Pengembangan Aplikasi Pengenalan Tulisan Tangan Abjad dan Angka Berbasis Convolutional Neural Network

Edelbert Strago Giamiko [1], Edwin Lesmana Tjiong [2]

[1,2]Teknik Informatika, Fakultas Ilmu Komputer dan Desain, Universitas Kalbis
Jalan Pulomas Selatan Kav. 22, Jakarta 13210
[1] *Email: edelbertsg@gmail.com*
[2] *Email: edwin.tjiong@kalbis.ac.id*

Abstract: *This research aims to make an application that recognizes and predicts handwritings of alphabets and numbers using Convolutional Neural Network (CNN). This application uses an incremental model with 2 steps for its development. The data used is EMNIST dataset, images of handwritten letters consists of Roman capital letters, Roman small letters, and Arabic numerals (0-9) that are split into 47 different classes. The model and application successfully predicted handwritings of alphabets and numbers with an average precision percentage of 76,24%.*
Keywords: *Computer vision, convolutional neural network, handwritings recognition, machine learning*

Abstrak: *Penelitian ini bertujuan untuk membuat sebuah aplikasi yang dapat mengenali dan memprediksi tulisan tangan abjad dan angka menggunakan Convolutional Neural Network (CNN). Metode pengembangan yang digunakan untuk mengembangkan aplikasi ini adalah model incremental dengan 2 tahap. Data yang digunakan merupakan gambar tulisan tangan yang bersumber dari EMNIST dataset berisikan huruf Romawi kapital, huruf Romawi kecil, dan angka Arab 0-9 yang terbagi menjadi 47 kelas yang berbeda. Model dan aplikasi berhasil memprediksi tulisan tangan abjad dan angka dengan nilai rata-rata presisi hingga sebesar 76,24%.*
Kata kunci: *Jaringan saraf konvolusional, pelihatan komputer, pembelajaran mesin, pengenalan tulisan tangan*

I. PENDAHULUAN

Masyarakat telah memasuki era Industri 4.0, di mana teknologi digunakan untuk mempermudah pekerjaan. Dengan perkembangan teknologi yang terus berkembang pesat, teknologi kecerdasan buatan atau AI (*Artificial Intelligence*) juga semakin disempurnakan [1]. Perkembangan AI ini lalu menghasilkan sebuah bagian daripada AI yang banyak digunakan yaitu pembelajaran mesin atau ML (*Machine Learning*).

Perkembangan teknologi digital membuat masyarakat mulai meninggalkan konsep lama. Salah satunya adalah menulis di atas kertas. Tradisi menulis di atas kertas menggunakan alat tulis mulai ditinggalkan karena sudah adanya aplikasi untuk mengetik dokumen dan menyimpannya di tempat yang lebih aman, dibandingkan sebuah kertas atau buku yang dapat rusak jika dibandingkan dengan teks digital. Tetapi, bukan berarti masyarakat meninggalkan tradisi menulis di atas kertas dengan sepenuhnya. Walaupun konvensional, tulisan tangan memiliki ciri artistik. Tulisan tangan dari orang ke orang juga berbeda-beda, dan mempunyai keunikan masing-masing.

Dengan menggunakan teknologi *machine learning* dan metode CNN, penulis melakukan penelitian dan mengembangkan sebuah mesin dan aplikasi yang dapat mengenali teks tulisan tangan abjad dan angka. Aplikasi ini dapat berguna untuk melakukan digitalisasi tulisan tangan abjad dan angka ke dalam bentuk digital.

II. METODE PENELITIAN

A. Teori Pendukung

Berikut merupakan beberapa teori pendukung yang membantu penulis dalam mengembangkan, meneliti dan menganalisa model.

1. Pengolahan Citra Digital

Pengolahan citra digital adalah suatu proses manipulasi dan analisis data citra menggunakan algoritma komputer [2]. Citra digital digunakan sebagai representasi numerik dari gambar atau video yang dapat diakses, diproses, dan disimpan oleh komputer. Pengolahan digital digunakan untuk memproses sebuah masukkan gambar ke aplikasi menggunakan OpenCV [3].

2. Kecerdasan Buatan

Kecerdasan Buatan (*Artificial Intelligence*) adalah suatu ilmu komputer yang digunakan untuk mengeksekusi tugas-tugas dan pengambilan keputusan seperti layaknya manusia dalam cara berpikir [4]. Kecerdasan buatan dikembangkan menggunakan sistem yang dapat beradaptasi sesuai kondisi, dan menyelesaikan masalah dengan algoritma yang digunakan sistem tersebut [5].

3. Pembelajaran Mesin

Pembelajaran mesin (*Machine Learning*) adalah suatu ilmu komputer dan cabang kecerdasan buatan yang fokus pada pengembangan sistem yang dapat belajar secara otomatis dari data yang diberikan [6]. Tujuan utama dari *machine learning* adalah membuat sistem yang dapat belajar dan beradaptasi secara otomatis dan meningkatkan kinerjanya tanpa perlu melakukan pemrograman ulang atau mengubah fungsi sistem hanya untuk melakukan tugas tertentu.

4. *Artificial Neural Network*

Artificial Neural Network, atau sering kali hanya disebut sebagai *Neural Network*, adalah salah satu bagian daripada pembelajaran mesin. Neural network meniru pemetaan dan alur syaraf otak manusia dengan tujuan untuk mengambil pola-pola penting sebagai input dan memproses pola data tersebut menjadi output. Konsep ini digunakan untuk memecahkan masalah, mengambil keputusan, dan mengenali pola pada suatu data.

5. Deep Learning

Deep learning adalah sebuah metode pembelajaran mesin yang menggunakan banyak lapisan saraf tiruan yang bertujuan untuk mempelajari representasi pola data kompleks dengan lebih mendalam. Deep learning menggunakan sebuah metode pembelajaran di mana mesin dilatih dengan mengulang aktivitas dan pengambilan keputusan dengan sebuah dataset berulang kali. Perulangan tersebut disebut sebagai epoch. Pada sebuah *epoch*, mesin akan berulang kali diberi beberapa data dalam bentuk *batch*. Misalnya pada suatu *epoch*, mesin akan diberi 1000 buah data dengan *batch size* 100, sehingga dalam satu *epoch*, mesin akan melakukan 10 kali iterasi pelatihan, di mana dalam satu pelatihan mesin akan diberi 100 buah data. Mesin juga dapat dilatih lebih lagi dengan menambahkan jumlah *epoch* [7].

6. *Convolutional Neural Network*

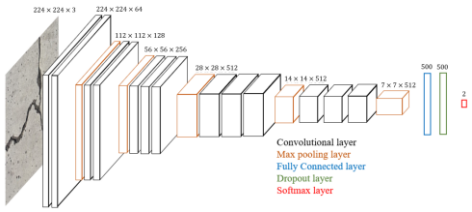
Convolutional Neural Network **(**CNN) adalah suatu tipe arsitektur jaringan saraf tiruan untuk pembelajaran mesin yang dirancang khusus untuk memproses data berpola *grid* seperti gambar dan video [8]. CNN menggunakan tiga buah lapisan. *Convolution layer*, yaitu lapisan utama dan lapisan yang memiliki fungsionalitas terpenting pada CNN. *Pooling layer*, yaitu lapisan yang digunakan untuk mengurangi dimensi gambar representasi yang telah dihasilkan oleh lapisan konvolusi. *Fully connected layer* (*FC layer*), yaitu lapisan terakhir pada CNN yang bekerja sebagai pengumpulan informasi yang telah diambil pada lapisan konvolusi dan *pooling* sebelumnya untuk melakukan klasifikasi.

7. *Confusion Matrix*

Confusion Matrix digunakan untuk menggambarkan kinerja sebuah model pembelajaran mesin untuk klasifikasi. *Confusion matrix* memberikan gambaran tentang seberapa baik model tersebut dapat memprediksi sebuah kelas. Confusion matrix memiliki 4 bagian utama, yaitu *True positive* (TP), *False positive* (FP), *True negative* (TN), dan *False negative* (FN). Dari keempat bagian tersebut, kita dapat menghitung *Precision, Recall,* dan *F1-score*.

8. *Visual Geometry Group*

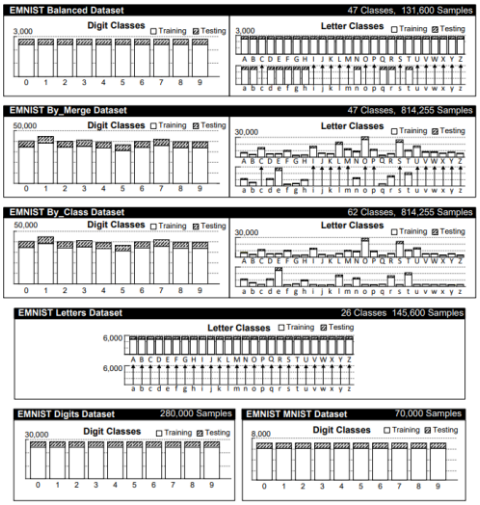
Visual Geometry Group **(VGG)** merupakan suatu arsitektur CNN sebagai fondasi pengembangan aplikasi klasifikasi gambar dan deteksi objek. VGG terdiri dari beberapa variasi, seperti VGG-16 dan VGG-19. VGG-16 artinya ada 16 buah lapisan neuron dalam arsitektur tersebut.



Gambar 1 *Architecture of the modified VGG16 model.* [9].

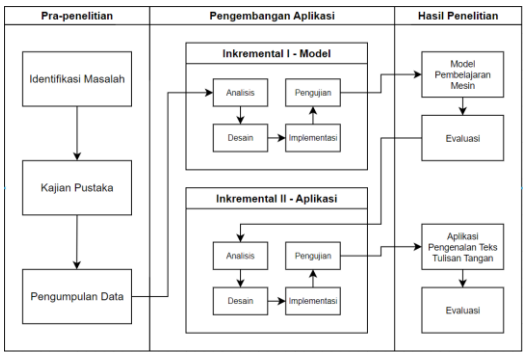
9. *Extended MNIST*

Extended MNIST **(EMNIST)** merupakan salah satu *dataset* populer yang digunakan untuk pembelajaran mesin di bidang klasifikasi gambar [10]. Seperti namanya, EMNIST adalah ekstensi daripada *dataset* MNIST yang hanya meliputi kelas tulisan tangan angka nol (0) sampai sembilan (9). EMNIST menambahkan 52 buah kelas baru, yaitu tulisan tangan huruf "a" sampai "z" dan "A" sampai "Z". EMNIST menyediakan beberapa jenis *dataset*, yaitu *ByClass, ByMerge, Balanced, Digits, Letters,* dan *MNIST.*



Gambar 2 *Visual breakdown of the EMNIST datasets* [10].

B. Kronologis Penelitian



Gambar 3 Proses Penelitian

1. Identifikasi Masalah

Terkadang tulisan tangan sulit dibaca dikarenakan beberapa hal seperti keterbatasan penulis, penulisan yang tidak konsisten, dan bervariasinya tulisan orang yang berbeda. Ciri khas penulisan tiap orang berasal dari terbiasanya orang tertentu pada cara mereka menuliskan huruf atau angka.

Solusi yang diambil oleh penulis adalah untuk membuat sebuah kecerdasan buatan yang dilatih untuk mengklasifikasi tulisan tangan abjad dan huruf agar dapat digunakan untuk mengubah tulisan tangan tersebut menjadi bentuk digital. Proses ini berguna agar tulisan mudah dibaca, karena tulisan digital lebih rapih dan penulisannya konsisten, berkat adanya sistem *font* dalam tulisan digital.

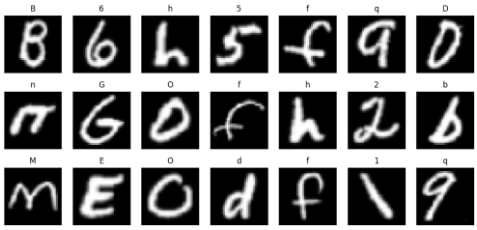
Hasil yang diharapkan oleh penulis adalah sebuah model pembelajaran mesin yang dapat mengklasifikasi tulisan tangan abjad dan huruf dengan akurasi lebih dari 75%.

Latar belakang masalah	Solusi	Hasil
• Terkadang tulisan tangan sulit dibaca • Tulisan tangan tidak konsisten • Tulisan tangan bervariasi dari orang ke orang	• Membuat sebuah kecerdasan buatan yang dapat mengenali dan mengklasifikasi tulisan tangan • Mengubah tulisan tangan tersebut ke dalam teks digital agar mudah dibaca	• Sebuah aplikasi yang dapat mengubah gambar berupa tulisan tangan ke bentuk teks digital

Gambar 4 Kerangka Pemikiran

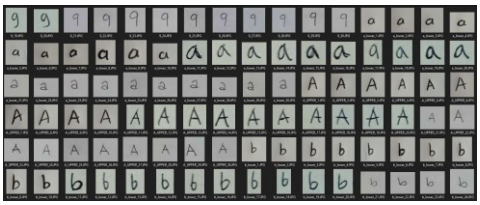
2. Pengumpulan data

Data yang diperlukan dalam penelitian ini adalah gambar-gambar tulisan tangan abjad dan angka. Untuk pengembangan model pembelajaran mesin, dataset yang digunakan untuk pelatihan mesin klasifikasi adalah dataset EMNIST (*Extended MNIST*) Balanced, yang memiliki 47 buah kelas dan jumlah gambar dari kelas-kelas tersebut sudah diseimbangkan.



Gambar 5 Contoh dataset EMNIST *Balanced*, ukuran 32x32 piksel

Lalu untuk percobaan dan evaluasi aplikasi, penulis meminta beberapa orang untuk menuliskan angka nol (0) sampai sembilan (9), huruf kecil "a" sampai "z", dan juga huruf kapital "A" sampai "Z". Gambar-gambar ini akan digunakan sebagai data mentah untuk mengevaluasi aplikasi yang telah jadi.



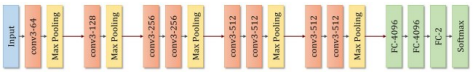
Gambar 6 Contoh beberapa gambar tulisan tangan yang terkumpul

3. Inkremental I

Dalam inkremental pertama, penulis mengembangkan model untuk klasifikasi tulisan tangan.

a. Analisis Model VGG-11

Model pembelajaran mesin ini akan dilatih menggunakan dataset *EMNIST Balanced*, dan menggunakan CNN dengan arsitektur VGG-11. Arsitektur VGG-11 memiliki 11 buah lapisan konvolusi, *fully connected*, dan *output layer*. Model ini akan dilatih dengan dataset yang telah ditransformasi dan dinormalisasi. Dataset yang dipilih ialah EMNIST balanced, berupa 47 buah kelas gambar tulisan tangan yaitu angka nol (0) sampai sembilan (9), huruf kapital "A" sampai "Z", dan beberapa huruf kecil yang tidak digabung kepada kelas yang memiliki penulisan yang sama dengan pasangan huruf kapitalnya (a, b, d, e, f, g, h, n, q, r, t). Gambar berukuran 28x28 piksel, dan memiliki 1 dimensi warna (*grayscale*) [10].



Gambar 7 Arsitektur dasar VGG-11 [11]

VGG-11 memiliki 5 buah lapisan *maxpool*, sehingga gambar-gambar yang digunakan untuk melatih model harus memiliki minimal resolusi sebesar 32x32 piksel. Dataset harus ditransformasi sedemikian rupa agar dapat digunakan dalam arsitektur VGG-11. Maka dari itu, dataset EMNIST akan ditransformasi dari 28x28 piksel menjadi ukuran 32x32 piksel.

Tabel 1 Kebutuhan Nonfungsional Guna Mengembangkan Model VGG-11

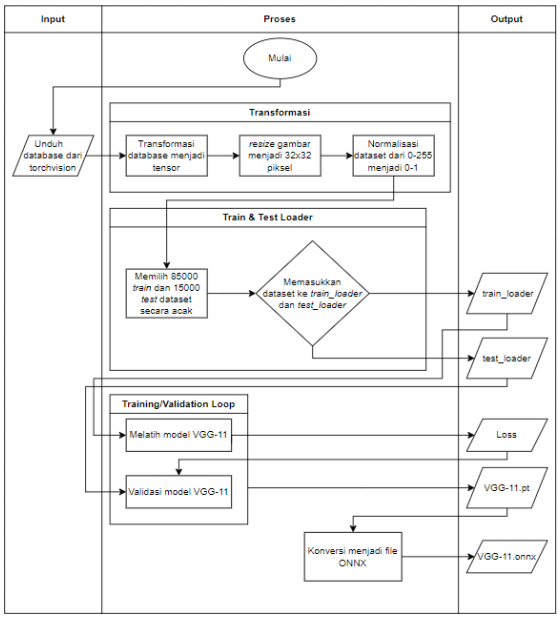
Perangkat Keras	Perangkat Lunak
CPU: AMD Ryzen 7 1700 Eight-Core Processor, 3000 Mhz, 8 Cores 16 Threads **GPU***: AMD Radeon RX580 4 GB VRAM 1077 MHz **RAM**: 1 slot 16 GB DDR4 2400 MHz **Disk**: 931.51 GB	**OS**: Windows Version 10.0.19045 Build 19045 **IDE**: PyCharm Community Edition Version 2023.3.4 **Bahasa Pemrogramman:** Python 3.12.2 **Pustaka**: • PyTorch 2.2.1 • Torchvision 0.17.1 • Numpy 1.26.4 • ONNX 1.16.0

b. Desain Model VGG-11

Pertama, dataset akan diunduh menggunakan pustaka PyTorch dan disimpan di dalam komputer. Jika file dataset sudah ada di dalam komputer, tidak perlu diunduh kembali. Karena dataset EMNIST memiliki dataset untuk *training* dan *testing*, dataset akan dimuat dua kali, pertama untuk *training* dan kedua untuk *testing*.

Tipe *split* yang diambil adalah *balanced*. Dataset yang sudah dimuat kemudian akan ditransformasi agar dapat dijadikan sebagai bahan pelatihan model. Transformasi yang dilakukan adalah mengubah dataset ke bentuk *tensor*, dan memperbesar ukuran gambar menjadi 32x32 piksel. Penulis hanya mengambil 85000 buah gambar untuk *training* dan 15000 buah gambar untuk testing dari dataset, dan gambar diambil secara acak. Dataset tersebut dijadikan *dataloader* untuk *training* dan *testing*. Jumlah siklus pelatihan (*epoch*) adalah 10, dan *batch* yang diambil setiap pelatihan adalah 1000 gambar. Hasil *loss* setiap *batch* dan nilai *precision*, *recall*, dan *F1-score* setiap evaluasi akan di *print* dalam *console*.

Setelah proses *training* selesai, model disimpan dan dikonversi menjadi ONNX untuk digunakan dalam aplikasi.



Gambar 8 Desain Proses Kinerja Inkremental I

c. Implementasi model VGG-11

Kode diimplementasikan dengan IDE *PyCharm Community Edition* versi 2023.3.4, dan bahasa pemrograman Python 3.12.2. Pustaka yang digunakan adalah PyTorch untuk pengembangan model VGG-11 dan pra-proses database, NumPy untuk memilih *subset dataloader* secara acak, dan Matplotlib untuk *plotting* dan menampilkan contoh gambar pada dataset.

d. Pengujian model VGG-11

Selanjutnya penulis menguji apakah *training* pada model berhasil dilakukan. Hasil dilihat dari turunnya nilai *loss* setiap *batch*, dan naiknya nilai *precision*, *recall* dan *F1-score* model dalam klasifikasi gambar setiap evaluasi.

4. Inkremental II

Dalam inkremental kedua, penulis mengembangkan aplikasi menggunakan model yang telah dilatih, membuat program untuk praproses gambar, dan juga merancang UI aplikasi.

a. Analisis Aplikasi

Aplikasi harus dapat melakukan praproses gambar. Gambar harus dapat diproses sedemikian rupa menjadi mirip dengan gambar yang dimiliki dataset EMNIST yang sudah ditransformasi. Aplikasi harus dapat melakukan prediksi terhadap gambar tersebut dan mengeluarkan hasil. Hasil yang diperoleh adalah tiga terbaik dengan masing-masing *confidence score*.

Ada beberapa ketentuan yang harus diikuti dalam pemilihan gambar. Tulisan tangan pada gambar harus berwarna hitam, dan ditulis di atas kertas putih. Foto tidak boleh pudar dan harus terlihat jelas dengan cahaya terang. Gambar hanya dapat memiliki satu buah angka atau huruf dan garis pada angka atau huruf tersebut tidak boleh putus. Format pada gambar harus berupa ".jpg".

Tabel 2 Kebutuhan Nonfungsional Guna Mengembangkan Aplikasi

Perangkat Keras	Perangkat Lunak
CPU: AMD Ryzen 7 1700 Eight-Core Processor, 3000 Mhz, 8 Cores 16 Threads **GPU***: AMD Radeon RX580 4 GB VRAM 1077 MHz **RAM**: 1 slot 16 GB DDR4 2400 MHz **Disk**: 931.51 GB	**IDE**: PyCharm Community Edition Version 2023.3.4 **Bahasa Pemrogramman:** Python 3.12.2 **Pustaka**: • PyTorch 2.2.1 • Numpy 1.26.4 • Matplotlib 3.8.3 • OpenCV2 4.9.0.80 • Tkinter 8.6.12 • PIL 10.2.0

b. Desain Aplikasi

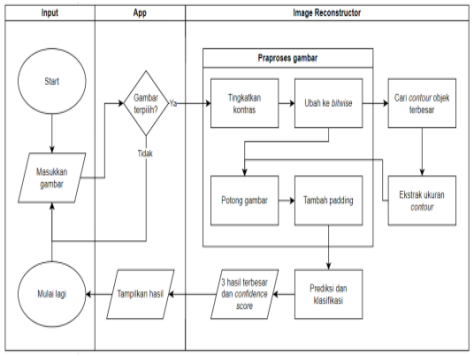
Aplikasi akan dikembangkan dengan menggunakan bahasa pemrograman Python. Akan ada dua bagian daripada aplikasi, yang pertama bagian praproses dan prediksi gambar, dan yang kedua bagian *user interface* (UI). Pada saat pengguna aplikasi menekan tombol dan memilih gambar, gambar yang dipilih tersebut akan diproses sedemikian rupa hingga menjadi gambar yang serupa dengan apa yang ada dalam dataset EMNIST, yaitu latar berwarna hitam dengan tulisan berwarna putih. Setelah itu aplikasi akan memprediksi tulisan tersebut, dan dikembalikan ke bagian UI untuk ditampilkan hasilnya.

Berikut adalah tahapan pada praproses gambar:

1. Kontras gambar akan ditingkatkan agar latar gambar menjadi sangat putih.
2. Gambar tersebut akan dijadikan *bitwise*, di mana dengan *threshold* tertentu, mengubah piksel hitam menjadi putih dan sebaliknya

3. Program akan mencari *contour* objek dalam gambar hitam putih tersebut
4. Program akan membuat potongan gambar baru dari *contour* terbesar dalam gambar
5. Potongan diubah ke ukuran 32x32 piksel

Setelah praproses, gambar dimasukkan ke model VGG-11 yang sudah dilatih untuk memprediksi dan mengklasifikasi huruf atau angka yang ada dalam gambar. Setelah itu, tiga hasil terbesar beserta nilai *confidence score* untuk ketiga kelas tersebut dikembalikan ke UI aplikasi untuk ditampilkan.



Gambar 3.7 Desain diagram flowchart aplikasi (inkremental II)

Gambar 9 Desain Proses Kinerja Inkremental II

c. Implementasi Aplikasi

Program diimplementasikan dengan mengikuti *flowchart* dan menggunakan pustaka-pustaka yang diperlukan. Seperti pada inkremental I, bahasa pemrograman yang digunakan adalah Python versi 3.12.2, diketik menggunakan IDE *PyCharm Community Edition* versi 2023.3.4.

Pada bagian praproses gambar, pustaka yang digunakan adalah OpenCV untuk memproses gambar, dan juga memuat model ONNX dan memprediksi gambar dengan model tersebut menggunakan cv2.dnn, Numpy untuk memproses *array*, PyTorch untuk memproses *output* daripada model, dan juga Matplotlib untuk menampilkan gambar yang telah diproses. Pada bagian UI aplikasi, pustaka yang digunakan adalah Tkinter sebagai penyedia UI, dan PIL untuk memuat gambar ke dalam UI aplikasi.

d. Pengujian Aplikasi

Pengujian dilakukan dalam beberapa tahap:

1. Menguji dan mengevaluasi hasil daripada praproses gambar, hingga dapat mencapai hasil yang diinginkan.
2. Melihat apakah model dapat mengklasifikasikan gambar yang sudah diproses dengan tepat.
3. Menguji aplikasi dan menggabungkan aplikasi dengan kode praproses gambar tersebut.

III. HASIL DAN PEMBAHASAN

A. Inkremental I

Pada inkremental I, penulis mengambil beberapa data dari hasil pelatihan model klasifikasi. Pada setiap *epoch*, data yang dikumpulkan adalah nilai *loss*, nilai TP, TN, FP dan FN dari tiap kelas, serta nilai *precision*, *recall* dan *F1-score* dari tiap kelas yang dikalkulasi menggunakan rumus *confusion matrix*.

$$recision = \frac{TP}{TP+FP} \quad (1)$$

$$recall = \frac{TP}{TP+FN} \quad (2)$$

$$F1Score = \frac{2*precision*recall}{precision+recall} \quad (3)$$

Dimana TP adalah *True Positive*, FP adalah *False Negative*, dan FN adalah *False Negative*.

Tabel 3 Hasil Nilai *Precision*, *Recall* Dan *F1-Score* Tiap Kelas Setelah Pelatihan Model VGG-11

Kelas	P	R	F1	Kelas	P	R	F1
0	71,28%	66,88%	69,01%	O	73,97%	77,14%	75,52%
1	58,73%	67,68%	62,89%	P	97,32%	97,61%	97,47%
2	87,26%	95,47%	91,18%	Q	95,78%	95,47%	95,62%
3	98,29%	98,63%	98,46%	R	95,48%	98,34%	96,89%
4	93,02%	95,13%	94,06%	S	91,85%	93,61%	92,72%
5	91,09%	91,39%	91,24%	T	95,78%	95,78%	95,78%
6	91,51%	95,41%	93,42%	U	91,28%	96,02%	93,59%
7	95,54%	98,47%	96,98%	V	92,52%	96,56%	94,50%
8	92,01%	94,12%	93,05%	W	98,47%	96,11%	97,27%
9	73,73%	85,02%	78,97%	X	97,27%	98,17%	97,72%
A	96,18%	97,90%	97,03%	Y	95,07%	88,92%	91,89%
B	97,35%	97,35%	97,35%	Z	94,04%	90,09%	92,02%
C	94,91%	97,54%	96,21%	a	88,96%	93,55%	91,20%
D	96,01%	96,98%	96,49%	b	97,64%	94,14%	95,85%
E	98,68%	97,40%	98,04%	d	97,07%	96,75%	96,91%
F	70,67%	63,86%	67,09%	e	96,68%	96,68%	96,68%
G	94,95%	91,56%	93,22%	f	68,05%	68,25%	68,15%
H	98,41%	96,26%	97,32%	g	79,02%	70,47%	74,50%
I	68,59%	69,93%	69,26%	h	97,16%	95,36%	96,25%
J	93,97%	98,28%	96,08%	n	96,50%	96,19%	96,34%
K	98,44%	97,22%	97,83%	q	73,23%	61,56%	66,89%
L	65,83%	50,16%	56,94%	r	96,32%	94,29%	95,30%
M	99,69%	98,47%	99,07%	t	92,71%	94,01%	93,36%
N	92,61%	99,15%	95,77%				

Setelah *precision*, *recall*, dan *F1-Score* untuk setiap kelas telah terakumulasi, selanjutnya program akan mengkalkulasikan *macro precision*, *macro recall*, dan *macro F1-Score*, yaitu nilai rata-rata dari keseluruhan kelas.

$$macro\ precision = \frac{\sum precision}{n\_classes} \quad (4)$$

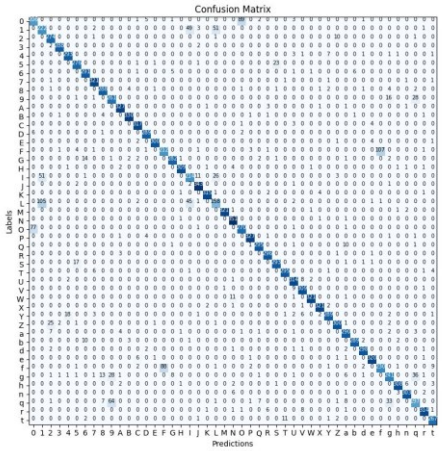
$$macro\ recall = \frac{\sum recall}{n\_classes} \quad (5)$$

$$macro\ F1score = \frac{\sum F1score}{n\_classes} \quad (6)$$

Tabel 4 Hasil Nilai *Loss, Macro Precision, Macro Recall* Dan *Macro F1-Score* Tiap Epoch Pada Pelatihan Model VGG-11

Epoch	*Loss*	*Precision*	*Recall*	*F1-score*
1	*3,0169*	*59,98%*	*44,71%*	*40,15%*
2	*1,5104*	*74,81%*	*73,70%*	*72,33%*
3	*0,8668*	*81,16%*	*80,30%*	*79,53%*
4	*0,6586*	*84,43%*	*83,84%*	*83,33%*
5	*0,5458*	*86,09%*	*85,67%*	*85,25%*
6	*0,4376*	*87,05%*	*86,89%*	*86,60%*
7	*0,4078*	*87,59%*	*87,65%*	*87,44%*
8	*0,3490*	*88,52%*	*88,28%*	*87,98%*
9	*0,3673*	*89,23%*	*89,16%*	*89,00%*
10	*0,3340*	*89,81%*	*89,90%*	*89,77%*

Pada *epoch* terakhir, program juga akan menampilkan grafik *confusion matrix* pada epoch ke-10.



Gambar 10 Hasil Plotting Confusion Matrix

Berdasarkan hasil diatas, dapat dilihat bahwa model memiliki kesulitan mengklasifikasi huruf atau angka tertentu. Contohnya seperti angka 0 dan huruf O. Karena keduanya memiliki penulisan yang sama, maka model dapat mengklasifikasikan 0 sebagai O, atau sebaliknya.

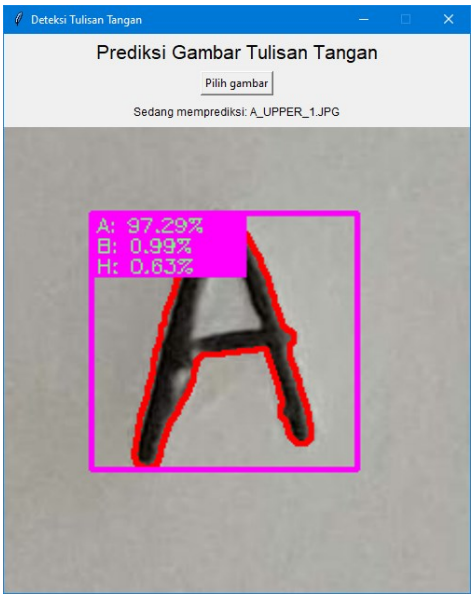
B. Inkremental II

Dengan bantuan Tkinter, penulis dapat membuat tampilan aplikasi sebagai berikut.

Gambar 11 Tampilan Aplikasi Sebelum Melakukan Prediksi Pada Gambar

Pengguna dapat memilih gambar yang diinginkan, sesuai dengan ketentuan gambar yang telah dijelaskan sebelumnya. Setelah memilih gambar, aplikasi akan melakukan prediksi terhadap gambar tersebut dan memperoleh hasil yang akan ditampilkan dalam aplikasi.

Dalam kasus gambar 32 dibawah, aplikasi melakukan prediksi dan klasifikasi pada gambar bernama ”A_UPPER_1.JPG”. Dalam gambar tersebut, terdapat tiga buah prediksi tertinggi berdasarkan nilai *confidence score*. Prediksi paling tinggi adalah ”A” dengan persentase *confidence score* sebesar 97.29%. Prediksi kedua tertinggi adalah ”B” dengan persentase *confidence score* sebesar 0.99%. Prediksi ketiga tertinggi adalah ”H” dengan persentase *confidence score* sebesar 0.63%.



Gambar 12 Tampilan Aplikasi Setelah Melakukan Prediksi pada Gambar

Setelah melakukan percobaan dengan 1860 buah gambar yang penulis minta dari teman penulis, dan menganalisis hasil tiap prediksi dan klasifikasi yang dilakukan oleh aplikasi satu per satu, hasil tersebut diakumulasi sehingga dapat menghasilkan akurasi daripada aplikasi.

Tabel 5 Hasil Nilai Persentase Akurasi Pada Prediksi Dan Klasifikasi Aplikasi Pada Tiap Kelas

Label	Prediksi Benar	Prediksi Percobaan	Akurasi	Label	Prediksi Benar	Prediksi Percobaan	Akurasi	Label	Prediksi Benar	Prediksi Percobaan	Akurasi
0	1	30	3,33%	G	28	30	93,33%	W/w	60	60	100,00%
1	0	30	0,00%	H	30	30	100,00%	X/x	59	60	98,33%
2	17	30	56,67%	I/i	38	60	63,33%	Y/y	59	60	98,33%
3	29	30	96,67%	J/j	60	60	100,00%	Z/z	60	60	100,00%
4	30	30	100,00%	K/k	60	60	100,00%	a	0	30	0,00%
5	27	30	90,00%	L/l	59	60	98,33%	b	30	30	100,00%
6	10	30	33,33%	M/m	59	60	98,33%	d	28	30	93,33%
7	29	30	96,67%	N	21	30	70,00%	e	18	30	60,00%
8	16	30	53,33%	O/o	0	60	0,00%	f	17	30	56,67%
9	7	30	23,33%	P/p	60	60	100,00%	g	30	30	100,00%
A	30	30	100,00%	Q	30	30	100,00%	h	25	30	83,33%
B	30	30	100,00%	R	30	30	100,00%	n	11	30	36,67%
C/c	36	60	60,00%	S/s	1	60	1,67%	q	30	30	100,00%
D	30	30	100,00%	T	30	30	100,00%	r	14	30	46,67%
E	30	30	100,00%	U/u	33	60	55,00%	t	30	30	100,00%
F	30	30	100,00%	V/v	56	60	93,33%	**Total**	1418	1860	76,24%

Dari tabel 5, dapat dilihat bahwa aplikasi masih belum dapat memprediksi beberapa kelas dengan konsisten, dan bahkan tidak dapat memprediksi beberapa kelas sama sekali. Pada kelas ”1”, ”O”, dan ”a”, aplikasi tidak dapat memprediksi kelas tersebut sama sekali. Dari 30 gambar yang diberikan untuk masing-masing kelas, aplikasi gagal untuk memprediksi seluruh gambar. Sedangkan pada kelas ”0”, ”6”, ”9”, ”S”, ”n” dan ”r”, aplikasi mengalami kesulitan dalam memprediksi kelas tersebut dikarenakan kelas tersebut memiliki karakteristik yang mirip dengan kelas lainnya. Penulis memiliki spekulasi bahwa model yang dilatih memiliki bias terhadap kelas tersebut.

Dari data yang telah diakumulasi, aplikasi berhasil memperoleh akurasi hingga 76,24%.

IV. SIMPULAN

Dari seluruh kegiatan penelitian yang telah dilakukan maka simpulan dari penelitian ini adalah bahwa mesin klasifikasi berhasil dibentuk dan dapat melakukan pelatihan dengan lancar. Lalu evaluasi VGG-11 menentukan bahwa model CNN dapat mengenali dan mengklasifikasi gambar tulisan tangan abjad dan angka, kemudian evaluasi model setelah pelatihan dapat menempuh nilai *precision* sebesar 89,81%, *recall* sebesar 89,90%, dan *F1-score* sebesar 89,77%, Pengujian daripada aplikasi dengan 1860 buah gambar tulisan tangan dapat dilakukan, tetapi memiliki kendala di mana klasifikasi terhadap beberapa kelas gagal dilakukan, Akurasi aplikasi pada klasifikasi adalah 76.24%, pada beberapa kelas, aplikasi memiliki presisi yang kecil dalam mengklasifikasikan kelas tersebut. Hal ini dikarenakan penulisan abjad/angka tersebut mirip dengan kelas abjad/angka lainnya, seperti huruf “o” memiliki penulisan yang sama dengan angka “0”, dan huruf “i” memiliki penulisan yang sama dengan angka “1”, presisi tertinggi yang diperoleh aplikasi adalah 100%, atau dapat melakukan klasifikasi pada 19 kelas tertentu dengan sempurna, dan presisi terendah yang diperoleh aplikasi adalah 0%, atau gagal melakukan klasifikasi pada 3 kelas tertentu.

DAFTAR RUJUKAN

[1] Y. S. Ong and A. Gupta, “AIR5: Five Pillars of Artificial Intelligence Research,” *IEEE Trans Emerg Top Comput Intell*, vol. 3, no. 5, pp. 411–415, Oct. 2019, doi: 10.1109/TETCI.2019.2928344.

[2] A. Fadjeri, B. A. Saputra, D. K. Adri Ariyanto, and L. Kurniatin, “Karakteristik Morfologi Tanaman Selada Menggunakan Pengolahan Citra Digital,” *Jurnal Ilmiah SINUS*, vol. 20, no. 2, p. 1, Jul. 2022, doi: 10.30646/sinus.v20i2.601.

[3] A. Parikesit, E. Lesmana Tjiong, F. Ilmu Komputer dan Desain, and I. Teknologi dan Bisnis Kalbis Jalan Pulomas Selatan Kav, “Perancangan Purwarupa Mobil Otonom Menggunakan Simulator Udacity dengan Metode Convolutional Neural Network,” 2023.

[4] A. Octaviani and P. Dewi, “Kecerdasan Buatan sebagai Konsep Baru pada Perpustakaan,” *ANUVA*, vol. 4, no. 4, pp. 453–460, 2020.

[5] S. Ramadhani, M. A. Hariyadi, and C. Crysdian, “The Evaluation of Computer Science

Curriculum for High School Education Based on Similarity Analysis," *International Journal of Advances in Data and Information Systems*, vol. 4, no. 2, pp. 201–213, Nov. 2023, doi: 10.25008/ijadis.v4i2.1307.

[6] N. Thomas Rincy and R. Gupta, "A Survey on Machine Learning Approaches and Its Techniques:," in *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science, SCEECS 2020*, Institute of Electrical and Electronics Engineers Inc., Feb. 2020. doi: 10.1109/SCEECS48394.2020.190.

[7] A. H. Jiang *et al.*, "Accelerating Deep Learning by Focusing on the Biggest Losers," Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.00762

[8] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," Nov. 2015, [Online]. Available: http://arxiv.org/abs/1511.08458

[9] D. Choi, W. Bell, D. Kim, and J. Kim, "UAV-Driven Structural Crack Detection and Location Determination Using Convolutional Neural Networks.," *Sensors (Basel)*, vol. 21, no. 8, p. 2650, Apr. 2021, doi: 10.3390/s21082650.

[10] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters," Feb. 2017, [Online]. Available: http://arxiv.org/abs/1702.05373

[11] R. Kundu, H. Basak, P. K. Singh, A. Ahmadian, M. Ferrara, and R. Sarkar, "Fuzzy rank-based fusion of CNN models using Gompertz function for screening COVID-19 CT-scans," *Sci Rep*, vol. 11, no. 1, p. 14133, Jul. 2021, doi: 10.1038/s41598-021-93658-y.