Jurnal Teknik Informatika (JUTIF)
Vol. 1, No. 2, Desember 2020, hlm. 61-68

DOI: https://doi.org/10.20884/1.jutif.2020.1.2.18
p-ISSN: 2723-3863
e-ISSN: 2723-3871

CONVOLUTIONAL NEURAL NETWORK PADA KLASIFIKASI SIDIK JARI MENGGUNAKAN RESNET-50

Novelita Dwi Miranda[*1], Ledya Novamizanti[2], Syamsul Rizal[3]

[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom, Bandung
Email: [1]novelitadwimiranda@student.telkomuniversity.ac.id, [2]ledyaldn@telkomuniversity.ac.id, [3]syamsul@telkomuniversity.ac.id

(Naskah masuk: 29 Juli 2020, diterima untuk diterbitkan: 03 Agustus 2020)

Abstrak

Pengenalan sidik jari merupakan bagian dari teknologi biometrik. Klasifikasi sidik jari yang paling popular adalah *Henry classification system*. Henry membagi sidik jari berdasarkan garis polanya menjadi lima kelas yaitu *arch* (A), *tented arch* (T), *left loop* (L), *right loop* (R), dan *whorl* (W). Penelitian ini menggunakan *Convolutional Neural Network* (CNN) dengan model arsitektur *Residual Network-50* (ResNet-50) untuk mengembangkan sistem klasifikasi sidik jari. Dataset yang digunakan diperoleh dari *website National Institute of Standards and Technology* (NIST) berupa citra sidik jari *grayscale* 8-bit. Hasil pengujian menunjukkan bahwa pemrosesan awal *Contrast Limited Adaptive Histogram Equalization* (CLAHE) dalam model CNN dapat meningkatkan performa akurasi dari sistem klasifikasi sidik jari sebesar 11,79%. Pada citra tanpa CLAHE diperoleh akurasi validasi 83,26%, sedangkan citra dengan CLAHE diperoleh akurasi validasi 95,05%.

Kata kunci: CLAHE, CNN, *henry classification system, Resnet-50, sidik jari*

CLASSIFICATION OF FINGERPRINT PATTERN USING CONVOLUTIONAL NEURAL NETWORK IN CLAHE IMAGE

Abstract

Fingerprint recognition is part of biometric technology. The most popular fingerprint classification is the Henry classification system. Henry divides fingerprints based on their pattern lines into five classes namely arch (A), tented arch (T), left loop (L), right loop (R), and whorl (W This study uses a Convolutional Neural Network (CNN) with a Residual Network-50 (ResNet-50) architectural model to develop a fingerprint classification system. The dataset used was obtained from the National Institutes of Standards and Technology (NIST) website in the form of an 8-bit grayscale fingerprint image. The test results show that the initial processing of Contrast Limited Adaptive Histogram Equalization (CLAHE) in the CNN model can improve the accuracy performance of the fingerprint classification system by 11.79%. In the image without CLAHE, the validation accuracy is 83.26%, while the image with CLAHE has the validation accuracy of 95.05%.

Keywords: *CLAHE, CNN, fingerprint, henry classification system, Resnet-50*

1. PENDAHULUAN

Biometrik merupakan teknologi untuk mengidentifikasi karakteristik unik pada diri manusia. Karakteristik unik tersebut antara lain pola sidik jari, bentuk geometri tangan, kunci frekuensi suara, pola iris, dan retina mata yang umumnya berbeda pada setiap individu [1]. Cara kerja teknologi biometrik adalah dengan deteksi pola, sehingga sering digunakan sebagai sistem keamanan untuk menjaga kerahasiaan data identitas seseorang. Salah satu teknologi biometrik yang sering digunakan dalam sistem keamanan adalah identifikasi sidik jari.

Pada tahun 1892, Sir Francis Galton menetapkan bahwa sidik jari merupakan ciri perseorangan yang tidak berubah. Galton merupakan orang pertama yang melakukan penelitian tentang sidik jari. Pada tahun 1901, Sir Edwar Henry mengembangkan metode perumusan Galton yang dikenal dengan "*Henry classification system*". Henry telah mengklasifikasikan pola sidik jari menjadi lima kategori yaitu *arch* (A), *tented arch* (T), *left loop* (L), *right loop* (R), dan *whorl* (W) [2].

Deteksi pola sidik jari ini dimanfaatkan oleh pihak kepolisian dalam membuat rumus sidik jari. Rumus tersebut digunakan sebagaai identitas data kriminal seseorang. Namun hingga saat ini, proses pendeteksian pola sidik jari masih dilakukan secara manual, yaitu dengan mengamati satu per satu sidik jari. Hal ini sangat tidak efisien dari sisi waktu dan

bergantung pada kemampuan individu. Untuk memudahkan proses pengenalan pola sidik jari, maka dirancang sistem klasifikasi sidik jari secara otomatis menggunakan pengolahan citra digital.

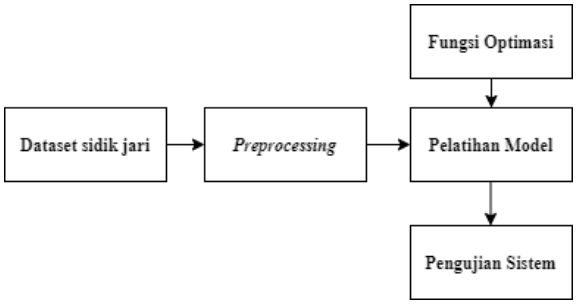
Deep learning sudah terbukti akurat dalam berbagai penelitian klasifikasi citra sidik jari. Beberapa algoritma *deep learning* yang sering digunakan seperti *Recurrent Neural Network* (RNN), *Generative Adversarial Networks* (GAN), dan *Convolutional Neural Network* (CNN). John M. Shrein [3] menggunakan *preprocesing* pada CNN dengan arsitektur LeNet-5, sehingga diperoleh akurasi 95,9%. *Preprocessing* dapat meningkatkan tingkat akurasi. Xiaomeng Guo, Fan Wu, dan Xiaoyong Tang [4] menggunakan metode CNN dengan memodifikasi arsitektur CaffeNet menjadi FCTP-Net yang terdiri dari 4 *convolutional layer*, 3 *max-pooling layer*, dan 3 *fully-connected layer*. Sehingga didapatkan akurasi sebesar 91,5%. Hasil ini lebih tinggi dibanding arsitektur LeNet sebesar 16,69%, AlexNet sebesar 57,84, dan CaffeNet sebesar 81,77%. Pada penelitian [4], perubahan jumlah layer pada CNN mempengaruhi performa sistem.

Penelitian oleh P. Nahar, S.Tanwani, dan NS Chaudhari [5] dengan membandingkan arsitektur ResNet-50, CoarseNet, FineNet, dan Unified FingerNet tanpa *preprocessing* sehingga didapatkan akurasi tertinggi pada ResNet-50 sebesar 90%, CoarseNet dan FineNet sebesar 73,45%, serta Unified FingerNet sebesar 89%, sebanding dengan penelitian Mubeen Ghafoor dkk [6] menggunakan arsitektur ResNet-50 dan mendapat akurasi 91,3%. Penelitian oleh Wang-Su Jeon dan Sang-Yong Rhee [7] menggunakan tiga bentuk model. Hasil pengujian diperoleh bahwa model 1 berupa arsitektur VGGNet dasar memiliki akurasi 82,1% dengan waktu belajar 8 jam 21 menit, model 2 yaitu VGGNet dengan *preprocessing* dihasilkan akurasi 94,2% dengan waktu 8 jam 48 menit, dan model 3 yaitu VGGNet dengan teknik ensemble bining menghasilkan akurasi 98,3% dengan waktu 10 jam 2 menit. Pada penelitian [5],[6], ResNet-50 memiliki hasil akurasi yang cukup baik untuk percobaannya. Pada penelitian [3],[7], gambar yang dilakukan *preprocessing* hasilnya memiliki tingkat akurasi yang lebih tinggi dibanding tanpa *preprocessing*.

Berdasarkan hasil penelitian-penelitian tersebut, maka penelitian ini merancang sistem untuk klasifikasi sidik jari melalui citra digital yang melalui tahap *preprocessing* terlebih dahulu kemudian diklasifikasi menggunakan metode CNN dengan arsitektur ResNet-50. Klasifikasi menggunakan 5 kelas sesuai *Henry classification system*, yaitu: *arch*, *left loop*, *right loop*, *tented arch*, dan *whorl*.

2. METODE PENELITIAN

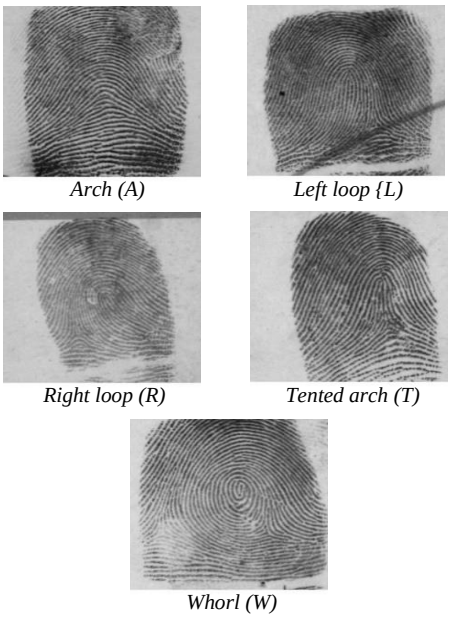
Penelitian ini dibagi menjadi 4 tahap, yaitu tahap pengambilan data, tahap *preprocessing* yaitu proses peningkatan kualitas citra. Tahap selanjutnya yaitu klasifikasi menggunakan CNN dengan arsitektur ResNet-50. Kemudian tahap pengujian sistem menggunakan dua skenario pengujian. Gambar 1 merupakan blok diagram tahapan penelitian.



Gambar 1. Metode Penelitian

2.3. Dataset Sidik Jari

Dataset dalam penelitian ini didapatkan melalui website *National Institute of Standards and Technology* (NIST) berupa citra sidik jari *grayscale* 8-bit berukuran 512×512 piksel [8]. Total dataset yang digunakan sejumlah 2100 citra dengan 429 citra berpola *arch*, 403 citra berpola *tented arch*, 402 citra berpola *left loop*, 410 citra berpola *right loop*, dan 456 citra berpola *whorl*. Hal tersebut dengan mempertimbangkan kualitas citra, yaitu kejelasan garis pola, dan ukuran pola pada citra. Gambar 2 merupakan contoh dataset dari NIST untuk setiap kelas sidik jari.



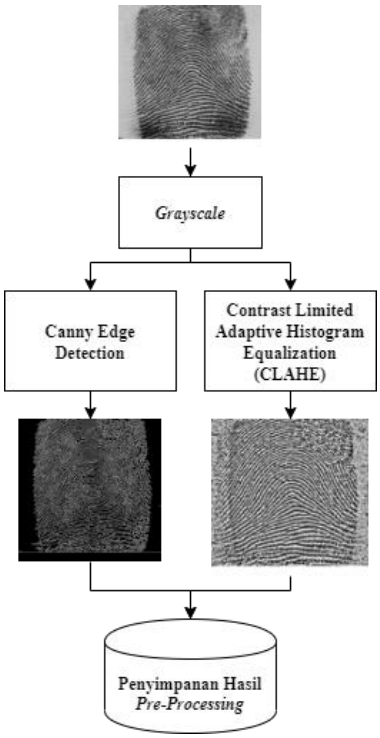
Gambar 2. Dataset sidik jari NIST

Arch adalah bentuk pokok sidik jari yang semua garisnya datang dari satu sisi pola, cenderung mengalir ke sisi yang lain, dan bergelombang naik di tengah-tengah. Pola *arch* diklasifikasikan menjadi 4 tipe yaitu *radial arch*, *tented arch*, *plain arch*, dan

ulnar arch. Hampir 50% dari seluruh sidik jari terdiri dari pola *arch*. *Loop* adalah bentuk pokok sidik jari dimana satu garis atau lebih datang dari salah satu sisi pola. Bentuk *loop* melengkung dan menyentuh suatu garis bayangan yang ditarik antara delta dan *core* (inti), serta cenderung kembali ke sisi semula. Pola *loop* diklasifikasikan menjadi 4 tipe yaitu *plain loop, lateral pocket loop, twinned loop,* dan *a central packet loop*. Persentasenya 60% hingga 65% dari seluruh sidik jari terdiri dari pola *loop*. *Whorl* adalah bentuk pokok sidik jari yang membentuk formasi melingkar di sekitar *core*. Pola *whorl* diklasifikasikan lagi menjadi 4 tipe yaitu *plain whorl, central pocket loop whorl, accidental whorl,* dan *double pocket loop whorl.* Persentasenya sekitar 30% hingga 35% dari seluruh sidik jari terdiri dari bentuk *whorl* [9].

2.4. *Preprocessing*

Pada penelitian ini, sistem klasifikasi sidik jari ditambahkan tahap *preprocessing*, agar akurasi dapat lebih optimal. *Preprocessing* bertujuan untuk meningkatkan kualitas citra, sehingga memudahkan dan mempercepat kinerja sistem dalam mengenali pola sidik jari. Semua dataset dilakukan *grayscale* kemudian diproses dengan *canny edge detection* dan *Contrast Limited Adaptive Histogram Equalization* (CLAHE). Gambar 3 menjelaskan alur kerja pada tahap *preprocessing.*



Gambar 3. Alur kerja tahap *preprocessing*

Citra *grayscale* merupakan citra hasil proses normalisasi dari 3-*layer* dari citra berwarna (RGB) menjadi 1 *layer*. Citra *grayscale* berisikan matriks data yang nilai-nilai didalamnya mewakili intensitas setiap piksel dengan nilai antara 0 sampai 255. Jika nilai semakin mendekati 0 artinya mendekati warna hitam dan jika nilai semakin mendekati 255 artinya mendekati warna putih. Setiap piksel pada citra *grayscale* membutuhkan 8-bit memori [10].

Canny edge detection merupakan deteksi tepi yang bertujuan mencari garis tepi pada suatu citra secara optimal. Canny hanya memberi satu tanggapan untuk satu tepi dengan memilih titik tepi berdasarkan pendekatan *threshold.* Deteksi tepi ini mampu menghilangkan *noise* karena menggunakan tapis Gaussian pada awal proses, tahap ini disebut *denoise* yang dapat dihitung dengan persamaan (1).

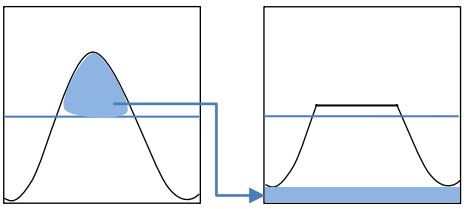
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (1)$$

dengan $\frac{1}{2\pi\sigma^2}$ merupakan konstanta koefisien normalisasi Gaussian. Kemudian menghitung amplitudo gradien dan arah gradien citra. Arah gradien mengambil 4 parameter suduk yaitu 0^o, 45^o, 90^o, dan 135^o, secara matematis dapat didefinisikan dengan persamaan (2) dan (3).

$$G = \sqrt{{G_x}^2 + {G_y}^2} \quad (2)$$

$$\theta = \tan^{-1}\left(\frac{G_x}{G_y}\right) \quad (3)$$

dengan G_x dan G_y adalah sepasang array konvolusi. citra. Lalu terjadi penindasan yang menghilangkan piksel non-tepi dan hanya menyisakan beberapa bagian. Dan terakhir dilakukan pemilihan *hysteresis threshold.* Sehingga canny dapat memberikan nilai *loss* yang rendah, karena piksel-piksel tepi yang ditemukan saat deteksi dan tepi sebenernya berjarak sangat pendek [11].



Gambar 4 Distribusi *excess pixel* ke area bawah *clip limit*

CLAHE merupakan pengembangan dari metode *Adaptive Histogram Equalization* (AHE). CLAHE memberikan solusi pada masalah peningkatan kontras yang berlebihan pada AHE dengan memberikan nilai batas atau *clip limit* pada histogram. *Clip limit* menyatakan batas maksimum tinggi suatu histogram, dapat dihitung dengan persamaan (4).

$$\beta = \frac{M}{N}\left(1 + \frac{\alpha}{100}(s_{max} - 1)\right) \quad (4)$$

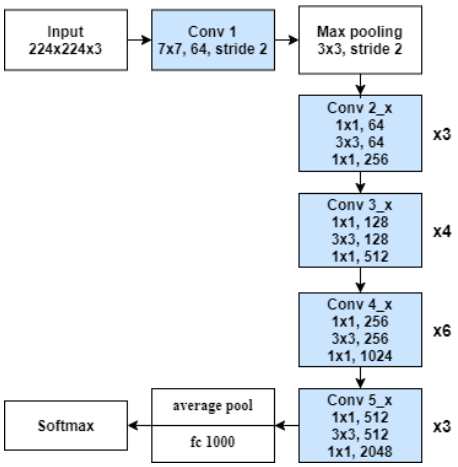
dengan *M* adalah luas *region size*, *N* adalah nilai *grayscale* (256), dan α adalah *clip factor* yang

menyatakan penambahan batas limit histogram bernilai antara 0-100. Histogram yang memiliki nilai diatas nilai *clip limit* dianggap sebagai *excess pixel* [12]. Akibatnya histogram jadi merata, karena didistribusikan ke area bawah *clip limit* yang diilustrasikan pada Gambar 4.

2.5. Pelatihan Model

Pelatihan model merupakan proses pelatihan mengenali objek dan mengklasifikasikannya sesuai dengan kelasnya. Penelitian ini menggunakan salah satu algoritma *deep learning* yaitu *Convolutional Neural Network* (CNN) dengan arsitektur ResNet-50. CNN merupakan salah jenis jaringan saraf yang bisa digunakan untuk mendeteksi objek pada sebuah citra. Alur kerja CNN untuk klasifikasi citra adalah dengan mengelompokkan citra tersebut berdasarkan kesamaannya, kemudian dilakukan pengenalan objek dalam beberapa *sample*. Gambar 5 merupakan blok diagram CNN dengan arsitektur ResNet-50.

Secara umum arsitektur CNN dibagi menjadi dua bagian yaitu *feature detection layers* dan *classification layer*. *Feature detection layer* melakukan perubahan citra menjadi angka yang kemudian dilakukan perhitungan matriksnya. *Features detection layer* menampilkan tiga tipe operasi pada sebuah data input yaitu *convolutional layer*, *pooling layer*, dan *rectified linear unit* (ReLU). Ketiga operasi ini dilakukan secara berulang, dengan setiap lapisan mempelajari untuk mendeteksi fitur yang berbeda. Pada *classification layer* terdapat *fully connected layer* yang menghasilkan keluaran vektor dimensi K. Dimensi K adalah jumlah kelas yang dapat membuat jaringan tersebut dapat memprediksi. Vektor ini berisi probabilitas untuk setiap kelas dari citra yang diklasifikasikan.



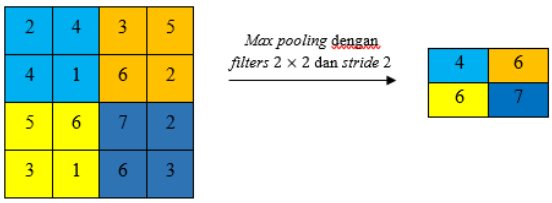
Gambar 5. Blok diagram CNN ResNet-50

Convolution layer menempatkan citra input melalui serangkaian *convolutional* filter, yang masing-masing mengaktifkan fitur tertentu dari sebuah citra [13]. Parameter yang digunakan pada Gambar 5 dalam *convolution layer* adalah *stride*. Perhitungan konvolusi dapat menggunakan persamaan (5).

$$h(x) = f(x) * g(x) \quad (5)$$

dengan $h(x)$ adalah *weight parameter*, hasil konvolusi dari $f(x)$ dan $g(x)$. $f(x)$ adalah citra input dan $g(x)$ adalah filter.

Pooling adalah proses menyederhanakan keluaran dengan melakukan *down sampling nonlinear* dan mengurangi jumlah parameter. *Pooling* digunakan untuk menghilangkan informasi tidak penting. Terdapat dua jenis *pooling* yang sering digunakan, yaitu *average pooling* dan *max pooling*. Konsep *max pooling* yaitu membuat matriks baru berukuran lebih kecil dengan cara mengambil fitur terbesar pada sebuah layer. Dalam implementasinya, *max pooling* mengambil piksel terbesar yang kemudian disusun menjadi matriks baru [13], diilustrasikan pada Gambar 6.



Gambar 6. Konsep *Max Pooling*

Gambar 6 menunjukkan operasi *max pooling* pada citra berukuran 4×4, menggunakan filter 2×2, dan nilai *stride* 2 yang menandakan pergeseran baris dan kolom sebanyak 2.

Rectified Linear Unit (ReLU) berfungsi untuk memungkinkan pelatihan data lebih cepat dan efektif dengan memetakan nilai negatif ke nol dan mempertahankan nilai positif [13]. ReLU dapat didefinisikan dengan persamaan (6).

$$f(x) = max(0, x) \quad (6)$$

Fully connected layer merupakan lapisan dimana semua saraf aktivasi dari lapisan sebelumnya terhubung dengan saraf pada lapisan selanjutnya. FC-layer biasanya dilakukan setelah proses *convolution layer* dan *pooling layer*. Hasil dari proses *convolution layer* dan *pooling layer* berupa *feature map* yang dijadikan masukan pada FC-layer. Secara umum keluaran pada FC-layer dapat ditentukan dengan persamaan (7).

$$h(x) = g(b + \sum_i w_i x_i) \quad (7)$$

dengan g sebagai fungsi aktivasi, b sebagai bias, w_i sebagai nilai masukan/nilai fitur, dan x_i sebagai nilai bobot koneksi saraf.

Softmax berfungsi untuk memetakan angka yang didapat dari perhitungan *fully connected layer* ke dalam sebuah probabilitas yang menunjukkan kelasnya. Fungsi aktivasi *softmax* digunakan untuk

proses klasifikasi yang mempunyai berbagai kelas [13].

Tahap pelatihan model ResNet-50 menggunakan 5 jenis konvolusi. Semua citra mengalami normalisasi menjadi 224×224 piksel. Fungsi aktivasi yang digunakan pada *fully-connected layer* yaitu *flatten* yang mengubah *output multi dimension array* dari proses pelatihan menjadi *array* satu dimensi. Setelah citra melalui *fully connected layer*, hasilnya dijadikan input fungsi aktivasi *softmax* untuk untuk menghitung probabilitas dari data hasil pelatihan terhadap objek yang terdiri dari 5 kelas [14].

2.6. Fungsi Optimasi

Fungsi Optimasi merupakan fungsi yang digunakan untuk meningkatkan proses pembelajaran pada sistem. Setiap algoritma optimasi menggunakan nilai *learning rate* tertentu yang menentukan kemampuan sistem belajar secara cepat ataupun lambat. Penelitian ini menggunakan tiga jenis optimasi yaitu *Root Mean Square Propagation* (RMSprop), *Adaptive Moment Estimation* (Adam), dan *Stochastic Gradient Descent* (SGD).

RMSprop merupakan modifikasi AdaGrad yang bekerja lebih baik untuk pengaturan non-convex dengan mengubah akumulasi gradien menjadi *exponentially weighted moving average*. Nilai *learning rate* standar pada SGD adalah 0,001. Berikut perhitungan pembaruan RMSprop pada persamaan (8), (9), dan (10).

$$r = \rho r + (1-\rho) g \odot g \tag{8}$$

$$\Delta\theta = -\frac{\alpha}{\delta+\sqrt{r}} \odot g \tag{9}$$

$$\theta = \theta + \Delta\theta \tag{10}$$

dengan r adalah *accumulate squared gradient*, ρ adalah *decay rate*, $\Delta\theta$ adalah *compute update*, α adalah *learning rate*, δ adalah konstanta bernilai 10^{-7}, dan θ adalah parameter inisial.

Adam merupakan kombinasi dari RMSprop dan momentum. Adam adalah hasil penurunan metode SGD yang didasarkan pada estimasi adaptif momen orde pertama dan kedua. algoritma optimasi yang menghitung tingkat pembelajaran secara adaptif untuk setiap parameter. Adam menyimpan rata-rata gradien proses sebelumnya secara eksponensial sama seperti RMSprop. Nilai *learning rate* standar pada Adam adalah 0,001 [15]. Rumus perhitungan optimasi Adam ditunjukkan pada persamaan (11).

$$\theta_{t+1} = \theta_t - \frac{\partial}{\sqrt{\hat{v}_t+\varepsilon}} . \hat{m}_t \tag{11}$$

dengan θ_{t+1} adalah parameter hasil pembaruan, θ_t adalah parameter hasil pembaruan sebelumnya, η adalah *learning rate*, $\hat{m}_t$ adalah gradien kuadrat momen orde pertama, $\hat{v}_t$ gradien kuadrat momen orde kedua, dan ε merupakan scalar kecil untuk mencegah pembagian dengan nol. Persamaan (11) menunjukkan perhitungan optimasi Adam dalam memperbarui nilai *error* dalam proses pelatihan dengan memanfaatkan nilai gradien pada momen orde pertama dan orde kedua.

SGD merupakan salah satu variasi dari optimasi *gradient decent* yang selalu melakukan pembaruan parameter untuk setiap data yang sedang dilatih. Saat melakukan pembaruan parameter, SGD tidak melakukan perulangan sehingga kinerjanya lebih cepat untuk dataset berjumlah besar. Nilai *learning rate* standar pada SGD adalah 0,01. Proses pembaruan parameter pada SGD dapat didefinisikan pada persamaan (12).

$$\theta = \theta - \eta * \nabla_\theta J\left(\theta; x^{(i)}; y^{(i)}\right) \tag{12}$$

dengan θ adalah parameter hasil pembaruan, η adalah *learning rate*, $x^{(i)}$ dan $y^{(i)}$ merupakan data yang sedang dilakukan pelatihan.

2.7. Pengujian Sistem

Untuk mendapatkan model sistem terbaik diperlukan parameter uji sebagai nilai pembanding untuk setiap model. Parameter performa yang digunakan dalam penelitian ini yaitu akurasi dan *loss*.

1. Akurasi

Akurasi merupakan salah satu parameter uji yang menentukan kelayakan dari sebuah sistem dalam mengklasifikasikan pola sidik jari. Akurasi (A) dapat dihitung dengan persamaan (13).

$$A = \frac{b}{n} \times 100\% \tag{13}$$

dengan A adalah persentase tingkat kebenaran, b adalah jumlah data yang diklasifikasi dengan benar, dan n adalah jumlah keseluruhan data.

2. *Loss*

Loss merupakan parameter uji yang digunakan untuk menghitung data yang tidak terdeteksi (*loss*) saat proses pelatihan maupun pengujian. Pada penelitian ini menggunakan jenis *sparse categorical cross entropy loss*. Secara matematis perhitungan *loss* dapat dituliskan dengan persamaan (14).

$$L(\Theta) = -\sum_{i=1}^{k} y_i \log\left(\hat{y}_i\right) \tag{14}$$

dengan $L(\Theta)$ sebagai *loss*, y_i sebagai jumlah data terdeteksi dengan benar, dan $\hat{y}_i$ sebagai jumlah data terdeteksi.

3. HASIL DAN PEMBAHASAN

Spesifikasi perangkat yang digunakan pada pengujian sistem ini, yaitu bahasa pemrograman

Python 3.7.7, s*oftware* Anaconda Navigator, dan h*ardware*, dengan spesifikasi: processor Intel® Core™ i5-10210u CPU @ 1.60GHz (8 CPUs), ~2.1GHz, memori 4096MB RAM, GPU Intel® UHD Graphics dan Radeon 530 Series

3.1. Pengujian *Preprocessing*

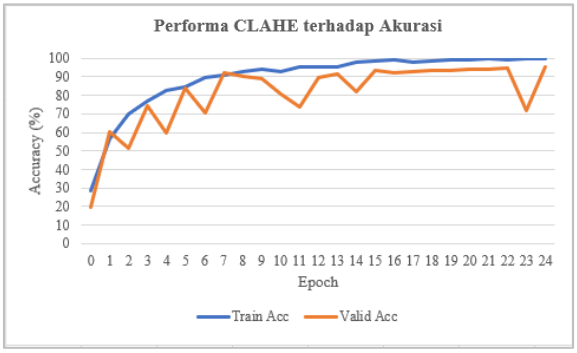
Bagian ini merupakan pengujian tentang pengaruh adanya *preprocessing* pada citra sebelum proses klasifikasi. *Preprocessing* yang digunakan yaitu *canny edge detection* dan CLAHE. Pengujian ini membandingkan hasil citra original dengan citra hasil *preprocessing*. Optimasi yang digunakan SGD dengan *default learning rate* 0,01, dan *epoch* 25. *Epoch* merupakan satu siklus proses pembelajaran dari seluruh dataset training. Proses pembelajaran yang berulang-ulang bertujuan untuk mencapai konvergensi nilai bobot [16]. Tabel 1 merupakan hasil pengujian *preprocessing* terhadap performa akurasi dan *loss*.

Tabel 1. Pengaruh *Preprocessing* terhadap Performa

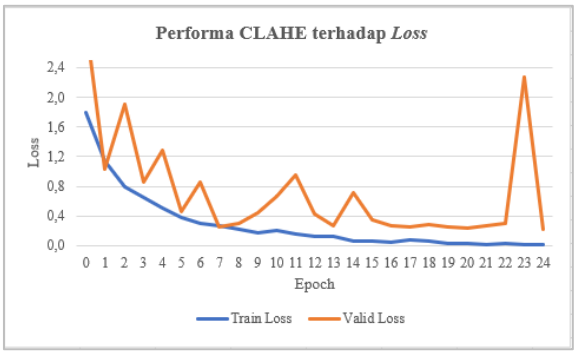
Citra	Akurasi		Loss	
	training (%)	validasi (%)	training	validasi
Original	93,56	83,26	0,191	0,631
Canny	95,31	77,83	0,155	0,762
CLAHE	99,52	95,05	0,016	0,229

Berdasarkan Tabel 1, hasil terbaik untuk pengujian *preprocessing* adalah CLAHE yang memiliki akurasi pelatihan 99,52%, akurasi validasi 95,05%, *loss* pelatihan 0,016, dan *loss* validasi 0,229. Untuk *canny edge detection* menghasilkan akurasi pelatihan 95,31%, akurasi validasi 77,83%, *loss* pelatihan 0,155 dan *loss* validasi 0,762. Sedangkan untuk citra asli menghasilkan akurasi pelatihan 93,56%, akurasi validasi 83,26%, *loss* pelatihan 0,191, dan *loss* validasi 0,631. Adanya *preprocessing* pada citra sebelum proses klasifikasi dapat jauh lebih mengoptimalkan hasil akurasi dan meminimalkan *loss*. Berdasarkan hasil pengujian *preprocessing*, CLAHE dinilai sebagai *preprocessing* paling baik untuk meningkatkan kualitas citra pada pola sidik jari. Hasil pengujian CLAHE terhadap performa akurasi dan *loss* dapat dilihat pada Gambar 7 dan Gambar 8.

Hasil pada preprocessing CLAHE dipengaruhi oleh nilai *clip limit*. Semakin besar *clip limit* maka peningkatan piksel pada citra juga semakin besar. Jika *clip limit* besar, maka *excess pixel* lebih sedikit, sehingga tidak banyak histogram yang didistribusikan ke area bawah *clip limit*. Citra hasil CLAHE memiliki kontras yang lebih baik dibandingkan dengan canny, sehingga garis pola sidik jari lebih terlihat jelas. Canny melakukan pendekatan *threshold* untuk menghilangkan *noise* tidak ada proses penajaman citra didalamnya, jadi dianggap kurang untuk memperjelas garis-garis pola sidik jari. Adanya *preprocessing* di tahap awal juga dapat meringankan beban perangkat, dimana penggunaan *preprocessing* dapat menambah tingkat akurasi cukup tinggi meskipun sistem klasifikasi dengan arsitektur *layer* sedikit.



Gambar 7. Performa CLAHE terhadap akurasi



Gambar 8. Performa CLAHE terhadap *loss*

3.2. Pengujian Fungsi Optimasi

Bagian ini merupakan pengujian terhadap pengaruh fungsi optimasi. Optimasi yang digunakan yaitu SGD, RMSprop, dan Adam. Semua optimasi menggunakan *learning rate* 0,01 untuk mengetahui kinerja setiap optimasi pada *learning rate* yang sama. Dalam pengujian ini digunakan citra CLAHE yang memiliki hasil terbaik pada skenario sebelumnya. Tabel 2 merupakan hasil pengujian fungsi optimasi.

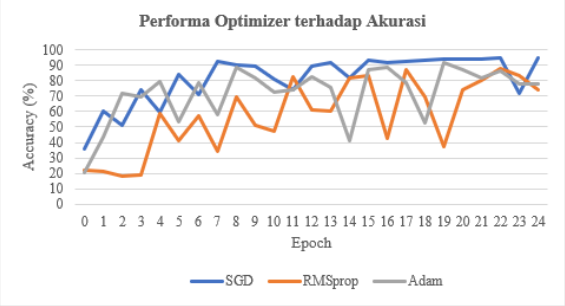
Tabel 2. Hasil Pengujian Fungsi Optimasi

Optimasi	Akurasi		Loss	
	training (%)	validasi (%)	training	validasi
SGD	99,52	95,05	0,016	0,229
RMSprop	96,98	74,29	0,108	1,716
Adam	96,42	77,83	0,117	1,375

Berdasarkan Tabel 2, optimasi SGD memiliki akurasi pelatihan 99,52%, akurasi validasi 95,05%, *loss* pelatihan 0,016, dan *loss* validasi 0,229. Untuk RMSprop menghasilkan akurasi pelatihan 96,98%, akurasi validasi 74,29%, *loss* pelatihan 0,108, dan *loss* validasi 0,716. Sedangkan untuk Adam menghasilkan akurasi pelatihan 96,42%, akurasi validasi 77,83%, *loss* pelatihan 0,117, dan *loss* validasi 1,375.

Performa Adam dan RMSprop tidak jauh berbeda, karena Adam merupakan kombinasi antara RMSprop dengan momentum. Adam dan RMSprop

melakukan penyimpanan rata-rata gradien proses sebelumnya secara eksponensial sehingga kinerjanya cenderung lebih lambat dibanding SGD. SGD tidak melakukan perulangan sehingga kinerjanya lebih cepat, terutama untuk jumlah data yang besar. Pada penelitian ini nilai *learning rate* yang digunakan sebesar 0,01. Nilai ini merupakan nilai standar pada SGD, namun tidak untuk RMSprop dan Adam memiliki standar *learning rate* sebesar 0,001. Gambar 9 merupakan performa akurasi untuk optimasi SGD, RMSprop dan Adam.



Gambar 9. Performa akurasi optimasi SGD, RMSprop, dan Adam

Berdasarkan hasil pengujian, optimasi SGD memiliki performa terbaik dibanding RMSprop dan Adam untuk klasifikasi pola sidik jari menggunakan model CNN dengan arsitektur ResNet-50.

4. KESIMPULAN

Penelitian ini mengusulkan sistem klasifikasi pola sidik jari otomatis menggunakan metode CNN dengan arsitektur ResNet-50. Sistem dapat mengidentifikasi 5 pola sidik jari dengan akurasi pelatihan 99,52%, akurasi validasi 95,05%, loss pelatihan 0,016, dan loss validasi 0,229. Hasil ini didapatkan dengan menggunakan *preprocessing* CLAHE, *learning rate* 0,01, dan optimasi SGD. Dari hasil pengujian, bahwa *preprocessing* CLAHE pada citra dan penggunaan fungsi optimasi SGD dapat meningkatkan performa sistem klasifikasi sidik jari.

DAFTAR PUSTAKA

[1] R. Clarke, "Roger Clarke ' s Human Id in Info. Systems Human Identification in Information Systems : Management Challenges and Public Policy Issues Introduction Roger Clarke ' s Human Id in Info . Systems Introduction Human Identity and Human Identification * Human ," pp. 1–20, 1994.

[2] N. A. Alias and N. H. M. Radzi, "Fingerprint classification using Support Vector Machine," *Proc. 2016 5th ICT Int. Student Proj. Conf. ICT-ISPC 2016*, pp. 105–108, 2016, doi: 10.1109/ICT-ISPC.2016.7519247.

[3] J. M. Shrein, "Fingerprint classification using convolutional neural networks and ridge orientation images," *2017 IEEE Symp. Ser. Comput. Intell. SSCI 2017 - Proc.*, vol. 2018-Janua, no. c, pp. 1–8, 2018, doi: 10.1109/SSCI.2017.8285375.

[4] X. Guo, F. Wu, and X. Tang, "Fingerprint pattern identification and classification," *ICNC-FSKD 2018 - 14th Int. Conf. Nat. Comput. Fuzzy Syst. Knowl. Discov.*, no. 1984, pp. 1045–1050, 2018, doi: 10.1109/FSKD.2018.8687199.

[5] P. Nahar, S. Tanwani, and N. S. Chaudhari, "Fingerprint Classification Using Resnet50," *Int. J. Res. Anal. Rev.*, vol. 5, no. 04, pp. 1521–1535, 2018.

[6] M. Ghafoor *et al.*, "Fingerprint Identification With Shallow Multifeature View Classifier," *IEEE Trans. Cybern.*, vol. PP, pp. 1–13, 2019, doi: 10.1109/TCYB.2019.2957188.

[7] W. S. Jeon and S. Y. Rhee, "Fingerprint pattern classification using convolution neural network," *Int. J. Fuzzy Log. Intell. Syst.*, vol. 17, no. 3, pp. 170–176, 2017, doi: 10.5391/IJFIS.2017.17.3.170.

[8] NIST Special Database 4, "NIST 8-Bit Gray Scale Images of Fingerprint Image Groups (FIGS),"2019. https://www.nist.gov/srd/nist-special-database-4 (accessed Maret. 11, 2020).

[9] S. R. Borra, G. J. Reddy, and E. S. Reddy, "A broad survey on fingerprint recognition systems," *Proc. 2016 IEEE Int. Conf. Wirel. Commun. Signal Process. Networking, WiSPNET 2016*, pp. 1428–1434, 2016, doi: 10.1109/WiSPNET.2016.7566372.

[10] P. N. Andono, *Pengolahan Citra Digital*. Penerbit Andi, Yogyakarta, 2017.

[11] S. A. Syakry, M. Syahronir, and M. Mulyadi, "Perbandingan Performa Segmentasi Citra Sidik Jari Menggunakan Deteksi Tepi Metode Sobel Dengan Metode Canny," *J. Infomedia*, vol. 1, no. 2, pp. 35–40, 2016, doi: 10.30811/.v1i2.332.

[12] G. F. C. Campos, S. M. Mastelini, G. J. Aguiar, R. G. Mantovani, L. F. de Melo, and S. Barbon, "Machine learning hyperparameter selection for Contrast Limited Adaptive Histogram Equalization," *Eurasip J. Image Video Process.*, vol. 2019, no. 1, 2019, doi: 10.1186/s13640-019-0445-4.

[13] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, 2018, vol. 2018-Janua, pp. 1–6, doi: 10.1109/ICEngTechnol.2017.8308186.

[14] X. Ou *et al.*, "Moving Object Detection Method via ResNet-18 with Encoder-Decoder Structure in Complex Scenes," *IEEE Access*, vol. 7, pp. 108152–108160,

2019, doi: 10.1109/ACCESS.2019.2931922.

[15] L. A. Andika, H. Pratiwi, and S. S. Handajani, "Klasifikasi Penyakit Pneumonia Menggunakan Metode Convolutional Neural Network Dengan Optimasi Adaptive Momentum," *Indones. J. Stat. Its Appl.*, vol. 3, no. 3, pp. 331–340, 2019, doi: 10.29244/ijsa.v3i3.560.

[16] M. E. Abdulfattah, L. Novamizanti, S. Rizal, "Super Resolution pada Citra Udara Menggunakan Convolutional Neural Network," *ELKOMIKA*, Vol. 8, No. 3, 2020.