PREDIKSI CACAT PERANGKAT LUNAK MENGGUNAKAN *RECURRENT NEURAL NETWORK* BERBASIS PCA

Eka Alifia Kusnanti[1)], Lauretha Devi Fajar Vantie[2)], dan Umi Laili Yuhana[3)]
[1, 2, 3)] Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember Surabaya
Jl. ITS Raya, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60111
e-mail: 6025231021@student.its.ac.id[1)], 6025231062@student.its.ac.id[2)], yuhana@if.its.ac.id[3)]

ABSTRAK

Software Quality adalah salah satu fase penting dalam pengembangan perangkat lunak. Software quality menilai kegunaan dan kualitas perangkat lunak yang dikembangkan. Dengan prediksi defect di awal pengembangan perangkat lunak, akan membantu dalam penjaminan kualitas perangkat lunak dengan mengurangi cacat perangkat lunak yang mungkin akan terjadi. Dengan prediksi yang baik akan memberikan keuntungan tambahan pada sumber daya dan efisiensi biaya. Dalam penelitian ini diusulkan metode prediksi cacat perangkat lunak menggunakan metode Recurent Neural Network (RNN) berbasis Principal Component Analysis (PCA). Dataset yang digunakan adalah dataset PROMISE yaitu JM1, CM1, PC1, KC1, dan KC2. Dari hasil uji coba menunjukkan metode PCA-RNN berhasil dit-erapkan dengan baik. Untuk akurasi tertinggi pada dataset PC1 dengan akurasi 93.99% dengan pembagian data training:data testing (70:30).

***Kata Kunci**: Cacat Perangkat Lunak, Recurrent neural network, Principal Component Analysis, Software Quality.*

SOFTWARE DEFECT PREDICTION USING PCA BASED RECURRENT NEURAL NETWORK

Eka Alifia Kusnanti[1)], Lauretha Devi Fajar Vantie[2)], and Umi Laili Yuhana[3)]
[1, 2,3]Informatics Engineering Department, Faculty of Intelligent Electrical and Informatics Technology
Jl. ITS Raya, Keputih, Sukolilo District, Surabaya, East Java 60111
e-mail: 6025231021@student.its.ac.id[1)], 6025231062@student.its.ac.id[2)], yuhana@if.its.ac.id[3)]

ABSTRACT

Software quality is one of the important phases in software development. Software quality assesses the usability and quality of the software developed. Defect prediction early in software development helps in software quality assurance by reducing software defects that may occur. With good predictions, it will provide additional benefits in terms of resource and cost efficiency. The researchers in this study have proposed a software defect prediction method that utilizes a Recurrent Neural Network (RNN) based on Principal Component Analysis (PCA). The dataset used is the PROMISE dataset, namely JM1, CM1, PC1, KC1, and KC2. The test results showed that the PCA-RNN method was successfully applied. For the highest accuracy on the PC1 dataset, with an accuracy of 93.99% with the division of training data by testing data (70:30).

***Keywords**: Software defects, Recurrent neural network, Principal Component Analysis, Software Quality.*

I. PENDAHULUAN

Perangkat lunak menjadi bagian penting dalam kehidupan manusia modern saat ini dan digunakan dalam banyak bidang seperti pendidikan, bisnis dan hiburan. Perangkat lunak ini dirancang untuk membantu melakukan proses kinerja dengan cepat, tepat, efektif, dan efisien [1]. Perangkat lunak yang berkualitas akan dapat memberikan manfaat yang maksimal bagi penggunanya. Oleh karena itu, pengujian kualitas software merupakan bagian penting dalam proses pengembangan perangkat lunak karena membantu menentukan kemampuan dan kualitas software. Hasil evaluasi dari *software quality* membantu mengidentifikasi masalah dan *software defect* yang mungkin terjadi.

Prediksi cacat perangkat lunak atau sering disebut *software defect prediction* (SDP) adalah metode atau teknik pengukuran kualitas perangkat lunak yang digunakan untuk memprediksi kesalahan atau *defect* pada perangkat lunak yang akan dibuat [2]. Teknik *software defect prediction* adalah teknik yang digunakan untuk membuat perangkat lunak yang sangat baik dengan meminimalkan kemungkinan cacat terjadi ketika perangkat lunak dijalankan.[3]. Teknik ini membantu pengembang perangkat lunak mengalokasikan sumber daya lebih efisien dan menghemat biaya dengan membuat produk perangkat lunak yang memiliki kualitas tinggi. *Software defect prediction* memfokuskan upaya pengujian pada modul yang memiliki risiko tinggi dan dapat menghemat dari biaya inspects atau pengujian kode [4].

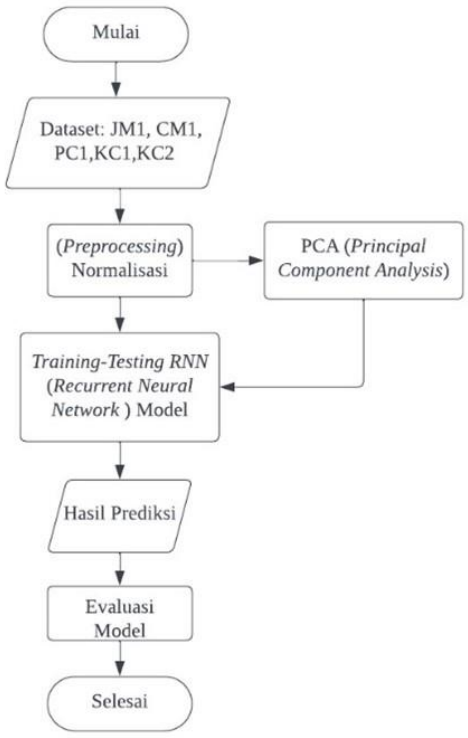
http://dx.doi.org/10.12962/j24068535.v22i1.a1199

Untuk meningkatkan kinerja prediksi *software defect*, para peneliti telah menggunakan berbagai jenis algoritma pembelajaran mesin dan teknik statistik. [5]. Penelitian terdahulu telah menggunakan beberapa algoritma machine learning diantaranya yaitu *Decision Tree* [6], *Naive Bayes Gain Ratio*[3],*Bagging-Naive Bayes* [5], *Bayesan Network, PSO-KNN* [2], *RNN* [7] dan penelitian lainnya sesuai pada Tabel 3. Pada penelitian algoritma RNN digunakan dataset SFP XP-TDD yang dibuat menggunakan tiga project perangkat lunak. Untuk dataset NADA MDP dan Promise digunakan dalam penelitian-penelitian sebelumnya

Penelitian ini mengusulkan penggunaan teknik *Recurrent Neural Network* (RNN) yang didasarkan pada *Principal Component Analysis* (PCA) untuk memprediksi *software defect* dengan menggunakan PROMISE dataset (JM1, CM1, PC1, KC1, dan KC2). RNN merupakan salah satu metode *machine learning*, yang dikembangkan dari Jaringan Syaraf Tiruan (JST), dan berfokus pada ekstraksi data dengan tingkat detail yang lebih rinci. Untuk mengurangi dimensi data, *Principal Component Analysis* (PCA) merupakan salah satu metode yang umumnya digunakan dalam tahap *preprocessing*. PCA mengekstraksi atribut data sehingga hanya atribut dengan bertujuan untuk mencapai hasil yang optimal yang dipertahankan. Dengan penggunaan PCA diharapkan akan meningkatkan nilai akurasi prediksi.

II. METODE PENELITIAN

Penelitian prediksi *software defect* ini tergolong ke dalam domain penelitian berorientasi kuantitatif karena penggunaan data dalam penelitian ini memiliki bentuk numerik. Langkah-langkah dalam penelitian ditunjukkan pada Gambar 1 yang terdiri dari pengumpulan dataset, *preprocessing* data, reduksi data menggunakan *Principal Component Analysis* (PCA), *training* model menggunakan RNN, *testing* model menggunakan RNN, dan evaluasi model hasil klasifikasi dengan mengukur akurasi.



Gambar 1. Flowchart Metode Penelitian

Langkah-langkah implementasi PCA-RNN untuk prediksi *software defect*:

1. Melakukan pengecekan data, apakah ada yang bernilai null jika ada maka data akan di hapus
2. Melalukan normalisasi pada dataset dengan metode Min-Max Scaling
3. Selanjutnya data diujicoba dengan 2 metode yang pertama langsung menggunakan model RNN, yang kedua dataset diproses terlebih dahulu menggunakan PCA untuk mereduksi dimensi data. Pada proses PCA data dimensi data direduksi dari 22 atribut menjadi 15 atribut.
4. Dataset diujicoba menggunakan RNN dengan pembagian data training dan testing sebagai berikut 70:30, 80:20, 90:10. Layer yang digunakan pada RNN adalah Simple RNN dengan 50 neuron dan *relu activation*. Uji coba menggunakan 100 *Epoch* dan adam *Optimizer*.

5. Setelah di dapatkan hasil dilakukan evaluasi model yaitu model Normalisasi-RNN dan Normalisasi-PCA-RNN

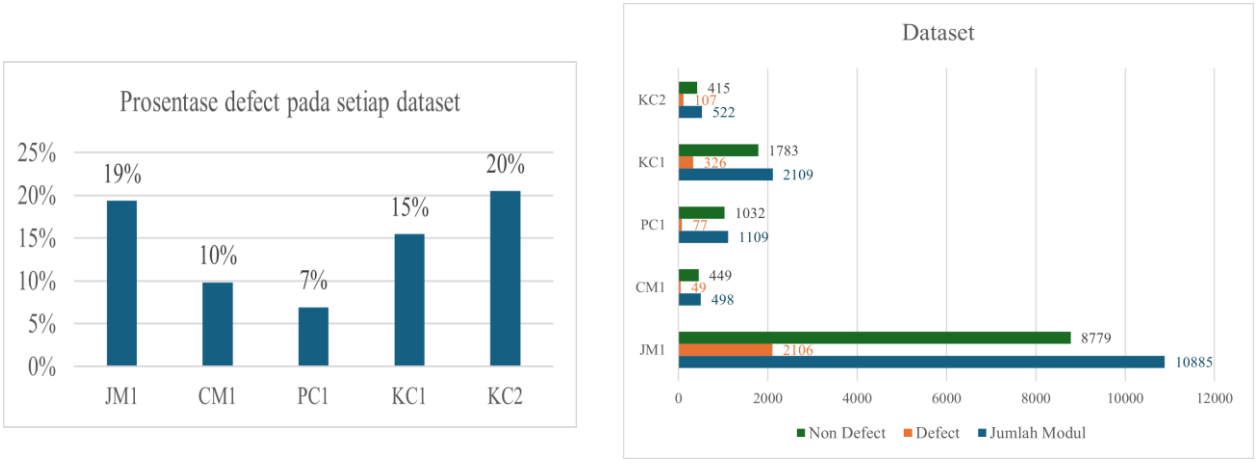
A. Dataset

Dataset PROMISE adalah sumber data yang digunakan dalam penelitian ini, yang mencakup lima subdataset, yaitu PC1, JM1, CM1, KC1, dan KC2. Setiap data set memiliki jumlah data yang berbeda baik itu yang defect maupun non defect. Keseluruhan dataset terdiri dari 22 atribut yang memiliki peran penting dalam analisis. Dataset akan dibagi menjadi dua bagian, yaitu data pelatihan dan pengujian dengan tiga skema, masing-masing 70% pelatihan dan 30% pengujian, 80% pelatihan dan 20% pengujian, serta 90% pelatihan dan 10% pengujian. Di bawah ini detail atribut yang terdapat dalam dataset pada Tabel 1.

Untuk setiap dataset memiliki prosentase sebaran data *defect* dan *non defect* yang berbeda-beda terlihat pada grafik Gambar 2 Pada dataset JM1 dan KC2 memiliki sebaran data *defect* yang hampir sama yaitu 19% dan 20%.

TABEL I
ATRIBUT DATASET

No.	Simbol	Keterangan
1	loc	McCabe's line count of code
2	v(g)	McCabe "cyclomatic complexity"
3	ev(g)	McCabe "essential complexity"
4	iv(g)	McCabe "design complexity"
5	n	Halstead total operators + operands
6	v	Halstead "volume"
7	l	Halstead "program length"
8	d	Halstead "difficulty"
9	i	Halstead "intelligence"
10	e	Halstead "effort"
11	b	Halstead "effort"
12	t	Halstead's time estimator
13	lOCode	Halstead's line count
14	lOComment	Halstead's count of lines of
15	lOBlank	Halstead's count of blank lines
16	lOCodeAnd	Line of comment and code Comment
17	uniq_Op	Halstead Unique operators
18	uniq_Opnd	Halstead unique operands
19	total_Op	Halstead Total operators
20	total_Opnd	Halstead Total operands
21	branchCount	Branch count of the flowgraph
22	defects Class	{false,true}



Gambar 2. Sebaran Dataset

B. *Preprocessing*

Pada tahap awal *preprocessing* data dilakukan untuk mengubah data masukan menjadi bentuk yang kompatibel. Di sini, nilai-nilai kategoris yang ada dalam kumpulan data diubah menjadi nilai-nilai numerik. Selanjutnya, dilakukan penskalaan data dengan menggunakan metode normalisasi, yang dijelaskan di bawah ini.

Normalisasi

Normalisasi adalah sebuah metode yang digunakan untuk mengubah data ke berbagai rentang nilai yang seimbang [8]. Salah satunya yaitu, Normalisasi *min-max* akan mengubah rentang nilai data ke dalam skala yang telah ditentukan, tetapi tidak mengubah karakteristik distribusi nilai yang ada [9].

$$Y_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) merupakan teknik yang biasa digunakan dalam tahap *preprocessing* data untuk mengurangi dimensi data [10]. Prinsip dasar metode ini mengekstraksi sehingga hanya atribut yang bertujuan untuk mencapai hasil optimal yang tetap dipertahankan. PCA juga merupakan salah satu metode dalam analisis faktor yang umumnya digunakan ketika melakukan pendekatan *multivariat* pada variabel yang berkorelasi, kemudian mentransformasikannya menjadi variabel tak berkorelasi yang disebut sebagai faktor [11].

Keunggulan metode PCA adalah kemampuannya untuk mengatasi analisis data dalam jumlah besar dan mengurangi dimensi data menjadi komponen yang lebih kecil. Metode ini juga dapat diterapkan langsung pada normalisasi data multivariat, kompresi data, analisis grafik, dan analisis statistik. Selain itu, penggunaan PCA dalam *preprocessing* jaringan saraf telah banyak diadopsi di beberapa bidang seperti klasifikasi dan analisis gambar [12].

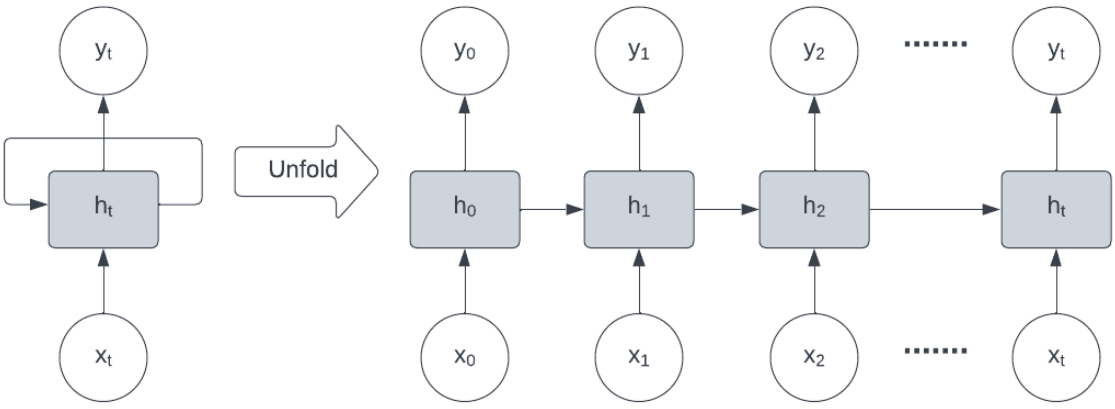
Berikut Algoritma PCA secara umum:

- Standarisasi data
- Menghitung matriks kovarians
- Menghitung nilai eigen dan vektor eigen
- Mengurutkan nilai eigen dan vektor eigen
- Memilih komponen utama yang menjelaskan sejumlah variasai tertentu
- Transformasikan data keruang komponen utama

C. *Recurrent Neural Network* (RNN)

Recurrent Neural Network (RNN) berasal dari jaringan Hopfield yang diusulkan pada tahun 1982 dan telah membawa terobosan penting dalam berbagai bidang seperti pengenalan suara, terjemahan mesin, dan analisis waktu. RNN, dengan strukturnya yang lebih efektif, memungkinkan penggunaan informasi sekuensial dan semantik dalam penambangan data.

RNN merupakan jenis jaringan saraf tiruan yang memungkinkan penggunaan informasi dari pengetahuan masa lalu menggunakan arsitektur berulang yang disebut jaringan rekuren [13]. Dalam analisis deret waktu, RNN menjadi salah satu jaringan saraf terpenting karena kemampuannya untuk memproses data terkait dengan waktu. Dibandingkan dengan jaringan saraf biasa, RNN memiliki keterkaitan antar lapisan yang memungkinkan penggunaan hasil perhitungan lapisan sebelumnya dan pengingatan informasi sebelumnya [14].



Gambar 3. Arsitektur RNN

Simple RNN adalah jenis jaringan saraf yang terdiri dari beberapa lapisan yang bekerja bersama-sama, dan tugas utamanya adalah mengirimkan informasi dari satu jaringan ke jaringan lainnya. Dalam *Artificial Neural Networks*

RNN, beberapa *Artificial Neural Networks* (ANN) digunakan untuk memecahkan masalah urutan atau sekuen data. Gambar 3 menunjukkan struktur dasar dari *Artificial Neural Networks* RNN [15].

Untuk membuat keputusan, RNN mempertimbangkan masukan saat ini dan menghasilkan keluaran berdasarkan masukan sebelumnya yang telah dipelajari. Setiap modul berulang dari RNN standar memiliki satu lapisan aktivasi tunggal yang mengandung fungsi tangen hiperbolik. RNN digunakan untuk aplikasi yang memiliki data dalam format sekuensial seperti pengenalan suara dan prediksi deret waktu. RNN memiliki kemampuan untuk memperhitungkan aspek urutan data masukan dan memori yang menyimpan informasi dari masa lalu untuk menghasilkan keluaran saat ini. Perhitungan RNN mencakup masukan, bobot, bias, dan fungsi aktivasi [16]. Sebuah RNN sederhana dapat dimodelkan menggunakan persamaan 2.

$$h^{(t)} = \sigma\left(W_{hx}x + W_{hh}h^{(t-1)} + b_h \right) \quad (2)$$

Di mana,

- Whx adalah matriks bobot antara lapisan masukan dan lapisan tersembunyi,
- Whh adalah matriks bobot rekuren antara lapisan tersembunyi pada langkah waktu yang berdekatan.
- bh dan by adalah vektor bias yang memungkinkan setiap simpul untuk belajar dan mengimbangi[14].

D. Evaluasi Hasil

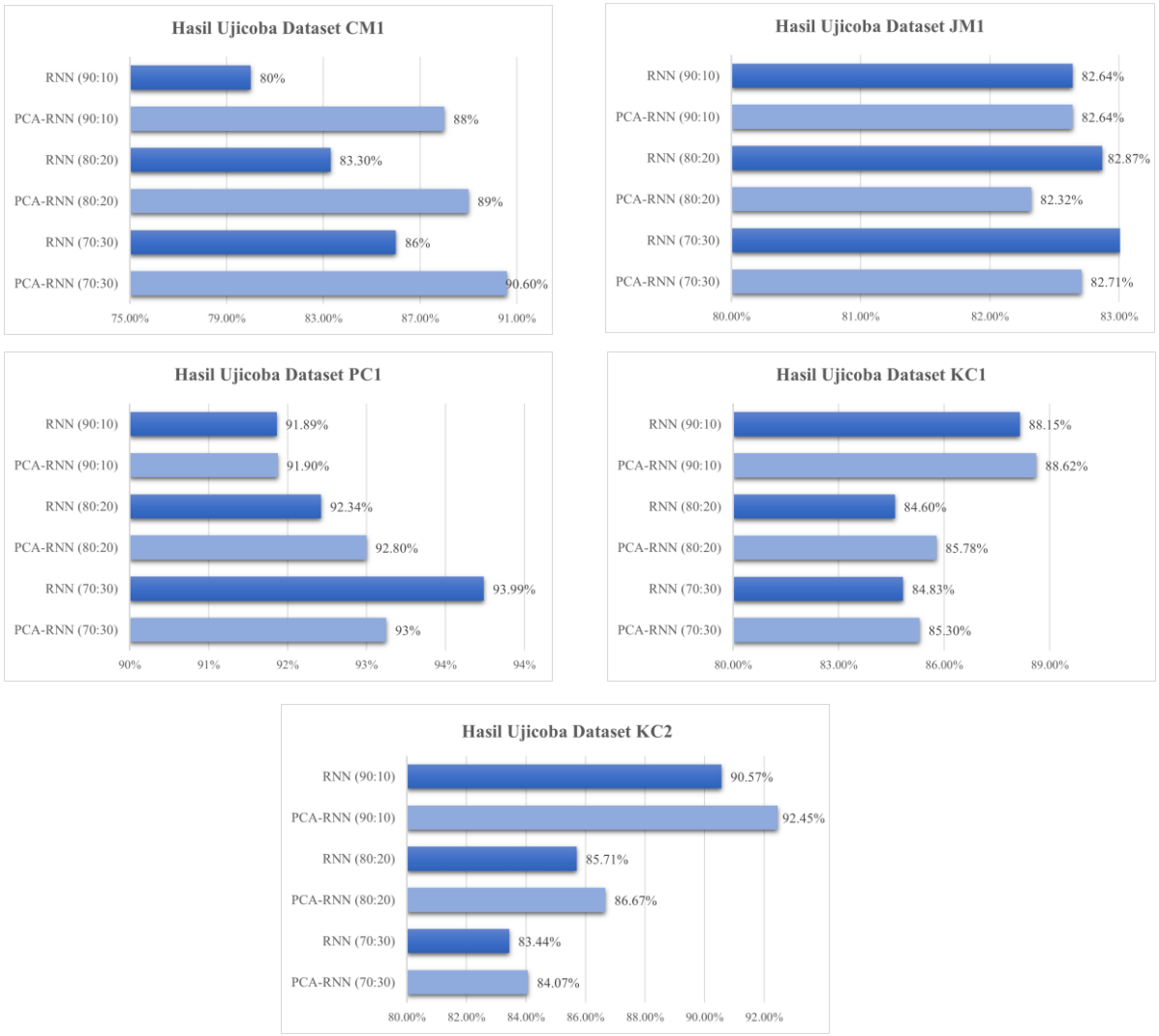
Evaluasi hasil akan digunakan untuk menilai sejauh mana model yang telah dibuat efektif dan efisien. Akurasi adalah ukuran ketepatan model prediksi [17]. Ini didefinisikan sebagai rasio dari data yang diklasifikasikan dengan benar terhadap [18].

III. HASIL DAN PEMBAHASAN

Pengujian akan dilakukan menggunakan lima dataset PROMISE yang meliputi JM1, CM1, PC1, KC1, dan KC2 dengan jumlah yang berbeda-beda tiap dataset. Dari kelima data tersebut JM1 memiliki jumlah dataset paling banyak (10885) sedangkan CM 1 memiliki jumlah dataset paling sedikit (498). Terdapat dua skema uji coba serta tiga model pembagian data training dan data testing. Dua skema dilakukan untuk mengetahui pengaruh hasil PCA pada hasil prediksi. Pada skema pertama dataset hanya dilakukan normalisasi, sedangkan pada skema kedua dataset akan dilakukan normalisasi dan PCA. Semua dataset akan dikelompokkan menjadi data pelatihan dan pengujian dengan tiga skema, masing-masing 70% pelatihan dan 30% pengujian, 80% pelatihan dan 20% pengujian, serta 90% pelatihan dan 10% pengujian. Model yang dihasilkan akan dievaluasi berdasarkan tingkat akurasi. Semakin besar nilai akurasi maka model yang dihasilkan semakin baik sehingga bisa mendapatkan output yang akurat. Hasil akurasi ditampilkan dalam Tabel II. Pada Gambar 4 menunjukkan visualisasi grafik hasil akurasi masing-masing uji coba dengan metode RNN dan PCA-RNN yang menggunakan 3 model pembagian data *training* dan data *testing*.

TABEL II
HASIL AKURASI PREDIKSI SOFTWARE DEFECT MENGGUNAKAN RNN

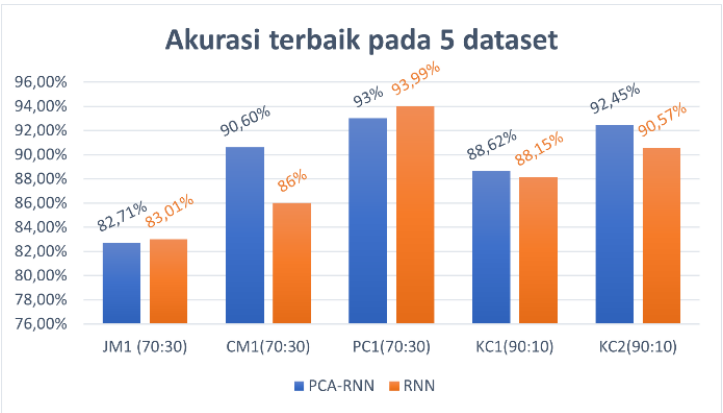
DataSet	Jumlah Modul	Defect	Non Defect	Pembagian Data	RNN	PCA-RNN
JM1	10885	2106	8779	70 : 30	83.01%	82.71%
				80 : 20	82.87%	82.32%
				90 : 10	82.64%	82.64%
CM1	498	49	449	70 : 30	86.00%	90.60%
				80 : 20	83.30%	89.00%
				90 : 10	80.00%	88.00%
PC1	1109	77	1032	70 : 30	93.99%	93.00%
				80 : 20	92.34%	92.80%
				90 : 10	91.89%	91.90%
KC1	2109	326	1783	70 : 30	84.83%	85.30 %
				80 : 20	84.60%	85.78%
				90 : 10	88.15%	88.62%
KC2	522	107	415	70 : 30	83.44%	84.07%
				80 : 20	85.71%	86.67%
				90 : 10	90.57%	92.45%



Gambar 4. Grafik akurasi semua model

Pada Gambar 4 menunjukkan Dataset JM1, CM1, dan PC1, model terbaik terdapat pada pembagian data 70%:30%. Hasil akurasi terbaik tiap dataset yaitu: JM1 83.01%, CM1 90.60%, dan PC1 93.99%. Sedangkan pada Dataset KC1 dan KC2, model terbaik terdapat pada pembagian 90%:10%. Hasil akurasi terbaik tiap dataset yaitu: KC1 88.62% dan KC2 92.45%. Hasil ini menunjukkan bahwa setiap dataset memiliki karakteristik yang berbeda sehingga penting untuk melakukan uji coba dengan berbagai model pembagian data training dan testing untuk menemukan model terbaik yang sesuai dengan karakteristik setiap dataset.

Selanjutnya, hasil akurasi pada model terbaik akan dibandingkan untuk mengevaluasi kinerja PCA-RNN dalam memprediksi software defect, seperti yang ditunjukkan dalam grafik pada Gambar 5.



Gambar 5. Grafik akurasi model terbaik

Pada Gambar 5 menunjukkan hasil akurasi dari model terbaik pada setiap dataset, 4 diantaranya menunjukkan hasil penggunaan PCA dapat meningkatan akurasi yaitu pada dataset PC1, CM1, KC1, dan KC2. Pada dataset PC1, PCA dapat meningkatkan akurasi tetapi tidak pada model terbaik, melainkan pada model pembagian data 80%:20% dan 90%:10%. Sedangkan pada dataset JM1, hasilnya tidak jauh beda baik menggunakan PCA atau tidak, bahkan terdapat hasil akurasi yang sama. Hal tersebut dapat terjadi karena dipengaruhi banyak hal, salah satunya yaitu karena ketidakseimbangan atau hal lainnya terkait dataset. Secara keseluruhan hasil terbaik pada uji coba terdapat pada dataset PC1 dengan pembagian data 70%:30% dengan hasil akurasi sebesar 93.99%. Pada dataset PC1, kenaikan akurasi tidak signifikan dalam penggunaan PCA dapat disebabkan oleh ketidakidealannya dalam sebaran dataset antara jumlah *defect* dan *non-defect*. Dimana pada dataset PC1 hanya memiliki data defect sebanyak 7% (presentase terkecil dibandingkan dataset lain) yang ditunjukkan oleh grafik pada Gambar 2. Kemungkinan besar, distribusi yang tidak seimbang ini mempengaruhi kemampuan PCA dalam mengekstraksi fitur-fitur yang signifikan untuk meningkatkan akurasi prediksi *software defect*. Meskipun PCA umumnya efektif dalam mereduksi dimensi dan menangkap variasi data, tingkat ketidakseimbangan yang signifikan dapat menghambat kemampuannya dalam mengenali pola yang berkaitan dengan kejadian *defect.*

Berdasarkan akurasi hasil semua uji coba tersebut menunjukkan bahwa PCA dan RNN bisa bekerja dengan baik untuk prediksi *software defect*. Beberapa dataset sudah bisa menghasilkasn akurasi yang baik hanya dengan normalisasi dan RNN. Namun, terdapat dataset yang akurasinya lebih baik jika menggunakan PCA dan RNN. Selanjutnya, hasil penelitian ini akan dibandingkan dengan penelitian sebelumnya. Perbandingan akurasi metode RNN, PCA RNN, dengan metode lain juga akan ditunjukkan pada grafik pada Gambar 5.

Tabel 3 dan Gambar 5 menunjukkan hasil perbandingan metode RNN, PCA-RNN, dengan beberapa metode pada penelitian sebelumnya seperti Decision Tree, KNN, PCA-SVM, ANN. Hasil akurasi PCA-RNN dan RNN menunjukkan hasil akurasi yang paling unggul dibandingkan dengan metode lain pada semua dataset. Hal ini menunjukkan bahwa metode PCA dan RNN dapat bekerja dengan baik untuk melakukan prediksi doftware defect terutama pada dataset PC1 dengan akurasi tertinggi 93.99% pada model pembagian data 70%:30%.

Dengan menggunakan PCA meningkatkan akurasi hasil pada beberapa dataset. Dalam konteks prediksi *software defect*, keunggulan PCA dalam peningkatan akurasi pada setiap dataset dapat dijelaskan dengan fokus pada kelebihan metode tersebut dalam mengidentifikasi dan mempertahankan komponen-komponen utama yang paling signifikan dalam dataset. Meskipun dataset awalnya memiliki sebaran yang tidak merata, PCA dapat membantu meningkatkan akurasi dengan mempertahankan fitur-fitur yang paling berkontribusi terhadap variasi dalam data. Penelitian selanjutnya bisa diujicobakan ke dataset lain yang memiliki sebaran data ideal.

TABEL III
PERBANDINAGN RNN, PCA-RNN, DAN PENELITIAN SEBELUMNYA.

DataSet	Metode dan Peneliti lain	Akurasi Penelitian Lain	Akurasi RNN	Akurasi PCA-RNN
JM1	[18]Algoritma CART (Decision Tree)	81.70	83.01	82.71
	[3]Naive Bayes Gain Ratio (NB GR)	80.57		
	[19]LSTM	81.76		
CM1	[2]PSO-KNN	88.34	86.00	90.60
	[18]Algoritma CART (Decision Tree)	86.10		
	[3]Naive Bayes Gain Ratio (NB GR)	80.57		
PC1	[2]PSO-KNN	92.54	93.99	93.00
	[20]Naive Bayes	85.00		
	[20]KNN	92.00		
KC1	[21]PCA-SVM	86.60	88.15	88.62
	[22]ANN	83.00		
	[22]Random Forest	85.00		
KC2	[23]SVM	83.85	90.57	92.45
	[23]Decision Tree	88.59		
	[22]Gradient Boosting Classifier	84.00		

Gambar 6. Grafik perbandingan akurasi berbagai metode

IV. KESIMPULAN

Dalam penelitian ini, kami menguji dan membandingkan penggunaan analisis *Principal Component Analysis* (PCA) dan *Recurrent Neural Network* (RNN) pada lima dataset yang berbeda, yaitu JM1, CM1, PC1, KC1, dan KC2. Hasil penelitian menunjukkan bahwa kedua metode ini berhasil diterapkan dengan baik. Hasil terbaik tercapai pada dataset PC1, di mana model yang menggunakan RNN mencapai tingkat akurasi yang sangat mengesankan sebesar 93.99%. Hasil ini menggambarkan potensi besar dari pendekatan RNN dalam memahami dan memprediksi pola dalam data perangkat lunak, khususnya pada dataset PC1. Begitu pun dengan PCA, dari 5 dataset terbukti 4 diantaranya PCA dapat meningkatkan akurasi dengan baik, sedangkan terdapat salah satu dataset yang hasilnya tidak jauh beda dengan atau tidak nya PCA. Hal tersebut bisa dipengaruhi berbagai hal terkait dataset itu sendiri baik karena ketidak seimbangan data atau hal lain. Pada perbandingan dengan metode lain pada penelitian sebelumnya juga terbukti PCA-RNN lebih unggul dan menghasilkan akurasi yang baik. Kesimpulannya, penggunaan RNN dalam kombinasi dengan PCA dapat memberikan hasil yang sangat baik dalam menganalisis dataset perangkat lunak, dengan hasil tertinggi tercapai pada dataset PC1 dengan akurasi mencapai 93.99%.

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih kepada PROMISE Dataset karena telah menyediakan data yang dapat diakses publik, memungkinkan penelitian ini untuk berhasil dan memberikan kontribusi penting dalam domain prediksi *software defect*.

DAFTAR PUSTAKA

[1] A. Bahtiar, Mulyawan, Suryani, and D. Firmansyah, “Prediksi Cacat Sofware Menggunakan Algoritma C4.5 Berbasis Particle Swarm Optimization,” *KOPERTIP: Jurnal Ilmiah Manajemen Informatika dan Komputer* , vol. 3, 2019.

[2] T. Hidayat, A. F. Habibi, and U. L. Yuhana, "Software Defect Prediction Menggunakan Algoritma K-NN Yang Dioptimasi Dengan PSO," *SCAN - Jurnal Teknologi Informasi dan Komunikasi*, vol. 15, no. 1, Feb. 2020, doi: 10.33005/scan.v15i1.1848.
[3] M. Sonhaji Akbar and S. Rochima, "Prediksi Cacat Perangkat Lunak Dengan Optimasi Naive Bayes," *Jurnal Sistem dan Informatika (JSI)*, vol. 11, pp. 147–155, May 2017.
[4] L. Qiao, G. Li, D. Yu, and H. Liu, "Deep Feature Learning to Quantitative Prediction of Software Defects," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, IEEE, Jul. 2021, pp. 1401–1402. doi: 10.1109/COMPSAC51774.2021.00204.
[5] S. A. Putri and R. S. Wahono, "Integrasi SMOTE dan Information Gain pada Naive Bayes untuk Prediksi Cacat Software," *Journal of Software Engineering*, vol. 1, no. 2, 2015, [Online]. Available: http://journal.ilmukomputer.org
[6] E. Borandag, "Software Fault Prediction Using an RNN-Based Deep Learning Approach and Ensemble Machine Learning Techniques," *Applied Sciences*, vol. 13, no. 3, p. 1639, Jan. 2023, doi: 10.3390/app13031639.
[7] S. Sinsomboonthong, "Performance comparison of new adjusted min-max with decimal scaling and statistical column normalization methods for artificial neural network classification," *Int J Math Math Sci*, vol. 2022, 2022.
[8] P. S. S. Gopi and M. Karthikeyan, "Red fox optimization with ensemble recurrent neural network for crop recommendation and yield prediction model," *Multimed Tools Appl*, pp. 1–21, 2023.
[9] D. Sartika and I. Saluza, "Penerapan Metode Principal Component Analysis (PCA) Pada Klasifikasi Status Kredit Nasabah Bank Sumsel Babel Cabang KM 12 Palembang Menggunakan Metode Decision Tree," *Generic*, vol. 14, no. 2, pp. 45–49, 2022.
[10] N. Sunarmi, R. Hasanah, R. Fitriana, and I. N. Hamidah, "Analisis Unsur Cuaca pada Pertanian Bawang Merah Kabupaten Nganjuk Tahun 2019 dengan Principal Component Analysis," in *SENKIM: Seminar Nasional Karya Ilmiah Multidisiplin*, 2022, pp. 40–50.
[11] H. H. Q. Hayqal, O. Soesanto, and Y. Sukmawaty, "K-Means Clustering dan Principal Component Analysis (PCA) Dalam Radial Basis Function Neural Network (RBFNN) Untuk Klasifikasi Data Multivariat," *Journal of Mathematics: Theory and Applications*, pp. 1–7, 2022.
[12] J. Wang, X. Li, J. Li, Q. Sun, and H. Wang, "NGCU: A new RNN model for time-series data prediction," *Big Data Research*, vol. 27, p. 100296, 2022.
[13] A. Ajitha, M. Goel, M. Assudani, S. Radhika, and S. Goel, "Design and development of Residential Sector Load Prediction model during COVID-19 Pandemic using LSTM based RNN," *Electric Power Systems Research*, vol. 212, p. 108635, 2022.
[14] N. M. Shahani, M. Kamran, X. Zheng, and C. Liu, "Predictive modeling of drilling rate index using machine learning approaches: LSTM, simple RNN, and RFA," *Pet Sci Technol*, vol. 40, no. 5, pp. 534–555, 2022.
[15] I. Amalou, N. Mouhni, and A. Abdali, "Multivariate time series prediction by RNN architectures for energy consumption forecasting," *Energy Reports*, vol. 8, pp. 1084–1091, 2022.
[16] A. Wicaksono, "Prediksi dan Deteksi Bug Pada Software Menggunakan Pendekatan Machine Learning: Machine Learning," *JURNAL SIGN IN: Jurnal Ilmiah Sistem Informasi dan Informatika*, vol. 2, no. 2, pp. 14–17, 2023.
[17] S. Goyal, "Effective software defect prediction using support vector machines (SVMs)," *International Journal of System Assurance Engineering and Management*, vol. 13, no. 2, pp. 681–696, 2022.
[18] N. Hidayati, J. Suntoro, and G. G. Setiaji, "Perbandingan Algoritma Klasifikasi untuk Prediksi Cacat Software dengan Pendekatan CRISP-DM," *Jurnal Sains dan Informatika*, vol. 7, no. 2, pp. 117–126, 2021.
[19] S. K. Pemmada, H. S. Behera, J. Nayak, and B. Naik, "Correlation-based modified long short-term memory network approach for software defect prediction," *Evolving Systems*, vol. 13, no. 6, pp. 869–887, 2022.
[20] M. Shafiq, F. H. Alghamedy, N. Jamal, T. Kamal, Y. I. Daradkeh, and M. Shabaz, "Scientific programming using optimized machine learning techniques for software fault prediction to improve software quality," *IET Software*, pp. 1–11, 2023.
[21] M. Mustaqeem and M. Saqib, "Principal component based support vector machine (PC-SVM): a hybrid technique for software defect detection," *Cluster Comput*, vol. 24, no. 3, pp. 2581–2595, 2021.
[22] M. Shah and N. Pujara, "A review on software defects prediction methods," *arXiv preprint arXiv:2011.00998*, 2020.
[23] A. Nawaz, A. U. Rehman, and M. Abbas, "A novel multiple ensemble learning models based on different datasets for software defect prediction," *arXiv preprint arXiv:2008.13114*, 2020.