APPENDIX

A SLR METHODOLOGY

A.1 Literature Search and Selection

To collect DL related papers in SE, we identified a search string including several DL related terms frequently appeared in SE papers that make use of DL. We then refined the search string by checking the title and abstract of a small number of relevant papers. After that, we used logical ORs to combine these terms, and the search string is:

("deep learning" OR "neural" OR "Intelligence" OR "reinforcement" OR "NN" OR "nn")

We specified the range the papers are published later: 2006- 2020. Following previous studies [7, 8, 11], we selected 32 widely read journals (10) and conferences (22) listed in Table 13 to conduct a comprehensive literature review. We run the search string on three databases (i.e., ACM digital library [7], IEEE Explore [8], and Web of Science [9]) looking for publications in the 32 publication venues whose meta data (including title, abstract and keywords) satisfies the search string. Our preliminary results search returns 824 relevant papers.

A.2 Literature Filtering

After retrieving studies that match our search string, it is necessary to filter unqualified studies, such as studies with insufficient contents or missing information. To achieve this, we applied our inclusion and exclusion criteria to determine the quality of candidate studies for ensuring that every study we kept implemented and evaluated a full DL approaches to tackle SE tasks.

The following inclusion and exclusion criteria are used:

✔ The paper must be written in English.
✔ The paper must adopt DL techniques to address SE problems.
✔ The length of paper must not be less than 6 pages.
✘ Books, keynote records, non-published manuscripts, and grey literature are dropped.
✘ If a conference paper has an extended journal version, the conference version is excluded.

The literature filtering consisted of three steps and was performed by three researchers with rich experience in software engineering. We first discarded duplicate papers from our preliminary results. After that, we applied the inclusion/exclusion criteria by reading their title, abstract and keywords, and narrow the candidate set to 271 studies. After looking through these 271 studies to ensure their relevance (i.e., if a study used deep learning techniques to address practical problems in software engineering, it will be kept as a primary study.), we retained 250 studies. We invited a third researcher to double-check the inconsistencies between the two researchers in the last two steps to minimize human error.

A.3 Data Extraction and Collection

After removing the irrelevant and duplicated papers, we extracted and recorded the essential data and performed overall analysis for answering our four RQs. Table 14 described the detailed information being extracted and collected from 250 primary studies, where the column *'ExtractedDataItems'* lists the related data items that would be extracted from each primary study, and the column *'RQ'* denotes the related research questions to be answered by the extracted data items on the right. To avoid making mistakes in data collection, two researchers extracted these data items from primary studies together and then another researcher double checked the results to make sure of the correctness of the extracted data.

[7] https://dl.acm.org
[8] https://ieeexplore.ieee.org
[9] http://apps.webofknowledge.com

Table 13. Publication venues for manual search

No.	Acronym	Full name	No.	Acronym	Full name
1.	ICSE	ACM/IEEE International Conference on Software Engineering	23.	TSE	IEEE Transactions on Software Engineering
2.	ASE	IEEE/ACM International Conference Automated Software Engineering	24.	TOSEM	ACM Transactions on Software Engineering and Methodology
3.	ESEC/FSE	ACM SIGSOFT Symposium on the Foundation of Software Engineering/European Software Engineering Conference	25.	ESE	Empirical Software Engineering
4.	ICSME	IEEE International Conference on Software Maintenance and Evolution	26.	JSS	Journal of Systems and Software
5.	ICPC	IEEE International Conference on Program Comprehension	27.	IST	Information and Software Systems
6.	ESEM	ACM/IEEE International Symposium on Empirical Software Engineering and Measurement	28.	ASEJ	Automated Software Engineering
7.	RE	IEEE International Conference on Requirements Engineering	29.	IETS	IET Software
8.	MSR	IEEE Working Conference on Mining Software Repositories	30.	STVR	Software Testing, Verification and Reliability
9.	ISSTA	ACM SIGSOFT International Symposium on Software Testing and Analysis	31.	JSEP	Journal of Software: Evolution and Process
10.	SANER	IEEE International Conference on Software Analysis, Evolution and Reengineering	32.	SQJ	Software Quality Journal
11.	ICST	IEEE International Conference on Software Testing, Verification and Validation			
12.	ISSRE	IEEE International Symposium on Software Reliability Engineering			
13.	COMPSAC	IEEE International Computer Software and Applications Conference			
14.	QRS	IEEE International Conference on Software Quality, Reliability and Security			
15.	SPLC	Software Product Line Conferences			
16.	OOPSLA	ACM SIGPLAN international conference on Object oriented programming systems languages	applications		
17.	PLDI	ACM SIGPLAN Conference on Programming Language Design and Implementation			
18.	AAAI	Proceedings of the AAAI Conference on Artificial Intelligence			
19.	ICML	The International Conference on Machine Learning			
20.	ICLR	International Conference on Learning Representations			
21.	NeurIPS	Annual Conference on Neural Information Processing Systems			
22.	IJCAI	International Joint Conference on Artificial Intelligence			

B BA: WHAT ARE THE TRENDS IN THE PRIMARY STUDIES ON USE OF DL IN SE?

We analyzed the basic information of primary studies to comprehend the trend of DL techniques used in SE in terms of the publication date, publication venues, and main contribution types of primary studies.

Table 14. Data Collection for Research Questions

RQs	**Extracted data items**
RQ1	Basic information of each primary study (i.e., title, publication year, authors, publication venue)
RQ1	The type of main contribution in each study (e.g., empirical study, case study, survey, or algorithm)
RQ2	DL techniques used in each study
RQ2	Whether and how the authors describe the rationale behind techniques selection
RQ3	Dataset source (e.g., industry data, open source data, or collected data)
RQ3	Dataset name
RQ3	The generation method of ground-truth for training/testing/validation sets
RQ3	Data type (e.g., source code, nature language text, and pictures)
RQ3	The process that datasets are transformed into input sets suitable for DNNs
RQ3	Presence / absence of replication package
RQ4	The practical problem that a SE task tries to solve
RQ4	The SE activity in which each SE task belongs
RQ4	The approach used for each SE task (e.g., regression, classification, ranking, and generation)
RQ5	Whether and what optimization techniques are used
RQ5	What measures are used to evaluate the DL model

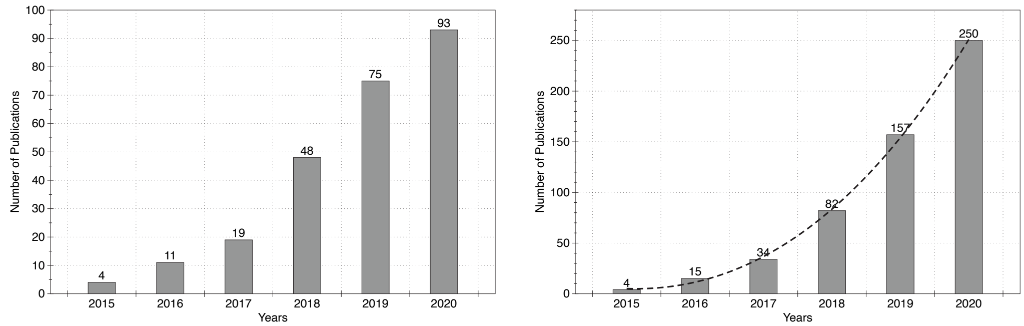
B.1 Publication trends of DL techniques for SE

We analyzed the publication trends of DL-based primary studies published between 2006 and 2020. Although the concept of "Deep Learning" has been proposed in 2006 and DL techniques had been widely used in many other fields in 2009, we did not find any studies using DL to address SE tasks before 2015. Fig. 5(a) shows the number of relevant studies published in predefined publication venues since 2020. It can be observed that the number of publications from 2015 to 2020 shows a significant increase, with the number researching 93 papers in 2020. Besides, another trend among DL studies is that a growing number of papers turn the research direction into using software engineering techniques at the service of DL models, i.e., SE4AI. For example, Guo et al. [5] utilized software testing techniques to locate bugs in the DL frameworks. They present a search-based approach, AUDEE, to identify three types of bugs: logical bugs, crashes, and Not-a-Number (NaN) errors. Specifically, they initialized diverse seeds by exploring hidden layers, inputs, parameters, and model structures and adopted three mutation strategies to generate test cases: Network-level mutation, Input-level mutation, and Weight-level mutation. They then leveraged a heuristic-based cross-checking approach to detect output inconsistencies among different DL frameworks. Finally, they located the corresponding layers in the network by using the causal-testing technique. To improve the ability to resist adversarial attacks, Du et al. [3] performed quantitative robustness analysis on RNN-based DL frameworks to estimate the capability of an RNN neural network in tolerating input perturbations. They built an abstract model to conduct light-weight robustness estimation for real-time Dl applications.

We also performed an analysis of the cumulative number of publications as shown in Fig. 5(b). We fit the cumulative number of publications as a Cubic function, showing the publication trend in the last five years. We can notice that the slope of the curve fitting the distribution increases substantially between 2015 and 2020, and the coefficient of determination (R^2) attains the peak value (0.99928), which indicates that the number of relevant studies using DL in SE intends to experience a strong rise in the future. Therefore, after analyzing Fig. 5, it can be foreseen that using DL techniques to address various SE tasks has become a prevalent trend since 2015, and huge numbers of studies will adopt DL to address further challenges of SE.

B.2 Distribution of publication venues

We reviewed 250 studies published in various publication venues, including 22 conference proceedings and symposiums as well as 10 journals, which covers most research areas in SE. Table 15 lists the number of relevant papers published in each publication venue. 79.6% of publications appeared in conferences and symposiums, while



(a) Number of publications per year.

(b) Cumulative number of publications per year.

Fig. 5. Publication trends of DL-based primary studies in SE.

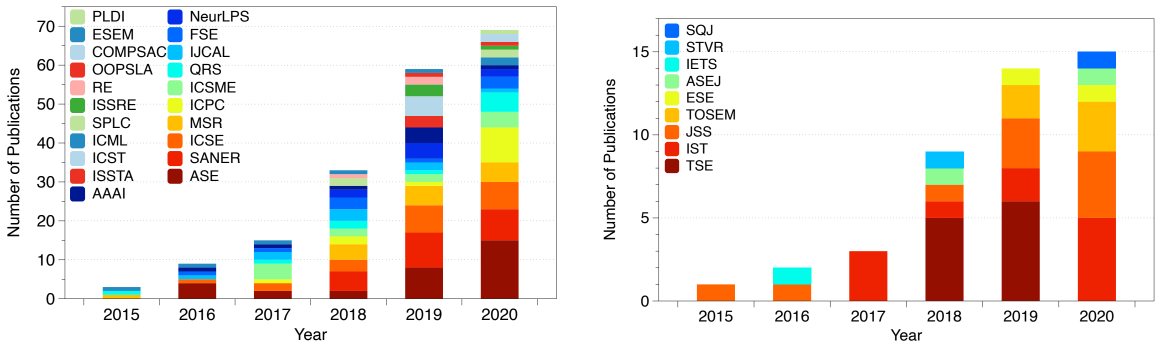
Table 15. Publication Venues with DL-based Studies in SE.

Conference venue				Journal venue	
Acronym	**# Studies**	**Acronym**	**# Studies**	**Acronym**	**# Studies**
ASE	31	AAAI	8	TSE	13
SANER	22	ISSTA	7	IST	11
ICSE	20	ICST	5	JSS	10
MSR	13	ICML	5	ESE	8
ICPC	12	SPLC	4	TOSEM	5
ICSME	12	ISSRE	4	ASEJ	1
ICLR	11	RE	3	IETS	1
QRS	10	OOPSLA	2	STVR	1
IJCAI	9	COMPSAC	2	SQJ	1
FSE	9	ESEM	1		
NeurIPS	8	PLDI	1		

only 20.4% of journal papers leveraged DL techniques for SE tasks. Among all conference papers, 8 different conferences include over 10 studies using DL in SE in the last five years, i.e., ASE, SANER, ICSE, MSR, ICPC, ICSME, ICLR, and QRS. Compared with other conference proceedings, ASE is the most popular one containing the highest number of primary study papers (31), followed by SANER (22). There are 20 and 13 relevant papers published in ICSE and MSR, respectively. Meanwhile, in all journals, TSE includes the highest number of relevant papers (13), followed by IST (11). 10 and 8 studies related to DL techniques were published in JSS and EMSE, respectively. And 5 were published in TOSEM.

We also checked the distribution of primary studies published in conferences and journals between 2015 and 2020, shown in Fig. 6. Fig 6(a) illustrates that the publication trend of various conference proceedings and symposiums has a noticeable increase from 2015 to 2020. 86.8% of conference papers were published between 2018 and 2020, while only a few different conferences or symposium venues included relevant papers between 2015 and 2017, which demonstrates a booming trend in the last few years.

Fig. 6(b) shows the number of primary study papers published in different journal venues. It can be seen that there is an increasing trend in the last five years, especially between 2018 and 2020. Furthermore, the relevant papers published in TSE, as one of the most popular journals, accounts for the largest proportion in 2018 and 2019; while another popular journals, IST and JSS, also make up a large percentage in 2019 and 2020.



(a) Number of primary studies published in various conference proceedings.

(b) Number of primary studies published in various journals.

Fig. 6. Distribution of papers in different publication venues.

Table 16. The definition of five main contributions in primary studies.

Main contribution	**Definition**
New technique or methodology	The study provided a solid solution or developed a novel framework to address specific SE issues.
Tool	The study implemented and published the source code of the DL model or a tool demo targeting SE issues.
Empirical study	The study collected primary data and performed a quantitative and qualitative analysis on the data to explore interesting findings.
Case study	The study analyzed certain SE issues based on one or more specific cases.
User study	The study conducted a survey to investigate the attitudes of different people (e.g., developers, practitioners, users, etc) towards SE issues.

B.3 Types of main contributions

We summarized the main contribution of each primary study and then categorized these studies according to their main contributions into five categories, i.e., New technique or methodology, Tool, Empirical study, Case study, and User study. We give the definition of each main contribution in Fig 7. The main contribution of 90.8% of the primary studies was to build a novel DNN as their proposed new technique or methodology for dealing with various problems in different SE activities. There are 101 studies whose source code of their DL tools or models is available, accounting for 40.4%. Sharing DL models proposed in studies is a valuable contribution for related researchers since it benefits to replicate and reproduce those models in their research. 27 relevant studies concentrated on performing assessment and empirical studies for exploring the benefits of DL towards different SE aspects, such as research on the differences between ML and DL to solve certain SE tasks, the performance of using DL to mine software repositories, applying DL in testing, etc. The main contribution of 5.6% was case studies, and 4 studies conducted user studies to evaluate the performance of DL models. 6 primary studies that both proposed a novel methodology and evaluated the novel methodology via a user study.

Summary

(1) DL has shown a booming trend in recent years.
(2) Most of primary study papers were published between 2018 and 2020.
(3) The number of conference papers employing DNNs for SE significantly exceeds that of journal papers.

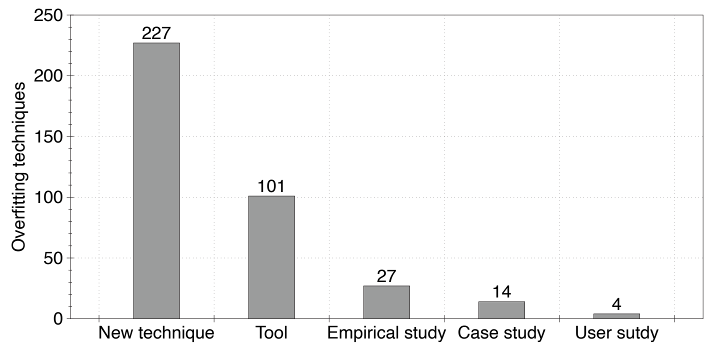


Fig. 7. Types of main contributions

(4) ASE is the conference venue publishing the most DL-based papers (31), while TSE includes the highest number of relevant papers among all journals (13).
(5) Most DL-based studies were only published in a few conference proceedings (e.g., ASE, SANER, ICSE, MSR, and ICPC) and journals (e.g., TSE, IST, JSS, EMSE, and TOSEM).
(6) The main contribution of 90.8% primary studies is to propose a novel methodology by applying various DL techniques, while only 4 primary studies performed a user study to better understand users' attitudes and experience toward various DNNs used for solving specific SE tasks.

C DL MODELS USED IN SE

Table 17 classifies the common DNNs used in primary studies based on their DL architectures and presents the distribution of DNNs in terms of publication time.

D DATA TYPES IN DATASETS

We classified the datasets used in all primary studies into six categories: code-, text-, metric-, graph-, software repository-based datasets, and combined datasets. Table 18 summarizes the specific data types in each category and also describes the usage distribution of those six categories according to the number of references.

E DL MODELS IN DIFFERENT SE ACTIVITIES

Table 19 to Table 24 illustrates the relationships of DNNs with respect to DL architectures, data types, task types, and problem types in six different SE activities, i.e., software requirement, software design, software development, software testing, software maintenance, and software management.

F EVALUATION METRICS FOR DL MODELS

Table 25 and Table 26 list common overfitting techniques and evaluation metrics for DNNs used in four different problem types, including regression, classification, recommendation, and generation SE tasks.

Table 17. The number of various DNNs applied in per year.

ArchitectureFamily		Model Name	2015	2016	2017	2018	2019	2020
Layered architecture	RNN-based model (72)	RNN	[IEEE83]	[IEEE15]	[ACM09, IEEE12, ACM22]	[ACM05, ACM06, IEEE20, IEEE52, ACM30, MITP02, IEEE114]	[IEEE04, IEEE06, SP04, ACM16, IEEE51, IEEE64,ACM29, EL05, EL21, IEEE126, IEEE107]	[ACM07, EL02, EL03, IEEE26]
		RtNN		[ACM02, IEEE15]		[IEEE80]		
		Bidirectional RNN (BRNN)						[ACM07, EL03]
		LSTM			[ACM09, ACM21, IEEE56, MK05, MK06, ICLR08]	[ACM05, ICLR09, IEEE49, IEEE42, IEEE53, MK04, MK09, ACM31, EL17, IEEE87, IEEE116, IEEE128, IEEE131]	[AAAI03, AAAI04, AAAI05, AAAI06, IEEE01, IEEE08, ICLR02, IEEE38, ACM17, IEEE63, EL14, IEEE74, IEEE77, IEEE134, IEEE110, IEEE104]	[AAAI07, ACM07, IEEE21, ICLR01, IEEE67, EL03, EL20, IEEE97, IEEE99, IEEE108, ACM32, IEEE135, IEEE141]
		Bi-LSTM			[IEEE54]		[ACM23, IEEE125, IEEE94]	[EL12, IEEE28, ACM18, EL03, IEEE35, IEEE92]
		GRU			[AAAI01, ICLR05]	[AAAI08]	[IEEE107, ACM37]	[ACM07, IEEE18, ACM10, EL16, EL03, EL11, IEEE136]
		Bidirectional GRU					[MK02]	[EL03]
		Recurrent Highway Network					[IEEE134]	
	CNN-based model	CNN		[AAAI02, AAAI05, IEEE13, IEEE16, ACM13, MK07]	[IEEE54, IEEE57, MK06, IEEE85]	[ACM03, ACM05, EL08, IEEE10, ICLR09, IEEE41, IEEE52, IEEE53, IEEE78, IEEE81, IEEE115, IEEE118, moran2018machine, IEEE130]	[EL07, ACM24, ACM19, IEEE50, IEEE62, IEEE65, IEEE69, EL05, IEEE73, IEEE76, IEEE125,MITP03, IEEE124, IEEE123, IEEE122, ACM34, IEEE111, IEEE109, IEEE106, IEEE103, IEEE95, IEEE94, MITP04]	[AAAI07, ACM08, IEEE18, IEEE22, IEEE23, IEEE02, IEEE30, SP08, IEEE46, IEEE47, IEEE24, IEEE67, ACM27, EL02, IEEE100, IEEE102, ACM36, IEEE135, IEEE139, IEEE140, IEEE142, IEEE132]
		Tree-based CNN (TBCNN)					[IEEE104]	[EL13]
		RCNN					[EL05]	[EL02]
		DPCNN						[MITP05]
		Deep Residual Network						[EL04]

Architecture	Family	Model Name	2015	2016	2017	2018	2019	2020
Layered architecture	FNN-based model	FNN		[IEEE14, IET01, EL18]	[IEEE55]	[ACM04, EL09, IEEE10, SP01, EL17, IEEE86, IEEE96, IEEE121, IEEE120, W01]	[IEEE07, IEEE09, EL15, IEEE88]	[IEEE60, EL01, IEEE71, IEEE82, IEEE119, IEEE91, IEEE93, IEEE25, IEEE117, IEEE58]
		RBFNN	[EL19, ACM20]					
		Deep Sparse FNN					[ACM14]	
		MLP					[SP04]	
	GNN-based model	GGNN			[ICLR10]		[IEEE01, IEEE06, MITP03, IEEE104]	[MITP05, IEEE28, IEEE138]
		Graph Matching Network (GMN)					[IEEE64]	[ICLR01, ACM39]
		GNN						[ACM38]
		ConvGNN						[IEEE136]
	Tailored model	Deep Beliefe Network (DBN)	[IEEE84]	[IEEE43]		[IEEE127]		
		HAN					[EL05]	[IEEE45]
		Deep Forest					[EL06]	[IEEE90]
		GAN				[MITP01]		
		Deep fusion learning model						[ACM28]
		Mata-learning						[SP06, IEEE48, ACM39]
	Transformer	Transformer						[IEEE34, IEEE36, ACM10, ACM26]
		Bert						[ACM07] [IEEE191, IEEE29]
Encoder-Decoder	RNN-based model	RNN		[IEEE15]		[MK08, IEEE79, IEEE113]	[gao2019automating, SP03, ACM15, IEEE75, ACM35, IEEE112]	[IEEE59]
		LSTM			[IEEE11]	[IEEE31, IEEE41]	[ICLR07, IEEE61, MITP06, MITP07]	[IEEE37, ACM11, IEEE47, MITP08, IEEE101, SP10]
		Bi-LSTM						[IEEE32, IEEE66]
		LSTM-CRF						[IEEE58]
		GRU			[ICLR11]			
	CNN-based model	CNN			[ACM12]		[IEEE05]	[IEEE44, ACM33]
	FNN-based model	FNN				[EL10]	[MK03]	[IEEE17, ACM26]

Architecture	Family	Model Name	2015	2016	2017	2018	2019	2020
AutoEncoder	RNN-based model	Bi-GRU						[IEEE137]
		GRU						[SP07]
		RNN					[AAAI05]	
		LSTM				[ICLR03]		
		Bi-LSTM						[IEEE33]
	CNN-based model	CNN						[ACM33]
	FNN-based model	FNN			[IEEE40]	[EL10, IEEE39]	[SP04, ICLR04, IEEE68, IEEE105]	[IEEE72]
Siamese Network	RNN-based model	GRU						[IEEE27]
		LSTM						[ACM25]
		Bi-LSTM						[IEEE48]

Table 18. Data types of datasets involved in primary studies.

Family	Data types	References
Code-based datasets	Source code	[AAAI01, AAAI02, AAAI03, AAAI04, AAAI05, AAAI06, ACM01, ACM02, ACM04, ACM06, ACM09, EL01, IEEE109, IEEE01, IEEE32, IEEE34, IEEE35, IEEE37, SP09, ACM10, ICLR06, ICLR07, ICLR10, ACM18, ACM20, ACM23, ACM17, IEEE67, MK08, MK09, EL16, EL20, IEEE82, IEEE83, MITP02, MITP06, MITP07, ACM38, IEEE25, IEEE112, IEEE121, IEEE141, IEEE17, IEEE66, IEEE63, IEEE64, IEEE65, IEEE68, ACM27, EL06, EL17, IEEE74, IEEE102, IEEE103, IEEE105, IEEE114, IEEE116, IEEE124, IEEE128, IEEE131, IEEE06, IEEE09, IEEE10, IEEE15, IEEE27, SP07, SP04, ICLR01, ICLR02, ICLR03, ICLR04, ICLR05, ACM11, ACM12, ACM13, IEEE38, ACM15, ACM16, IEEE42, IEEE43, IEEE47, IEEE51, IEEE55, MK03, MK04, MK05, MK06, MK07, ACM28, EL08, IEEE77, IEEE80, MITP01, MITP03, MITP04, MITP05, MITP08, ACM37, IEEE84, IEEE85, IEEE86, IEEE87, IEEE91, IEEE92, IEEE101, IEEE28, IEEE104, IEEE110, IEEE113, IEEE115, IEEE115, W01, IEEE122, IEEE126, IEEE138, IEEE81, ACM39, IEEE132, IEEE98]
	Source code in DSL (Domain-Special Language)	[IET01, ICLR08, ICLR11, IEEE72]
	Test case	[ACM26, ACM31, EL03, ACM30]
	Defects	[ACM29, EL10, ACM35]
	Patch	[IEEE29, IEEE123]
	Execution trace	[IEEE08]
	Code change	[IEEE127]
	Game bug	[IEEE07]
Text-based datasets	Bug report	[IEEE40, IEEE54, EL07, EL12, EL13, IEEE39, IEEE60, IEEE71, IEEE137, IEEE139, IEEE142]
	Requirement documentation	[IEEE58, EL09, IEEE94, IEEE95, IEEE96, IEEE118, IEEE119, IEEE58, IEEE12]
	Issue report	[IEEE191, IEEE134, SP02, IEEE130]
	Log information	[IEEE21, IEEE45, IEEE88]
	Code comment	[IEEE108, IEEE36, ACM34, IEEE98]
	Incident report	[ACM08, IEEE05, IEEE22]
	Dialog	[ACM05] [IEEE48, SP10, IEEE03]
	configuration documentation	[ACM14, ACM21]
	API	[SP05, SP03]
	User behavior	[ACM22, SP06]
	Protocol message	[IEEE61]
	Vulnerability descriptions in CVE Details websit	[IEEE57]
	Defect report	[IEEE62]
	Use case	[EL02]
	Method names	[IEEE59]
	Design documentation	[IEEE97]
	Certification	[IEEE52]
	SATD	[IEEE100]
	Web request	[IEEE30]
	Text information (PDF)	[IEEE12]
Metric-based datasets	Software metric	[EL19, ACM32, IEEE46, IEEE56, EL18, IEEE50, IEEE93]
	Code metric	[IEEE117]

	Driver properties	[IEEE14]
	resource utilization traces	[EL11]
Graph-based datasets	GUI images	[ACM33, moran2018machine, IEEE41, IEEE23, IEEE69, ACM25, IEEE44]
	program screenshot	[SP08, ACM24, IEEE140, ACM36]
	Vedio screenshot	[IEEE78]
	behaviour trajectory of the model class	[EL04]
Software repository -based datasets	Q&A pair (Knowledge unit) in forums	[ACM03, IEEE13, IEEE16, IEEE125, IEEE70, IEEE76, IEEE79]
	Pull-requests	[IEEE04, IEEE99]
	Issues and commits	[ACM07, EL14]
	Tags in forum	[SP01, EL05]
	Discuss	[IEEE111]
	Commits	[IEEE90]
Combined datasets	source code and comment	[AAAI07, AAAI08, IEEE33, IEEE20, IEEE24, IEEE26, IEEE136, IEEE31, IEEE49, EL21, IEEE53]
	Source code and bug report	[MK01, IEEE89, IEEE135]
	Source code and commit message	[ACM19]
	Code change and commit message	[IEEE73, IEEE75]
	Source code and Q&A pair in SO	[IEEE02]
	Source code, diff files, and commit message	[MK02]
	Diff file and commit message	[IEEE11]
	Code review and code clone pair	[IEEE106]
	Test case and log file	[IEEE18]

Table 19. The relationships of DNNs with respect to DL architecture, task types, problem types, as well as data types in software requirement.

Task type	**Problem type**	**DNN**	**DL architecture**	**Data type**	**Reference**
Requirement extraction	classification	LSTM-CRF	Encoder-decoder	text-based data	[IEEE58]
		LSTM, CNN		text-based data	[IEEE94]
		Bi-LSTM, deep siamese network		text-based data	[IEEE48]
		LSTM, Bert	Transformer	text-based data	[IEEE19]
		CNN		text-based data	[IEEE118]
		FNN		text-based data	[EL09]
	recommendation	FNN		text-based data	[IEEE581]
		FNN		text-based data	[IEEE120]
Requirement validation	classification	CNN		text-based data	[IEEE95]
Requirement traceability	recommendation	FNN		text-based data	[IEEE96]

Table 20. The relationships of DNNs with respect to DL architecture, task types, problem types, as well as data types in software design.

Task type	Problem type	DNN	DL architecture	Data type	Reference
Software design pattern detection	classification	CNN		text-based data	[IEEE109]
		CNN	Autoencoder	image-based data	[ACM33]
GUI modeling	Generation	CNN		image-based data	[IEEE129]
		CNN, RNN	Encoder-decoder	image-based data	[IEEE41]

Table 21. The relationships of DNNs with respect to DL architecture, task types, problem types, as well as data types in software development.

Task type	Problem type	DNN	DL architecture	Data type	Reference
Code representation	Generation	RNN		code-based data	[IEEE83]
		RNN		code-based data	[MITP02]
		LSTM		code-based data	[ACM09]
		LSTM		code-based data	[EL20]
		LSTM	Encoder-decoder	code-based data	[ICLR07]
		Bi-LSTM		code-based data	[ACM18]
		GRU		code-based data	[EL16]
		CNN		code-based data	[AAAI02]
		FNN	Encoder-decoder	code-based data	[ACM20]
		Bi-RNN		code-based data	[ACM23]
		FNN		code-based data	[IEEE82]
		GGNN		code-based data	[ICLR10]
		GNN		code-based data	[ACM38]
Code generation	Generation	RNN	Encoder-decoder	code-based data	[IEEE32]
		RNN	Encoder-decoder	code-based data	[IEEE37]
		LSTM		code-based data	[SP09]
		LSTM		code-based data	[ICLR08]
		LSTM		code-based data	[MITP07]
		LSTM	Encoder-decoder	code- and text-based data	[MITP08]
		LSTM	Encoder-decoder	code-based data	[ICLR03]
		LSTM, CNN		code- and text-based data	[ICLR09]
		Bi-LSTM	Encoder-decoder	code-based data	[IEEE66]
		Bi-LSTM		code-based data	[MITP06]
		GRU	Encoder-decoder	code-based data	[ICLR11]
		Transformer	Transformer	code-based data	[ACM10]
Code comment generation	Generation	GRU		code- and text-based data	[AAAI08]
		RNN		code- and text-based data	[EL21]
		LSTM		code- and text-based data	[IEEE49]
		LSTM	encoder-decoder	code-based data	[IEEE101]
		LSTM		code- and text-based data	[IEEE108]
		Bi-LSTM	encoder-decoder	code- and text-based data	[IEEE33]
		Bi-LSTM		code- and text-based data	[IEEE35]
		multiple DNN models		code- and text-based data	[IEEE36]

		GRU	autoencoder	code-based data	[SP07]
Code search	Recommendation	RNN	Encoder-decoder	code-based data	[ACM02]
		RNN		code- and text-based data	[IEEE26]
		CNN		code- and text-based data	[IEEE02]
		CNN		code- and text-based data	[IEEE24]
		FNN		code-based data	[IEEE25]
		CODEnn		code- and text-based data	[IEEE20]
		GGNN		code-based data	[IEEE01]
		DPCNN, GNN		code-based data	[MITP05]
		multiple		code-based data	[ACM01]
Code localization	Classification	CNN		image-based data	[SP08]
		CNN		image-based data	[IEEE140]
		CNN		image-based data	[ACM24]
		CNN		image-based data	[ACM36]
		CNN		image-based data	[IEEE78]
Code completion	Generation	Bert	Transformer	code-based data	[IEEE3]
		LSTM		code-based data	[MK09]
		LSTM		code-based data	[IEEE141]
		FNN		code-based data	[IEEE121]
Code summarization	Generation	RNN	Encoder-decoder	text-based data	[IEEE31]
		RNN	Encoder-decoder	code-based data	[MK08]
		RNN		code-based data	[ACM16]
		CNN		code-based data	[ACM13]
		ConvGNNs, GRU		code- and text-based data	[IEEE136]
Method name generation	Generation	RNN	Encoder-decoder	code- and text-based data	[IEEE112]
		seq2seq	encoder-decoder	code- and text-based data	[IEEE59]

Table 22. The relationships of DNNs with respect to DL architecture, task types, problem types, as well as data types in software testing.

Task type	**Problem type**	**DNN**	**DL architecture**	**Data type**	**Reference**
bug-related detection	Classification	CNN		image-based data	[IEEE23]
		LSTM		metric-based data	[ACM32]
		Bi-LSTM		text-based data	[IEEE54]
		Bi-LSTM		text-based data	[EL12]
		GRU		code-based data	[ACM37]
		LSTM, FNN		code-based data	[EL17]
		LSTM, GNN		code-based data	[ICLR01]
		CNN		image-based data	[IEEE69]
		CNN		metric-based data	[IEEE102]
		CNN		code- and text-based data	[ACM19]
		CNN		code-based data	[IEEE50]
		FNN		text-based data	[IEEE88]
		FNN	Autoencoder	metric-based data	[IEEE68]
Bug localization	Classification	FNN	Autoencoder	text-based data	[IEEE40]
		CNN, RNN		code-based data	[MITP04]

	recommendation	CNN, LSTM, Deep walk		code- and text-based data	[AAAI07]
		CNN, LSTM	Encoder-decoder	code- and text-based data	[MK01]
		CNN, LSTM		code- and text-based data	[IEEE135]
		LSTM		code-based data	[IEEE116]
		CNN		code-based data	[MK07]
		CNN		text-based data	[EL07]
		CNN		code-based data	[IEEE103]
		CNN, LSTM		code-based data	[MK06]
	Recommendation	FNN		code- and text-based data	[IEEE89]
		RNN		code-based data	[ACM29]
		CNN		code- and text-based data	[IEEE124]
Vulnerability detection	Classification	LSTM		code-based data	[IEEE131]
		CNN		text-based data	[IEEE57]
		GRU, CNN		code-based data	[MITP03]
		RNN, LSTM, GRU, BRNN		code-based data	[EL03]
		FNN	Encoder-decoder	code-based data	[ICLR04]
		FNN	Encoder-decoder	text-based data	[MK03]
Testing techniques	Generation	RNN		text-based data	[IEEE12]
		LSTM		text-based data	[ACM31]
	Regression	LSTM	Siamese Network	metric-based data	[ACM25]
		CNN		metric-based data	[IEEE65]
		FNN		metric-based data	[IEEE14]
	Classification	FNN		code-based data	[IEEE07]
Test case generation	Generation	RNN		code-based data	[AAAI06]
		RNN		text-based data	[ACM22]
		RNN	Encoder-decoder	code-based data	[ACM26]
		FNN		image-based data	[IEEE93]
		FNN	Encoder-decoder	code-based data	[IEEE17]
		LSTM		code-based data	[IEEE63]
		FNN, LSTM		metric-based data	[IEEE61]
		RNN		text-based code	[ACM30]
Program analysis	Classification	LSTM		code-based data	[IEEE14]
		CNN		text-based data	[IEEE65]
		CNN, GRU		text-based data	[IEEE07]
		CNN, RNN, LSTM		text-based data	[ACM05]
	Recommendation	RNN		code-based data	[IEEE12]
		LSTM		code-based data	[ACM17]
		GGNN		code-based data	[ACM39]
Bug classification	Classification	TBCNN		code- and text-based data	[EL13]
	recommendation	FNN		text-based data	[IEEE60]
	CNN	Encoder	metric- and text-based data	[IEEE05]	
Certification validation	Recommendation	CNN, RNN		text-based data	[IEEE52]
Stateful service virtualization	Generation	LSTM	Encoder-deocder	text-based data	[SP10]

Table 23. The relationships of DNNs with respect to DL architecture, task types, problem types, as well as data types in software maintenance.

Task type	Problem type	DNN	DL architecture	Data type	Reference
defect prediction	classification	CNN, FNN		metric-based data	[IEEE10]
		RNN		metric-based data	[IEEE114]
		RNN		code- and text-based data	[IEEE135]
		LSTM		code-based data	[IEEE74]
		LSTM		metric-based data	[IEEE87]
		LSTM		code-based data	[IEEE128]
		Bi-LSTM, Tree-LSTM		code-based data	[IEEE92]
		CNN		text-based data	[IEEE30]
		CNN		code-based data	[IEEE46]
		CNN		code- and text-based data	[IEEE73]
		FNN	Autoencoder	metric-based data	[EL10]
		FNN		code-based data	[IEEE86]
		FNN		metric-based data	[IEEE117]
		FNN		metric-based data	[EL15]
		FNN	Autoencoder	code-based data	[IEEE72]
		DBN		metric-based data	[IEEE127]
		DBN		code-based data	[IEEE43]
		DBN		code-based data	[IEEE84]
		Deep forest		metric-based data	[EL06]
		Deep forest		metric-based data	[IEEE90]
		RBM		code-based data	[SP04]
		CNN		code-based data	[IEEE85]
Program repair	generation	RNN	Encoder-decoder	code-based data	[ACM35]
		LSTM		code-based data	[AAAI03]
		LSTM	Encoder-decoder	code-based data	[IEEE46]
		LSTM		code-based data	[IEEE42]
		GRU		code-based data	[ICLR05]
		GRU	Encoder-deocder	code-based data	[AAAI01]
		CNN		code-based data	[ACM27]
		CNN		code- and text-based data	[IEEE132]
		GGNN, Bi-GRU		code-based data	[IEEE138]
		GAN		code-based data	[MITP01]
		Bert	Transformer	code- and text-based data	[IEEE29]
	classifcation	LSTM		code-based data	[ICLR02]
	recommendation	LSTM	Encoder-decoder	code-based data	[ACM11]
		FNN	Autoencoder	code-based data	[IEEE105]
Code clone detection	classification	RNN		code-based data	[IEEE51]
		RtNN, RvNN		code-based data	[IEEE15]
		LSTM		code-based data	[AAAI04]
		LSTM		code-based data	[IEEE38]
		LSTM		code-based data	[MK04]
		LSTM		code-based data	[MK05]
		LSTM		code-based data	[IEEE77]
		LSTM		code-based data	[IEEE110]
		GRU	SiameseNetwork	code-based data	[IEEE27]

		GGNN, GMN		code-based data	[IEEE98]
	regression	RTNN		code-based data	[IEEE80]
bug report related	classification	CNN		text-based data	[IEEE142]
		CNN		text-based data	[IEEE139]
		FNN		text-based data	[IEEE7]
	generation	Bi-GRU	autoencoder	text-based data	[IEEE137]
		FNN	autoencoder	text-based data	[IEEE39]
	recommendation	LSTM		text-based data	[SP02]
		BERT, CNN, RNN-LSTM, RNN-GRU, Bi-RNN		text-based data	[ACM07]
software reliability	regression	FNN		code-based data	[W01]
software maintainability		FNN		metric-based data	[EL18]
software readability	classification	CNN		code-based data	[EL08]
software trustworthiness		deep residual network		image-based data	[EL04]
software traceability	recommendation	LSTM		text-based data	[ACM21]
compiled-related	generation	RNN	Encoder-decoder	code-based data	[IEEE113]
		RNN, GGNN		code-based data	[IEEE06]
	recommendation	LSTM, CNN		code-based data	[IEEE67]
		FNN		code-based data	[IEEE91]
SATD detection	classification	Bi-LSTM	autoencoder	text-based data	[IEEE28]
		CNN		text-based data	[IEEE100]
		CNN		text-based data	[ACM34]
code smell detection	classification	CNN		code-based data	[IEEE115]
		CNN		code-based data	[IEEE122]
	.	CNN, LSTM		code-based data	[ACM12]
Code review	classification	CNN		code-based data	[IEEE106]
		CNN, LSTM	autoencoder	code-based data	[AAAI05]
	recommendation	LSTM		text-based data	[IEEE99]
software/code classification	classification	CNN, LSTM		code-based data	[IEEE53]
		GNN		code-based data	[IEEE64]
		TBCNN, LSTM, GGNN		code- and text-based data	[IEEE104]
code change	generation	RNN	Encoder-decoder	code-based data	[ACM15]
		HAN		text-based data	[IEEE45]
incident detection	recommendation	CNN		text-based data	[IEEE22]
	classification	CNN		text-based data	[ACM08]

Table 24. The relationships of DNNs with respect to DL architecture, task types, problem types, as well as data types in software management.

Task type	Problem type	DNN	DL architecture	Data type	Reference
effort cost prediction	regression	LSTM, recurrent highway network		text-based data	[IEEE134]
		CNN, RNN, CRNN		text-based data	[EL02]
		FNN		metric-based data	[IET01]
		FNN		metric-based data	[EL01]
		RBFNN		metric-based data	[EL19]
mining GitHub	generation	Bi-GRU		code- and text-based data	[MK02]
		RNN	Encoder-decoder	text-based data	[IEEE11]
		RNN	Encoder-decoder	code- and text-based data	[IEEE75]
		LSTM		code- and text-based data	[EL14]
		RNN		text-based data	[IEEE04]
	classification	RNN, GRU		code- and text-based data	[IEEE107]
mining Stack-Overflow	classification	RNN	Encoder-decoder	text-based data	[IEEE79]
		CNN		text-based data	[IEEE76]
		CNN		text-based data	[IEEE81]
	recommendation	CNN		text-based data	[IEEE16]
app mining	classification	CNN		text-based data	[IEEE111]
	generation	RNN	Encoder-decoder	text-based data	[IEEE03]
		RNN	Encoder-decoder	text-based data	[SP03]
tag mining	recommendation	FNN		metric-based data	[SP01]
		TagCNN, TagRNN, TagHAN and TagRCNN		text-based data	[EL05]
	generation	LSTM	Encoder-decoder	text-based data	[SP05]
		CNN	Encoder-decoder	image-based data	[IEEE44]
developer-based mining	classification	CNN		text-based data	[IEEE13]
		CNN		text-based data	[IEEE130]
	recommendation	meta learning		text-based data	[SP06]

Table 26. Evaluation metrics for different problem types.

problem type	Metric	# Studies	Reference
regression	MAE	3	[EL01, EL18, IEEE134]
	Standardized Accuracy (SA)	3	[EL01, EL02, IEEE134]
	MMRE	3	[IET01, EL18, IEEE134]
	MdAE	2	[EL02, IEEE134]

classification	precision	76	[AAAI04, ACM03, ACM04, ACM05, ACM07, ACM08, IEEE08, IEEE09, IEEE10, IEEE13, IEEE15, IEEE18, IEEE19, IEEE21, IEEE22, IEEE23, IEEE27, IEEE28, IEEE30, IEEE32, SP08, ICLR04, IEEE38, ACM17, ACM19, IEEE43, IEEE46, IEEE48, IEEE50, IEEE53, IEEE54, IEEE55, IEEE57, ACM24, MK03, MK04, IEEE67, MK05, ACM28, ACM30, EL03, EL06, EL12, IEEE73, IEEE76, IEEE77, IEEE79, IEEE81, ACM37, IEEE84, IEEE85, IEEE86, IEEE87, IEEE88, IEEE94, IEEE28, IEEE100, IEEE102, IEEE106, IEEE107, IEEE109, IEEE111, IEEE115, IEEE117, ACM34, IEEE122, IEEE123, IEEE125, IEEE127, IEEE128, IEEE130, IEEE131, IEEE139, IEEE140, IEEE14]
	recall	73	[AAAI04, ACM03, ACM04, ACM05, ACM07, ACM08, IEEE08, IEEE09, IEEE10, IEEE13, IEEE18, IEEE19, IEEE21, IEEE22, IEEE23, IEEE27, IEEE28, IEEE30, IEEE32, SP08, ICLR04, IEEE38, ACM17, ACM19, IEEE43, IEEE46, IEEE48, IEEE50, IEEE53, IEEE54, IEEE55, IEEE57, ACM24, MK04, IEEE67, MK05, ACM28, ACM30, EL03, EL06, EL12, IEEE73, IEEE76, IEEE77, IEEE79, IEEE81, ACM37, IEEE84, IEEE85, IEEE86, IEEE87, IEEE88, IEEE94, IEEE28, IEEE100, IEEE102, IEEE106, IEEE107, IEEE109, IEEE111, IEEE117, ACM34, IEEE122, IEEE123, IEEE125, IEEE130, IEEE131, IEEE127, IEEE139, IEEE140, IEEE142]
	F1-score	62	[AAAI04, AAAI05, ACM04, ACM05, ACM07, ACM08, IEEE09, IEEE10, IEEE13, IEEE18, IEEE19, IEEE21, IEEE23, IEEE27, IEEE28, IEEE32, ICLR04, IEEE38, ACM17, ACM19, IEEE43, IEEE46, IEEE48, IEEE50, IEEE53, IEEE57, ACM24, MK03, MK04, IEEE67, MK05, ACM28, ACM30, EL03, EL08, EL12, EL13, IEEE73, IEEE76, IEEE77, MITP03, IEEE84, IEEE85, IEEE87, IEEE88, IEEE94, IEEE95, IEEE28, IEEE100, IEEE102, IEEE106, IEEE107, IEEE109, IEEE111, ACM34, IEEE123, IEEE125, IEEE130, IEEE127, IEEE139, IEEE142]
	Accuracy	27	[ICLR02, ICLR04, ACM19, IEEE54, IEEE57, IEEE67, EL03, EL04, EL06, EL08, MITP03, IEEE86, IEEE88, IEEE95, IEEE104, IEEE109, IEEE122, IEEE123, IEEE130, IEEE139, IEEE140]
	AUC	22	[AAAI05, IEEE22, SP04, ICLR04, EL06, EL10, IEEE73, IEEE76, IEEE79, IEEE87, IEEE90, IEEE92, IEEE97, IEEE100, IEEE19, IEEE110, IEEE114, IEEE117, ACM32, IEEE123, IEEE135, IEEE139]
	MCC	8	[EL10, IEEE86, IEEE90, IEEE92, IEEE100, IEEE102, IEEE109, IEEE115]
	ROC	6	[ACM05, IEEE79, IEEE92, IEEE110, IEEE115, IEEE117]
	True positve rate	3	[MK03, ICLR02, IEEE86]
	False positive rate	2	[MK03, EL03]
	False negative rate	2	[MK03, EL03]
Recommendation	MRR	17	[AAAI07, IEEE02, IEEE16, IEEE20, IEEE24, IEEE25, IEEE26, SP06, EL07, IEEE89, IEEE99, IEEE116, ACM33, IEEE124, ACM36, IEEE126, IEEE135]
	MAP/MAP@k	11	[AAAI07, IEEE16, IEEE24, ACM21, MK06, EL07, IEEE89, IEEE124, ACM36, IEEE126, IEEE135]
	precision@k	7	[IEEE16, IEEE20, IEEE25, SP01, EL05, ACM33, ACM36]
	recall@k	6	[SP01, SP02, IEEE60, EL05, MITP05, IEEE99]
	F1-score@k	4	[SP01, EL05, IEEE96, ACM36]
	Recall	22	[IEEE04, IEEE06, IEEE29, IEEE32, SP05, IEEE35, SP07, ACM10, ICLR07, IEEE39, ACM20, IEEE45, ACM23, IEEE58, IEEE59, MK02, MK08, EL14, EL21, ACM38, IEEE101, IEEE137, IEEE119]
Generation	precision	21	[IEEE04, IEEE11, IEEE29, IEEE32, SP05, SP07, ICLR07, IEEE39, ACM20, IEEE45, ACM23, IEEE58, IEEE59, MK08, EL14, EL21, ACM38, IEEE101, IEEE58, IEEE129, IEEE137, IEEE119]
	BLEU	19	[IEEE03, IEEE11, IEEE31, IEEE33, SP03, IEEE36, SP07, SP09, ACM16, IEEE41, IEEE44, MK02, IEEE75, EL21, MITP06, MITP07, IEEE112, IEEE121, IEEE136]

ROUGE	11	[AAAI08, IEEE04, IEEE31, SP09, IEEE39, IEEE44, IEEE75, MITP07, IEEE101, IEEE136, IEEE137]
F1-score	10	[IEEE04, IEEE32, SP05, SP09, ICLR07, ICLR10, ACM23, IEEE59, EL21, IEEE101]
Exact match	7	[ICLR03, ICLR09, IEEE41, MITP06, MITP08, IEEE112, IEEE138]
Running time	6	[IEEE41, ACM22, IEEE63, ACM31, IEEE113, IEEE121]
METEOR	5	[IEEE31, SP07, MK02, EL21, MITP07]
perplexity (PP)	2	[IEEE66, IEEE83]

G LIST OF PRIMARY STUDIES IN THE SLR

In this section, we classified all primary studies according to their publishers and listed them as follows: (**EL**: Elsevier; **MITP**: MIT Press; **MK**: Morgan Kaufmann; **SP**: Spring; **W**: Wiley)

AAAI01: Rahul Gupta, Soham Pal, Aditya Kanade, and Shirish Shevade. 2017. Deepfix: Fixing common c language errors by deep learning. In Thirty-First AAAI Conference on Artificial Intelligence.

AAAI02: Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. 2016. Convolutional neural networks over tree structures for programming language processing. In Thirtieth AAAI Conference on Artificial Intelligence.

AAAI03: Rahul Gupta, Aditya Kanade, and Shirish Shevade. 2019. Deep reinforcement learning for syntactic error repair in student programs. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. 930–937.

AAAI04: Yan-Ya Zhang and Ming Li. 2019. Find Me if You Can: Deep Software Clone Detection by Exploiting the Contest between the Plagiarist and the Detector. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. 5813–5820.

AAAI05: Shu-Ting Shi, Ming Li, David Lo, Ferdian Thung, and Xuan Huo. 2019. Automatic code review by learning the revision of source code. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. 4910–4917.

AAAI06: Xiao Liu, Xiaoting Li, Rupesh Prajapati, and Dinghao Wu. 2019. Deepfuzz: Automatic generation of syntax valid c programs for fuzz testing. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. 1044–1051.

AAAI07: Xuan Huo, Ming Li, and Zhi-Hua Zhou. 2020. Control flow graph embedding based on multi-instance decomposition for bug localization. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 4223–4230.

AAAI08: Yuding Liang and Kenny Zhu. 2018. Automatic generation of text descriptive comments for code blocks. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32.

ACM01: Jose Cambronero, Hongyu Li, Seohyun Kim, Koushik Sen, and Satish Chandra. 2019. When deep learning met code search. In FSE. 964–974.

ACM02: Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. 2016. Deep API learning. In FSE. 631–642.

ACM03: Bowen Xu, Amirreza Shirani, David Lo, and Mohammad Amin Alipour. 2018. Prediction of relatedness in stack overflow: deep learning vs. SVM: a reproducibility study. In ESEM. 1–10.

ACM04: Gang Zhao and Jeff Huang. 2018. Deepsim: deep learning code functional similarity. In FSE. 141–151.

ACM05: Jinman Zhao, Aws Albarghouthi, Vaibhav Rastogi, Somesh Jha, and Damien Octeau. 2018. Neural-augmented static analysis of Android communication. In FSE. 342–353.

ACM06: Vincent J Hellendoorn, Christian Bird, Earl T Barr, and Miltiadis Allamanis. 2018. Deep learning type inference. In FSE. 152–162.

ACM07: Yutong Zhao, Lu Xiao, Pouria Babvey, Lei Sun, Sunny Wong, Angel A Martinez, and Xiao Wang. 2020. Automatically identifying performance issue reports with heuristic linguistic patterns. In Proceedings of the

Table 25. The distribution of various overfitting techniques used in primary studies.

Optimization	#Studies	References
Dropout	47	[AAAI01, AAAI06,AAAI07, ACM06, IEEE02, IEEE21, IEEE24, IEEE26, IEEE28, IEEE33, SP02, IEEE34, IEEE35, SP09, ACM10,ICLR07, ACM11, ACM13, ICLR10, IEEE39, ACM16, ACM18, ACM20, IEEE45, IEEE48, IEEE53, EL10, EL13,EL16, IEEE75, IEEE79, EL21, IEEE87, IEEE108, IEEE125, IEEE134, IEEE135, IEEE138, IEEE62, MK02,IEEE66, IEEE67, MK06, MK07, EL02]
Pooling	26	[ACM01, ACM04, ACM19, ACM23, ACM34, ACM33, IEEE05, IEEE06, IEEE15, IEEE20, IEEE38, IEEE41, IEEE54, IEEE62, EL02, EL05, EL20, IEEE78, IEEE94, IEEE96, IEEE106, IEEE113, IEEE123, IEEE124, IEEE125, IEEE126]
Regularization	24	[ACM14, ACM17, ACM21, ACM29, IEEE83, IEEE86 IEEE50, IEEE57, IEEE128, IEEE100, IEEE106, IEEE133, IEEE13, IEEE16, IEEE25, IEEE128, SP02, ICLR04, ICLR06, ICLR09, EL09, EL11, MITP03, IET01]
Data augmentation	9	[IEEE46, IEEE78, IEEE83, IEEE97, IEEE116, IEEE129, ACM25, ACM36, EL21]
Data balancing	5	[ICLR08, IEEE46, IEEE60, ACM35, ACM34]
Early stoping	4	[IEEE48, EL10, IEEE73, IEEE134]

28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 964–975.

ACM08: Yujun Chen, Xian Yang, Hang Dong, Xiaoting He, Hongyu Zhang, Qingwei Lin, Junjie Chen, Pu Zhao, Yu Kang, Feng Gao, et al. 2020. Identifying linked incidents in large-scale online service systems. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 304–314.

ACM09: Vincent J Hellendoorn and Premkumar Devanbu. 2017. Are deep neural networks the best choice for modeling source code?. In FSE. 763–773.

ACM10: Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. 2020. Intellicode compose: Code generation using transformer. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 1433–1443.

ACM11: Michihiro Yasunaga and Percy Liang. 2020. Graph-based, self-supervised program repair from diagnostic feedback. In International Conference on Machine Learning. PMLR, 10799–10808.

ACM12: Dor Levy and Lior Wolf. 2017. Learning to align the source code to the compiled object code. In International Conference on Machine Learning. PMLR, 2043–2051.

ACM13: Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In International conference on machine learning. PMLR, 2091–2100.

ACM14: Huong Ha and Hongyu Zhang. 2019. Deepperf: performance prediction for configurable software with deep sparse neural network. In ICSE. IEEE, 1095–1106.

ACM15: Michele Tufano, Jevgenija Pantiuchina, Cody Watson, Gabriele Bavota, and Denys Poshyvanyk. 2019. On learning meaningful code changes via neural machine translation. In ICSE. IEEE, 25–36.

ACM16: Alexander LeClair, Siyuan Jiang, and Collin McMillan. 2019. A neural model for generating natural language summaries of program subroutines. In ICSE. IEEE, 795–806.

ACM17: Rabee Sohail Malik, Jibesh Patra, and Michael Pradel. 2019. Nl2type: inferring javascript function types from natural language information. In ICSE. IEEE, 304–315.

ACM18: Aditya Kanade, Petros Maniatis, Gogul Balakrishnan, and Kensen Shi. 2020. Learning and evaluating contextual embedding of source code. In International Conference on Machine Learning. PMLR, 5110–5121.

ACM19: Kui Liu, Dongsun Kim, Tegawendé F Bissyandé, Taeyoung Kim, Kisub Kim, Anil Koyuncu, Suntae Kim, and Yves Le Traon. 2019. Learning to spot and refactor inconsistent method names. In ICSE. IEEE, 1–12.

ACM20: Chris Piech, Jonathan Huang, Andy Nguyen, Mike Phulsuksombati, Mehran Sahami, and Leonidas Guibas. 2015. Learning program embeddings to propagate feedback on student code. In International conference on machine Learning. PMLR, 1093–1102.

ACM21: Jin Guo, Jinghui Cheng, and Jane Cleland-Huang. 2017. Semantically enhanced software traceability using deep learning techniques. In ICSE. IEEE, 3–14.

ACM22: Peng Liu, Xiangyu Zhang, Marco Pistoia, Yunhui Zheng, Manoel Marques, and Lingfei Zeng. 2017. Automatic text input generation for mobile testing. In ICSE. IEEE, 643–653.

ACM23: Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, Kaixuan Wang, and Xudong Liu. 2019. A novel neural source code representation based on abstract syntax tree. In ICSE. IEEE, 783–794.

ACM24: Dehai Zhao, Zhenchang Xing, Chunyang Chen, Xin Xia, and Guoqiang Li. 2019. ActionNet: vision-based workflow action recognition from programming screencasts. In ICSE. IEEE, 350–361.

ACM25: Minxue Pan, An Huang, Guoxin Wang, Tian Zhang, and Xuandong Li. 2020. Reinforcement learning based curiosity-driven testing of Android applications. In ISSTA. 153–164.

ACM26: Muyang Liu, Ke Li, and Tao Chen. 2020. DeepSQLi: Deep Semantic Learning for Testing SQL Injection. arXiv preprint arXiv:2005.11728 (2020).

ACM27: Thibaud Lutellier, Hung Viet Pham, Lawrence Pang, Yitong Li, Moshi Wei, and Lin Tan. 2020. CoCoNuT: combining context-aware neural translation models using ensemble for program repair. In ISSTA. 101–114.

ACM28: Chunrong Fang, Zixi Liu, Yangyang Shi, Jeff Huang, and Qingkai Shi. 2020. Functional code clone detection with syntax and semantics fusion learning. In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. 516–527.

ACM29: Xia Li, Wei Li, Yuqun Zhang, and Lingming Zhang. 2019. Deepfl: Integrating multiple fault diagnosis dimensions for deep fault localization. In ISSTA. 169–180.

ACM30: Tien-Duy B Le and David Lo. 2018. Deep specification mining. In Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. 106–117.

ACM31: Chris Cummins, Pavlos Petoumenos, Alastair Murray, and Hugh Leather. 2018. Compiler fuzzing through deep learning. In ISSTA. 95–105.

ACM32: Yangguang Li, Zhen Ming Jiang, Heng Li, Ahmed E Hassan, Cheng He, Ruirui Huang, Zhengda Zeng, Mian Wang, and Pinan Chen. 2020. Predicting Node Failures in an Ultra-large-scale Cloud Computing Platform: an AIOps Solution. TOSEM 29, 2 (2020), 1–24.

ACM33: Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xin Xia, Liming Zhu, John Grundy, and Jinshui Wang. 2020. Wireframe-based UI design search through image autoencoder. TOSEM 29, 3 (2020), 1–31.

ACM34: Xiaoxue Ren, Zhenchang Xing, Xin Xia, David Lo, Xinyu Wang, and John Grundy. 2019. Neural network-based detection of self-admitted technical debt: from performance to explainability. TOSEM 28, 3 (2019), 1–45.

ACM35: Michele Tufano, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, Martin White, and Denys Poshyvanyk. 2019. An empirical study on learning bug-fixing patches in the wild via neural machine translation. TOSEM 28, 4 (2019), 1–29.

ACM36: Lingfeng Bao, Zhenchang Xing, Xin Xia, David Lo, Minghui Wu, and Xiaohu Yang. 2020. psc2code: Denoising Code Extraction from Programming Screencasts. TOSEM 29, 3 (2020), 1–38.

ACM37: Yi Li, Shaohua Wang, Tien N Nguyen, and Son Van Nguyen. 2019. Improving bug detection via context-based code representation learning and attention-based neural networks. Proceedings of the ACM on Programming Languages 3, OOPSLA (2019), 1–30.

ACM38: Yu Wang, Ke Wang, Fengjuan Gao, and Linzhang Wang. 2020. Learning semantic program embeddings with graph interval neural network. Proceedings of the ACM on Programming Languages 4, OOPSLA (2020), 1–27.

ACM39: Miltiadis Allamanis, Earl T Barr, Soline Ducousso, and Zheng Gao. 2020. Typilus: Neural type hints. In Proceedings of the 41st acm sigplan conference on programming language design and implementation. 91–105.
EL01: Passakorn Phannachitta. 2020. On an optimal analogy-based software effort estimation. IST (2020), 106330.
EL02: Mirosław Ochodek, Sylwia Kopczynska, and Miroslaw Staron. 2020. Deep learning model for end-to-end approximation of COSMIC functional size based on use-case names. IST (2020), 106310.
EL03: Junfeng Tian, Wenjing Xing, and Zhen Li. 2020. BVDetector: A program slice-based binary code vulnerability intelligent detection system. IST 123 (2020), 106289.
EL04: Junfeng Tian and Yuhui Guo. 2020. Software trustworthiness evaluation model based on a behaviour trajectory matrix. IST 119 (2020), 106233.
EL05: Pingyi Zhou, Jin Liu, Xiao Liu, Zijiang Yang, and John Grundy. 2019. Is deep learning better than traditional approaches in tag recommendation for software information sites? IST 109 (2019), 1–13.
EL06: Tianchi Zhou, Xiaobing Sun, Xin Xia, Bin Li, and Xiang Chen. 2019. Improving defect prediction with deep forest. IST 114 (2019), 204–216.
EL07: Yan Xiao, Jacky Keung, Kwabena E Bennin, and Qing Mi. 2019. Improving bug localization with word embedding and enhanced convolutional neural networks. IST 105 (2019), 17–29.
EL08: Qing Mi, Jacky Keung, Yan Xiao, Solomon Mensah, and Yujin Gao. 2018. Improving code readability classification using convolutional neural networks. IST 104 (2018), 60–71.
EL09: Aysh Al-Hroob, Ayad Tareq Imam, and Rawan Al-Heisa. 2018. The use of artificial neural networks for extracting actions and actors from requirements document. IST 101 (2018), 1–15.
EL10: Haonan Tong, Bin Liu, and Shihai Wang. 2018. Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning. IST 96 (2018), 94–111.
EL11: Sukhpal Singh Gill, Shreshth Tuli, Adel Nadjaran Toosi, Felix Cuadrado, Peter Garraghan, Rami Bahsoon, Hanan Lutfiyya, Rizos Sakellariou, Omer Rana, Schahram Dustdar, et al. 2020. ThermoSim: Deep learning based framework for modeling and simulation of thermal-aware resource management for cloud computing environments. JSS (2020), 110596.
EL12: Cheng Zhou, Bin Li, and Xiaobing Sun. 2020. Improving software bug-specific named entity recognition with deep neural network. JSS (2020), 110572.
EL13: Zhen Ni, Bin Li, Xiaobing Sun, Tianhao Chen, Ben Tang, and Xinchen Shi. 2020. Analyzing bug fix for automatic bug cause classification. JSS 163 (2020), 110538.
EL14: Hang Ruan, Bihuan Chen, Xin Peng, and Wenyun Zhao. 2019. DeepLink: Recovering issue-commit links based on deep learning. JSS 158 (2019), 110406.
EL15: Zhou Xu, Shuai Li, Jun Xu, Jin Liu, Xiapu Luo, Yifeng Zhang, Tao Zhang, Jacky Keung, and Yutian Tang. 2019. LDFR: Learning deep feature representation for software defect prediction. JSS 158 (2019), 110402.
EL16: Yasir Hussain, Zhiqiu Huang, Yu Zhou, and Senzhang Wang. 2020. CodeGRU: Context-aware deep learning with gated recurrent unit for source code modeling. IST (2020), 106309.
EL17: Ruibo Yan, Xi Xiao, Guangwu Hu, Sancheng Peng, and Yong Jiang. 2018. New deep learning method to detect code injection attacks on hybrid applications. JSS 137 (2018), 67–77.
EL18: Lov Kumar and Santanu Ku Rath. 2016. Hybrid functional link artificial neural network approach for predicting maintainability of object-oriented software. JSS 121 (2016), 170–190.
EL19: Cuauhtémoc López-Martín and Alain Abran. 2015. Neural networks for predicting the duration of new software projects. JSS 101 (2015), 127–135.
EL20: Fang Liu, Lu Zhang, and Zhi Jin. 2020. Modeling programs hierarchically with stack-augmented LSTM. Journal of Systems and Software 164 (2020), 110547.
EL21: Yu Zhou, Xin Yan, Wenhua Yang, Taolue Chen, and Zhiqiu Huang. 2019. Augmenting Java method comments generation with context information based on neural networks. JSS 156 (2019), 328–340.

ICLR01: Elizabeth Dinella, Hanjun Dai, Ziyang Li, Mayur Naik, Le Song, and Ke Wang. 2020. Hoppity: Learning graph transformations to detect and fix bugs in programs. In International Conference on Learning Representations (ICLR).

ICLR02: Marko Vasic, Aditya Kanade, Petros Maniatis, David Bieber, and Rishabh Singh. 2019. Neural Program Repair by Jointly Learning to Localize and Repair. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

ICLR03: Xinyun Chen, Chang Liu, and Dawn Song. 2018. Execution-guided neural program synthesis. In International Conference on Learning Representations.

ICLR04: Tue Le, Tuan Nguyen, Trung Le, Dinh Phung, Paul Montague, Olivier De Vel, and Lizhen Qu. 2018. Maximal divergence sequential autoencoder for binary software vulnerability detection. In International Conference on Learning Representations.

ICLR05: Ke Wang, Rishabh Singh, and Zhendong Su. 2018. Dynamic Neural Program Embeddings for Program Repair. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018. OpenReview.net

ICLR06: Zhan Shi, Kevin Swersky, Daniel Tarlow, Parthasarathy Ranganathan, and Milad Hashemi. 2020. Learning Execution through Neural Code fusion. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

ICLR07: Uri Alon, Shaked Brody, Omer Levy, and Eran Yahav. 2019. code2seq: Generating Sequences from Structured Representations of Code. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

OCLR08: Xinyun Chen, Chang Liu, and Dawn Song. 2018. Towards Synthesizing Complex Programs From Input-Output Examples. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018. OpenReview.net

ICLR09: Rudy Bunel, Matthew J. Hausknecht, Jacob Devlin, Rishabh Singh, and Pushmeet Kohli. 2018. Leveraging Grammar and Reinforcement Learning for Neural Program Synthesis. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018. OpenReview.net.

ICLR10: Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. 2018. Learning to Represent Programs with Graphs. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net.

ICLR11: Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. 2017. DeepCoder: Learning to Write Programs. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017. OpenReview.net

IEEE01: Yao Wan, Jingdong Shu, Yulei Sui, Guandong Xu, Zhou Zhao, Jian Wu, and Philip Yu. 2019. Multi-modal attention network learning for semantic source code retrieval. In ASE. IEEE, 13–25.

IEEE02: Qihao Zhu, Zeyu Sun, Xiran Liang, Yingfei Xiong, and Lu Zhang. 2020. OCoR: an overlapping-aware code retriever. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 883–894.

IEEE03: Cuiyun Gao, Jichuan Zeng, Xin Xia, David Lo, Michael R Lyu, and Irwin King. 2019. Automating app review response generation. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 163–175.

IEEE04: Zhongxin Liu, Xin Xia, Christoph Treude, David Lo, and Shanping Li. 2019. Automatic generation of pull request descriptions. In ASE. IEEE, 176–188.

IEEE05: Junjie Chen, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2019. Continuous incident triage for large-scale online service systems. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 364–375.

IEEE06: Jeremy Lacomis, Pengcheng Yin, Edward Schwartz, Miltiadis Allamanis, Claire Le Goues, Graham Neubig, and Bogdan Vasilescu. 2019. Dire: A neural approach to decompiled identifier naming. In ASE. IEEE, 628–639.

IEEE07: Yan Zheng, Xiaofei Xie, Ting Su, Lei Ma, Jianye Hao, Zhaopeng Meng, Yang Liu, Ruimin Shen, Yingfeng Chen, and Changjie Fan. 2019. Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning. In ASE. IEEE, 772–784.

IEEE08: Dongliang Mu, Wenbo Guo, Alejandro Cuevas, Yueqi Chen, Jinxuan Gai, Xinyu Xing, Bing Mao, and Chengyu Song. 2019. RENN: efficient reverse execution with neural-network-assisted alias analysis. In ASE. IEEE, 924–935.

IEEE09: Kawser Wazed Nafi, Tonny Shekha Kar, Banani Roy, Chanchal K Roy, and Kevin A Schneider. 2019. CLCDSA: cross language code clone detection using syntactical features and API documentation. In ASE. IEEE, 1026–1037.

IEEE10: Hui Liu, Zhifeng Xu, and Yanzhen Zou. 2018. Deep learning based feature envy detection. In ASE. 385–396.

IEEE11: Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically generating commit messages from diffs using neural machine translation. In ASE. IEEE, 135–146.

IEEE12: Patrice Godefroid, Hila Peleg, and Rishabh Singh. 2017. Learn&fuzz: Machine learning for input fuzzing. In ASE. IEEE, 50–59.

IEEE13: Bowen Xu, Deheng Ye, Zhenchang Xing, Xin Xia, Guibin Chen, and Shanping Li. 2016. Predicting semantically linkable knowledge in developer online forums via convolutional neural network. In ASE. IEEE, 51–62.

IEEE14: Raja Ben Abdessalem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. 2016. Testing advanced driver assistance systems using multi-objective search and neural networks. In ASE. 63–74.

IEEE15: Martin White, Michele Tufano, Christopher Vendome, and Denys Poshyvanyk. 2016. Deep learning code fragments for code clone detection. In ASE. IEEE, 87–98.

IEEE16: Guibin Chen, Chunyang Chen, Zhenchang Xing, and Bowen Xu. 2016. Learning a dual-language vector space for domain-specific cross-lingual question retrieval. In ASE. IEEE, 744–755.

IEEE17: Devjeet Roy, Ziyi Zhang, Maggie Ma, Venera Arnaoudova, Annibale Panichella, Sebastiano Panichella, Danielle Gonzalez, and Mehdi Mirakhorli. 2020. DeepTC-Enhancer: Improving the readability of automatically generated tests. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 287–298.

IEEE18: Mohammad Jafar Mashhadi and Hadi Hemmati. 2020. Hybrid deep neural networks to infer state models of black-box systems. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 299–311.

IEEE19: Mingyang Li, Lin Shi, Ye Yang, and Qing Wang. 2020. A deep multitask learning approach for requirements discovery and annotation from open forum. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 336–348.

IEEE20: Xiaodong Gu, Hongyu Zhang, and Sunghun Kim. 2018. Deep code search. In ICSE. IEEE, 933–944.

IEEE21: Zhenhao Li, Tse-Hsun Chen, and Weiyi Shang. 2020. Where shall we log? studying and suggesting logging locations in code blocks. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. 361–372.

IEEE22: Junjie Chen, Shu Zhang, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Yu Kang, Feng Gao, Zhangwei Xu, Yingnong Dang, et al. 2020. How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. 373–384.

IEEE23: Zhe Liu, Chunyang Chen, Junjie Wang, Yuekai Huang, Jun Hu, and Qing Wang. 2020. Owl eyes: Spotting ui display issues via visual understanding. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 398–409.

IEEE24: Wei Li, Haozhe Qin, Shuhan Yan, Beijun Shen, and Yuting Chen. 2020. Learning Code-Query Interaction for Enhancing Code Searches. In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 115–126.

IEEE25: Qing Huang, An Qiu, Maosheng Zhong, and Yuan Wang. 2020. A Code-Description Representation Learning Model Based on Attention. In SANER. IEEE, 447–455.

IEEE26: Chunyang Ling, Zeqi Lin, Yanzhen Zou, and Bing Xie. 2020. Adaptive deep code search. In Proceedings of the 28th International Conference on Program Comprehension. 48–59.

IEEE27: Yueming Wu, Deqing Zou, Shihan Dou, Siru Yang, Wei Yang, Feng Cheng, Hong Liang, and Hai Jin. 2020. SCDetector: software functional clone detection based on semantic tokens analysis. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. 821–833.

IEEE28: Xin Wang, Jin Liu, Li Li, Xiao Chen, Xiao Liu, and Hao Wu. 2020. Detecting and explaining self-admitted technical debts with attention-based neural networks. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. 871–882.

IEEE29: Haoye Tian, Kui Liu, Abdoul Kader Kaboré, Anil Koyuncu, Li Li, Jacques Klein, and Tegawendé F Bissyandé. 2020. Evaluating representation learning of code changes for predicting patch correctness in program repair. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 981–992.

IEEE30: Lian Yu, Lihao Chen, Jingtao Dong, Mengyuan Li, Lijun Liu, Bai Zhao, and Chen Zhang. 2020. Detecting Malicious Web Requests Using an Enhanced TextCNN. In 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, 768–777.

IEEE31: Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S Yu. 2018. Improving automatic source code summarization via deep reinforcement learning. In ASE. 397–407.

IEEE32: Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, Yanjun Pu, and Xudong Liu. 2020. Learning to handle exceptions. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 29–41.

IEEE33: Bolin Wei, Yongmin Li, Ge Li, Xin Xia, and Zhi Jin. 2020. Retrieve and refine: exemplar-based neural comment generation. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 349–360.

IEEE34: Fang Liu, Ge Li, Yunfei Zhao, and Zhi Jin. 2020. Multi-task learning based pre-trained language model for code completion. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. 473–485.

IEEE35: Zhongxin Liu, Xin Xia, Meng Yan, and Shanping Li. 2020. Automating just-in-time comment updating. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. 585–597.

IEEE36: David Gros, Hariharan Sezhiyan, Prem Devanbu, and Zhou Yu. 2020. Code to Comment "Translation": Data, Metrics, Baselining & Evaluation. In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 746–757.

IEEE37: Yating Zhang, Wei Dong, Daiyan Wang, Binbin Liu, and Jiaxin Liu. 2020. Accuracy Improvement for Neural Program Synthesis via Attention Mechanism and Program Slicing. In 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, 963–972.

IEEE38: Hao Yu, Wing Lam, Long Chen, Ge Li, Tao Xie, and Qianxiang Wang. 2019. Neural detection of semantic code clones via tree-based convolution. In ICPC. IEEE, 70–80.

IEEE39: Xiaochen Li, He Jiang, Dong Liu, Zhilei Ren, and Ge Li. 2018. Unsupervised deep bug report summarization. In ICPC. IEEE, 144–14411.

IEEE40: An Ngoc Lam, Anh Tuan Nguyen, Hoan Anh Nguyen, and Tien N Nguyen. 2017. Bug localization with combination of deep learning and information retrieval. In ICPC. IEEE, 218–229.
IEEE41: Chunyang Chen, Ting Su, Guozhu Meng, Zhenchang Xing, and Yang Liu. 2018. From ui design image to gui skeleton: a neural machine translator to bootstrap mobile gui implementation. In ICSE. 665–676.
IEEE42: Sahil Bhatia, Pushmeet Kohli, and Rishabh Singh. 2018. Neuro-symbolic program corrector for introductory programming assignments. In ICSE. IEEE, 60–70.
IEEE43: Song Wang, Taiyue Liu, and Lin Tan. 2016. Automatically learning semantic features for defect prediction. In ICSE. IEEE, 297–308.
IEEE44: Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xiwei Xu, Liming Zhut, Guoqiang Li, and Jinshui Wang. 2020. Unblind your apps: Predicting natural-language labels for mobile gui components by deep learning. In 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). IEEE, 322–334.
IEEE45: Thong Hoang, Hong Jin Kang, David Lo, and Julia Lawall. 2020. Cc2vec: Distributed representations of code changes. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 518–529.
IEEE46: Jinyin Chen, Keke Hu, Yue Yu, Zhuangzhi Chen, Qi Xuan, Yi Liu, and Vladimir Filkov. 2020. Software visualization and deep transfer learning for effective software defect prediction. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 578–589.
IEEE47: Yi Li, Shaohua Wang, and Tien N Nguyen. 2020. Dlfix: Context-based code transformation learning for automated program repair. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 602–614.
IEEE48: Lin Shi, Mingzhe Xing, Mingyang Li, Yawen Wang, Shoubin Li, and Qing Wang. 2020. Detection of hidden feature requests from massive chat messages via deep siamese network. In 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). IEEE, 641–653.
IEEE49: Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018. Deep code comment generation. In ICPC. IEEE, 200–20010.
IEEE50: Antoine Barbez, Foutse Khomh, and Yann-Gaël Guéhéneuc. 2019. Deep Learning Anti-patterns from Code Metrics History. In ICSME. IEEE, 114–124.
IEEE51: Yi Gao, Zan Wang, Shuang Liu, Lin Yang, Wei Sang, and Yuanfang Cai. 2019. TECCD: A Tree Embedding Approach for Code Clone Detection. In ICSME. IEEE, 145–156.
IEEE52: Chao Chen, Wenrui Diao, Yingpei Zeng, Shanqing Guo, and Chengyu Hu. 2018. DRLgencert: Deep learning-based automated testing of certificate verification in SSL/TLS implementations. In ICSME. IEEE, 48–58.
IEEE53: Alexander LeClair, Zachary Eberhart, and Collin McMillan. 2018. Adapting neural text classification for improved software categorization. In ICSME. IEEE, 461–472.
IEEE54: Jayati Deshmukh, KM Annervaz, Sanjay Podder, Shubhashis Sengupta, and Neville Dubash. 2017. Towards accurate duplicate bug retrieval using deep learning techniques. In ICSME. IEEE, 115–124.
IEEE55: Liuqing Li, He Feng, Wenjie Zhuang, Na Meng, and Barbara Ryder. 2017. Cclearner: A deep learning-based clone detection approach. In ICSME. IEEE, 249–260.
IEEE56: Stephen Romansky, Neil C Borle, Shaiful Chowdhury, Abram Hindle, and Russ Greiner. 2017. Deep green: Modelling time-series of software energy consumption. In ICSME. IEEE, 273–283.
IEEE57: Zhuobing Han, Xiaohong Li, Zhenchang Xing, Hongtao Liu, and Zhiyong Feng. 2017. Learning to predict severity of software vulnerability using only vulnerability description. In ICSME. IEEE, 125–136.
IEEE58: Mingyang Li, Ye Yang, Lin Shi, Qing Wang, Jun Hu, Xinhua Peng, Weimin Liao, and Guizhen Pi. 2020. Automated Extraction of Requirement Entities by Leveraging LSTM-CRF and Transfer Learning. In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 208–219.
IEEE59: Son Nguyen, Hung Phan, Trinh Le, and Tien N Nguyen. 2020. Suggesting natural method names to check name consistencies. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 1372–1384.

IEEE60: Wei Zhang. 2020. Efficient Bug Triage For Industrial Environments. In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 727–735.

IEEE61: Hui Zhao, Zhihui Li, Hansheng Wei, Jianqi Shi, and Yanhong Huang. 2019. SeqFuzzer: An industrial protocol fuzzing framework from a deep learning perspective. In ICST. IEEE, 59–67.

IEEE62: Seongmin Lee, Shin Hong, Jungbae Yi, Taeksu Kim, Chul-Joo Kim, and Shin Yoo. 2019. Classifying false positive static checker alarms in continuous integration using convolutional neural networks. In ICST. IEEE, 391–401.

IEEE63: Jinkyu Koo, Charitha Saumya, Milind Kulkarni, and Saurabh Bagchi. 2019. Pyse: Automatic worst-case test generation by reinforcement learning. In ICST. IEEE, 136–147.

IEEE64: Ugur Koc, Shiyi Wei, Jeffrey S Foster, Marine Carpuat, and Adam A Porter. 2019. An empirical assessment of machine learning approaches for triaging reports of a java static analysis tool. In ICST. IEEE, 288–299.

IEEE65: Dongyu Mao, Lingchao Chen, and Lingming Zhang. 2019. An extensive study on cross-project predictive mutation testing. In ICST. IEEE, 160–171.

IEEE66: Cody Watson, Michele Tufano, Kevin Moran, Gabriele Bavota, and Denys Poshyvanyk. 2020. On learning meaningful assert statements for unit test cases. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 1398–1409.

IEEE67: Davide Pizzolotto and Katsuro Inoue. 2020. Identifying Compiler and Optimization Options from Binary Code using Deep Learning Approaches. In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 232–242.

IEEE68: Xiaohui Wan, Zheng Zheng, Fangyun Qin, Yu Qiao, and Kishor S Trivedi. 2019. Supervised Representation Learning Approach for Cross-Project Aging-Related Bug Prediction. In ISSRE. IEEE, 163–172.

IEEE69: Yaohui Wang, Hui Xu, Yangfan Zhou, Michael R Lyu, and Xin Wang. 2019. Textout: Detecting Text-Layout Bugs in Mobile Apps via Visualization-Oriented Learning. In ISSRE. IEEE, 239–249.

IEEE70: Tianyi Zhang, Cuiyun Gao, Lei Ma, Michael Lyu, and Miryung Kim. 2019. An empirical study of common challenges in developing deep learning applications. In ISSRE. IEEE, 104–115.

IEEE71: Irving Muller Rodrigues, Daniel Aloise, Eraldo Rezende Fernandes, and Michel Dagenais. 2020. A soft alignment model for bug deduplication. In Proceedings of the 17th International Conference on Mining Software Repositories. 43–53.

IEEE72: Timofey Bryksin, Victor Petukhov, Ilya Alexin, Stanislav Prikhodko, Alexey Shpilman, Vladimir Kovalenko, and Nikita Povarov. 2020. Using large-scale anomaly detection on code to improve kotlin compiler. In Proceedings of the 17th International Conference on Mining Software Repositories. 455–465.

IEEE73: Thong Hoang, Hoa Khanh Dam, Yasutaka Kamei, David Lo, and Naoyasu Ubayashi. 2019. DeepJIT: an end-to-end deep learning framework for just-in-time defect prediction. In MSR. IEEE, 34–45.

IEEE74: Hoa Khanh Dam, Trang Pham, Shien Wee Ng, Truyen Tran, John Grundy, Aditya Ghose, Taeksu Kim, and Chul-Joo Kim. 2019. Lessons learned from using a deep tree-based model for software defect prediction in practice. In MSR. IEEE, 46–57.

IEEE75: Qin Liu, Zihe Liu, Hongming Zhu, Hongfei Fan, Bowen Du, and Yu Qian. 2019. Generating commit messages from diffs using pointer-generator network. In 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR). IEEE, 299–309.

IEEE76: Shaohua Wang, NhatHai Phan, Yan Wang, and Yong Zhao. 2019. Extracting API tips from developer question and answer websites. In MSR. IEEE, 321–332.

IEEE77: Daniel Perez and Shigeru Chiba. 2019. Cross-language clone detection by learning over abstract syntax trees. In MSR. IEEE, 518–528.

IEEE78: Jordan Ott, Abigail Atchison, Paul Harnack, Adrienne Bergh, and Erik Linstead. 2018. A deep learning approach to identifying source code in images and video. In MSR. IEEE, 376–386.

IEEE79: Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. Learning to mine aligned code and natural language pairs from stack overflow. In MSR. IEEE, 476–486.

IEEE80: Michele Tufano, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, Martin White, and Denys Poshyvanyk. 2018. Deep learning similarities from different representations of source code. In MSR. IEEE, 542–553.

IEEE81: Tim Menzies, Suvodeep Majumder, Nikhila Balaji, Katie Brey, and Wei Fu. 2018. 500+ times faster than deep learning:(a case study exploring faster methods for text mining stackoverflow). In MSR. IEEE, 554–563.

IEEE82: Rhys Compton, Eibe Frank, Panos Patros, and Abigail Koay. 2020. Embedding java classes with code2vec: Improvements from variable obfuscation. In Proceedings of the 17th International Conference on Mining Software Repositories. 243–253.

IEEE83: Martin White, Christopher Vendome, Mario Linares-Vásquez, and Denys Poshyvanyk. 2015. Toward deep learning software repositories. In MSR. IEEE, 334–345.

IEEE84: Xinli Yang, David Lo, Xin Xia, Yun Zhang, and Jianling Sun. 2015. Deep learning for just-in-time defect prediction. In 2015 IEEE International Conference on Software Quality, Reliability and Security. IEEE, 17–26.

IEEE85: Jian Li, Pinjia He, Jieming Zhu, and Michael R Lyu. 2017. Software defect prediction via convolutional neural network. In 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS). IEEE, 318–328.

IEEE86: Caesar Jude Clemente, Fehmi Jaafar, and Yasir Malik. 2018. Is predicting software security bugs using deep learning better than the traditional machine learning algorithms?. In 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS). IEEE, 95–102.

IEEE87: Xian Zhang, Kerong Ben, and Jie Zeng. 2018. Cross-entropy: A new metric for software defect prediction. In 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS). IEEE, 111–122.

IEEE88: Wensheng Xia, Ying Li, Tong Jia, and Zhonghai Wu. 2019. BugIdentifier: An Approach to Identifying Bugs via Log Mining for Accelerating Bug Reporting Stage. In 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS). IEEE, 167–175.

IEEE89: Shasha Cheng, Xuefeng Yan, and Arif Ali Khan. 2020. A Similarity Integration Method based Information Retrieval and Word Embedding in Bug Localization. In 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS). IEEE, 180–187.

IEEE90: Kunsong Zhao, Zhou Xu, Tao Zhang, Yutian Tang, and Meng Yan. 2021. Simplified Deep Forest Model Based Just-in-Time Defect Prediction for Android Mobile Apps. IEEE Transactions on Reliability (2021).

IEEE91: Wu Jiang, Xu Jianjun, Meng Xiankai, Zhang Zhuo, Zhang Nan, and Zhang Haoyu. 2020. High-Reliability Compilation Optimization Sequence Generation Framework Based ANN. In 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS). IEEE, 347–355.

IEEE92: Xuan Zhou and Lu Lu. 2020. Defect Prediction via LSTM Based on Sequence and Tree Structure. In 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS). IEEE, 366–373.

IEEE93: Zijie Li, Long Zhang, Jun Yan, Jian Zhang, Zhenyu Zhang, and TH Tse. 2020. PEACEPACT: Prioritizing Examples to Accelerate Perturbation-Based Adversary Generation for DNN Classification Testing. In 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS). IEEE, 406–413.

IEEE94: Florian Pudlitz, Florian Brokhausen, and Andreas Vogelsang. 2019. Extraction of system states from natural language requirements. In RE. IEEE, 211–222.

IEEE95: Jonas Paul Winkler, Jannis Grönberg, and Andreas Vogelsang. 2019. Predicting How to Test Requirements: An Automated Approach. In RE. IEEE, 120–130.

IEEE96: Wentao Wang, Nan Niu, Hui Liu, and Zhendong Niu. 2018. Enhancing automated requirements traceability by resolving polysemy. In RE. IEEE, 40–51.

IEEE97: Alvi Mahadi, Karan Tongay, and Neil A Ernst. 2020. Cross-Dataset Design Discussion Mining. In SANER. IEEE, 149–160.

IEEE98: Wenhan Wang, Ge Li, Bo Ma, Xin Xia, and Zhi Jin. 2020. Detecting Code Clones with Graph Neural Network and Flow-Augmented Abstract Syntax Tree. In SANER. IEEE, 261–271.
IEEE99: Jing Kai Siow, Cuiyun Gao, Lingling Fan, Sen Chen, and Yang Liu. 2020. CORE: Automating Review Recommendation for Code Changes. In SANER. IEEE, 284–295.
IEEE100: Fiorella Zampetti, Alexander Serebrenik, and Massimiliano Di Penta. 2020. Automatically learning patterns for self-admitted technical debt removal. In SANER. IEEE, 355–366.
IEEE101: Zachary Eberhart, Alexander LeClair, and Collin McMillan. 2020. Automatically Extracting Subroutine Summary Descriptions from Unstructured Comments. In SANER. IEEE, 35–46.
IEEE102: Guangjie Li, Hui Liu, Jiahao Jin, and Qasim Umer. 2020. Deep Learning Based Identification of Suspicious Return Statements. In SANER. IEEE, 480–491.
IEEE103: Zhuo Zhang, Yan Lei, Xiaoguang Mao, and Panpan Li. 2019. CNN-FL: An effective approach for localizing faults using convolutional neural networks. In SANER. IEEE, 445–455.
IEEE104: Bui Nghi DQ, Yijun Yu, and Lingxiao Jiang. 2019. Bilateral dependency neural networks for cross-language algorithm classification. In SANER. IEEE, 422–433.
IEEE105: Martin White, Michele Tufano, Matias Martinez, Martin Monperrus, and Denys Poshyvanyk. 2019. Sorting and transforming program repair ingredients via deep learning code similarities. In SANER. IEEE, 479–490.
IEEE106: Chenkai Guo, Dengrong Huang, Naipeng Dong, Quanqi Ye, Jing Xu, Yaqing Fan, Hui Yang, and Yifan Xu. 2019. Deep review sharing. In SANER. IEEE, 61–72.
IEEE107: Rui Xie, Long Chen, Wei Ye, Zhiyu Li, Tianxiang Hu, Dongdong Du, and Shikun Zhang. 2019. DeepLink: A code knowledge graph based deep learning approach for issue-commit link recovery. In SANER. IEEE, 434–444.
IEEE108: Adelina Ciurumelea, Sebastian Proksch, and Harald C Gall. 2020. Suggesting Comment Completions for Python using Neural Language Models. In SANER. IEEE, 456–467.
IEEE109: Hannes Thaller, Lukas Linsbauer, and Alexander Egyed. 2019. Feature maps: A comprehensible software representation for design pattern detection. In SANER. IEEE, 207–217.
IEEE110: Lutz Büch and Artur Andrzejak. 2019. Learning-based recursive aggregation of abstract syntax trees for code clone detection. In SANER. IEEE, 95–104.
IEEE111: Chenkai Guo, Weijing Wang, Yanfeng Wu, Naipeng Dong, Quanqi Ye, Jing Xu, and Sen Zhang. 2019. Systematic comprehension for developer reply in mobile system forum. In SANER. IEEE, 242–252.
IEEE112: Sa Gao, Chunyang Chen, Zhenchang Xing, Yukun Ma, Wen Song, and Shang-Wei Lin. 2019. A neural model for method name generation from functional description. In SANER. IEEE, 414–421.
IEEE113: Deborah S Katz, Jason Ruchti, and Eric Schulte. 2018. Using recurrent neural networks for decompilation. In SANER. IEEE, 346–356.
IEEE114: Yibin Liu, Yanhui Li, Jianbo Guo, Yuming Zhou, and Baowen Xu. 2018. Connecting software metrics across versions to predict defects. In SANER. IEEE, 232–243.
IEEE115: Eddie Antonio Santos, Joshua Charles Campbell, Dhvani Patel, Abram Hindle, and José Nelson Amaral. 2018. Syntax and sensibility: Using language models to detect and correct syntax errors. In SANER. IEEE, 311–322.
IEEE116: Sarah Fakhoury, Venera Arnaoudova, Cedric Noiseux, Foutse Khomh, and Giuliano Antoniol. 2018. Keep it simple: Is deep learning good for linguistic smell detection?. In SANER. IEEE, 602–611.
IEEE117: Stefan Strüder, Mukelabai Mukelabai, Daniel Strüber, and Thorsten Berger. 2020. Feature-oriented defect prediction. In Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A-Volume A. 1–12.
IEEE118: Yang Li. 2018. Feature and variability extraction from natural language software requirements specifications. In Proceedings of the 22nd International Systems and Software Product Line Conference-Volume 2. 72–78.

IEEE119: Yang Li, Sandro Schulze, Helene Hvidegaard Scherrebeck, and Thomas Sorensen Fogdal. 2020. Automated extraction of domain knowledge in practice: the case of feature extraction from requirements at danfoss. In Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A-Volume A. 1–11.

IEEE120: Yang Li, Sandro Schulze, and Gunter Saake. 2018. Reverse engineering variability from requirement documents based on probabilistic relevance and word embedding. In Proceedings of the 22nd International Systems and Software Product Line Conference-Volume 1. 121–131.

IEEE121: Anh Tuan Nguyen, Trong Duc Nguyen, Hung Dang Phan, and Tien N Nguyen. 2018. A deep neural network language model with contexts for source code. In SANER. IEEE, 323–334.

IEEE122: Hui Liu, Jiahao Jin, Zhifeng Xu, Yifan Bu, Yanzhen Zou, and Lu Zhang. 2019. Deep learning based code smell detection. TSE (2019).

IEEE123: Thong Hoang, Julia Lawall, Yuan Tian, Richard J Oentaryo, and David Lo. 2019. PatchNet: Hierarchical Deep Learning-Based Stable Patch Identification for the Linux Kernel. TSE (2019).

IEEE124: Xuan Huo, Ferdian Thung, Ming Li, David Lo, and Shu-Ting Shi. 2019. Deep transfer bug localization. TSE (2019).

IEEE125: Suyu Ma, Zhenchang Xing, Chunyang Chen, Cheng Chen, Lizhen Qu, and Guoqiang Li. 2019. Easy-to-Deploy API Extraction by Multi-Level Feature Embedding and Transfer Learning. TSE (2019).

IEEE126: Zhongxin Liu, Xin Xia, David Lo, Zhenchang Xing, Ahmed E Hassan, and Shanping Li. 2019. Which variables should i log? TSE (2019).

IEEE127: Song Wang, Taiyue Liu, Jaechang Nam, and Lin Tan. 2018. Deep semantic feature learning for software defect prediction. TSE (2018).

IEEE128: Ming Wen, Rongxin Wu, and Shing-Chi Cheung. 2018. How well do change sequences predict defects? sequence learning from software changes. TSE (2018).

IEEE129: Kevin Patrick Moran, Carlos Bernal-Cárdenas, Michael Curcio, Richard Bonett, and Denys Poshyvanyk. 2018. Machine learning-based prototyping of graphical user interfaces for mobile apps. TSE (2018).

IEEE130: Qiao Huang, Xin Xia, David Lo, and Gail C Murphy. 2018. Automating intention mining. TSE (2018).

IEEE131: Hoa Khanh Dam, Truyen Tran, Trang Thi Minh Pham, Shien Wee Ng, John Grundy, and Aditya Ghose. 2018. Automatic feature learning for predicting vulnerable software components. TSE (2018).

IEEE132: Kui Liu, Dongsun Kim, Tegawendé F Bissyandé, Shin Yoo, and Yves Le Traon. 2018. Mining fix patterns for findbugs violations. IEEE Transactions on Software Engineering (2018).

IEEE133: Ming Wen, Rongxin Wu, and Shing-Chi Cheung. 2018. How well do change sequences predict defects? sequence learning from software changes. TSE (2018).

IEEE134: Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. 2018. A deep learning model for estimating story points. TSE 45, 7 (2018), 637–656.

IEEE135: Jinglei Zhang, Rui Xie, Wei Ye, Yuhan Zhang, and Shikun Zhang. 2020. Exploiting code knowledge graph for bug localization via bi-directional attention. In Proceedings of the 28th International Conference on Program Comprehension. 219–229.

IEEE136: Alexander LeClair, Sakib Haque, Lingfei Wu, and Collin McMillan. 2020. Improved code summarization via a graph neural network. In Proceedings of the 28th International Conference on Program Comprehension. 184–195.

IEEE137: Haoran Liu, Yue Yu, Shanshan Li, Yong Guo, Deze Wang, and Xiaoguang Mao. 2020. Bugsum: Deep context understanding for bug report summarization. In Proceedings of the 28th International Conference on Program Comprehension. 94–105.

IEEE138: Liwei Wu, Fei Li, Youhua Wu, and Tao Zheng. 2020. Ggf: A graph-based method for programming language syntax error correction. In Proceedings of the 28th International Conference on Program Comprehension. 139–148.

IEEE139: Jianjun He, Ling Xu, Meng Yan, Xin Xia, and Yan Lei. 2020. Duplicate bug report detection using dual-channel convolutional neural networks. In Proceedings of the 28th International Conference on Program Comprehension. 117–127.

IEEE140: Mohammad Alahmadi, Abdulkarim Khormi, and Sonia Haiduc. 2020. UI screens identification and extraction from mobile programming screencasts. In Proceedings of the 28th International Conference on Program Comprehension. 319–330.

IEEE141: Fang Liu, Ge Li, Bolin Wei, Xin Xia, Zhiyi Fu, and Zhi Jin. 2020. A self-attentional neural architecture for code completion with multi-task learning. In Proceedings of the 28th International Conference on Program Comprehension. 37–47.

IEEE142: Jianjun He, Ling Xu, Yuanrui Fan, Zhou Xu, Meng Yan, and Yan Lei. 2020. Deep Learning Based Valid Bug Reports Determination and Explanation. In 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE). IEEE, 184–194.

IET01: Manjubala Bisi and Neeraj Kumar Goyal. 2016. Software development efforts prediction using artificial neural network. IETS 10, 3 (2016), 63–71.

MITP01: Jacob Harer, Onur Ozdemir, Tomo Lazovich, Christopher P. Reale, Rebecca L. Russell, Louis Y. Kim, and Sang Peter Chin. 2018. Learning to Repair Software Vulnerabilities with Generative Adversarial Networks. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 7944–7954.

MITP02: Tal Ben-Nun, Alice Shoshana Jakobovits, and Torsten Hoefler. 2018. Neural Code Comprehension: A Learnable Representation of Code Semantics. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 3589–3601.

MITP03: Yaqin Zhou, Shangqing Liu, Jing Kai Siow, Xiaoning Du, and Yang Liu. 2019. Devign: Effective Vulnerability Identification by Learning Comprehensive Program Semantics via Graph Neural Networks. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 10197–10207

MITP04: Rahul Gupta, Aditya Kanade, and Shirish K. Shevade. 2019. Neural Attribution for Semantic Bug-Localization in Student Programs. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 11861–11871.

MITP05: Zeping Yu, Wenxin Zheng, Jiaqi Wang, Qiyi Tang, Sen Nie, and Shi Wu. 2020. CodeCMR: Cross-Modal Retrieval For Function-Level Binary Source Code Matching. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

MITP06: Eui Chul Richard Shin, Miltiadis Allamanis, Marc Brockschmidt, and Alex Polozov. 2019. Program Synthesis and Semantic Parsing with Learned Code Idioms. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 10824–10834.

MITP07: Bolin Wei, Ge Li, Xin Xia, Zhiyi Fu, and Zhi Jin. 2019. Code Generation as a Dual Task of Code Summarization. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information

Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 6559–6569.

MITP08: Kavi Gupta, Peter Ebert Christensen, Xinyun Chen, and Dawn Song. 2020. Synthesize, Execute and Debug: Learning to Repair for Neural Program Synthesis. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

MK01: Prantik Chatterjee, Abhijit Chatterjee, José Campos, Rui Abreu, Subhajit Roy, A Panichella, G Fraser, D Paterson, GM Kapfhammer, P McMinn, et al. 2020. Diagnosing Software Faults Using Multiverse Analysis.. In IJCAI. 1629–1635.

MK02: Shengbin Xu, Yuan Yao, Feng Xu, Tianxiao Gu, Hanghang Tong, and Jian Lu. 2019. Commit message generation for source code changes. In IJCAI.

MK03: Xu Duan, Jingzheng Wu, Shouling Ji, Zhiqing Rui, Tianyue Luo, Mutian Yang, and Yanjun Wu. 2019. VulSniper: Focus Your Attention to Shoot Fine-Grained Vulnerabilities.. In IJCAI. 4665–4671.

MK04: Huihui Wei and Ming Li. 2018. Positive and Unlabeled Learning for Detecting Software Functional Clones with Adversarial Training. In IJCAI. 2840–2846.

MK05: Huihui Wei and Ming Li. 2017. Supervised Deep Features for Software Functional Clone Detection by Exploiting Lexical and Syntactical Information in Source Code.. In IJCAI. 3034–3040.

MK06: Xuan Huo and Ming Li. 2017. Enhancing the Unified Features to Locate Buggy Files by Exploiting the Sequential Nature of Source Code. In IJCAI. 1909–1915.

MK07: Xuan Huo, Ming Li, Zhi-Hua Zhou, et al. 2016. Learning unified features from natural and programming languages for locating buggy source code.. In IJCAI, Vol. 16. 1606–1612.

MK08: Xing Hu, Ge Li, Xin Xia, David Lo, Shuai Lu, and Zhi Jin. 2018. Summarizing Source Code with Transferred API Knowledge. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, Jérôme Lang (Ed.). ijcai.org, 2269–2275.

MK09: Jian Li, Yue Wang, Michael R. Lyu, and Irwin King. 2018. Code Completion with Neural Attention and Pointer Networks. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, Jérôme Lang (Ed.). ijcai.org, 4159–4165.

SP01: Jin Liu, Pingyi Zhou, Zijiang Yang, Xiao Liu, and John Grundy. 2018. FastTagRec: fast tag recommendation for software information sites. ASEJ 25, 4 (2018), 675–701.

SP02: Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Chaiyong Ragkhitwetsagul, and Aditya Ghose. 2021. Automatically recommending components for issue reports using deep learning. Empirical Software Engineering 26, 2 (2021), 1–39.

SP03: Xu Wang, Chunyang Chen, and Zhenchang Xing. 2019. Domain-specific machine translation with recurrent neural network for software localization. ESE 24, 6 (2019), 3514–3545.

SP04: Masanari Kondo, Cor-Paul Bezemer, Yasutaka Kamei, Ahmed E Hassan, and Osamu Mizuno. 2019. The impact of feature reduction techniques on defect prediction models. Empirical Software Engineering 24, 4 (2019), 1925–1963.

SP05: Di Wu, Xiao-Yuan Jing, Hongyu Zhang, Bing Li, Yu Xie, and Baowen Xu. 2021. Generating API tags for tutorial fragments from Stack Overflow. Empirical Software Engineering 26, 4 (2021), 1–37.

SP06: Zhenyu Zhang, Hailong Sun, and Hongyu Zhang. 2020. Developer recommendation for Topcoder through a meta-learning based policy model. Empirical Software Engineering 25, 1 (2020), 859–889.

SP07: Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2020. Deep code comment generation with hybrid lexical and syntactical information. ESE 25, 3 (2020), 2179–2217.

SP08: Mohammad Alahmadi, Abdulkarim Khormi, Biswas Parajuli, Jonathan Hassel, Sonia Haiduc, and Piyush Kumar. 2020. Code Localization in Programming Screencasts. ESE 25, 2 (2020), 1536–1572.

SP09: Chen Lyu, Ruyun Wang, Hongyu Zhang, Hanwen Zhang, and Songlin Hu. 2021. Embedding API dependency graph for neural code generation. Empirical Software Engineering 26, 4 (2021), 1–51.

SP10: Hasan Ferit Eni¸ser and Alper Sen. 2020. Virtualization of stateful services via machine learning. SQJ 28, 1 (2020), 283–306.

W01: Pooja Rani and GS Mahapatra. 2018. Neural network for software reliability analysis of dynamically weighted NHPP growth models with imperfect debugging. STVR 28, 5 (2018), e1663.