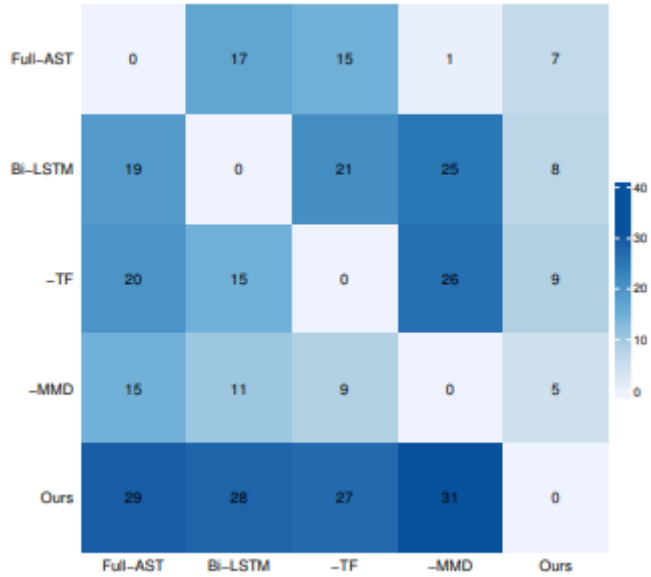


Fig shows a win-loss heat map among the above five methods for 4×9 CPDP tasks. The darker the color in the figure, the more the method wins in our experiment. From the last row of this figure, we can find that our proposed method has the darkest color and wins at

least 27 out of 36 CPDP tasks, which illustrates the combined effect of all components is the key to success. Besides, considering each comparative method, "-MMD" has the lightest color and loses more than half in 36 trials. Consistent with Table IV, MMD is more effective for CPDP, while the other components are equivalent to each other.

Summary for RQ2: For semantic learning, SLT has a higher impact on CPDP, while MMD is more effective for transfer training. There is no significant difference between the individual components, thus the joint effect of all components in our proposed method plays a more critical role in CPDP.