USULAN PENELITIAN S2

PERBAIKAN PENGENALAN GESTUR TANGAN TERHADAP KONDISI LINGKUNGAN BERINTENSITAS CAHAYA RENDAH MENGGUNAKAN RETINEX

ANTHONIUS ADI NUGROHO
19/448690/PPA/05773

PROGRAM MAGISTER ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
YOGYAKARTA

2020

HALAMAN PENGESAHAN USULAN PENELITIAN S2

Judul Tesis : Perbaikan Pengenalan Gestur Tangan Terhadap Kondisi Lingkungan Berintensitas Cahaya Rendah Menggunakan Retinex

Nama Mahassiwa : Anthonius Adi N

NIM : 19/448690/PPA/05773

Proposal telah diuji pada tanggal _____ dan sudah diperbaiki sesuai saran penguji dan sudah disetujui para penguji.

Nama Penguji	**Tanda Tangan**
1. .	1.
2. .	2.

Yogyakarta, ...2020

Mengetahui,
Pembimbing

Pengusul

Dr. Raden Sumiharto, S.Si., M.Kom.
NIP. 197706252005011001

Anthonius Adi N.
NIM. 19/448690/PPA/05773

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

INTISARI

PERBAIKAN PENGENALAN GESTUR TANGAN TERHADAP KONDISI LINGKUNGAN BERINTENSITAS CAHAYA RENDAH MENGGUNAKAN RETINEX

Oleh

ANTHONIUS ADI NUGROHO

19/448690/PPA/05773

Bahasa isyarat adalah salah satu bentuk komunikasi non-verbal antar manusia yang memiliki makna tersendiri. *American Sign Language*(ASL) merupakan bentuk dari bahasa isyarat tangan yang digunakan oleh penyandang disabilitas untuk berkomunikasi satu sama lain. Penggunaan bahasa isyarat membantu penyetaraan hak atas informasi yang diberikan. Perkembangan ilmu yang sangat maju membantu seseorang dapat memahami suatu bahasa isyarat tanpa harus mempelajari hal tersebut. Salah satu bagian disiplin ilmu tersebut adalah pengenalan gestur, dimana di dalamnya terdapat bahasa isyarat yang dapat diterjemahkan dengan bantuan kamera.

Kamera digunakan sebagai alat yang mengimitasi mata manusia dalam mengenali sebuah gestur. Proses pengenalan bahasa isyarat memiliki beberapa hal penting di dalamnya. Intensitas cahaya merupakan salah satu faktor penting dalam pengambilan sebuah citra yang ditangkap oleh kamera, dimana merupakan suatu tantangan tersendiri ketika informasi citra tidak dapat ditangkap dengan jelas yang menyebabkan penurunan performa. Permasalahan ini dapat diatasi dengan menggunakan teori *Retinex* yang diambil dari kata retina dan cortex. *Retinex* mampu meningkatkan tingkat kecerahan citra dengan cara layaknya mata manusia yang dapat melihat walaupun dalam ruangan minim cahaya. Citra yang telah dilakukan perbaikan kontras akan dilakukan deteksi tangan kemudian akan dikenali bahasa isyarat tersebut menggunakan teknik *Convolution Neural Network*.

Kata kunci - Retinex, Hand Gesture Recognition, Object Detection

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Disabilitas adalah kelompok masyarakat yang memiliki keterbatasan yang dapat menghambat partisipasi dan peran serta mereka dalam kehidupan bermasyarakat. Penyandang disabilitas memiliki berbagai kategori, yaitu disabilitas fisik, intelektual, mental dan sensorik. menurut data dari kemenkes tahun 2015, presentase 3 teratas penyandang disabilitas di provinsi indonesia adalah 6.36% kesulitan melihat, 3.76% kesulitan berjalan dan 3.35% kesulitan mendengar (Kemenkes., 2018). Pemerintah Indonesia telah menandatangani konvensi tentang Hak-Hak Penyandang Disabilitas pada tanggal 30 Maret 2007 di New York. Adanya penandatanganan tersebut menunjukan bangsa indonesia menghormati, melindungi, memenuhi dan memajukan hak-hak penyandang disabilitas. Untuk itu perlu adanya dukungan dari masyarakat dalam mewujudkanya.

Bagi penyandang disabilitas, mereka memiliki hambatan akses dalam melakukan aktivitas sehari hari. Dengan adanya perkembangan pengetahuan dan teknologi, mereka mulai terbantu dan dapat melakukan aktivitas layaknya masyarakat pada umumnya. Mulai banyak penyandang disabilitas yang melakukan mobilitas tinggi dengan kursi roda, mengakses informasi dengan adanya penerjemah bahasa. Dengan adanya hal tersebut, penyandang disabilitas mendapat tempat dan peranan yang sama dengan masyarakat lainnya.

Teknologi yang semakin dewasa membuat mobilitas penyandang disabilitas menjadi lebih tinggi. Kursi roda mungkin dapat digunakan untuk membantu penyandang bergerak dari suatu tempat ke tempat lain, namun bagi beberapa penyandang disabilitas tertentu yang tidak memiliki kemampuan normal pada kondisi tangan atau lumpuh sebagian tidak dapat menggunakan kursi roda tersebut. Alhasil perlu adanya orang lain untuk membantu menngerakan kursi roda tersebut.

Teknologi komputer dan robotika saat ini memiliki peranan penting dalam membantu sebuah permasalahan dari mulai kegiatan industri hingga kegiatan masyarakat. (Posada-Gómez et al., 2007) Membuat kursi roda pintar dengan kontrol

gestur tangan, namun pada penelitian tersebut memiliki kelemahan terhadap cahaya. Intensitas cahaya yang cenderung rendah membuat sistem tidak mampu mendeteksi kontrol dari gestur tangan, sehingga hanya dapat digunakan dalam keadaan cahaya yang cenderung terang. Penelitian tersebut menggunakan teknologi pemrosesan citra yang dikombinasikan dengan elektronika dan mekanika untuk pergerakan kursi roda.

Penelitian yang menggunakan variasi cahaya juga dilakukan oleh (Saputra., 2016) dengan variasi metode *Retinex* dan variasi intensitas cahaya 439,75 lux, 273,25 lux, 150 lux dan 9 lux. Penelitian yang dilakukan untuk meningkatkan akurasi deteksi wajah menggunakan *adaboost haar-like*. Hasil pengujian pada penelitian ini menghasilkan nilai akurasi dengan rata-rata tertinggi dari 10 kali uji 96,67% pada kondisi 439,75 lux, 90,59% pada kondisi 273,25 lux, 42,29% pada kondisi 150 lux dan 0% pada kondisi 9 lux.

Pemrosesan citra memiliki beberapa hal fundamental permasalahan diantaranya adalah proses perbaikan citra. Perbaikan citra digunakan untuk memperbaiki sebuah citra yang bermasalah agar informasi citra terlihat lebih jelas secara visual maupun perhitungan. Penggunaan *image processing* sangat dibutuhkan untuk membantu menyelesaikan permasalahan dalam kehidupan sehari hari. Aplikasi dari implementasi *image processing* beberapa diantaranya adalah pengenalan dan deteksi pada sebuah objek. Untuk menyelesaikan permasalahan tersebut perlu membuat sistem yang tahan terhadap kondisi cahaya berintensitas rendah.

Pengenalan dan deteksi sebuah objek memiliki ruang lingkup yang sangat luas untuk di kembangkan, pada penelitian ini berfokus pada gesture recognition. Dalam pemrosesan citra algoritme pada gesture recognition dapat di implementasikan pada komunikasi non verbal ataupun suatu gerakan yang dapat membantu seseorang menyelesaikan permasalahan. Pola pada gesture dapat dikenali oleh seseorang dengan cara melihat gesture tersebut, hal yang sama terjadi pada kamera yang mengadopsi apa yang dilakukan mata manusia untuk mengenali sebuah objek. *Hand gesture* salah satu contoh implementasi *image processing* yang dapat dikembangkan untuk membantu seseorang menyelesaikan permasalahan dengan meng-

gunakan gestur tangan.

Salah satu faktor yang dapat menurunkan kualitas suatu citra yaitu pencahayaan dari sebuah citra, proses pengambilan citra dalam intensitas rendah akan menghasilkan citra yang buruk (Saputra., 2016). Implementasi algoritme untuk meningkatkan kualitas citra terhadap kondisi cahaya merupakan permasalahan yang menarik untuk diteliti. Kondisi cahaya pada image processing adalah sesuatu topik yang menantang dalam permasalahan *image processing* dan *computer vision* untuk meningkatkan visibilitas maupun kualitas yang lebih baik dari suatu citra. Beberapa penelitian terdahulu telah melakukan peningkatan algoritme pada kondisi cahaya yang minim untuk suatu citra (Loh et al., 2019). Berkembangnya algoritme dalam ruang lingkup kondisi cahaya yang minim bertujuan mendapatkan kualitas kontras yang lebih jelas untuk dapat dilakukan komputasi lebih lanjut.

Retinex merupakan metode yang diusulkan oleh Land dengan memodelkan pencahayaan dan persepsi warna berdasarkan penglihatan mata manusia. Mata manusia dapat membedakan sebuah objek sekalipun dalam kondisi intensitas cahaya yang rendah. Metode *Retinex* terus mengalami berbagai pengembangan dari *Single Scale Retinex* hingga *Multiscale Retinex* berupaya untuk memperoleh keseimbangan kontras dalam pencahayaan berintensitas rendah(Saputra., 2016).

Peningkatan kontras menjadi salah satu solusi untuk meningkatkan visibilitas dari sebuah citra. Penelitian ini menggunakan algoritme *Multiscale Retinex Color Restoration* untuk meningkatkan kontras serta *object detection* untuk melakukan segmentasi pada suatu citra, kemudian untuk melakukan pengenalan sebuah gesture yang telah dilakukan perbaikan kontras, akan dilanjutkan dengan metode *Convolutional Neural Network*.

Implementasi dari peningkatan kontras untuk pengenalan gestur tangan ini diharapkan dapat dibawa untuk menyelesaikan permasalahan pada sistem yang bergantung pada cahaya. Beberapa sistem yang dimaksud seperti kursi roda pintar, ataupun sistem lainnya seperti pengenalan wajah dan pengenalan gestur maupun pendeteksi suatu objek yang menggunakn kamera.

1.2. Rumusan Masalah

Berdasarkan rumusan masalah yang telah dijelaskan sebelumnya, maka rumusan masalah pada penelitian ini adalah nilai akurasi dari deteksi wajah dengan intensitas cahaya 150 lux hanya mampu 42,39%, sehingga deteksi dan pengenalan sebuah objek sangat dipengaruhi oleh cahaya. Intensitas cahaya yang semakin rendah menyebabkan penurunan akurasi suatu deteksi dan pengenalan. Hal tersebut dapat berdampak pada performa sebuah sistem yang semakin menurun.

1.3. Batasan Masalah

Penelitian ini memiliki batasan masalah yang bertujuan untuk tidak memperluas pokok bahasan. Batasan masalah yang digunakan dalam penelitian ini adalah sebagai berkut:

1. Acuan dataset menggunakan *American Sign Language*, dengan sepuluh klasifikasi yaitu angka 0 hingga angka 9.
2. Data yang digunakan menggunakan dataset private dan dataset public.
3. Proses pengujian akan dilakukan dengan 3 subjek dengan warna kulit yang berbeda.
4. Penurunan intensitas cahaya dikurangi sebesar 50% lux sebelumnya hingga nilai lux kurang dari 50 lux.

1.4. Tujuan Penelitian

Penelitian yang dilakukan bertujuan untuk mengatasi permasalahan sistem yang bergantung pada pencahayaan dengan mengimplementasikan algoritme *Retinex* pada sebuah sistem deteksi dan pengenalan gestur tangan, dengan tingkat intensitas cahaya yang bervariasi.

1.5. Manfaat Penelitian

Penelitian ini diharapkan dapat memberi manfaat berupa:

1. Memberikan fitur tambahan untuk sistem yang memiliki permasalahan dengan pencahayaan dalam intensitas cahaya rendah.

2. Meningkatkan akurasi pengenalan gestur tangan dan deteksi objek pada kondisi cahaya berintensitas rendah.

BAB II
TINJAUAN PUSTAKA

Penelitian untuk menghasilkan algoritme deteksi dan pengenalan gestur sudah banyak dilakukan, sehingga banyak metode yang memiliki tingkat keberhasilan tinggi. Namun pada pemrosesan citra tingkat keberhasilan tidak hanya didukung pada algoritme deteksi dan pengenalan saja, namun kualitas pada citra tersebut harus memiliki kualitas yang bagus pula sehingga dapat menghasilkan akurasi yang tinggi(Saputra., 2016).

Salah satu hal yang mempengaruhi kualitas tersebut adalah kondisi cahaya yang mengarah pada objek, sehingga mempengaruhi hasil yang ditangkap oleh kamera. Peningkatan kontras sangat dibutuhkan untuk meningkatkan kualitas image dalam kondisi cahaya yang minim. Salah satu pendekatan yang populer dipakai adalah Retinex. (Tanaka et al., 2019) mengimplementasikan Retinex pada prepocessing untuk meningkatkan kontras citra. Percobaan yang dilakukan dengan membandingkan hasil segmentasi citra asli dan citra dengan proses preprocessing. Hasil yang didapat secara kualitatif terlihat setelah dilakukan segmentasi Gaussian Mixture Model. Hasil segmentasi objek dengan citra preprocessing mendapatkan foreground yang jelas daripada citra asli. Namun pada citra dengan Retinex peningkatan kontras berubah menghasilkan warna yang tidak natural.

Peningkatkan pencahayaan pada citra dapat dilakukan dengan banyak metode. Selain Retinex, metode yang paling populer adalah *Histogram Equalization*. (Srinivasan, 2016) Melakukan perbandaingan antara algoritme *Retinex* dan *Histogram Equalization*. Pada penelitian tersebut *Retinex* yang di implementasikan adalah *Single Scale Retinex* (SSR) dan *Multiscale Retinex* (MSR). Kemudian *Histogram Equalization* yang diimplementasikan adalah BBHE, DSIHE dan RLBHE Kedua algoritme tersebut diuji menggunakan citra yang sama kemudian dilakukan perbandingan.

Pada citra keluaran *Histogram Equalization* (BBHE dan DSIE), citra mengalami meningkatkan kontras dengan keterbatasan beberapa fitur yang tidak terlihat, Keluaran RLBHE mereduksi kualitas dari piksel. Kemudian dengan citra keluaran

Retinex, citra mengalami peningkatan kontras dengan fitur yang terlihat lebih jelas daripada citra *Histogram Equalization*.

Algoritme Retinex telah mengalami pengembangan untuk meningkatkan kualitas citra pada kondisi lingkungan tertentu. Saputra ditahun 2016 melakukan perbandingan variasi Retinex untuk peningkatan deteksi wajah yang dilakukan pada kondisi ruangan berintensitas rendah. Algoritme *Adaptive Multiscale Retinex* (AMSR), *Multiscale Retinex Color Restoration*(MSRCR) di implementasikan pada 4 kondisi cahaya. Hasil peningkatan maksimal yang didapatkan pada MSRCR dapat meningkatkan 1,46 kali dan ASMR mampu meningkatkan 1,11 kali. Namun peningkatan tersebut terjadi pada parameter intensitas 273,25 lux(Saputra., 2016).

Pengembangan MSR juga dilakukan oleh (Shen et al., 2017) dengan menambah layer preprocessing/post processing pada proses konvolusi citra pada CNN menjadi MSR-Net. Kemudian dibandingkan dengan MSRCR dan beberapa metode lainnya. Hasil MSR-Net mengalami peningkatan kontras citra dengan warna natural dibandingkan MSRCR dan beberapa metode lainnya.

Pengenalan sebuah gestur adalah yang menentukan hasil akhir dari sistem. CNN merupakan salah satu algoritme yang sering dijadikan solusi untuk proses klasifikasi dalam *machine learning*. Beberapa penelitian menggunakan CNN dengan beberapa variasi parameter. Penelitian (Yingxin et al., 2017) menggunakan CNN untuk mengenali sebuah gestur tangan dengan dataset *Cambridge hand gesture datasets* (CHGD). Pada penelitian ini dilakukan parameter illuminasi cahaya pada beberapa kondisi. Pengenalan gestur menghasilkan angka presentase yang tinggi sebesar 94.1%. Proses dalam preprocessing menggunakan *canny edge* untuk menghilangkan efek illuminasi. *Canny edge* sangat membantu untuk kondisi illuminasi cahaya dibandingkan dengan algoritme CNN saja yang menggunakan citra asli di dapat hasil 70.0%.

Pada citra gelap informasi atau fitur fitur penting dari sebuah citra akan tersembunyi. Informasi dalam sebuah citra penting untuk merepresentasikan sebuah citra itu sendiri. Untuk mendapatkan informasi tersebut pada penelitian (Loh et al., 2019) berfokus pada perbaikan citra dengan tujuan memperoleh fitur untuk men-

dukung sistem visi otomatis dimana sebuah citra memiliki kontras dan pencahayaan yang rendah. Dalam penelitian ini memodelkan sebuah citra dengan cahaya rendah sebagai distribusi peningkatan fungsi lokal menggunakan proses gaussian yang dilatih pada saat runtime menggunakan data referensi yang dihasilkan dari sebuah CNN. CNN sendiri dilatih menggunakan dengan data yang sangat besar berdasarkan statistik pencahayaan. Sehingga proses pembelajaran dapat mempelajari hubungan antara fitur dengan piksel. Dengan demikian refrensi yang dihasilkan melatih gaussian proses untuk melakukan representasi fitur dengan benar.

Dasar-dasar penelitian sebelumnya yang menjadi tinjauan pustaka pada penelitian ini dirangkum dalam Tabel 2.1.

Tabel 2.1 Tinjauan Pustaka

No	Nama	Penelitian	Metode	Hasil
1	(Loh et al., 2019)	*Low-light image enhancement using Gaussian Process for features retrieval*	*Gaussian process and Convolutional Neural Network*	Citra dengan kontras yang sangat rendah dapat di perbaiki menggunakan gaussian prosses dan CNN untuk memperoleh detail informasi dari sebuah objek
2	(Tanaka et al., 2019)	*Retinex Foreground Segmentation for Low Light Environments*	*Retinex* dan *Gaussian Mixture Model*	Secara kualitatif citra yang dihasilkan setelah melalui *preprocessing* algoritme Retinex dapat meningkatkan pencahayaan dari sebuah citra
3	(Yingxin et al., 2017)	*A Robust Hand Gesture Recognition Method via Convolutional Neural Network*	*Edge detection* dan *Convolutional Neural Network*	Pengenalan gestur tangan mendapatkan hasil 94.1% digabungkan dengan proses deteksi tepi.

Tabel 2.2 Lanjutan Tabel

4	(Shen et al., 2017)	*MSR-Net:Low-light Image Enhancement using Deep Convolutional Network*	MSR-net	Hasil dari implementasi algoritme MSR-Net di bandingkan dengan MSRCR dan beberapa algoritme lain mendapatkan kontras yang lebih tinggi dengan warna yang natural dibandingkan algoritme lain.
5	(Saputra., 2016)	Perbandingan Varian *Metode Multiscale Retinex* Untuk Peningkatan Akurasi Deteksi Wajah *Adaboost HAAR-like*	Variasi metode *Multiscale Retinex*	Kondisi 439,75 lux MSRCR meningkatkan akurasi 1,31 kali dan AMSR hanya 1,11 kali. Kondisi 273,25 lux MSRCR 1,46 kali dan AMSR 1,31 kali. Kondisi 150 lux MSRCR 1,38 kali dan AMSR 0,97 kali. Kondisi 9 lux kedua algoritme tidak dapat mendeteksi wajah sebuah citra.
6	(Srinivasan, 2014)	Perbandingan antara *Retinex* dan *Histogram Equalization*	SSR, MSR, BBHE, DSIHE, RLBHE	Citra keluaran *Retinex* memiliki *output* kontras yang baik tanpa menghilangkan fitur pada citra.
7	(Arabi, 2019)	Mendeteksi peralatan konstruksi yang diaplikasikan pada embedded system dan PC	*SSD MobileNet*	Untuk semua device mAP $> 90\%$
8	(Huang et al., 2019)	*Hand Gesture Recognition with Skin Detection and Deep Learning Method* Hand Gesture Recognition with Skin Detection and Deep Learning Method	*Skin detection* dan *Convolutional Neural Network*	Menghasilkan akurasi 98,41%

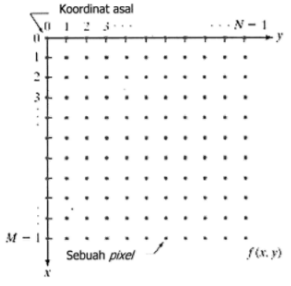
BAB III
LANDASAN TEORI

3.1. Computer Vision

Computer vision adalah teknologi yang membuat computer dapat melihat layaknya mata manusia. Penggunaan computer vision tanpa kita sadari telah di implementasikan dalam membantu menyelesaikan persoalan sehari hari. Contoh implementasi *computer vision* antara lain *face recognition, object classification, medical imaging,gesture recognition, video surveillance , 3D reconstruction*. Pada pengolahan citra, sebuah citra memiliki fitur fitur dari penting yang digunakan sebagai informasi saat pengolahan. Namun untuk mendapatkan suatu hasil, beberapa tahap proses harus dilakukan seperti *preprocessing*, ekstraksi ciri, *post-processing* dan sebagainya.

3.2. Citra Digital

Citra dapat didefinisikan sebagai fungsi $f(x, y)$ berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial dan amplitudo f di titik koordinat (x, y) dinamakan tingkat keabuan dari suatu citra. Apabila nilai (x, y) dan f secara keseluruhan berhingga dan bernilai diskrit maka citra tersebut adalah citra digital. Citra digital dalam bentuk matrik dapat dilihat pada Persamaan 3.1 dan posisi koordinat citra digital dapat dilihat pada gambar 3.1(Sarifudin.,2015).



Gambar 3.1 Koordinat Citra Digital (Sarifudin.,2015)

$$f(x,y)\begin{bmatrix} f(0,0) & f(0,1) & ... & f(0,n-1) \\ f(1,0) & f(1,1) & ... & f(1,n-1) \\ ... & ... & ... & ... \\ f(m-1,0) & f(m-1,1) & ... & f(m-1,n-1) \end{bmatrix} \quad (3.1)$$

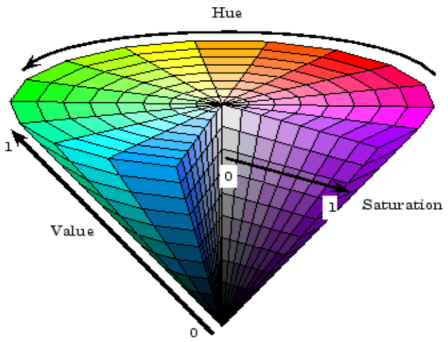
3.2.1. Citra RGB

Citra RGB disebut juga sebagai citra berwarna, citra ini menyajikan tiga layer warna yaitu Red, Green dan Blue. Setiap piksel dari citra RGB merupakan gabungan dari variasi nilai intensitas tiga warna dasar yaitu merah(R), hijau(G), biru(B). Tiga warna tersebut dikodekan dengan 8 bit,dengan total ketiganya 3 x 8 = 24 bit. Sehingga variasi warna sebanyak 2^{24} = 16.777.216 variasi warna (Sarifudin., 2015). Contoh citra RGB dapat dilihat pada Gambar 3.2.

Gambar 3.2 Citra RGB

3.2.2. Citra HSV

Ruang warna HSV memiliki tiga komponen warna, yaitu Hue(H), Saturation(S) dan Value(V). Hue memiliki variasi warna yang digambarkan secara melingkar, dimana merepresentasikan warna dari merah, kuning, hijau, cyan, biru, magenta,dan kembali lagi ke merah. Saturation memiliki variasi nilai 0 hingga 1, dimana merepresentasikan saturasi warna dari merah ke merah muda dan Value memiliki nilai 0 hingga 1 yang merepresentasikan intensitas warna atau tingkat kecerahan dari hitam ke putih, dimana nilai semakin tinggi semakin cerah (Kolkur et al., 2017). Ruang warna HSV dapat di gambarkan seperti Gambar 3.3.



Gambar 3.3 Ruang warna HSV (Kolkur et al., 2017)

Pada RGB suatu warna di representasikan dengan nilai tiap komponen, namun untuk ruang warna HSV memiliki nilai yang berbeda dari RGB. Hue memiliki range dari 0° sampai 360°. Transformasi warna dari RGB ke HSV dapat dilihat dalam Persamaan 3.2, 3.3, 3.4, 3.5.

$$V = max(R, G, B) \tag{3.2}$$

$$S = \begin{cases} \frac{max(R,G,B) - min(R,G,B)}{max(R,G,B)} & V \neq 0 \\ 0 & lainnya \end{cases} \tag{3.3}$$

$$H = \begin{cases} \frac{60(G-B)}{V - min(R,G,B)} & V = R \\ 120 + \frac{60(B-R)}{V - min(R,G,B)} & V = G \\ 240 + \frac{60(R-G)}{V - min(R,G,B)} & V = B \end{cases} \tag{3.4}$$

$$H = H + 360 \tag{3.5}$$

Penggunaan citra dengan ruang HSV mampu memisahkan informasi warna sesuai dengan kemampuan mata manusia(Afrianto & Amalia., 2016).

3.3. Operasi Morfologi

3.3.1. Erosi

Operasi erosi adalah operasi penipisan objek yang terdapat pada citra biner. Operasi erosi dilakukan dengan cara mengurangi piksel pada kontir dari objek citra sesuai dengan kernel. Operasi erosi dinotasikan pada Persamaan 3.6 (Hidayatullah., 2017).

$$A \ominus B = A^c \oplus B^c \tag{3.6}$$

3.3.2. Dilasi

Operasi dilasi adalah operasi penebalan objek yang terdapat pada citra biner. Operasi ini dilakukan dengan menambah piksel pada kontur dari objek sesuai dengan kernel. Operasi ini berguna untuk menghaluskan citra dan menutupi lubang-lubang yang kosong. Operasi dilasi dinotasikan pada Persamaan 3.7 (Hidayatullah., 2017).

$$A \oplus B = t \, \epsilon \, Z^2 : t = a + b, a \, \epsilon \, A, \; b \, \epsilon \, B \tag{3.7}$$

3.3.3. *Opening*

Operasi *opening* merupakan operasi yang biasa digunakan untuk memperhalus kontur citra serta menghilangkan lubang-lubang kecil pada citra. Operasi ini terdiri dari 2 dua tahap yaitu erosi kemudian dilanjutkan dilasi. Operasi erosi berguna untuk menghilangkan noise pada citra karena struktur latar depan yang berukuran kecil tereliminasi, sedangkan dilasi digunakan untuk menebalkan citra. Operasi opening dinotasikan dalam Persamaan 3.8(Hidayatullah., 2017).

$$A \bullet B = (A \ominus B) \oplus B \tag{3.8}$$

3.3.4. *Closing*

Operasi *closing* merupakan kebalikan dari operasi *opening*. Terdiri dari dua tahap yaitu operasi dilasi kemudian dilanjutkan dengan operasi erosi. Kegunaan operasi ini adalah untuk menutupi lubang yang kosong pada citra dengan menggunakan dilasi kemudian dilakukan erosi citra untuk menipiskan suatu citra. Operasi closing dinotasikan pada Persamaan 3.9 (Hidayatullah., 2017).

$$A \bullet B = (A \oplus B) \ominus B \tag{3.9}$$

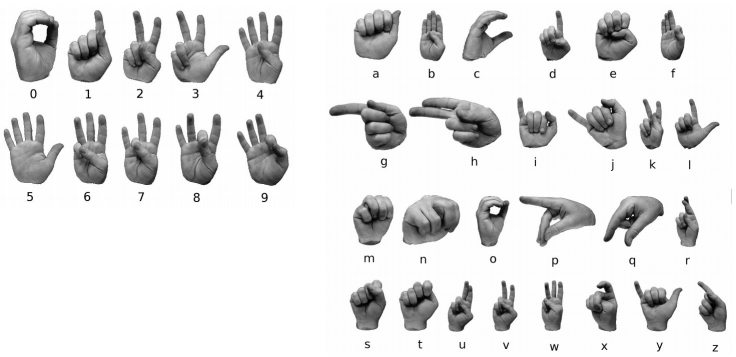
3.4. *Hand Gesture Recognition*

3.4.1. *American Sign Language*(ASL)

Bahasa isyarat merupakan media komunikasi utama bagi kaum difabel khusunya penyandang tunarungu di dunia. Setiap negara mempunyai bahasa isyarat

masing masing. Dengan adanya bahasa isyarat, penyandang tunarungu dapat melakukan komunikasi antar sesama penderita maupun berkomunikasi dengan setiap orang. Bahasa isyarat mulai banyak digunakan pada siaran televisi, seperti acara debat ataupun channel berita dengan dibantu penerjemah bahasa isyarat. Dengan begitu memberi mereka hak yang sama untuk mendapatkan informasi.

Salah satu bahasa isyarat yang digunakan adalah *American Sign Language*(ASL). *American Sign Language* menjadi salah satu alat bantu pembelajaran komunikasi untuk penggunanya. penggunaan bahasa isyarat dilakukan dengan gerakan gestur tangan yang memiliki makna untuk setiap pose. Pada penelitian ini, bahasa isyarat yang digunakan untuk menguji sistem adalah *American Sign Language*. Pada gestur ASL, mempunyai beberapa bentuk gestur tangan huruf dan angka seperti pada Gambar 3.4.



Gambar 3.4 American Sign Language (Barczak et al., 2011)

3.5. Retinex(Single Scale Retinex)

Algoritme *Retinex* merupakan algoritme yang berusaha untuk mempertahankan ketetapan warna dimana warna suatu objek yang dilihat dalam keadaan pencahayaan yang berbeda(Aribowo et al., 2009). Algoritme *Retinex* ini sering juga disebut dengan *Single Scale Retinex* (SSR) karena hanya memiliki satu kanal. Menurut Land sebuah citra terbentuk sebagai perkalian reflektansi dan iluminasi. Berdasarkan teori *Retinex* dapat dituliskan secara matematis seperti Persamaan 3.10 berikut(Parihar & Singh., 2018)(Petro et al., 2014):

$$I(x, y) = L(x, y) \cdot R(x, y) \tag{3.10}$$

$$\log I(x,y) = \log L(x,y) + \log R(x,y) \tag{3.11}$$

$$\log R(x,y) = \log I(x,y) - \log L(x,y) \tag{3.12}$$

dimana

$$L(x,y) = F(x,y) * I(x,y) \tag{3.13}$$

$$F(x,y) = \frac{1}{2\pi\sigma^2} e^{-(\frac{x^2+y^2}{2\sigma^2})} \tag{3.14}$$

$$\int F(x,y)dxdy = 1 \tag{3.15}$$

sehingga

$$\log R = \log I(x,y) - \log[F(x,y) * I(x,y)] \tag{3.16}$$

Keterangan :

$I(x,y)$: representasi dari kumpulansinyal dari citra asli.

$R(x,y)$: representasi dari komponen reflektansi dari objek.

$L(x,y)$: representasi komponen pencahayaan yang memenuhi Persamaan 3.12.

$F(x,y)$: *gaussian function* yang memenuhi Persamaan 3.13.

Bentuk logaritmik pada Persamaan 3.11 digunakan untuk memisahkan antara komponen pencahayaan dan komponen reflektansinya. Pemisahan ini dilakukan untuk mendapatkan informasi dari citra tersebut sehingga hasil *Single Scale Retinex* dapat dilihat pada Persamaan 3.17, dimana $\log R = R_{SSRi}$ adalah hasil *Single Scale Retinex* pada kanal i (Parihar & Singh., 2018)(Petro et al., 2014).

$$R_{SSRi}(x,y) = \log I_i(x,y) - \log[F(x,y) * I_i(x,y)] \tag{3.17}$$

3.5.1. Multiscale Retinex(MSR)

Multiscale Retinex merupakan pengembangan dari *Single Scale Retinex* dengan menggabungkan beberapa scale yang berbeda dengan pembobotan tertentu, dimana bobot tersebut jika dijumlahkan menghasilkan nilai 1. Sehingga nilai sangat mempengaruhi detail warna dari sebuah citra. Bentuk matematis dari *multiscale Retinex* dapat dilihat pada Persamaa 3.15 dan 3.16 berikut(Petro et al., 2014):

$$R_{MSRi} = \sum_{n=1}^{N} \omega_n R_{SSRi} = \omega_n[\log I(x,y) - \log(F_n(x,y) * I_i(x,y))] \tag{3.18}$$

Keterangan :

R_{MSRi} : hasil *Multiscale Retinex* pada kanal i

ω_n : bobot pada skala n

Pada SSR dapat menyediakan kompresi rentang dinamis dan penampakan warna, namun keduanya tidak secara bersamaan. Oleh karena itu *Multiscale Retinex* menggabungkan kompresi rentang dinamis dari *Single Scale Retinex* dengan penampakan warna(Aribowo et al., 2009).

3.5.2. Multiscale Retinex Color Restoration(MSRCR)

MSRCR merupakan pengembangan dari metode MSR yang mampu memperbaiki kualitas citra yang berhubungan dengan pencahayaan yaitu dengan mempertahankan *color constancy*. *Color constancy* berfungsi untuk mempertahankan komposisi warna suatu citra tetap terlihat sama walaupun kondisi pencahayaan yang berbeda-beda (Saputra., 2016). Perhitungan MSRCR dapat dilihat pada Persamaan 3.19, 3.20, 3.21 dan 3.22(Petro et al., 2014).

$$R'_{MSRCRi}(x,y) = C_i(x,y)\ R_{MSR_i}(x,y) \tag{3.19}$$

$$C_i(x,y) = \beta \log[\alpha I'_i(x,y)] \tag{3.20}$$

$$I'(x,y) = \frac{I_i(x,y)}{\sum_{i=0}^{S} I_i(x,y)} \tag{3.21}$$

$$R_{MSRCRi}(x,y) = G\ [R'_{MSRCRi}(x,y) - b] \tag{3.22}$$

Keterangan :

i = 3 kanal warna RGB

α = konstanta kontrol ketidaklinieran

β = konstanta gain

G,b = gain dan offset

3.6. MobileNet

MobileNet merupakan salah satu arsitektur CNN yang dibangun oleh *Google* dan dikhususkan untuk *mobile device* karena proses komputasi yang ringan.

Perbedaan arsitektur *MobileNet* dan CNN pada umumnya terdapat pada penggunaan layer konvolusi dengan ketebalan sesuai input dari citra. Arsitektur SSD *MobileNet* menggunakan *Depthwise layer* dan *Pointwise layer*.

3.6.1. Depthwise Separable Convolution

Depthwise separable convolution merupakan kunci utama untuk membangun arsitektur *neural network*. Ide utamanya adalah mengganti operator konvolusi secara menyeluruh dengan memisahkan konvolusi menjadi dua lapisan terpisah. Layer pertama adalah *depthwise convolution*, layer ini memiliki *light-weight filter* dengan menggunakan *single convolutional filter* untuk setiap input channel. Layer kedua menggunakan 1x1 convolution yang disebut dengan *pointwise convolution* dimana layer ini membentuk fitur fitur baru melalui perhitungan kombinasi linear dari input channel.

Pada konvolusi biasa memiliki input citra D_F x D_F x M fitur map pada F dan menghasilkan D_G x D_G x N fitur map pada G, dimana D_F adalah tinggi dan lebar dari fitur map. M merupakan jumlah input channel(input depth). D_G merupakan tinggi dan lebar keluaran fitur map dan N adalah keluaran channel dari fitur map*(output depth)*. Kemudian kernel pada konvolusi biasa D_K x D_K x M x N, dimana D_K adalah dimensi kernel, M jumlah input channel dan N jumlah keluaran channel. Total cost dari konvolusi biasa memiliki ketergantungan pada M pada Persamaan 3.23.

$$D_K \bullet D_K \bullet M \bullet N \bullet D_F \bullet D_F \tag{3.23}$$

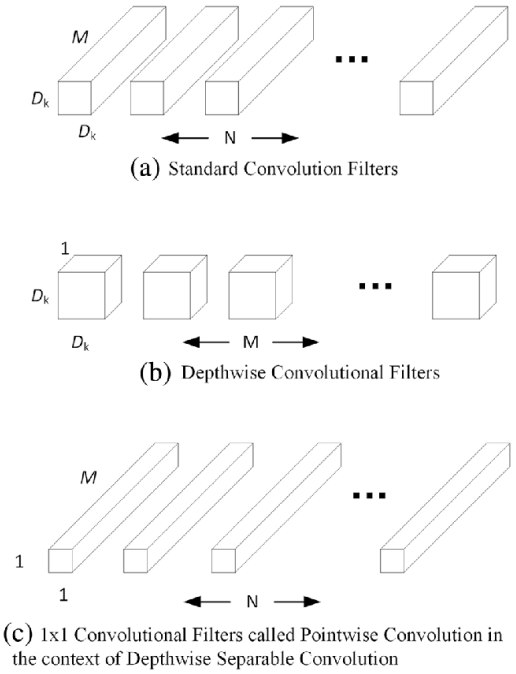
Depthwise separable convolution terbentuk dari layer *depthwise convolution* dan *pointwise convolution*. *Depthwise convolution* sangat efisien daripada konvolusi biasa, namun pada layer *depthwise* tersebut tidak menghasilkan fitur baru, oleh karena itu digunakan tambahan layer *pointwise convolution* untuk menghitung linier kombinasi dari keluaran *depthwise convolution* menggunakan 1x1 *convolution* ,sehingga menghasilkan fitur baru. Total cost dari *depthwise convolution* pada Persamaan 3.24.

$$D_K \bullet D_K \bullet M \bullet D_F \bullet D_F \tag{3.24}$$

Kombinasi kedua layer tersebut dinamakan *deptwise separable convolution* yang memiliki total cost Persamaan 3.25 :

$$D_K \bullet D_K \bullet M \bullet N \bullet D_F \bullet D_F + M \bullet N \bullet D_F \bullet D_F \tag{3.25}$$

MobileNet menggunakan 3x3 *depthwise separable convolution* yang mana mereduksi komputasi 8 hingga 9 kali lebih cepat dari konvolusi biasa. Perbedaan konvolusi biasa dan *depthwise separable convolution* digambarkan pada Gambar 3.5(Howard et al., 2017).



Gambar 3.5 Convolutional standard dan depthwise separable (Howard et al., 2017)

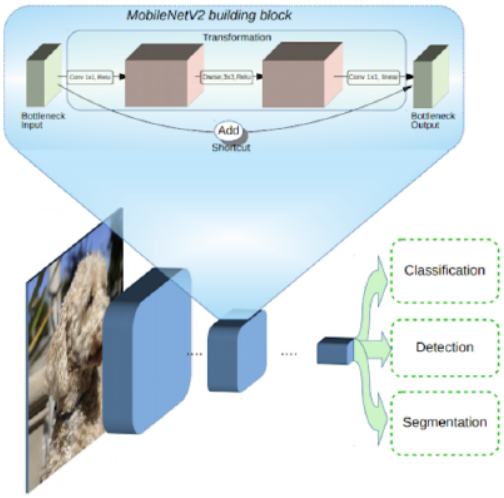
3.7. MobileNetV2

Perbedaan *MobileNet* versi kedua merupakan pengembangan dari *MobileNet* versi pertama. Pada *MobileNet* memiliki modifikasi arsitektur linier *bottleneck* dan *shortcut* koneksi antar *bottleneck* seperti Gambar 3.6. Sehingga arsitektur dalam *MobileNetV2* menjadi *bottleneck depth-separable convolution residuals*. Arsitektur *bottleneck* dituliskan pada Tabel 3.1.

Tabel 3.1 Blok Arsitektur *Bottleneck MobileNetV2* (Sandler et al., 2018)

Input	Operator	Output
$h\ x\ w\ x\ k$	$1x1$ Conv2D, ReLU6	$h\ x\ w$ (tk)
$h\ x\ w\ x\ tk$	$3x3$ dwise s=s, ReLU6	$\frac{h}{s}\ x\ \frac{w}{s}$ (tk)
$\frac{h}{s}\ x\ \frac{w}{s}\ x\ tk$	linear $1x1$ Conv2D	$\frac{h}{s}\ x\ \frac{w}{s}$ k'

Tabel 3.1 menunjukan blok arsitektur *bottleneck MobileNetV2* terdiri dari *fully convolution* dengan 32 filter diikuti 19 layer *residual bottleneck*. Kernel yang digunakan menggunaka ukuran 3x3 (standard untuk modern network) dan menggunakan dropout dan normalisasi selama pelatihan(Sandler et al., 2018).



Gambar 3.6 Struktur Dasar Pada *MobileNetV2* (Google AI Blog., 2018)

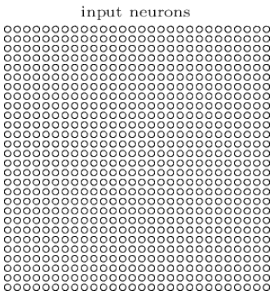
Pada bagian *bottleneck* terdapat *input* dan *output*, sedangkan layer di bagian dalam mengenkapsulasi kemampuan model untuk merubah input dari piksel ke gambar. Tambahan *shortcut* antar *bottleneck* memungkinkan saat melakukan *training* menjadi lebih cepat dengan peningkatan akurasi yang lebih baik(Google AI Blog,. 2018).

3.8. Convolutional Neural Network(CNN)

Convolutional Neural Network (CNN) merupakan jaringan syaraf tiruan yang memiliki arsitektur khusus yang dapat bekerja dengan baik dalam penggunaannya pada pemrosesan data berbentuk larik, seperti data runtun waktu dan citra digital. Dalam arsitekturnya CNN menggunakan operasi matematika konvolusi pada layer jaringan. Operasi konvolusi antara a dan b disimbolkan dengan (a x b). *Convolutional neural network* pada penerapannya menggunakan tiga ide dasar yaitu local receptive fields, shared weights, dan pooling(Nielsen., 2015).

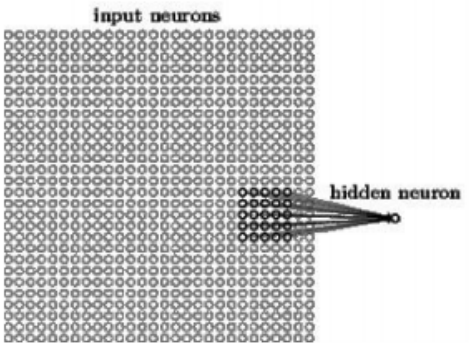
3.8.1. Local Receptive Fields

Pada jaringan syaraf tiruan dengan *fully-connected*, input dari jaringan digambarkan dengan garis vertikal dari kumpulan neuron. Pada CNN, input digambarkan sebagai persegi dengan ukuran o × o sesuai dengan ukuran input yang diberikan. Misalkan terdapat input dengan ukuran 28 × 28 neuron, yang berkesesuaian dengan 28 × 28 intensitas piksel pada citra seperti terlihat pada Gambar 3.7.



Gambar 3.7 Ilustrasi Citra 28x28 Piksel (Nielsen., 2015)

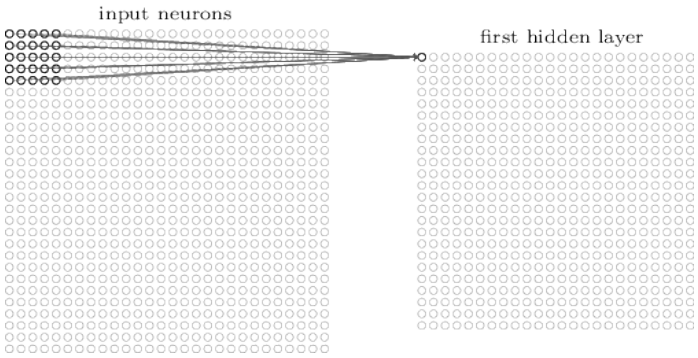
Seperti jaringan syaraf tiruan pada umumnya, setiap neuron dari input terhubung dengan layer dari hidden neuron. Sedikit berbeda, pada CNN neuron input tidak terhubung secara *fully-connected* dengan setiap hidden neuron, tetapi hanya region lokal kecil dari input terhubung dengan sebuah hidden neuron (Nielsen., 2015). Pada Gambar 3.8 terlihat bahwa hanya sebagian kecil region yang *localized* dari input neuron yang terhubung dengan hidden neuron.



Gambar 3.8 Ilustrasi *local receptive fields* (Nielsen., 2015)

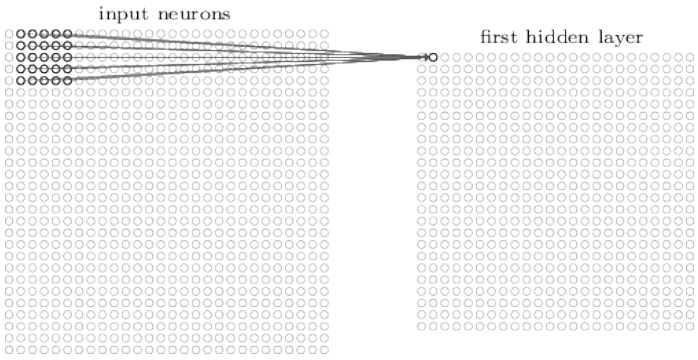
Pada Gambar 3.9 setiap neuron pada hidden layer terhubung dengan region berukuran 5×5 piksel pada neuron input. Region pada input layer inilah yang dinamakan dengan *local receptive fields* dan setiap koneksi memiliki bobot yang akan disesuaikan seiring dengan proses pelatihan. Selain itu hidden neuron juga memiliki dan mempelajari bias secara keseluruhan. Oleh karena itu, setiap hidden neuron dilatih untuk menganalisa masing-masing local receptive fields yang bersesuaian.

Tahap selanjutnya, local receptive fields akan digeser (slide) sepanjang ukuran citra dari posisi paling kiri atas hingga kanan bawah. Setiap local receptive fields akan memiliki pasangan hidden neuron yang berbeda pada hidden layer. Ilustrasi proses pergeseran *local receptive fields* terdapat pada Gambar 3.9 dan Gambar 3.10 berikut.



Gambar 3.9 Ilustrasi Pergesaran *local receptive fields* (Nielsen., 2015)

Selanjutnya *local receptive fields* akan digeser sebanyak satu piksel ke kanan seperti pada Gambar 3.8 dan terhubung dengan hidden neuron yang berbeda dengan local receptive fields sebelumnya.



Gambar 3.10 Ilustrasi Pergeseran dengan Stride = 1 (Nielsen., 2015)

Pergeseran piksel dengan stride = 1 di ilustrasikan pada Gambar 3.9.Proses ini berlangsung hingga *local receptive fields* berada pada posisi piksel paling kanan bawah. Sehingga setelah proses slide dilakukan akan terbentuk hidden layer dengan ukuran sesuai dengan ukuran local receptive fields dan panjang pergeseran (stride) yang digunakan (Nielsen, 2015). Maka jika citra dengan ukuran 28 × 28 piksel dan digunakan *local receptive fields* dengan ukuran 5 × 5, dan digeser dengan stride = 1, maka akan terbentuk hidden layer dengan ukuran 24 × 24 neuron.

3.8.2. Shared Weight

Pada CNN, sebuah neuron pada hidden layer yang terhubung dengan 5 × 5 neuron pada input layer (sesuai dengan ukuran *local receptive fields* yang digunakan). Hal ini menunjukkan bahwa neuron tersebut memiliki sebuah bias dan matriks bobot dengan ukuran 5 × 5 yang menghubungkan 5 × 5 neuron inputnya. Matriks bobot ini, dalam CNN, disebut sebagai kernel. Matriks bobot pada jaringan syaraf tiruan biasa sedikit berbeda dengan matriks bobot pada CNN, yaitu nilai matriks bobot pada CNN bernilai sama untuk setiap 24 × 24 neuron pada hidden layer. Hal tersebut menunjukkan bahwa semua neuron pada hidden layer akan mendeteksi fitur yang sama dengan lokasi pada input citra yang berbeda (Nielsen., 2015).

3.8.3. Konvolusi

Lapisan yang pertama kali akan dilewati oleh data masukan adalah lapisan konvolusi, bertujuan untuk memperoleh feature map yang merepresentasikan

masukan. Parameter yang digunakan untuk menentukan ukuran feature map keluaran berupa filter, ukuran dari filter, besarnya langkah pergeseran filter pada operasi konvolusi atau biasa disebut stride, dan ukuran padding. Cara menghitung ukuran dari keluaran operasi konvolusi ditunjukkan pada Persamaan 3.26 dan 3.27 berikut(Nielsen., 2015):

$$H_0 = \frac{H - F + 2P}{S} + 1 \tag{3.26}$$

$$W_0 = \frac{W_i - F + 2P}{S} + 1 \tag{3.27}$$

Keterangan :

H_0 = tinggi fitur map keluaran

W_0 = lebar fitur map keluaran

H_i = tinggi fitur map masukan

W_i = lebar fitur map masukan

F = ukuran filter

P = ukuran padding

S = ukuran stride

Nilai semua elemen pada persamaan tersebut harus merupakan bilangan bulat karena akan merepresentasikan suatu ukuran feature map. Apabila terdapat salah satu nilai yang bukan merupakan bilangan bulat, nilai tersebut harus dibulatkan dengan melakukan pembulatan kebawah. Adapun cara menghitung nilai keluaran dari proses konvolusi dapat dilihat pada Persamaan 3.278berikut(Nielsen., 2015):

$$O_{mn} = \sum I_{i,j}^{k} . F_{i,j} \tag{3.28}$$

Keterangan :

O_{mn}= elemen matriks keluaran pada baris ke-m kolom ke –n.

$I_{i,j}^{k}$= elemen matriks masukan bagian ke-k pada baris ke-i kolom ke-j.

$F_{i,j}$= elemen matriks filter pada baris ke-i kolom ke –j.

3.8.4. Fungsi Aktivasi

Salah satu faktor signifikan mempengaruhi kinerja algoritme Convolutional Neural Network adalah penerapan fungsi aktivasi dalam jaringan. Fungsi ini

membantu menyelesaikan permasalah-permasalahan yang bersifat non-trivial dalam suatu jaringan dengan cara mengambil sebuah nilai dan melakukan operasi matematika. Fungsi aktivasi ini diletakkan di perhitungan akhir dari keluaran feature map atau setelah layer konvolusi dan subsampling layer. Fungsi aktivasi yang sering digunakan adalah Rectified Linear Unit (ReLU) karena fungsi ini lebih cepat daripada fungsi aktivasi non-linear lainnya seperti sigmoid atau tanh. Fungsi ReLU dapat dilihat pada Persamaan 3.29 dan 3.30(Nielsen., 2015):

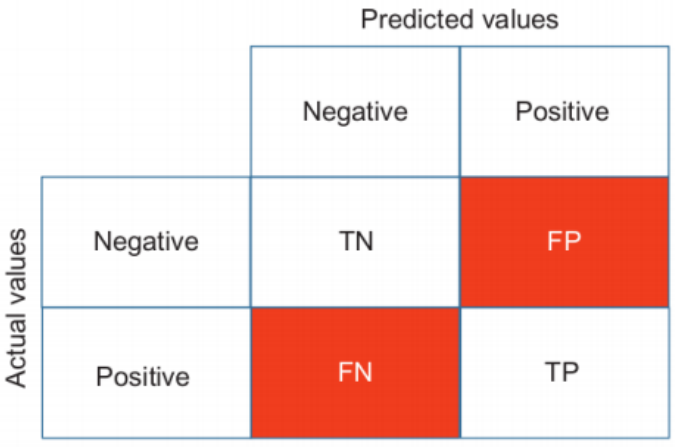
$$f(x) = ReLU(x) = max(0, x) \tag{3.29}$$

$$f(x) = 0 \; jika \; x \leq 0 \; atau \; x \; jika > 0 \tag{3.30}$$

3.9. Evaluasi

3.9.1. Confussion Matrix

Pengukuran evaluasi dari pengujian dapat dilakukan menggunakan *confusion matrix*. tabel *confusion matrix* merupakan tabel klasifikasi yang bersifat prediktif seperti ditunjukan pada Gambar 3.11. Evaluasi menggunakan confussion matrix dapat digunakan untuk mengukur nilai akurasi dengan Persaman 3.31.



Gambar 3.11 Confusion Matrix (Leonard., 2017)

$$akurasi = \frac{TP}{TP + FP} \tag{3.31}$$

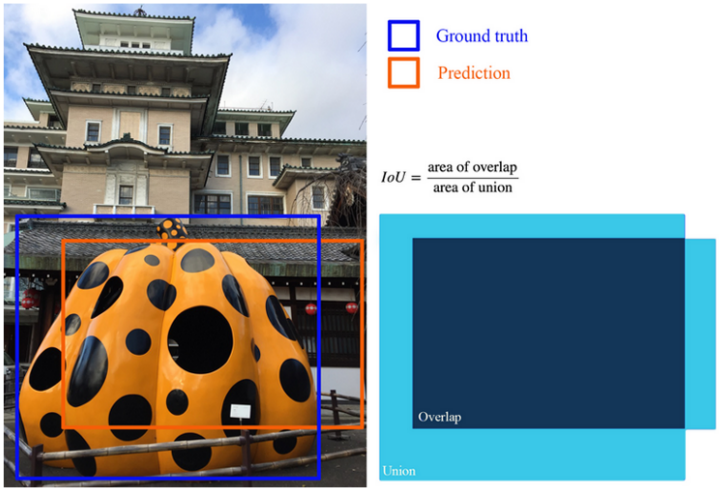
Dengan :

TP = True Positif, objek berupa gestur tangan dan terkenali

FP = False Positif, objek berupa gestur tangan tapi tidak terkenali

3.9.2. Average Precicion(AP)

Average Precicion adalah metrik populer untuk melakukan evaluasi pada objek detektor seperti *Faster-RCNN, SSD, dan lainnya*. AP menghitung nilai rata rata nilai presisi untuk nilai recall 0 hingga 1 (Hui., 2018). Evaluasi deteksi menggunakan konsep IoU(*intersection over union*) IoU menghitung interseksi kedua bounding boxes dimana terdiri dari ground truth dan prediksi dari bounding box digambarkan pada Gambar 3.12 dan perhitungan IoU pada Persamaan 3.32.



Gambar 3.12 Ilustrasi IoU(Hui., 2018)

$$IoU = \frac{area\_of\_overlap}{area\_of\_union} \tag{3.32}$$

IoU yang digunakan untuk memprediksi suatu bounding box, apabila nilai IoU melebihi *threshold* yang ditentukan maka akan terdeteksi sebuah objek. Sebaliknya jika IoU kurang dari batas yang ditentukan maka menandakan tidak terdeteksi apapun pada citra.

Terdapat 4 kategori dalam penentuan deteksi yaitu, FN(*False Negative*) terdapat objek namun tidak mendeteksi apapun. FP(*False Positive*) tidak terdapat ob-

jek namun hasilnya terdeteksi. TP(*True Positive*) terdapat objek dan terdeteksi. TN(*True Negative*) akan terjadi jika tidak ada objek dan tidak terdeteksi apapun. Berdasarkan nilai yang didapat maka diketahui presisi dan recall menggunakan Persamaan 3.33 dan Persamaan 3.34 (Hui., 2018).

$$presisi = \frac{TP}{TP + FP} \tag{3.33}$$

$$recall = \frac{TP}{TP + FN} \tag{3.34}$$

Metrik untuk mengukur detektor dari *object detection* menggunakan mAP *(mean average precicion)*. Nilai mAP dan AP dalam COCO dataset tidak memiliki perbedaan, dimana mAP adalah rata rata dari setiap AP yang memiliki nilai IoU sebagai *threshold*. Total metrik evaluasi pada COCO memiliki 12 metrik seperti pada Gambar 3.13 (Chen et al., 2015).

```
Average Precision (AP):
  AP                  % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50          % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75          % AP at IoU=.75 (strict metric)
AP Across Scales:
  AP^small            % AP for small objects: area < 32^2
  AP^medium           % AP for medium objects: 32^2 < area < 96^2
  AP^large            % AP for large objects: area > 96^2
Average Recall (AR):
  AR^max=1            % AR given 1 detection per image
  AR^max=10           % AR given 10 detections per image
  AR^max=100          % AR given 100 detections per image
AR Across Scales:
  AR^small            % AR for small objects: area < 32^2
  AR^medium           % AR for medium objects: 32^2 < area < 96^2
  AR^large            % AR for large objects: area > 96^2
```

Gambar 3.13 Metrik Evaluasi COCO (Chen et al., 2015)

BAB IV
METODOLOGI PENELITIAN

4.1. Alat dan Bahan

4.1.1. Alat

Penelitian ini menggunakan beberapa peralatan yang digunakan untuk membantu kegiatandari mulai pengumpulan data hingga pengujian sistem, beberapa peralatan tersebut adalah sebagai berikut :

1. PC/Laptop dengan spesifikasi processor Intel (R) Core i5-8300H CPU @2,4 GHz, GPU NVIDIA 1050, RAM 8 GB, sistem operasi Linux 64 bit.
2. Webcam Logitech C270
3. Dimmer
4. Kain
5. Lux Meter

Laptop merupakan peralatan yang paling utama yang digunakan untuk melakukan proses pengolahan citra dengan webcam sebagai alat untuk menangkap citra. Lux meter digunakan untuk mengukur intensitas cahaya saat pengambilan data dan pengujian data dengan menurunkan intensitas cahaya yang diatur menggunakan dimmer. Penggunaan kain bersifat opsional untuk latar belakang yang bervariasi.

4.1.2. Bahan

Bahan yang digunakan untuk melakukan penelitian ini berupa dataset, diantaranya sebagai berikut:

1. Dataset gestur tangan ASL
2. Dataset gambar tangan

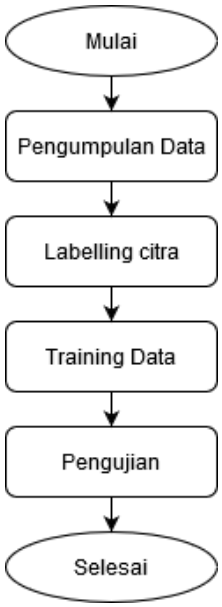
Dataset gestur tangan ASL merupakan dataset dari *Massey University* yang digunakan untuk melatih model pengenalan gestur tangan, sementara dataset gambar

tangan merupakan dataset untuk melakukan pelatihan deteksi tangan. Dataset gambar tangan diambil dari 3 subjek yang diambil gambar menggunakan webcam.

4.2. Prosedur Kerja

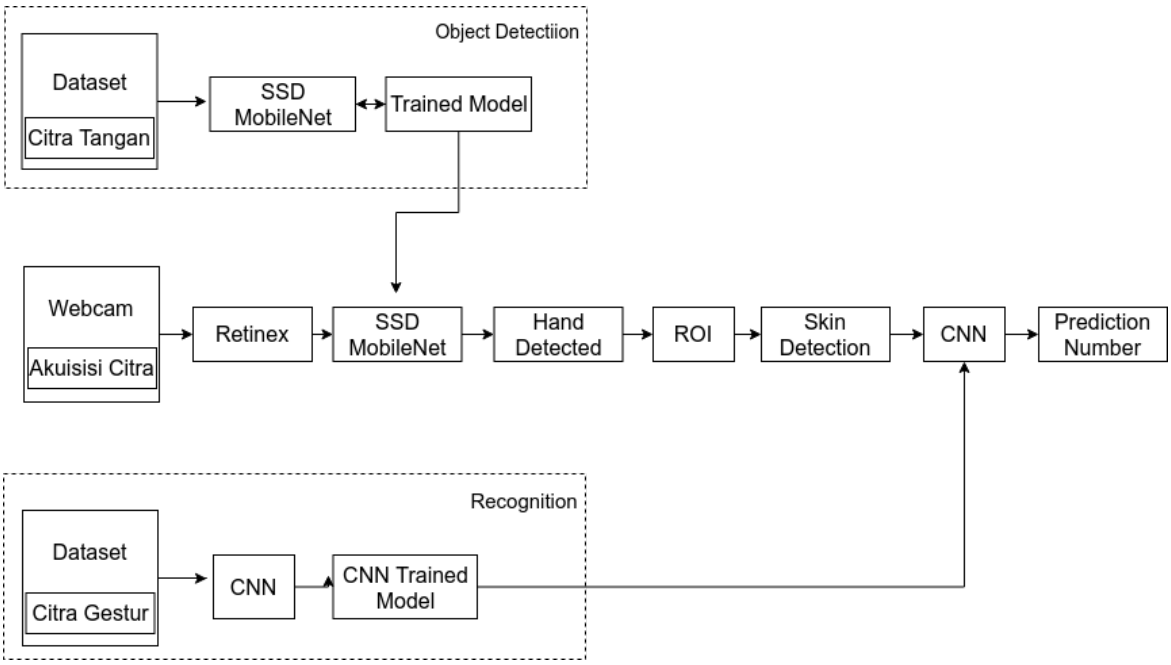
4.2.1. Analisis dan Perancangan Sistem

Alur penelitian yang akan dilakukan dalam penelitian ini memiliki beberapa tahapan diantaranya pengambilan dataset gestur, training data dan tahap paling akhir adalah pengujian. Alur kegiatan penelitian yang akan dilakukan secara garis besar ditunjukan pada Gambar 4.1.



Gambar 4.1 Alur Kegiatan Penelitian

Rancangan pengujian sistem ditunjukan pada Gambar 4.2 yang dimulai dengan *input* dari webcam secara *realtime*, sehingga setiap frame akan di proses pada tahap berikutnya. Setiap frame dengan citra RGB akan dilewatkan pada algoritme *Retinex* untuk dilakukan perbaikan kualitas citra dengan harapan meningkatkan kontras pada sebuah citra.



Gambar 4.2 Rancangan Sistem

Citra yang mengalami peningkatan kontras kemudian sistem akan melakukan deteksi tangan dengan *SSD MobileNet* yang kemudian citra tangan tersebut akan diambil sebagai ROI. Citra hasil ROI akan dilakukan segmentasi menggunakan *skin detection*. Hasil segmentasi akan diklasifikasikan dengan model CNN yang telah di training sebelumnya. Keluaran dari sistem ini akan menampilkan klasifikasi angka yang terdeteksi.

Penelitian yang digambarkan pada rencana sistem memiliki 3 bagian penting diantaranya *Retinex*, *SSD MobileNet* dan *Convolutional Neural Network*. Peranan Retinex pada penelitian ini merupakan solusi untuk mengatasi permasalahan intensitas cahaya, sehingga pada dataset yang dilatih tidak perlu menggunakan variasi intensitas yang rendah. Penggunaan SSD dengan arsitektur *MobileNet* bertujuan untuk mempercepat waktu komputasi supaya tidak terlalu berat dengan harapan lebih cepat. *Convolutional Neural Network* sendiri digunakan untuk melakukan ekstraksi fitur dari sebuah citra. Ekstraksi fitur ini sangat sering digunakan dalam pemrosesan sebuah citra untuk memperoleh informasi.

4.2.2. Pra-proses

Tahap pra-proses pada penelitian ini terdapat pada proses *enhancement* yang dilakukan oleh *Retinex*. Citra dengan intensitas cahaya rendah akan dilakukan proses perbaikan dahulu sebelum dilakukan proses deteksi maupun pengenalan. *Input* citra untuk proses *Retinex* menggunakan 3 kanal(R,G,B).

Proses pertama yang dilakukan adalah membuat filter *gaussian* menggunakan Persamaan 3.14 dengan permisalan σ = 10.

$$\begin{bmatrix} \frac{1}{2\pi(10)^2} \times \exp^{-\frac{(-1)^2+1^2}{2(10)^2}} & \frac{1}{2\pi(10)^2} \times \exp^{-\frac{0^2+1^2}{2(10)^2}} & \frac{1}{2\pi(10)^2} \times \exp^{-\frac{1^2+1^2}{2(10)^2}} \\ \frac{1}{2\pi(10)^2} \times \exp^{-\frac{(-1)^2+0^2}{2(10)^2}} & \frac{1}{2\pi(10)^2} \times \exp^{-\frac{0^2+0^2}{2(10)^2}} & \frac{1}{2\pi(10)^2} \times \exp^{-\frac{1^2+0^2}{2(10)^2}} \\ \frac{1}{2\pi(10)^2} \times \exp^{-\frac{(-1)^2+(-1)^2}{2(10)^2}} & \frac{1}{2\pi(10)^2} \times \exp^{-\frac{0^2+(-1)^2}{2(10)^2}} & \frac{1}{2\pi(10)^2} \times \exp^{-\frac{1^2+(-1)^2}{2(10)^2}} \end{bmatrix}$$

Filter *gaussian* yang dihasilkan menggunakan σ=10 adalah sebagai berikut.

$$\begin{bmatrix} 0,001575 & 0,001583 & 0,001575 \\ 0,001583 & 0,001591 & 0,001583 \\ 0,001575 & 0,001583 & 0,001575 \end{bmatrix}$$

Berdasarkan Persamaan 3.15, integral dari filter *gaussian* harus sama dengan 1. Integral dari filter tersebut dapat dicari dengan menjumlahkan semua elemen. Pada kasus ini nilai integral atau jumlahan dari elemen adalah 0.0142288. Apabila hasil dari integral tersebut tidak sama dengan 1 maka akan dikalikan menggunakan formula berikut.

$new\_a_{(x,y)} = a_{(x,y)} \times \frac{1}{total}$

$new\_a_{(-1,1)} = 0,001575 \times \frac{1}{0,142288} = 0,110740$

$new\_a_{(-1,0)} = 0,001583 \times \frac{1}{0,142288} = 0,111295$

Perhitungan nilai filter *gaussian* dilakukan untuk setiap distribusi kernel sehingga membentuk filter dengan nilai baru. Nilai filter yang baru menghasilkan hasil integral sama dengan 1.

$$\begin{bmatrix} 0,110740 & 0,111295 & 0,110740 \\ 0,111295 & 0,111853 & 0,111295 \\ 0,110740 & 0,111295 & 0,110740 \end{bmatrix}$$

Filter *gaussian* ini akan dikonvolusikan dengan input citra, contoh citra yang akan

dikonvolusikan sebagai berikut.

$$I_{(x,y)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 120 & 125 & 124 & 126 & 0 \\ 0 & 126 & 135 & 146 & 189 & 0 \\ 0 & 170 & 187 & 200 & 210 & 0 \\ 0 & 189 & 188 & 197 & 221 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * F_{(x,y)} = \begin{bmatrix} 0,110740 & 0,111295 & 0,110740 \\ 0,111295 & 0,111853 & 0,111295 \\ 0,110740 & 0,111295 & 0,110740 \end{bmatrix}$$

Perhitungan konvolusi untuk mendapatkan dimensi yang sama dengan citra *input* digunakan *zero padding*. Hasil konvolusi di ilustrasikan pada perhitungan berikut.

$O_{(1,1)}$ =(0*0,110740)+(0*0.111295)+(0*0,110740)+(0*0,111295)+(120*0,111853) +(125*0,111295)+(0*0,110740)+(126*0,111295)+(135*0,110740)=56,307698

$O_{(2,1)}$ =(0*0,110740)+(0*0,111295)+(0*0,110740)+(120*0,111295)+(125*0,111853) +(124*0,111295)+(126*0,110740)+(135*0,111295)+(146*0,110740)=86,284315

Konvolusi dilakukan dengan pergeseran 1 piksel sehingga menghasilkan citra dengan dimensi yang sama. Hasil konvolusi($O_{(x,y)}$) merupakan nilai dari Persamaan 3.13 yang mendapatan nilai sebagai berikut.

$$O_{(x,y)} = \begin{bmatrix} 56,307698 & 86,284315 & 93,934305 & 65,097310 \\ 95,945406 & 148,09183 & 160,21034 & 110,66494 \\ 110,65489 & 170,91206 & 185,90262 & 129,36380 \\ 81,692776 & 125,77508 & 133,77841 & 92,165220 \end{bmatrix}$$

$$\log O_{(x,y)} = \begin{bmatrix} 1,750568 & 1,935932 & 1,972825 & 1,813563 \\ 1,982024 & 2,170531 & 2,204691 & 2,044011 \\ 2,043971 & 2,232773 & 2,269286 & 2,111813 \\ 1,912183 & 2,099594 & 2,126386 & 1,964568 \end{bmatrix}$$

Nilai akhir dari *Single Scale Retinex* adalah pengurangan antara hasil logaritmik citra asli dengan logaritmik citra hasil konvolusi dimana dituliskan dalam Persamaan 3.17. Berikut adalah hasil *Single Scale Retinex*(σ=10).

$$R_{(SSR(\sigma=10))=} \begin{bmatrix} 0,328612 & 0,160977 & 0,120596 & 0,286806 \\ 0,118345 & -0,040198 & -0,04033 & 0,232450 \\ 0,186477 & 0,039068 & 0,031743 & 0,210405 \\ 0,364278 & 0,174563 & 0,168079 & 0,379824 \end{bmatrix}$$

Perhitungan *Multiscale Retinex* jumlahan dari setiap *Single Scale Retinex*. *Single Scale Retinex* dengan σ=65 dan σ=180 didapatkan nilai sebagai berikut.

$$R_{(SSR(\sigma=65))=} \begin{bmatrix} 0,329257 & 0,161283 & 0,120802 & 0,287442 \\ 0,118596 & -0,040252 & -0,040368 & 0,232871 \\ 0,186859 & 0,039126 & 0,031776 & 0,210871 \\ 0,364991 & 0,174929 & 0,168434 & 0,380586 \end{bmatrix}$$

$$R_{(SSR(\sigma=180))=} \begin{bmatrix} 0,329271 & 0,161289 & 0,120806 & 0,287455 \\ 0,118601 & -0,040253 & -0,040369 & 0,232879 \\ 0,186867 & 0,0391275 & 0,0317764 & 0,210880 \\ 0,365006 & 0,1749365 & 0,1684419 & 0,380602 \end{bmatrix}$$

Multiscale Retinex memiliki bobot pada setiap skala, dimana jika bobot dijumlahkan = 1. Bobot tersebut dikalikan dengan citra untuk masing masing skala, sehingga mendapatkan nilai berikut dengan masing masing bobot$\omega_n = 0.3, 0.3, 0.4$.

$$R_{(SSR(\sigma=10))*0,3} \begin{bmatrix} 0,098583 & 0,048293 & 0,036178 & 0,086041 \\ 0,035503 & -0,012059 & -0,012101 & 0,069735 \\ 0,055943 & 0,011720 & 0,009523 & 0,063121 \\ 0,109283 & 0,052368 & 0,050423 & 0,113947 \end{bmatrix}$$

$$R_{(SSR(\sigma=65))*0,3} \begin{bmatrix} 0,098777 & 0,048385 & 0.036240 & 0.086232 \\ 0,035578 & -0,012075 & -0.012110 & 0.069861 \\ 0,056057 & 0,011737 & 0.009532 & 0.063261 \\ 0,109497 & 0,052478 & 0.050530 & 0.114175 \end{bmatrix}$$

$$R_{(SSR(\sigma=180))*0,4} \begin{bmatrix} 0,131708 & 0,064515 & 0,048322 & 0,114982 \\ 0,047440 & -0,016101 & -0,016147 & 0,093151 \\ 0,074746 & 0,015651 & 0,012710 & 0,084352 \\ 0,146002 & 0,069974 & 0,067376 & 0,152240 \end{bmatrix}$$

Setiap nilai *Single Scale Retinex* dijumlahkan seperti pada Persamaan 3.18, kemu-

dian hasil dari penjumlahan akan dilakukan normalisasi dengan hasil akhir yang dibulatkan.

$$R_{(MSR)} = \begin{bmatrix} 0,329069 & 0,161194 & 0,120742 & 0,287256 \\ 0,118523 & -0,040236 & -0,040360 & 0,232748 \\ 0,186748 & 0,039109 & 0,031766 & 0,210735 \\ 0,364783 & 0,174822 & 0,168331 & 0,380363 \end{bmatrix}$$

Normalisasi = $\frac{(ABS(Nilai_{(x,y)}) - Nilai\_min)}{(Nilai\_max - Nilai\_min)} \times 255$

$$R_{(MSR)} = \begin{bmatrix} 224 & 122 & 98 & 198 \\ 96 & 0 & 1 & 165 \\ 138 & 48 & 44 & 152 \\ 246 & 130 & 126 & 255 \end{bmatrix}$$

Gambar 4.3 menunjukan ilustrasi peningkatan kualitas citra dengan intensitas cahaya rendah dikenai algoritme *retinex* sehingga menghassilkan citra dengan penginggkatan kontras.

a b

Gambar 4.3 Ilustrasi Perbaikan Kontras Menggunakan Retinex (a) Citra Asli; (b) Citra Hasil Perbaikan(Petro et al., 2014)

4.2.3. Pengumpulan Data

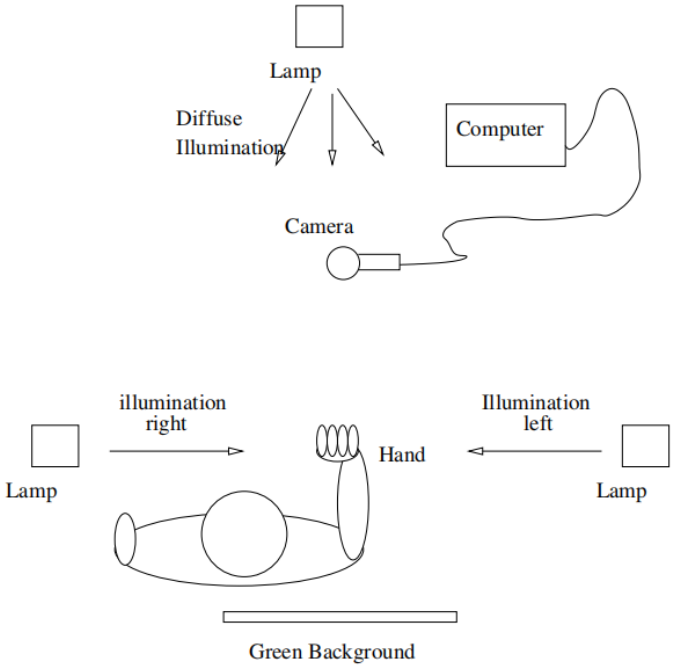
Data pelatihan dibagi menjadi dua, yaitu dataset citra gestur tangan yang mengacu pada ASL dan dataset citra tangan biasa untuk *object detection*. Kedua dataset ini merupakan dua hal yang berbeda tujuan. Dataset ASL digunakan untuk model pengenalan gestur ASL dan dataset citra tangan digunakan untuk model deteksi tangan.

Dataset gestur tangan ASL menggunakan dataset public dari *Massey University*. Dataset tersebut berisi 2425 citra yang dibagai dalam 36 kelas, yaitu 26

untuk huruf A-Z dan 10 kelas untuk angka 0-9. Pengambilan dataset dilakukan oleh 5 individu dengan variasi cahaya yang berbeda(Barczak et al., 2011). Acuan dalam penelitian ini menggunakan dataset angka 0 sampai 9, dengan citra yang sudah di segmentasi. Dataset ini akan dilatih dalam CNN untuk pengenalan gestur tangan. Pengambilan dataset *Massey University* dilakukan menggunakan *green screen* sehingga mudah untuk disegmentasi, setup pengambilan dataset dilakukan dengan skema seperti Gambar 4.4 (Barczak et al., 2011).

Pengambilan dataset berikutnya berbeda dengan dataset ASL, dataset berikutnya diambil menggunakan webcam untuk keperluan deteksi tangan, citra diambil dari 10 orang dengan random pose. Dataset ini digunakan untuk pelatihan pada *object detection*. Pengambilan dataset dilakukan masing masing 100 capture untuk setiap orang, sehingga pada dataset ini diperoleh 1000 citra.

Proses pengumpulan dataset tidak menghiraukan nilai intensitas cahaya karena permasalahan pencahayaan sudah diatasi pada algoritme *Retinex*, namun akan tetap diukur kondisi intensitas pada lingkungan tersebut menggunakan lux meter. Penggunaan dataset ASL dari *Massey University* tidak memiliki keterangan nilai intesitas cahaya, namun berdasarkan citra tersebut cenderung memiliki nilai intensitas yang tinggi, dimana dapat dilihat pada isi dataset. Tujuan dari dataset yang dilatih hanya untuk membuat model klasifikasi dan deteksi tanpa memperhitungkan intensitas cahaya rendah.



Gambar 4.4 Skema Pengambilan Dataset (Barczak et al., 2011)

4.3. Proses Pelatihan

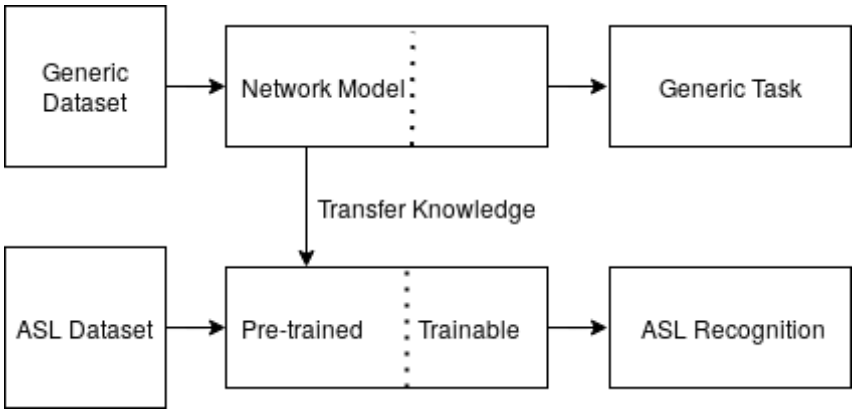
Proses pelatihan dalam penelitian ini dibagi menjadi dua bagian, yaitu pelatihan object detection dan pelatihan gestur recognition.

4.3.1. Pelatihan Gesture Recognition

Dataset pada pelatihan *gesture recognition* menggunakan dataset dari *Massey Univertisy* yang di labeli sesuai acuan ASL dari angka 0 hingga 9. Jumlah dataset adalah 700 citra kemudian dibagi yang akan dibagi dalam *train* dan *test* menggunakan *k-fold cross validation* dengan k = 5. Proses pelatihan untuk pengenalan gestur tangan menggunakan teknik *transfer learning* yang memiliki kelebihan pada waktu pelatihan dan penggunaan dataset yang cenderung sedikit. *Pre-trained* model yang digunakan adalah *MobileNetV2*. *Pre-trained* model memiliki pengetahuan dari dataset yang telah dilatih sebelumnya, pada *MobileNetV2* telah dilatih menggunakan dataset dari *ImageNet*, kemudian pengetahuan tersebut akan di pindahkan ke model baru dan dilatih sesuai dataset baru, sehingga memiliki keluaran dalam

bentuk klasifikasi atau tugas sesuai yang diinginkan.

Masukan citra memiliki dimensi 224x224 piksel yang terdiri 3 channel RGB, sehingga node *input* berjumlah 50176 dengan range 0-255 untuk setiap piksel. Arsitektur jaringan pada pelatihan ini dilakukan dengan mengganti bagian blok klasifikasi dari *MobileNetV2* menjadi klasifikasi dataset ASL. Arsitektur transfer learning dapat dilihat pada Gambar 4.5.



Gambar 4.5 Transfer Learning Pelatihan Gestur

4.3.2. Pelatihan Object Detection

Proses pelatihan pada *object detection* menggunakan *pre-trained model SSD MobilenetV2 COCO* yang sudah di latih sebelumnya dengan dataset COCO. Dataset yang baru akan dilatih menggunakan *pre-trained* model, sehingga dengan pengetahuan yang sudah ada dilatih agar menghasilkan model baru sesuai dengan *output* yang diinginkan. Dataset yang akan dilatih sejumlah 1000 citra yang telah dilabeli secara manual dengan pembagian 80% *training* dan 20% *testing*. Arsitektur yang digunakan dalam pelatihan *object detection* dapat dilihat pada Gambar 4.6.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Gambar 4.6 Arsitektur Mobilenet V2 (Sandler et al., 2018)

4.4. Pengujian dan Evaluasi

Pengujian dan evaluasi adalah tahap untuk mengukur performa dari sebuah sistem. Pengujian ini akan dibagi menjadi 6 tahap, yaitu evaluasi deteksi tangan, evaluasi pengenalan gestur, pengujian deteksi tangan menggunakan *Retinex*, pengujian pengenalan gestur tangan menggunakan *Retinex*, pengujian SNR dan pengujian keseluruhan sistem. Setiap pengujian yang menggunakan Retinex dilakukan hal yang sama, yaitu dengan menurunkan intensitas cahaya.

4.4.1. Evaluasi Deteksi Tangan

Evaluasi model deteksi tangan dilakukan setelah *training* menggunakan metrik *mean average precicion* (mAP). *Mean average precicion* merupakan metrik yang populer untuk mengevaluasi model dari detektor. *Mean average precicion* merupakan rata rata dari *Average Precicion* (AP) pada setiap nilai IoU. Sesuai dengan acuan COCO dataset untuk mengevaluasi model pada *object detection* dengan menghitung mAP dari IoU=0,5 hingga 0,95 dengan pertambahan 0,05.

4.4.2. Evaluasi Pengenalan Gestur Tangan

Evaluasi model CNN untuk pengenalan gestur tangan didapatkan saat *training* dan *testing* menggunakan *k-fold cross validation* dengan k = 5, dimana dilakukan validasi sebanyak 5 kali dengan skenario pengujian yang berbeda pada proses *training* dan *testing*. *Training accuracy* dan *testing accuracy* didapatkan pada setiap *fold*. Model dengan akurasi terdinggi akan digunakan sebagai model untuk pengenalan gestur tangan.

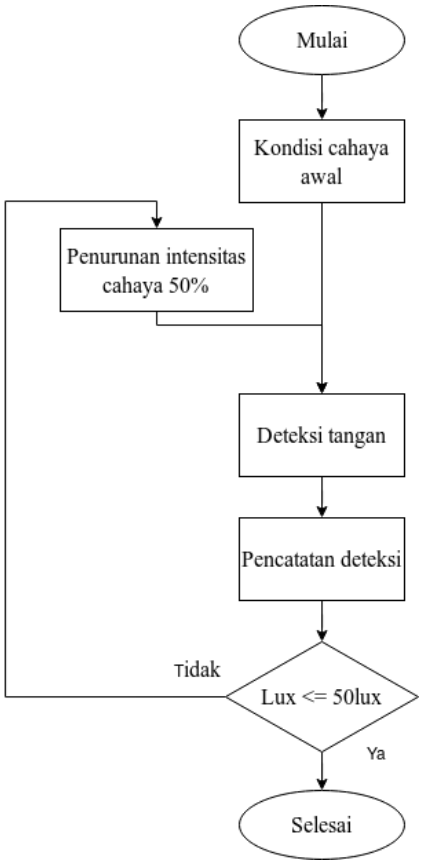
4.4.3. Pengujian SNR

SNR*(Signal to Noise Ratio)* adalah ukuran untuk mengukur kualitas citra terhadap citra yang dilakukan perbaikan. Citra hasil perbaikan dibandingkan dengan citra asli untuk mendapatkan nilai SNR nya. Nilai SNR yang tinggi mengindikasikan kualitas citra yang semakin baik karena rasio sinyal terhadap metode juga tinggi, sebaliknya nilai SNR yang rendah berarti kualitas citra yang di hasilkan semakin buruk atau semakin kecil dalam peningkatan kualitas citra. Pengujian SNR ini dilakukan untuk menentukan nilai dari parameter *retinex* yang akan dipakai pada sistem.

4.4.4. Pengujian Deteksi Tangan Menggunakan Retinex

Tujuan pengujian deteksi tangan dilakukan untuk menguji sistem dalam melakukan deteksi tangan terkait dengan penurunan intensitas cahaya. Ilustrasi pengujian deteksi tangan digambarkan pada Gambar 4.7. Pengujian dilakukan dengan kondisi cahaya awal sesuai yang terjadi pada ruangan. Pada kondisi awal tersebut akan dilakukan deteksi tangan, setiap hasil dari deteksi tersebut berupa *true positive* atau *false positive*. Hasil deteksi akan dicatat dalam bentuk tabel seperti pada Tabel 4.1.

Gambar 4.7 Ilustrasi Pengujian Deteksi Tangan



Gambar 4.8 Skema Kegiatan Pengujian Deteksi Tangan

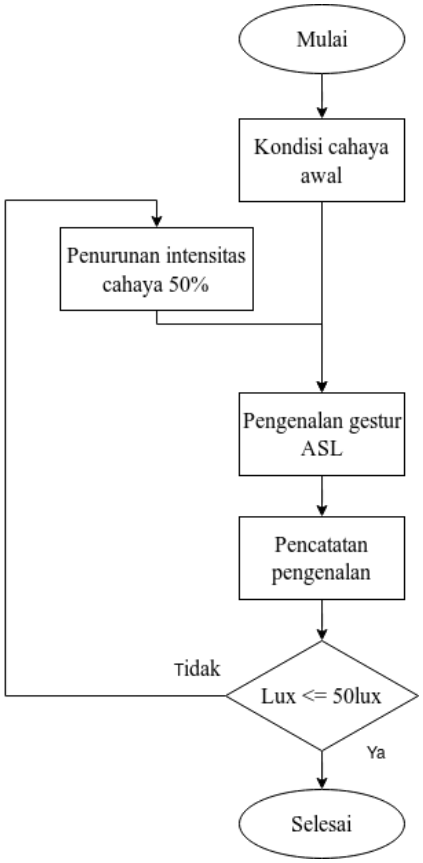
Pada pengujian ini alur kegiatan yang dilakukan saat pengujian deteksi tangan dapat dilihat pada Gambar 4.8. Pengujian deteksi tangan dilakukan secara manual oleh 3 subjek yang berbeda, setiap subjek akan dicatat apakah sistem mampu mendeteksi tangan atau tidak pada intensitas cahaya pada saat itu selama 10 kali, kemudian dilakukan penurunan intensitas sebesar 50% untuk kondisi selanjutnya dan dilakukan hal yang sama. Penurunan ini dilakukan hingga nilai lux yang terukur bernilai kurang dari atau sama dengan 50lux. Evaluasi pengujian dilakukan dengan *confussion matrix* menggunakan data yang dicatat pada Tabel 4.1, sehingga dapat diperoleh nilai akurasi untuk setiap kondisi lux dari hasil pengujian.

Tabel 4.1 Pengujian Deteksi Tangan

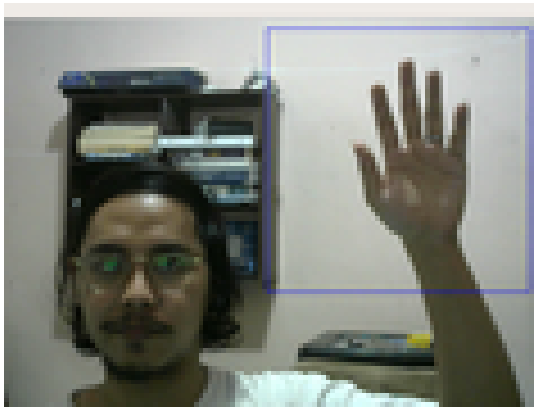
Nilai Lux(X)	Hasil Deteksi Subjek_1									Hasil Deteksi Subjek_2									Hasil Deteksi Subjek_3								
X																											
X=X*50%																											
X=X*50%																											
X=X*50%																											
...																											
X=X $\leq$ 50Lux																											

4.4.5. Pengujian Pengenalan Gestur Tangan Menggunakan Retinex

Pengujian tahap kedua yaitu pengenalan gestur tangan dengan *retinex*. Tujuan dari pengujian ini untuk mengukur performa sistem dalam melakukan pengenalan gestur tangan dengan acuan ASL sesuai dengan dataset yang telah dilatih. Skema kegiatan pengujian pengenalan gestur tangan dapat dilihat pada Gambar 4.9. Pengujian dilakukan dengan kondisi cahaya awal sesuai yang terjadi pada ruangan. Pada kondisi awal tersebut akan dilakukan pengenalan gestur tangan, setiap hasil dari pengenalan tersebut berupa *true positive* atau *false positive*. Nilai *true positive* terjadi apabila *input* gestur tangan dari webcam sesuai dengan klasifikasi gestur tangan ASL yang sebenarnya. Nilai *false positive* terjadi apabila *input* gestur tangan tidak sesuai dengan klasifikasi gestur ASL yang sebenarnya. Hasil pengenalan akan dicatat dalam bentuk tabel seperti pada Tabel 4.1.



Gambar 4.9 Skema Kegiatan Pengujian Pengenalan Gestur Tangan



Gambar 4.10 Ilustrasi Pengujian Pengenalan Gestur Tangan

Ilustrasi pengujian ini digambarkan pada Gambar 4.10. Pengujian pengenalan gestur tangan dilakukan secara manual dan terpisah dari deteksi tangan, tahap ini memiliki perlakuan sama dengan tahap sebelumnya, yaitu menggunakan 3 subjek yang berbeda, kemudian dilakukan pengenalan gestur tangan sebanyak 10 kali

untuk setiap klasifikasi. Total citra yang didapat dalam satu kondisi lux oleh satu subjek adalah 100 citra. Tahap berikutnya dilakukan penurunan intensitas cahaya, dengan mengatur cahaya lampu di ruangan uji sebesar 50% lux menggunakan dimmer. Pengujian dilakukan hingga batas lux kurang dari atau sama dengan 50 lux. Evaluasi yang dilakukan dengan *confussion matrix* menggunakan data yang diperoleh pada Tabel 4.2. Hasil evaluasi pengujian akan diperoleh nilai akurasi untuk setiap kondisi lux.

Tabel 4.2 Pengujian Pengenalan Gestur Tangan

Nilai Lux(X)

Klasifikasi	Pengenalan Subjek_1										Pengenalan Subjek_2										Pengenalan Subjek_3									
Angka 0																														
Angka 1																														
Angka 2																														
...																														
Angka 9																														

Nilai Lux(...)

Klasifikasi	Pengenalan Subjek_1										Pengenalan Subjek_2										Pengenalan Subjek_3									
Angka 0																														
Angka 1																														
Angka 2																														
...																														
Angka 9																														

Nilai Lux(X≤50)

Klasifikasi	Pengenalan Subjek_1										Pengenalan Subjek_2										Pengenalan Subjek_3									
Angka 0																														
Angka 1																														
Angka 2																														
...																														
Angka 9																														

4.4.6. Pengujian Sistem Keseluruhan

Pengujian tahap terakhir adalah pengujian keseluruhan sistem yang dilakukan secara manual, dengan menggabungkan *retinex*, deteksi tangan dan pengenalan gestur tangan dalam satu program utuh. Pengenalan akan terjadi apabila sebuah tangan dideteksi terlebih dahulu, jika tidak terdeteksi maka tidak akan dilakukan pengenalan. Nilai yang didapatkan dari hasil pengujian ini berupa *true positive* dan *false positive*. *True positive* didapatkan ketika sebuah *input* citra gestur dari webcam menghasilkan klasifikasi ASL yang sama dengan gestur ASL yang sebenarnya. Nilai *false positive* terjadi apabila *input* citra gestur dari webcam menghasilkan klasifikasi yang tidak sesuai dengan gestur ASL yang sebenarnya.

Penurunan intensitas dilakukan sama seperti pengujian sebelumnya. Tabel pengujian yang digunakan mengacu pada Tabel 4.2. Evaluasi dari akurasi sistem keseluruhan dapat diperoleh menggunakan *confusion matrix* pada data yang dicatat pada tabel. Nilai akurasi dihitung untuk setiap kondisi lux yang tercatat pada tabel pengujian.

BAB V
IMPLEMENTASI

Tahap implementasi pada penelitian ini menggunakan memiliki beberapa tahap yang terbagi menjadi empat yaitu *data collection*, *data labelling*, *training*, *testing*. Data collection pada penelitian ini adalah proses pengumpulan dataset yang dilakukan dengan melakukan akuisisi citra menggunakan webcam dan pengunduhan dataset *public*. Data yang di dapatkan dari proses akuisisi citra akan dilakukan *preprocessing* terlebih dahulu sebelum dilakukan proses labelling sesuai dengan klasifikasi. Tahap *training* digunakan untuk ekstraksi fitur dataset yang disimpan dalam sebuah model. Model yang terbentuk akan dilakukan proses pengujian pada tahap testing.

5.1. Data Collection

Data yang digunakan dalam penelitian ini adalah data yang diampbil dari proses akuisisi citra dan dataset *American Sign Language* dari *Massey University*. Dataset yang diambil dari proses akuisisi citra sebanyak 1000 citra tangan yang diambil dari 10 orang. Selanjutnya untuk dataset *American Sign Language* akan dilakukan preprocessing terlebih dahulu sebelum dijadikan format csv yang berisi nama dan lokasi citra.

5.1.1. Implementasi Proses Akuisisi Citra

Tahap akuisisi citra dilakukan bertujuan mendapatkan dataset untuk deteksi objek tangan sehingga dataset yang diambil adalah pose tangan dari subjek. Setiap subjek akan diambil sebanyak 100 citra sehingga menghasilkan 1000 citra. Pada program yang dibuat akan melakukan penyimpanan citra pada folder tertentu dengan setiap nama masing masing subjek. Gambar 5.1 menunjukan kode implementasi saat proses akuisisi citra berlangsung.

```
    if interrupt & 0xFF == 27:
        break
    #! save untuk setiap nama(folder) dengan tekan huruf
    if interrupt & 0xFF == ord('t'):
        cv2.imwrite(directory+'thomas/'+'thom'+str(count['
tangan_thomas'])+'.jpg',img)
    if interrupt & 0xFF == ord('m'):
        cv2.imwrite(directory+'meisy/'+'meis'+str(count['
tangan_meisy'])+'.jpg',img)
    if interrupt & 0xFF == ord('o'):
        cv2.imwrite(directory+'tongam/'+'tong'+str(count['
tangan_tongam'])+'.jpg',img)
    if interrupt & 0xFF == ord('c'):
        cv2.imwrite(directory+'cbnw/'+'cbn'+str(count['tangan_cbnw
'])+'.jpg',img)
    if interrupt & 0xFF == ord('a'):
        cv2.imwrite(directory+'aries/'+'ar'+str(count['
tangan_aries'])+'.jpg',img)
    if interrupt & 0xFF == ord('h'):
        cv2.imwrite(directory+'hendrik/'+'hen'+str(count['
tangan_hendrik'])+'.jpg',img)
    if interrupt & 0xFF == ord('v'):
        cv2.imwrite(directory+'viany/'+'via'+str(count['
tangan_viany'])+'.jpg',img)
    if interrupt & 0xFF == ord('e'):
        cv2.imwrite(directory+'edwin/'+'edw'+str(count['
tangan_edwin'])+'.jpg',img)
    if interrupt & 0xFF == ord('n'):
        cv2.imwrite(directory+'novi/'+'nov'+str(count['tangan_novi
'])+'.jpg',img)
    if interrupt & 0xFF == ord('s'):
        cv2.imwrite(directory+'silfany/'+'sil'+str(count['
tangan_silfany'])+'.jpg',img)
```

Gambar 5.1 Source code akuisisi citra

5.1.2. Implementasi Data Augmentasi

Proses augmentasi data digunakan untuk menduplikat citra dengan variasi yang berbeda. Pada proses augmentasi ini diterapkan untuk dataset *American Sign Language* karena hanya memiliki 700 citra untuk 10 klasifikasi. Sebelum dilakukan augmentasi, citra pada dataset *American Sign Language* akan dilakukan *resize* pada ukuran (224x224). Augmentasi yang dilakukan berupa variasi intensitas cahaya, intensitas warna, operasi morfologi, *filtering* dan rotasi.

```
def add_light(image, gamma=1.0,nama=name):
    invGamma = 1.0 / gamma
    table = np.array([((i / 255.0) ** invGamma) * 255 for i in np.
   arange(0, 256)]).astype("uint8")
    image=cv2.LUT(image, table)
    if gamma>=1:
        cv2.imwrite(Folder_name + "/light-"+str(gamma)+nama+
   Extension, image)
    else:
        cv2.imwrite(Folder_name + "/dark-"+ str(gamma)+nama +
   Extension, image)
def erosion_image(image,shift,nama=name):
    kernel = np.ones((shift,shift),np.uint8)
    image = cv2.erode(image,kernel,iterations = 1)
    cv2.imwrite(Folder_name + "/Erosion-"+"*"+str(shift)+nama +
   Extension, image)
def rotate_image(image,deg,nama=name):
    rows, cols,c = image.shape
    M = cv2.getRotationMatrix2D((cols/2,rows/2), deg, 1)
    image = cv2.warpAffine(image, M, (cols, rows))
    cv2.imwrite(Folder_name + "/Rotate-" + str(deg)+nama +
   Extension, image)
def hue_image(image,saturation,nama=name):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    v = image[:, :, 2]
    v = np.where(v <= 255 + saturation, v - saturation, 255)
    image[:, :, 2] = v
    image = cv2.cvtColor(image, cv2.COLOR_HSV2BGR)
    cv2.imwrite(Folder_name + "/hue-" + str(saturation)+nama +
   Extension, image)
def dilation_image(image,shift,nama=name):
    kernel = np.ones((shift, shift), np.uint8)
    image = cv2.dilate(image,kernel,iterations = 1)
    cv2.imwrite(Folder_name + "/Dilation-" + "*" + str(shift)+nama
   + Extension, image)
def median_blur(image,shift,nama=name):
    image=cv2.medianBlur(image,shift)
    cv2.imwrite(Folder_name + "/MedianBLur-" + str(shift)+nama +
   Extension, image)
```

Gambar 5.2 Source Code Augmentasi Citra

5.1.3. Pembuatan file CSV

Dataset dari pengenalan gestur akan dibuat file CSV yang berisi label dan lokasi dari setiap citra dalam dataset. Fungsi tocsv berikut adalah implementasi dari pembuatan file CSV

```
def to_csv(pth,dirname):
    for i in os.listdir(path):
        new_path=path+i
        for img  in os.walk(new_path):
            for n in img[2]:
                data=new_path+"/"+n
                label = i
                my_data.append(data)
                my_label.append(label)
        with open (dirname+'.csv','w') as csvfile:
            writer = csv.writer(csvfile)
            writer.writerow(['data', 'label'])
            for index_data in range(len(my_data)):
                print(my_data[index_data])
                print(*my_label[index_data])
                writer.writerow([my_data[index_data] ,  my_label[
   index_data]])
```

Gambar 5.3 Source Code Pembuatan file CSV

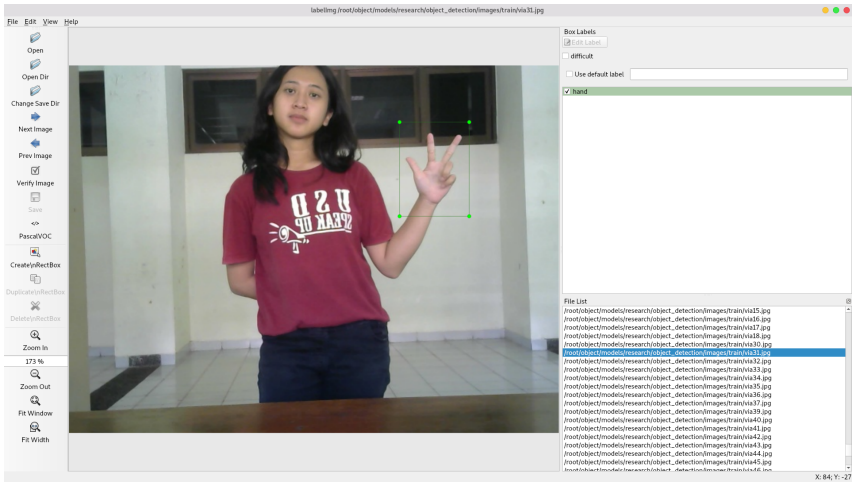
Hasil dari implementasi csv tersebut akan digunakan untuk refrensi input dalam proses pelatihan pengenalan gestur. Berikut adalah hasil dari pembuatan file csv dataset.

	data	label
2	/home/m448690/ASL4/augmented_data/1/Dilation-*5me7satu.jpg	1
3	/home/m448690/ASL4/augmented_data/1/Erosion-*6e37satu.jpg	1
4	/home/m448690/ASL4/augmented_data/1/dark-0.4hand5_1_dif_seg_2_cropped.jpg	1
5	/home/m448690/ASL4/augmented_data/1/Rotate-25hand1_1_top_seg_4_cropped.jpg	1
6	/home/m448690/ASL4/augmented_data/1/hue-170hand5_1_bot_seg_3_cropped.jpg	1
7	/home/m448690/ASL4/augmented_data/1/hue-50hand1_1_bot_seg_5_cropped.jpg	1
8	/home/m448690/ASL4/augmented_data/1/Rotate--15e19satu.jpg	1
9	/home/m448690/ASL4/augmented_data/1/Erosion-*3hand2_1_bot_seg_1_cropped.jpg	1
10	/home/m448690/ASL4/augmented_data/1/Dilation-*5hand4_1_bot_seg_1_cropped.jpg	1
11	/home/m448690/ASL4/augmented_data/1/me6.jpg	1
12	/home/m448690/ASL4/augmented_data/1/light-2.0e17satu.jpg	1

Gambar 5.4 hasil file csv

5.2. Implementasi Data Labelling

Pelabelan sebuah citra dilakukan untuk memberikan pengenalan suatu objek yang ada dalam sebuah citra, pada proses pelabelan menyimpan anotasi yang berisi informasi gambar yang disimpan dalam file XML yang berformat PASCAL VOC. Setiap data dilakukan pelabelan sesuai dengan nama objek yang mau di deteksi dalam hal ini adalah tangan yang diberi label *hand*. Proses pelabelan ditunjukan pada gambar xxxx.



Gambar 5.5 Proses Pelabelan Citra

Hasil citra terlabel dijadikan refrensi dalam informasi XML atau anotasi dari citra tertentu. berikut adalah contoh hasil anotasi dari salah satu citra.

```
<annotation>
        <folder>train</folder>
        <filename>via31.jpg</filename>
        <path>/root/object/models/research/object_detection/images/train/via31.jpg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>640</width>
                <height>480</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>hand</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>432</xmin>
                        <ymin>74</ymin>
                        <xmax>523</xmax>
                        <ymax>197</ymax>
                </bndbox>
        </object>
</annotation>
```

Gambar 5.6 hasil anotasi citra

5.3. Implementasi Pembuatan TFRecord file

Proses selanjutnya adalah mengubah informasi XML ke CSV dengan tujuan membuat format *Tensorflow* dataset atau *TFRecord*. Berikut program konversi file dataset.

```
def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     member[0].text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
                     int(member[4][3].text)
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class', 'xmin',
    'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df
def main():
    for folder in ['train','test']:
        image_path = os.path.join(os.getcwd(), ('images/' + folder
    ))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv(('images/' + folder + '_labels.csv'), index=
    None)
        print('Successfully converted xml to csv.')
```

Gambar 5.7 Source Code Pengubahan XML ke CSV

```
def class_text_to_int(row_label):
    if row_label == 'hand':
        return 1
    else:
        None
def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in zip
    (gb.groups.keys(), gb.groups)]
def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path, '{}'.format(group.
    filename)), 'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size
    filename = group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
    classes = []
    for index, row in group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)
        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))
    tf_example = tf.train.Example(features=tf.train.Features(
    feature={
        'image/height': dataset_util.int64_feature(height),
        'image/width': dataset_util.int64_feature(width),
        'image/filename': dataset_util.bytes_feature(filename),
        'image/source_id': dataset_util.bytes_feature(filename),
        'image/encoded': dataset_util.bytes_feature(encoded_jpg),
        'image/format': dataset_util.bytes_feature(image_format),
```

5.4. Konfigurasi *Pipeline Object Detection*

Proses pelatihan Tensorflow Object Detection API menggunakan konfigurasi pipeline, beberapa konfigurasi tersebut diantaranya model, training_config, eval_config, train_input_config, eval_input_config. Dataset yang berformat *TFRecord akan di lewatkan sebagai input* dan model pada ssd mobilenet v2 akan digunakan sebagai *pretrained* model. Berikut adalah konfigurasi yang digunakan

5.5. Implementasi Training

5.5.1. Object Detection

Tahap pelatihan dalam *object detection* menggunakan arsitektur mobilenet v2, proses pelatihan berlangsung dengan menggunakan parameter pada konfigurasi pipeline. berikut adalah program untuk melakukan pelatihan.

```
def main(unused_argv):
  flags.mark_flag_as_required('model_dir')
  flags.mark_flag_as_required('pipeline_config_path')
  tf.config.set_soft_device_placement(True)
  if FLAGS.checkpoint_dir:
    model_lib_v2.eval_continuously(
        pipeline_config_path=FLAGS.pipeline_config_path,
        model_dir=FLAGS.model_dir,
        train_steps=FLAGS.num_train_steps,
        sample_1_of_n_eval_examples=FLAGS.
    sample_1_of_n_eval_examples,
        sample_1_of_n_eval_on_train_examples=(
            FLAGS.sample_1_of_n_eval_on_train_examples),
        checkpoint_dir=FLAGS.checkpoint_dir,
        wait_interval=300, timeout=FLAGS.eval_timeout)
  else:
    if FLAGS.use_tpu:
      resolver = tf.distribute.cluster_resolver.TPUClusterResolver
    (
          FLAGS.tpu_name)
      tf.config.experimental_connect_to_cluster(resolver)
      tf.tpu.experimental.initialize_tpu_system(resolver)
      strategy = tf.distribute.experimental.TPUStrategy(resolver)
    elif FLAGS.num_workers > 1:
      strategy = tf.distribute.experimental.
    MultiWorkerMirroredStrategy()
    else:
      strategy = tf.compat.v2.distribute.MirroredStrategy()

    with strategy.scope():
      model_lib_v2.train_loop(
          pipeline_config_path=FLAGS.pipeline_config_path,
          model_dir=FLAGS.model_dir,
          train_steps=FLAGS.num_train_steps,
          use_tpu=FLAGS.use_tpu,
          checkpoint_every_n=FLAGS.checkpoint_every_n,
          record_summaries=FLAGS.record_summaries)
if __name__ == '__main__':
  tf.compat.v1.app.run()
```

Gambar 5.9 Source Code Pelatihan *Object Detection*

5.5.2. Pengenalan Gestur

Pada proses pelatihan gestur menggunakan *k-fold cross validation* dengan k=5. Arsitektur yang digunakan adalah mobilenet v2, pada pelatihan ini *pretrained* model akan di *load* menjadi base model dan jumlah *classifier* diganti sesuai dengan objek yang akan di klasifikasikan dan menambahkan *hidden layer*. Proses tersebut ditunjukan pada program berikut.

```
base_model_path = "/home/448690/ASL4/mobilenetv2.h5"
train_path='/home/m448690/ASL4/augmented_data/'
in_img=(224,224,3)
colname=["data","label"]
train_data = pd.read_csv('train_augmented.csv',dtype=str)
Y = train_data['label']
X = train_data['data']
kf = KFold(n_splits = 5,shuffle=True)
for train_index, val_index in kf.split(X):
    training_data = train_data.iloc[train_index]
    validation_data = train_data.iloc[val_index]
    base_model = tf.keras.applications.MobileNetV2(input_shape=
    in_img ,include_top=False, weights="mobilenetv2.h5")
    for layer in (base_model.layers):
        layer.trainable=True
    x=base_model.output
    x=GlobalAveragePooling2D()(x)
    x=Flatten()(x)
    x=Dense(1024,activation='relu')(x) #dense layer 1
    x=Dense(812,activation='relu')(x) #dense layer 2
    preds=Dense(10,activation='softmax')(x) #final layer with
    softmax activation for N classes
    model=Model(inputs=base_model.input,outputs=preds) #specify
    the inputs and outputs
    model.summary()
```

Gambar 5.10 Source Code Pelatihan *Pengenalan Gestur*

5.6. Implementasi Testing

Pada implementasi testing dilakukan beberapa pengujian model SNR dan mAP

BAB VI
HASIL DAN PEMBAHASAN

6.1. Evaluasi Deteksi Tangan

6.2. Evaluasi Pengenalan Gestur Tangan

6.3. Pengujian SNR

6.4. Pengujian Deteksi Tangan Menggunakan Retinex

6.5. Pengujian Pengenalan Gestur Tangan Menggunakan Retinex

6.6. Pengujian Sistem Keseluruhan

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

7.2. Saran

DAFTAR PUSTAKA

Aribowo, E., Yustina, E., Studi, P., Informatika, T., Teknologi, F., Universitas, I. & Dahlan, A., 2009, *Implementasi Metode Retinex Untuk Pencerahan Citra*, Jurnal Informatika, 3, 2, 323–330.

Loh, Y.P., Liang, X. & Chan, C.S., 2019,*Low-light image enhancement using Gaussian Process for features retrieval*, Signal Processing: Image Communication, 74, 175–190. https://doi.org/10.1016/j.image.2019.02.001,.

Saputra, L.K.P., 2016, *Perbandingan Varian Metode Multiscale Retinex Untuk Peningkatan Akurasi Deteksi Wajah Adaboost HAAR-like*, Jurnal Teknik Informatika dan Sistem Informasi, 2, 1, 89–98.

Shen, L., Yue, Z., Feng, F., Chen, Q., Liu, S. & Ma, J., 2017, *MSR-net:Low-light Image Enhancement Using Deep Convolutional Network*, , , January. http://arxiv.org/abs/1711.02488,.

Tanaka, Y., Yamashita, Y., Nishikawa, K., Yamaguchi, T. & Nishitani, T., 2019,*Retinex Foreground Segmentation for Low Light Environments*, 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA ASC 2018 - Proceedings, , November, 285–290.

Yingxin, X., Jinghua, L., Lichun, W. & Dehui, K., 2017,*A Robust Hand Gesture Recognition Method via Convolutional Neural Network*, Proceedings - 2016 International Conference on Digital Home, ICDH 2016, 64–67.

Madenda,Sarifudin., 2015, *Pengolahan & Video Digital*, Erlangga, Jakarta[BUKU].

Nielsen, M., 2015, *Neural Networks and Deep Learning, Determination Press*., [Daring]. tersedia di *http://neuralnetworksanddeeplearning.com*.

Rinaldi, Munir., 2004, *Pengolahan Citra Digital*, Bandung, Informatika[BUKU].

Hidayatullah, Priyanto,. 2017,*Pengolahan Citra Digital Teori dan Aplikasi Nyata*, Bandung, Informatika[BUKU].

Barczak, A.L.C., Reyes, N.H., Abastillas, M., Piccio, A. & Susnjak, T., 2011, *A New 2D Static Hand Gesture Colour Image Dataset for ASL Gestures*,Res. Lett. Inf. Math. Sci., 15, 12-20.*http://iims.massey.ac.nz/research/letters/*.

Kolkur, S., Kalbande, D., Shimpi, P., Bapat, C. & Jatakia, J., 2017, *Human Skin Detection Using RGB , HSV and YCbCr Color Models*, , 137, 324-332.

Arabi, S., Haghighat, A. & Sharma, A., 2019, *A deep learning based solution for construction equipment detection: from development to deployment* , , , April.

Srinivasan, R., 2016, *Implementing Histogram Equalization and Retinex*, , , July.

Huang, H., Chong, Y., Nie, C. & Pan, S., 2019, *Hand Gesture Recognition with Skin Detection and Deep Learning Method Hand Gesture Recognition with Skin Detection and Deep Learning Method*, J. Phys.: Conf. Ser. 1213 022001.

Posada-Gómez, Rubén & Sanchez Medel, Luis & Alor-Hernández, Giner & Martinez Sibaja, Albino & Aguilar-Laserre, A. & Leija-Salas, L.. 2007. *A Hands Gesture System Of Control For An Intelligent Wheelchair*. 68 - 71. 10.1109/ICEEE.2007.4344975.

Kemenkes, 2018, *Indonesia Inklusi dan Ramah Disabilitas*[Daring], tersedia di *https://www.kemkes.go.id/resources/download/pusdatin/infodatin/infodatin-disabilitas.pdf*.

Afrianto, T. & Amalia, F., 2016, *Pengaruh Komponen Krominan Pada Ruang Warna*, Prosiding Seminar Nasional Teknologi Terapan (SNTT), August 2017, 282-285.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H., 2017, *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*, *http://arxiv.org/abs/1704.04861*.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.C., 2018, *MobileNetV2: Inverted Residuals and Linear Bottlenecks*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 45104520.

Google AI Blog., 2018, *MobileNetV2: The Next Generation of On-Device Computer Vision Networks*[Daring], tersedia di *ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html?m=1*.

Leonard, L.C., 2017, *Web-Based Behavioral Modeling for Continuous User Authentication (CUA)*, 1 edisi, Elsevier Inc., [Daring]. tersedia di DOI:10.1016/bs.adcom.2016.12.001.

Hui, Jonathan., 2018, *mAP (mean Average Precision) for Object Detection*[Daring], tersedia di *www.medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173*.

Chen, X., Fang, H., Lin, T.Y., Vedantam, R., Gupta, S., Dollár, P & Zitnick, C.L., 2015, *Microsoft COCO Captions: Data Collection and Evaluation Server*, arXiv:1504.00325.

Petro., Ana-Belen & Sbert., Catalina & Morel., Jean-Michel., 2014. *Multiscale Retinex*. Image Processing On Line, 4. 71-88, DOI:10.5201/ipol.2014.107.

X. He., T. Wang., Y. Jia., Y. Wang., Z. Xie and D. Xie., 2016, *Studying fidelity issues in image enhancement by means of multi-scale retinex with color restoration*, 2016 3rd International Conference on Systems and Informatics (ICSAI), Shanghai, pp. 536-540, DOI: 10.1109/ICSAI.2016.7811013.

A. S. Parihar and K. Singh., *A study on Retinex based method for image enhancement*, 2018, 2nd International Conference on Inventive Systems and Control (ICISC), Coimbatore, pp. 619-624, DOI: 10.1109/ICISC.2018.8398874.