

**IMPLEMENTASI *FASTER REGION CONVOLUTIONAL NEURAL  
NETWORK* UNTUK VERIFIKASI TANDA TANGAN DAN  
TINGKAT KEMIRIPAN BERBASIS *MOBILE***

**SKRIPSI**

**GILBERT SORAI ARO SIHURA**

**171402071**



**PROGRAM STUDI TEKNOLOGI INFORMASI  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA**

**MEDAN**

**2024**

IMPLEMENTASI *FASTER REGION CONVOLUTIONAL NEURAL  
NETWORK* UNTUK VERIFIKASI TANDA TANGAN DAN  
TINGKAT KEMIRIPAN BERBASIS *MOBILE*

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah Sarjana  
Teknologi Informasi

GILBERT SORAI ARO SIHURA

171402071



PROGRAM STUDI TEKNOLOGI INFORMASI  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2024

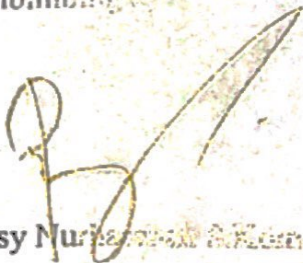
## PERSETUJUAN

Judul : Implementasi *Faster Region Convolutional Neural Network* untuk  
Verifikasi Tanda Tangan dan Tingkat Kemiripan Berbasis *Mobile*  
Kategori : Skripsi  
Nama : Gilbert Sorai Aro Sihura  
Nomor Induk Mahasiswa : 171402071  
Program Studi : Teknologi Informasi  
Fakultas : Ilmu Komputer dan Teknologi Informasi  
Universitas Sumatera Utara

Medan, 14 Juni 2024

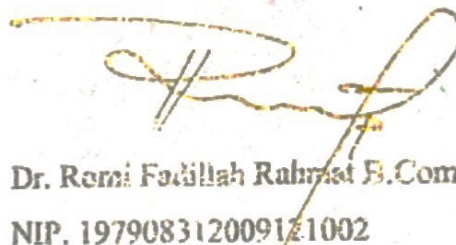
Komis Pembimbing:

Pembimbing 2,



Rossy Nurhasanah S.Kom., M.Kom.  
NIP. 196210262017042001

Pembimbing 1,



Dr. Renni Fadillah Rahmat B.Comp.Sc., M.Sc.  
NIP. 197908312009121002

Diketahui/disetujui oleh  
Program Studi S1 Teknologi Informasi  
Ketua,



Dedy Arisandy S.T., M. Kom.  
NIP. 197908312009121002

**PERNYATAAN**

IMPLEMENTASI *FASTER REGION CONVOLUTIONAL NEURAL*  
*NETWORK* UNTUK VERIFIKASI TANDA TANGAN DAN  
TINGKAT KEMIRIPAN BERBASIS *MOBILE*

**SKRIPSI**

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 14 Juni 2024

Gilbert Sorai Aro Sihura  
171402071

## UCAPAN TERIMA KASIH

Puji dan syukur penulis ucapkan kepada Tuhan Yang Maha Esa yang senantiasa memberikan berkat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi ini sebagai salah satu syarat untuk mendapat gelar Sarjana Komputer, pada Program Studi S1 Teknologi Informasi Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.

Dalam menyelesaikan penulisan skripsi ini, penulis telah banyak mendapatkan bimbingan, dukungan, bantuan serta doa dari berbagai pihak. Adapun pada kesempatan ini penulis ingin mengucapkan terimakasih kepada:

1. Keluarga penulis, Bapak Iradat Sihura dan Ibu Yurdame Wau selaku orang tua penulis dan juga Giovanni Natasha Sihura selaku adik penulis yang telah membesarkan dengan penuh kasih sayang dan juga dukungan kepada penulis.
2. Keluarga besar penulis yang telah banyak membantu dengan berbagai cara hingga dapat menyelesaikan penulisan ini.
3. Bapak Romi Fadillah Rahmat B.Comp.Sc., M.Sc. selaku dosen pembimbing 1 dan juga Ibu Rossy Nurhasanah S.Kom., M.Kom selaku dosen pembimbing 2, yang telah membimbing, mengarahkan serta memberikan kritik dan saran yang sangat membantu dalam menyelesaikan penulisan.
4. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. selaku Dekan Fasilkom-TI Universitas Sumatera Utara.
5. Bapak Dedi Arisandi ST., M.Kom. selaku Ketua Program Studi S1 Teknologi Informasi Universitas Sumatera Utara.
6. Ibu Dr. Erna Budhiarti Nababan M.IT selaku dosen pembimbing 1 dan juga Bapak Ivan Jaya S.Si., M.Kom. selaku dosen pembimbing 2.
7. Seluruh Dosen Program Studi S1 Teknologi Informasi yang telah memberikan ilmu yang bermanfaat bagi penulis selama masa perkuliahan.
8. Staff dan pegawai Fasilkom-TI Universitas Sumatera Utara yang membantu segala urusan administrasi dalam menyelesaikan skripsi.
9. Teman - teman dekat penulis yaitu M Rizki Fatihah, M Rafif Rasyidi, Ibnu Maulana, M Fajar Harahap, Dinul Iman, Tongku Malim Noor Hadi, Yusuf

Fathurrahman, Jackie Chandra, Farras Polem, Deo Silitonga, Ade Rizky, Muhammad Ulwan, Taufik Baskoro, Talitha Rayhan, Bella Olivia, Widya Anggi yang telah membantu, mendukung dan memberikan dukungan dalam masa perkuliahan.

10. Teman-teman angkatan 2017 Teknologi Informasi yang telah membantu dan berjuang bersama penulis dalam menghadapi perkuliahan.

Penulis menyadari bahwa dalam penulisan skripsi ini masih terdapat kekurangan, oleh karena itu penulis mengharapkan kritik dan saran yang membangun untuk penyempurnaan skripsi ini.

Medan, 14 Juni 2024

Penulis

IMPLEMENTASI FASTER REGION CONVOLUTIONAL NEURAL  
NETWORK UNTUK VERIFIKASI TANDA TANGAN DAN  
TINGKAT KEMIRIPAN BERBASIS *MOBILE*

**ABSTRAK**

Tanda tangan telah menjadi salah satu metode otentikasi yang penting dalam berbagai aplikasi keamanan dan identifikasi. Namun, tantangan muncul dalam memverifikasi tanda tangan dengan cepat dan akurat, terutama dalam lingkungan *mobile*. Dalam upaya mengatasi kendala ini, penelitian ini mengusulkan *implementasi Faster Region Convolutional Neural Network (Faster R-CNN)* untuk verifikasi tanda tangan dan penilaian tingkat kemiripan, khususnya dalam *platform mobile*. Metode yang diusulkan menggabungkan teknologi *Deep Learning* dengan kemampuan deteksi objek tingkat tinggi *Faster R-CNN* untuk mengenali tanda tangan dalam gambar. Selanjutnya, model ini memanfaatkan teknik pemrosesan citra dan pembelajaran mendalam untuk mengekstrak fitur dan menganalisis kemiripan antara tanda tangan yang diuji dengan data referensi. Implementasi ini dirancang agar dapat berjalan secara efisien pada perangkat *mobile*, mengoptimalkan penggunaan sumber daya dengan tetap mempertahankan tingkat akurasi yang tinggi. Evaluasi kinerja dilakukan menggunakan dataset tanda tangan yang luas, termasuk variasi gaya tulisan dan kondisi pencahayaan yang berbeda. Sebelum proses verifikasi, data citra tanda tangan akan terlebih dahulu melalui proses *pre-processing* yang meliputi *labeling*, *resizing*, *grayscale*, *thresholding* dan dilanjutkan dengan *feature extraction* menggunakan *canny edge detection*. Berdasarkan hasil pengujian, sistem mampu melakukan verifikasi tanda tangan dengan memperoleh nilai *accuracy* 90%.

**Kata Kunci:** Tanda Tangan Digital, Deteksi Objek, Pengolahan Citra, *Faster Region Convolution Neural Network*, Perangkat *Mobile*.

*IMPLEMENTATION OF FASTER REGION CONVOLUTIONAL NEURAL NETWORK  
FOR MOBILE – BASED SIGNATURE VERIFICATION AND LEVELS OF  
SIMILIARITY*

**ABSTRACT**

*Signatures have become one of the important authentication methods in various security and identification applications. However, challenges arise in verifying signatures quickly and accurately, especially in mobile environments. In an effort to overcome these obstacles, this research proposes the implementation of Faster Region Convolutional Neural Network (Faster R-CNN) for signature verification and similarity assessment, especially in mobile platforms. The proposed method combines Deep Learning technology with the high-level object detection capability of Faster R-CNN to recognize signatures in images. Furthermore, it leverages image processing and deep learning techniques to extract features and analyze the similarity between the tested signature and reference data. The implementation is designed to run efficiently on mobile devices, optimizing resource usage while maintaining a high level of accuracy. Performance evaluation was conducted using an extensive signature dataset, including a variety of writing styles and different lighting conditions. Before the verification process, the signature image data will first go through a pre-processing process which includes labeling, resizing, grayscaling, thresholding and continued with feature extraction using canny edge detection. Based on the test results, the system is able to verify signatures with an accuracy rate of 90%*

**Keywords:** *Digital Signature, Object Detection, Image Processing, Faster Region Convolution Neural Network, Mobile Device*



## DAFTAR ISI

<b>PERSETUJUAN</b>	<b>ii</b>
<b>PERNYATAAN</b>	<b>iii</b>
<b>UCAPAN TERIMA KASIH</b>	<b>iv</b>
<b>ABSTRAK</b>	<b>vi</b>
<b>ABSTRACT</b>	<b>vii</b>
<b>DAFTAR ISI</b>	<b>viii</b>
<b>DAFTAR TABEL</b>	<b>xi</b>
<b>DAFTAR GAMBAR</b>	<b>xii</b>
<b>BAB 1 PENDAHULUAN</b>	<b>1</b>
1.1. Latar Belakang	1
1.2. Rumusan Masalah	3
1.3. Batasan Masalah	3
1.4. Tujuan Penelitian	3
1.5. Manfaat Penelitian	3
1.6. Metodologi Penelitian	4
<b>BAB 2 LANDASAN TEORI</b>	<b>5</b>
2.1. Tanda Tangan Digital	5
2.2. Pengolahan Citra Digital ( <i>Image Processing</i> )	7
2.3. Citra Warna	8
2.4. Citra Grayscale (Skala Keabuan)	9
2.5. <i>Convolutional Neural Network</i>	9
2.6. R-CNN ( <i>Regional Convolution Neural Network</i> )	11

2.7. Fast R-CNN	12
2.8. Faster R-CNN	13
2.9. Canny	14
2.10. Aplikasi Mobile	15
2.11. React Native	15
2.12. Python	16
2.13. Tensorflow	16
2.14. Keras	16
2.15. Google Colab	16
2.16. Ngrok	17
2.17. Expo SDK	17
2.18. Metode Perhitungan Performa Machine Learning	17
2.19. Penelitian Terdahulu	20
2.20. Perbedaan Penelitian Terdahulu	23
<b>BAB 3 ANALISIS DAN PERANCANGAN SISTEM</b>	<b>25</b>
3.1. Datas yang Digunakan	25
3.2. Arsitektur Umum	28
3.2.1. Image Acquisition	30
3.2.2. Image Pre-Processing	31
3.2.3. Feature Extraction	35
3.2.4. Image Classifier	36
3.3. Perancangan Antarmuka Sistem	38
3.3.1. Rancangan Tampilan Halaman Home	39
3.3.2. Rancangan Tampilan Halaman Testing	39
3.3.3. Rancangan Tampilan Halaman Result	40
<b>BAB 4 IMPLEMENTASI DAN PENGUJIAN</b>	<b>41</b>
4.1. Implementasi Sistem	41
4.1.1. Spesifikasi perangkat keras dan perangkat lunak	41

4.1.2. Implementasi data	41
4.1.3. Implementasi model	43
4.1.4. Implementasi perancangan antarmuka	43
4.2. Prosedur Operasional	46
4.3. Pengujian Sistem	48
<b>BAB 5 KESIMPULAN DAN SARAN</b>	<b>54</b>
5.1 Kesimpulan	54
5.2 Saran	54
<b>DAFTAR PUSTAKA</b>	<b>56</b>

**DAFTAR TABEL**

Tabel 2.1 <i>Confussion Matrix</i>	18
Tabel 2.2 Penelitian Terdahulu	21
Tabel 3.1 Jumlah Data Latih dan Uji	26
Tabel 4.1 Hasil Implementasi Model 1200 data	43
Tabel 4.2 Sampel Hasil Pengujian <i>Faster R-CNN</i>	48
Tabel 4.3 Sampel Hasil <i>Confussion Matrix</i>	51
Tabel 4.4 <i>Classification Report</i>	53

## DAFTAR GAMBAR

Gambar 2.1 Tanda Tangan Digital	7
Gambar 2.2 Citra Warna	8
Gambar 2.3 Citra Grayscale	9
Gambar 2.4 Convolutioanal Neural Network	10
Gambar 2.5 R-CNN	11
Gambar 2.6 Fast R-CNN	12
Gambar 2.7 Faster R-CNN	13
Gambar 3.1 Contoh Data Citra Tanda Tangan	25
Gambar 3.2 Arsitektur Umum	29
Gambar 3.3 Citra Sebelum dan Sesudah <i>Labelling</i>	31
Gambar 3.4 Citra Hasil <i>Grayscale</i>	33
Gambar 3.5 Gambar hasil proses <i>testing</i>	38
Gambar 3.6 Rancangan Tampilan <i>Home</i>	39
Gambar 3.7 Rancangan Tampilan <i>Testing</i>	39
Gambar 3.8 Rancangan Tampilan Result.	40
Gambar 4.1 Data Citra Tanda Tangan 1	42
Gambar 4.2 Data Citra Tanda Tangan 2	42
Gambar 4.3 Data Citra Tanda Tangan 3	42
Gambar 4.4 Tampilan Halaman Home	44
Gambar 4.5 Tampilan Halaman Kamera Sebelum Diambil	44
Gambar 4.6 Tampilan Halaman Kamera Setelah Diambil	45
Gambar 4.7 Tampilan Halaman Hasil	45
Gambar 4.8 Tampilan Halaman Home	46
Gambar 4.9 Tampilan Halaman Kamera	46
Gambar 4.10 Tampilan Halaman Kamera Sesudah Diambil	47
Gambar 4.11 Tampilan Halaman Hasil	47

# **BAB 1**

## **PENDAHULUAN**

### **1.1. Latar Belakang**

Perkembangan teknologi yang pesat telah membuat keamanan menjadi suatu kebutuhan esensial bagi manusia, baik dalam upaya menjaga keamanan pribadi maupun melibatkan segala aspek yang memengaruhi kehidupannya, termasuk dokumen-dokumen legal. Salah satu pendekatan keamanan yang umum digunakan adalah implementasi sistem pengenalan identitas, di mana salah satu bentuk identifikasi melibatkan penggunaan sistem biometrik. Ada dua kategori biometrik utama, yakni biometrik fisik dan biometrik perilaku. Salah satu contoh dari biometrik perilaku adalah penggunaan tanda tangan (Arifin & Naf'an, 2017).

Tanda tangan umumnya digunakan dalam berbagai jenis dokumen kegiatan sebagai langkah untuk mengidentifikasi dan mengesahkan keaslian dokumen tersebut. Contoh paling umum dari penggunaan tanda tangan terdapat dalam kegiatan administratif di mana tanda tangan seringkali menjadi syarat pada berbagai dokumen. Pengembangan sistem identifikasi biometrik memiliki dua tujuan utama, yaitu identifikasi dan verifikasi. Sistem biometrik dapat dibuat berdasarkan ciri-ciri fisik dan perilaku individu. Ciri-ciri fisik seperti wajah, sidik jari, dan iris mata dianggap sebagai karakteristik unik setiap individu, sementara ciri-ciri perilaku seperti suara, gaya berjalan, dan tanda tangan juga menjadi bagian dari identifikasi biometrik (Novandra et al., 2018).

Verifikasi tanda tangan dapat dibagi menjadi dua jenis, yaitu online dan offline (Widodo & Harjoko, 2015). Verifikasi tanda tangan secara online melibatkan suatu proses di mana penandatangan menggunakan pena atau stylus khusus untuk melakukan tanda tangan. Di sisi lain, sistem verifikasi offline beroperasi dengan memproses gambar tanda tangan yang diperoleh dari pemindai atau kamera digital.

Verifikasi tanda tangan secara offline bergantung pada perbandingan gambar statis dua dimensi, terutama dalam memeriksa fitur statis seperti tekstur tanda tangan, rasio aspek bentuk dan ukuran, posisi goresan, dan perbedaan dalam tulisan tangan. Tanda tangan tersebut kemudian dianalisis dengan membandingkan tanda tangan yang mencurigakan dengan tanda tangan asli, tanpa mencari persis kesamaan. Tanda tangan asli dapat bervariasi tergantung pada faktor-faktor seperti suasana hati penandatangan, keadaan emosional, usia, kondisi saat menandatangani, dan beberapa faktor lainnya.

Dalam penelitian Gideon et al. (2018), dilakukan deteksi tanda tangan dengan hasil yang baik menggunakan metode *Convolutional Neural Network* (CNN) dimana terdapat 3 pembagian rasio data yang dilakukan, pembagian rasio data 8:2 mendapatkan hasil yang terbaik dengan 98 % *training* dan *validation accuracy*, sedangkan untuk rasio data 7:3 mendapatkan hasil 97 % *training accuracy* dan 94 % *validation accuracy*, dan yang terakhir rasio data 6:4 mendapatkan 97 % *training accuracy* dan 96 % *validation accuracy*.

Pada penelitian yang dilakukan oleh Pansare et al. (2012) metode yang digunakan pada verifikasi tanda tangan adalah *neural network*. Dimana pada metode tersebut digunakan *feature extracted* dari citra tanda tangan yang telah diproses sebelumnya, *extracted feature* digunakan untuk melatih *neural network* menggunakan algoritma *error back propagation training*. Secara keseluruhan, *correct classification rate* dari sistem adalah 82,66 %.

Selain itu, terdapat penelitian yang dilakukan oleh Drott et al. (2015) menggunakan *Machine Learning Technique* dengan pendekatan *Deep Learning* untuk melakukan verifikasi tanda tangan *online*. Adapun hasil dari model akhir mencatatkan *true positive rate (sensitivity)* 96,7 % dan *false positive rate* 0,6 %.

Dengan merujuk pada latar belakang dan penelitian-penelitian terdahulu tentang proses verifikasi tanda tangan, penulis melakukan penelitian dengan judul "IMPLEMENTASI *FASTER REGION CONVOLUTIONAL NEURAL NETWORK* UNTUK VERIFIKASI TANDA TANGAN DAN PENILAIAN TINGKAT KEMIRIPAN BERBASIS *MOBILE*".

## **1.2. Rumusan Masalah**

Karena banyaknya tanda tangan palsu, hal ini menimbulkan keraguan terhadap keabsahan suatu dokumen atau transaksi. Begitu pula dengan kemajuan teknologi saat ini, dibutuhkan suatu aplikasi yang membantu memverifikasi jenis tanda tangan dan kemiripannya pada proses pengambilan citra.

## **1.3. Batasan Masalah**

Ada beberapa keterbatasan yang membuat ruang lingkup permasalahan yang dibahas dalam penelitian ini tidak dapat diperluas, yaitu:

1. Jumlah tanda tangan yang digunakan pada penelitian ini sebanyak 500 gambar tanda tangan.
2. Output penelitian yang dihasilkan hanya terbatas deteksi jenis tanda tangan dan tingkat kemiripan tanda tangan.
3. Metode penelitian yang digunakan adalah *Faster Region Convolutional Neural Network*.
4. Pengambilan citra tanda tangan dilakukan secara digital.

## **1.4. Tujuan Penelitian**

Tujuan dari penelitian ini adalah untuk mengimplementasikan metode *Faster Region Convolutional Networks* pada perangkat mobile untuk melakukan verifikasi tanda tangan dan tingkat kemiripannya.

## **1.5. Manfaat Penelitian**

Manfaat penelitian ini adalah memberikan kontribusi penting dengan mengevaluasi keakuratan algoritma Faster R-CNN dalam verifikasi tanda tangan, serta menawarkan solusi untuk otomatisasi deteksi tanda tangan dan pengukuran tingkat kemiripan secara efisien. Tidak hanya berguna untuk aplikasi praktis dalam bidang keamanan, tetapi juga mendorong pengembangan lebih lanjut dalam teknologi deteksi objek dan analisis citra, membuka potensi untuk inovasi dan peningkatan sistem yang lebih baik di masa depan.



## 1.6. Metodologi Penelitian

Metode penelitian yang akan dilakukan dalam penelitian ini meliputi tahapan-tahapan sebagai berikut:

1. Studi Literatur

Pada tahap ini, penulis meneliti dan memahami sejumlah literatur dan dokumen yang mendukung pelaksanaan penelitian. Beberapa literatur dan dokumen tambahan tersedia dalam bentuk buku, jurnal dan skripsi atau sumber lainnya.

2. Pengumpulan Data

Setelah meneliti informasi dan belajar dari sumber tambahan, penulis mulai mengumpulkan data untuk digunakan dalam penelitian.

3. Analisis Permasalahan

Kemudian, penulis menganalisis informasi yang dikumpulkan untuk memahami metode yang digunakan agar sesuai untuk memecahkan masalah penelitian ini.

4. Perancangan Sistem

Selanjutnya, setelah menganalisis informasi yang telah diperoleh sebelumnya, penulis mulai merancang sistem untuk penelitian.

5. Implementasi

Ditahap ini penulis melakukan pengujian dan evaluasi pada metode *Faster R-CNN* untuk penerapannya.

6. Penyusunan Laporan

Langkah terakhir, penulis menulis laporan berupa makalah penelitian yang telah dilakukan untuk menyajikan hasil penelitian yang dilakukan.

## **BAB 2**

### **LANDASAN TEORI**

#### **2.1. Tanda Tangan Digital**

Tanda tangan digital adalah salah satu inovasi di era *digital* yang menggantikan fungsi tanda tangan tradisional dalam memvalidasi dokumen elektronik. Konsep dasarnya melibatkan penggunaan teknologi kriptografi, di mana tanda tangan digital terdiri dari data digital unik yang terkait dengan identitas penandatangan dan konten dokumen. Hal ini dimungkinkan oleh pasangan kunci kriptografi, publik dan privat, di mana kunci privat digunakan untuk membuat tanda tangan digital, sedangkan kunci publik digunakan untuk memverifikasinya. *Public Key Infrastructure* (PKI) mendukung proses ini dengan menyediakan sertifikat digital yang memverifikasi identitas penandatangan dan kunci publik. Namun, terlepas dari manfaat signifikan tanda tangan digital dalam meningkatkan kecepatan, efisiensi, dan keamanan di berbagai industri, terdapat tantangan yang harus diatasi, seperti masalah keamanan yang terkait dengan penyimpanan dan pengelolaan *private key*, serta keharusan untuk mematuhi hukum dan peraturan yang terkait dengan penggunaannya. Dengan pemahaman yang komprehensif tentang konsep, teknologi, peraturan, dan tantangan yang terlibat dalam tanda tangan digital, penelitian ini bertujuan untuk memberikan wawasan dan kontribusi yang berharga di bidang ini (Chen, 2019).

Tanda tangan digital merupakan bentuk tanda tangan yang dihasilkan secara elektronik dan memiliki fungsi serupa dengan tanda tangan pada dokumen kertas. Keunggulan tanda tangan digital terletak pada tingkat keamanan yang lebih tinggi dibandingkan dengan tanda tangan konvensional. Tanda tangan digital dapat memberikan jaminan terhadap keaslian suatu dokumen digital, mencakup identitas individu dan keabsahan dokumen tersebut. Keamanan yang diberikan oleh tanda tangan

digital melibatkan aspek-aspek seperti kerahasiaan data, keaslian data, keutuhan data, dan ketidakbisaan untuk menyangkal keberadaannya (Novel, 2018).

Dalam konteks penerapan tanda tangan digital, penggunaannya meluas di berbagai sektor seperti perbankan, perusahaan teknologi, hukum, dan pemerintahan. Implementasi tanda tangan digital telah membawa manfaat signifikan dalam mempercepat proses bisnis, mengurangi biaya administrasi, dan meningkatkan keamanan dalam pertukaran dokumen elektronik. Namun, seperti halnya dengan setiap teknologi, terdapat tantangan yang muncul seiring dengan penyebaran tanda tangan digital, termasuk kebutuhan untuk memastikan interoperabilitas antara platform yang berbeda, meningkatkan kesadaran akan keamanan digital di kalangan pengguna, dan mengembangkan solusi yang lebih baik untuk mengatasi ancaman keamanan yang terus berkembang. Dengan demikian, penelitian dan pengembangan terus berlanjut untuk meningkatkan efektivitas, keamanan, dan adopsi tanda tangan digital dalam ekosistem digital yang semakin kompleks dan terhubung ini (Wang, 2017).

Selain itu, perkembangan teknologi seperti blockchain telah menjadi sorotan utama dalam konteks tanda tangan digital. Dengan memanfaatkan konsep desentralisasi dan keamanan yang kuat yang ditawarkan oleh teknologi blockchain, banyak penelitian telah dilakukan untuk mengintegrasikan tanda tangan digital dengan platform blockchain. Pendekatan ini tidak hanya menawarkan tingkat keamanan yang lebih tinggi melalui distribusi yang terdistribusi dan konsensus jaringan, tetapi juga meningkatkan tingkat transparansi dan kepercayaan dalam pertukaran dokumen digital. Namun, ada juga tantangan yang perlu diatasi, termasuk keterbatasan skala, efisiensi, dan masalah hukum yang berkaitan dengan penggunaan blockchain untuk tanda tangan digital. Dengan demikian, penelitian dan eksperimen terus dilakukan untuk mengembangkan solusi yang lebih baik dan lebih terukur dalam penggabungan teknologi tanda tangan digital dengan platform blockchain guna meningkatkan keamanan, keandalan, dan efisiensi dalam pertukaran informasi digital di masa mendatang (Zheng, 2017).

Selain integrasi dengan teknologi blockchain, terdapat juga penelitian yang menggali potensi tanda tangan digital dalam konteks artificial intelligence dan machine learning. Dalam beberapa penelitian, teknik machine learning telah diterapkan untuk meningkatkan keamanan dan otentikasi tanda tangan digital dengan mengidentifikasi pola unik dari gaya tulisan tangan digital.

Pendekatan ini memanfaatkan kemampuan artificial intelligence untuk mengenali dan membedakan pola tulisan tangan yang sah dari yang palsu, dengan demikian meningkatkan keandalan proses verifikasi tanda tangan digital. Namun, ada juga tantangan yang muncul, seperti kebutuhan akan dataset yang besar dan bervariasi untuk melatih model *machine learning* dengan akurasi yang tinggi, serta masalah privasi yang berkaitan dengan penggunaan data tulisan tangan. Oleh karena itu, pengembangan teknologi tanda tangan digital yang terintegrasi dengan kecerdasan buatan membutuhkan pendekatan holistik yang mempertimbangkan aspek teknis, keamanan, dan etika yang terlibat. Dengan demikian, penelitian ini menawarkan prospek yang menarik untuk meningkatkan keamanan dan efisiensi dalam penggunaan tanda tangan digital di era digital yang terus berkembang (Rattani, 2018).



**Gambar 2.1** Tanda Tangan Digital (<https://pintu.co.id>)

## **2.2. Pengolahan Citra Digital (*Image Processing*)**

Pemrosesan citra digital merujuk pada manipulasi gambar dua dimensi menggunakan komputer. Tujuan utama dari pemrosesan citra adalah untuk mempermudah interpretasi oleh manusia dan mesin dengan mengubah suatu citra menjadi citra lainnya (Gusa, 2013).

*Image processing* merupakan cabang ilmu yang berkaitan dengan analisis dan manipulasi gambar atau citra digital untuk memperbaiki kualitasnya, mengekstrak informasi, dan menghasilkan output yang bermakna. Konsep dasar dalam *image*

*processing* meliputi penggunaan algoritma dan teknik untuk memanipulasi citra digital, yang sering kali melibatkan operasi matematis dan statistik.

Tahapan umum dalam pemrosesan citra meliputi *pre-processing*, *core processing*, dan *post-processing*. *Pre-processing* melibatkan operasi seperti penajaman, penormalan, dan penyesuaian kontras untuk mempersiapkan citra untuk analisis lebih lanjut. *Core processing* adalah tahap di mana analisis atau transformasi utama dilakukan, seperti deteksi tepi, segmentasi, pengenalan pola, atau pemrosesan filtrasi untuk menghilangkan noise atau meningkatkan fitur. *Post-processing* melibatkan tahapan seperti pemrosesan berbasis objek, interpretasi hasil, dan penggabungan citra untuk mendapatkan informasi yang lebih bervariasi. Teknik yang umum digunakan dalam image processing mencakup *Fourier transformation*, operasi morfologi, deteksi tepi, segmentasi berbasis warna atau tekstur, serta teknik *deep learning* seperti *convolutional neural networks* (CNN) untuk tugas pengenalan dan klasifikasi gambar. Dalam aplikasinya, *image processing* digunakan dalam berbagai bidang, termasuk *computer vision*, pengolahan citra medis, pengenalan pola, pengolahan citra satelit, dan *computer graphics*. Dengan kemajuan teknologi dan perkembangan algoritma, *image processing* terus menjadi bidang yang beragam dan berkembang pesat dengan berbagai aplikasi yang luas dan potensial dalam berbagai industri dan disiplin ilmu (Gonzalez, 2017).

### 2.3. Citra Warna

Citra berwarna sering juga disebut dengan istilah RGB atau citra true color. Citra ini memiliki kemampuan untuk mendefinisikan warna objek dengan sangat mendekati warna aslinya. Hal ini dicapai dengan menggabungkan tiga warna dasar, yaitu red (R), green (G), dan blue (B). Contoh citra berwarna dapat dilihat pada Gambar 2.2.



**Gambar 2.2** Citra Warna (<https://wall.alphacoders.com>)

## 2.4. Citra Grayscale (Skala Keabuan)

Citra skala abu-abu merupakan citra digital di mana setiap pikselnya memiliki warna yang bervariasi dari putih hingga hitam. Setiap piksel dalam citra skala abu-abu dapat diwakili oleh 1 bit atau 8 bit, tergantung pada tingkat kedalaman bit yang digunakan. Citra ini sering digunakan dalam berbagai aplikasi, dan salah satu contoh penerapannya terdapat dalam bidang medis, khususnya dalam citra sinar X (*x-ray*). Berikut ini adalah contoh citra skala abu-abu.

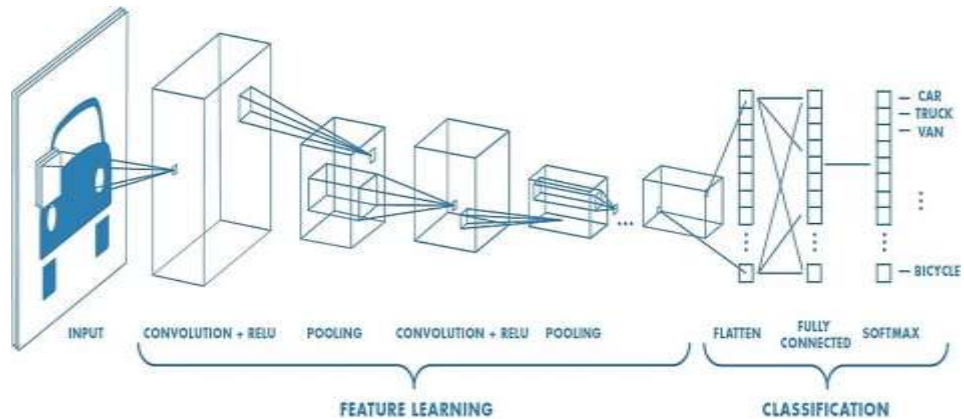


**Gambar 2.3** Citra Grayscale (<http://www.brianrrichards.com>)

## 2.5. *Convolutional Neural Network*

*Convolutional Neural Network* (CNN) merupakan metode dalam jaringan saraf yang bermanfaat untuk mendeteksi objek, menganalisis gambar, dan mengenali objek dalam suatu gambar. CNN melibatkan berbagai parameter untuk mengkarakterisasi dan mengklasifikasikan objek menggunakan kecerdasan buatan. Arsitektur CNN terdiri dari tiga jenis lapisan, yaitu lapisan konvolusi, lapisan *pooling*, dan lapisan *fully-connected*. Lapisan konvolusi menjadi bagian sentral dalam CNN, memiliki koneksi lokal dan bobot spesifik yang sama. Fungsi utama dari lapisan ini adalah memahami cara memetakan suatu fungsi dari masukan yang diberikan. Fitur yang diekstraksi diteruskan melalui lapisan ini ke lapisan pemrosesan berikutnya untuk mengekstrak fitur yang lebih kompleks (Bui & Chang, 2016). Proses konvolusi merupakan konsep matematikayang menerapkan fungsi berulang-ulang untuk mengekstrak fitur dari gambar yang diinputkan. Bobot

pada lapisan ini menggunakan spesifikasi kernel konvolusi untuk melatih kernel dengan data masukan dari CNN.



**Gambar 2.4** Convolutioanal Neural Network (Yitong Shao, *et al.* 2018)

*Convolutional layer* dan *pooling layer* merupakan dua komponen dari layer ekstraksi fitur. Sebagaimana terlihat pada Gambar 2.4, *convolutional layer* terdiri dari serangkaian neuron yang membentuk piksel. Sebagai contoh, lapisan utama dapat berupa lapisan konvolusi dengan ukuran  $5 \times 5 \times 3$ , yang artinya memiliki panjang 5 piksel, tinggi 5 piksel, dan ketebalan 3 piksel. Filter - filter tersebut bergerak dan melakukan operasi "dot" pada gambar secara berulang, menghasilkan keluaran berupa *feature map*.

Terdapat parameter yang dapat mengatur jumlah pergeseran filter, atau yang sering disebut sebagai *stride*. Selain itu, terdapat juga parameter yang menentukan jumlah piksel dengan nilai 0 yang harus diisi di kedua sisi gambar masukan. Hal ini memungkinkan manipulasi terhadap lapisan konvolusional atau peta fitur dalam ukuran hasil. Persamaan untuk menghitung dimensi peta fitur target adalah:

$$Output = \frac{W - N + 2P}{S} + 1 \quad (2.1)$$

Keterangan:

W = Tinggi/Panjang Input

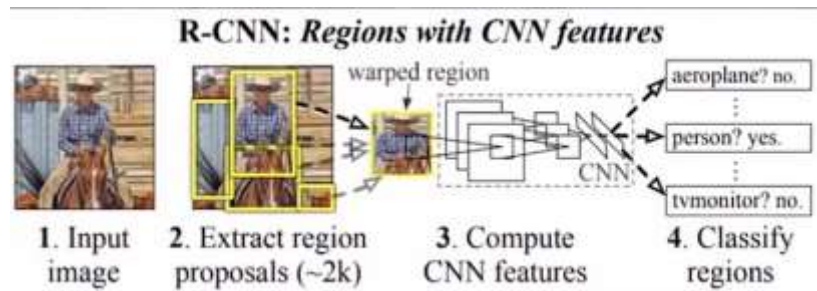
N = Tinggi/Panjang Filter

P = *Zero Padding*

S = *Stride*

## 2.6. R-CNN (*Regional Convolutional Neural Network*)

R-CNN (*Region-based Convolutional Neural Network*) awalnya diperkenalkan pada tahun 2014 oleh peneliti dari *UC Berkeley*, *Ross Girshick*, untuk mengatasi masalah teknik penentuan wilayah (*regions*). Tahapan pengolahan R-CNN dapat dilihat pada Gambar 2.5.



**Gambar 2.5** R-CNN (Girshick, et al. 2014)

- Melakukan pencarian bagian gambar yang mungkin berisi objek.
- Dengan menggunakan metode proposal wilayah (*region proposal*).
- Mengolah hasil dari wilayah - wilayah tersebut menjadi masukan untuk *Convolutional Neural Network* (CNN) sebagai *extractor* fitur.
- Selain itu, fitur - fitur yang dihasilkan akan digunakan sebagai masukan untuk *Support Vector Machine* (SVM) untuk
- Menghasilkan kotak pembatas (*bounding box*)

Algoritma yang digunakan dalam pencarian selektif adalah menemukan dua area yang paling mirip dan menggabungkannya. Persamaan kesamaan ( $S$ ) antara area (a) dan (b) didefinisikan sebagai berikut:

$$S(a, b) = S_{texture}(a, b) + S_{size}(a, b) \quad (2.2)$$

Keterangan:

$S_{texture}(a, b)$  = Mengukur kesamaan visual

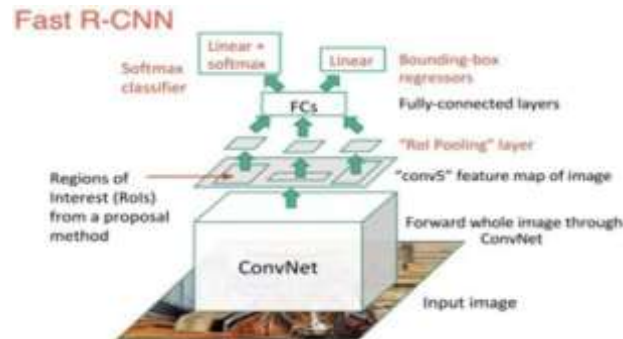
$S_{size}(a, b)$  = Penggabungan wilayah yang kecil secara bersamaan untuk menghindari area yang lain



## 2.7. Fast R-CNN

Pada tahun 2015, Ross Gershick melakukan upgrade pada R-CNN menjadi Fast R-CNN, yang menunjukkan beberapa perbedaan dari versi sebelumnya dan mengatasi beberapa kendala dalam proses pelatihan R-CNN. Langkah-langkah kerja dari metode Fast R-CNN adalah setiap wilayah (*region*) yang dihasilkan oleh *Region Proposal Network* (RPN) memiliki fungsi ekstraksi dari *Convolutional Neural Network* (CNN) masing-masing. Dalam Fast R-CNN, hanya terdapat satu CNN yang digunakan. Hasil *feature map* dari CNN tersebut digabungkan dengan *region of interest* (ROI) untuk proses selanjutnya, yaitu klasifikasi kelas.

Fast R-CNN menggantikan fungsi *Support Vector Machine* (SVM) sebagai pengklasifikasi dengan menggunakan *ROI pooling* dan *fully-connected layers*. Pendekatan ini dilakukan pada CNN, *feed forward network*, dan *ROI pooling layer*. Hal ini berfungsi sebagai penambahan kapabilitas R-CNN menjadi *end-to-end differentiable*, memudahkan proses pelatihan dan tidak hanya meningkatkan performa R-CNN, sebagaimana yang ditunjukkan pada Gambar 2.6.



**Gambar 2.6** Fast R-CNN (Kaul S, 2018)

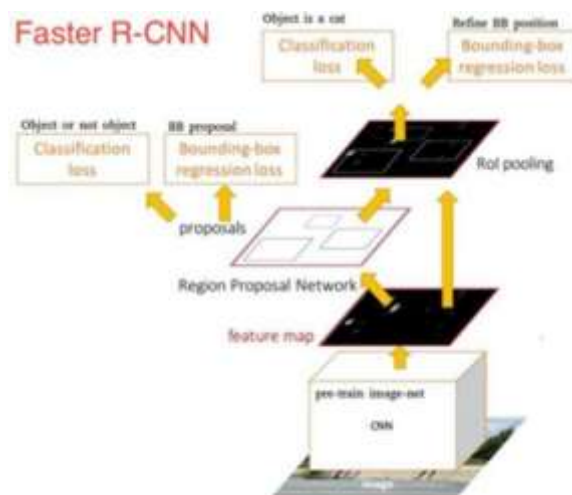
Gambar 2.6. Menunjukkan bahwa Fast R-CNN membutuhkan waktu yang lebih cepat dibandingkan dengan R-CNN. Hal ini disebabkan oleh fakta bahwa dalam proses pelatihan, Fast R-CNN tidak perlu memasukkan 2000 proposal region ke jaringan neural konvolusional setiap kali. Sebaliknya, operasi konvolusi hanya dilakukan satu kali per gambar, menghasilkan satu feature map yang dapat digunakan untuk semua proposal region. Hal ini mengurangi redundansi dan meningkatkan efisiensi proses pelatihan.

## 2.8. Faster R-CNN

Metode dalam *deep learning* yang digunakan untuk pengenalan objek pada gambar adalah *Faster Regional Convolutional Neural Network* (Faster R-CNN). Algoritma R-CNN dan Fast R-CNN menggunakan pencarian selektif untuk menemukan proposal wilayah, yang dapat menjadi proses yang memakan waktu dan mempengaruhi kinerja sistem. Untuk mengatasi ini, muncul algoritma deteksi objek yang dapat mengeliminasi langkah pencarian selektif, memungkinkan jaringan untuk mempelajari proposal wilayah secara otomatis.

Metode ini mirip dengan Fast R-CNN, di mana citra dijadikan input ke jaringan konvolusional untuk menghasilkan peta fitur konvolusional. Proposal wilayah yang diprediksi kemudian direkonstruksi menggunakan lapisan penggabungan RoI (*Region of Interest*) yang digunakan untuk mengklasifikasikan gambar dalam wilayah yang diusulkan dan memprediksi nilai offset pada *bounding box*.

Faster R-CNN memperkenalkan metode *Region Proposal Network* (RPN) yang dapat mendeteksi berbagai ukuran dan objek dengan kecepatan tinggi, yaitu sekitar 0,2 detik per gambar. Struktur arsitektur Faster R-CNN dapat dilihat pada Gambar 2.7.



**Gambar 2.7** Faster R-CNN (Ren, et al. 2017)

Deteksi objek dilakukan dengan melacak sifat-sifat objek pada citra. Proses pencarian ini melibatkan serangkaian lapisan, yang dalam konteks jaringan saraf disebut sebagai proses konvolusi, atau secara umum dikenal sebagai *Convolutional Neural Network* (CNN). Faster R-CNN menggunakan dua arsitektur CNN, yaitu ZF Net dan

VGG-16, yang akan mengalami proses konvolusi hingga mencapai 13 lapisan, menghasilkan *feature map*. *Feature map* ini akan digunakan dalam tahap berikutnya untuk melakukan deteksi dan klasifikasi objek pada citra.

Setelah mendapatkan *feature map*, tahap selanjutnya adalah masuk ke tahap *RegionProposal Network* (RPN). Faster R-CNN memperkenalkan RPN yang terdiri dari 9 anchor dan berfungsi untuk menilai apakah suatu wilayah mengandung objek atau tidak. *Bounding box* dihasilkan oleh RPN dan memiliki dua skor pada setiap bounding box sebagai indikator keberadaan objek dalam wilayah tersebut atau tidak. Metode RPN dalam Faster R-CNN memungkinkan pendeteksian objek dengan berbagai ukuran dan memproses gambar dengan lebih cepat.

Persamaan untuk menghitung fungsi kerugian (loss function) dari Faster R-CNN adalah:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*). \quad (2.3)$$

Keterangan:

$P_i$  = Prediksi *anchor* pada gambar, dan persamaan *ground-truth* label

$$P_i^* = \begin{cases} 0 & \text{negative label} \\ 1 & \text{positive label} \end{cases}$$

$t_*$  =  $8px, py, pw, p-9$  adalah representasi vector dari 4 parameter pada bounding box dimana didalamnya ada sebuah objek yang terprediksi

## 2.9. Canny

*Canny Edge Detector* merupakan algoritma deteksi tepi yang dikembangkan oleh John F. Canny pada tahun 1986. Dalam penjelasannya, Canny menyajikan tiga kriteria evaluasi untuk detektor tepi, yakni *Signal-to-Noise Ratio* (SNR), presisi lokal, dan respon tepi tunggal. Berdasarkan kriteria-kriteria tersebut, Canny menyimpulkan teknik deteksi terbaik (Canny, 1986). Metode yang digunakan oleh Canny untuk mendeteksi tepi pada suatu gambar melibatkan serangkaian langkah, termasuk penerapan filter Gaussian, perhitungan gradien, *non-maximum suppression*, *double thresholding*, dan *edge tracking by hysteresis*.

Tujuan utama dari penggunaan algoritma *Canny Edge Detector* adalah untuk mengurangi kemungkinan kesalahan dalam pendeteksian tepi, yang dapat mengakibatkan terjadinya tepi ganda. Selain itu, algoritma ini juga bertujuan untuk meminimalkan kesalahan dalam mendeteksi tepi secara keseluruhan, serta untuk meminimalkan perbedaan antara tepi yang terdeteksi dengan tepi yang sebenarnya.

## **2.10. Aplikasi Mobile**

Aplikasi *mobile* menggabungkan kata "*application*" (aplikasi) dan "*mobile*" (bergerak). Dalam konteks teknologi informasi, aplikasi merujuk pada program siap pakai yang dirancang untuk menjalankan berbagai fungsi bagi pengguna atau aplikasi lain (Buyens, 2001). Sementara itu, istilah "*mobile*" mengindikasikan perpindahan dari satu tempat ke tempat lain.

Aplikasi mobile secara khusus merujuk pada aplikasi yang berjalan pada perangkat yang dapat dengan mudah dibawa ke mana-mana, seperti smartphone. Ini memungkinkan pengguna untuk melakukan berbagai kegiatan dan aktivitas secara mobile, menyediakan kemudahan dan praktisitas dalam pekerjaan, hiburan, dan bahkan berbelanja.

## **2.11. React Native**

React Native adalah suatu *framework JavaScript* yang digunakan untuk membangun aplikasi iOS dan Android *secara native*, yang dirender (Eisenman, 2015). Fitur dari React Native didasarkan pada *React*, sebuah pustaka *JavaScript* yang digunakan untuk membuat antarmuka pada situs web. Dengan kata lain, aplikasi yang dikembangkan dengan React Native adalah aplikasi *native*. Berbeda dengan aplikasi *hybrid*, aplikasi ini sangat kompatibel dengan *platform* di mana aplikasi tersebut dijalankan. *Platform* tersebut tidak hanya mendukung fungsionalitas aplikasi *hybrid*, yang menggunakan konsep presentasi web yang dibuat dengan HTML5.

React Native dibangun menggunakan kombinasi XML dan *JavaScript*, yang disebut JSX. Dalam hal rendering, React Native menggunakan *bridge* yang memanggil rendering asli *Objective C* untuk iOS dan Java untuk Android.

## 2.12. Python

Python adalah bahasa pemrograman yang bersifat interpretatif, berorientasi objek, dan dinamis. Python memiliki struktur data tingkat lanjut, pengetikan dinamis, serta pengikatan dinamis. Keunggulan Python meliputi penggunaan sintaks yang sederhana dan mudah dipahami. Python mendukung modul dan paket, memudahkan modularitas program dan penggunaan ulang kode. Selain itu, Python tersedia secara gratis dan didistribusikan secara bebas untuk semua platform.

## 2.13. Tensorflow

TensorFlow adalah sebuah perpustakaan (*library*) yang diciptakan khusus oleh tim *Google Brain* untuk keperluan pembelajaran mesin dan penelitian dalam jaringan syaraf. Beberapa fungsi utama dari perpustakaan ini mencakup pemrosesan matriks multidimensi, yang disebut tensor, untuk keperluan perhitungan dan optimasi. TensorFlow sangat mendukung pemrograman dalam konteks pembelajaran mesin.

Selain itu, TensorFlow memiliki kemampuan untuk menjalankan kode langsung pada *Central Processing Unit* (CPU) maupun *Graphics Processing Unit* (GPU), memberikan fleksibilitas dalam penggunaan daya komputasi untuk tugas-tugas yang melibatkan pembelajaran mesin dan pengolahan data besar.

## 2.14. Keras

Keras adalah sebuah API (*Application Programming Interface*) *deep learning* yang ditulis dalam bahasa pemrograman Python dan dijalankan di platform *machine learning* TensorFlow. Keras dikembangkan dengan fokus utama untuk memfasilitasi pengujian yang cepat, baik pada *Central Processing Unit* (CPU) maupun *Graphic Processing Unit* (GPU). Keras mendukung pengembangan jaringan saraf konvolusional, *recurrent neural network*, maupun kombinasi keduanya. Kombinasi ini memungkinkan pengguna untuk membuat dan menguji model *deep learning* dengan lebih mudah dan efisien.

## 2.15. Google Colab

*Google Colab*, atau *Google Colaboratory*, adalah dokumen yang dapat dieksekusi (*executable document*) yang memungkinkan pengguna untuk menulis, mengedit, dan

berbagi program yang telah disimpan melalui *Google Drive*. Salah satu fitur utama dari *Google Colab* adalah adanya *built-in machine learning*, yang menjadi salah satu fitur yang paling populer. Platform ini juga menyediakan beberapa perpustakaan (*library*) machine learning terkemuka, termasuk *Keras*, *PyTorch*, dan *TensorFlow*.

Dengan adanya *Google Colab*, pengguna dapat dengan mudah mengakses lingkungan pengembangan yang telah terkonfigurasi secara baik untuk tugas-tugas *machine learning*, termasuk keberlanjutan integrasi dengan layanan *Google Drive* dan kemampuan untuk menjalankan kode pada unit pemrosesan grafis (GPU) secara gratis. Ini membuatnya menjadi pilihan yang populer bagi banyak pengembang dan peneliti dalam mengembangkan dan berbagi proyek *machine learning*.

## **2.16. Ngrok**

Ngrok merupakan sebuah *server proxy* yang digunakan untuk menciptakan atau membuka jaringan privat melalui NAT atau *firewall*. Selanjutnya, Ngrok akan mengaitkan *localhost* ke internet melalui sebuah tunnel yang aman. Dengan kata lain, ngrok dapat dijelaskan sebagai layanan gratis yang memberikan kemampuan kepada aplikasi kita untuk dapat diakses secara *online*.

## **2.17. Expo SDK**

*Expo* adalah suatu *framework* yang dirancang untuk menyederhanakan proses pengembangan aplikasi menggunakan *React Native*. *Expo* dapat dijelaskan sebagai kumpulan alat, pustaka, dan layanan yang dirancang khusus untuk mempermudah pengembangan kode dalam aplikasi *React Native*. Dengan *Expo*, pengembang dapat memanfaatkan berbagai fitur dan fungsi yang disediakan untuk menyederhanakan dan mempercepat siklus pengembangan aplikasi mereka. Ini mencakup berbagai alat pengembangan, modul dan komponen siap pakai, serta layanan tambahan yang membantu memfasilitasi proses pembuatan aplikasi *React Native*.

## **2.18. Metode Perhitungan Performa Machine Learning**

Saat menggunakan algoritma *machine learning*, perlu diterapkan metode yang dapat mengukur kinerja model, sehingga diperhitungkan saat memilih model dengan kinerja terbaik berdasarkan data yang digunakan dalam perhitungan atau pengambilan

keputusan. Ada dua jenis metode penilaian berdasarkan jumlah kelas klasifikasi, yaitu klasifikasi biner (dua kelas) dan klasifikasi multi-kelas (lebih dari dua kelas) (Tharawat, 2018).

Pada penelitian ini, penulis menggunakan metode perhitungan performa *machine learning* menggunakan teknik *confusion matrix* dalam menerapkan klasifikasi multi-kelas dengan metode *supervised learning*. *Confussion matrix* atau sering disebut juga dengan *error matrix* adalah metode untuk melakukan perhitungan performa padakonsept sistem pendukung keputusan dengan mempresentasikan prediksi dan kondisi aktual dari data yang dihasilkan oleh algoritma *machine learning*. *Confusion matrix* berbentuk tabel matriks yang menggambarkan kinerja model terhadap data uji. Tabel matriks dapat dilihat pada Tabel 2.1.

**Tabel 2.1** *Confussion Matrix*

		Actual values				
Predicted Values		A	B	C	D	E
	A	TP <sub>A</sub>	EB <sub>A</sub>	EC <sub>A</sub>	ED <sub>A</sub>	EE <sub>A</sub>
	B	EAB	TP <sub>B</sub>	EC <sub>B</sub>	ED <sub>B</sub>	EE <sub>B</sub>
	C	EAC	EBC	TP <sub>C</sub>	ED <sub>C</sub>	EE <sub>C</sub>
	D	EAD	EBD	ECD	TP <sub>D</sub>	EED
	E	EAE	EBE	ECE	EDE	TP <sub>E</sub>

*Confussion matrix* juga dapat menghitung berbagai *performance metrics*, diantaranya yaitu:

1. *Accuracy*

*Accuracy* merupakan persentase tingkat kebenaran nilai prediksi dengan nilai aktual pada keseluruhan data. Semakin tinggi nilai *accuracy*, maka semakin akurat model yang digunakan untuk mengklasifikasikan data dengan benar. Nilai *accuracy* dapat diperoleh dengan persamaan 2.6.

$$\text{Jumlah data prediksi benar} = TP_A + TP_B + TP_C + TP_D$$

(2.4)

$$\begin{aligned}
\text{Jumlah keseluruhan data} = & TPA + EAB + EAC + EAD + EAE + TPB + EBA + EBC \\
& + EBD + EBE + TPC + ECA + ECB + ECD + ECE + TPD \\
& + EDA + EDB + EDC + EDE + TPE + EEA + EEB + EEC \\
& + EED
\end{aligned}
\tag{2.5}$$

$$\text{Akurasi} = \frac{\text{Jumlah data diprediksi benar}}{\text{Jumlah keseluruhan data}}
\tag{2.6}$$

Keterangan:

TP = *True Positive*

E = *Error*

## 2. Precision

*Precision* adalah sebuah metrik yang mengukur persentase tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Dalam konteks ini, *precision* dapat diartikan sebagai persentase nilai prediksi benar positif dibandingkan dengan jumlah prediksi positif terhadap nilai yang diminta. Rumus untuk menghitung nilai *precision* dapat dinyatakan dalam persamaan 2.7.

$$\text{Precision A} = \frac{TP_A}{TP_A + E_{AB} + E_{AC} + E_{AD} + E_{AE}}
\tag{2.7}$$

Keterangan:

TP = *True Positive*

E = *Error*

## 3. Recall

*Recall* merupakan persentase keberhasilan model dalam menemukan kembali sebuah informasi yang diminta. Maka, *recall* merupakan nilai prediksi benar positif dibandingkan dengan jumlah aktual positif terhadap nilai yang diminta. Nilai *recall* dapat diperoleh dengan persamaan 2.8.



$$Recall A = \frac{TP_A}{TP_A + E_{BA} + E_{CA} + E_{DA} + E_{EA}} \quad (2.8)$$

Keterangan:

TP = *True Positive*

E = *Error*

#### 4. *F1 Score*

*F1 Score* merupakan fungsi dari *precision* dan *recall* yang merupakan persentase tingkat keseimbangan antara *precision* dan *recall*. Nilai *F1 Score* dapat diperoleh dengan persamaan 2.9.

$$F1 Score = 2 \times \frac{Precision A \times Recall A}{Precision A + Sensitivity A} \quad (2.9)$$

### 2.19. Penelitian Terdahulu

Gideon et al. (2018) menjelaskan bahwa penggunaan metode *Convolutional Neural Network* (CNN) dapat mendeteksi tanda tangan dengan baik dimana terdapat 3 pembagian rasio data yang dilakukan, pembagian rasio data 8:2 mendapatkan hasil yang terbaik dengan 98 % *training* dan *validation accuracy*, sedangkan untuk rasio data 7:3 mendapatkan hasil 97 % *training accuracy* dan 94 % *validation accuracy*, dan yang terakhir rasio data 6:4 mendapatkan 97 % *training accuracy* dan 96 % *validation accuracy*. Pada penelitian ini digunakan dataset yang berisi 6000 tanda tangan dengan 1000 tanda tangan asli dan 1000 tanda tangan palsu per subjek, dimana terdapat 1 orang membuat tanda tangan asli dan 2 orang berbeda membuat tiruannya.

Pansare et al. (2012) memverifikasi tanda tangan menggunakan metode neural network. Dimana pada metode tersebut digunakan *feature extracted* dari citra tanda tangan yang telah diproses sebelumnya, *extracted feature* digunakan untuk melatih *neural network* menggunakan algoritma *error back propagation training*. Dengan *CCR in Recall* 100 % pada tes *neural network* pada sampel tanda tangan yang telah dilatih, sistem dapat mengklasifikasikan semua tanda tangan asli dan palsu dengan benar. Namun ketika disajikan dengan sampel tanda tangan dari *database* berbeda daripada

yang digunakan pada fase *training*, dari 300 tanda tangan terdapat 248 tanda tangan yang dapat dikenali dengan benar. Secara keseluruhan, *correct classification rate* dari sistem adalah 82,66 %.

Hafemann et al. (2017) yang menganalisis *learning features* untuk verifikasi tanda tangan *offline* menggunakan *Deep Convolutional Neural Network*. Fitur yang dipelajari dengan cara *writer-independent* lebih efektif untuk verifikasi tanda tangan dibandingkan metode fitur *hand-engineered*. Dan juga pada penelitian ini diaplikasikan metode yang berbeda untuk representasi pembelajaran untuk verifikasi tanda tangan *offline*.

Bhattacharya et al. (2013) memverifikasi tanda tangan *offline* menggunakan metode *Pixel Matching Technique*. Metode ini merupakan metode yang sangat simpel dan mudah untuk diimplementasikan. Menggunakan persamaan dari geometri koordinat membuat metode ini lebih cepat daripada metode lainnya.

Drott et al. (2015) menggunakan *Machine Learning Technique* dengan pendekatan *Deep Learning* untuk melakukan verifikasi tanda tangan *online*. Algoritma *Deep Learning* telah dikembangkan untuk menangani masalah verifikasi tanda tangan. Untuk memecahkan masalah, *code framework* dibangun dengan *database* tanda tangan, *classifier* dan metode evaluasi. Beberapa algoritma klasifikasi yang berbeda diuji sebelum konvolusi akhir jaringan saraf dikembangkan. Hasil dari model akhir mencatatkan *true positive rate (sensitivity)* 96,7 % dan *false positive rate* 0,6 %.

**Tabel 2.2 Penelitian Terdahulu**

No.	Peneliti	Tahun	Metode	Hasil
1	Gideon, et al	2019	<i>Handwritten Signature Forgery Detection using Convolutional Neural Networks</i>	Sistem dapat mendeteksi tanda tangan dengan baik dimana pembagian rasio data 8:2 mendapatkan hasil yang terbaik dengan 98 % <i>training</i> dan <i>validation accuracy</i>
2	Pansare, et al	2012	<i>Handwritten Signature</i>	Menghasilkan sistem yang memiliki <i>correct classification</i>

No.	Peneliti	Tahun	Metode	Hasil
			<i>Verification using rate</i>	sebesar 82,66 % dalam verifikasi tanda tangan
3	Haffemann, et al	2017	<i>Learning Features for Offline Handwritten Signature Verification using Deep Convolutional Neural Networks</i>	Melakukan analisa learning features untuk verifikasi tanda tangan <i>offline</i> menggunakan Deep Convolutional Neural Network.
4	Bhattacharya, et al	2013	<i>Offline Signature Verification Using Pixel Matching Technique</i>	Menghasilkan sistem yang dapat melakukan verifikasi tanda tangan offline menggunakan metode Pixel Matching Technique
5	Drott, et al	2015	<i>On-line Handwritten Signature Verification using Machine Learning Techniques with a Deep Learning Approach</i>	Mencatatkan hasil verifikasi tanda tangan online dimana <i>true positive rate</i> (sensitivity) 96,7 % dan <i>false positive rate</i> 0,6 %.
6	Poddar, et al,	2020	<i>Offline Signature Recognition and Forgery Detection Using Deep Learning</i>	Mendeteksi pemalsuan tanda tangan dengan menggunakan 2 algoritma, yaitu Harris Algorithm dan Surf Algoritim, sedangkan untuk identifikasi pemilik tanda tangan digunakan

No.	Peneliti	Tahun	Metode	Hasil
				Convolutional Neural Network (CNN).

## 2.20. Perbedaan Penelitian Terdahulu

Perbedaan pada penelitian ini dengan penelitian yang dilakukan oleh (Gideon, *et al.* 2016) dan penelitian (Pansare, *et al.* 2016), kedua penelitian tersebut menggunakan *convolutional neural network* sebagai metode yang digunakan untuk mendeteksi tanda tangan, adapun perbedaan jumlah data yang digunakan pada kedua penelitian tersebut, pada penelitian oleh (Gideon, *et al.* 2016) digunakan 6000 tanda tangan dimana per subjek terdapat 1000 tanda tangan asli dan 1000 tanda tangan palsu, dimana terdapat 1 subjek untuk tanda tangan asli dan 2 subjek lainnya untuk tanda tangan tiruan. Sedangkan, untuk penelitian yang dilakukan oleh (Pansare, *et al.* 2016) jumlah tanda tangan yang digunakan sebanyak 1440 tanda tangan yang diambil dari 30 subjek berbeda, dimana setiap subjek memberikan 24 sampel tanda tangan asli dan 24 sampel tanda tangan tiruan.

Selanjutnya perbedaan penelitian yang penulis lakukan dengan penelitian (Haffeman, *et al.* 2017) bertujuan untuk mengevaluasi apakah pendekatan menggunakan CNN mampu menghasilkan model yang efektif dalam membedakan antara tanda tangan asli dan palsu. Terdapat beberapa metode dalam pelatihan CNN pada penelitian tersebut seperti *backpropagation* dan *stochastic gradient descent*.

Perbedaan penelitian penulis dengan penelitian (Bhattacharya, *et al.* 2013) yaitu metode yang digunakan berbeda, dimana *pixel matching technique* merupakan metode yang digunakan pada penelitian tersebut untuk melakukan verifikasi tanda tangan. Pada penelitian ini, data tanda tangan diambil dari hasil *scan* gambar dari suatu dokumen dimana hasil penelitian ini menunjukkan performa metode PMT lebih baik dibandingkan ANN (*Artificial Neural Network*) dan SVM (*Support Vector Machine*).

Perbedaan penelitian berikutnya (Drott, *et al.* 2015) dan (Poddar, *et al.* 2020) terletak pada metode dan algoritma yang digunakan, penelitian (Drott, *et al.* 2015) berfokus pada metode *machine learning* dengan pendekatan *deep learning*, dimana pada proses perbandingan tanda tangan menggunakan beberapa algoritma yang berbeda seperti *easy compare algorithm*, *logistic regression* dan *multi-layer perceptron*.

Penelitian (Poddar, *et al.* 2020) menggunakan *convolution neural network* (CNN), *crest-through method*, *surf algorithm*, dan *harris corner detection* yang menghasilkan akurasi 85 – 89% untuk deteksi pemalsuan dan 90 – 94% untuk pengenalan tanda tangan.

## BAB 3

### ANALISIS DAN PERANCANGAN SISTEM

#### 3.1. Datas yang Digunakan

Data yang akan digunakan terdiri dari citra tanda tangan yang sebelumnya telah diambil menggunakan *smartphone*. Citra-citra ini disimpan dalam format JPG, dan totalnya terdapat sebanyak 1500 citra. Contoh data dapat dilihat pada Gambar 3.1.



**Gambar 3.1** Contoh Data Citra Tanda Tangan

Data yang digunakan dalam penelitian ini terdiri dari citra tanda tangan dari 3 orang yang berbeda. Selanjutnya, data ini akan dibagi menjadi dua bagian dengan rasio pembagian 80:20, yaitu 80% atau sebanyak 1200 citra sebagai data latih, dan 20% atau sebanyak 300 citra sebagai data uji, sebagaimana terlihat pada Tabel 3.1.

**Tabel 3.1** Jumlah Data Latih dan Uji

Dataset	Data Pelatihan	Data Pengujian	Jumlah Dataset
Tanda Tangan 1	24	6	30
Tanda Tangan 2	24	6	30
Tanda Tangan 3	24	6	30
Tanda Tangan 4	24	6	30
Tanda Tangan 5	24	6	30
Tanda Tangan 6	24	6	30
Tanda Tangan 7	24	6	30
Tanda Tangan 8	24	6	30
Tanda Tangan 9	24	6	30
Tanda Tangan 10	24	6	30
Tanda Tangan 11	24	6	30
Tanda Tangan 12	24	6	30
Tanda Tangan 13	24	6	30
Tanda Tangan 14	24	6	30
Tanda Tangan 15	24	6	30
Tanda Tangan 16	24	6	30
Tanda Tangan 17	24	6	30
Tanda Tangan 18	24	6	30
Tanda Tangan 19	24	6	30
Tanda Tangan 20	24	6	30
Tanda Tangan 21	24	6	30
Tanda Tangan 22	24	6	30
Tanda Tangan 23	24	6	30
Tanda Tangan 24	24	6	30
Tanda Tangan 25	24	6	30
Tanda Tangan 26	24	6	30
Tanda Tangan 27	24	6	30
Tanda Tangan 28	24	6	30
Tanda Tangan 29	24	6	30

Dataset	Data Pelatihan	Data Pengujian	Jumlah Dataset
Tanda Tangan 30	24	6	30
Tanda Tangan 31	24	6	30
Tanda Tangan 32	24	6	30
Tanda Tangan 33	24	6	30
Tanda Tangan 34	24	6	30
Tanda Tangan 35	24	6	30
Tanda Tangan 36	24	6	30
Tanda Tangan 37	24	6	30
Tanda Tangan 38	24	6	30
Tanda Tangan 39	24	6	30
Tanda Tangan 40	24	6	30
Tanda Tangan 41	24	6	30
Tanda Tangan 42	24	6	30
Tanda Tangan 43	24	6	30
Tanda Tangan 44	24	6	30
Tanda Tangan 45	24	6	30
Tanda Tangan 46	24	6	30
Tanda Tangan 47	24	6	30
Tanda Tangan 48	24	6	30
Tanda Tangan 49	24	6	30
Tanda Tangan 50	24	6	30
Jumlah Seluruh Data	1200	300	1500

Data *training* atau data latih bertujuan untuk mengajari algoritma dengan menyesuaikan parameter-parameter yang ada agar sesuai dengan informasi yang terdapat pada data, memungkinkan algoritma untuk memahami dan menangkap pola dalam data tersebut. Sebaliknya, data *testing* atau data uji bertujuan untuk melakukan evaluasi terhadap performa algoritma yang sudah mengalami proses latihan sebelumnya.

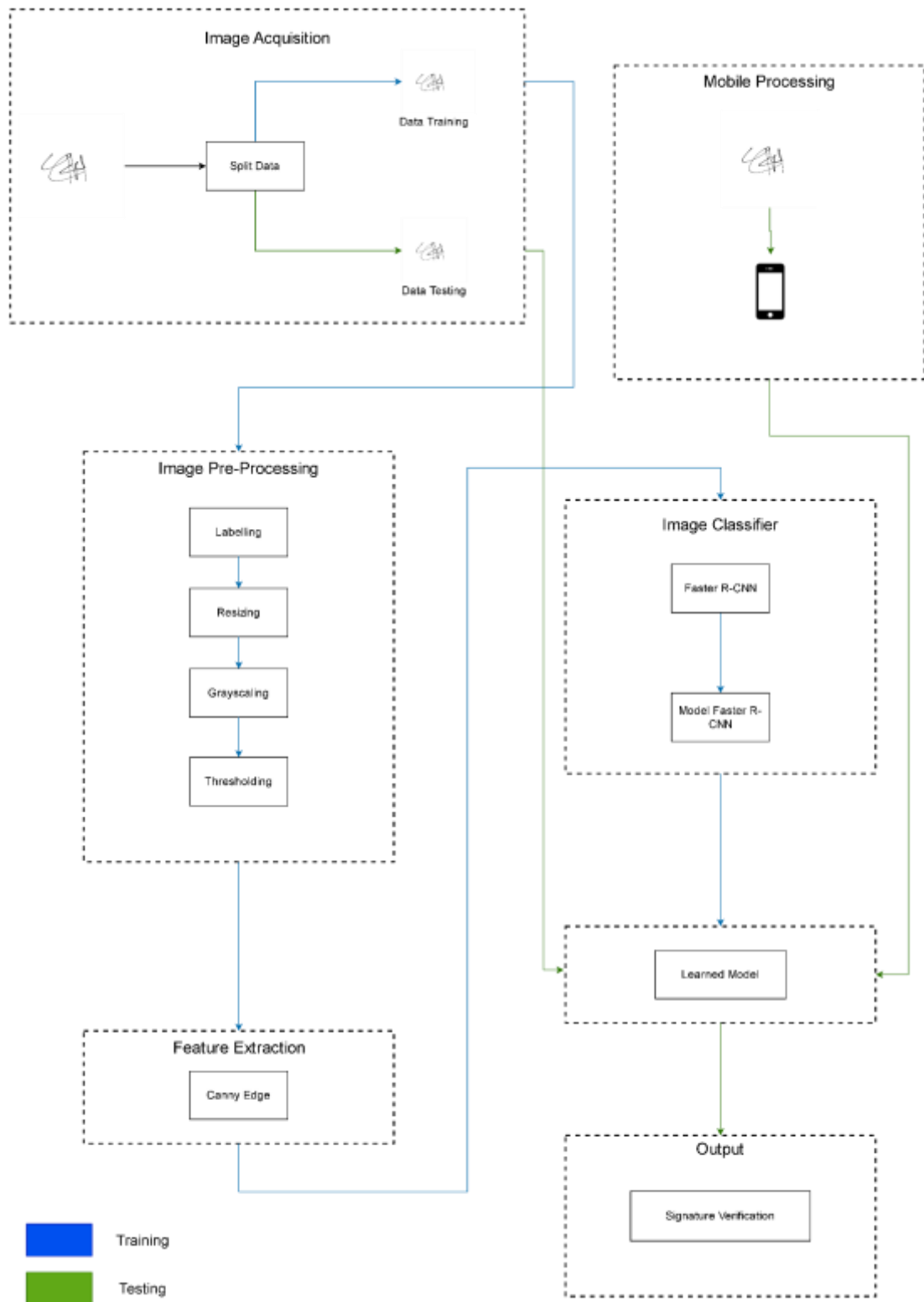


### 3.2. Arsitektur Umum

Penelitian ini melibatkan beberapa tahapan. Tahap pertama adalah pengumpulan data citra tanda tangan yang diambil dari 50 orang yang berbeda. Data yang terkumpul akan digunakan sebagai data latih dan data uji. Dalam proses pengumpulan data ini, citra diambil menggunakan aplikasi yang terdapat pada *smartphone*.

Tahap kedua adalah pre-processing. Pada tahap ini, data akan melalui proses *Labeling*, *Resizing*, *Grayscale* dan *Thresholding*. Proses *Labeling* bertujuan untuk membedakan data sesuai dengan jenisnya, yakni data untuk pelatihan dan data untuk pengujian. Setelah melalui tahap *labeling*, data akan mengalami *resizing* untuk menyamakan ukuran piksel pada citra agar memiliki ukuran yang seragam. Pada tahap *grayscale* citra akan diubah seluruhnya menjadi citra dengan skala keabuan. Terakhir, data akan dipisahkan menjadi beberapa kelas berdasarkan nilai intensitas piksel masing-masing citra pada proses *thresholding*.

Tahap terakhir adalah tahap klasifikasi data menggunakan metode *Faster Region Convolution Neural Network* (Faster R-CNN). Setelah melalui tahap-tahap dan proses sebelumnya, tahap ini menghasilkan *output* berupa tingkat kemiripan tanda tangan. Tahapan analisis yang telah dilakukan di atas dapat dilihat dalam bentuk arsitektur umum pada Gambar 3.2.



**Gambar 3.2** Arsitektur Umum

Penjelasan arsitektur umum pada gambar 3.2 adalah sebagai berikut:

### 3.2.1. Image Acquisition

Pada tahap ini, fokusnya adalah mengumpulkan data citra tanda tangan yang akan digunakan sebagai input awal. Penelitian ini menggunakan data citra tanda tangan dari 50 orang yang berbeda. Proses pengambilan data citra tanda tangan dilakukan secara langsung dengan menggunakan aplikasi pada *smartphone*. Seluruh citra yang digunakan dalam penelitian ini memiliki format file .jpg dan akan dibagi menjadi dua bagian, yaitu data latih dan data uji.

Dalam proses implementasinya, variabel dibuat menjadi dua, pertama adalah `'train_ds'` yang merupakan variabel untuk data latih dengan *validation split* sebesar 0.8. Angka 0.8 tersebut mencerminkan pembagian data dengan rasio 80:20. Selanjutnya, subset ini diberi nama "*training*". Variabel kedua adalah `'val_ds'` yang merupakan variabel untuk data uji dengan *validation split* sebesar 0.2. Ini juga digunakan untuk pembagian data dengan rasio 80:20, dan diberi nama subset "*validation*".

```
# Create training dataset
Call image_dataset_from_directory with the following
arguments:
- directory: "drive/MyDrive/Dataset 4"
- validation_split: 0.2
- subset: "training"
- seed: 123
- color_mode: 'rgb'
- image_size: (image height, image width)
- batch_size: batch size
Store the result in train_ds

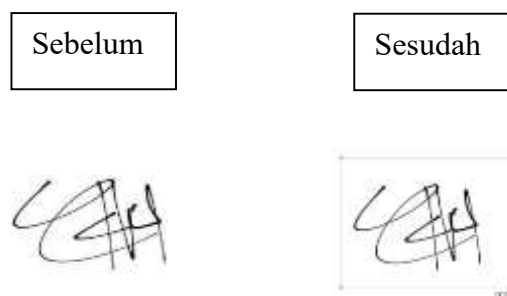
# Create validation dataset
Call image_dataset_from_directory with the following
arguments:
- directory: "drive/MyDrive/Dataset 4"
- validation_split: 0.2
- subset: "validation"
- seed: 123
- color_mode: 'rgb'
- image_size: (image height, image width)
- batch_size: batch size
Store the result in val_ds
```

### 3.2.2. Image Pre-Processing

Langkah berikutnya adalah tahap *pre-processing*, yang bertujuan untuk menghasilkan citra tanda tangan yang lebih optimal untuk diproses pada tahap selanjutnya. Dalam tahap ini, dilakukan proses *labelling*, di mana bagian-bagian pada tanda tangan yang akan digunakan ditandai. Setelah melalui proses *labelling*, data kemudian melewati tahap *resizing*, yaitu tahap di mana ukuran piksel pada data diatur agar seragam, memudahkan proses data *testing*. Setelah itu, data akan melalui proses *grayscale* dan *thresholding*.

#### 1. Labelling

Tahap ini adalah tahap di mana proses *labelling* digunakan untuk memberikan label atau nama pada citra tanda tangan yang akan diproses. Dengan demikian, data tersebut dapat diolah untuk melakukan verifikasi tanda tangan. Contoh dari proses labelling dapat dilihat pada Gambar 3.3.



**Gambar 3.3** Citra Sebelum dan Sesudah *Labelling*

#### 2. Resizing

Tahap ini merupakan proses di mana citra diolah untuk memastikan ukuran pikselnya seragam pada setiap citra. Tujuan dari proses ini adalah untuk mengurangi waktu yang diperlukan saat menguji data. Ukuran citra akan diubah menjadi 360 x 360 piksel. Proses resizing dilakukan secara langsung di *Google Colab*. *Pseudocode* untuk proses *resizing* dapat dilihat di bawah ini:

```
# Define parameters
Set image height to desired value (e.g., 360)
Set image width to desired value (e.g., 360)
Set batch size to desired value (e.g., 25)
```

Fungsi dari `img\_height` adalah untuk mengatur dimensi ketinggian pada citra, sementara `img\_width` berfungsi untuk mengatur dimensi lebar pada citra.

### 3. Grayscale

Tahap ini bertujuan untuk mengubah citra menjadi warna abu-abu sehingga warna keabuan merata secara keseluruhan pada data citra tandatangan. Berikut adalah contoh *pseudocode* untuk proses *grayscale*.

```
// Function to convert an image to grayscale
using OpenCV
Function ConvertToGrayscale(image_path):
    // Step 1: Load the image in BGR format
    img = LoadImage(image_path)

    // Step 2: Convert the image to float32
format
    img_float32 = ConvertImageToFloat32(img)

    // Step 3: Convert the image from BGR to
grayscale
    grayscale_img =
ConvertBGRToGrayscale(img_float32)

    // Step 4: Return the grayscale image
    Return grayscale_img

// Function to load an image
Function LoadImage(image_path):
    img = cv2.imread(image_path)
    Return img

// Function to convert an image to float32
format
Function ConvertImageToFloat32(img):
    img_float32 = np.float32(img)
    Return img_float32

// Function to convert BGR image to grayscale
Function ConvertBGRToGrayscale(img_float32):
    grayscale_img = cv2.cvtColor(img_float32,
cv2.COLOR_BGR2GRAY)
    Return grayscale_img
```

Contoh hasil proses *grayscale* dapat dilihat pada Gambar 3.4.



**Gambar 3.4** Citra Hasil *Grayscale*

#### 4. *Thresholding*

*Thresholding* dalam pemrosesan gambar adalah teknik yang digunakan untuk membagi gambar menjadi dua atau lebih kelas berdasarkan nilai intensitas pikselnya. Tujuannya adalah untuk memisahkan objek atau fitur yang menarik dari latar belakang atau *noise* dengan cara yang sederhana dan efektif.

Ada beberapa jenis *thresholding* yang umum digunakan dalam pemrosesan gambar:

##### 1. *Thresholding Biner (Binary Thresholding)*

Teknik ini membagi gambar menjadi dua kelas yaitu hitam dan putih. Setiap piksel yang memiliki nilai intensitas di atas atau di bawah nilai ambang tertentu diberi label putih atau hitam, secara berturut-turut. Misalnya, dalam gambar grayscale, piksel dengan intensitas di atas nilai ambang akan menjadi putih, sementara yang di bawahnya menjadi hitam.

##### 2. *Thresholding Global (Global Thresholding)*

Pada metode ini, nilai ambang yang sama diterapkan ke seluruh gambar. Ini bergantung pada sifat global distribusi intensitas piksel. Metode Otsu adalah salah satu contoh dari *thresholding* global yang populer.

##### 3. *Thresholding Adaptif (Adaptive Thresholding)*

*Thresholding* ini memungkinkan penggunaan nilai ambang yang berbeda untuk setiap bagian gambar. Ini bermanfaat ketika gambar memiliki variasi pencahayaan yang signifikan. Metode ini membagi gambar menjadi bagian-bagian kecil dan menerapkan *thresholding* pada masing-masing bagian dengan ambang yang sesuai.

##### 4. *Thresholding Multilevel (Multilevel Thresholding)*

Jenis *thresholding* ini membagi gambar menjadi lebih dari dua kelas berdasarkan beberapa nilai ambang. Ini berguna ketika gambar memiliki struktur kompleks atau benda dengan intensitas yang berbeda-beda.

*Thresholding* umumnya digunakan dalam berbagai aplikasi pemrosesan gambar, termasuk segmentasi objek, pengenalan karakter, deteksi tepi, dan banyak lagi. Keuntungan dari teknik ini adalah kesederhanaannya dan efektivitasnya dalam menghasilkan representasi yang jelas dari objek-objek di dalam gambar. Namun, pemilihan nilai ambang yang tepat bisa menjadi tantangan tergantung pada karakteristik gambar yang diproses. Berikut adalah contoh *pseudocode* untuk proses *thresholding*.

```
Function SetThresholds(mag_array,
weak_threshold, strong_threshold):
    // Step 1: Find the maximum value in the
mag array
    mag_max = FindMaxValue(mag_array)

    // Step 2: Check if weak_threshold is not
defined
    If weak_threshold is not defined:
        // Set weak_threshold to 10% of mag_max
        weak_threshold = 0.10 * mag_max

    // Step 3: Check if strong_threshold is not
defined
    If strong_threshold is not defined:
        // Set strong_threshold to 50% of
mag_max
        strong_threshold = 0.50 * mag_max

    Return weak_threshold, strong_threshold

// Helper function to find the maximum value in
an array
Function FindMaxValue(array):
    max_value = -Infinity
    For each value in array:
        If value > max_value:
            max_value = value
    Return max_value
```

### 3.2.3. Feature Extraction

Pada langkah ini, digunakan *Canny Edge Detection* sebagai metode Ekstraksi Fitur. Data yang sebelumnya telah melalui tahap *Pre-Processing* dimasukkan ke dalam proses ini

#### 1. *Canny Edge Detection*

*Canny edge detection* adalah teknik yang umum digunakan dalam pemrosesan gambar untuk menemukan tepi objek dalam gambar. Metode ini dikembangkan oleh John F. Canny pada tahun 1986 dan sering digunakan karena kemampuannya yang baik dalam menemukan tepi dengan akurasi tinggi serta dapat mengurangi efek dari *noise*.

Berikut adalah langkah-langkah utama dari algoritma *Canny Edge Detection*:

- *Noise Reduction*

Langkah pertama adalah mengurangi *noise* dalam gambar, karena *noise* dapat mempengaruhi hasil deteksi tepi. Ini biasanya dilakukan dengan menggunakan filter *smoothing*, seperti filter Gaussian, untuk meratakan gambar.

- Perhitungan Gradien

Setelah gambar bersih dari *noise*, gradien atau turunan gambar dihitung menggunakan operator seperti operator Sobel. Ini menghasilkan magnitudo gradien dan arah gradien untuk setiap piksel di dalam gambar. Magnitudo gradien menunjukkan kekuatan perubahan intensitas di sekitar setiap piksel, sementara arah gradien menunjukkan orientasi perubahan intensitas tersebut.

- *Non-maximum Suppression*

Langkah ini bertujuan untuk menyempitkan tepi menjadi garis tipis. Untuk setiap piksel, dilakukan pengecekan apakah magnitudo gradien piksel tersebut merupakan maksimum lokal di arah gradien. Jika ya, piksel tersebut dipertahankan, jika tidak, piksel tersebut dihapus (diasumsikan bukan tepi).

- *Hysteresis Thresholding*

Tahap terakhir melibatkan penggunaan dua nilai ambang (ambang



atas dan ambang bawah) untuk menentukan piksel mana yang merupakan tepi yang sebenarnya. Piksel dengan magnitudo gradien di atas ambang atas dinyatakan sebagai tepi kuat, sedangkan piksel yang terletak antara ambang atas dan ambang bawah dinyatakan sebagai tepi lemah. Tepi lemah hanya dianggap sebagai tepi yang sebenarnya jika terhubung dengan tepi kuat.

Algoritma *canny edge detection* menghasilkan tepi yang halus, akurat, dan tajam, dengan kemampuan untuk mengurangi efek dari *noise* dan menghilangkan banyak garis tepi palsu yang mungkin dihasilkan oleh metode deteksi tepi lainnya. Ini membuatnya sangat populer dalam aplikasi pengolahan gambar yang memerlukan deteksi tepi yang andal, seperti deteksi objek, pengenalan pola, dan segmentasi gambar.

#### 3.2.4. Image Classifier

*Faster Region Convolution Neural Network (Faster R-CNN)* digunakan sebagai metode penelitian untuk menentukan tingkat kemiripan tanda tangan. Data yang diinputkan pada metode ini adalah citra tanda tangan yang telah melalui tahapan sebelumnya. Dalam pemrosesan data menggunakan *Faster Region Convolution Neural Network*, citra dibagi menjadi dua proses dengan data citra yang berbeda. Langkah-langkah yang dilakukan dalam pembelajaran citra menggunakan *Faster Region Convolution Neural Network* adalah sebagai berikut:

*Faster R-CNN* mengambil *tensor* dengan bentuk (image\_height, image\_width, color\_channels) sebagai input. Untuk melengkapi model, *tensor* keluaran terakhir dari basis konvolusi dimasukkan ke dalam satu atau lebih lapisan dense untuk melakukan klasifikasi. Lapisan padat ini menerima vektor sebagai *input*, dan outputnya saat ini berupa *tensor* 3D

Untuk pseudocodenya dapat dilihat dibawah ini.

```
// Function to convert an image to grayscale using OpenCV
Function ConvertToGrayscale(image_path):
    // Step 1: Load the image in BGR format
    img = LoadImage(image_path)
    // Step 2: Convert the image to float32 format
    img_float32 = ConvertImageToFloat32(img)
    // Step 3: Convert the image from BGR to grayscale
```

```

    grayscale_img = ConvertBGRToGrayscale(img_float32)
    // Step 4: Return the grayscale image
    Return grayscale_img
// Function to load an image
Function LoadImage(image_path):
    img = cv2.imread(image_path)
    Return img
// Function to convert an image to float32 format
Function ConvertImageToFloat32(img):
    img_float32 = np.float32(img)
    Return img_float32
// Function to convert BGR image to grayscale
Function ConvertBGRToGrayscale(img_float32):
    grayscale_img = cv2.cvtColor(img_float32,
cv2.COLOR_BGR2GRAY)
    Return grayscale_img
// Usage
grayscale_image =
ConvertToGrayscale('path_to_your_image.jpg')
DisplayImage(grayscale_image) // To display the image
SaveImage(grayscale_image, 'path_to_save_grayscale_image.jpg')

```

## 1. Proses Training

Pada tahap ini, sistem akan melakukan pembelajaran terhadap data input untuk memilih atau menghasilkan sebuah model menggunakan metode *Faster Region Convolution Neural Network*. Data input yang digunakan untuk data latih lebih banyak daripada data input yang digunakan pada data uji, karena telah melalui proses pembagian data pada tahap akuisisi citra dengan rasio 80:20. Hasil dari data yang telah dilatih adalah model yang selanjutnya akan digunakan dalam proses pengujian. Saat melakukan pelatihan, data input akan dibagi lagi menjadi data pelatihan dan data pengujian. Perbedaan antara kedua jenis data tersebut adalah:

- *Data Training*

Dalam penelitian ini, data training digunakan untuk melatih metode *Faster Region Convolution Neural Network*, di mana data tersebut akan menyesuaikan parameter klasifikasi untuk membentuk modelnya. Jumlah data yang digunakan untuk pelatihan adalah sebanyak 1200 citra tanda tangan.

- *Data Testing*

Data *testing* digunakan untuk mengevaluasi kecocokan model, di mana model tersebut tidak memiliki pengaruh terhadap set data *training* saat

menyesuaikan atau membuat parameter. Dalam penelitian ini, terdapat 300 citra tanda tangan yang digunakan sebagai data *testing*.

Metode *Faster Region Convolution Neural Network (Faster R-CNN)* digunakan dalam pembelajaran *machine learning* pada citra untuk membuat model yang efektif. Dalam penelitian ini, *Faster R-CNN* bertujuan membuat model yang akan digunakan untuk verifikasi tanda tangan.

## 2. Proses Testing

Proses pengujian adalah tahap untuk menguji model yang telah dibentuk selama proses pelatihan. Tujuan dari tahap ini adalah untuk mengevaluasi efektivitas penggunaan metode *Faster Region Convolution Neural Network* pada aplikasi ini. Algoritma yang digunakan akan melalui serangkaian tahapan, dengan model yang telah ditentukan sebelumnya melewati beberapa lapisan (*layers*) sesuai dengan konfigurasi yang telah dijelaskan sebelumnya.



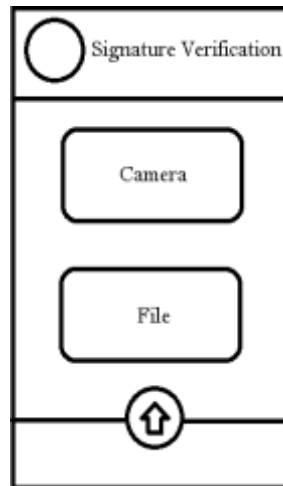
**Gambar 3.5** Gambar hasil proses *testing*

### 3.3. Perancangan Antarmuka Sistem

Dalam tahap ini, dirancang antarmuka pengguna untuk memberikan gambaran aplikasi *Handsign Verification* kepada pengguna. Hal ini bertujuan untuk membantu pengguna memahami dan mengoperasikan aplikasi. Rancangan antarmuka sistem terdiri dari tiga bagian utama, yaitu *Dashboard*, *Testing*, dan *Result*.

### 3.3.1. Rancangan Tampilan Halaman Home

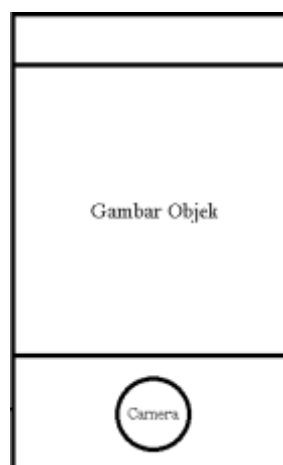
Pada layar utama, atau tampilan *home*, pengguna akan menemui tombol untuk memulai aplikasi dan memulai proses deteksi tanda tangan. Desain tampilan *home* dapat dilihat pada Gambar 3.6.



**Gambar 3.6** Rancangan Tampilan *Home*

### 3.3.2. Rancangan Tampilan Halaman Testing

Pada tampilan *testing*, pengguna dapat melihat gambar langsung melalui kamera smartphone yang akan digunakan untuk mengambil gambar objek yang akan dideteksi. Rancangan tampilan ini dapat dilihat pada Gambar 3.7.



**Gambar 3.7** Rancangan Tampilan *Testing*

### 3.3.3. Rancangan Tampilan Halaman Result

Tampilan hasil atau tampilan result adalah tampilan terakhir pada aplikasi ketika pengguna telah mengambil gambar objek berupa tanda tangan. Tampilan ini menampilkan informasi singkat mengenai hasil dari gambar yang telah diambil sebelumnya. Rancangan tampilan ini dapat dilihat pada Gambar 3.8.



**Gambar 3.8** Rancangan Tampilan Result.

## **BAB 4**

### **IMPLEMENTASI DAN PENGUJIAN**

Pada bab ini akan membahas mengenai implementasi dari metode *Faster Region Convolution Neural Network* dalam proses verifikasi tanda tangan dalam bentuk analisis dan perancangan sistem juga membahas hasil pengujian yang telah dirancang

#### **4.1. Implementasi Sistem**

Dalam tahap ini, implementasi metode *Faster Region Convolution Neural Network* (*Faster R-CNN*) untuk verifikasi tanda tangan dan tingkat kemiripan berbasis *mobile*. Oleh karenanya sistem membutuhkan perangkat keras dan perangkat lunak pendukung, yaitu:

##### *4.1.1. Spesifikasi perangkat keras dan perangkat lunak*

Spesifikasi perangkat keras dan perangkat lunak yang digunakan untuk membangun sistem verifikasi tanda tangan ini adalah:

1. Perangkat laptop yang digunakan adalah: Acer E-5 476G
2. *Processor* Intel Core i5
3. Kapasitas RAM sebesar 8 GB
4. Kapasitas *Hardisk* sebesar 1 TB
5. Kapasitas SSD sebesar 256 GB
6. Sistem operasi yang digunakan Windows 10

##### *4.1.2. Implementasi data*

Data yang digunakan untuk melakukan penelitian ini adalah data yang dikumpulkan dan diambil dari beberapa relawan. Citra tanda tangan diambil melalui *smartphone*, jumlah

data yang dikumpulkan yaitu sebanyak 1500 citra. Berberapa contoh data sample tanda tangan dapat dilihat secara berurut pada Gambar 4.1, Gambar 4.2, Gambar 4.3



**Gambar 4.1** Data Citra Tanda Tangan 1



**Gambar 4.2** Data Citra Tanda Tangan 2



**Gambar 4.3** Data Citra Tanda Tangan 3

#### 4.1.3. Implementasi model

Implementasi model merupakan tahap dimana model yang sebelumnya telah dibuat selanjutnya di *compile* lalu dilakukan uji model seperti pada tabel 4.1.

**Tabel 4.1** Hasil Implementasi Model 1200 data

<i>Epoch</i>	<i>Loss</i>	<i>Accuracy</i>	<i>Val_Loss</i>	<i>Val_Accuracy</i>
1	0.9107	0.6225	0.9531	0.7677
2	0.9156	0.6828	0.8258	0.7427
3	0.8527	0.7410	0.8374	0.7918
4	0.5144	0.6334	0.5499	0.6826
5	0.4510	0.7510	0.3887	0.7829
6	0.0572	0.7677	0.0799	0.8130
7	0.0454	0.7832	0.0737	0.8652
8	0.0271	0.8440	0.0150	0.9058
9	0.0259	0.8888	0.0131	0.9260
10	0.0101	0.9058	0.0069	1.0000

#### 4.1.4. Implementasi perancangan antarmuka

Implementasi perancangan antarmuka merupakan hasil implementasi dari perancangan antarmuka yang sudah dibahas sebelumnya di bab 3, yaitu:

1. Tampilan Halaman Home

Tampilan home akan menampilkan tombol yang diperlukan untuk memulai verifikasi tanda tangan. Tampilan halaman dapat dilihat pada gambar 4.4





**Gambar 4.4** Tampilan Halaman Home

## 2. Tampilan Halaman Kamera

Tampilan halaman kamera adalah tampilan untuk menunjukan objek yang akan diambil gambarnya yaitu citra tanda tangan. Untuk tampilan halaman kamera dapat dilihat pada gambar 4.5 dan gambar 4.6



**Gambar 4.5** Tampilan Halaman Kamera Sebelum Diambil



**Gambar 4.6** Tampilan Halaman Kamera Setelah Diambil

3. Tampilan Halaman Hasil

Tampilan halaman hasil akan menampilkan informasi mengenai verifikasi tanda tangan dan tingkat kemiripannya. Tampilan halaman hasil dapat dilihat pada gambar 4.7.



**Gambar 4.7** Tampilan Halaman Hasil

## 4.2. Prosedur Operasional

Tampilan utama aplikasi yang dibangun terdiri dari halaman *home*, halaman kamera, dan halaman hasil.



**Gambar 4.8** Tampilan Halaman Home

Tampilan halaman *home* dapat dilihat pada Gambar 4.8, pada halaman *dashboard* bagian tengah halaman akan menunjukkan sebuah tombol untuk memulai menjalankan program agar dapat memverifikasi tanda tangan.



**Gambar 4.9** Tampilan Halaman Kamera



**Gambar 4.10** Tampilan Halaman Kamera Sesudah Diambil

Halaman kamera muncul akan muncul jika pengguna sudah menekan tombol mulai untuk menjalankan programnya, halaman kamera akan menunjukkan objek yang akan diambil gambarnya dapat dilihat pada gambar 4.9. Jika objek sudah diambil maka akan muncul tombol sebelumnya akan berubah menjadi tombol untuk memverifikasi bahwa objek yang diambil sudah benar dan akan melanjutkan untuk melihat hasilnya dapat dilihat pada gambar 4.10.

Pada halaman terakhir atau halaman hasil, dapat dilihat nilai akurasi deteksi jenis tanda tangan dan tingkat kemiripan.

















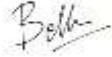
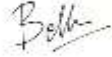
**Gambar 4.11** Tampilan Halaman Hasil










### 4.3. Pengujian Sistem

Pada tahap ini dilakukan pengujian sistem untuk melihat hasil menggunakan *Faster Region Convolution Neural Network* untuk dapat menentukan jenis tanda tangan dan juga tingkat kemiripannya, sebagai contoh yaitu Tanda Tangan 1, Tanda Tangan 2, dan Tanda Tangan 3, Tanda Tangan 4, Tanda Tangan 5, Tanda Tangan 6, Tanda Tangan 7, Tanda Tangan 8, Tanda Tangan 9, Tanda Tangan 10.

**Tabel 4.2** Sampel Hasil Pengujian *Faster R-CNN*

No	Citra	Jenis Tanda Tangan	Tingkat Kemiripan(%)	Status
1		Tanda Tangan 1	91.6	True
2		Tanda Tangan 1	87.1	True
3		Tanda Tangan 1	92.5	True
4		Tanda Tangan 1	90.2	True
5		Tanda Tangan 1	-	False
6		Tanda Tangan 1	88.2	True
7		Tanda Tangan 2	90.5	True

No	Citra	Jenis Tanda Tangan	Tingkat Kemiripan(%)	Status
8		Tanda Tangan 2	91.9	True
9		Tanda Tangan 2	-	False
10		Tanda Tangan 2	88.6	True
11		Tanda Tangan 2	92.3	True
12		Tanda Tangan 2	89.1	True
13		Tanda Tangan 3	90.8	True
14		Tanda Tangan 3	92.0	True
15		Tanda Tangan 3	88.1	True
16		Tanda Tangan 3	93.4	True

No	Citra	Jenis Tanda Tangan	Tingkat Kemiripan(%)	Status
17		Tanda Tangan 3	94.0	True
18		Tanda Tangan 3	89.9	True
19		Tanda Tangan 4	89.6	True
20		Tanda Tangan 4	91.7	True
21		Tanda Tangan 4	93.6	True
22		Tanda Tangan 4	87.7	True
23		Tanda Tangan 4	91.0	True
24		Tanda Tangan 4	90.7	True
25		Tanda Tangan 5	87.2	True

No	Citra	Jenis Tanda Tangan	Tingkat Kemiripan(%)	Status
26		Tanda Tangan 5	-	False
27		Tanda Tangan 5	91.6	True
28		Tanda Tangan 5	92.9	True
29		Tanda Tangan 5	88.9	True
30		Tanda Tangan 5	90.5	True

**Tabel 4.3** Sampel Hasil *Confussion Matrix*

True Values	True	False	Total
Tanda Tangan 1	5	1	6
Tanda Tangan 2	5	1	6
Tanda Tangan 3	6	0	6
Tanda Tangan 4	6	0	6
Tanda Tangan 5	5	1	6



a. *Accuracy*

$$\begin{aligned}
 Accuracy &= \frac{\text{jumlah data yang benar}}{\text{jumlah keseluruhan data uji}} \times 100\% \\
 &= \frac{27}{30} \times 100\% \\
 &= 90\%
 \end{aligned}$$

b. *Precision*

$$\begin{aligned}
 Precision\ 1 &= \frac{5}{5 + 1 + 0 + 0 + 1} \times 100\% \\
 &= 71\%
 \end{aligned}$$

$$\begin{aligned}
 Precision\ 2 &= \frac{5}{5 + 1 + 0 + 0 + 1} \times 100\% \\
 &= 71\%
 \end{aligned}$$

$$\begin{aligned}
 Precision\ 3 &= \frac{6}{6 + 1 + 1 + 0 + 1} \times 100\% \\
 &= 66\%
 \end{aligned}$$

$$\begin{aligned}
 Precision\ 4 &= \frac{6}{6 + 1 + 1 + 0 + 1} \times 100\% \\
 &= 66\%
 \end{aligned}$$

$$\begin{aligned}
 Precision\ 5 &= \frac{5}{5 + 1 + 1 + 0 + 0} \times 100\% \\
 &= 71\%
 \end{aligned}$$

c. *Recall*

$$\begin{aligned}
 Recall\ 1 &= \frac{5}{5 + 1} \times 100\% \\
 &= 83\%
 \end{aligned}$$

$$\begin{aligned}
 Recall\ 2 &= \frac{5}{5 + 1} \times 100\% \\
 &= 83\%
 \end{aligned}$$

$$\begin{aligned}
 Recall\ 3 &= \frac{6}{6 + 0} \times 100\% \\
 &= 100\%
 \end{aligned}$$

$$\begin{aligned}
 Recall\ 4 &= \frac{6}{6 + 0} \times 100\%
 \end{aligned}$$

$$= 100\%$$

$$\begin{aligned} \text{Recall 5} &= \frac{5}{5+1} \times 100\% \\ &= 83\% \end{aligned}$$

d. *F1 Score*

$$\begin{aligned} \text{F1 Score 1} &= 2 \times \left( \frac{0,71 \times 0,83}{0,71 + 0,83} \right) \times 100\% \\ &= 76\% \end{aligned}$$

$$\begin{aligned} \text{F1 Score 2} &= 2 \times \left( \frac{0,71 \times 0,83}{0,71 + 0,83} \right) \times 100\% \\ &= 76\% \end{aligned}$$

$$\begin{aligned} \text{F1 Score 3} &= 2 \times \left( \frac{0,66 \times 1,00}{0,66 + 1,00} \right) \times 100\% \\ &= 79\% \end{aligned}$$

$$\begin{aligned} \text{F1 Score 4} &= 2 \times \left( \frac{0,66 \times 1,00}{0,66 + 1,00} \right) \times 100\% \\ &= 79\% \end{aligned}$$

$$\begin{aligned} \text{F1 Score 5} &= 2 \times \left( \frac{0,71 \times 0,83}{0,71 + 0,83} \right) \times 100\% \\ &= 76\% \end{aligned}$$

**Tabel 4.4** *Classification Report*

	Precision	Recall	F1-Score	Support
Tanda Tangan 1	0,71	0,83	0,76	6
Tanda Tangan 2	0,71	0,83	0,76	6
Tanda Tangan 3	0,66	1,00	0,79	6
Tanda Tangan 4	0,66	1,00	0,79	6
Tanda Tangan 5	0,71	0,83	0,76	6

## **BAB 5**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Didapatkan kesimpulan dari keseluruhan penelitian menggunakan metode *Faster Region Convolution Neural Network* hingga pengujian adalah sebagai berikut:

1. Implementasi *Faster Region Convolution Neural Network* menghasilkan akurasi mencapai 90% dengan rasio perbandingan sebesar 80:20 yang menggunakan data latih sebanyak 80% atau sebanyak 1200 citra tanda tangan dan data uji sebanyak 20% atau sebanyak 300 citra tanda tangan.
2. Pada percobaan menggunakan model dari jumlah data yang relatif sedikit yaitu sebanyak 45 citra tanda tangan dengan rasio pembagian citra 80:20 dengan data latih sebanyak 80% yang berjumlah 36 citra tanda tangan dan data uji sebanyak 20% yang berjumlah 9 citra tanda tangan dihasilkan akurasi sebesar 77% untuk verifikasi tanda tangan menggunakan *faster region convolution neural network*.

#### **5.2 Saran**

Adapun saran untuk pengembangan penelitian berikutnya adalah sebagai berikut:

1. Membuat aplikasi yang dapat melakukan deteksi dengan objek yang lebih beragam untuk meningkatkan keberagaman dan keuniversalan model.
2. Menggunakan data latih yang lebih besar untuk setiap jenisnya guna meningkatkan akurasi dan generalisasi model.
3. Membuat aplikasi yang dapat melakukan klasifikasi secara *real-time* untuk meningkatkan responsivitas dan kepraktisan penggunaan.

4. Mengevaluasi metode klasifikasi lainnya dan membandingkannya dengan hasil yang diperoleh dari metode *Faster Region Convolution Neural Network* untuk mendapatkan pemahaman lebih mendalam tentang keunggulan dan kelemahan masing-masing metode.

## DAFTAR PUSTAKA

- Ansari, M. A., Kurchaniya, D., & Dixit, M. 2017. A Comprehensive Analysis of Image Edge Detection Techniques. *International Journal of Multimedia and Ubiquitous Engineering*, 12(11), 1-12.
- Arifin, J., & Naf'an, M.Z. 2017. Verifikasi Tanda Tangan Asli Atau Palsu Berdasarkan Sifat Keacakan (Entropi).
- Azman, A. A., & Ismail, F. S. 2017. Convolutional Neural Network for Optimal Pineapple Harvesting. *Journal of Electrical Engineering*, 1-4.
- Batista, L., Granger, E., & Sabourin, R. 2012. Dynamic Selection of Generative-Discriminative Ensembles for Off-line Signature Verification. *Pattern Recognition*.
- Bhattacharya, I., Ghosh, P., & Biswas, S. 2017. Offline Signature Verification Using Pixel Matching Technique. International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA).
- Drott, B., & Hassan-Reza, T. 2015. On-line Handwritten Signature Verification using Machine Learning Techniques with a Deep Learning Approach.
- Gideon, J., Kandulna, A., Kujur, A.A., Diana, & Raimond, K. 2018. Handwritten Signature Forgery Detection using Convolutional Neural Networks. *8th International Conference on Advances in Computing and Communication (ICACC-2018)*.
- Girshick, R., Donahue, J., Darrell, T., & Malik, J. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE.
- Gonzalez, R. C., & Woods, R. E. 2017. Digital Image Processing. *Pearson Education*.
- Hafemann, L.G., Sabourin, R., & Oliviera, L.S. 2017. Learning Features for Offline Handwritten Signature Verification using Deep Convolutional Neural Networks.
- Hamza, R., & Chtourou, M. 2018. Apple Ripeness Estimation Using Artificial Neural Network. *Proceedings - 2018 International Conference on High Performance Computing and Simulation*, 229-234.
- Karpathy. 2018. *Introduction to Convolution Neural Network*.

- Mantoro, T., Ayu, M. A., & Suhendi. (2018). Multi-Faces Recognition Process Using Haar Cascades and Eigenface Methods. *2018 6th International Conference on Multimedia Computing and Systems (ICMCS)*. Rabat, Morocco: IEEE.
- Kaul, S. 2018. *Region Based Convolutional Neural Networks for object detection and recognition in ADAS application*.
- Kumar, G. R., Kumar, R. K., & Sanyal, G. 2017. Facial emotion analysis using deep convolution neural network. *2017 International Conference on Signal Processing and Communication (ICSPC)*. Coimbatore, India: IEEE.
- Kumar, T., & Verma, K. 2010. A Theory Based on Conversion of RGB image to Gray image. *International Journal of Computer Applications* , 7(2).
- Larkins, R., & Mayo, M. 2008. Adaptive Feature Thresholding for Off-Line Signature Verification. *In: Image and vision computing New Zealand*.
- Li, Y., Chen, J., & Shao, J. 2019. A Secure Proxy Blind Signature Scheme Based on Smart Contract. *IEEE Access*, 7, 29831-29839.
- Mulyanto, F. 2020. Aplikasi Pendeteksian Objek Buah-buahan Yang Memiliki Kemiripan Menggunakan Algoritma Faster R-CNN Berbasis Android. *Skripsi*. Medan. Universitas Sumatera Utara.
- Novandra, G., Naf'an, M.Z., & Laksana, T.G. 2018. Perancangan Aplikasi Android Identifikasi Tanda Tangan Menggunakan MultiLayer Perceptron.
- Pansare, A., & Bhatia, S. 2012. Handwritten Signature Verification using Neural Network. *International Journal of Applied Information Systems*.
- Poddar, J., Parikh, V., & Bharti, S.K. 2020. Offline Signature Recognition and Forgery Detection Using Deep Learning. *The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40)*.
- Prasanth, C.R., Raja, K.B., Venugopal, K.R., & Patnaik, L.M. 2009. Standard Scores Correlation based Off-line Signature Verification System. *2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies*.
- Rizki, Y., Taufiq, R. M., Mukhtar, H., & Putri, D. 2021. Klasifikasi Pola Kain Tenun Melayu Menggunakan Faster R-CNN. *IT Journal Research and Development (ITJRD)*, 5(2).
- Shah, A.S., Shah, M., Fayaz, & Wahid, F. 2017. Forensic Analysis of Offline Signatures Using Multilayer Perceptron and Random Forest. *Int. J. Database Theory Appl.*

- Viola P, Jones M, Rapid *Object detection using a boosted cascade of simple features*, *Proceeding of Conference on Computer Vision and Pattern Recognition*, 1, 2001, pp. 511-518.
- Widodo, A.W., & Harjoko, A. 2015. Sistem Verifikasi Tanda Tangan Off-line Berdasarkan Ciri Histogram of Oriented Gradient (HOG) dan Histogram of Curvature (HOC).



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,  
RISET, DAN TEKNOLOGI  
UNIVERSITAS SUMATERA UTARA  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

Jalan Universitas No. 9A Gedung A, Kampus USU Medan 20155, Telepon: (061) 821007  
Laman: <http://Fasilkomti.usu.ac.id>

KEPUTUSAN  
DEKAN FAKULTAS ILMU KOMPUTER  
DAN TEKNOLOGI INFORMASI  
NOMOR : 1963/UN5.2.14.D/SK/SPB/2024  
DEKAN FAKULTAS ILMU KOMPUTER  
DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA

- Membaca : Surat Permohonan Mahasiswa Fasilkom-TI USU tanggal 13 Juni 2024 perihal permohonan ujian skripsi:  
Nama : GILBERT SORAI ARO SIHURA  
NIM : 171402071  
Program Studi : Sarjana (S-1) Teknologi Informasi  
Judul Skripsi : Implementasi Faster Region Convolutional Neural Network Untuk Verifikasi Tanda Tangan Dan Tingkat Kemiripan Berbasis Mobile
- Memperhatikan : Bahwa Mahasiswa tersebut telah memenuhi kewajiban untuk ikut dalam pelaksanaan Meja Hijau Skripsi Mahasiswa pada Program Studi Sarjana (S-1) Teknologi Informasi Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara TA 2023/2024.
- Menimbang : Bahwa permohonan tersebut diatas dapat disetujui dan perlu ditetapkan dengan surat keputusan
- Mengingat : 1. Undang-undang Nomor 20 Tahun 2003 tentang Sistem Pendidikan Nasional.  
2. Peraturan Pemerintah Nomor 17 tahun 2010 tentang pengelolaan dan penyelenggara pendidikan.  
3. Keputusan Rektor USU Nomor 03/UN5.1.R/SK/SPB/2021 tentang Peraturan Akademik Program Sarjana Universitas Sumatera Utara.  
4. Surat Keputusan Rektor USU Nomor 1876/UN5.1.R/SK/SDM/2021 tentang pengangkatan Dekan Fasilkom-TI USU Periode 2021-2026

MEMUTUSKAN

- Menetapkan :  
Pertama : Membentuk dan mengangkat Tim Penguji Skripsi mahasiswa sebagai berikut:  
Ketua : Dr. Erna Budhiarti Nababan M.IT  
NIP: 196210262017042001  
Sekretaris : Ivan Jaya S.Si., M.Kom.  
NIP: 198407072015041001  
Anggota Penguji : Dr. Romi Fadillah Rahmat, B.Comp.Sc., M.Sc.  
NIP: 198603032010121004  
Anggota Penguji : Rossy Nurhasanah S.Kom., M.Kom  
NIP: 198707012019032016  
Moderator : -  
Panitera : -
- Kedua : Segala biaya yang diperlukan untuk pelaksanaan kegiatan ini dibebankan pada Dana Penerimaan Bukan Pajak (PNPB) Fasilkom-TI USU Tahun 2024.
- Ketiga : Keputusan ini berlaku sejak tanggal ditetapkan dengan ketentuan bahwa segala sesuatunya akan diperbaiki sebagaimana mestinya apabila dikemudian hari terdapat kekeliruan dalam surat keputusan ini.

- Tembusan :
- 1. Ketua Program Studi Sarjana (S-1) Teknologi Informasi
  - 2. Yang bersangkutan
  - 3. Arsip

Medan, 14 Juni 2024  
Ditandatangani secara elektronik oleh:  
Dekan



Maya Silvi Lydia  
NIP 197401272002122001