Table 5: Precision, recall and F1 comparison of all.

Approaches	BigCloneBench			OJClone		
	P	R	F1	P	R	F1
CDLH	0.92	0.74	0.82	0.47	0.73	0.57
DLC+H	0.24	0.60	0.35	0.07	0.01	0.02
P-CNN+H	0.20	0.87	0.33	0.07	0.04	0.05
Doc2Vec+H	0.25	0.96	0.39	0.07	0.03	0.05
CDLH+LSH	0.64	0.45	0.53	0.25	0.72	0.37
DLC+LSH	0.14	0.68	0.24	0.07	0.33	0.11
P-CNN+LSH	0.13	0.00	0.01	0.07	0.02	0.03
Doc2Vec+LSH	0.18	0.17	0.17	0.07	0.36	0.11

Table 6: Comparison of F1 values with respect to various clone types for approaches using learning to hash techniques on BigCloneBench, the best performed across each clone type is emphasized with boldface.

Clone types	CDLH	DLC +H	P-CNN +H	Doc2Vec +H
Type-1	**1.00**	**1.00**	0.95	**1.00**
Type-2	**1.00**	0.39	0.33	0.40
Strong Type-3	**0.94**	0.32	0.82	0.40
Mid Type-3	**0.88**	0.32	0.75	0.40
Type-4	**0.81**	0.34	0.33	0.39

To eliminate the effect of learning to hash has on representations extracted by CDLH, and to show that without learning to hash CDLH is still a good way to extract binary hash codes for programming language, we use unsupervised data-independent hashing LSH instead of learning to hash method used by CDLH, and compare following methods:

- representations extracted by CDLH, and LSH, we name it CDLH+LSH;
- representations extracted by DLC, and LSH, we name it DLC+LSH;
- representations extracted by P-CNN, and LSH, we name it P-CNN+LSH;
- representations extracted by Doc2Vec, and LSH, we name it Doc2Vec+LSH;

We treat above approaches as two groups according to hashing techniques they used.

Table 5 shows the precision, recall and F1 comparison of above approaches. We can see that CDLH+H and CDLH+LSH achieve highest F1 values in each group, which validates that representations extracted by AST-based LSTM can obtain good performance despite of the hashing techniques used. Approaches using representations extracted by DLC, P-CNN and Doc2Vec can either obtain low precision or low recall no matter using learning to hash technique or not, suggesting that representations extracted by them are less distinguished compared with representations extracted by AST-based LSTM. Approaches using LSH rather than learning to hash can obtain relatively inferior performance. Since LSH is an unsupervised data-independent hashing method, it can not learn task-specific pattern from training data and lack of guidance of supervised information. Even in this case, the representations extracted by CDLH still outperform other representations, which confirms the effectiveness of representations extracted by CDLH.

Table 7: Comparison of F1 values with respect to various clone types for approaches using LSH on BigCloneBench, the best performed across each clone type is emphasized with boldface.

Clone types	CDLH +LSH	DLC +LSH	P-CNN +LSH	Doc2Vec +LSH
Type-1	**1.00**	**1.00**	0.00	0.73
Type-2	**1.00**	0.25	0.03	0.28
Strong Type-3	**0.77**	0.01	0.01	0.27
Mid Type-3	**0.69**	0.23	0.01	0.25
Type-4	**0.52**	0.23	0.01	0.16

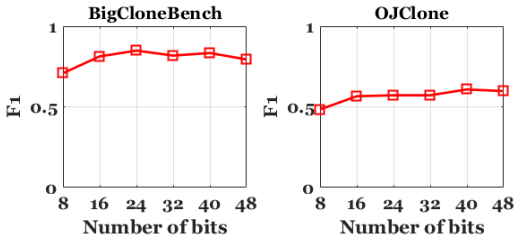

Figure 3: The influence of hash code length on CDLH.

F1 values with respect to various clone types for approaches using learning to hash techniques and LSH are listed in Table 6 and Table 7 respectively. As observed, CDLH and CDLH+LSH outperform other approaches in each group, they obtain high F1 values for Type-4 clone, together with other clone types. This further validates the effectiveness of the representations extraction layer of CDLH.

4.4 Sensitivity to Hyper-Parameter

In previous experimental settings, we use code length 32 for learned binary hash codes. Now we study the influence of different length of hash code on clone detection performance of CDLH, measured by F1 value. Figure 3 shows the F1 values of CDLH with respect to different code lengths, which ranges from 8 to 48. We can find that the overall performance of CDLH is not sensitive to hash code length in this range.

5 Conclusion

In this paper, we address the software functional clone detection problem by learning supervised deep features. We formulate the clone detection as a supervised learning to hash problem and propose an end-to-end deep feature learning framework called CDLH for functional clone detection, where an AST-based LSTM is used to exploit the lexical and syntactical information and the supervision on the functional similarity is used to guide the feature learning. Experiments on software clone detection benchmarks indicate that the CDLH approach is effective and outperforms the state-of-the-art approaches in software functional clone detection.