*Jurnal Informatika: Jurnal pengembangan IT*

*Vol. 10, No. 4, 2025*
*ISSN : 2477-5126, DOI:10.30591/jpit.v10i4.8793*

# Studi Perbandingan Metode *CNN-MLP* dan *CNN-SVM* untuk Pengenalan Pola Aksara Ulu Banyuasin

*Hardiman [1], Yesi Novaria Kunang [2]*
*[1,2]Universitas Bina Darma, Jl. Jend. Ahmad Yani. No.3, Palembang, 30111, Indonesia*

**Info Artikel**

***Riwayat Artikel:***

Received 2025-05-20
Revised 2025-09-17
Accepted 2025-09-18

***Corresponding Author:***

Hardiman

Email: hardimandmi@gmail.com



***Abstract*** *– Script pattern recognition is a major challenge in digital image processing due to the complex and diverse visual structures of each character. This study focuses on the recognition of Ulu Banyuasin script, a regional script that needs to be preserved through digitization. The main objective of this study is to compare the performance of two classification approaches, CNN-MLP and CNN-SVM, in recognizing these script patterns. The dataset consists of 16,800 color images at 224×224 resolution representing 168 script classes collected from digitized historical Ulu Banyuasin manuscripts. All data underwent preprocessing, including image resizing, pixel normalization, and simple data augmentation to improve training diversity. The VGG16 architecture was selected for feature extraction due to its ability to capture hierarchical visual representations from low-level to high-level features. After feature extraction, classification was performed using Multi-Layer Perceptron (MLP) and Support Vector Machine (SVM). Performance was evaluated using accuracy, precision, recall, and F1-score metrics. Experimental results showed that CNN-SVM achieved 99% accuracy, 98.7% precision, 98.5% recall, and 98.6% F1-score, while CNN-MLP achieved 93% accuracy, 92.4% precision, 91.8% recall, and 92.1% F1-score. CNN-SVM demonstrated superior performance in handling non-linear data in the high-dimensional feature space produced by CNN. However, signs of overfitting were observed in both approaches. Based on these initial results, the CNN-SVM approach has the potential to be applied and further explored for recognizing script patterns or characters from other writing systems.*

***Keywords:*** *CNN; Feature Extraction; Pattern Recognition; SVM; Ulu Banyuasin Script.*

***Abstrak*** *– Pengenalan pola aksara merupakan tantangan penting dalam pengolahan citra digital karena setiap karakter memiliki bentuk visual yang kompleks dan bervariasi. Penelitian ini berfokus pada pengenalan aksara Ulu Banyuasin, salah satu aksara daerah yang perlu dilestarikan melalui digitalisasi. Tujuan utama penelitian ini adalah membandingkan kinerja dua pendekatan klasifikasi, yaitu CNN-MLP dan CNN-SVM, dalam mengenali pola aksara tersebut. Dataset yang digunakan terdiri dari 16.800 citra berwarna berukuran 224×224 piksel yang mencakup 168 kelas aksara, dikumpulkan dari dokumentasi digital naskah kuno Ulu Banyuasin. Seluruh data telah melalui tahap praproses seperti konversi ukuran, normalisasi nilai piksel, dan augmentasi sederhana untuk meningkatkan keragaman data pelatihan. Arsitektur VGG16 dipilih sebagai metode ekstraksi fitur karena kemampuannya dalam menangkap representasi visual secara hierarkis dari fitur sederhana hingga kompleks. Setelah fitur diekstraksi, proses klasifikasi dilakukan menggunakan Multi-Layer Perceptron (MLP) dan Support Vector Machine (SVM). Evaluasi kinerja dilakukan menggunakan metrik akurasi, precision, recall, dan F1-score. Hasil eksperimen menunjukkan bahwa CNN-SVM memperoleh akurasi 99%, precision 98,7%, recall 98,5%, dan F1-score 98,6%, sementara CNN-MLP mencapai akurasi 93%, precision 92,4%, recall 91,8%, dan F1-score 92,1%. CNN-SVM menunjukkan kinerja yang lebih unggul dalam menangani data non-linear pada ruang fitur berdimensi tinggi yang dihasilkan CNN. Meskipun demikian, indikasi overfitting masih terdeteksi pada kedua pendekatan. Berdasarkan hasil awal ini, pendekatan CNN-SVM berpotensi diterapkan dan dieksplorasi lebih lanjut untuk pengenalan pola aksara atau karakter dari sistem tulisan lainnya.*

***Kata Kunci****: Aksara Ulu Banyuasin, CNN, Ekstraksi Fitur, Pengenalan Pola, SVM,*

## I. PENDAHULUAN

Pemanfaatan teknologi dapat diaplikasikan pada kebudayaan daerah, salah satunya melalui pengembangan media pembelajaran. Media pembelajaran modern dapat digunakan sebagai alternatif untuk melestarikan budaya, khususnya dalam mengenali pola aksara. Salah satu teknik yang dapat digunakan adalah teknik pengenalan pola citra [1]. Aksara merupakan salah satu dari unsur kebudayaan yang menjadi identitas suatu masyarakat karena berfunsgi sebagai media penyampaian informasi penting. Di dalamnya terkandung nilai-nilai kearifan lokal, hukum adat, kisah atau sejarah, pengobatan, atau ajaran agama dan lain-lain. Oleh karena itu harus dilestarikan agar masyarakat atau bangsa tidak kehilangan identitasnya di era modern saat ini [2].

Aksara ulu Banyuasin adalah surat Ulu yang digunakan untuk menuliskan bahasa-bahasa Ulu Banyuasin. Secara umum penulisannya sulit dipelajari, sehingga banyak generasi yang tidak menggunakan dan hampir

melupakan aksara Ulu Banyuasin. Tantangan utama dalam pengenalan pola Aksara Ulu Banyuasin terletak pada kompleksitas bentuk dan variasi penulisannya. Setiap huruf memiliki struktur yang rumit dengan banyak lengkungan, sudut, dan detail kecil yang membedakannya. Selain itu, gaya penulisan aksara dapat bervariasi antar naskah, tergantung pada penulis dan kondisi fisik naskah. Variasi ini menyebabkan proses pengenalan otomatis menjadi lebih sulit. Oleh karena itu pengetahuan dan teknologi dapat dimanfaatkan dalam pengenalan pola huruf aksara Ulu Banyuasin.

Untuk mengatasi tantangan teknis tersebut, pengetahuan dan teknologi pengolahan citra digital dapat dimanfaatkan. Salah satu teknik yang banyak dipakai adalah *Convolutional Neural Network* (*CNN*). *CNN* sangat populer dalam *machine learning* karena memiliki kinerja baik pada klasifikasi citra, anotasi gambar, dan berbagai bidang visi komputer lainnya [3]. Berbagai studi sebelumnya menunjukkan efektivitas *CNN* pada pengenalan aksara, misalnya penelitian tulisan tangan aksara Jawa yang dilakukan oleh Jonathan dkk (2023) pola aksara dengan metode *CNN* yang menghasilkan akurasi pengujian 84% dengan *error* sebesar 16% [4], serta penelitian oleh Kamal dkk (2022) dengan metode *CNN* menghasilkan akurasi yang baik sebesar 96% dan 99% untuk pengenalan karakter tulisan tangan Arab [5].

Pada penelitian yang dilakukan ini *CNN* akan dikombinasikan dengan *Support Vector Machine* (*SVM*). Kombinasi ini bertujuan untuk membandingkan akurasi yang dihasilkan antar metode *CNN* saja dengan *CNN-SVM*. *SVM* adalah algoritma *machine learning* yang efektif dalam tugas klasifikasi. Ayu Safitri dkk (2020) dalam penelitiannya menyebutkan *SVM* bekerja dengan memisahkan data ke dalam dua atau lebih kategori menggunakan *hyperplane* yang optimal [6]. Penelitian dalam mendeteksi wajah oleh Rita Widiarti dkk (2023) menghasilkan akurasi yang cukup baik yaitu sebesar 90%. *SVM* juga mendapatkan hasil akurasi yang baik [7] pada 1001 citra aksara Bali sebesar 82.32 %. Dan penelitian pengolahan citra huruf hijaiyah Amelia Putri dkk (2024) dengan *SVM* didapatkan hasil 99% dengan *input* 400 citra digital dengan rasio *train*:*test* adalah sebesar 8:2 [8].

Dari uraian di atas dan kajian penelitian terdahulu penulis tertarik mengambil topik penelitian ini untuk mempermudah proses pembelajaran aksara Ulu Banyuasin melalui sistem pengenalan pola. Penelitian ini menggunakan *CNN* sebagai pengekstrak ciri dan pengolah fitur, kemudian membandingkan kinerja klasifikasi antara *CNN* murni (dengan *MLP* pada tahap akhir) dan *CNN* dikombinasikan dengan *SVM*. Tujuan penelitian adalah mengidentifikasi pendekatan yang paling efektif untuk mengenali pola huruf aksara Ulu Banyuasin sehingga dapat meningkatkan aksesibilitas dan upaya pelestarian aksara daerah secara digital.

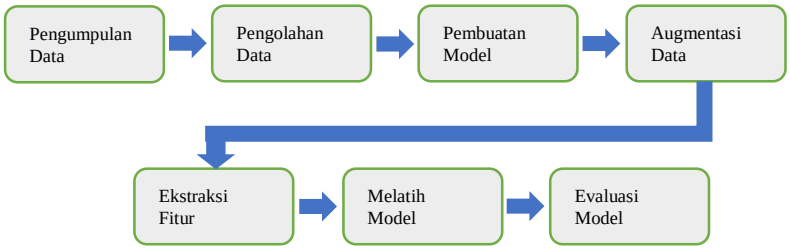
## II. METODE

Penelitian ini menggunakan pendekatan *machine learning*. Metode yang digunakan adalah *CNN* dan *SVM* untuk klasifikasi aksara Ulu Banyuasin. *Convolutional Neural Network* (*CNN*) merupakan salah satu metode *machine learning* yang sangat populer dan dapat digunakan untuk klasifikasi pengenalan pola dengan cara memanfaatkan data citra pola sebagai masukan (*input*) [9]. Selain itu, *CNN* melibatkan pra-pemrosesan yang relatif sedikit dibandingkan dengan algoritma klasifikasi gambar lainnya [10]. Penelitian ini juga membandingkan tingkat akurasi antara model *CNN-MLP* dan *CNN-SVM*. Tujuan dari penelitian ini adalah untuk memperoleh akurasi tertinggi dalam klasifikasi dengan label yang telah ditentukan.

Proses penelitian melibatkan beberapa tahapan utama yang sistematis, dimulai dari pengumpulan data citra dari tulisan pola aksara Ulu Banyuasin. Data yang dipoto kemudian melalui tahap *cropping* untuk menghilangkan bagian sisi gambar agar lebih jelas. Tahapan selanjutnya membuat sebuah sistem pengenalan pola citra digital dengan penyesuaian metode dan objek yang digunakan yaitu metode *CNN* dan *SVM* untuk pengenalan citra Ulu Banyuasin. Kemudian melakukan augmentasi pada data seperti rotasi, *flipping*, *zooming*, *flip horizontal*, *flip vertical*, *share* dan perubahan kontras untuk meningkatkan variasi dan ketahanan model terhadap berbagai pola aksara. Hal ini untuk mempermudah proses pengujian citra yang telah dikumpulkan untuk diklasifikasi ke dalam *multiclass*. Ekstraksi fitur dilakukan menggunakan arsitektur *CNN*, yaitu *VGG16*, yang merupakan salah satu model *deep learning* populer dengan performa tinggi dalam pengolahan dan klasifikasi citra. *VGG16* digunakan untuk mengambil representasi fitur dari citra aksara Ulu Banyuasin sebelum dilanjutkan ke tahap klasifikasi.

Arsitektur *CNN* yang digunakan dalam penelitian ini dirancang untuk menangani karakteristik visual dari citra aksara Ulu Banyuasin. *CNN* berperan sebagai alat ekstraksi fitur, yang hasilnya kemudian digunakan sebagai masukan untuk proses klasifikasi. Dua pendekatan klasifikasi digunakan, yaitu *CNN* yang dikombinasikan dengan *Multi-Layer Perceptron* (*MLP*) dan *CNN* yang dikombinasikan dengan *Support Vector Machine* (*SVM*). Model ini dilatih menggunakan dataset yang telah dipersiapkan, dengan pembagian data menjadi set pelatihan dan pengujian. Untuk mengukur akurasi sistem, penelitian ini menggunakan metode *confusion matrix*. Metode ini sangat populer dan sering digunakan dalam evaluasi sistem yang berkaitan dengan *deep learning* [11] [12]. Evaluasi performa model dilakukan dengan menggunakan berbagai metrik standar seperti *accuracy*, *precision*, *recall* dan *f1-score* [13].

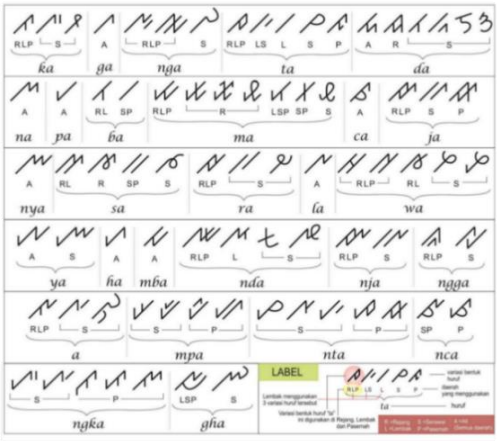
Bagian berikut akan menjelaskan secara detail setiap tahapan dalam metodologi penelitian, mencakup proses pengumpulan data, augmentasi data, ekstraksi fitur, arsitektur model, proses pelatihan dan evaluasi. Penjelasan akan dilengkapi dengan diagram alur kerja untuk memberikan gambaran yang komprehensif tentang keseluruhan sistem yang dikembangkan.




Gambar 1. Langkah-langkah Pengembangan Model Klasifikasi Aksara Ulu Banyuasin

## A. *Pengumpulan Data*

Dataset yang digunakan dalam penelitian ini adalah data citra pola aksara Ulu Banyuasin yang berjumlah 15.143 gambar dengan pembagian 168 kelas. Jumlah data pada setiap kelas dibuat seimbang, sehingga setiap kelas memiliki jumlah sampel yang relatif sama dan dapat menghindari terjadinya ketidakseimbangan data (data *imbalance*) saat pelatihan model. Data tersebut merupakan hasil poto dari berbagai tulisan pola aksara Ulu Banyuasin. Format gambar yang digunakan adalah *.jpg, .jpeg, .png*. Kemudian data yang didapatkan akan diolah untuk dijadikan sebagai data *training* dan *testing* sampai mendapatkan hasil yang di inginkan. Data *training* digunakan untuk pembelajaran dari model yang akan dibangun, dan data *testing* untuk pengujian sistem. Data pola yang digunakan dalam penelitian ini akan diubah menjadi dimensi 224x224. Gambar 2 merupakan pola Aksara Ulu Banyuasin:




Gambar 2. Pola Aksara Ulu Banyuasin

Pada gambar 3 merupakan pembagian kelas untuk klasifikasi pola aksara Ulu Banyuasin.


Gambar 3. Jumlah Kelas Dataset

## B. *Pengolahan Data*

Dataset berupa poto akan dipotong (*cropping*) untuk menghilangkan bagian tepi gambar agar lebih rapi dan mudah digunakan dalam pemrosesan lebih lanjut. Proses *cropping* bertujuan untuk memastikan bahwa hanya

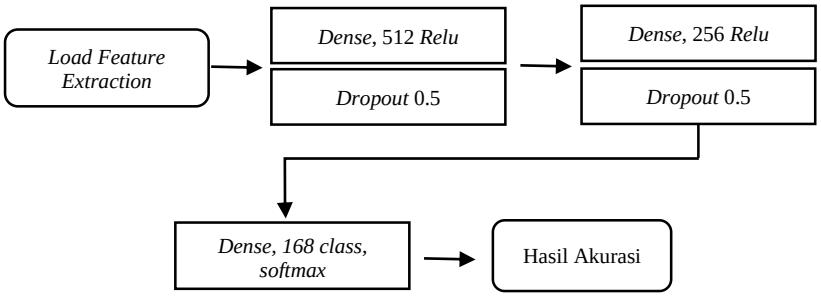
bagian yang relevan dari tulisan tangan yang digunakan, sehingga meningkatkan akurasi dalam ekstraksi fitur dan klasifikasi. Dataset dibagi menjadi 80% untuk *training* dan 20% untuk *testing* guna memastikan model mendapatkan proporsi data yang cukup untuk pembelajaran serta evaluasi. Pembagian ini dilakukan agar model dapat mengenali pola dengan baik sekaligus menghindari *overfitting*.

Gambar 4. Hasil *Cropping* Pola Aksara

*C. Pembuatan Model*

Pada penelitian ini menggunakan dua pendekatan metode yaitu *CNN-MLP* dan *CNN-SVM* untuk pengenalan pola aksara Ulu Banyuasin. Dalam tahap pembuatan model, arsitektur *CNN* yang digunakan adalah *VGG16*. Arsitektur *VGG16* sebagai ekstraksi fitur karena kemampuannya dalam menangkap pola visual yang kompleks melalui jaringan konvolusional yang dalam. Model ini telah terbukti efektif dalam berbagai tugas pengolahan gambar, serta mendukung penerapan *transfer learning* yang memungkinkan pemanfaatan bobot pralatih untuk mempercepat proses pelatihan dan meningkatkan akurasi. Dalam implementasinya, arsitektur *VGG16* hanya digunakan hingga lapisan *fully connected* terakhir. Keluaran dari lapisan tersebut berupa representasi fitur berdimensi tinggi yang merepresentasikan karakteristik visual dari citra masukan.

1) Model *VGG16-MLP*: Model ini memberikan gambaran berupa proses pengenalan pola pada sistem menggunakan algoritma *VGG16* dan *Multi-Layer Perceptron* (*MLP*). Berikut ini merupakan model dari *VGG16-SVM* dalam penelitian ini:



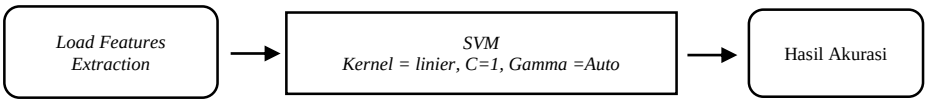

Gambar 5. Model *VGG16-MLP*

Arsitektur klasifikasi dengan *MLP* dimulai dengan lapisan *Dense* yang memiliki 512 *neuron* dengan fungsi aktivasi *ReLU*. Lapisan ini bertugas untuk menangkap pola kompleks dari fitur yang telah diekstraksi. Setelah itu, diterapkan *Dropout* sebesar 50% untuk mengurangi *overfitting* dengan menonaktifkan 50% neuron secara acak selama proses pelatihan. Lapisan berikutnya adalah *Dense Layer* dengan 256 neuron yang juga menggunakan fungsi aktivasi *ReLU* untuk memperdalam pemrosesan fitur. Selanjutnya, kembali diterapkan *Dropout* sebesar 50% untuk meningkatkan daya generalisasi model. Lapisan terakhir dalam *MLP* adalah *Dense Layer* dengan jumlah *neuron* sesuai dengan *num_classes*, yaitu jumlah kelas yang akan diklasifikasikan. Pada lapisan *MLP* ini, digunakan *optimizer Adam* dengan *learning rate* sebesar 0,0001. Fungsi *loss* yang digunakan adalah *sparse categorical cross-entropy*, dengan *batch size* sebesar 64 dan jumlah *epoch* sebanyak 50. Fungsi aktivasi yang digunakan pada lapisan *output* adalah softmax, yang berfungsi untuk mengonversi *output* menjadi probabilitas untuk setiap kelas. Kelas dengan nilai probabilitas tertinggi akan dipilih sebagai hasil prediksi akhir.

TABEL 1
PARAMETER MLP *YANG* DIGUNAKAN

| | |
|---|---|
| *Optimizer* | *Adam* |
| *Learning Rate* | 0.0001 |
| *Loss* | *Sparse categorical cross-entropy* |
| *Batch size* | 64 |
| *Epochs* | 50 |
| Aktivasi | *softmax* |

2) Model *VGG16-SVM*: *SVM* berperan sebagai pengklasifikasi utama yang bekerja dengan memisahkan data berdasarkan *margin* optimal antar kelas. Penggunaan metode ini dinilai lebih akurat dibandingkan dengan lapisan *softmax* konvensional, terutama pada data dengan distribusi yang kompleks dan jumlah yang terbatas. Keunggulan lainnya terletak pada kemampuannya dalam meminimalkan *overfitting* saat mengolah fitur hasil ekstraksi yang informatif. Model sistem dengan *VGG16-SVM* merupakan pendekatan yang menggabungkan *VGG16* sebagai ekstraksi fitur dan *SVM* sebagai klasifikasi. Pada tahap klasifikasi, *SVM* digunakan untuk memisahkan data ke dalam kelas-kelas berdasarkan fitur yang telah diekstraksi yang bekerja dengan mencari *hyperplane* optimal yang memaksimalkan margin antara kelas yang berbeda, sehingga dapat meningkatkan akurasi klasifikasi, terutama pada dataset dengan jumlah kelas yang besar seperti dalam penelitian ini.




Gambar 6. *Flowchart* Model Sistem *VGG16-SVM*

Dalam model *VGG16-SVM* yang dibangun terdapat parameter penting yang berperan dalam meningkatkan kinerja klasifikasi, di antaranya adalah *kernel*, C, dan *gamma*. *Kernel* berfungsi untuk mengubah data dari ruang fitur asli ke dalam dimensi yang lebih tinggi agar lebih mudah dipisahkan. Parameter *C* (*Regularization Parameter*) mengontrol keseimbangan antara *margin* pemisah yang besar dan kesalahan klasifikasi. Pada model *SVM* ini nilai *C* dalam penelitian ini adalah 1, karena nilai *C* yang lebih kecil memberikan batas pemisah yang lebih longgar sehingga model lebih toleran terhadap kesalahan klasifikasi. *Kernel* yang digunakan adalah *linier* dan nilai parameter *gamma* (γ) diatur secara otomatis (*auto*).

TABEL 2
PARAMETER *SVM* YANG DIGUNAKAN

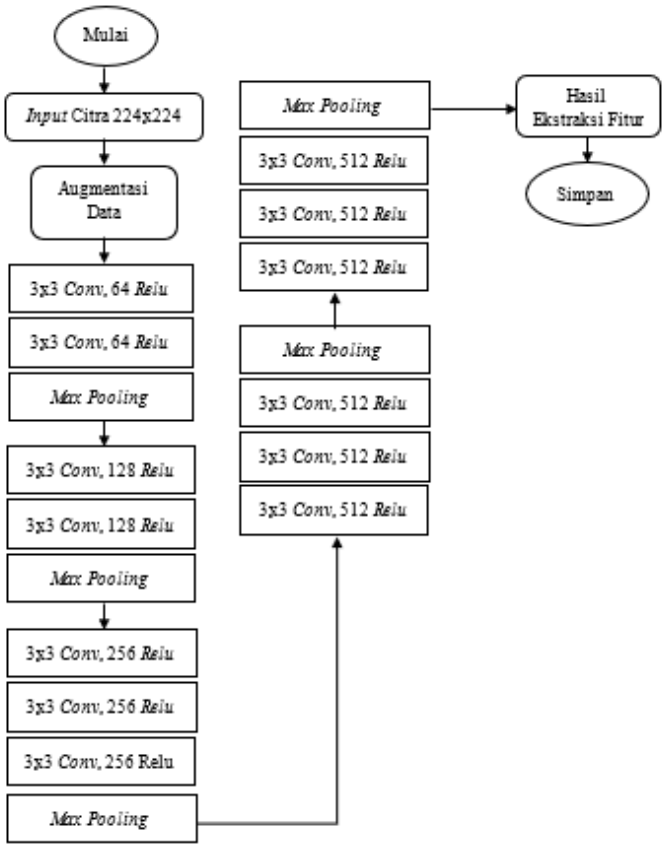
| | |
|---|---|
| *Optimizer* | *Adam* |
| *Learning Rate* | 0.0001 |
| *Loss* | *Sparse categorical cross-entropy* |
| *Batch size* | 64 |
| *Epochs* | 50 |
| *Kernel* | *Linier* |
| *C* | 1 |
| *Gamma* | *Auto* |

*D. Augmentasi Data*

Augmentasi data merupakan suatu teknik yang banyak digunakan untuk meningkatkan jumlah serta variasi dataset *deep learning*. Tujuan utama proses augmentasi data ini untuk meningkatkan performa dan akurasi dari model serta mencegah terjadinya *overfitting* [14] . Augmentasi data melakukan transformasi pada data atau dengan kata lain membuat salinan dari sumber data tanpa mengubah label pada setiap bagian dari data tersebut. [15]. Proses augmentasi data ini bertujuan untuk membuat variasi dari data asli dengan menggunakan transformasi yaitu rotasi ($40^o$), *width shift* (0.2), *height shift* (0.2), *zoom* (0.2), *shear* (0.2) *horizontal flip, all combined*.

*E. Ekstraksi Fitur*

Arsitektur *VGG16* (*Visual Geometry Group*) merupakan hasil pengembangan dari *Alexnet* di mana arsitektur ini berfokus pada memperbanyak proses fitur ekstraksi pada *layer convolution* sehingga mampu mendapatkan representasi citra yang banyak untuk dapat di klasifikasikan [16]. Arsitektur *VGG16* merupakan bagian dari model *deep learning* meliputi dari 16 layer. Arsitektur *VGG16* memiliki 13 lapisan konvolusi, dan 2 lapisan digunakan sebagai *fully_connected*, serta 1 lapisan klasifikasi [17]. Hasil dari ekstraksi fitur digunakan untuk klasifikasi dengan *MLP* dan *SVM* pada penelitian ini.




Gambar 7. Arsitektur Model *VGG16*

Arsitektur yang ditampilkan pada gambar 6 ini memiliki 14.7 juta parameter dan terdiri dari 13 lapisan konvolusi (*Conv2D*), 5 lapisan *pooling* (*MaxPooling2D*), serta *fully connected layer*.

1. *Input Layer*
   *Input_layer*_2 (224 x224 x3)
   *Input* ini menerima gambar berwarna *RGB* (*Red, Green, Blue*) dengan resolusi 224 x224 piksel dan 3 *channel* (*R, G, B*).
2. Blok 1
   2 lapisan *Conv2D* dengan 64 *filter* ukuran 3 x 3, menggunakan fungsi aktivasi *ReLU* (*Rectified Linear Unit*).
   1 lapisan *MaxPooling2D* dengan ukuran 2x2 dan *stride* 2, mengurangi dimensi menjadi 112 x 112 x 64.
3. Blok 2
   2 lapisan *Conv2D* dengan 128 *filter* ukuran 3x3 dengan aktivasi *ReLU*.
   1 lapisan *MaxPooling2D*, mengurangi dimensi menjadi 56 x 56 x 128.
4. Blok 3
   3 lapisan *Conv2D* dengan 256 *filter* ukuran 3x3, menggunakan aktivasi *ReLU*
   1 lapisan *MaxPooling2D*, mengurangi dimensi menjadi 28 x 28 x 256.
5. Blok 4
   3 lapisan *Conv2D* dengan 512 *filter* ukuran 3x3, mengunakan *ReLu*.
   1 lapisan *MaxPooling2D*, mengurangi dimensi menjadi 14 x 14 x 512.
6. Blok 5
   3 lapisan *Conv2D* dengan 512 *filter* ukuran 3x3, mengunakan *ReLu*.
   1 lapisan *MaxPooling2D*, mengurangi dimensi menjadi 14 x 14 x 512.
7. *Fully Connected Layer*
   Setelah melewati 5 blok konvolusi dan *pooling*, fitur yang diekstrak kemudian dilewatkan ke lapisan *fully connected* untuk proses klasifikasi.
8. Parameter Model
   Total Parameter: 14,7 Juta
   *Trainable parameters*: 14,7 Juta (semua parameter dapat dilatih)
   *Non-trainable parameters*: 0

*F. Melatih Model*

Pelatihan model dilakukan setelah mendapatkan hasil dari ekstraksi fitur *VGG16*. Pelatihan model dengan dua pendekatan yaitu *MLP* dan *SVM*. Data latih digunakan untuk melatih model agar dapat mempelajari pola dari setiap kelas huruf aksara Ulu Banyuasin, sedangkan data uji dipakai untuk mengevaluasi performa model pada data baru. Selama pelatihan, diterapkan teknik *early stopping* menggunakan *callback CustomEarlyStopping*. Tujuannya adalah mencegah *overfitting* dengan menghentikan pelatihan saat nilai kerugian validasi mencapai 0.15 dan akurasi validasi melampaui 96%. Penerapan *early stopping* seperti ini membantu model menjaga kemampuan generalisasinya saat berhadapan dengan data baru.

*G. Evaluasi Model*

Untuk mengevaluasi kinerja dan kehandalan model *VGG16-MLP* dan *VGG16-SVM* yang dikembangkan, penelitian ini menggunakan beberapa metrik standar seperti akurasi, presisi, *recall*, dan *F1-score*. Selain itu, visualisasi kurva akurasi/*loss* selama pelatihan juga digunakan untuk mendapatkan wawasan yang lebih dalam tentang kinerja model dan mengidentifikasi area-area yang perlu perbaikan lebih lanjut.

*H. Alat dan Bahan*

Penelitian ini dilaksanakan dengan menggunakan bahasa pemrograman *Python* yang diimplementasikan pada lingkungan komputasi berbasis *cloud*, yaitu *Google Colab Pro*. Pemanfaatan *Google Colab Pro* memungkinkan proses komputasi berjalan lebih efisien karena menyediakan sumber daya komputasi yang lebih tinggi, seperti *GPU* dan *RAM* yang lebih besar, sehingga mendukung pelatihan model secara lebih cepat dan optimal.

## III. HASIL DAN PEMBAHASAN

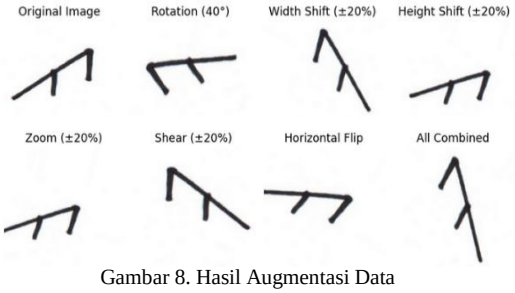
*A. Pengolahan Data*

1) *Preprocessing:* Tahap pertama pada penelitian ini adalah pembagian dataset menjadi set pelatihan dan set pengujian dengan rasio 80:20, sebagaimana ditunjukkan pada Tabel 1. Pembagian ini bertujuan untuk memastikan bahwa model dapat dilatih pada sebagian besar data yang tersedia, sementara tetap memiliki set terpisah untuk mengevaluasi performa model pada data yang belum pernah dilihat sebelumnya.

TABEL 3
*JUMLAH DATA*

| | Jumlah Data |
|---|---|
| Pelatihan | 12.624 |
| Pengujian | 2.519 |
| Jumlah Data | 15.143 |

*2)* Augmentasi Data: Setelah dataset dibagi menjadi data latih dan data uji, langkah selanjutnya adalah melakukan proses augmentasi data. Hasil dari proses augmentasi data dapat dilihat pada gambar 9, sistem telah melakukan augmentasi dataset dengan menampilkan variasi transformasi yang telah ditentukan. Proses ini telah meningkatkan keanekaragaman data latih, sehingga model lebih baik dalam mengenali pola dari berbagai sudut dan kondisi pencahayaan.




Gambar 8. Hasil Augmentasi Data

*3) Hasil Ekstraksi Fitur dengan Arsitektur VGG16*: Tahap selanjutnya adalah ekstraksi fitur menggunakan teknik *VGG16*. Dalam *transfer learning*, model *VGG16* yang telah dilatih sebelumnya sering digunakan sebagai ekstraksi fitur. Agar fitur yang telah dipelajari oleh *VGG16* tetap terjaga dan tidak berubah selama pelatihan model baru, kita perlu membekukan (*freeze*) semua layer dalam model tersebut.

```
# Freeze layer VGG16 agar tidak dilatih ulang
for layer in base_model.layers:
    layer.trainable = False
```

Gambar 9. *Source Code Freeze Layer VGG16*

Kode di atas melakukan iterasi pada setiap layer dalam *base_model* (*VGG16*) dan menetapkan *trainable = False*, yang berarti bobot (*weights*) dan bias dalam layer tersebut tidak akan diperbarui selama proses pelatihan. Dengan cara ini, model hanya akan melatih lapisan tambahan yang ditambahkan setelah *VGG16* untuk tugas klasifikasi yang spesifik. Fitur *VGG16* yang diekstraksi akan menjadi input untuk model *MLP* dan *SVM* dalam tahap klasifikasi dan evaluasi.

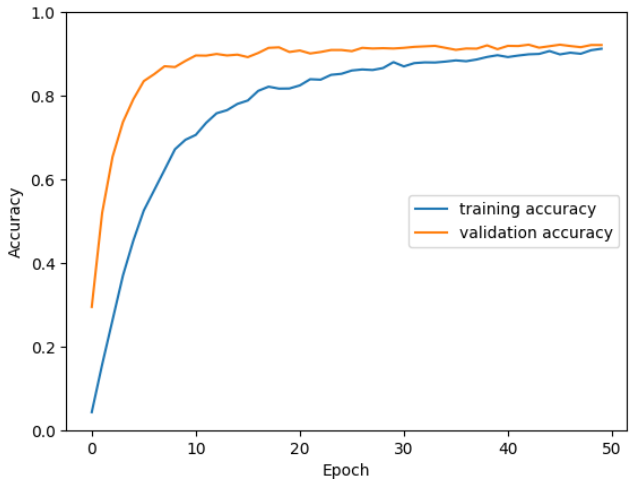
TABEL 2
HASIL EKSTRAKSI FITUR

| Tipe Data | Jumlah Sampel | Jumlah Fitur Per Sampel | Dimensi Data |
|---|---|---|---|
| ***Training*** | 12.624 | 25.088 | (12624, 25088) |
| ***Testing*** | 2.519 | 25.088 | (2519,25088) |

Dari hasil ekstraksi fitur setiap gambar diubah menjadi vektor fitur berdimensi (7 × 7 × 512) = 25.088, yang kemudian digunakan sebagai masukan untuk tahap klasifikasi. Dengan ekstraksi fitur, proses klasifikasi dapat dilakukan lebih efisien karena hanya memanfaatkan informasi yang paling relevan dari gambar, mengurangi kompleksitas komputasi dibandingkan dengan menggunakan gambar mentah secara langsung.

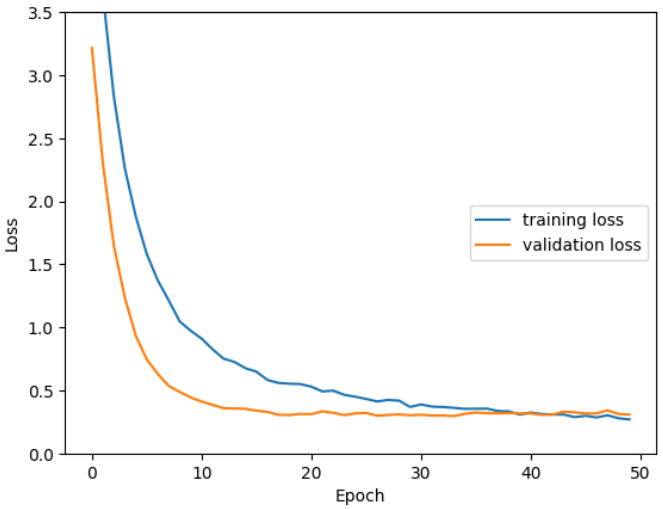
*B. Pelatihan Model VGG16-MLP*:

Hasil evaluasi model menunjukkan bahwa pada akhir pelatihan, akurasi yang diperoleh pada data training mencapai 93 % dengan nilai loss sebesar 0.2087. Hal ini menandakan bahwa model mampu mengklasifikasikan sebagian besar data validasi dengan benar. Namun, pada data validasi, akurasi sedikit menurun menjadi 91% dengan nilai *loss* sebesar 0.3401. Perbedaan ini menunjukkan adanya *overfitting*, di mana model terlalu menyesuaikan diri dengan data pelatihan tetapi kurang optimal dalam menggeneralisasi pada data baru. Meskipun demikian, akurasi yang tetap tinggi di atas 93% menunjukkan bahwa model tetap memiliki performa yang baik dalam melakukan klasifikasi.




Gambar 10. Grafik Akurasi *VGG16-MLP*

Grafik hasil pelatihan model menunjukkan performa akurasi dan *loss* pada data *training* serta *validation* selama 50 *epoch*. Pada grafik akurasi, terlihat bahwa *validation accuracy* meningkat lebih cepat dibandingkan *training accuracy*. *Validation accuracy* mencapai lebih dari 85%, sementara *training accuracy* hanya sekitar 50%. Hal ini mengindikasikan bahwa model lebih cepat belajar dari data validasi dibandingkan data *training*, yang bisa terjadi akibat perbedaan distribusi data atau regulasi yang cukup kuat dalam arsitektur model.




Gambar 11. Grafik *Loss MLP*

Evaluasi dari pelatihan model *VGG16-MLP* dilakukan menggunakan *confusion matrix*. Hasil perhitungan presisi, *recall*, dan *F1-score* disajikan pada Tabel 4 berikut.

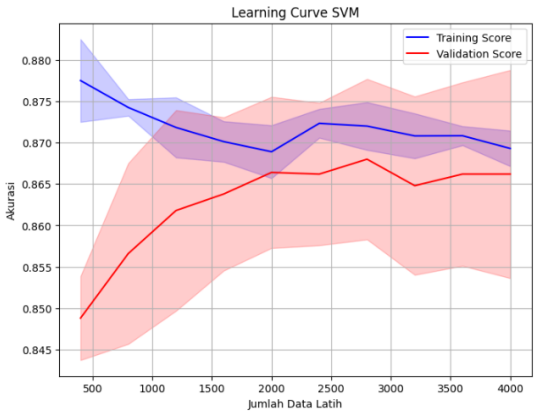
TABEL 4
TABEL *CONFUSION MATRIX VGG16-MLP*

| | **Presisi** | ***Recall*** | ***F1-Score*** |
|---|---|---|---|
| ***Accuracy*** | | | 93 % |
| ***Macro Avg*** | 93 % | 93 % | 93 % |
| ***Weighted Avg*** | 93 % | 93 % | 93 % |

Dari hasil tabel di atas, model klasifikasi yang digunakan memiliki akurasi sebesar 93%, yang berarti model mampu mengklasifikasikan 93% sampel dengan benar. Selain itu, nilai *macro average* dan *weighted average* untuk *precision*, *recall*, dan *F1-score* juga sebesar 93%, yang menunjukkan bahwa model memiliki performa yang stabil di semua kelas. *Macro average* menghitung rata-rata metrik tanpa memperhitungkan jumlah sampel di tiap kelas, sedangkan *weighted average* mempertimbangkan jumlah sampel di setiap kelas sehingga lebih mencerminkan performa keseluruhan model. Karena nilai *macro avg* dan *weighted avg* hampir sama, ini mengindikasikan bahwa model tidak terlalu bias terhadap kelas tertentu dan memiliki kinerja yang seimbang.

### *C. Pelatihan Model VGG16 – SVM*

Model *SVM* dengan pendekatan ini menggunakan *kernel linear*, dengan nilai *C* = 1 dan gamma = *auto*. Kombinasi metode ini menerapkan *SVM* pada lapisan output. Hasil akurasi yang diperoleh dari kombinasi ini lebih tinggi dibandingkan dengan model *VGG16-MLP*. *Learning curve* disajikan pada Gambar 12. Adapun hasil akurasi dari metode *VGG16-SVM* dapat dilihat pada Tabel 4.




Gambar 12. Pelatihan Model *SVM*

Grafik tersebut menunjukkan *learning curve* untuk model *SVM*, dengan sumbu *x* mewakili jumlah data latih dan sumbu *y* mewakili akurasi.

1. Kurva Biru (*Training Score*)
   Ini menunjukkan akurasi model pada data laith. Akurasi awalnya tinggi tetapi sedikit menurun saat jumlah data latih bertambah, hal ini karena model semakin diuji dengan lebih banyak variasi data.
2. Kurva Merah (*Validation Score*)
   Ini menunjukkan akurasi model pada data validasi. Akurasi meningkat seiring bertambahnya jumlah data latih, yang menunjukkan bahwa model semakin generalisasi dengan lebih banyak data.
3. Area Berwarna (*Shaded Region*)
   Area biru dan merah menunjukkan variabilitas atau *confidence interval* dari skor pelatihan dan validasi. Semakin besar daerah yang diarsir, semakin besar variasi skor, yang bisa mengindikasikan ketidakstabilan model dalam beberapa iterasi.

Akurasi validasi meningkat saat jumlah data latih bertambah, yang berarti model semakin baik dalam mengenali pola. Tidak ada tanda *overfitting* yang parah karena perbedaan antara kurva *training* dan *validation* tidak terlalu jauh. Perbedaan skor *training* dan *validation* masih ada, sehingga model bisa lebih distabilkan dengan lebih banyak data atau teknik regularisasi pada *SVM*.

TABEL 5
*Confusion Matrix VGG16-SVM*

| | ***Presisi*** | ***Recall*** | ***F1-Score*** |
|---|---|---|---|
| ***Accuracy*** | | | 99% |
| ***Macro Avg*** | 99% | 99% | 99% |
| ***Weighted Avg*** | 99% | 99% | 99% |

Hasil evaluasi model klasifikasi menunjukkan metrik yang sangat baik dengan nilai *precision, recall*, dan *F1-score*. Akurasi model mampu mengklasifikasikan seluruh data uji dengan benar tanpa kesalahan. *Precision* yang sempurna menunjukkan bahwa model tidak membuat kesalahan dalam mengklasifikasikan sampel ke dalam kelas yang salah, sedangkan *recall* menandakan bahwa model tidak melewatkan satu pun sampel dari kelas yang benar. Dengan nilai *F1-score* yang maksimal, model terbukti memiliki keseimbangan sempurna antara *precision* dan *recall*. Selain itu, nilai *macro average* dan *weighted average* mengindikasikan bahwa performa model merata di semua kelas, baik dalam kondisi data yang seimbang maupun tidak.

*D. Analisis Perbandingan*

Penelitian ini dilakukan untuk melihat perbandingan tingkat akurasi antara metode *VGG16-MLP* dan kombinasi *VGG16-SVM* dalam proses pengenalan pola aksara Ulu Banyuasin. Perbandingan ini bertujuan untuk mengevaluasi performa kedua pendekatan dalam mengenali karakter secara akurat dan efisien. Pada tabel 4.4 dapat dilihat bahwa model *VGG16-SVM* memiliki tingkat akurasi yang lebih tinggi dibandingkan dengan *VGG16-MLP*. Kombinasi *VGG16-SVM* meningkatkan kemampuan model dalam memisahkan karakter aksara secara lebih akurat.

TABEL 6
PERBANDINGAN AKURASI DENGAN *VGG16-MLP* DAN *VGG16-SVM*

| **Model** | **Akurasi *Training* (%)** | **Akurasi *Testing* (%)** |
|---|---|---|
| *VGG16 + MLP* | 93% | 93% |
| *VGG16 + SVM* | 99% | 99% |

Dari hasil penelitian, diperoleh bahwa model *VGG16* yang dikombinasikan dengan *MLP* menghasilkan akurasi sebesar 93% baik pada data training maupun data testing. Sementara itu, kombinasi *VGG16* dengan *SVM* menunjukkan performa yang lebih tinggi, dengan akurasi mencapai 99% pada kedua dataset. Perbedaan ini menunjukkan bahwa *SVM* lebih efektif dalam mengklasifikasikan fitur yang diekstraksi oleh *VGG16* dibandingkan dengan *MLP*. Hal ini karena kemampuan *SVM* dalam menangani data dengan dimensi tinggi dan menemukan *hyperplane* optimal yang memisahkan kelas secara lebih baik. Di sisi lain, meskipun *MLP* juga dapat melakukan klasifikasi, performanya cenderung lebih rendah karena tergantung pada jumlah *neuron*, fungsi aktivasi, serta bobot yang diperoleh selama pelatihan.

Keunggulan *SVM* dibandingkan *MLP* dalam penelitian ini terletak pada cara kedua metode tersebut mengolah fitur hasil ekstraksi *VGG16*. *MLP* menggunakan pendekatan berbasis jaringan saraf yang

mengandalkan optimasi bobot melalui proses pelatihan. Hal ini membuatnya bergantung pada jumlah *neuron*, fungsi aktivasi, serta *hyperparameter* yang dipilih, yang bisa menyebabkan model rentan terhadap *overfitting* atau kesulitan menemukan representasi optimal.

Sebaliknya, *SVM* bekerja dengan prinsip pencarian *hyperplane* optimal yang memisahkan kelas-kelas secara maksimal dalam ruang fitur. Dengan menggunakan *kernel trick*, *SVM* mampu menangani data dengan dimensi tinggi yang dihasilkan oleh *CNN*. Karena *SVM* mencari batas pemisah terbaik untuk setiap kelas, metode ini lebih unggul dalam klasifikasi dengan margin yang lebih jelas dibandingkan *MLP* yang belajar berdasarkan propagasi bobot. *SVM* tidak bergantung pada pembaruan bobot seperti *MLP*, sehingga lebih stabil dalam menghasilkan prediksi yang baik pada data baru.

## IV. SIMPULAN

Berdasarkan hasil penelitian, kombinasi model *VGG16* dengan *SVM* menunjukkan kinerja yang lebih unggul dalam melakukan klasifikasi dibandingkan dengan model *VGG16* yang langsung dihubungkan dengan *MLP*. Pendekatan *VGG16-MLP* menghasilkan akurasi sebesar 93%, sedangkan model *VGG16-SVM* mampu mencapai akurasi hingga 99%. Hasil ini menunjukkan bahwa penerapan *SVM* pada lapisan *output* lebih optimal dalam mengenali pola pada dataset aksara yang digunakan. Dengan tingkat akurasi yang tinggi, kombinasi *VGG16-SVM* dinilai sangat efektif untuk menyelesaikan tugas klasifikasi aksara. Sebagai saran untuk penelitian selanjutnya, studi serupa dapat dikembangkan dengan menggunakan variasi parameter yang berbeda pada algoritma *MLP* maupun *SVM* untuk mengevaluasi pengaruhnya terhadap kinerja model.

## DAFTAR PUSTAKA

[1] P. Ginting, H. Rumapea, A. P. Silalahi, P. Lumbanraja, M. Aritonang, and F. I. Komputer, "Pengenalan Pola Aksara Karo Berdasarkan Citra Pola Menggunakan Metode *K-Nearest Neighbor*," 2022. [Online]. *Available*: http://ojs.fikom-methodist.net/index.php/METHOTIKA38

[2] M. Affan Ridhollah, P. Sejarah Peradaban Islam, and F. Adab dan Humaniora, "Pelestarian Aksara Ulu Sumatera Selatan Sebagai Kearifan Lokal Masyarakat Desa Sugihwaras Melalui Pelatihan Baca Tulis Aksara Ulu," *Jurnal Magister Sejarah Peradaban Islam*, vol. 02, no. 02, 2023.

[3] A. Mulyanto, E. Susanti, F. Rosi, Wajiran, and R. Indra Borman, "Penerapan *Convolutional Neural Network* (*CNN*) pada Pengenalan Aksara Lampung Berbasis *Optical Character Recognition* (*OCR*)," 2021, [Online]. Available: https://colab.research.google.com.

[4] A. Jonathan and I. Wasito, "Perancangan Aplikasi Pengenalan Aksara Jawa Digital Menggunakan *Convulotional Neural Network* dan *Computer Vision*," *Decode: Jurnal Pendidikan Teknologi Informasi*, vol. 3, no. 2, pp. 364–377, Aug. 2023, doi: 10.51454/decode.v3i2.209.

[5] M. Kamal, F. Shaiara, C. M. Abdullah, S. Ahmed, T. Ahmed, and Md. H. Kabir, "Huruf: *An Application for Arabic Handwritten Character Recognition Using Deep Learning*," Dec. 2022, doi: 10.1109/ICCIT57492.2022.10054769.

[6] K. Ayu Safitri, R. Wulanningrum, K. Kunci -Grafologi, and T. Tangan, "Aplikasi Pengenalan Pola Tulisan Tangan Menggunakan Metode *Support Vector Machine*," 2020.

[7] A. Rita Widiarti and H. Suparwito, "Penelitian Pendahuluan Transliterasi Citra Aksara Bali Menggunakan Ciri Momen Invarian Dan Algoritma Klasifikasi *SVM* Atau *CNN*," 2023.

[8] L. Amelia Putri, A. Sitorus, N. Fitriah, H. Virul, and S. Putri Rangkuti, "Pengolahan Citra Huruf Hijaiyah Menggunakan Algoritma *Support Vector Machine*," *Jurnal Ilmu Komputer Dan Teknologi Informasi*, vol. 2, no. 3, 2024, doi: 10.61132/neptunus.v2i2.168.

[9] C. Josulin and Y. E. Kurniawati, "*Development of An Application Transforming Handwriting into Digital Form using CNN*," *Journal of Applied and Research Computer Science and Information Systems*, vol. 1, no. 2, pp. 86–99, 2023, doi: 10.61098/jarcis.v1i2.87.

[10] H. huang Zhao and H. Liu, "*Multiple classifiers fusion and CNN feature extraction for handwritten digits recognition*," *Granular Computing*, vol. 5, no. 3, pp. 411–418, Jul. 2020, doi: 10.1007/s41066-019-00158-6.

[11] F. Maisa Hana, "Perbandingan Algoritma *Neural Network Dengan Linier Discriminant Analysis* (*LDA*) Pada Klasifikasi Penyakit," 2020.

[12] M. Ihsan Ananto, W. Setya Winahju, and K. Fithriasari, "Klasifikasi Kategori Pengaduan Masyarakat Melalui Kanal LAPOR! Menggunakan *Artificial Neural Network*," 2019.

[13] Weny Indah Kusumawati and Adisaputra Zidha Noorizki, "Perbandingan Performa Algoritma *VGG16* Dan *VGG19* Melalui Metode *CNN* Untuk Klasifikasi Varietas Beras," *Journal of Computer, Electronic, and Telecommunication*, vol. 4, no. 2, 2023, doi: 10.52435/complete.v4i2.387.

[14] R. Santoso, "Augmentasi Data pada Prasasti Logam untuk Deteksi Aksara Kawi," 2023.

[15] R. Z. Fadillah, A. Irawan, M. Susanty, and I. Artikel, "Data Augmentasi Untuk Mengatasi Keterbatasan Data Pada Model Penerjemah Bahasa Isyarat Indonesia (BISINDO)," *JURNAL INFORMATIKA*, vol. 8, no. 2, 2021, [Online]. Available: http://ejournal.bsi.ac.id/ejurnal/index.php/ji

[16] E. Tanuwijaya and A. Roseanne, "Modifikasi Arsitektur *VGG16* untuk Klasifikasi Citra Digital Rempah-Rempah Indonesia," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 21, no. 1, pp. 189–196, Nov. 2021, doi: 10.30812/matrik.v21i1.1492.

[17] A. Saputro, S. Mu'min, M. Lutfi, and H. Putri, "*Deep Transfer Learning* Dengan Model Arsitektur *Vgg16* Untuk Klasifikasi Jenis Varietas Tanaman Lengkeng Berdasarkan Citra Daun," 2022.