Algoritma : Jurnal Matematika, Ilmu pengetahuan Alam, Kebumian dan Angkasa
Vol. 2, No. 3 Mei 2024
e-ISSN: 3046-5427; p-ISSN: 3032-6230, Hal 108-120
DOI : https://doi.org/10.62383/algoritma.v2i3.69

Penerapan Algoritma CNN Untuk Mendeteksi Tulisan Tangan Angka Romawi dengan Augmentasi Data

Mochammad Toyib
Teknik Informatika, Universitas Nurul Jadid
Email: mochtoyib118@gmail.com

Tegar Decky Kurniawan Pratama
Teknik Informatika, Universitas Nurul Jadid
Email: tegardcky11@gmail.com

Ibnu Aqil
Teknik Informatika, Universitas Nurul Jadid
Email: ibnu22097@gmail.com

Abstract. *This research aims to develop and apply a Convolutional Neural Network (CNN) algorithm to detect handwritten Roman numerals. Handwriting recognition is a classic challenge in the fields of image processing and machine learning, especially for less common characters such as Roman numerals. In this research, we use data augmentation techniques to increase the diversity and number of datasets used in model training, which is expected to increase model accuracy and generalization. The dataset used consists of 1,120 images for testing and 280 images for validation, each of which is divided into 14 classes of Roman numerals I, II, III, IV, V, VI, VII, VIII, IX, X, L, C, D , and M. Image data was created directly using simple software, namely Paint version 6.3. This research uses the Python programming language and Google Colab as a computing platform. Model training was carried out for 300 epochs and showed significant accuracy in the 150th to 300th iteration. The results at the 300th epoch show an accuracy of 0.9607 and a loss of 0.1162. The implementation of this algorithm shows significant potential in practical applications, such as in the fields of education and historical documentation. The conclusion of this research is that data augmentation is an effective technique to improve the performance of CNN models in detecting handwritten Roman numerals.*

Keywords: *Convolutional Neural Network, Handwriting Detection, Roman Numerals, Data Augmentation, Image Processing, Model Accuracy.*

Abstrak. Penelitian ini bertujuan untuk mengembangkan dan menerapkan algoritma Convolutional Neural Network (CNN) untuk mendeteksi tulisan tangan angka Romawi. Pengenalan tulisan tangan merupakan tantangan klasik dalam bidang pengolahan citra dan pembelajaran mesin, terutama untuk karakter yang kurang umum seperti angka Romawi. Dalam penelitian ini, kami menggunakan teknik augmentasi data untuk meningkatkan keragaman dan jumlah dataset yang digunakan dalam pelatihan model, sehingga diharapkan dapat meningkatkan akurasi dan generalisasi model. Dataset yang digunakan terdiri dari 1.120 gambar untuk pengujian dan 280 gambar untuk validasi, yang masing-masing terbagi dalam 14 kelas angka Romawi I, II, III, IV, V, VI, VII, VIII, IX, X, L, C, D, dan M. Data gambar dibuat langsung menggunakan software sederhana yaitu Paint versi 6.3. Penelitian ini menggunakan bahasa pemrograman Python dan Google Colab sebagai platform komputasi. Pelatihan model dilakukan sebanyak 300 epoch dan menunjukkan akurasi signifikan pada iterasi ke-150 sampai ke-300. Hasil pada epoch ke-300 menunjukkan akurasi sebesar 0.9607 dan loss sebesar 0.1162. Implementasi algoritma ini menunjukkan potensi yang signifikan dalam aplikasi praktis, seperti dalam bidang pendidikan dan dokumentasi sejarah. Kesimpulan dari penelitian ini adalah bahwa augmentasi data merupakan teknik yang efektif untuk meningkatkan performa model CNN dalam mendeteksi tulisan tangan angka Romawi.

Kata Kunci : *Convolutional Neural Network,* Deteksi Tulisan Tangan, Angka Romawi, Augmentasi Data, Pengolahan Citra, Akurasi Model.

Received April 10, 2024; Accepted Mei 27, 2024; Published Mei 31, 2024
* Mochammad Toyib , *mochtoyib118@gmail.com*

PENDAHULUAN

Angka Romawi adalah sistem penomoran kuno yang berasal dari Roma dan digunakan secara luas di seluruh Kekaisaran Romawi. Sistem ini menggunakan kombinasi huruf dari alfabet Latin seperti I, V, X, L, C, D, dan M yang masing-masing mewakili nilai numerik 1, 5, 10, 50, 100, 500, dan 1000. Meskipun tidak lagi digunakan untuk perhitungan sehari-hari, angka Romawi tetap penting dalam beberapa konteks, seperti penomoran halaman dalam pendahuluan buku, tahun dalam film, dan penomoran bab atau acara dalam dokumen resmi. Sejarah penggunaan angka Romawi dapat dilacak dari berbagai dokumen dan monumen di Roma Kuno, dan penggunaannya terus berlanjut dalam berbagai aspek kehidupan modern.

Namun, salah satu tantangan utama dalam mengenali angka Romawi, terutama dalam bentuk tulisan tangan, adalah kesulitan dalam interpretasi yang akurat, terutama dalam bentuk digital. Masalah ini sering diperparah oleh variasi dalam gaya tulisan tangan, ketidakseragaman bentuk huruf, dan noise pada gambar atau dokumen yang dipindai. Ketidakmampuan untuk mendeteksi dan mengenali angka Romawi secara otomatis dapat menyebabkan ketidakakuratan dalam aplikasi seperti digitalisasi dokumen sejarah, pengenalan karakter optik (OCR), dan pendidikan. Oleh karena itu, diperlukan solusi yang efektif untuk meningkatkan akurasi pengenalan tulisan tangan angka Romawi.

Salah satu solusi yang menjanjikan adalah penggunaan Convolutional Neural Network (CNN), sebuah algoritma deep learning yang dirancang untuk pemrosesan data gambar. CNN telah terbukti sangat efektif dalam tugas-tugas seperti klasifikasi gambar, deteksi objek, dan pengenalan pola. Dengan melatih model pada dataset yang besar dan beragam, CNN dapat mengotomatisasi proses pengenalan angka Romawi dengan akurasi tinggi. Teknik augmentasi data juga dapat digunakan untuk meningkatkan kinerja deteksi dengan memperkenalkan variasi tambahan dalam dataset pelatihan melalui transformasi seperti rotasi, penskalaan, dan penambahan noise.

Penelitian sebelumnya telah menunjukkan efektivitas CNN dalam pengenalan karakter. Sebagai contoh, studi oleh Muhaafidz Md Saufi dkk. menggunakan dataset Chars74K dan membandingkan model Simple CNN dengan AlexNet dan GoogleNet, dengan hasil bahwa GoogleNet memiliki akurasi tertinggi. Studi lain oleh Aathira Manoj dkk. memodifikasi model LeNet5 dan menggunakan dataset MNIST, berhasil mereduksi error rate menjadi 0,8%. Penelitian oleh El-sawy et al. pada huruf hijaiyah menggunakan metode CNN dan mendapatkan akurasi sebesar 94,9%. Berdasarkan fondasi ini, penelitian kami berfokus pada

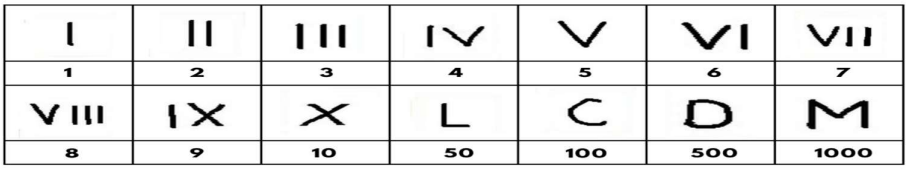
penggunaan augmentasi data untuk lebih meningkatkan performa model CNN dalam mendeteksi tulisan tangan angka Romawi.

METODOLOGI PENELITIAN

Metodologi yang diadopsi dalam penelitian ini dapat dilihat pada Gambar. 1. Tahap awal melibatkan persiapan dataset. Setelah itu, dilakukan praproses dan augmentasi data. Langkah berikutnya adalah merancang arsitektur CNN yang akan dijadikan model. Kemudian, pelatihan model dilakukan menggunakan dataset yang telah melalui tahap praproses dan augmentasi. Tahap akhir melibatkan evaluasi model yang telah dilatih.

Gambar 1. Metode Penelitian.

Dataset yang digunakan terdiri dari 1400 sampel tulisan tangan yang dibuat secara manual menggunakan software Paint versi 6.3. Beberapa contoh dataset dapat dilihat pada Gambar. 2. Saat ini, data tersebut masih bersifat pribadi dan belum dipublikasikan.



Gambar 2. Sampel *dataset.*

Data yang dihasilkan dari tulisan tangan yang dibuat secara manual menggunakan Paint awalnya dalam format PNG. Kami kemudian mengonversinya ke format JPG untuk mempermudah proses dan memungkinkan langsung dimasukkan ke dalam sistem pelatihan tanpa perlu konversi tambahan. Selain itu, kami menyesuaikan ukuran gambar agar sesuai dengan kebutuhan sistem pelatihan. Ukuran gambar yang awalnya bervariasi diubah menjadi 150x150 piksel dengan bantuan pustaka Python, yaitu pustaka cv2.

TABEL I. DESKRIPSI *DATASET*

NO	ANGKA	DATA LATIH	DATA UJI
1	I	80	20
2	II	80	20
3	III	80	20
4	IV	80	20
5	V	80	20
6	VI	80	20
7	VII	80	20
8	VIII	80	20
9	IX	80	20
10	X	80	20
11	L	80	20
12	C	80	20
13	D	80	20
14	M	80	20
TOTAL		1.120	280

Dataset terdiri dari 14 kelas angka Romawi, yaitu I, II, III, IV, V, VI, VII, VIII, IX, X, L, C, D, dan M, seperti yang ditampilkan pada Tabel I. Setiap kelas memiliki 100 sampel data. Dataset ini dibagi menjadi dua bagian: 80% untuk data pelatihan dan 20% untuk data pengujian, sesuai dengan prinsip Pareto. Dengan demikian, data pelatihan terdiri dari 1.120 sampel, sedangkan data pengujian terdiri dari 280 sampel.

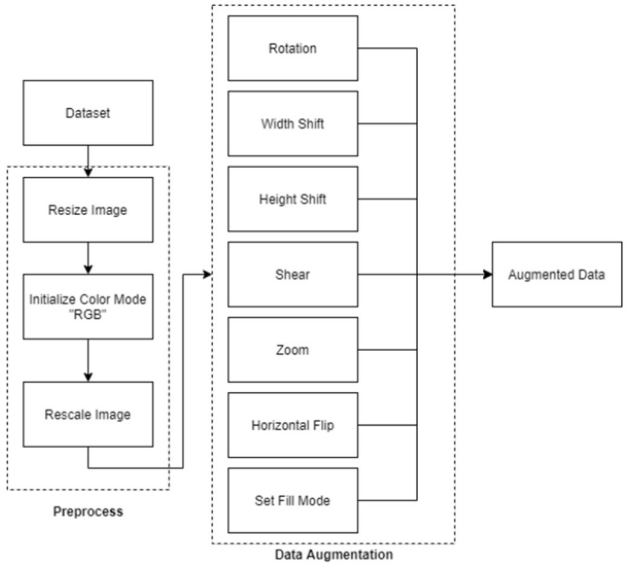
B. Preprocess and Augmentation

Praproses dan augmentasi data diilustrasikan pada Gambar. 3. Tahap praproses meliputi penyesuaian ukuran citra dari berbagai ukuran menjadi 150×150 piksel dan menetapkan mode warna Red, Green, and Blue (RGB) dengan 3 kanal. Meskipun gambar yang digunakan hanya terdiri dari warna hitam dan putih, mode warna tetap dipertahankan dalam format RGB. Dengan demikian, setiap citra memiliki dimensi *H x W x C*, yaitu 150×150×3 untuk height (tinggi), width (lebar), dan channel (kanal).

Selain itu, input RGB yang memiliki nilai koefisien antara 0-255 di-rescale menggunakan rumus berikut:

$$Output = input \times \frac{1}{255}$$

Rumus ini digunakan untuk mengubah rentang nilai dari 0-255 menjadi 0-1.

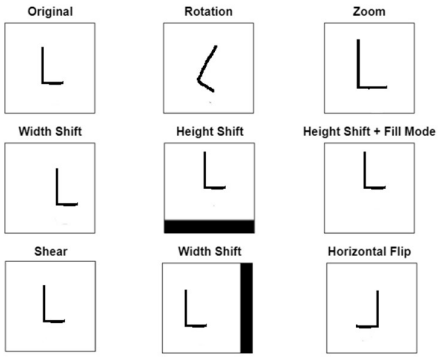


Gambar 3. Praproses dan augmentasi data.

Dengan 1.400 data, jumlah ini mungkin terlihat sedikit jika dibandingkan dengan dataset yang telah disebutkan sebelumnya. Namun, ada metode yang efektif untuk memperkaya variasi fitur, yaitu dengan menggunakan teknik augmentasi data. Berikut beberapa teknik augmentasi data yang diterapkan:

1. **Rotasi (*rotation*)** : Memutar gambar pada sudut tertentu untuk menciptakan variasi orientasi.
2. **Pergeseran lebar (*width shift)***: Menggeser gambar secara horizontal untuk mengubah posisi objek dalam gambar.
3. **Pergeseran tinggi (*height shift*)**: Menggeser gambar secara vertikal untuk memodifikasi posisi objek dalam gambar.
4. **Mencukur (*shear*)**: Mengubah bentuk gambar dengan menggeser sebagian gambar secara horizontal atau vertikal.
5. **Memperbesar (*zoom*)**: Memperbesar bagian tertentu dari gambar untuk menghasilkan variasi skala.
6. **Membalik secara horizontal (*horizontal flip*)**: Membalik gambar dari kiri ke kanan untuk menciptakan citra cermin dari gambar asli.

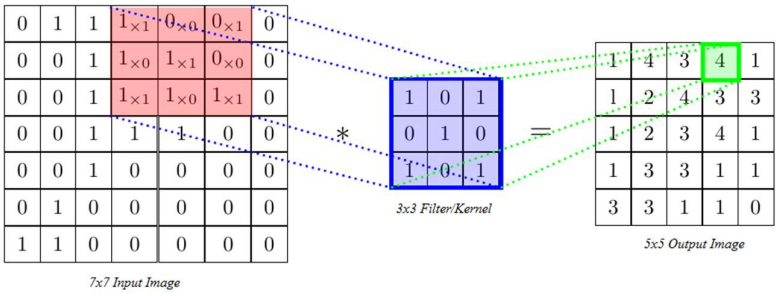
Untuk mengisi area kosong yang muncul akibat pergeseran lebar atau tinggi, dilakukan pengisian piksel baru. Hasil dari praproses dan augmentasi ini diilustrasikan pada Gambar. 4.



Gambar 4. Praproses dan Augmentasi Data.

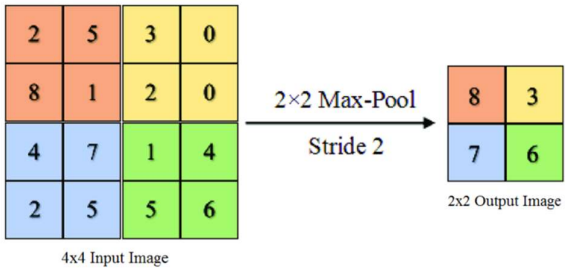
C. CNN Architecture

Convolutional Neural Network (CNN) adalah salah satu metode yang populer dalam deep learning. CNN dapat digunakan untuk pelatihan baik melalui supervised learning maupun unsupervised learning. Metode ini umumnya digunakan untuk klasifikasi gambar.



Gambar 5. *Convolution layer.*

Seperti namanya, convolution layer memainkan peran penting dalam CNN. Convolution layer bertugas mengekstrak fitur dari gambar menggunakan filter atau kernel tertentu yang melakukan forward propagation pada data latih. Parameter yang dapat dipelajari, seperti kernel dan bobot, diperbarui berdasarkan nilai loss melalui backpropagation menggunakan optimisasi gradient descent. Seperti yang ditunjukkan pada Gambar. 5, gambar input berukuran 7x7 mengalami operasi dot antara input dan filter. Setelah operasi pada bagian pertama selesai, filter digeser hingga seluruh bagian gambar dihitung, menghasilkan output berupa gambar berukuran 5x5.

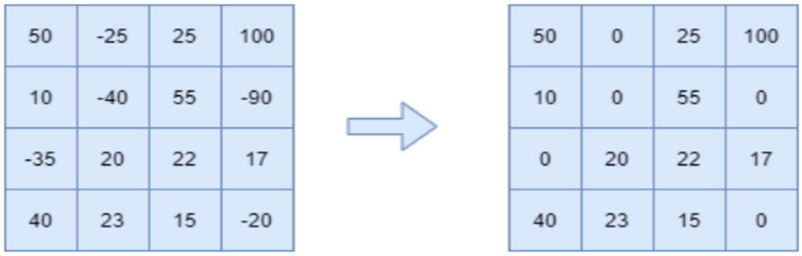


Gambar 6. *Pooling layer*

Gambar 6. menunjukkan ilustrasi dari pooling layer. Pooling layer digunakan untuk mengurangi jumlah parameter sambil mempertahankan informasi penting yang merepresentasikan gambar. Ada beberapa jenis pooling layer, yaitu MaxPooling, AveragePooling, dan SumPooling. Pada penelitian ini, tipe pooling yang digunakan adalah MaxPooling. Pooling layer dengan ukuran 2x2 akan digunakan untuk memeriksa gambar input dan mencari elemen terbesar di setiap area pooling. Rectified Linear Unit (ReLU) digunakan untuk operasi non-linear. Oleh karena itu, output dari operasi ReLU adalah nilai input itu sendiri atau 0. Rumus yang digunakan adalah sebagai berikut:

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

Rumus ini diilustrasikan pada Gambar. 7. Dengan demikian, setiap elemen dengan nilai negatif akan diubah menjadi 0.



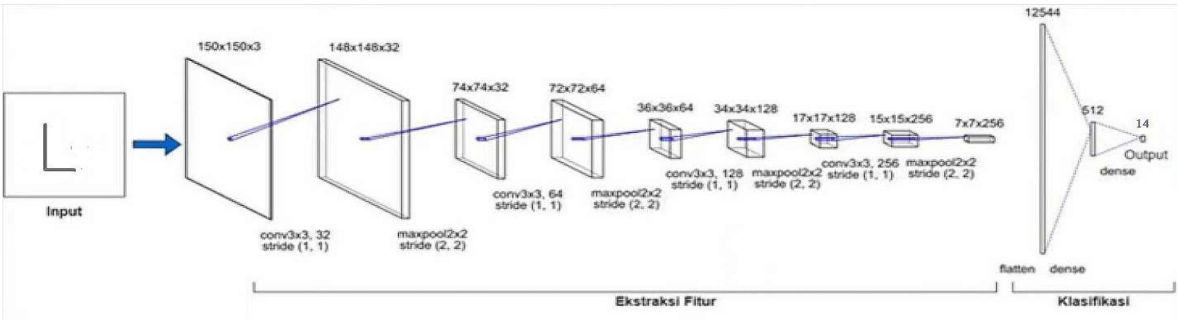
Gambar 7. Operasi ReLU.

Softmax adalah fungsi aktivasi yang digunakan dalam jaringan neural untuk menghitung distribusi sebuah vektor dari bilangan real. Fungsi ini menghasilkan output dengan nilai antara 0 dan 1, di mana total dari semua kemungkinan sama dengan 1. Softmax biasanya diterapkan pada model multi-kelas dengan tujuan mengidentifikasi kemungkinan tertinggi untuk setiap kelas.

Softmax digunakan pada output layer dari arsitektur deep learning, seperti yang digunakan oleh A. Krizhevsky et al. dan V. Badrinarayanan et al. Fungsi aktivasi softmax dihitung menggunakan rumus berikut:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Fungsi ini memastikan bahwa semua output berada dalam rentang 0 hingga 1 dan jumlah totalnya adalah 1, yang sangat berguna untuk model klasifikasi multi-kelas dalam menentukan kelas dengan probabilitas tertinggi.



Gambar 8. Arsitektur CNN.

Arsitektur Convolutional Neural Network (CNN) pada Gambar 8 tersebut menunjukkan sebuah model yang terdiri dari beberapa tahapan untuk ekstraksi fitur dan klasifikasi. Model dimulai dengan lapisan input yang menerima gambar berukuran 150x150x3, menunjukkan gambar berwarna dengan tiga saluran (RGB). Tahapan pertama adalah lapisan konvolusi dengan filter berukuran 3x3 dan 32 filter, yang menghasilkan output berukuran 148x148x32 setelah operasi konvolusi dengan stride (1,1). Selanjutnya, output ini dilewatkan melalui lapisan max-pooling berukuran 2x2 dengan stride (2,2), menghasilkan output berukuran 74x74x32. Proses ini diulangi dengan variasi filter dan ukuran pooling, menghasilkan serangkaian output dengan ukuran yang semakin kecil tetapi dengan kedalaman yang meningkat.

Lapisan konvolusi kedua menggunakan 64 filter 3x3, diikuti oleh max-pooling yang menghasilkan output 72x72x64 dan 36x36x64 secara berurutan. Lapisan ketiga dan keempat masing-masing menggunakan 128 dan 256 filter, dengan lapisan max-pooling yang

menurunkan ukuran output menjadi 34x34x128 dan 7x7x256. Setelah ekstraksi fitur, output dari lapisan terakhir diratakan menjadi vektor berukuran 12544, yang kemudian dihubungkan ke lapisan dense dengan 512 neuron. Lapisan terakhir adalah dense layer yang memiliki 14 neuron, sesuai dengan jumlah kelas untuk klasifikasi.

Rumus yang digunakan untuk menghitung ukuran output pada lapisan konvolusi dan max-pooling adalah

$$\text{Output Size} = \frac{\text{Input Size} - \text{Filter Size} + 2 \times \text{Padding}}{\text{Stride}} + 1$$

- **Input Size**: Ukuran input (misalnya, 150 untuk dimensi spasial pertama).
- **Filter Size**: Ukuran filter konvolusi (misalnya, 3 untuk filter 3×33×3).
- **Padding**: Jumlah padding yang ditambahkan di sekitar input (biasanya 0 untuk tidak ada padding).
- **Stride**: Langkah pergeseran filter (misalnya, 1 untuk stride (1,1)).

Untuk Ukuran output pooling dihitung dengan rumus yang sama dengan konvolusi, tanpa ada filter size:

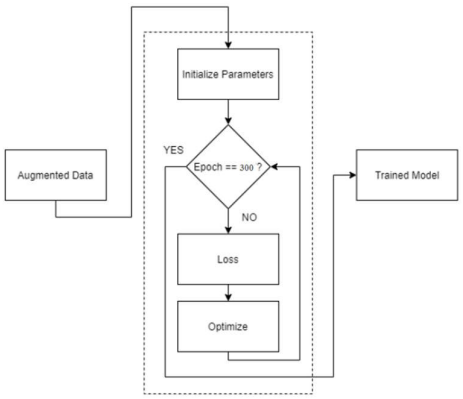
$$\text{Output Size} = \frac{\text{Input Size} - \text{Pooling Size}}{\text{Stride}} + 1$$

- **Pooling Size**: Ukuran pooling (misalnya, 2 untuk pooling 2×22×2).

dimana setiap parameter disesuaikan dengan spesifikasi layer. Sebagai contoh, lapisan konvolusi pertama dengan input 150x150, filter 3x3, stride (1,1), dan padding 0 menghasilkan output 148x148. Proses yang sama digunakan untuk menghitung output dari lapisan max-pooling dengan ukuran pooling yang sesuai. Kombinasi dari lapisan-lapisan ini menghasilkan model yang dapat mengekstraksi fitur penting dari gambar dan melakukan klasifikasi berdasarkan fitur-fitur tersebut.

D. Training

Proses pelatihan digambarkan pada Gambar 9, dimulai dengan inisialisasi berbagai parameter penting. Parameter-parameter ini meliputi jumlah epoch sebanyak 300, jumlah langkah per epoch yang disesuaikan dengan jumlah data uji dan pelatihan, dan ukuran batch sebesar 80. Optimizer yang digunakan adalah Adam, sementara fungsi loss yang diterapkan adalah *Sparse Categorical Cross Entropy* (SCCE). Metrik yang digunakan untuk evaluasi adalah akurasi. Selain itu, data yang digunakan untuk validasi dan kebutuhan lainnya disesuaikan oleh masing-masing peneliti.



Gambar 9. Pelatihan model.

Untuk menghitung error atau loss dalam pelatihan penelitian ini, digunakan fungsi loss Sparse Categorical Cross Entropy (SCCE). SCCE adalah fungsi loss yang dirancang untuk klasifikasi multi-kelas di mana label kelas dinyatakan sebagai integer. Fungsi ini sangat efektif untuk situasi di mana setiap sampel input termasuk dalam salah satu dari banyak kategori yang saling eksklusif, dengan label yang berupa integer daripada vektor one-hot. SCCE mengukur selisih antara distribusi probabilitas yang diprediksi oleh model dan distribusi sebenarnya dari label data. Fungsi loss ini sangat cocok digunakan dalam berbagai tugas klasifikasi multi-kelas seperti pengenalan gambar, pengenalan tulisan tangan, dan aplikasi lain yang menggunakan data label dalam bentuk integer.

Rumus SCCE dapat dituliskan sebagai berikut:

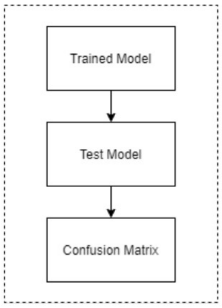
$$\text{SCCE}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\log(\hat{y}_{i,y_i})$$

Untuk optimasi, penelitian ini menggunakan algoritma Adam. Adam adalah algoritma optimasi yang menggabungkan keunggulan dari RMSprop dan Stochastic Gradient Descent (SGD) dengan momentum. Algoritma ini populer karena kemudahan implementasinya dan efisiensi komputasinya, yang tidak memerlukan banyak memori. Selain itu, Adam sangat cocok untuk digunakan pada data atau parameter yang besar, membuatnya ideal untuk berbagai aplikasi dalam machine learning.

E. Evaluate

Proses evaluasi yang ditampilkan pada Gambar 10 menggunakan confusion matrix untuk menganalisis distribusi hasil prediksi dibandingkan dengan label sebenarnya dari 14 kelas angka Romawi, yaitu I, II, III, IV, V, VI, VII, VIII, IX, X, L, C, D, dan M. Confusion

matrix ini membantu dalam mengidentifikasi kinerja model dalam mengklasifikasikan setiap kelas dengan benar serta kesalahan yang terjadi.



Gambar 10. Evaluasi model.

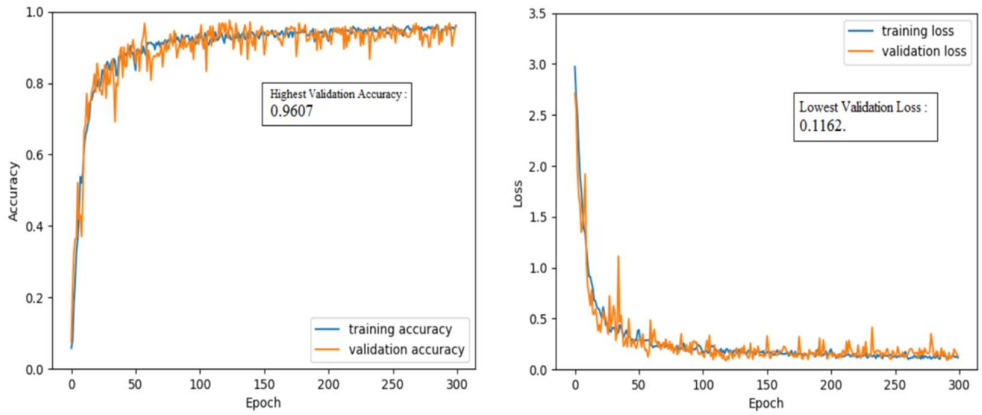
Confusion matrix digunakan untuk mengevaluasi performa model dengan menghitung beberapa metrik seperti accuracy, precision, recall, dan f1 score. Ada beberapa kategori kasus yang bisa terjadi dalam confusion matrix:

- True Positive (TP): ketika model memprediksi kelas positif dengan benar.
- True Negative (TN): ketika model memprediksi kelas negatif dengan benar.
- False Positive (FP): ketika model memprediksi kelas positif padahal seharusnya negatif.
- False Negative (FN): ketika model memprediksi kelas negatif padahal seharusnya positif.

Metrik-metrik ini membantu dalam memahami seberapa baik model bekerja dalam berbagai situasi dan memungkinkan penyesuaian yang lebih tepat untuk meningkatkan akurasinya.

HASIL DAN PEMBAHASAN

Lingkungan percobaan yang digunakan dalam penelitian ini melibatkan platform Google Colab dengan bahasa pemrograman Python serta framework deep learning TensorFlow.Keras. Platform Google Colab dipilih karena menyediakan sumber daya komputasi yang cukup untuk melatih model deep learning secara efisien dan tanpa biaya tambahan. TensorFlow.Keras digunakan sebagai framework karena kemudahannya dalam mengembangkan dan menguji berbagai arsitektur neural network. Beberapa hasil yang diperoleh dari penelitian ini diantaranya :



Gambar 11. Metrik akurasi dan loss

Gambar 11menampilkan metrik selama pelatihan model menggunakan arsitektur CNN. Hasilnya sangat menjanjikan, ditunjukkan oleh grafik yang tidak menunjukkan adanya overfitting. Pada epoch 0 hingga 50, akurasi model untuk training dan validasi meningkat dengan cepat dari sekitar 0% hingga mencapai 80%, menunjukkan bahwa model dengan cepat mempelajari pola dasar dalam data. Selanjutnya, pada epoch 50 hingga 150, terjadi beberapa fluktuasi, terutama dalam akurasi validasi, yang mengindikasikan bahwa model mungkin sedang menyesuaikan parameter dengan data yang bervariasi. Akurasi training dan validasi tetap cukup berdekatan, yang menunjukkan bahwa overfitting belum terjadi secara signifikan. Akhirnya, pada epoch 150 hingga 300, akurasi model stabil di sekitar 95% baik untuk training maupun validasi. Fluktuasi kecil yang terlihat menunjukkan bahwa model telah mencapai stabilitas dalam kinerjanya.

Secara keseluruhan, kedua grafik tersebut menunjukkan bahwa model telah dilatih dengan baik, mencapai akurasi yang tinggi pada set training dan validasi. Fluktuasi kecil pada akurasi validasi mencerminkan adaptasi model terhadap variasi dalam data validasi, sedangkan stabilitas pada akhir pelatihan menunjukkan bahwa model telah berhasil mempelajari pola secara efektif tanpa overfitting yang berlebihan. Akurasi validasi tertinggi yang mencapai 96.07% menunjukkan kinerja yang sangat baik dari model tersebut.

TABEL II. *CONFUSION MATRIX*

NO	ANGKA	PRECISION	RECALL	F1 SCORE	NO	ANGKA	PRECISION	RECALL	F1 SCORE
1	I	1.00	1.00	1.00	8	VIII	1.00	1.00	1.00
2	II	1.00	1.00	1.00	9	IX	1.00	1.00	1.00
3	III	1.00	1.00	1.00	10	X	1.00	1.00	1.00
4	IV	0.00	0.00	0.0	11	L	1.00	1.00	1.00
5	V	1.00	1.00	1.00	12	C	1.00	1.00	1.00
6	VI	1.00	1.00	1.00	13	D	1.00	1.00	1.00
7	VII	1.00	1.00	1.00	14	M	1.00	1.00	1.00
AKURASI PELATIHAN		0.9607							
AKURASI PENGUJIAN		0.9286							

Hasil evaluasi menggunakan *confusion matrix* dapat dilihat pada Tabel II. Secara keseluruhan, akurasi pelatihan model ini adalah 0.9607 atau 96.07%, sementara akurasi pengujian adalah 0.9286 atau 92.86%. Perbedaan ini menunjukkan bahwa model sedikit mengalami penurunan performa ketika diaplikasikan pada data yang belum pernah dilihat sebelumnya (data pengujian), yang merupakan hal yang wajar dan mengindikasikan tingkat overfitting yang tidak terlalu signifikan. Namun, performa yang buruk pada kelas IV menandakan adanya masalah spesifik dalam pengenalan kelas tersebut. Hal ini bisa disebabkan oleh berbagai faktor seperti jumlah data pelatihan yang tidak cukup untuk kelas IV, fitur yang tidak informatif untuk kelas tersebut, atau bahkan kesalahan dalam labeling data. Evaluasi lebih lanjut diperlukan untuk memahami akar penyebab dan memperbaiki performa model pada kelas IV tersebut.

KESIMPULAN DAN SARAN

Teknik augmentasi data sangat efektif dalam meningkatkan performa model Convolutional Neural Network (CNN) untuk mendeteksi tulisan tangan angka Romawi. Berbagai transformasi seperti rotasi, penskalaan, dan penambahan noise diterapkan untuk menghasilkan variasi data pelatihan yang lebih banyak, membantu model mengenali pola-pola tulisan tangan dengan lebih baik. Setelah pelatihan selama 300 epoch, model mencapai akurasi 0.9607 dengan loss 0.1162, menunjukkan kemampuan deteksi yang tinggi pada data baru. Implementasi algoritma CNN ini memiliki potensi signifikan dalam berbagai aplikasi praktis seperti pendidikan, digitalisasi dan interpretasi dokumen sejarah, pembacaan otomatis jam dinding klasik, dan penomoran urutan acara. Penelitian ini membuka jalan bagi inovasi lebih lanjut dalam teknologi OCR (Optical Character Recognition) dan pengenalan tulisan tangan, memberikan solusi yang menjanjikan untuk berbagai kebutuhan deteksi tulisan tangan angka Romawi.

DAFTAR PUSTAKA

M. M. Saufi, M. A. Zamanhuri, N. Mohammad dan Z. Ibrahim, "Deep Learning for Roman Handwritten Character Recognition," Indonesian Journal of Electrical Engineering and Computer Science, pp. 455-460, 2018.

T. M. Iqbal, "Huruf Hijaiyah: 30 Huruf Arab yang Luar Biasa [PENJELASAN LENGKAP]," 2020. [Online], https://hasana.id/hurufhijaiyah/, tanggal akses: 04 Oktober 2020.

A. El-sawy, M. Loey, and H. El-Bakry, "Arabic Handwritten Characters Recognition using Convolutional Neural Network," WSEAS Trans. Comput. Res., 2017.

F. A. Hadi, T. A. Budi dan K. N. Ramadhani, "Pengenalan Angka Tulisan Tangan Dengan Penerapan Freeman Chain Code yang Dimodifikasi," eProceeding of Engineering, pp. 6101, 2015.

C. K. Dewa, A. L. Fadhilah and Afiahayati, "Convolutional Neural Networks for Handwritten Javanese Character Recognition," IJCCS (Indonesian Journal of Computing and Cybernetics Systems), Vol. 12, No. 1, p. 83~94, 2018.

A. R. Widiarti and K. Pinaryanto, "Segmentasi Citra Huruf Daun Lontar," LPPM, Yogyakarta, 2019.

P. P. Nair, A. James dan C. Saravanan, "Malayalam Handwritten Character Recognition Using Convolutional Neural Network," International Conference on Inventive Communication and Computational Technologies, pp. 278-281, 2017

K. Wu, E. Otoo dan K. Suzuki, "Optimizing two-pass connectedcomponent labeling algorithms," Springer, pp. 117-135, 2009.

F. F. Maulana dan N. Rochmawati, "Klasifikasi Citra Buah Menggunakan Convolutional Neural Network," JINACS, pp. 104-108, 2019.

A. R. Widiarti and C. K. Adi, "Clustering Balinese Script Image in Palm Leaf Using Hierarchical K-Means Algorithm," in International Conference on Innovation in Science and Technology (ICIST 2020), Semarang, 2020.

I. S. Hanindria and Hendry, "Pengklasifikasian Aksara Jawa Metode Convolutional Neural Network," JATISI (Jurnal Teknik Informatika dan Sistem Informasi), Vol. 9, No. 3, pp. 2727 - 2736, 2022.