Contents

Acknowledgments iv

Contents vi

List of Figures xi

List of Tables xiii

Abbreviations xv

Symbols xvi

PREFACE xix

1 Introduction 1

1.1 Software Defect Prediction 2
1.1.1 Cross Version Defect Number Prediction 4
1.1.2 Cross Project Defect Prediction 5
1.1.3 Hybrid Regression Analysis 6
1.2 Limitations of Existing Software Defect Prediction Approaches 6
1.2.1 Limitation on Cross Version Defect Number Prediction 7
1.2.2 Cross Project Defect Prediction 7
1.2.3 Limitations on Hybrid Regression Analysis 8
1.3 Motivation 9
1.4 Research Goal 12
1.5 Contributions 13
1.5.1 BPDET: Classification Based Software Defect Prediction Method 14
1.5.2 BCV-Predictor: Cross-Version Defect Count Vector Predictor 15
1.5.3 DNNAttention: Cross-Project Defect Number Prediction Approach 16
1.5.4 Hybrid Regression Analysis: Entire Software Prediction 18

1.6 Literature Review of Various SDP Methods . . . 18
1.6.1 Literature Review of Classification based Model . . . 19
1.6.2 Literature Review over Cross-Version Defect Prediction . . . 20
1.6.3 Literature Review over Cross-Project Fault Prediction and Attention layer . . . 22
1.6.4 Literature Review Over Hybrid Regression and Deep Learning 23
1.7 Structure of the Thesis . . . 23

2 Preliminary **25**
2.1 Software Metrics . . . 25
2.2 Dataset Description . . . 26
2.2.1 Challenges Over Datasets . . . 29
2.3 Methodologies . . . 30
2.3.1 Techniques Involve in Preprocessing . . . 30
2.3.2 Learning Techniques . . . 31
2.4 Performance metrics . . . 36
2.5 Baseline Methods . . . 38

3 BPDET: Classification Based Software Defect Prediction Model **40**
3.1 Introduction . . . 40
3.2 Architectural design . . . 41
3.2.1 Proposed model . . . 42
3.2.1.1 Implication of combining SDA and EL techniques . . 43
3.2.1.2 Data preprocessing . . . 44
3.2.1.3 Deep learning phase . . . 44
3.2.1.4 Ensemble learning phase . . . 46
3.2.2 Experiment setup . . . 53
3.2.2.1 Datasets . . . 54
3.2.2.2 Tools and its parameters . . . 54
3.2.2.3 Software metrics . . . 54
3.2.2.4 Performance evaluation parameters . . . 54
3.3 Result and discussion . . . 55
3.3.1 Effect of corruption rate over performance of BPDET . . . 63
3.3.2 Research queries discussion . . . 65
3.3.2.1 Effectiveness of BPDET in terms of performance metrics compared with other fault prediction model . . . 65
3.3.2.2 How useful BPDET model compared with traditional methods regarding the class imbalance and over-fitting problem . . . 66
3.3.2.3 How much training time taken by the BPDET model 68
3.3.3 Significant analysis . . . 69
3.3.4 Insightful discussion . . . 71
3.4 Threats to validity . . . 72

3.5 Summary 72

4 BCV-Predictor: Cross-Version Defect Count Vector Predictor 74

4.1 Introduction 74
4.2 Problem statement 76
4.2.1 Dataset description 77
4.2.2 Software metrics 77
4.2.3 Meta-dataset collection process 77
4.3 Proposed approach 78
4.3.1 Preprocessing 80
4.3.2 Deep learning architecture 81
4.3.3 Proposed model 81
4.3.4 Experimental arrangements 85
4.3.5 Performance metrics 85
4.3.6 Baseline methods 85
4.3.7 Results and explanations 87
4.3.7.1 Justification of RQ-1 87
4.3.7.2 Justification of RQ-2 92
4.3.8 Insightful discussion 92
4.3.8.1 Justification of RQ-3 96
4.3.9 Non-Parametric test 96
4.4 Summary 98

5 DNNAttention: Cross-Project Defect Number Prediction Approach 100

5.1 Introduction 100
5.1.1 Problem statement 102
5.2 Premises 103
5.2.1 Cross project defect number prediction 103
5.2.2 Software projects 104
5.2.3 Evaluation metrics 104
5.3 Proposed work 104
5.3.1 Data synthesis 105
5.3.2 Deep neural network 107
5.3.2.1 Long short term memory 107
5.3.2.2 Attention layer 108
5.3.3 Preprocessing 108
5.3.4 Proposed algorithm 109
5.3.5 Transfer learning phase 111
5.3.6 Experimental setup 112
5.4 Result and discussion 113
5.4.1 What is the accuracy that DNNAttention reached while predicting defect numbers over the target projects? 114

5.4.2 What losses the DNNAttension sufferers while predicting defect numbers over a new project? . . . 116
5.4.3 Compare the performance of the proposed model over exiting baseline methods? . . . 117
5.4.4 How much the DNNAtention subjugate the CI and overfitting problem? How much is the proposed model stable? . . . 119
5.4.5 Some insightful discussion . . . 119
5.5 Threats to validity . . . 120
5.6 Summary . . . 122

6 Predicting the next version of the Software System: A Hybrid Regression Analysis 129
6.1 Introduction . . . 129
6.2 Problem Identification . . . 131
6.2.1 Problem definition . . . 131
6.3 Premises . . . 133
6.3.1 Deep Learning Architecture . . . 133
6.3.2 Software systems . . . 133
6.3.3 Performance Measures . . . 133
6.4 Proposed Approach . . . 135
6.4.1 Proposed algorithm . . . 136
6.4.2 Experimental setup . . . 136
6.4.3 Optimization and Hyperparameters Tunning . . . 137
6.5 Results and Discussion . . . 138
6.5.1 How is the proposed approach effective in predicting the bug count of every module in the next version of a software system? 139
6.5.2 How is the proposed approach effective in predicting the each software metric values of every module in the next version of a software system? . . . 140
6.5.3 How much the proposed model is significantly effective over existing learning methods? . . . 140
6.5.4 Insightful discussion . . . 141
6.6 Threats to validity . . . 146
6.7 Summary . . . 146

7 Summary 150

8 Conclusion and Future Direction 153
8.1 Conclusion . . . 153
8.2 Future Direction . . . 156

A Publications 158
A.1 Journal Papers . . . 158

A.2 Conference Papers 159
A.3 Communicated Papers 159

Bibliography **160**

List of Figures

1.1 Basic architecture of software defect prediction technique. 3
1.2 Thesis Structure. 24

2.1 Percentage of studies for different dataset that are used in SFP. . . . 28
2.2 Underlying architecture of LSTM. 32
2.3 Attention layer architecture. 32
2.4 The architecture of Sequence to Sequence model using LSTM cell. . . 33

3.1 Working architecture of BPDET. 41
3.2 Design of Denoising auto encoder, the initial input is a and it is corrupted to à, à is mapped with b, then finally c tries to regenerate to a. The regeneration error is represented by M_H(a,c). Taken from [254]. 42
3.3 Staked denoising auto-encoder (SDA), the input layer nodes represented by the solid circle(green) at the bottom, the middle circles are hidden layer nodes. The class label c_i of deep representation is the same as the input. 43
3.4 Comparison of MCC, ROC, PRC, F-measure between best fault prediction techniques over all dataset. 58
3.5 Comparison of MCC, ROC, PRC, F-measure of BPDET over all 12 dataset. 64
3.6 Boxplot of positive parameters λ and δ for corruption rate(σ_d). 65
3.7 Variation of MCC, ROC, PRC and F-measure of BPDET with respect to corruption rate for every dataset. 66
3.8 Boxplot representation of MCC. ROC, PRC and F-measure for every dataset when the corruption rate(σ_d) varies. 67

4.1 Basic architecture of software bug count prediction model. 75
4.2 Meta-data collection process. 78
4.3 Graphical abstract of Proposed work. 79
4.4 Basic architecture of LSTM network, and LSTM cell. 80
4.5 Accuracy of all seven metadata. 90
4.6 Loss of all seven metadata. 93
4.7 Bar-graph of MAE, MSE, & accuracy over all meta-datasets. 94

4.8 Boxplot of MAE, MSE, & accuracy over all meta-datasets after 300 epoch. 95

5.1 Underlying framework of cross project defect number prediction. . . . 103
5.2 Graphical abstract. 106
5.3 Underlying framework of Data Synthesis. 106
5.4 Bar-graph of MSE, MAE, & Accuracy over every project. 114
5.5 Accuracy of datasets. 123
5.6 Accuracy of datasets. 124
5.7 Accuracy of datasets. 125
5.8 Loss of datasets. 126
5.9 Loss of datasets. 127
5.10 Loss of datasets. 128

6.1 Structure of next version prediction of a software system. 129
6.2 The graphical abstract of proposed approach. 135
6.3 Comparison of average MSE, MAE, and Accuracy value(%) of all learning model at different iteration. 142
6.4 Cost effectiveness of Proposed model compared with regular LSTM in terms of MSE, MAE, and Accuracy. 144
6.5 Accuracy and loss versus epoch of all eight metadataset in data augmentation phase. 147
6.6 Accuracy and loss versus epoch of all eight meta dataset in next version prediction phase. 148

List of Tables

1.1 Method level metrics description. 2

2.1 Method level metrics description. 26
2.2 Object oriented level metrics description. 27
2.3 Dataset description before and after preprocessing. 28
2.4 PROMISE Dataset description. 29

3.1 Comparison of MCC over baseline and EL methods with BPDET. Note** "-" symbol implies model cannot give output. 49
3.2 Comparison of MCC with various classification methods with BPDET. Note** "-" symbol implies model cannot give output. 50
3.3 Comparison of ROC over baseline and EL methods with BPDET. . . 51
3.4 Comparison of ROC with various classification methods against BPDET. 52
3.5 Comparison of PRC with various EL method. 53
3.6 Comparison of PRC with various classification method. 57
3.7 Comparison of F-measure over baseline and EL methods with BPDET. Note** "-" symbol implies model cannot produce output. 59
3.8 Comparison of F-measure with various classification method. Note** "-" symbol implies model cannot give output. 60
3.9 Comparison of total training time(seconds) taken by the various ensemble learning methods and some base classifier with respect to BPDET. 61
3.10 Comparison of MCC, ROC, PRC, f-score on k-fold cross validation over BPDET 62
3.11 Adjustment of parameters(λ, δ) with defective rate to achieve threshold T(σ_d). 69
3.12 Two different sample of BPDET and other best models. The full description of the table in the section 3.3.3. 69
3.13 Ranking of values from samples. The full description of the table in the section 3.3.3. 70

4.1 Meta-datasets description. 77
4.2 Performance of BCV-Predictor. 87
4.3 MAE comparison of BCV-Predictor with other techniques. 91
4.4 MSE comparison of BCV-Predictor with other techniques. 91

4.5 Accuracy comparison of BCV-Predictor with other techniques. 91
4.6 Training time (second) of various models compared with BCV-Predictor. 97
4.7 Actual and predicted values of bugs in different samples. 98

5.1 Software projects description. 105
5.2 Performance of DNNAttention. 113
5.3 MSE comparison of DNNAttention over baseline methods. 115
5.4 MAE comparison of DNNAttention over baseline methods. 116
5.5 Accuracy comparison of DNNAttention over baseline methods. . . . 118
5.6 Training time (sec) comparison over cross-heap data of DNNAttention over baseline methods. 118
5.7 Comparison of accuracy produced by the DNNAttention over most optimal baseline model. 121

6.1 Dataset description. 132
6.2 Performance at data augmentation phase. 139
6.3 Performance at next version prediction phase. 142
6.4 Comparison of MSE (%) of next version prediction phase with baseline learning methods. The results are in the form of mean ± standard deviation. 143
6.5 Comparison of MAE (%) of next version prediction phase with baseline learning methods. 143
6.6 Comparison of accuracy (%) of next version prediction phase with baseline learning methods. 143
6.7 P-Value and Cliff's Delta (δ) of proposed model (PM) compared with the existing approaches in terms of MSE. 145
6.8 P-Value and Cliff's Delta (δ) of proposed model (PM) compared with the existing approaches in terms of MAE. 145
6.9 P-Value and Cliff's Delta (δ) of proposed model (PM) compared with the existing approaches in terms of Accuracy. 145
6.10 Effectiveness level [43] of cliff's delta(δ). 146
6.11 Number of projects in which PM is statistically significantly improves over existing approaches (+); performs more or less or approximately equally well (=); and performs loses (–) compared to the baseline techniques in form of MSE/MAE/Accuracy. 146