Building of Informatics, Technology and Science (BITS)
Volume 4, No 2, September 2022 Page: 405−412
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v4i2.1776

Eksperimen Pengujian Optimizer dan Fungsi Aktivasi Pada Code Clone Detection dengan Pemanfaatan Deep Neural Network (DNN)

Errissya Rasywir[1], Yovi Pratama[2,*], Fachruddin[3]

[1,2]Fakultas Ilmu Komputer, Program Studi Teknik Informatika, Universitas Dinamika Bangsa, Jambi, Indonesia
[3] Fakultas Ilmu Komputer, Program Studi Sistem Informasi, Universitas Dinamika Bangsa, Jambi, Indonesia
Email: [1]errissya.rasywir@gmail.com, [2]yovi.pratama@gmail.com, [3]fachruddin.stikom@gmail.com
Email Penulis Korespondensi: yovi.pratama@gmail.com
Submitted: **28/06/2022**; Accepted: **15/07/2022**; Published**: 30/09/2022**

Abstrak−Masalah *similarity* (kemiripan) kode program dapat menjadi salah satu solusi pendekatan deteksi tindakan plagiarisme. Plagiarisme memunculkan suatu bentuk tindakan dan konsekuensi dari plagiarisme itu sendiri apabila *source* yang digunakan bukanlah *open source*. Plagiarisme merupakan tindakan penipuan hasil karya orang lain tanpa sepengetahuan dari penulis aslinya, yang melanggar suatu Hak Cipta dan Hak Moral. Dengan meningkatnya jumlah data dan kompleksitas data, *deep learning* memberikan solusi untuk prediktif analisis, dengan kemampuan *processing* yang meningkat dan pemanfaatan prosesor secara optimal. *Deep learning* memperlihatkan keberhasilan *serta* meningkatkan model klasifikasi pada bidang tersebut. Di sisi lain code *clone detection* dengan volume data masiv, bervariasi dan berkecepatan tinggi sangat memerlukan ektraksi fitur. Dengan potensi yang dimiliki oleh *deep learning* yang bisa mengektraksi fitur lebih baik maka teknik de*ep learning* sesuai untuk code *clone detection*. Untuk itu perlu dikembangkan *code clone detection* yang bisa memproses data dari sebuah bahasa pemrograman dengan memanfaatkan *deep learning*. Berdasarkan hasil eksperimen yang dilakukan pada 100 file data kode program berbahasa PHP, dengan eksperimen beberapa jenis metode fungsi aktivasi dan optimizer. Nilai rata-rata akurasi yang dihasilkan adalah baik. Dengan variasi fungsi aktivasi yang kami gunakan seperti *Relu, Linear, Sigmoid, Softmax, Tanh, Elu, Selu, Softplus, Softsign, hard*, dan *sigmoid*, serta variasi *optimizer* yang digunakan adalah *Adagrad, RMSProp, SGD, Adadelta, Adam, Adamax* dan *Nadam*, pemilihan atribut terbaik adalah pada fungsi *Selu* dan optimizer *RMSProp*. Jumlah *Epoch* yang digunakan adala 1000, jumlah *neuron per layer* adalah 500 dan jumlah *hidden layer* terbaik adalah 10, akurasi rata-rata memeproleh nilai 0.900.

Kata Kunci: DNN; Code; PHP; Eksperimen; Akurasi

Abstract−The problem of similarity (similarity) of program code can be one solution to the plagiarism detection approach. Plagiarism raises a form of action and consequences of plagiarism itself if the source used is not open source. Plagiarism is an act of deception of the work of others without the knowledge of the original author, which violates a Copyright and Moral Rights. With the increasing amount of data and data complexity, deep learning provides solutions for predictive analytics, with increased processing capabilities and optimal processor utilization. Deep learning shows success and improves the classification model in this field. On the other hand, clone detection code with massive, varied and high-speed data volumes requires feature extraction. With the potential of deep learning to extract better features, deep learning techniques are suitable for code clone detection. For this reason, it is necessary to develop a clone detection code that can process data from a programming language by utilizing deep learning. Based on the results of experiments conducted on 100 PHP program code data files, experimented with several types of activation function and optimizer methods. The average value of the resulting accuracy is good. With a variety of activation functions that we use such as Relu, Linear, Sigmoid, Softmax, Tanh, Elu, Selu, Softplus, Softsign, hard, and sigmoid, as well as variations of the optimizer used are Adagrad, RMSProp, SGD, Adadelta, Adam, Adamax and Nadam , the best attribute selection is in the Selu function and the RMSProp optimizer. The number of epochs used is 1000, the number of neurons per layer is 500 and the best number of hidden layers is 10, the average accuracy is 0.900.

Keywords: DNN; Code; PHP; Experiment; Accuracy

1. PENDAHULUAN

Dalam ilmu komputer, kode program adalah kelompok kumpulan pernyataan atau deklarasi bahasa pemrograman pada komputer yang ditulis dan dapat dibaca oleh manusia. Kode program memungkinkan *programmer* untuk berkomunikasi dengan komputer menggunakan beberapa perintah yang telah terdefinisi [1], [2]–[6] . Kode program dapat dibuat dalam satu atau lebih file teks, dan disimpan dalam *database* yang disimpan sebagai metode prosedur dan muncul sebagai potongan kode yang tercetak di artikel, file, maupun buku atau media lainnya. Koleksi file kode program biasanya dapat diatur dalam direktori pohon, dalam hal ini juga dikenal sebagai *source tree* [7], [2], [3], [8]–[10].

Masalah kemiripan pada kode program dapat menjadi solusi pendekatan fungsi dalam tugas deteksi tindakan plagiarisme. Indikasi *similarity* perlu dicermati dengan seksama karena memerlukan penafsiran serta pembahasan khusus agar dapat memberikan keputusan seberapa besar tingkat kemiripan sebuah kode program dengan kode program yang lainnya [11], [2], [8], [12], [13]. Besarnya pengukuran dari tingkat kemiripan tersebut dapat digunakan menjadi salah satu pedoman dalam mempertimbangkan apakah suatu kode program telah melakukan tindakan plagiarisme atau tidak. Terdapat banyak jenis *similarity* pada kode program yang sering dijumpai pada pemrograman [14]. Deteksi *similarity* bisa dilihat dari penggantian nama kode program, seperti memindahkan posisi beberapa potongan kode serta mengganti nama variabel maupun *class, method* ataupun fungsinya. *Code Plagiarism* merupakan istilah yang digunakan untuk menggambarkan adanya kemiripan isi kode program antara dua kode berbeda. *Similarity* kode program dapat terjadi secara keseluruhan maupun sebagian, ataupun dengan sedikit membuat perubahan pada bagian yang tidak signifikan. Plagiarisme dapat memunculkan suatu bentuk tindakan dan konsekuensi dari

Building of Informatics, Technology and Science (BITS)
Volume 4, No 2, September 2022 Page: 405−412
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v4i2.1776

plagiarisme itu sendiri apabila *source* yang digunakan bukanlah *open source*. Karena bagaimanapun tindakan plagiarisme merupakan tindakan penipuan hasil karya orang lain tanpa sepengetahuan dari penulis aslinya, yang dapat melanggar suatu Hak Cipta dan Hak Moral [1].

Kelebihan dari teknik *machine learning* pada code *clone* detection antara lain *machine learning* memiliki kemampuan untuk belajar dari data (*training*) untuk mendeteksi atau mengelompokan data [15]. *Machine learning* juga memungkinkan bisa mendeteksi *code clone detection* tanpa intervensi manusia serta menangkap kompleksitas *code clone detection* sehingga bisa meningkatkan keakurasian dan kecepatan pendeteksian. Teknik *machine learning* yang sering digunakan pada system *code clone detection* antara lain *Decision Tree*, *Neural Network*, *Naïve Bayes*, SVM, Algoritma Genetik, KNN , K-*Means* dan *Fuzzy Logic* [15].

Namun, terdapat kelemahan dari beberapa algoritma *machine learning* tradisional antara lain: (1) Beberapa algoritma *Machine Learning* bekerja kurang efektif dengan bertambahnya jumlah data yang diproses; (2) Algoritma seperti SVM membutuhkan memori dan kompleksitas pemrosesan yang tinggi untuk data besar.; (3) *shallow learning architec*ture seperti deci*sion trees*, *support vector machines*, dan *case-based reasoning* [16]–[19] gagal mengekstrak informasi penting dari *input* yang memiliki struktur dan relasi yang komplek. ; (4) Metoda konvensional *machine learning* juga memiliki kelemahan dalam proses *feature learning* pada high dimensional data. Selain itu issue pemrosesan data dengan volume data besar menjadi issue penting pada dalam menggunakan algoritma *machine learning* tradisional [20].

Dengan meningkatnya jumlah data dan kompleksitas data, *deep learning* memberikan solusi untuk prediktif analisis big data, dengan kemampuan *prosesing* yang meningkat dan pemanfaatan *prosesor* grafis secara optimal. *Deep learning* memperlihatkan keberhasilan di domain big data antara lain di bidang *computer vision*, *speech recognition* dan pemrosesan bahasa alami [1]. *Deep Learning* berhasil meningkatkan model klasifikasi pada bidang tersebut. Di sisi lain *code clone detection* dengan *volume* data *massive*, bervariasi dan berkecepatan tinggi sangat memerlukan ektraksi fitur. Dengan potensi yang dimiliki oleh *deep learning* yang bisa mengektraksi fitur lebih baik maka teknik *deep learning* sesuai untuk *code clone detection* [21].

Kinerja dari algoritma *machine learning* terutama *deep learning* sangat tergantung dengan parameter seting. Sayangnya tidak mudah untuk mendapatkan perancangan arsitektur *deep learning* yang optimal. Proses *tuning hyperparameter* menjadi tantangan tersendiri untuk mendapatkan hasil yang optimal. Untuk meningkatkan kinerja pendeteksian serangan perlu dilakukan proses *fine tunning* berbagai parameter *code clone detection*. Sistem *code clone detection* telah dikembangkan dengan berbagai pendekatan. Beberapa penelitian mengembangkan *code clone detection* dikembangkan memiliki keterbatasan berbagai *platform*. Untuk itu perlu dilakukan eksperimen untuk melakukan *code clone detection* yang bisa memproses data dari berbagai *platform* bahasa pemrograman dengan memanfaatkan *deep learning*.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Kerangka kerja penelitian Eksperimen *Code Clone Detection Dengan Pemanfaatan Deep Neural Network* merepresentasikan tahapan proses yang dilakukan dalam penelitian agar penelitian dapat berjalan dengan baik dan tujuan yang telah ditetapkan dapat tercapai.

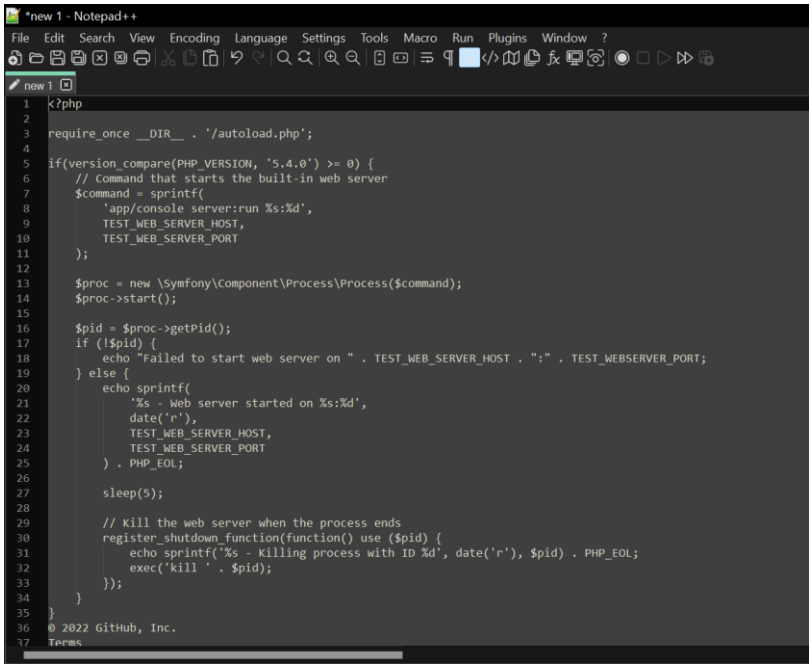
a. Penelitian Awal

Pada tahapan ini dikumpulkan bahan penelitian dari berbagai sumber pustaka, seperti buku, jurnal (baik cetak maupun online), prosiding, majalah, artikel dan sumber lain yang relevan. Masalah yang dibahas adalah bagaimana Eksperimen *Code Clone Detection Dengan Pemanfaatan Deep Neural Network*. Teori yang diambil tentang kode program, *deep learning*, *deep neural network*, pemrograman PHP, SQL dan perancangan UML dan akan diujikan guna melakukan eksperimen yang dibahas dalam penelitian ini.

b. Pengumpulan Dataset Kode Program untuk dipersiapkan dalam eksperimen.

Data diambil dari beberapa proyek perangkat lunak berupa kode program. Kode program ini akan ditransformasi menjadi dataset dengan *feature* adalah berupa *term* yang berasal dari sintak pemrograman. Selanjutnya data *testing* dan data *training* akan dibagi menjadi dua bagian yakni data *training* berasal dari kode program yang telah diberikan label sebagai kelas plagiat atau tidak. Gambar 1 berikut ini adalah cuplikan contoh file kode program dengan bahasa pemrograman PHP yang digunakan untuk dideteksi dengan dengan algoritma *Deep Neural Network* (DNN).

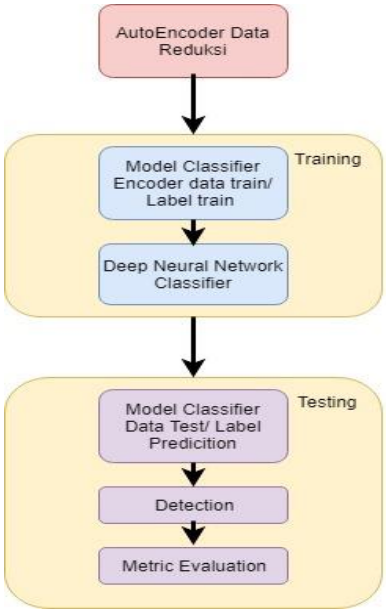
Building of Informatics, Technology and Science (BITS)
Volume 4, No 2, September 2022 Page: 405−412
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v4i2.1776



Gambar 1. *Code Clone Detection Dengan Pemanfaatan Deep Neural*

c. Merancang & Implementasi *Code Clone Detection Dengan Pemanfaatan Deep Neural Network.*

Kegiatan ini berupa pengukuran perangkat lunak yang mana hasil pengukuran tersebut digunakan dalam melakukan *Code Clone Detection Dengan Pemanfaatan Deep Neural Network*. Kemudian hasil deteksi akan dievaluasi dengan menggunakan Akurasi.



Gambar 2. Arsitektur *Code Clone Detection Dengan Pemanfaatan Deep Neural*

d. Melakukan Eksperimen dengan Kode program

Kegiatan ini melakukan eksperimen dengan cara melakukan komparasi terhadap Kode program terhadap data *cloning code* menggunakan *Deep Neural Network* sesuai dengan skema pada gambar 5 diatas. Pendeteksian *code clone* dilakukan dengan alur yang terdapat pada gambar 5 yakni dimulai dengan encoder data kode program (data training). Setelah dilakukan proses *auto enco*der data reduksi pada data *training* dilanjutkan dengan proses *training* dengan algoritma *deep neural network.* Setelah selesai di *training* selanjutnya proses *testing*. Proses testing ini terdiri dari model classifier dengan pengujian menggunakan data test untuk melakukan label prediction, selanjutnya dilaukan *detection* dan *metric evaluation* untuk mengevaluasi hasil eksperimen.

e. Hasil awal *Code Clone Detection* Dengan Pemanfaatan *Deep Neural Network.*

Kegiatan ini pelaporan hasil eksperimen awal dari alur sistem pada gambar 5 dengan cara melakukan komparasi terhadap Kode program terhadap data *cloning code* menggunakan *Deep Neural Network.*

Building of Informatics, Technology and Science (BITS)
Volume 4, No 2, September 2022 Page: 405−412
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v4i2.1776

f. *Progress Report* eksperimen.

Kegiatan ini dilakukan untuk melaporkan langkah kegiatan apa saja yang telah dilakukan setelah eksperimen awal diperoleh dengan cara melakukan komparasi terhadap Kode program terhadap data *cloning code* menggunakan *Deep Neural Network*.

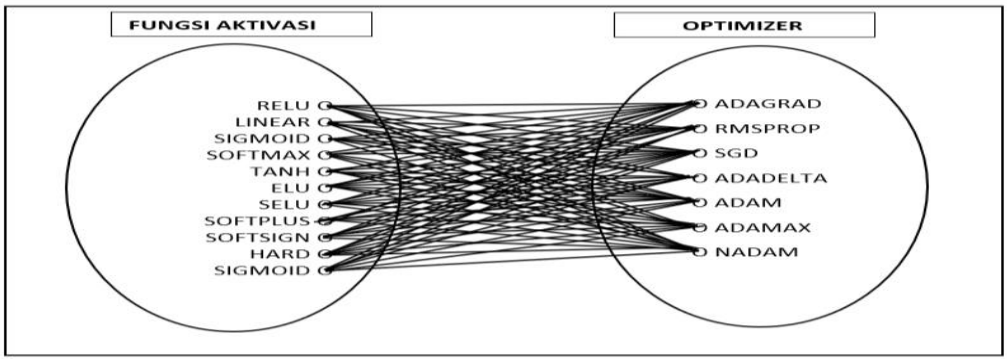
g. Evaluasi Akhir *Code Clone Detection* Dengan Pemanfaatan *Deep Neural Network.*

Pada kegiatan ini dilakukan evaluasi akhir terhadap *Code Clone Detection Dengan Pemanfaatan Deep Neural Network* dengan menggunakan sistem eksperimen penelitian yang telah dibangun. Hasil evaluasi dianalisis yakni dengan mengukur nilai Akurasi.

3. HASIL DAN PEMBAHASAN

3.1 Analisis Hasil Eksperimen pada *Deep Neural Network* untuk *Code Clone Detection*

Pada bagian ini kami menampilkan skema eksperimen yang dilakukan untuk melakukan *Code Clone Detection*. Model pemetaan yang kami buat adalah model pemetaan eksperimen yang dilakukan pada fungsi aktivasi dan optimizer dalam algoritma. Beberapa jenis metode fungsi aktivasi yang kami gunakan adalah Relu, Linear, Sigmoid, Softmax, Tanh, Elu, Selu, Softplus, Softsign, hard, and sigmoid. Untuk optimizer yang digunakan adalah Adagrad, RMSProp, SGD, Adadelta, Adam, Adamax dan Nadam.

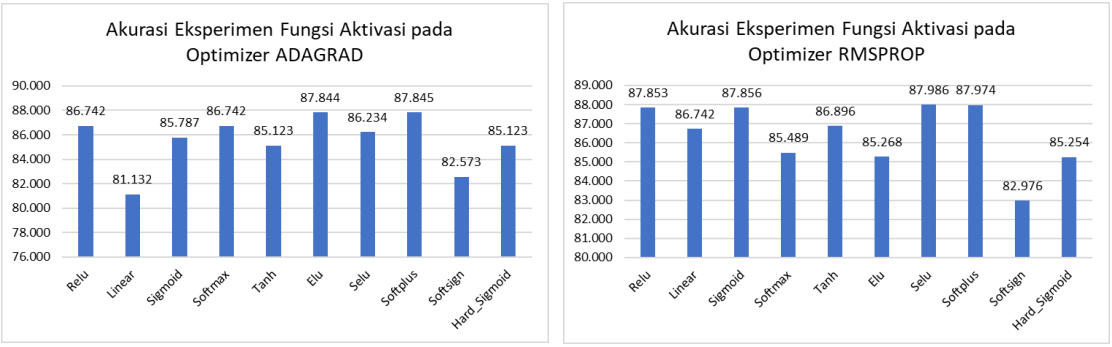


Gambar 3. Pemetaan fungsi aktivasi dan optimizer dalam DNN untuk *Code Clone Detection*

Pada gambar 3 ditampilkan gambar pemetaan skema eksperimen yang dituangkan dalam himpunan pada fungsi aktivasi dan optimizer. Pemetaan himpunan yang digambarkan pada gambar 3 di atas, menunjukan bahwa setiap jenis fungsi aktivasi dipetakan ke masing-masing optimizer. Dengan jumlah variasi fungsi aktivasi berjumlah 11 jenis dan jumlah optimizer sebanyak 7 jenis, maka total kombinasi hasil eksperimen fungsi aktivasi dan optimizer pada DNN ini berjumlah 77 variasi.

3.2 Analisis Hasil Pemilihan Fungsi Aktivasi pada Optimizer dalam DNN untuk *Code Clone Detection*.

Pada bagian berikut ini ditampilkan grafik hasil eksperimen setiap penggunaan fungsi aktivasi pada berbagia jenis optimizer yang digunakan dalam algoritma DNN untuk melakukan fungsi Deteksi Plagiarisme Kode Program atau *Code Clone Detection*. Berikut ini ditampilkan berbagai varian eksperimen fungsi aktivasi dan *optimizer* dalam DNN dengan pendeteksian file kode program berbahasa PHP. Satuan evaluasi yang dihitung adalah akurasi. Split data latih dan data uji yang digunakan adalah sebesar 65% dan satuan akurasi yang dihitung adalah persentasi.

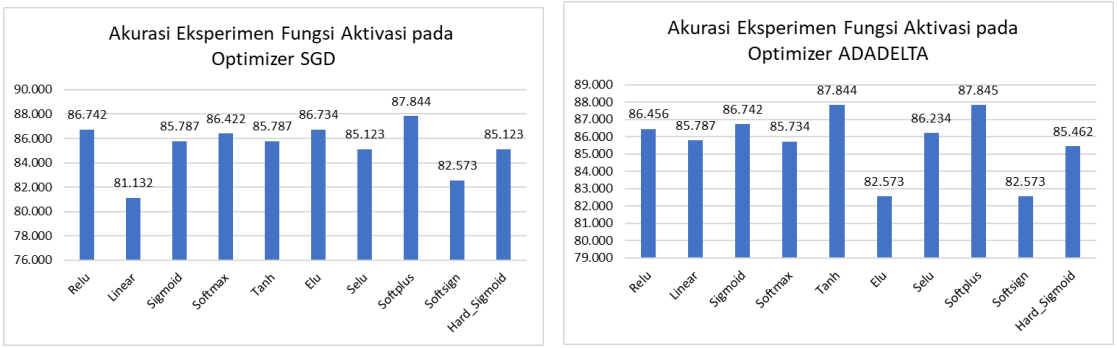


Gambar 4. Akurasi Eksperimen Fungsi Aktivasi pada *Optimizer* ADAGRAD dan RMSPROP

Gambar 4 di atas menampilkan hasil akurasi eksperimen fungsi aktivasi pada *optimizer Adagrad* dan *RMSProp*. Pada *optimizer Adagrad*, nilai akurasi tertinggi diperoleh pada fungsi aktivasi *Softplus* dengan nilai 87.845%. Sedangkan pada *optimizer RMSProp*, nilai tertinggi diperoleh pada fungsi aktivasi *Selu* dengan nilai sebesar 87.986%.

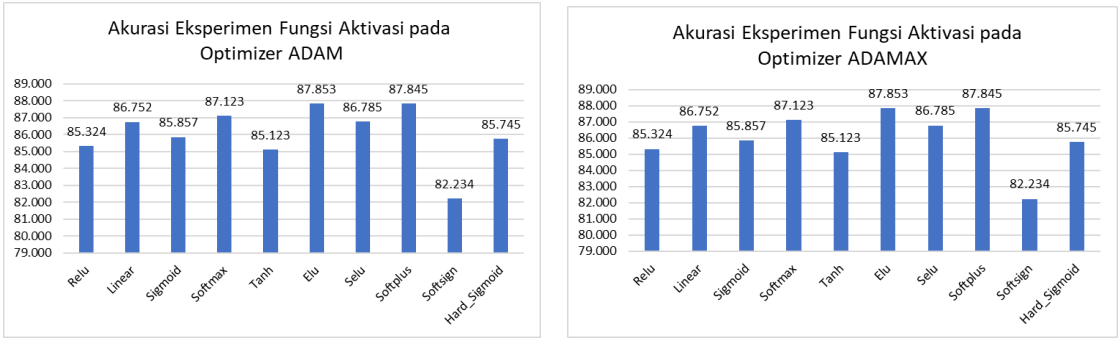
Building of Informatics, Technology and Science (BITS)
Volume 4, No 2, September 2022 Page: 405−412
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v4i2.1776

Nilai akurasi terendah pada *Optimizer Adagrad* adalah diperoleh oleh fungsi aktivasi *Linear* dengan nilai sebesar 81.132 %. Untuk nilai terendah pada *optimizer RMSprop* diperoleh oleh fungsi aktivasi *Softsign* sebesar 82.976%. Perbedaan perolehan akurasi ini dapat berbeda-beda sesuai dengan pemilihan *optimizer* yang diujikan. Hal ini dipengaruhi nilai variabel dan konstanta pada fungsi aktivasi. Sehingga berbeda skemanya, maka berbeda pula hasil relevansinya dengan label.



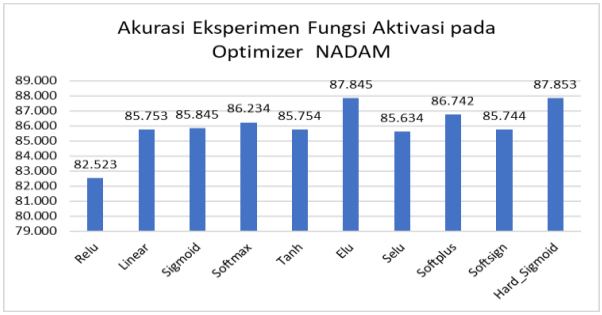
Gambar 5. Akurasi Eksperimen Fungsi Aktivasi pada *Optimizer* SGD dan ADADELTA

Gambar 5 di atas menampilkan hasil akurasi eksperimen fungsi aktivasi pada *optimizer SGD* dan *Adadelta*. Pada optimizer SGD, nilai akurasi tertinggi diperoleh pada fungsi aktivasi *Softplus* dengan nilai 87.844%. Sedangkan pada optimizer *Adadelta*, nilai tertinggi diperoleh pada fungsi aktivasi *Softplus* dengan nilai sebesar 87.845%. Nilai akurasi terendah pada *Optimizer SGD* adalah diperoleh oleh fungsi aktivasi Linear dengan nilai sebesar 81.132 %. Untuk nilai terendah pada *optimizer Adadelta* diperoleh oleh fungsi aktivasi *Softsign dan Elu* sebesar 82.573%. Perbedaan perolehan akurasi ini dapat berbeda-beda sesuai dengan pemilihan *optimizer* yang diujikan. Hal ini dipengaruhi nilai variabel dan konstanta pada fungsi aktivasi. Nilai pada Optimizer *SGD* dan *Adadelta* hasilnya mendekati optimizer yang dihasilkan *Adagrad* dan *RMSProp*.



Gambar 6. Akurasi Eksperimen Fungsi Aktivasi pada *Optimizer* ADAM dan ADAMAX

Gambar 6 di atas menampilkan hasil akurasi eksperimen fungsi aktivasi pada optimizer *Adam* dan *Adamax*. Pada optimizer *Adam*, nilai akurasi tertinggi diperoleh pada fungsi aktivasi Softplus dengan nilai 87.845%. Sedangkan pada optimizer *Adamax*, nilai tertinggi diperoleh pada fungsi aktivasi Selu dengan nilai sebesar 87.845%. Nilai akurasi terendah pada Optimizer *Adam* adalah diperoleh oleh fungsi aktivasi *Softsign* dengan nilai sebesar 81.234 %. Untuk nilai terendah pada optimizer *Adamax* diperoleh oleh fungsi aktivasi *Softsign* sebesar 82.234%. Perbedaan perolehan akurasi ini dapat berbeda-beda sesuai dengan pemilihan optimizer yang diujikan. Nilai pada Optimizer *Adam* sama dengan optimizer yang dihasilkan *Adamax*.



Gambar 7. Akurasi Eksperimen Fungsi Aktivasi pada *Optimizer* NADAM

Gambar 7 di atas menampilkan hasil akurasi eksperimen fungsi aktivasi pada optimizer *Nadam*. Pada optimizer *Nadam*, nilai akurasi tertinggi diperoleh pada fungsi aktivasi *Hard Sigmoid* dengan nilai 87.853%. Nilai akurasi

Building of Informatics, Technology and Science (BITS)
Volume 4, No 2, September 2022 Page: 405−412
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v4i2.1776

terendah pada *Optimizer Adagrad* adalah diperoleh oleh fungsi aktivasi Relu dengan nilai sebesar 82.523 %. Untuk optimizer nadam terdapat perbedaan hasil dengan optimizer yang lain. Fungsi aktivasi yang bernilai tertinggi maupun terendah akurasinya berbeda sama sekali denga fungsi aktivasi yang meghasilkan nilai tertinggi dan terendah setelah diujikan pada jenis optimizer yang berbeda.

3.4 Analisis Hasil Pemilihan *Neuron* Terbaik pada DNN untuk *Code Clone Detection*.

Tabel 1. Pemilihan *Neuron* Terbaik pada DNN untuk *Code Clone Detection*

Jumlah neuron	Akurasi (%)
N=100	81.252
N=1000	84.743
N=500	**88.123**
N=750	83.845
N=250	81.234

Dari hasil pemilihan fungsi aktivasi terbaik pada seluruh jenis optimizer dihasilkan bahwa fungsi aktivasi yang digunakan adalah *Selu* dengan optimizer *RMSProp*. Nilai akurasi yang dihasilkan pada eksperimen jumlah *neuron* dengan variasi 100, 1000, 500, 750, dan 250 adalah 88.123% sebagai nilai tertinggi. Nilai tertinggi yang diperoleh pada eksperimen jumlah neuron adalah N=500. Selanjutnya, dilakukan eksperimen untuk menentukan jumlah hidden layer terbaik yang akan diseleksi.

3.5 Analisis Hasil Pemilihan *Hidden Layer* Terbaik pada DNN untuk *Code Clone Detection*.

Tabel 2. Pemilihan *Hidden Layer* Terbaik pada DNN untuk *Code Clone Detection*

No	Jumlah *Hidden Layer*	*Neuron*	Akurasi (%)
1	5	500	81.864
2	**10**	**500**	**89.674**
3	15	500	80.232
4	5	250	83.251
5	10	250	85.623
6	15	250	84.134
7	5	100	86.732
8	10	100	83.120
9	15	100	83.341

Tabel 2 diatas adalah nilai akurasi yang dihasilkan untuk memilih jumlah hidden layer yang ada. Pemilihan jumlah hidden layer berdasarkan dari hasil pemilihan fungsi aktivasi terbaik pada seluruh jenis optimizer yakni *Selu* dengan optimizer *RMSProp*. Nilai akurasi yang dihasilkan pada eksperimen jumlah *hidden layer* dengan variasi 1, 10 dan 15 adalah 89.674% sebagai nilai tertinggi. Nilai tertinggi yang diperoleh pada eksperimen jumlah *neuron* adalah N=500. Selanjutnya, dilakukan eksperimen untuk menentukan jumlah *Epoch* dan *Neuron* terbaik yang akan diseleksi.

Tabel 3. Pemilihan *Epoch* dan *Neuron* Terbaik pada DNN untuk *Code Clone Detection*

Jumlah *Epoch*	Jumlah *Neuron* per *Layer*	Akurasi (%)
100	500	83.341
500	500	90.121
1000	**500**	**90.132**
100	250	83.314
500	250	88.312
1000	250	81.851

Tabel 3 adalah Tabel akurasi hasil pemilihan *Epoch* dan *Neuron* pada DNN untuk *Code Clone Detection*. Hasil pemilihan berdasarkan optimizer fungsi aktivasi terpilih, yakni *Selu* dengan optimizer *RMSProp*. Nilai akurasi yang dihasilkan pada eksperimen jumlah *neuron* dengan variasi 100, 1000, 500, 750, dan 250 adalah 90.132% sebagai nilai tertinggi. Nilai tertinggi yang diperoleh pada eksperimen jumlah neuron adalah N=500 dan jumlah epoch =1000. Selanjutnya, dilakukan eksperimen untuk menentukan jumlah hidden layer terbaik yang akan diseleksi.

Tabel 4. Data *Actual* dan *Predicted* dengan Bahasa PHP menggunakan Hasil Seleksi Atribut Terbaik

Dataset: 100 File Kode Program (PHP)			
		Predicted	
		Negatif	**Positif**
Actual	**Negatif**	**1**	**6**
	Positif	**4**	**89**

Building of Informatics, Technology and Science (BITS)
Volume 4, No 2, September 2022 Page: 405−412
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v4i2.1776

Tabel 4 adalah hasil akurasi dari jumlah Data *Actual* dan *Predicted* dengan file kode program PHP. Hasil klasifikasi label terdeteksi plagiat dan tidak berdasarkan dari nilai pemilihan fungsi aktivasi terbaik yakni *Selu* dengan optimizer *RMSProp*. Nilai data yang terklasifikasi benar dengan label data sesuai denga nisi data tabel 4 di atas.

Tabel 5. Tabel Jumlah Data *Confussion Matriks* Hasil Seleksi Atribut Terbaik

Matriks	Nilai
True Negative(TN)	1
True Positive (TP)	89
False Negative (FN)	4
False Positive (FP)	6

Tabel 5 merupakan data pada hasil klasifikasi 100 file kode program berbahasa PHP. Matriks yang dihasilkan antara lain nilai *True Negative* (TN) sebanyak 1 data, *True Positive* (TP) sebanyak 89 data, *False Negative* (FN) sebanyak 4 data, dan *False Positive* (FP) sebanyak 6 data. Secara keseluruhan nilai *true positif* yang tinggi merepresentasikan bahwa data klasifikasi yang digunakan sebagai data uji dan data latih adalah data yang bagus. Begitu juga algoritma DNN yang digunakan mampu memberikan nilai klafisikasi yang baik. Dengan nilai *true negative* hanya 1 data, artinya hamper seluruh data yang dilabeli diplagiat, memang benar merupakan *code clone.*

Tabel 6. Evaluasi Confussion Matrik Code Clone Detection dengan DNN

Evaluasi	Perhitungan	Nilai
Recall Sensitivity True Positive Rate (TPR)	TP /(FN+TP)	0.956989
False Positive Rate (FPR) *False Alarm rate*	FP / (TN +FP)	0.857143
Specificity True Negative Rate (TNR)	TN/(TN +FP)	0.142857
Precision	TP/(TP +FP)	0.936842
False Negative Rate (FNR)	FN / (FN +TP)	0.043011
Accuracy	(TP+TN)/(TP+TN+FP+FN)	0.9000

Nilai akhir evaluasi yang terdapat pada tabel 6 merupakan hasil pemilihan fungsi aktivasi terbaik yakni *Selu* dengan optimizer *RMSProp*. Nilai *Recall Sensitivity True Positive Rate* (TPR) dengan hitungan variabel TP /(FN+TP) memperoleh nilai 0.956989. Nilai *False Positive Rate* (FPR) *False Alarm rate* dengan hitungan variabel FP / (TN +FP) memperoleh nilai 0.857143. Nilai *Specificity True Negative Rate* (TNR) dengan hitungan variabel TN/(TN +FP) memperoleh nilai 0.142857. Nilai *Precision* dengan hitungan variabel TP/(TP +FP) memperoleh nilai 0.936842. Nilai *False Negative Rate* (FNR) dengan hitungan variabel FN / (FN +TP) memperoleh nilai 0.043011. Nilai *Accuracy* dengan hitungan variabel (TP+TN)/(TP+TN+FP+FN) memperoleh nilai 0.9000.

4. KESIMPULAN

Berdasarkan hasil eksperimen yang dilakukan pada 100 file data kode program berbahasa PHP, dengan eksperimen beberapa jenis metode fungsi aktivasi dan optimizer. Nilai rata-rata akurasi yang dihasilkan adalah baik. Dengan variasi fungsi aktivasi yang kami gunakan seperti *Relu, Linear, Sigmoid, Softmax, Tanh, Elu, Selu, Softplus, Softsign, hard*, dan *sigmoid*, serta variasi *optimizer* yang digunakan adalah *Adagrad, RMSProp, SGD, Adadelta, Adam, Adamax* dan *Nadam*, pemilihan atribut terbaik adalah pada fungsi *Selu* dan optimizer *RMSProp*. Jumlah *Epoch* yang digunakan adala 1000, jumlah *neuron per layer* adalah 500 dan jumlah *hidden layer* terbaik adalah 10, akurasi rata-rata memeproleh nilai 0.900.

REFERENCES

[1] L. Li, H. Feng, W. Zhuang, N. Meng, and B. Ryder, "CCLearner: A deep learning-based clone detection approach," *Proc. - 2017 IEEE Int. Conf. Softw. Maint. Evol. ICSME 2017*, pp. 249–260, 2017, doi: 10.1109/ICSME.2017.46.

[2] L. Nichols, M. Emre, and B. Hardekopf, *Structural and nominal cross-language clone detection*, vol. 11424 LNCS, no. 2. Springer International Publishing, 2019.

[3] Q. U. Ain, W. H. Butt, M. W. Anwar, F. Azam, and B. Maqbool, "A Systematic Review on Code Clone Detection," *IEEE Access*, vol. 7, pp. 86121–86144, 2019, doi: 10.1109/ACCESS.2019.2918202.

[4] Fachruddin, Saparudin, E. Rasywir, and Y. Pratama, "Network and layer experiment using convolutional neural network for content based image retrieval work," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 20, no. 1, pp. 118–128, 2022, doi: 10.12928/TELKOMNIKA.v20i1.19759.

[5] M. R. Borroek, E. Rasywir, Y. Pratama, Fachruddin, and M. Istoningtyas, "Analysis on Knowledge Layer Application for Knowledge Based System," in *Proceedings of 2018 International Conference on Electrical Engineering and Computer Science, ICECOS 2018*, 2019, pp. 177–182, doi: 10.1109/ICECOS.2018.8605262.

[6] Y. Pratama and E. Rasywir, "Automatic Cost Estimation Analysis on Datawarehouse Project with Modified Analogy Based Method," in *Proceedings of 2018 International Conference on Electrical Engineering and Computer Science, ICECOS 2018*, 2019, pp. 171–176, doi: 10.1109/ICECOS.2018.8605195.

[7] E. B. Prasetya, "Deteksi Similarity Source Code Menggunakan Metode Deteksi Abstract Syntax Tree," in *Seminar Nasional Sains dan Teknologi 2014*, 2014, no. November, pp. 1–7.

Building of Informatics, Technology and Science (BITS)
Volume 4, No 2, September 2022 Page: 405−412
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v4i2.1776

[8] K. W. Nafi, T. S. Kar, B. Roy, C. K. Roy, and K. A. Schneider, "CLCDSA: Cross language code clone detection using syntactical features and API documentation," *Proc. - 2019 34th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2019*, pp. 1026–1037, 2019, doi: 10.1109/ASE.2019.00099.

[9] Y. Wu *et al.*, "SCDetector: Software Functional Clone Detection Based on Semantic Tokens Analysis," *Proc. - 2020 35th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2020*, pp. 821–833, 2020, doi: 10.1145/3324884.3416562.

[10] A. Walker, T. Cerny, and E. Song, "Open-source tools and benchmarks for code-clone detection," *ACM SIGAPP Appl. Comput. Rev.*, vol. 19, no. 4, pp. 28–39, 2020, doi: 10.1145/3381307.3381310.

[11] E. B. Prasetya and A. F. D. Jalal, "Deteksi Similarity Source Code Menggunakan Metode Deteksi Abstract Syntax Tree," in *Prosiding Semnastek*, 2014, pp. 1–7.

[12] Q. Zheng, X. Tian, M. Yang, and H. Wang, "Differential learning: A powerful tool for interactive content-based image retrieval," *Eng. Lett.*, vol. 27, no. 1, pp. 202–215, 2019.

[13] J. Zeng, K. Ben, X. Li, and X. Zhang, "Fast code clone detection based on weighted recursive autoencoders," *IEEE Access*, vol. 7, pp. 125062–125078, 2019, doi: 10.1109/ACCESS.2019.2938825.

[14] M. A. Nishi and K. Damevski, "Scalable code clone detection and search based on adaptive prefix filtering," *J. Syst. Softw.*, vol. 137, pp. 130–142, 2018, doi: 10.1016/j.jss.2017.11.039.

[15] M. White, M. Tufano, C. Vendome, and D. Poshyvanyk, "Deep learning code fragments for code clone detection," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 87–98, doi: 10.1145/2970276.2970326.

[16] P. G. V. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, and R. Buyya, "FOCAN: A Fog-supported smart city network architecture for management of applications in the Internet of Everything environments," *J. Parallel Distrib. Comput.*, 2018, doi: 10.1016/j.jpdc.2018.07.003.

[17] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J. S. Oh, "Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 624–635, 2018, doi: 10.1109/JIOT.2017.2712560.

[18] J. You, W. Liu, and J. Lee, "A DNN-based semantic segmentation for detecting weed and crop," *Comput. Electron. Agric.*, vol. 178, no. September, p. 105750, 2020, doi: 10.1016/j.compag.2020.105750.

[19] A. B. Adege and H. Lin, "applied sciences Applying Deep Neural Network (DNN) for Robust Indoor Localization in Multi-Building Environment," *applsci*, vol. 8, pp. 1–14, 2018, doi: 10.3390/app8071062.

[20] J. Huang, Y. F. Li, and M. Xie, "An empirical analysis of data preprocessing for machine learning-based software cost estimation," *Inf. Softw. Technol.*, vol. 67, pp. 108–127, 2015, doi: 10.1016/j.infsof.2015.07.004.

[21] H. H. Wei and M. Li, "Supervised deep features for Software functional clone detection by exploiting lexical and syntactical information in source code," *IJCAI Int. Jt. Conf. Artif. Intell.*, pp. 3034–3040, 2017.