IMAGE RECOGNITION ALFABET BAHASA ISYARAT INDONESIA (BISINDO) MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK*

Disusun Oleh:

N a m a : Achmad Noer Aziz

NIM : 17523001

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

IMAGE RECOGNITION ALFABET BAHASA ISYARAT INDONESIA (BISINDO) MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK*

TUGAS AKHIR

Disusun Oleh:

N a m a : Achmad Noer Aziz

NIM : 17523001

Yogyakarta, 4 Mei 2021

Pembimbing,

(Arrie Kurniawardhani, S.Si., M.Kom.)

HALAMAN PENGESAHAN DOSEN PENGUJI

IMAGE RECOGNITION ALFABET BAHASA ISYARAT INDONESIA (BISINDO) MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK*

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 4 Mei 2021

Tim Penguji

Arrie Kurniawardhani, S.Si., M.Kom. _____

Anggota 1

Dr. Yudi Prayudi, S.Si., M.Kom. _____

Anggota 2

Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Achmad Noer Aziz

NIM : 17523001

Tugas akhir dengan judul:

IMAGE RECOGNITION ALFABET BAHASA ISYARAT INDONESIA (BISINDO) MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK*

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 4 Mei 2021

(Achmad Noer Aziz)

HALAMAN PERSEMBAHAN

Tugas Akhir ini penulis persembahkan kepada orang-orang yang telah memberikan dukungan dan bantuan kepada penulis sehingga Tugas Akhir ini dapat diselesaikan dengan baik. Tugas Akhir ini juga penulis persembahkan kepada pembaca dan kepada orang yang menjadikan Tugas Akhir ini sebagai referensi ataupun panduan.

HALAMAN MOTO

“There is a fine line between bravery and stupidity”

KATA PENGANTAR

Bismillahirrahmannirrahiim

Puja dan puji syukur penulis panjatkan kehadirat Allah SWT atas segala rahmat dan hidayah yang diberikan kepada penulis, sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul: "*Image Recognition* Alfabet Bahasa Isyarat Indonesia (BISINDO) Menggunakan Metode *Convolutional Neural Network*" ini dengan baik.

Sebagaimana manusia biasa yang tidak luput dari kelemahan dan kekurangan, penulis menyadari sepenuhnya bahwa Tugas Akhir ini masih jauh dari kata sempurna. Oleh karena itu, dengan segala kerendahan hati penulis mengharapkan saran-saran dan kritik yang bersifat membangun demi kesempurnaan Tugas Akhir ini.

Penulis juga menyadari bahwa selesainya Tugas Akhir ini tidak lepas dari bantuan berbagai pihak, baik berupa ilmu pengetahuan, moril maupun materiil. Untuk itu, pada kesempatan ini penulis mengucapkan banyak terima kasih yang tak terhingga kepada:

1. Keluarga khususnya kedua orang tua yang telah memberikan dorongan dan semangat dalam pelaksanaan Tugas Akhir.
2. Ibu Arrie Kurniawardhani, S.Si., M.Kom. sebagai Dosen Pembimbing yang telah memberikan bimbingan, petunjuk, pengarahan, serta saran-saran sehingga Tugas Akhir ini dapat diselesaikan.
3. Bapak dan Ibu Dosen Program Studi Informatika Fakultas Teknologi Industri Universitas Islam Indonesia yang telah memberikan ilmu pengetahuan yang bermanfaat sehingga dapat terselesaikannya Tugas Akhir ini.
4. Semua rekan-rekan Mahasiswa seperjuangan Program Studi Informatika angkatan 2017 dan semua pihak yang telah membantu dalam penyusunan Tugas Akhir ini.

Semoga Allah SWT membalas segala kebaikan mereka yang telah membantu penyusunan Tugas Akhir ini dan penulis berharap semoga tugas akhir yang sederhana ini dapat bermanfaat bagi kita semua. Aamiin.

Yogyakarta, 4 Mei 2021

(Achmad Noer Aziz)

SARI

Penelitian ini menerapkan teknologi *Machine Learning* tepatnya dengan menggunakan algoritme *Convolutional Neural Network* dalam pengenalan dan klasifikasi alfabet Bahasa Isyarat Indonesia (BISINDO). Tiap negara memiliki bahasa isyaratnya masing-masing dan di Indonesia sendiri terdapat dua jenis bahasa isyarat yaitu BISINDO dan SIBI. Alasan memilih BISINDO sebagai objek penelitian ini karena lebih sering digunakan oleh penyandang tunawicara dan tunarungu dalam berkomunikasi sehari-hari. Diperkuat dengan fakta bahwa penyandang tunawicara dan tunarungu sangat mengandalkan bahasa isyarat untuk berkomunikasi, tetapi tidak banyak orang normal yang mampu menguasai bahasa isyarat, maka penelitian ini mengarah kepada meningkatkan aksesibilitas penyandang tunawicara dan tunarungu dalam berkomunikasi dengan bantuan teknologi yang sudah mumpuni saat ini, yaitu dengan menggunakan aplikasi *smartphone* untuk mengenali bahasa isyarat supaya pengenalan bahasa isyarat bisa dilakukan di manapun. Kecepatan dan keringanan komputasi juga menjadi pertimbangan penting dalam penelitian ini dengan maksud untuk mendapatkan hasil yang cukup baik dengan waktu komputasi secepat mungkin. Berdasarkan hasil penelitian yang dilakukan, dengan 100 *epoch* pelatihan model diselesaikan dengan menghasilkan akurasi *training* sebesar 98.94% dan akurasi *validation* sebesar 99.38%. Model juga mampu memberikan hasil yang cukup baik pada perangkat *smartphone* berbasis Android dengan akurasi kumulatif tertingginya mencapai 88.46% dan waktu pendeteksian citra selama 24 *millisecond* sekaligus membuktikan bahwa model yang dilatih cukup ringan untuk melakukan klasifikasi pada perangkat *smartphone* secara *realtime*. Ditambah lagi model juga sudah mampu untuk mengklasifikasikan data video.

Kata kunci: Bahasa Isyarat Indonesia (BISINDO), CNN, pengenalan citra, *realtime*, *smartphone* Android.

GLOSARIUM

Glosarium memuat daftar kata tertentu yang digunakan dalam laporan dan membutuhkan penjelasan, misalnya kata serapan yang belum lazim digunakan. Contohnya seperti di bawah ini:

Algoritme	dalam ilmu komputer, algoritme adalah sekumpulan aturan atau instruksi yang didesain untuk melakukan tugas dan menyelesaikan permasalahan.
Dataset	sekumpulan data atau contoh-contoh yang terdiri dari satu atau lebih fitur.
Epoch	satu proses pelatihan penuh atas seluruh dataset sehingga setiap contoh data telah melewati model sebanyak satu kali.
Hyperparameter	variabel yang digunakan untuk mengontrol proses pelatihan model.
Learning Rate	*hyperparameter* yang mengontrol seberapa banyak model harus diubah sebagai respons terhadap estimasi *error* setiap kali bobot model diperbarui.
Library	kumpulan rutinitas yang telah dikompilasi sebelumnya yang dapat digunakan oleh program.
Model	representasi dari apa yang telah dipelajari oleh sistem machine learning dari data pelatihan. Model mendefinisikan relasi antara fitur dan label.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Menurut data di Sistem Informasi Manajemen Penyandang Disabilitas (SIMPD) dari Kementrian Sosial, diantara penyandang disabilitas di Indonesia, sebanyak 13.026 orang merupakan penyandang disabilitas tuna rungu, dan 5.020 orang merupakan penyandang disabilitas tuna wicara ("Sistem Informasi Manajemen Penyandang Disabilitas Kementerian Sosial," n.d.). Tidak hanya di Indonesia saja tentunya, penyandang disabilitas juga banyak ditemukan di belahan dunia lain. Menurut konten yang diterbitkan oleh Pusat Data dan Informasi Kementerian Kesehatan Republik Indonesia, dijelaskan bahwa pada tahun 2019, *World Health Organization* (WHO) memperkirakan terdapat sekitar 466 juta orang di dunia mengalami gangguan pendengaran, di mana 34 juta diantaranya merupakan anak-anak. Sebanyak 360 juta atau sekitar 5,3% penduduk dunia mengalami ketulian. WHO juga memperkirakan pada tahun 2050 terdapat lebih dari 900 juta orang atau setiap satu dari sepuluh orang di dunia memiliki gangguan pendengaran. Berdasarkan hasil Riset Kesehatan Dasar (Riskesdas) yang dilaksanakan oleh Badan Penelitian dan Pengembangan Kesehatan (Balitbangkes) Kementerian Kesehatan tahun 2018, proporsi tuna rungu sejak lahir pada anak umur 24-59 bulan di Indonesia yaitu sebesar 0,11% (Harpini, 2019).

Berdasarkan data statistik yang sudah disebutkan, cukup banyak penyandang disabilitas tunawicara dan tunarungu di dunia dan diprediksi jumlahnya akan terus bertambah, sangat disayangkan tidak banyak media yang bisa digunakan untuk berkomunikasi bagi mereka dan salah satunya adalah bahasa isyarat. Sayangnya, tidak banyak orang normal yang mau belajar ataupun mampu menguasai bahasa isyarat mungkin karena minimnya edukasi terhadap penggunaan bahasa isyarat itu sendiri. Ditambah lagi susah menemukan media untuk belajar bahasa isyarat secara mandiri. Hal tersebut menyebabkan orang normal kesulitan apabila ingin berkomunikasi dengan penyandang disabilitas tunawicara dan tunarungu, dan begitupun sebaliknya.

Selain untuk berkomunikasi dengan sesama manusia, bahasa isyarat atau dalam konteks umum adalah gestur tangan, juga dapat digunakan sebagai media komunikasi antara manusia dengan komputer atau yang biasa disebut dengan *human-computer interaction*. Gerakan tubuh,

ekspresi wajah, gestur tangan dan sejenisnya merupakan media yang biasa dijadikan *input* pada komputer dalam kasus *human-computer interaction*.

Berdasarkan fakta dan masalah umum yang telah disebutkan sebelumnya, peneliti memutuskan untuk melakukan penelitian dalam mengembangkan model pengenalan bahasa isyarat. Tidak sampai situ saja, peneliti juga membuat model yang akan dikembangkan tersebut dapat dijalankan pada perangkat *smartphone* sehingga bahasa isyarat bisa dikenali secara *realtime*. Perangkat *smartphone* juga cukup ringan dan bisa dibawa ke mana saja yang berarti pengenalan bahasa isyarat juga bisa dilakukan di manapun dan kapanpun. Apabila terbukti berhasil, hal tersebut membuktikan bahwa pengenalan bahasa isyarat sudah cukup ringan dalam artian tidak membutuhkan sumber daya yang besar karena dapat dijalankan pada perangkat *smartphone*. Solusi tersebut dirasa cukup sesuai untuk mengatasi masalah bagi para penyandang tunawicara dan tunarungu dalam berkomunikasi kepada orang yang tidak paham bahasa isyarat, dan juga berlaku sebaliknya.

1.2 Rumusan Masalah

Adapun rumusan masalah seperti di bawah ini:

1. Arsitektur *Convolutional Neural Network* (CNN) seperti apa yang sesuai dalam mengenali alfabet Bahasa Isyarat Indonesia (BISINDO)?
2. Apa latar yang cukup sesuai untuk digunakan dalam tahap pengujian sehingga dapat menghasilkan nilai akurasi yang cukup baik?
3. Apakah model mampu mengenali alfabet Bahasa Isyarat Indonesia (BISINDO) secara *realtime* menggunakan *smartphone* dengan baik?

1.3 Batasan Masalah

Adapun batasan masalah pada penelitian ini yaitu;

1. Bahasa yang digunakan adalah alfabet Bahasa Isyarat Indonesia (BISINDO)
2. Citra isyarat tangan Bahasa Isyarat Indonesia (BISINDO) diambil dari tampak depan.
3. Latar dari citra yang digunakan adalah latar berwarna putih atau sejenisnya.
4. Aplikasi TensorFlow Lite dijalankan di *smartphone* dengan sistem operasi Android.

1.4 Tujuan Penelitian

Adapun tujuan penelitian sebagai berikut:

1. Menemukan arsitektur *Convolutional Neural Network* (CNN) yang paling sesuai dalam mengembangkan model pengenalan alfabet Bahasa Isyarat Indonesia (BISINDO) pada penelitian ini.
2. Menguji kemampuan model dalam mengenali data dengan pemilihan latar yang beragam.
3. Melihat performa pengenalan yang dihasilkan oleh model saat diujikan secara *realtime*.

BAB II
TINJAUAN PUSTAKA

2.1 Bahasa Isyarat di Indonesia

Bahasa isyarat mampu menunjukan identitas seorang penyandang tunawicara dan tunarungu. Saat mereka berada di tengah-tengah masyarakat, bahasa isyarat lah yang menjadi penanda keberadaan tunawicara dan tunarungu untuk mudah dikenali. Selain itu, keberadaan bahasa merupakan bagian dari budaya seseorang tak hanya untuk tunawicara dan tunarungu tetapi juga bagi masyarakat pada umumnya. Bahasa isyarat pun demikian, keberadaan nya tak bisa terlepas dari hasil budaya penyandang tunawicara dan tunarungu. Bahasa isyarat merupakan ciri khas dan hasil interaksi alami yang terjadi antara tunawicara dan tunarungu dan lingkungannya (Gumelar, Hafiar, & Subekti, 2018).

2.1.1 Sejarah Bahasa Isyarat di Indonesia

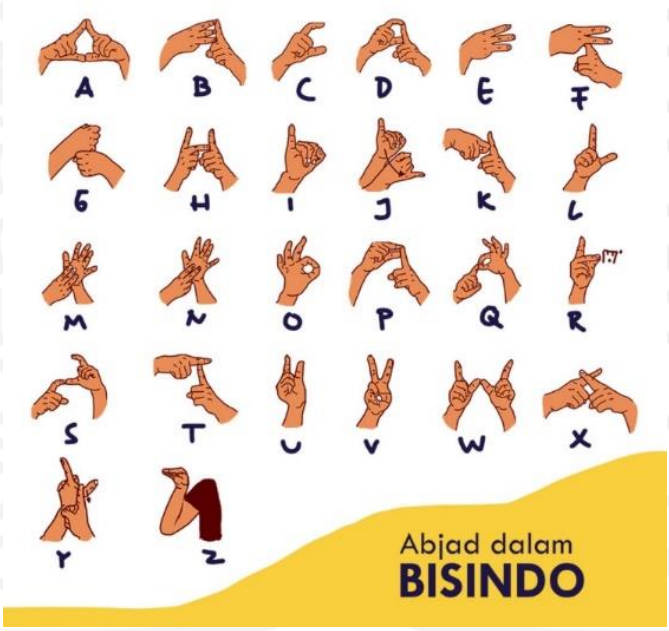
Di Indonesia, terdapat dua bahasa isyarat yang sering digunakan. Yang pertama namanya Sistem Bahasa Isyarat Indonesia (SIBI) dan yang kedua namanya Bahasa Isyarat Indonesia (BISINDO). SIBI merupakan bahasa isyarat yang diciptakan oleh Alm. Anton Widyatmoko mantan kepala sekolah SLB/B Widya Bakti Semarang bekerjasama dengan mantan kepala sekolah SLB/B di Jakarta dan Surabaya. SIBI telah memiliki kamus yang diterbitkan oleh pemerintah dan disebarluaskan melalui sekolah-sekolah khususnya SLB/B untuk tunarungu di Indonesia sejak tahun 2001. Keberadaan SIBI begitu populer di sekolah-sekolah SLB/B di Indonesia ditambah lagi SIBI merupakan bahasa isyarat yang secara resmi dirilis oleh pemerintah Indonesia (Gumelar et al., 2018).

Penggunaan SIBI tidak sepenuhnya diterima dan digunakan oleh penyandang tunawicara dan tunarungu. Seringkali mereka mengalami kesulitan dalam menggunakan SIBI untuk komunikasi sehari-hari. Hal ini karena penerapan kosakata yang tidak sesuai dengan aspirasi dan nurani tunawicara dan tunarungu, terlebih penerapan bahasa yang terlalu baku dengan tata bahasa kalimat bahasa Indonesia yang membuat kesulitan tunawicara dan tunarungu untuk berkomunikasi. Kemudian dalam SIBI ditemukan banyak pengaruh alami, budaya, dan isyarat tunawicara dan tunarungu dari luar negeri yang sulit dimengerti sehingga SIBI sulit dipergunakan oleh penyandang tunawicara dan tunarungu untuk berkomunikasi. SIBI hanya dapat digunakan sebagai bahasa isyarat di sekolah dan tidak dapat dipergunakan sebagai bahasa

isyarat komunikasi sehari-hari tunawicara dan tunarungu dalam berkomunikasi (Gumelar et al., 2018).

2.1.2 Bahasa Isyarat Indonesia (BISINDO)

Seperti yang sudah dijelaskan sebelumnya, bahasa isyarat di Indonesia terdapat dua macam yaitu Sistem Bahasa Isyarat Indonesia (SIBI) dan Bahasa Isyarat Indonesia (BISINDO). Pada penelitian ini, BISINDO dipilih menjadi topik utama dalam penelitian, untuk isyaratnya bisa dilihat pada Gambar 2.1Gambar 2.1. Selain itu, BISINDO lebih identik dengan Indonesia dibandingkan dengan SIBI, karena SIBI diadopsi dari *American Sign Language* (ASL).



Gambar 2.1 Bahasa Isyarat Indonesia (BISINDO)
Sumber: www.ypedulikasihabk.org

BISINDO lebih umum di kalangan pengguna bahasa isyarat daripada SIBI. BISINDO juga muncul secara alami di antara pengguna bahasa isyarat yang membuat bahasanya terasa lebih fleksibel. SIBI secara resmi dirilis oleh pemerintah, tetapi kemungkinan besar akan digunakan untuk pembelajaran di sekolah luar biasa. SIBI dianggap lebih sulit karena mengandung kosakata standar dan rumit, serta memiliki prefiks dan sufiks. Berbeda dengan BISINDO, SIBI hanya menggunakan satu tangan (KLOBILITY, n.d.).

2.2 Referensi Penelitian

Cukup banyak penelitian yang membahas tentang isyarat tangan secara umum dan tujuannya pun bermacam-macam. Beberapa penelitian dapat dijadikan referensi yang baik untuk mendalami lebih jauh tentang perkembangan teknologi pengenalan isyarat tangan.

2.2.1 Pengumpulan Referensi Penelitian

Dalam menyelesaikan sebuah masalah, terdapat metode yang bisa digunakan untuk membantu menyelesaikan masalah tersebut. Dalam kasus ini, masalah yang dihadapi adalah pengenalan citra bahasa isyarat dan gestur tangan. Dengan menggunakan metode yang tepat, dapat membantu untuk mendapatkan hasil yang maksimal juga. Oleh karena itu, memilih metode menjadi penting dalam menyelesaikan suatu masalah dalam penelitian.

Untuk mendukung semua itu, telah terkumpul sebanyak 23 literatur yang membahas topik pengenalan bahasa isyarat dan gestur tangan menggunakan teknologi *Machine Learning*. Literatur yang dikumpulkan dibatasi hanya literatur yang terbit dalam satu dekade terakhir, atau dengan kata lain literatur yang terbit antara tahun 2010 sampai 2020. Terdapat empat kata kunci yang digunakan sebagai referensi dalam mencari literatur yaitu *Hand Gesture Recognition*, *Sign Language Recognition*, *Human-Computer Interaction*, dan *Neural Network*. Keempat kata kunci tersebut juga diterjemahkan dan kemudian dicari dalam bahasa Indonesia. Pencarian kata kunci dalam bahasa Indonesia juga dilakukan karena di Indonesia terdapat dua versi bahasa isyarat yaitu SIBI dan BISINDO. Oleh karena itu, semoga literatur-literatur ini dapat memberikan wawasan tentang keberadaan bahasa isyarat dalam versi bahasa Indonesia.

Tabel 2.1 Jenis algoritme dan jumlah penggunaannya

Algoritme Ekstraksi Fitur dan Klasifikasi	Tipe	Jumlah
Convolutional Neural Network	Ekstraksi Fitur	14
Support Vector Machine	Pengklasifikasi	5
K-Nearest Neighbor	Ekstraksi Fitur	4
Multi-Layer Perceptron	Ekstraksi Fitur	3
Multiclass Support Vector Machine	Pengklasifikasi	3
Fuzzy Inference System	Pengklasifikasi	2
Convex Hull	Ekstraksi Fitur	1
Convexity Defects	Ekstraksi Fitur	1
Long Short-Term Memory	Ekstraksi Fitur	1
Random Forest	Pengklasifikasi	1

Dilihat dari Tabel 2.1, ditemukan bahwa *Convolutional Neural Network* adalah metode yang paling banyak digunakan, ditemukan dalam 14 literatur. Lalu ada *Support Vector Machine* sebagai pengklasifikasi yang paling banyak digunakan, ditemukan dalam 5 literatur. Beberapa literatur juga mencantumkan lebih dari satu metode yang digunakan dalam satu penelitian, sehingga masing-masing metode akan dihitung sebagai satu.

Kombinasi dari beberapa metode dapat digunakan untuk mendapatkan hasil yang optimal, seperti dalam literatur yang ditulis oleh Li et al. (Li et al., 2019) dan Yolanda et al. (Yolanda, Gunadi, & Setyati, 2020). Beberapa literatur juga menggunakan lebih dari satu metode dalam sebuah penelitian tetapi kemudian membandingkan satu metode dengan metode lainnya untuk menentukan metode mana yang terbaik. Contohnya dapat ditemukan dalam literatur yang ditulis oleh Elsayed et al. (Elsayed, Sayed, & Abdalla, 2018).

2.2.2 Penelitian Terdahulu

Menurut literatur yang ditulis oleh Li dan kawannya (Li et al., 2019). Akurasi model *Machine Learning* mereka berhasil menyentuh angka 98,52%. Setelah dianalisis, terdapat beberapa kemungkinan sehingga dapat menghasilkan kinerja yang terbilang cukup baik. Hal pertama adalah menggabungkan beberapa metode. Dinyatakan bahwa mereka menggunakan *Convolutional Neural Network* sebagai metode ekstraksi fitur, kemudian menggunakan *Support Vector Machine* sebagai pengklasifikasi citra. Kedua, mereka melakukan *denoise processing* karena terdapat banyak *noise* pada citra yang menyebabkan rendahnya kualitas *dataset,* sehingga citranya perlu disaring terlebih dahulu. Mereka menggunakan algoritme *mean filter*, tetapi itu hanya dapat mengurangi *noise* dan tidak menghilangkan *noise*. Beralih ke metode, mereka juga mengklaim bahwa metode *Long Short-Term Memory* sebenarnya dapat digunakan karena memiliki kinerja yang lebih baik daripada *Convolutional Neural Network* dalam dependensi waktu yang panjang. Namun, *Convolutional Neural Network* memiliki keunggulan yaitu dapat dikonfigurasi untuk menggunakan *feature map* yang berbeda untuk menangkap objek dan juga objek diambil berkali-kali. Jadi, mereka lebih memilih menggunakan *Convolutional Neural Network* daripada *Long Short-Term Memory* karena alasan tersebut.

Ada juga beberapa alasan yang diyakini dapat membuat kinerja menjadi lebih baik. Melihat kepada literatur yang ditulis oleh Elsayed et al. (Elsayed et al., 2018) dan Mufarroha & Utaminingrum (Mufarroha & Utaminingrum, 2017) di mana mereka sama-sama menggunakan metode *K-Nearest Neighbor*. Mereka menambahkan proses penghilangan latar

agar proses pendeteksian lebih cepat daripada citra yang memiliki latar menurut klaim mereka. Ada pun literatur seperti yang ditulis oleh Nguyen et al. (Nguyen, Huynh, & Meunier, 2013) yang mengubah citra dengan format *red*, *green*, dan *blue* (RGB) atau berwarna menjadi citra biner atau hitam putih. Selain itu, disebutkan juga bahwa intensitas cahaya juga sangat berpengaruh dalam proses pelatihan karena intensitas cahaya dapat memanipulasi lensa kamera dalam menangkap warna kulit bisa menjadi lebih terang ataupun lebih gelap dari biasanya. Hal inilah yang dapat mempengaruhi kinerja pembelajaran model. Oleh karena itu, beberapa metode digunakan untuk memanipulasi perubahan warna kulit tersebut, salah satunya yang telah disebutkan, yaitu mengubah format citra yang awalnya berwarna menjadi hitam putih.

Bicara soal durasi komputasi, hanya tulisan Oyedotun & Khashman (Oyedotun & Khashman, 2017) yang menuliskan waktu komputasinya. Dalam penelitiannya, metode yang digunakan yaitu *Convolutional Neural Network* tetapi dimanipulasi dengan berbagai konfigurasi yang berbeda. Dengan jumlah sampel citra dan *learning rate* yang sama, Oyedotun & Khashman merubah jumlah *epoch* dan jumlah *hidden layer*. Meski perubahannya hanya sedikit, efek yang dihasilkan cukup terlihat. Pada konfigurasi pertama, dibutuhkan waktu pelatihan selama 523 detik, dengan 14.400 *epoch*, dan 2 *hidden layer*. Pada konfigurasi kedua, dibutuhkan waktu pelatihan selama 620 detik, dengan 14.400 *epoch*, dan 3 *hidden layer*. Dalam konfigurasi ketiga, dibutuhkan waktu pelatihan selama 745 detik, dengan 20.000 *epoch*, dan 4 *hidden layer*. Dari sini dapat dilihat bahwa konfigurasi juga dapat mempengaruhi kinerja. Untuk mendapatkan hasil yang optimal, proses optimasi *hyperparameter* atau beberapa orang mungkin mengatakan *tuning* dapat dilakukan dalam kondisi seperti ini. Dengan konfigurasi yang sesuai, tentunya model dapat dilatih dalam waktu yang lebih singkat dan dengan hasil yang baik.

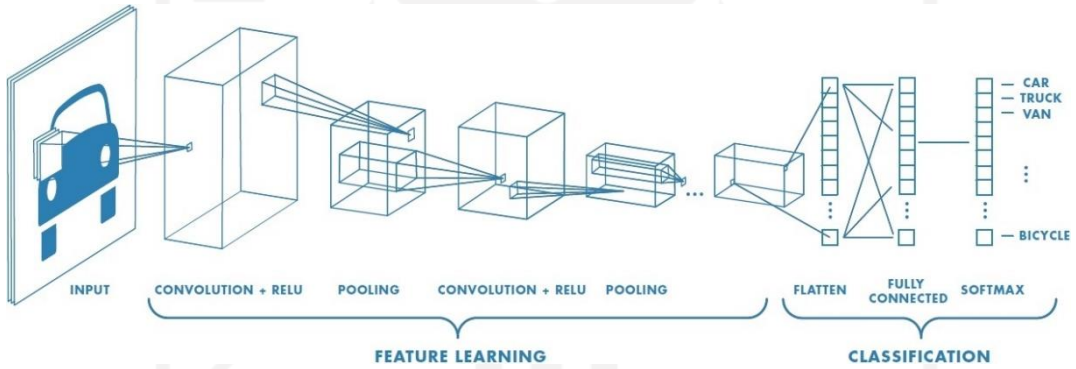
2.2.3 Penelitian yang Diusulkan

Penelitian yang akan dilakukan pada Tugas Akhir ini adalah mengembangkan model pengenalan alfabet Bahasa Isyarat Indonesia (BISINDO) menggunakan metode *Convolutional Neural Network* (CNN). Apabila pada penelitian terdahulu kebanyakan melakukan pengenalan pada data citra saja dan hanya dua literatur yang mengunakan data video milik Jayaprakash (Jayaprakash & Majumder, 2011) dan Patel (Patel, Dhakad, Desai, Gupta, & Correia, 2018). Penelitian ini berusaha menggapai titik yang lebih jauh lagi yaitu melakukan pengenalan bahasa isyarat secara *realtime* dan dilakukan di perangkat *smartphone*. Tantangannya adalah mengembangkan model yang cukup ringan sehingga dapat dijalankan pada perangkat

smartphone. Data dengan latar yang asing disiapkan untuk pengujian guna melihat kemampuan model dalam mengenali data yang belum pernah dilihat atau dipelajari sebelumnya.

2.3 *Convolutional Neural Network*

Convolutional Neural Network (CNN) sangat berguna untuk menemukan pola pada objek dalam konteks untuk mengenali benda, wajah, pemandangan, dan lain-lain. CNN juga sangat efektif untuk menglasifikasikan objek non-citra seperti audio, *time series*, dan data sinyal. Menggunakan CNN dalam dunia *Deep Learning* sangat populer karena tiga faktor penting yaitu, CNN dapat mempelajari fitur secara otomatis sehingga proses ekstraksi fitur tidak perlu dilakukan secara manual, CNN dapat menghasilkan hasil pengenalan yang cukup akurat, dan model CNN dapat dilatih ulang untuk pengenalan data baru, yang di mana memungkinkan untuk membangun model baru menggunakan jaringan yang sudah ada sebelumnya (MATLAB, n.d.). Atau lebih populer dengan sebutan *transfer learning*.



Gambar 2.2 Ilustrasi arsitektur CNN

Sumber: www.mathworks.com

Seperti jaringan saraf lainnya, CNN terdiri dari *input layer*, *output layer*, dan terdapat *hidden layer* diantaranya. Lapisan-lapisan ini melakukan operasi yang mengubah data dengan maksud mempelajari fitur-fitur khusus yang dimiliki oleh data. Tiga lapisan yang paling umum adalah konvolusi, fungsi aktivasi atau *ReLU*, dan pooling. Konvolusi menaruh citra masukan melalui serangkaian filter konvolusi, yang di mana masing-masing filter berfungsi mengaktifkan fitur tertentu dari citra. *Rectified linear unit* (*ReLU*) memungkinkan pelatihan yang lebih cepat dan lebih efektif dengan memetakan nilai negatif menjadi nol dan mempertahankan nilai positif. Hal ini kadang disebut sebagai fungsi aktivasi, karena hanya fitur yang diaktifkan yang dibawa ke lapisan berikutnya. *Pooling* menyederhanakan keluaran dengan melakukan non-linear *downsampling*, mengurangi jumlah parameter yang perlu

dipelajari oleh jaringan. Operasi ini diulang lebih dari puluhan atau bahkan ratusan lapisan, dengan setiap lapisan belajar untuk mengidentifikasi atau mengenali fitur-fitur yang berbeda-beda (MATLAB, n.d.). Ilustrasi arsitektur CNN bisa dilihat pada Gambar 2.2.

2.3.1 Lapisan Konvolusi

Sebuah jaringan saraf biasa mengenali citra berdasarkan *pixel-pixel* yang terdapat pada citra. Teknik yang lebih optimal adalah dengan menggunakan lapisan konvolusi di mana alih-alih mengenali objek berdasarkan *pixel-pixel*, jaringan saraf dapat mengenali objek berdasarkan atribut-atribut yang memiliki lebih banyak informasi. Lapisan konvolusi berfungsi untuk mengenali fitur-fitur unik pada sebuah objek. Lapisan konvolusi dapat mengenali fitur pada objek dengan menggunakan filter. Filter hanyalah sebuah matriks n x n yang berisi angka-angka.

Lapisan konvolusi adalah pembangkit tenaga utama model CNN. Bekerja untuk mendeteksi fitur-fitur penting secara otomatis hanya bergantung pada citra dan label bukanlah tugas yang mudah. Lapisan konvolusi mempelajari fitur kompleks dengan saling menumpuk lapisannya satu sama lain. Misalnya lapisan pertama untuk mendeteksi tepi, lapisan berikutnya menggabungkannya untuk mendeteksi bentuk, lapisan berikutnya menggabungkan informasi-informasi tadi untuk menyimpulkan objek apa yang sedang dideteksi. Sebagai catatan, CNN pada dasarnya tidak tahu objek apa yang sedang dilihatnya. Dengan melihat objeknya secara berkali-kali, model mulai belajar untuk mendeteksinya sebagai sebuah fitur (Dertat, 2017).

2.3.2 Lapisan *Pooling*

Pada sebuah jaringan saraf, umumnya setelah proses konvolusi pada citra masukan, akan masuk ke dalam proses *pooling*. *Pooling* adalah proses untuk mengurangi resolusi citra dengan tetap mempertahankan informasi atau fitur-fitur penting pada citra.

Salah satu contoh dari *pooling* adalah *maxpooling*. Pada *maxpooling*, di antara setiap area dengan luas *pixel* tertentu, akan diambil satu buah *pixel* dengan nilai tertinggi. Hasilnya akan dikumpulkan sehingga menjadi citra baru. Walaupun disebut sebagai citra baru, sebenarnya citra tersebut merupakan kumpulan fitur yang terpilih dari citra pada lapisan sebelumnya.

2.3.3 Lapisan *Fully-Connected*

Lapisan *fully-connected* bertugas untuk membaca fitur-fitur yang dihasilkan oleh lapisan konvolusi untuk menglasifikasikan objek dengan benar. Neuron dalam lapisan *fully-connected*

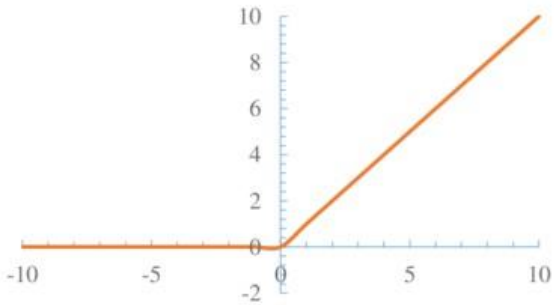
memiliki keterikatan penuh terhadap seluruh aktivasi yang berada di lapisan-lapisan sebelumnya, sama seperti yang terlihat pada jaringan syaraf pada umumnya.

2.4 Fungsi Aktivasi

Fungsi aktivasi digunakan untuk memetakan nilai yang dihasilkan menjadi nilai yang diperlukan, misalnya antara (0, 1) atau (-1, 1). Fungsi aktivasi lah yang memungkinkan jaringan saraf dapat mengenali pola non-linier yang kompleks. Tanpa fungsi aktivasi, jaringan saraf hanya bisa mengenali pola linier seperti garis pada regresi linier ("Dicoding Indonesia," n.d.-a). Ada dua fungsi aktivasi yang digunakan dalam penelitian ini yaitu *rectified linear unit* (*ReLU*) dan *sigmoid*. Berikut adalah penjelasannya dari keduanya.

Rectified Linear Unit (*ReLU*)

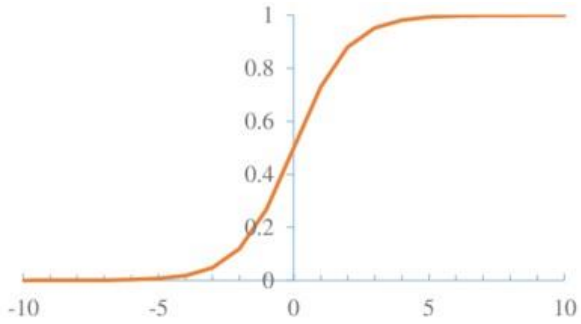
Fungsi *ReLU* bersifat kontinu meski kemiringannya berubah secara tiba-tiba dan nilai turunannya bernilai 0 pada $z < 0$, lihat Gambar 2.3. Akan tetapi, fungsi ini bekerja dengan sangat baik dan membuat jaringan bekerja secara efisien sehingga mempercepat waktu komputasi. Karena hal inilah, fungsi aktivasi ini sering digunakan sebagai fungsi aktivasi *default* pada jaringan saraf ("Dicoding Indonesia," n.d.-a).



Gambar 2.3 Fungsi aktivasi *ReLU*
Sumber: (Yang & Yang, 2018)

Sigmoid

Fungsi aktivasi ini berada di antara nilai 0 hingga 1 sehingga biasanya digunakan untuk memprediksi model probabilitas yang outputnya ada di kisaran 0 dan 1, lihat Gambar 2.4. Dengan kemiringan yang halus atau bisa disebut *smooth gradient* memungkinkan *Gradient Descent* berprogres pada setiap langkahnya ("Dicoding Indonesia," n.d.-a). *Gradient Descent* merupakan algoritme pengoptimalan yang mampu menemukan solusi optimal untuk berbagai masalah.



Gambar 2.4 Fungsi aktivasi *Sigmoid*
Sumber: (Yang & Yang, 2018)

2.5 *Batch Normalization*

Batch normalization adalah teknik *supervised learning* yang mengubah keluaran *interlayer* menjadi jaringan saraf ke dalam format terstandar, yang disebut normalisasi. Hal ini secara efektif mengatur ulang distribusi keluaran dari lapisan sebelumnya agar lebih efisien saat diproses oleh lapisan berikutnya. Pendekatan ini menghasilkan kecepatan pembelajaran yang lebih cepat karena normalisasi memastikan tidak ada nilai aktivasi yang terlalu tinggi atau terlalu rendah, serta memungkinkan setiap lapisan untuk belajar secara mandiri. Sebagai tambahan, lapisan *batch normalization* digunakan untuk menstabilkan proses pembelajaran dengan cara menstandarisasi *input* yang akan masuk ke sebuah lapisan (Goodfellow, Bengio, & Courville, 2016).

2.6 *Dropout*

Dropout adalah metode regularisasi yang mengira-ngira pelatihan jaringan syaraf dalam jumlah besar dengan arsitektur yang berbeda secara paralel. *Dropout* bekerja secara probabilistik menghapus atau melepaskan masukan ke dalam lapisan yang mungkin saja merupakan variabel masukan dalam data sampel atau aktivasi dari lapisan sebelumnya. Hal ini memiliki efek terhadap mensimulasikan jaringan dengan jumlah yang besar dengan struktur jaringan yang sangat berbeda dan pada gilirannya, membuat node dalam jaringan secara umum lebih kuat untuk masukan (Brownlee, 2018).

2.7 *Loss Function*

Untuk *loss function*, dapat menggunakan *sparse categorical entropy* pada kasus klasifikasi tiga kelas atau lebih. Untuk masalah dua kelas, *loss function* yang lebih tepat adalah *binary cross entropy*. Singkatnya, pemilihan *loss function* secara langsung berkaitan dengan

fungsi aktivasi yang digunakan di lapisan keluaran jaringan saraf karena kedua elemen ini saling berhubungan.

2.8 *Optimizer*

Optimizer adalah sebuah metode yang digunakan untuk merubah atribut model *Machine Learning* seperti bobot dan *learning rate* untuk mengurangi *loss*. *Optimizer* sederhananya mengarahkan model untuk bisa memaksimalkan performanya dengan memainkan bobot model. *Loss function* hanya memberi tahu *optimizer* apakah ia bergerak ke arah yang benar atau salah (Ritacheta, 2020). Ada banyak algoritme lain yang telah terbentuk dari turunan *gradient descent* seperti *Adaptive Moment Estimation* (Adam), *RMSprop*, dan *Stochastic Gradient Descent* (SGD). Ketiga *optimizer* ini dikenal dari kecepatannya, ketangguhannya, dan fleksibilitasnya.

2.9 *Callback*

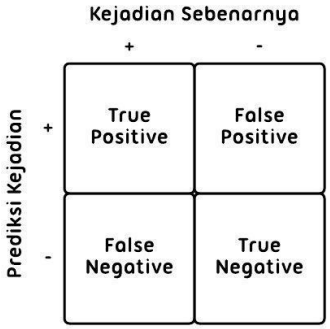
Keras menyediakan kemampuan untuk memanggil *callback* saat melatih model. Salah satu *callback* yang biasa digunakan saat melatih model adalah *callback* histori. Histori ini mencatat metrik pelatihan untuk setiap *epoch*nya termasuk *loss* dan akurasi serta *loss* dan akurasi untuk data *validation* jika ada. Objek histori dikembalikan dari fungsi yang digunakan untuk melatih model. Metrik disimpan dalam sebuah variabel sehingga bisa dipanggil kapan saja (Brownlee, 2016).

2.10 *Tuning Hyperparameter*

Tuning model *Machine Learning* merupakan masalah pengoptimalan saja. Yang perlu dilakukan adalah mengubah nilai-nilai *hyperparameter* dan menemukan kombinasi yang tepat sehingga dapat mencapai *loss* yang minimum dan akurasi yang maksimum. Hal ini penting dalam menemukan model *Machine Learning* dengan performa terbaik (Ippolito, n.d.)

2.11 *Confusion Matrix*

Untuk mengukur performa sebuah model dalam kemampuan klasifikasi dapat dilihat dari parameter pengukuran performanya seperti akurasi, *recall*, presisi, dan skor f1. Parameter tersebut bisa dihitung dengan melihat *confusion matrix*.



Gambar 2.5 *Confusion matrix*

Pada *confusion matrix* ada empat kemungkinan keputusan yang mungkin terjadi yaitu, *True Positive*, *True Negative*, *False Positive*, dan *False Negative* seperti pada Gambar 2.5. *True Positive* (TP) yaitu teridentifikasi positif terjadi X, dan model keputusan juga memutuskan adanya kejadian X. *True Negative* (TN) yaitu persentase tidak terjadi X, dan model keputusan juga tidak memutuskan ada kejadian X. *False Positive* (FP) yaitu tidak teridentifikasi positif terjadi X, namun model keputusan memutuskan ada kejadian X. *False Negative* (FN) yaitu teridentifikasi positif terjadi X, namun model keputusan tidak mengidentifikasi kejadian X.

Berdasarkan nilai *confusion matrix*, barulah nilai akurasi, *recall*, presisi, dan skor f1 bisa dihitung. Akurasi merupakan rasio prediksi benar dengan keseluruhan data. *Recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar. Presisi merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. Skor F1 merupakan perbandingan rata-rata presisi dan *recall* yang dibobotkan.

Nilai akurasi dapat dihitung dengan menggunakan persamaan berikut.

$$\text{Akurasi} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} . 100\% \quad (2.1)$$

Nilai *recall* dapat dihitung dengan menggunakan persamaan berikut.

$$Recall = \frac{\text{T}P}{\text{TP} + \text{FN}} . 100\% \quad (2.2)$$

Nilai presisi dapat dihitung dengan menggunakan persamaan berikut.

$$\text{Presisi} = \frac{\text{T}P}{\text{TP} + \text{FP}} . 100\% \quad (2.3)$$

Nilai skor f1 dapat dihitung dengan menggunakan persamaan berikut.

$$\text{Skor F1} = 2 . \frac{\text{Nilai } Recall \text{ x Nilai Presisi}}{\text{Nilai } Recall + \text{Nilai Presisi}} \quad (2.4)$$

2.12 *Overfit* dan *Underfit*

Salah satu hal yang paling penting untuk diperhatikan saat mengembangkan model *Machine Learning* adalah mengecek apakah model tersebut *underfit* atau *overfit*. Sebuah model yang tidak *overfit* dan *underfit* juga disebut dengan model *good fit*. ("Dicoding Indonesia," n.d.-b). *Overfit* terjadi ketika model memiliki prediksi yang terlalu baik pada data *training*, namun prediksinya buruk pada data *testing*. Ketika sebuah model *overfit*, model tidak dapat melakukan generalisasi dengan baik sehingga akan membuat banyak kesalahan dalam memprediksi data-data baru yang ditemuinya. Sebuah model dapat dikatakan *underfit* jika memiliki eror yang tinggi pada data *training*. *Underfit* menandakan bahwa model tersebut belum cukup baik dalam mengenali pola yang terdapat pada data *training*. Pada kasus klasifikasi, *underfitting* ditandai ketika model memiliki akurasi yang rendah pada data *training* ("Dicoding Indonesia," n.d.-b).

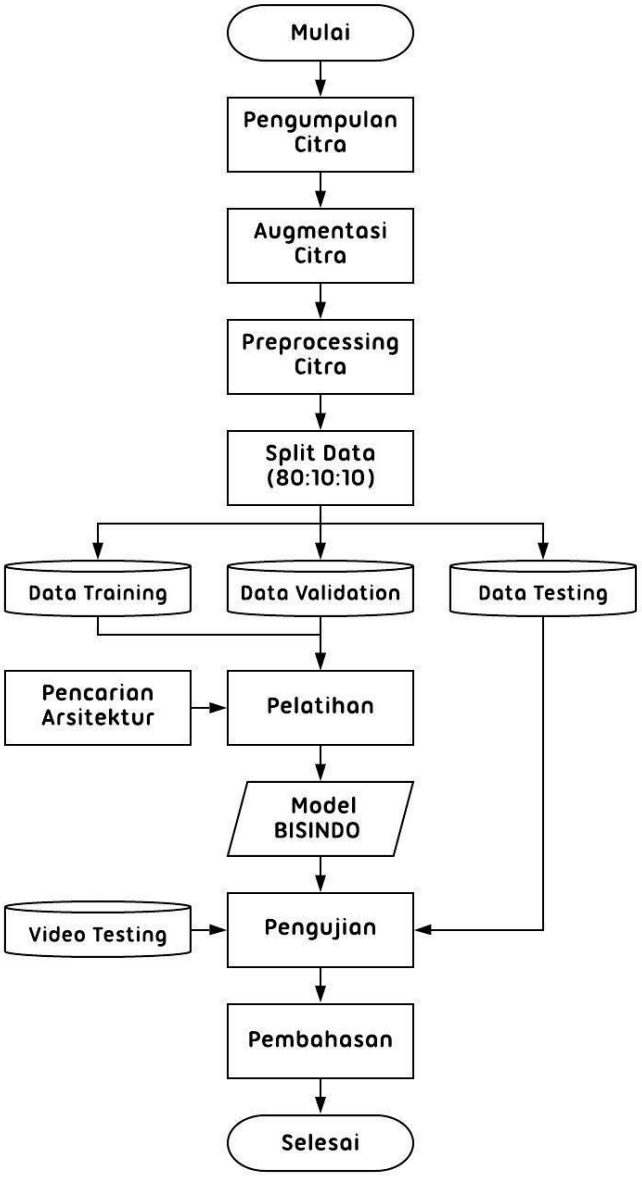
2.13 *Random Seed*

Fungsi *seed* digunakan untuk mengunci kondisi dari fungsi *random*, sehingga dapat menghasilkan nomor *random* yang sama setiap kali kode dieksekusi. Penggunaan nilai *seed* ini memastikan bahwa program akan mendapatkan hasil dan performa yang sama jika mengeksekusi program dengan nilai *seed* yang sama setiap kali menjalankan proses yang sama. Hal ini membuat program bisa dijalankan di mesin manapun tanpa khawatir hasilnya akan berbeda.

BAB III
METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

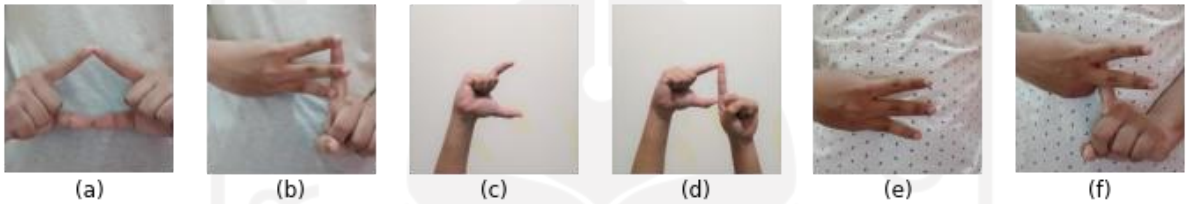
Flowchart pada Gambar 3.1 menunjukan urutan penelitian yang dilakukan dari awal sampai akhir. Dari tahap mengumpulkan *dataset* berupa citra, membuat model dan melatihnya untuk mengenali citra, hingga model tersebut diuji dan dilihat kemampuannya.



Gambar 3.1 *Flowchart* tahapan penelitian

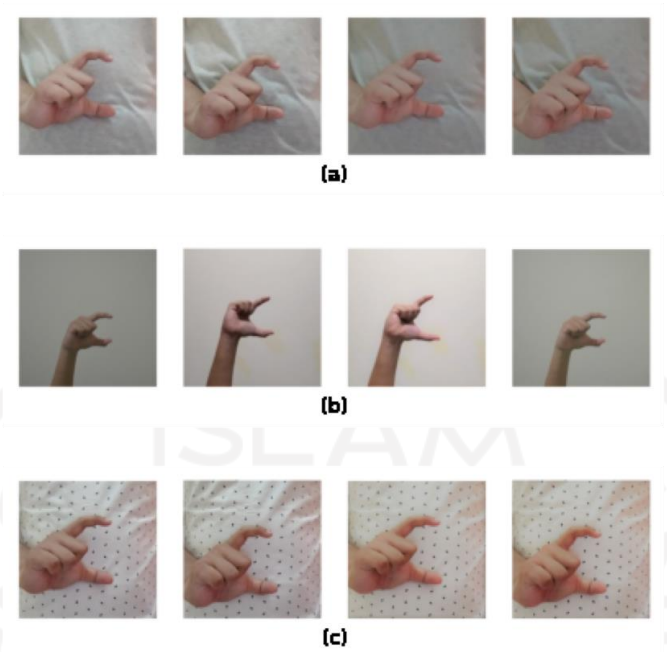
3.2 Pengumpulan Citra

Data primer dari penelitian ini adalah citra Bahasa Isyarat Indonesia (BISINDO) yang diambil menggunakan kamera peneliti. Ada tiga jenis kamera yang digunakan dalam mengambil citra, yaitu kamera DSLR, kamera belakang *smartphone*, dan kamera depan *smartphone*. Sudut pandang citra yang diambil adalah bahasa isyarat dari tampak depan dengan jarak antara objek dengan lensa kurang lebih sejauh 70 cm. Seluruh citra diambil saat mengenakan pakaian lengan pendek. Ada tiga latar yang digunakan dalam pengambilan citra, yaitu latar kaos berwarna putih, tembok berwarna putih, dan kemeja bercorak berwarna putih. Ketiga latar tersebut sebagian besar berwarna putih atau sejenisnya. Lokasi pengambilan citra dilakukan di dalam ruangan tertutup dan sumber cahaya yang didapatkan hanya berasal dari lampu. Sampel citra tangkapan kamera bisa dilihat pada Gambar 3.2.



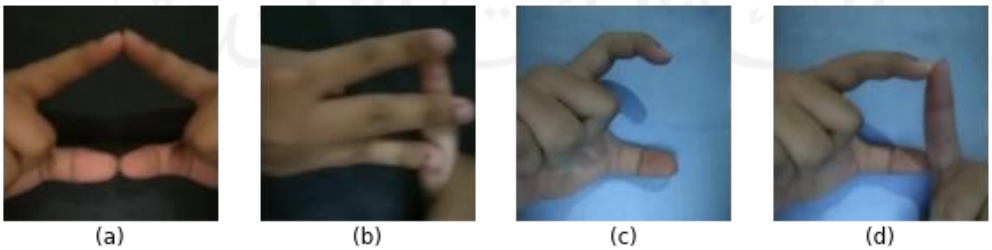
Gambar 3.2 Sampel citra tangkapan kamera (a) huruf A latar kaos putih (b) huruf B latar kaos putih (c) huruf C latar tembok putih (d) huruf D latar tembok putih (e) huruf E latar kemeja bercorak (f) huruf F latar kemeja bercorak

Selama proses pengambilan citra, diperoleh empat citra untuk masing-masing latar yang berarti 12 citra untuk masing-masing alfabetnya. Total citra dari huruf A sampai Z yang diperoleh sebanyak 312 citra. Gambar 3.3 memperlihatkan seluruh variasi citra tangkapan gambar untuk huruf C. Untuk huruf lainnya kurang lebih sama.



Gambar 3.3 Seluruh citra tangkapan kamera untuk huruf C (a) latar kaos (b) latar tembok (c) latar kemeja

Pengambilan citra yang diceritakan sebelumnya adalah citra yang digunakan pada proses pelatihan. Selanjutnya peneliti mengambil citra yang digunakan untuk tahap pengujian. Citra diambil dengan jarak antara objek dengan lensa kurang lebih sejauh 70 cm. Seluruh citra diambil saat mengenakan pakaian lengan pendek. Ada lima latar yang digunakan dalam pengambilan citra, yaitu latar kaos berwarna putih, tembok berwarna putih, kemeja bercorak berwarna putih, baju berwarna hitam, dan jubah berwarna biru. Contoh dari masing-masing latar bisa dilihat pada Gambar 3.2 dan Gambar 3.4. Lokasi pengambilan citra dilakukan di dalam ruangan tertutup dan sumber cahaya yang didapatkan hanya berasal dari lampu. Seluruh citra yang diambil akan dipangkas dan disimpan dengan resolusi 360 x 360 *pixel*.



Gambar 3.4 Sampel citra tangkapan kamera (a) huruf A latar baju hitam (b) huruf B latar baju hitam (c) huruf C latar jubah biru (d) huruf D latar jubah biru

Satu citra yang bersumber dari internet (Deaf, 2015) seperti pada Gambar 3.5 juga ditambahkan dalam tahap pengujian. Citra diperagakan oleh seseorang yang dirasa kompeten atau ahli dalam menggunakan bahasa isyarat khususnya BISINDO. Latar dari citra yang digunakan adalah kaos berwarna putih. Citra yang diambil akan dipangkas dan disimpan dengan resolusi 360 x 360 *pixel*. Selanjutnya, citra akan digandakan sebanyak 10 kali dengan konfigurasi suntingan yang berdeda-beda.

Gambar 3.5 Sampel citra bersumber dari internet (a) huruf E (b) huruf F

3.3 Augmentasi Citra

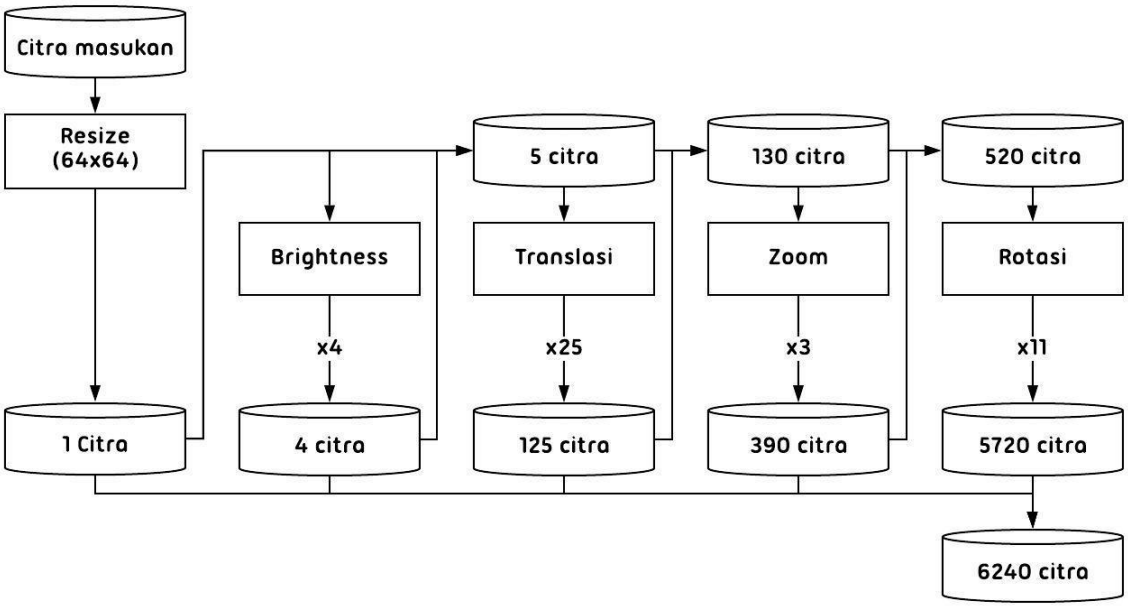
Berhubung citra yang diambil menggunakan kamera tidak cukup banyak, maka proses augmentasi bisa dilakukan supaya citra menjadi bervariasi sehingga jumlahnya akan semakin banyak. Ada empat jenis augmentasi yang dilakukan yaitu mengatur *brightness*, translasi citra, *zoom*, dan rotasi citra. Seluruh citra yang dihasilkan dalam proses augmentasi ini akan disimpan dengan resolusi 64 x 64 *pixel* dengan format *red*, *green*, dan *blue* (RGB). Gambar 3.6 memperlihatkan sampel gambar hasil augmentasi untuk masing-masing kelas.

Gambar 3.6 Sampel citra hasil proses augmentasi untuk masing-masing kelas

Pada Gambar 3.7 dijelaskan alur dari proses augmentasi apabila diilustrasikan hanya menggunakan satu citra masukan yang akan disebut citra awal. Pertama, citra awal akan di*resize* ke resolusi 64 x 64 lalu masuk ke tahap pengaturan *brightness* dan menghasilkan 4 citra sebagai keluarannya. Berarti sejauh ini terkumpul 5 citra yang berasal dari gabungan 4 citra keluaran dari tahap pengaturan *brightness* dan satu citra awal yang telah di*rezise*. Kemudian masuk ke tahap selanjutnya yaitu translasi citra. Pada tahap ini, setiap satu citra yang masuk akan menghasilkan 25 citra sebagai keluarannya. Berarti sejauh ini sudah terkumpul 130 citra gabungan dari 125 citra keluaran dari tahap translasi, 4 citra keluaran dari tahap pengaturan *brightness*, dan 1 citra awal yang telah di*resize.* Selanjutnya akan masuk ke tahap berikutnya yaitu *zoom*. Untuk satu citra yang masuk akan menghasilkan 4 citra sebagai keluaran dari tahap *zoom*, yang berarti sejauh ini terkumpul 520 citra gabungan dari 390 citra keluaran dari tahap *zoom*, 125 citra keluaran dari tahap translasi, 4 citra keluaran dari tahap pengaturan *brightness*, dan 1 citra awal yang telah di*resize*. Terakhir adalah tahap rotasi yang di mana menghasilkan 11 citra sebagai keluaran untuk satu citra yang masuk. Berarti total citra yang dihasilkan dari satu citra masukan menghasilkan total sebanyak 6240 citra gabungan dari

5720 citra keluaran dari tahap rotasi, 390 citra keluaran dari tahap *zoom*, 125 citra keluaran dari tahap translasi, 4 citra keluaran dari tahap pengaturan *brightness*, dan satu citra awal yang telah di*resize*.

Dapat dirangkum bahwa proses augmentasi menghasilkan sebanyak 6240 citra hanya dari satu citra yang masuk. Untuk masing-masing kelas terdapat 12 citra yang berarti setelah masuk proses augmentasi jumlahnya menjadi sebanyak 74880 citra. Tiap kelas memiliki 74880 citra lalu akan dikalikan dengan jumlah kelas alfabet yaitu 26 dan akan menghasilkan citra dengan totalnya sebanyak 1946880 citra.



Gambar 3.7 *Flowchart* proses augmentasi untuk satu citra masukan

Pada saat tahap platihan, khusus untuk data *training* dan data *validation* akan dipanggil melalui fungsi *ImageDataGenerator* sebelum citra dipelajari oleh model. Dalam fungsi tersebut citra ditambahkan filter *blur* dengan kernel 0x0 atau tanpa *blur*, kernel 1x1, dan kernel 3x3. Ketiga opsi tadi akan dipilih secara acak menggunakan fungsi *random*. Menambahkan filter *blur* juga dimaksudkan untuk memungkinkan model memiliki pengetahuan tentang data *blur* sehingga tidak akan menjadi masalah apabila citra yang diujikan nantinya kemungkinan terdapat *blur*. Gambar 3.8 memperlihatkan sampel gambar hasil augmentasi dengan filter blur untuk masing-masing kelas yang digunakan pada tahap pelatihan.

Gambar 3.8 Sampel citra hasil proses augmentasi dengan filter *blur* untuk masing-masing kelas

3.3.1 Pengaturan *Brightness*

Tahap pertama dalam proses augmentasi ialah mengatur tingkat *brightness* pada citra. Masing-masing citra akan dimanipulasi intensitas cahayanya berdasarkan tiga persentase yang telah ditentukan. Kategorinya yaitu intensitas cahaya diubah ke persentase 75% yang berarti digelapkan lalu ada 125% dan 150% yang berarti diterangkan. Dengan catatan, intensitas cahaya 100% setara dengan data masukannya.

3.3.2 Translasi Citra

Pada tahap translasi, setiap citra masukan akan digeser posisinya sebanyak satu sampai dua langkah ke setiap arah (atas, bawah, kiri, kanan, dan diagonal). Dengan catatan satu langkahnya bergeser sejauh 8 *pixel*.

Setelah dilihat keluarannya, ditemukan bahwa citra akan meninggalkan jejak hitam di sudut-sudutnya. Untuk mengatasi hal tersebut, maka seluruh citra keluaran akan di*zoom* dengan skala 107 % sehingga citra akan bersih dari jejak hitam.

3.3.3 *Zoom*

Pada tahap *zoom*, citra akan di*zoom* dengan tiga skala yang telah ditentukan. Skalanya yaitu 105 %, 110 %, dan 115 %. Dengan catatan, *zoom* dengan skala 100% setara dengan data masukannya.

3.3.4 Rotasi Citra

Pada tahap rotasi, citra akan dirotasi dimulai dari -5 derajat yang berarti rotasi miring ke kanan lalu akan ditambahkan derajatnya satu per satu hingga mencapai +5 derajat yang berarti rotasi miring ke kiri. Dengan catatan, rotasi citra sebanyak 0 derajat setara dengan data masukannya.

Setelah dilihat keluarannya, ditemukan bahwa citra akan meninggalkan jejak hitam di sudut-sudutnya. Untuk mengatasi hal tersebut, maka seluruh citra keluaran akan di*zoom* dengan skala 105 % sehingga citra akan bersih dari jejak hitam.

3.4 *Preprocessing* Citra

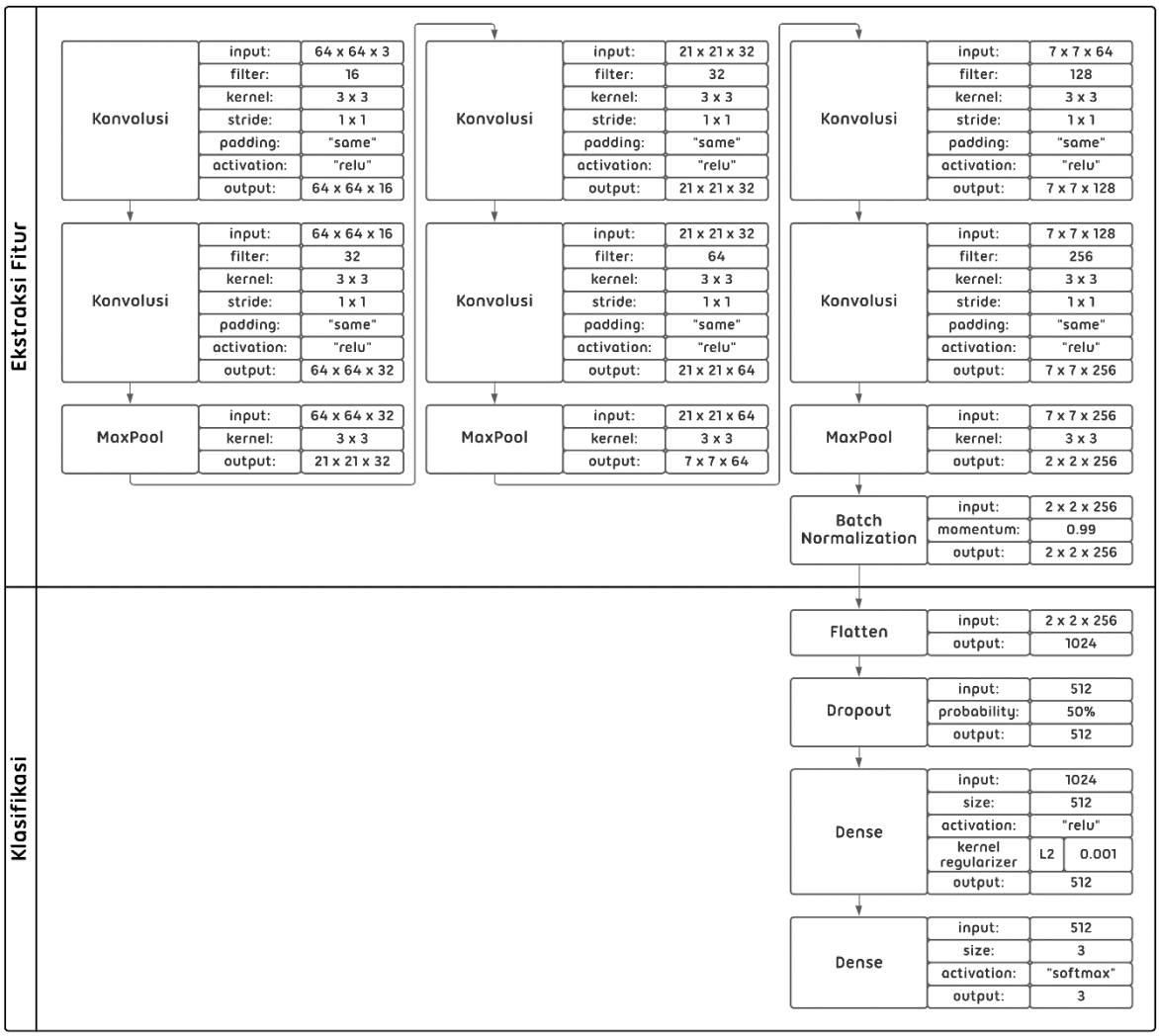
Preprocessing citra adalah tahap di mana data diolah lebih lanjut sehingga siap untuk dipakai dalam pengembangan model. Pada kasus ini, citra yang dihasilkan dari proses augmentasi berformat RGB yang di mana rentangnya 0 sampai 255 dan rentang tersebut akan di normalisasi. Normalisasi data secara umum adalah proses untuk mengubah skala sebuah data ke rentang 0 sampai 1.

3.5 Pemecahan *Dataset*

Seperti yang sudah dijelaskan pada proses augmentasi citra, masing-masing kelas alfabet menyumbang sebanyak 74880 citra yang berarti terkumpul total citra sebanyak 1946880 citra dengan resolusi 64 x 64 *pixel*. Selanjutnya, seluruh citra akan dipecah menjadi tiga bagian yaitu data *training*, data *validation*, dan data *testing* dengan rasio masing-masing secara berurutan sebesar 80% : 10% : 10%. Rasio tersebut dipilih berdasarkan salah satu dari tiga rekomendasi yang disarankan oleh Draelos (Draelos, 2019). Khusus untuk data *training* dan data *validation* akan masuk ke dalam pelatihan model sedangkan untuk data *testing* baru akan terpakai saat proses pengujian.

3.6 Pencarian Arsitektur

Pada penelitian ini, model akan dikembangkan menggunakan metode *Convolutional Neural Network* (CNN). Supaya lebih mudah, peneliti memutuskan untuk menggunakan arsitektur CNN yang sudah pernah digunakan pada penelitian sebelumnya. Arsitektur yang sudah pernah digunakan diyakini sudah melewati pemikiran yang panjang sehingga tidak perlu diragukan lagi performanya. Walaupun begitu, masih perlu dilakukan beberapa penyesuaian agar model dapat berfungsi dengan baik juga pada penelitian ini. Arsitektur CNN yang digunakan dalam penelitian ini didapatkan dari penelitian yang sudah ada sebelumnya milik Fadillah (Fadillah, 2020) dengan melakukan sedikit perubahan. Gambar 3.9 memperlihatkan arsitektur CNN beserta parameternya pada penelitian yang sudah ada. Parameter-parameter yang tidak diperlihatkan menggunakan nilai *default* menurut dokumentasinya masing-masing.



Gambar 3.9 Arsitektur CNN menurut penelitian sebelumnya

Arsitektur CNN yang dikembangkan pada penelitian sebelumnya menggunakan enam lapisan konvolusi, tiga lapisan *maxpool* dengan *kernel pooling* sebesar 3x3, dan fungsi aktivasi *sigmoid* untuk klasifikasi. Selanjutnya model di*compile* dengan menggunakan *Adam* model *optimizer* dan *loss function categorical crossentropy* untuk klasifikasi kemudian model dilatih dengan menggunakan lima *epoch*. Resolusi citra yang digunakan berukuran 64 x 64 *pixel*.

Arsitektur CNN yang didapatkan dari penelitian sebelumnya dirasa masih bisa di konfigurasi lagi supaya sesuai pada penelitian ini. Oleh karena itu, melakukan *tuning hyperparameter* dan mencari kombinasi yang tepat dapat dilakukan untuk mendapatkan performa terbaik. *Hyperparameter* yang akan dikonfigurasi pertama kali adalah jumlah *epoch* yang digunakan dalam pelatihan. Setelah didapatkan nilai *epoch* yang sesuai, berikutnya membandingkan performa pelatihan model dengan menggunakan model *optimizer* yang berbeda. Arsitektur CNN juga turut dikonfigurasi guna mencari mana yang terbaik. Arsitektur yang dikonfigurasi adalah jumlah lapisan konvolusi dan jumlah lapisan *maxpool* beserta ukuran *kernel pooling*nya juka ikut dibandingkan.

Berhubung hasil dari pelatihan tidak mungkin persis sama, maka seluruh konfigurasi yang dilakukan pada penelitian ini tidak bisa dijadikan sebagai patokan terhadap baik atau buruknya performa model. Semua hal itu tergantung kepada masalah yang ingin diselesaikan, *dataset* yang digunakan, dan lain sebagainya.

3.6.1 Jumlah *Epoch*

Terdapat tujuh kategori yang digunakan dalam membandingkan nilai *epoch*. Kategorinya adalah nilai *epoch* 5, nilai *epoch* 15, nilai *epoch* 25, nilai *epoch* 50, nilai *epoch* 75, nilai *epoch* 100, dan nilai *epoch* 250. Semua nilai tersebut akan dibandingkan dan dicari nilai *loss* yang terbaik atau yang paling kecil.

Dalam melakukan perbandingan jumlah *epoch*, arsitektur CNN yang digunakan adalah enam lapisan konvolusi dan tiga lapisan *maxpool* dengan *kernel pooling* sebesar 3x3. Lalu model di*compile* dengan menggunakan *Adam* model *optimizer*. Kurang lebih sama dengan arsitektur pada penelitian sebelumnya.

3.6.2 Penggunaan Model *Optimizer*

Terdapat banyak model *optimizer* yang bisa digunakan namun hanya dipilih enam model *optimizer* yang akan saling dibandingkan satu sama lain. Keenam model *optimizer* itu adalah *SGD* (Chollet & Others, 2015g), *RMSprop* (Chollet & Others, 2015f), *Adam* (Chollet & Others,

2015b), *Adagrad* (Chollet & Others, 2015a), *Adamax* (Chollet & Others, 2015c), dan *Nadam* (Chollet & Others, 2015e). Khusus untuk *SGD optimizer*, nilai *learning rate* secara sengaja untuk tidak disamakan dengan model *optimizer* lainnya melainkan menggunakan nilai *learning rate default* yang tertulis pada dokumentasinya. Peneliti menggunakan nilai *learning rate default* karena diasumsikan bahwa nilai *default* adalah nilai yang terstandar.

Dalam melakukan perbandingan model *optimizer*, arsitektur CNN yang digunakan adalah enam lapisan konvolusi dan tiga lapisan *maxpool* dengan *kernel pooling* sebesar 3x3. Lalu model dilatih dengan menggunakan nilai *epoch* yang didapatkan dari tahap perbandingan nilai *epoch*.

3.6.3 Jumlah Lapisan Pengekstraksi Fitur

Lapisan konvolusi dan lapisan *pooling* merupakan bagian terpenting dalam arsitektur *Convolutional Neural Network*. Lapisan konvolusi menjadi tempat di mana fitur-fitur citra akan diekstraksi lalu fitur-fiturnya akan diringkas di lapisan *pooling* sebelum masuk ke dalam tahap klasifikasi. Mengubah konfigurasi kedua lapisan ini berarti juga mengubah arsitektur dari *Convolutional Neural Network* itu sendiri.

Terdapat empat kategori yang digunakan dalam membandingkan lapisan konvolusi dan lapisan *pooling*. Kategorinya adalah menggunakan empat lapisan konvolusi dengan dua lapisan *maxpool* dengan *kernel pooling* 3x3, enam lapisan konvolusi dengan tiga lapisan *maxpool* dengan *kernel pooling* 3x3, enam lapisan konvolusi dengan tiga lapisan *maxpool* dengan *kernel pooling* 2x2, dan delapan lapisan konvolusi dengan empat lapisan *maxpool* dengan *kernel pooling* 2x2. Semua arsitektur tersebut akan dibandingkan dan dicari nilai *loss* yang terbaik atau yang paling kecil.

Dalam melakukan perbandingan jumlah lapisan pengekstraksi fitur, model di*compile* dengan menggunakan model *optimizer* yang didapatkan dari tahap perbandingan model *optimizer*. Lalu model dilatih dengan menggunakan nilai *epoch* yang didapatkan dari tahap perbandingan nilai *epoch*.

3.7 Pelatihan

Model dibangun dengan arsitektur yang sudah diperhitungkan berdasar pada perbandingan yang sudah dilakukan pada sub-bab sebelumnya. Arsitektur dibangun dengan menggunakan enam lapisan konvolusi dan tiga lapisan *maxpool* dengan *kernel pooling* 3x3, menggunakan *RMSprop* sebagai model *optimizer*nya, dan *epoch* pelatihan sebanyak 100.

Resolusi citra yang diterima oleh model telah ditentukan sebesar 64 x 64 x 3 yang artinya citra memiliki resolusi sebesar 64 x 64 dengan 3 *channel* yang merepresentasikan warna *red*, *green*, dan *blue* (RGB).

Model CNN yang sudah selesai dilatih akan dikonversi lalu disimpan ke dalam format tflite (TensorFlow Lite). Hal tersebut dilakukan supaya model bisa digunakan di perangkat *smartphone*.

3.8 Pengujian

Terdapat tiga jenis pengujian yang ada di penelitian ini yaitu pengujian dengan data foto, pengujian dengan data video, dan pengujian dengan data *realtime*. Pengujian dengan data foto dan data video dilakukan di *Google Colabolatory*, sedangkan pengujian *realtime* yang juga berbentuk video akan dilakukan menggunakan perangkat *smartphone Android*.

Pengujian dilakukan dengan menguji kemampuan model dalam mengenali bahasa isyarat dari huruf A sampai Z. Untuk pengujian foto akan diuji dengan memasukan foto dari citra huruf A sampai huruf Z. Untuk pengujian video akan berbentuk rekaman video peneliti melakukan gerakan bahasa isyarat dari huruf A sampai Z. Sedangkan untuk pengujian *realtime*, gerakan bahasa isyarat dari huruf A sampai Z akan diperagakan secara langsung di depan kamera.

Selain itu, terdapat lima jenis latar uji seperti latar tembok, latar kaos, dan latar kemeja bercorak. Mayoritas warna dari ketiga latar tersebut berwarna putih. Latar baju berwarna hitam dan jubah berwarna biru ditambahkan sebagai data uji dengan latar yang belum pernah dilihat atau dipelajari sebelumnya oleh model.

Pengujian dengan seseorang yang mampu memperagakan bahasa isyarat dengan ahli juga dicoba dalam penelitian ini. Peneliti memutuskan untuk menggunakan citra yang diperagakan oleh ahli dari internet. Oleh karena itu, citra yang didapatkan hanya bisa digunakan dalam pengujian foto dan pengujian video saja.

Seluruh hasil pengujian akan dicatat satu per satu secara manual untuk setiap alfabetnya pada masing-masing latar. Pengujian dilakukan sebanyak 10 kali pada masing-masing alfabetnya dengan target terdeteksi benar setidaknya sebanyak 5 dari 10 kali percobaan, apabila target berhasil tercapai akan dikategorikan sebagai berhasil ditebak dengan benar dan apabila tidak tercapai akan dikategorikan sebagai belum berhasil ditebak dengan benar. Sebuah pengujian akan dikatakan berhasil apabila telah mencapai akurasi kumulatifnya sebesar 46.8%, sesuai dengan akurasi rata-rata yang dihasilkan pada penelitian referensi (Fadillah, 2020). Pada

pengujuan video dan *realtime*, isyarat akan dicatat terdeteksi benar apabila berhasil terdeteksi dengan benar setidaknya selama 1 detik berturut-turut.

3.8.1 Jenis Pengujian

Pengujian foto dilakukan dengan mengunggah citra ke dalam program lalu akan diprediksi citra tersebut termasuk ke kelas yang mana. Benar atau salahnya prediksi kembali lagi kepada manusia atau pengguna yang menilai.

Pengujian video dilakukan dengan mengunggah video ke dalam program. Video tersebut akan dipecah menjadi *frame-frame* sehingga masing-masing dari *frame* tersebut bisa masuk ke dalam tahap prediksi oleh model. Hasil prediksinya akan ditempelkan langsung pada *frame* dalam bentuk teks lalu seluruh *frame* akan disatukan kembali menjadi sebuah video. Benar atau salahnya prediksi kembali lagi kepada manusia atau pengguna yang menilai.

Pengujian *realtime* dilakukan dengan menggunakan aplikasi *mobile* TensorFlow Lite. Cara melakukan pengujian cukup dengan berdiri di depan kamera sambil menggerakan bahasa isyarat dan hasil prediksinya akan terlihat langsung secara *realtime*. Benar atau salahnya prediksi kembali lagi kepada manusia atau pengguna yang menilai.

Perlu diperhatikan bahwa dalam melakukan pengujian *realtime*, kamera yang digunakan memiliki efek *mirror* sehingga seolah-olah tampilannya terbalik secara horizontal. Walaupun begitu, pengujian tetap dilakukan dengan melakukan gerakan bahasa isyarat seperti seharusnya. Efek *mirror* hanya terjadi pada pengujian *realtime* saja, sedangkan pengujian foto dan pengujian video tampilannya masih normal seperti seharusnya.

3.8.2 Latar *Dataset*

Terdapat total lima latar yang digunakan dalam tahap pengujian. Latar yang digunakan adalah latar yang sama dengan yang digunakan pada tahap pelatihan, yaitu latar kaos berwarna putih, latar tembok putih, dan latar kemeja putih bercorak. Dua latar ditambahkan pada proses pengujian yaitu Latar baju berwarna hitam dan latar jubah berwarna biru. Kelima latar tersebut digunakan dalam pengujian foto, pengujian video, dan pengujian *realtime*.

Latar baju berwarna hitam dan latar jubah berwarna biru digunakan sebagai latar baru atau latar yang belum pernah dipelajari oleh model. Model diuji kemampuannya untuk mengenali citra yang belum pernah dipelajari sebelumnya. Latar baju berwarna hitam dipilih karena warna hitam kontras dengan warna putih, sedangkan latar jubah berwarna biru dipilih karena warna biru masih memiliki unsur warna putih atau cerah.

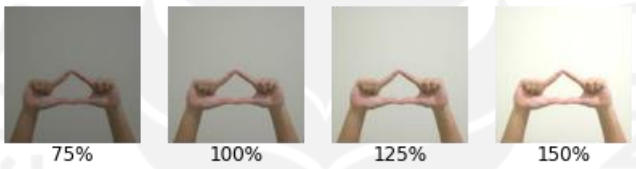
BAB IV
HASIL DAN PEMBAHASAN

4.1 Augmentasi Citra

Ada empat jenis augmentasi yang dilakukan yaitu mengatur *brightness*, translasi citra, *zoom*, dan rotasi citra. Seluruh citra yang dihasilkan dalam proses augmentasi ini akan disimpan dengan resolusi 64 x 64 *pixel* dengan format *red*, *green*, dan *blue* (RGB). Berikut hasil dari masing-masing tahap augmentasi.

4.1.1 Pengaturan *Brightness*

Tahap augmentasi pengaturan *brightness* menghasilkan 4 citra dari satu citra masukan. Gambar 4.1 memperlihatkan keluaran dari proses augmentasi tahap mengatur brightness.



Gambar 4.1 Keluaran proses augmentasi tahap mengatur *brightness*

4.1.2 Translasi Citra

Tahap augmentasi translasi citra menghasilkan 25 citra dari satu citra masukan. Gambar 4.2 memperlihatkan keluaran dari proses augmentasi tahap translasi citra. Citra tanpa jejak hitam yang akan disimpan pada tahap augmentasi ini.

Gambar 4.2 Keluaran proses augmentasi tahap translasi citra (a) dengan jejak hitam (b) tanpa jejak hitam

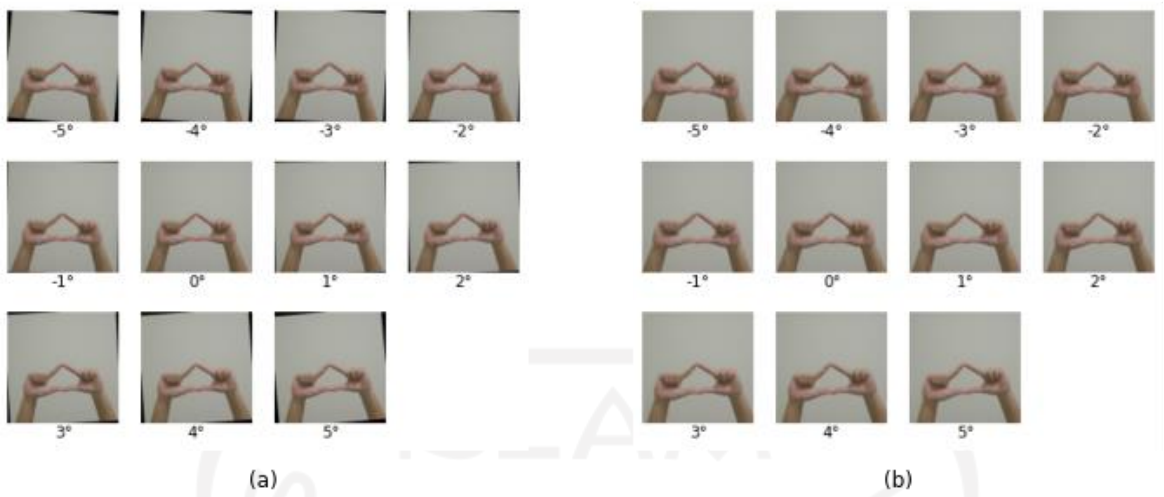
4.1.3 *Zoom*

Tahap augmentasi *zoom* menghasilkan 3 citra dari satu citra masukan. Gambar 4.3 memperlihatkan keluaran dari proses augmentasi tahap *zoom*.

Gambar 4.3 Keluaran proses augmentasi tahap *zoom*

4.1.4 Rotasi Citra

Tahap augmentasi rotasi citra menghasilkan 11 citra dari satu citra masukan. Gambar 4.4 memperlihatkan keluaran dari proses augmentasi tahap rotasi citra. Citra tanpa jejak hitam yang akan disimpan pada tahap augmentasi ini.



Gambar 4.4 Keluaran proses augmentasi tahap rotasi citra (a) dengan jejak hitam (b) tanpa jejak hitam

4.2 Pemecahan *Dataset*

Dataset akan dipecah menjadi tiga bagian yaitu data *training*, data *validation*, dan data *testing* dengan rasio masing-masing secara berurutan sebesar 80% : 10% : 10%. Teknik *split* data yang digunakan pada penelitian ini adalah dengan menggunakan *library splitfolders*. *Library splitfolders* akan memecah *folder dataset* hasil augmentasi dengan nama *folder* Data pada Gambar 4.5 menjadi tiga folder yang berbeda dalam direktori baru yang diberi nama NewData sehingga memudahkan dalam proses pelatihan dan pengujian. Data *training* disimpan dalan folder *train*, data *validation* disimpan dalam folder *val*, dan data *testing* disimpan dalam folder *test*. Data *training* berjumlah 1557504 citra, data *validation* berjumlah 194688 citra, dan data *testing* berjumlah 194688 citra.

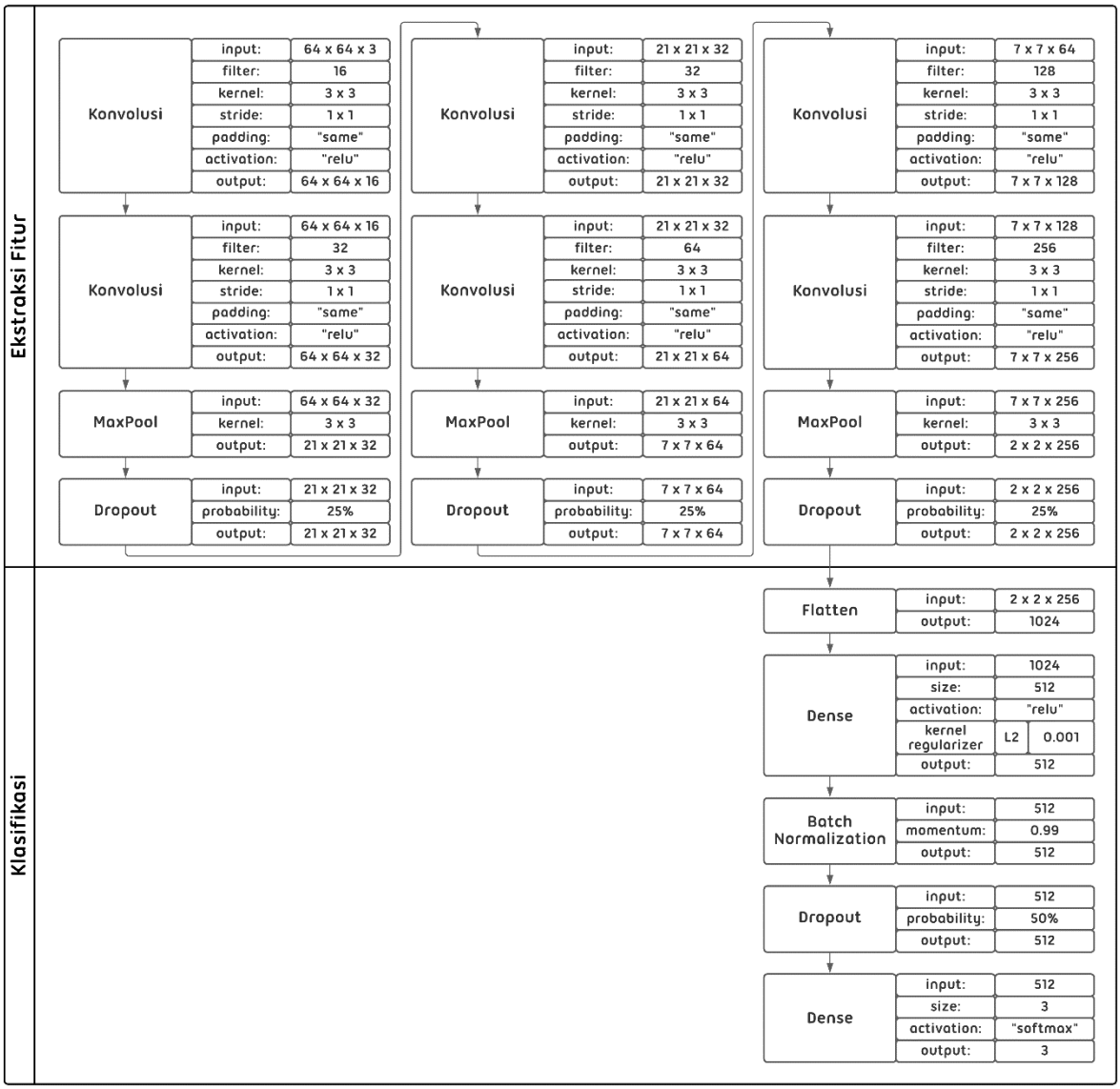
```
import splitfolders

splitfolders.ratio("Data", output="NewData", seed=42, ratio=(0.8, 0.1, 0.1))

train_dir      = "./NewData/train"
validation_dir = "./NewData/val"
test_dir       = "./NewData/test"
```

Gambar 4.5 Kode dari fungsi *splitfolders* untuk pemecahan dataset

4.3 Pencarian Arsitektur

Arsitektur CNN yang didapatkan dari penelitian sebelumnya perlu diubah sedikit agar dapat bekerja dengan baik dalam penelitian ini. Perubahan yang dimaksud seperti menambahkan lapisan *dropout* ke dalam arsitektur CNN dan memindahkan *batch*

normalization ke bagian klasifikasi. Menurut salah satu contoh dari dokumentasi Keras (Chollet & others, 2015), diperlihatkan bahwa lapisan *dropout* diletakan setelah lapisan *maxpool* dengan probabilitas 25% dan setelah lapisan *dense* pertama dengan probabilitas 50%. Selanjutnya *batch normalization* yang sebelumnya terletak setelah *maxpool* ketiga akan dipindahkan ke setelah lapisan *dense* yang pertama dengan fungsi aktivasi *ReLU* karena menurut Mishkin (Mishkin, Sergievskiy, & Matas, 2017), *batch normalization* lebih efektif apabila diletakan tepat setelah fungsi aktivasi *ReLU*. Gambar 4.6 memperlihatkan arsitektur CNN yang digunakan pada penelitian ini beserta nilai parameternya, parameter-parameter yang tidak ditampilkan menggunakan nilai *default* berdasarkan dokumentasinya masing-masing.



Gambar 4.6 Arsitektur CNN setelah konfigurasi

Berdasarkan konfigurasi yang dilakukan, dapat diambil keputusan bahwa arsitektur CNN yang digunakan memiliki enam lapisan konvolusi, tiga lapisan *maxpool* dengan kernel *pooling* 3x3. Model akan dilatih dengan menggunakan 100 epoch. Lalu model akan dicompile dengan menggunakan *RMSprop* model *optimizer* dan dan *loss function categorical crossentropy* untuk klasifikasi. Terakhir, c*allback* histori juga ditambahkan guna memudahkan melihat performa pelatihan yang dihasilkan oleh model. Hasil dari perubahan nilai *epoch*, penggunaan model *optimizer*, dan konfigurasi arsitektur CNN akan dijelaskan secara masing-masing.

4.3.1 Jumlah *Epoch*

Jumlah *epoch* 100 dipilih dalam penelitian ini karena sudah terjadi kenaikan *loss* dan penurunan akurasi pada epoch ke 150, bisa dilihat pada Tabel 4.1. Selain hasilnya yang bagus, durasi pelatihan yang tidak terlalu lama juga menjadi pertimbangan dalam memilih jumlah *epoch*.

Tabel 4.1 Perbandingan performa model dengan jumlah *epoch*

Jumlah *Epochs*	Durasi Pelatihan	*Loss*	Akurasi
5 epoch	70 detik	2.47878	36.750%
10 epoch	144 detik	1.14834	73.750%
25 epoch	358 detik	0.41584	94.375%
50 epoch	715 detik	0.21923	96.438%
75 epoch	1052 detik	0.16874	97.875%
100 epoch	1423 detik	0.11889	98.875%
150 epoch	2224 detik	0.13797	98.438%

4.3.2 Penggunaan Model *Optimizer*

Pada Tabel 4.2 terlihat bahwa *RMSprop* memiliki nilai *loss* yang paling kecil dan akurasi yang paling tinggi atau terbaik di antara model *optimizer* lainnya. Ada juga parameter-parameter lain dari masing-masing model *optimizer* yang bisa disesuaikan guna mendapatkan performa yang maksimal.

Tabel 4.2 Perbandingan performa model dengan model *optimizer* yang berbeda

Nama *Optimizer*	*Learning Rate*	*Loss*	Akurasi
Adagrad	0.001	2.17444	52.750%
Adam	0.001	0.12300	98.813%
Adamax	0.001	0.28622	98.563%
Nadam	0.001	0.13835	98.688%
RMSprop	0.001	0.10289	98.875%
SGD	0.01	0.98349	87.813%

4.3.3 Jumlah Lapisan Pengekstraksi Fitur

Dari Tabel 4.3 dapat dilihat bahwa dengan menggunakan lapisan konvolusi dan lapisan *pooling* yang banyak bukan berarti hasilnya akan selalu bagus, begitupun juga saat menggunakan jumlah lapisan yang sedikit. Selain itu, memperbanyak *trainable params* dengan cara mengurangi dimensi kernel *pooling* juga tidak membuat model menjadi lebih teliti dalam mempelajari citra. Dapat disimpulkan bahwa dalam mencari arsitektur dengan performa yang baik perlu melakukan *trial and error*.

Tabel 4.3 Perbandingan performa model CNN dengan berbagai arsitektur

Jumlah Lapisan *Conv2D*	Jumlah Lapisan *MaxPool*	*Trainable Params*	Durasi Pelatihan	*Loss*	Akurasi
4	2 (Kernel 3x3)	1,653,338	1214 detik	0.25030	97.313%
6	3 (Kernel 3x3)	941,018	1571 detik	0.08760	98.938%
6	3 (Kernel 2x2)	8,805,338	2976 detik	0.21407	97.750%
8	4 (Kernel 2x2)	6,381,274	4209 detik	0.18879	97.875%

Dari keempat opsi yang dubandingkan, terpilih arsitektur dengan enam lapisan konvolusi dan tiga lapisan *max pooling* dengan ukuran *kernel pooling* 3x3. Selain karena durasi pelatihannya yang bisa dibilang relatif cepat, pemilihan arsitektur tersebut juga didasari oleh nilai *loss* yang paling kecil dan nilai akurasi yang paling tinggi dibandingkan dengan ketiga opsi lainnya.

4.4 Pelatihan

Berikut akan dijelaskan secara singkat alur bagaimana sebuah citra dipelajari oleh model. Pertama, citra akan masuk ke dalam lapisan *input* dengan ukuran 64 x 64 x 3 dan akan dibawa ke konvolusi 1 dengan *filter* sebanyak 16, dan menggunakan fungsi aktivasi *ReLU*. Lalu citra

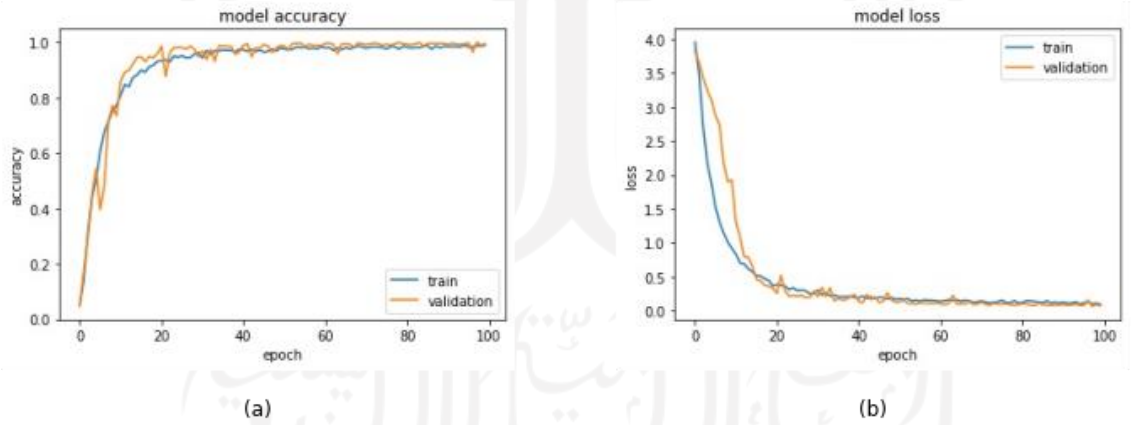
masuk lagi ke konvolusi 2 dengan *filter* sebanyak 32, dan menggunakan fungsi aktivasi *ReLU*. Setelah melewati 2 konvolusi, citra akan dibawa ke proses *maxpooling* dengan ukuran *kernel pooling* 3x3 sebelum akhirnya masuk ke lapisan *dropout* dengan persentase *dropout* sebesar 25%. Berlanjut ke konvolusi 3 dengan *filter* sebanyak 32, dan menggunakan fungsi aktivasi *ReLU*. Lalu citra masuk lagi ke konvolusi 4 dengan *filter* sebanyak 64, dan menggunakan fungsi aktivasi *ReLU*. Setelah melewati total 4 konvolusi, citra akan dibawa ke proses *maxpooling* dengan ukuran *kernel pooling* 3x3 sebelum akhirnya masuk ke lapisan *dropout* dengan persentase *dropout* sebesar 25%. Berlanjut ke konvolusi 5 dengan *filter* sebanyak 128, dan menggunakan fungsi aktivasi *ReLU*. Lalu citra masuk lagi ke konvolusi 6 dengan *filter* sebanyak 256, dan menggunakan fungsi aktivasi *ReLU*. Setelah melewati total 6 konvolusi, citra akan dibawa ke proses *maxpooling* dengan ukuran *kernel pooling* 3x3 sebelum akhirnya masuk ke lapisan *dropout* dengan persentase *dropout* sebesar 25%. Selanjutnya citra masuk ke lapisan dense 1 dengan *filter* sebanyak 512, dan menggunakan fungsi aktivasi *ReLU*. Lalu citra dinormalisasi dengan *batch normalization* dengan parameter momentum sebesar 0.99 sebelum masuk ke lapisan *dropout* dengan persentase *dropout* sebesar 50%. Terakhir, citra diklasifikasi pada lapisan dense 2 dengan *filter* sebanyak 26 sesuai dengan jumlah kelas dari *dataset* dan menggunakan fungsi aktivasi *sigmoid*. Langkah tersebut diulang terus menerus sampai citra dan *epoch*nya habis.

Pada kelas *ImageDataGenerator* terdapat tiga metode yang bisa digunakan, pada penelitian ini menggunakan *flow_from_directory*. Metode ini berguna ketika citra ditempatkan di *folder* alfabet masing-masing. Metode ini akan mengidentifikasi alfabet sebagai kelas secara otomatis berdasarkan nama *folder*nya. Tidak hanya itu, *ImageDataGenerator* akan dipanggil di tahap selanjutnya yaitu *train generator* dan *validation generator*. Dalam *train generator*, citra yang ada dalam data *training* akan diambil secara acak berdasarkan ukuran batchnya. Ukuran batch yang digunakan adalah 64 sama seperti yang digunakan pada penelitian terdahulu milik Fadillah (Fadillah, 2020), yang berarti saat proses pelatihan data yang diambil secara acak sejumlah 64 citra dari seluruh direktori data *train*. Akan dilakukan hal yang sama untuk *epoch* berikutnya hingga mencapai *epoch* yang terakhir. Begitu pula yang terjadi pada *validation generator*.

Langkah selanjutnya adalah menetapkan parameter-parameter yang dibutuhkan untuk pelatihan model CNN. Input citra pada model ini adalah 64 x 64, pastikan ukuran citra yang akan dilatih memiliki ukuran yang sama dengan parameter input yang ditentukan. *Epoch* yang digunakan dalam pelatihan sejumlah 100 *epoch*. Berhubung jumlah *dataset* yang dipakai tidak

bisa dihitung karena terus di*generate* secara acak menggunakan *ImageDataGenerator*, alangkah baiknya jumlah citra yang dilatih dibatasi setiap *epoch*nya dengan menggunakan *steps_per_epoch*, jumlah step yang digunakan sebanyak 25 langkah setiap iterasi. Opsi lainnya adalah *batch_size*, bisa digunakan tetapi hanya cocok untuk *dataset* yang jumlahnya spesifik diketahui. Apabila data *training* menggunakan *steps_per_epoch*, maka untuk data *validation* bisa menggunakan *validation_steps* dengan fungsi yang sama. Jumlah *validation_steps* disini sebanyak 10 langkah. Nilai *steps_per_epoch* dan *validation_steps* dapat di*tuning* atau disesuaikan dengan keinginan masing-masing peneliti, tetapi dalam penelitian ini nilai tersebut dirasa sudah cocok tidak terlalu besar dan tidak terlalu kecil.

Setelah pelatihan selesai, sekarang waktunya untuk melihat performa dari model. Keras menyediakan opsi untuk mencatat *callback* saat melatih model. Salah satu *callback* yang dicatat saat melatih model adalah *callback* Histori. Fungsinya mencatat metrik pelatihan untuk setiap *epoch*. Metrik yang dicatat berupa *training loss* dan *training* akurasi serta *validation loss* dan *validation* akurasi apabila ada. Metrik-metrik histori didapatkan dari fungsi *fit* yang digunakan untuk melatih model. Metrik dikembalikan dan disimpan dalam histori selama proses pelatihan berlangsung. Maka dari itu, metrik-metrik yang sudah terkumpul dapat ditampilkan ke dalam grafik seperti Gambar 4.7.



Gambar 4.7 Grafik (a) akurasi pelatihan data *training* dan data *validation* (b) *loss* pelatihan data *training* dan data *validation*

Dari Gambar 4.7 bisa dilihat bahwa model sudah mulai mendekati akurasi terbaiknya disekitar *epoch* ke-70 dan terus stabil sampai proses pelatihan selesai. Hasil akhir dari pelatihan yaitu *training* akurasi sebesar 98.94% dan *validation* akurasi sebesar 99.38% dengan durasi pelatihan yang dihabiskan selama 1766 detik. Berdasarkan nilai akurasi *training* dan akurasi *validation*, dapat disimpulkan bahwa model termasuk ke dalam model yang *good fit* karena

menghasilkan nilai *loss* dan akurasi yang bagus pada data *training* dan juga menghasilkan nilai *loss* dan akurasi yang bagus juga pada data *validation*.

Model CNN yang sudah selesai dilatih selanjutnya akan dikonversi supaya bisa digunakan di perangkat *smartphone*. Menggunakan fungsi *TFLiteConverter* dengan metode *from_keras_model*. Akhirnya, model berubah ke dalam format tflite (TensorFlow Lite) dengan menggunakan nilai float 32-bit untuk semua data parameternya.

4.5 Evaluasi Model

Untuk mengukur performa sebuah model dalam kemampuan klasifikasi dapat dilihat dari parameter pengukuran performanya seperti akurasi (2.1), *recall* (2.2), presisi (2.3), dan skor f1 (2.4). Sebelumnya sudah ditulis rumus yang bisa digunakan untuk menghitung nilai-nilai tersebut. Untuk mempersingkat waktu dan mempermudah pengukuran, bisa menggunakan fungsi-fungsi pengukuran yang sudah disediakan oleh *library sklearn*. Berikut hasil dari masing-masing pengukuran.

4.5.1 Skor Loss dan Akurasi

Cara termudah untuk menghitung nilai akurasi dan *loss* yaitu dengan menggunakan fungsi *evaluate* dari *library* Keras. Fungsi *evaluate* berfungsi untuk mengevaluasi dua variabel masukannya dengan *loss* dan akurasi sebagai keluarannya.

Variabel *test_images* berupa larik dari seluruh citra data *testing* dan variabel *test_labels* adalah larik dari label masing-masing citra tersebut. Masing-masing data di larik citra akan ditebak oleh model lalu hasilnya akan dibandingkan dengan kelas aslinya yang berada di larik label. Gambar 4.8 memperlihatkan nilai keluaran *loss* dan akurasi dari fungsi *evaluate* menggunakan data *testing*.

```
evaluate_test = model.evaluate(test_images, test_labels, verbose=1)

print("\nAccuracy  =", "{:.5f}%".format(evaluate_test[1]*100))
print("Loss      =", "{:.7f}".format(evaluate_test[0]))
```

```
6084/6084 [==============================] - 377s 62ms/step - loss: 0.0654 - accuracy: 0.9986

Accuracy  = 99.86491%
Loss      = 0.0653786
```

Gambar 4.8 Kode dan keluaran dari fungsi *evaluate* dengan data *testing*

4.5.2 Skor Presisi

Setelah melakukan prediksi pada data *testing*, skor presisi dihitung untuk setiap kelas berdasarkan hasil prediksi dari model. Skor presisi dihitung dengan menggunakan fungsi *precision_score* dari *library sklearn*. Keluaran menunjukan nilai skor presisi yang dihasilkan menggunakan data *testing* sebesar 99.86563%. Gambar 4.9 memperlihatkan cara menggunakan fungsi *precision_score.*

```
from sklearn.metrics import precision_score

precision = precision_score(y_test, yhat_classes, average='macro')
print("Precision =", "{:.5f}%".format(precision*100))
```

Gambar 4.9 Kode dari fungsi *precision_score* dengan data *testing*

4.5.3 Skor *Recall*

Setelah melakukan prediksi pada data *testing*, skor *recall* dihitung untuk setiap kelas berdasarkan hasil prediksi dari model. Skor *recall* dihitung dengan menggunakan fungsi *recall_score* dari *library sklearn*. Keluaran menunjukan nilai skor *recall* yang dihasilkan menggunakan data *testing* sebesar 99.86491%. Gambar 4.10 memperlihatkan cara menggunakan fungsi *recall_score.*

```
from sklearn.metrics import recall_score

recall = recall_score(y_test, yhat_classes, average='macro')
print("Recall    =", "{:.5f}%".format(recall*100))
```

Gambar 4.10 Kode dari fungsi *recall_score* dengan data *testing*

4.5.4 Skor F1

Setelah melakukan prediksi pada data *testing*, skor F1 dihitung untuk setiap kelas berdasarkan hasil prediksi dari model. Skor F1 dihitung dengan menggunakan fungsi *f1_score* dari *library sklearn*. Keluaran menunjukan nilai skor recall yang dihasilkan menggunakan data *testing* sebesar 99.86491%. Gambar 4.11 memperlihatkan cara menggunakan fungsi *f1_score.*

```
from sklearn.metrics import f1_score

f1 = f1_score(y_test, yhat_classes, average='macro')
print("F1 Score  =", "{:.5f}%".format(f1*100))
```

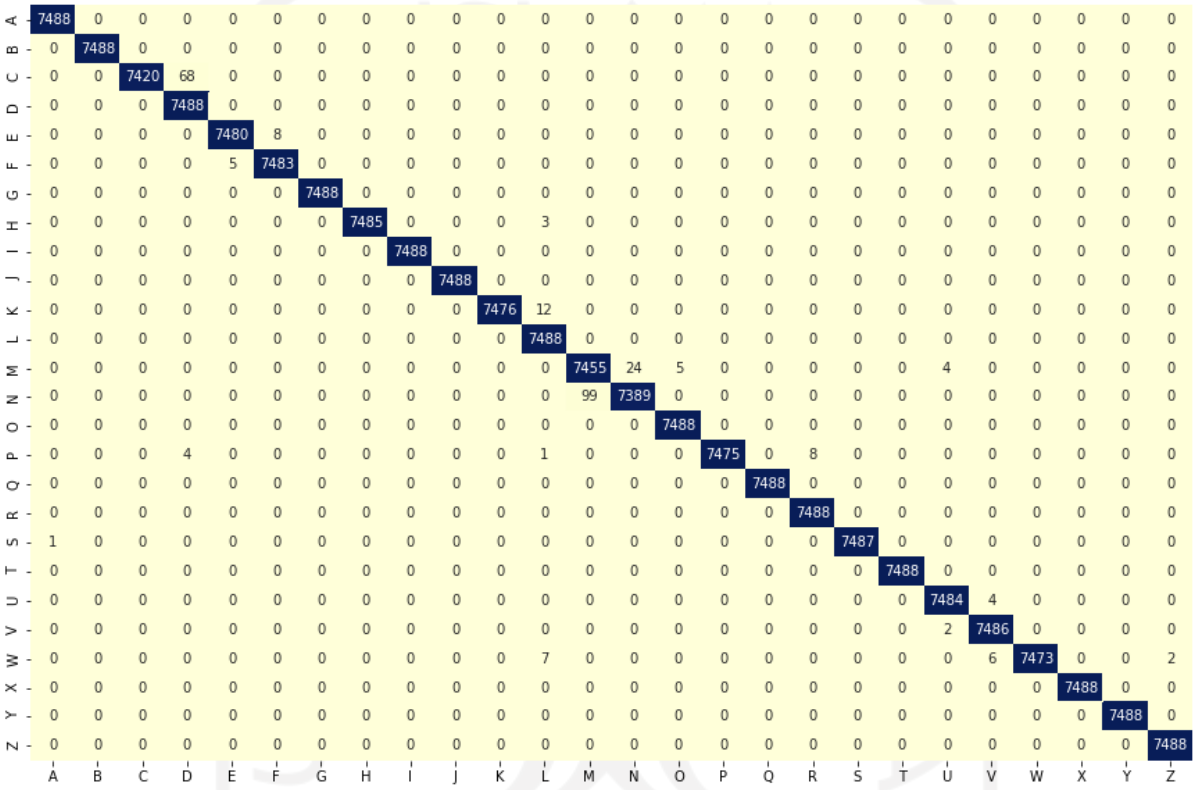
Gambar 4.11 Kode dari fungsi *f1_score* dengan data *testing*

4.5.5 Skor *Confusion Matrix*

Proses pengujian menggunakan data *testing* dengan masing-masing kelas sebanyak 7488 yang berarti total data *testing* sebanyak 194688. *Confusion* matriks dipanggil dengan

menggunakan fungsi *confusion_matrix* dari *library sklearn*. Hasil pengujian ditampilkan dengan *confusion* matriks seperti Gambar 4.12.

```
import seaborn as sn
import pandas as pd
from sklearn.metrics import confusion_matrix

conf_matrix = confusion_matrix(y_test, yhat_classes)
df_cm = pd.DataFrame(conf_matrix,
                     index = [chr(alphabet) for alphabet in range(65,91)],
                     columns = [chr(alphabet) for alphabet in range(65,91)])
sn.heatmap(df_cm, cmap="YlGnBu", fmt="d", cbar=False, annot=True)
```

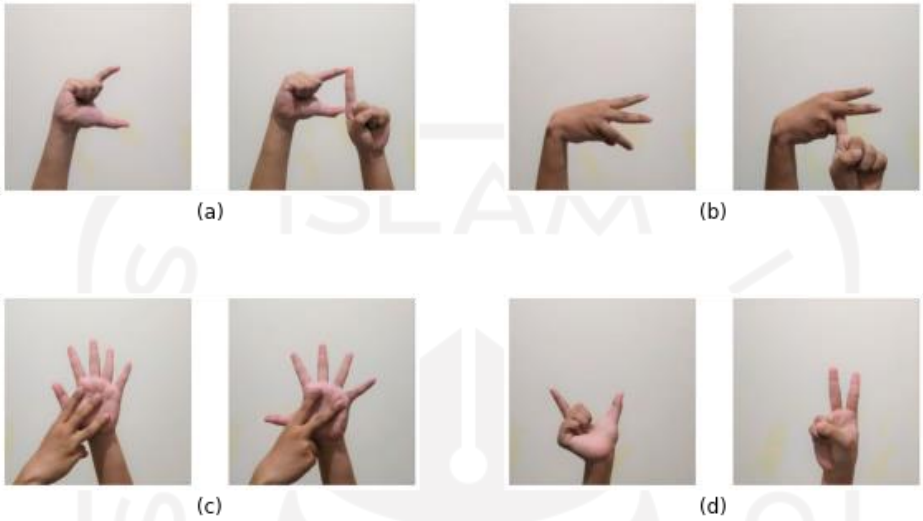


Gambar 4.12 Kode penggunaan fungsi *confusion_matrix* beserta keluarannya

Berdasarkan confusion matrix, huruf N menjadi kelas yang paling dideteksi dengan benar di antara kelas yang lainnya dengan hanya berhasil menebak 7389 citra dengan benar dari total 7488 citra.

Dapat dilihat juga terdapat 68 data *testing* huruf C yang teridentifikasi sebagai huruf D. Ada juga 8 data *testing* huruf E yang teridentifikasi huruf F dan 5 data *testing* huruf F yang teridentifikasi huruf E. Ada juga 24 data *testing* huruf M yang teridentifikasi huruf N dan 99 data *testing* huruf N yang teridentifikasi huruf M. Ada juga 4 data *testing* huruf U yang teridentifikasi huruf V dan 2 data *testing* huruf V yang teridentifikasi huruf U.

Hal tersebut dirasa wajar karena memang bentuk gestur dari huruf C dan D sekilas terlihat mirip, huruf E dan F sekilas terlihat mirip, huruf M dan N sekilas terlihat mirip, dan huruf U dan V sekilas terlihat mirip. Kemiripan antara huruf-huruf tersebut bisa dilihat pada Gambar 4.13.



Gambar 4.13 Sampel isyarat yang sekilas terlihat mirip (a) huruf C dengan huruf D (b) huruf E dengan huruf F (c) huruf M dengan huruf N (d) huruf U dengan huruf V

4.6 Pengujian

Terdapat tiga jenis pengujian yang ada di penelitian ini yaitu pengujian dengan data foto, pengujian dengan data video, dan pengujian dengan data *realtime* menggunakan perangkat *smartphone*. Pengujian dengan data foto dan data video dilakukan di *Google Colabolatory*, sedangkan pengujian *realtime* yang juga berbentuk video akan dilakukan menggunakan perangkat *smartphone* dengan sistem operasi *Android*.

Adapun langkah-langkah yang terjadi selama proses pengujian. Walaupun ada berbagai jenis pengujian, intinya tetap sama yaitu menguji citra. Citra yang masuk perlu di*resize* supaya resolusinya menjadi 64 x 64. Ukuran 64 x 64 x 3 merupakan ukuran yang bisa diterima oleh model yang dihasilkan dari proses pelatihan. Setelah di*resize*, citra akan diubah menjadi larik lalu dimasukkan ke dalam fungsi *predict*. Fungsi *predict* dari *library* Keras berfungsi untuk menebak atau memprediksi kelas dari citra masukan berdasarkan pengetahuan yang dimiliki oleh model selama proses pembelajarannya.

Berikut akan dijelaskan secara singkat alur bagaimana sebuah citra diklasifikasi oleh model. Pertama, citra akan masuk ke dalam lapisan *input* dengan ukuran 64 x 64 x 3 dan akan dibawa ke konvolusi 1 dengan *filter* sebanyak 16, dan menggunakan fungsi aktivasi *ReLU*.

Lalu citra masuk lagi ke konvolusi 2 dengan *filter* sebanyak 32, dan menggunakan fungsi aktivasi *ReLU*. Setelah melewati 2 konvolusi, citra akan dibawa ke proses *maxpooling* dengan ukuran *kernel pooling* 3x3 sebelum akhirnya masuk ke lapisan *dropout* dengan persentase *dropout* sebesar 25%. Berlanjut ke konvolusi 3 dengan *filter* sebanyak 32, dan menggunakan fungsi aktivasi *ReLU*. Lalu citra masuk lagi ke konvolusi 4 dengan *filter* sebanyak 64, dan menggunakan fungsi aktivasi *ReLU*. Setelah melewati total 4 konvolusi, citra akan dibawa ke proses *maxpooling* dengan ukuran *kernel pooling* 3x3 sebelum akhirnya masuk ke lapisan *dropout* dengan persentase *dropout* sebesar 25%. Berlanjut ke konvolusi 5 dengan *filter* sebanyak 128, dan menggunakan fungsi aktivasi *ReLU*. Lalu citra masuk lagi ke konvolusi 6 dengan *filter* sebanyak 256, dan menggunakan fungsi aktivasi *ReLU*. Setelah melewati total 6 konvolusi, citra akan dibawa ke proses *maxpooling* dengan ukuran *kernel pooling* 3x3 sebelum akhirnya masuk ke lapisan *dropout* dengan persentase *dropout* sebesar 25%. Selanjutnya citra masuk ke lapisan dense 1 dengan *filter* sebanyak 512, dan menggunakan fungsi aktivasi *ReLU*. Tahap *batch normalization* tidak dilibatkan seperti pada tahap pengujian karena menurut dari dokumentasinya (Chollet & Others, 2015d), lapisan *batch normalization* diatur supaya tidak aktif saat melakukan pengujian. Dari lapisan dense, citra masuk ke lapisan *dropout* dengan persentase *dropout* sebesar 50%. Terakhir, citra diklasifikasi pada lapisan dense 2 dengan *filter* sebanyak 26 sesuai dengan jumlah kelas dari *dataset* dan menggunakan fungsi aktivasi *sigmoid*. Disinilah hasil bisa terlihat, dicari nilai probabilitas yang paling besar diantara 26 kelas dan kelas tersebutlah yang menjadi hasil akhir dari klasifikasi.

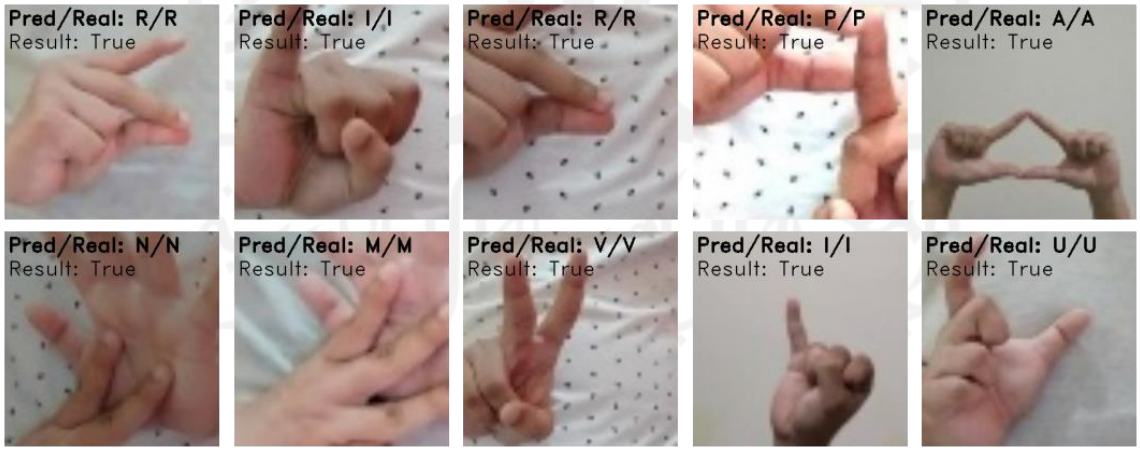
4.6.1 Pengujian Data Foto

Hal pertama yang disiapkan adalah citra yang akan diuji. Citra bisa berukuran apa saja karena di dalam algoritme pengujian terdapat fungsi *resize* yang di mana mengubah resolusi citra masukan menjadi 64 x 64, sesuai dengan ukuran citra yang bisa diterima oleh model. Setelah di*resize*, citra akan diubah menjadi larik lalu dimasukkan ke dalam fungsi *predict* yang di mana akan mengembalikan nilai prediksi. Manusialah yang menilai apakah model berhasil memprediksi citra atau tidak. Bisa dilihat pada Gambar 4.14 bahwa model berhasil memprediksi citra masukan dengan benar.

Pada pengujian foto, terdapat dua kemungkinan yang bisa dilakukan dalam menguji. Yang pertama dengan mengunggah foto ke dalam program lalu foto tersebut masuk ke dalam proses pengujian. Opsi kedua yaitu menggunakan data *testing* yang sudah disimpan semenjak tahap pemecahan *dataset*.

Gambar 4.14 Pengujian citra dengan citra yang diunggah (a) huruf A terdeteksi A (b) huruf B terdeteksi B (c) huruf C terdeteksi C

Pada opsi kedua yaitu menguji dengan menggunakan data *testing*, proses yang dilakukan adalah dengan memuat seluruh data *testing* yang ada di folder *test* ke dalam sebuah larik. Begitupun juga dengan label dari masing-masing citra, disimpan ke dalam sebuah larik. Cara pengujian yang dilakukan peneliti adalah dengan memilih sepuluh *index* secara acak yang ada di larik citra lalu satu per satu dimasukan ke dalam fungsi *predict*. Hasil dari masing-masing prediksi akan dibandingkan dengan label dari citra. Hasilnya akan berbentuk *True* apabila prediksi dan labelnya sesuai dan *False* apabila sebaliknya. Tangkap layar pengujian citra dengan data *testing* bisa dilihat pada Gambar 4.15 yang kebetulan hasil prediksinya benar semua.



Gambar 4.15 Pengujian citra dengan data *testing*

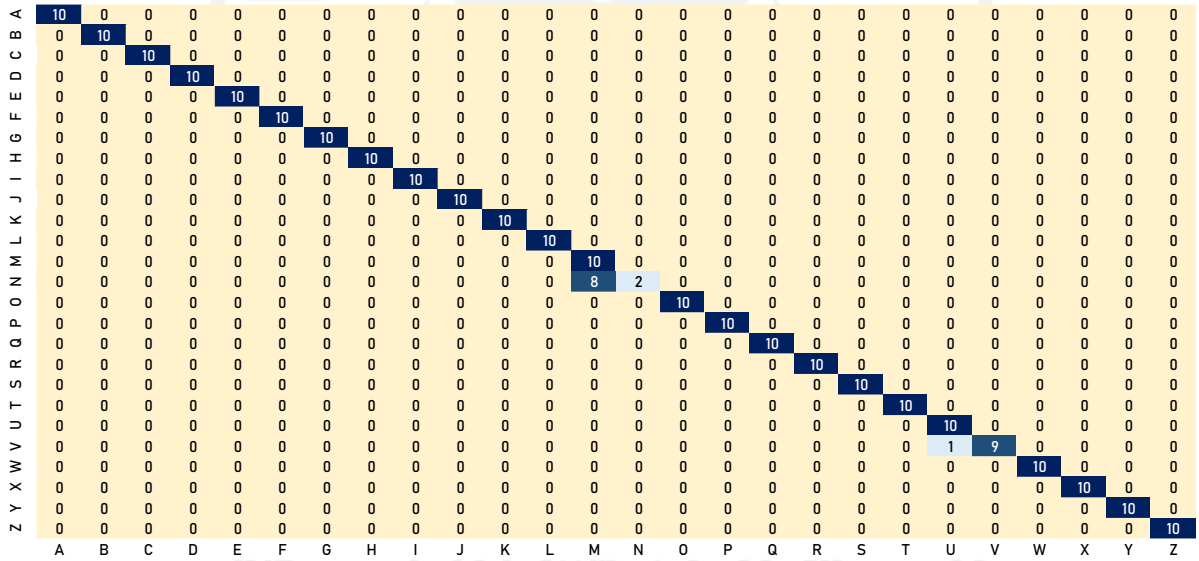
Pengujian foto dengan cara mengunggah citra telah dilakukan dan sudah dicatat hasilnya satu per satu. Hasil pengujian menunjukan performa yang cukup bagus, walaupun masih ada

beberapa huruf termasuk ke dalam kategori belum berhasil ditebak dengan benar. Supaya lebih jelas, berikut penjelasan hasil pengujian pada masing-masing latar.

Pengujian Foto dengan Latar Kaos

Pengujian pada kategori ini menggunakan kaos berwarna putih sebagai latarnya. Contoh latar kaos putih bisa dilihat pada Gambar 3.2 pada huruf A dan B. Sebelum menguji, perlu disiapkan dahulu citra alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.14. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.16, dari 26 alfabet hanya ada 1 huruf yang belum berhasil ditebak dengan benar yaitu huruf N. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 25 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 96.54% pada pengujian foto dengan latar kaos.



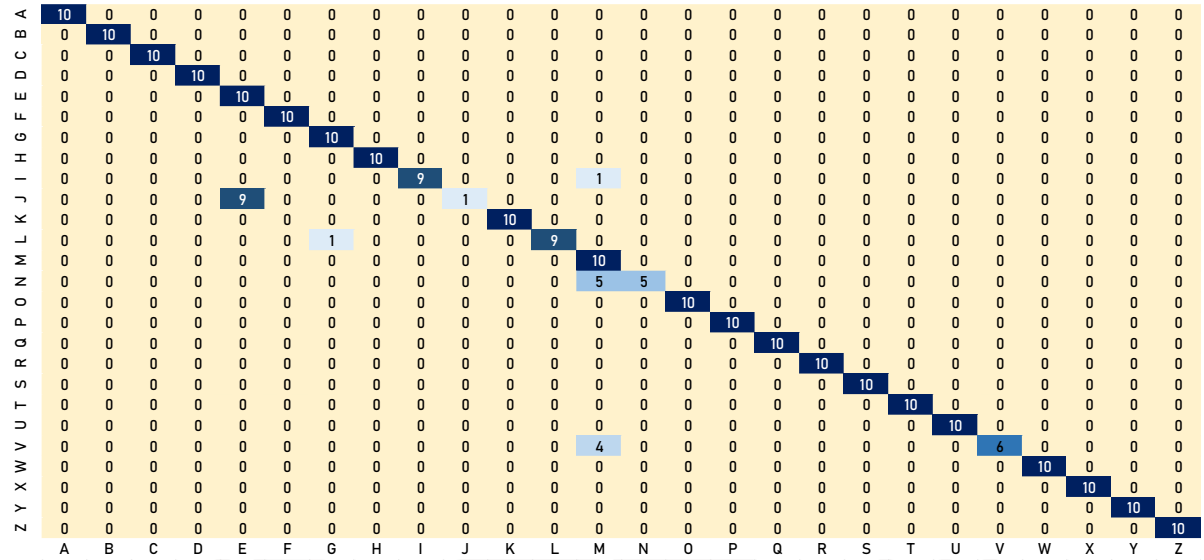
Gambar 4.16 *Confusion matrix* pengujian foto latar kaos

Pengujian Foto dengan Latar Tembok

Pengujian pada kategori ini menggunakan tembok berwarna putih sebagai latarnya. Contoh latar tembok putih bisa dilihat pada Gambar 3.2 pada huruf C dan D. Sebelum menguji, perlu disiapkan dahulu citra alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.14. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.17, dari 26 alfabet hanya ada 1 huruf yang belum berhasil ditebak dengan benar yaitu huruf J. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut

menandakan 25 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 92.31% pada pengujian foto dengan latar tembok.

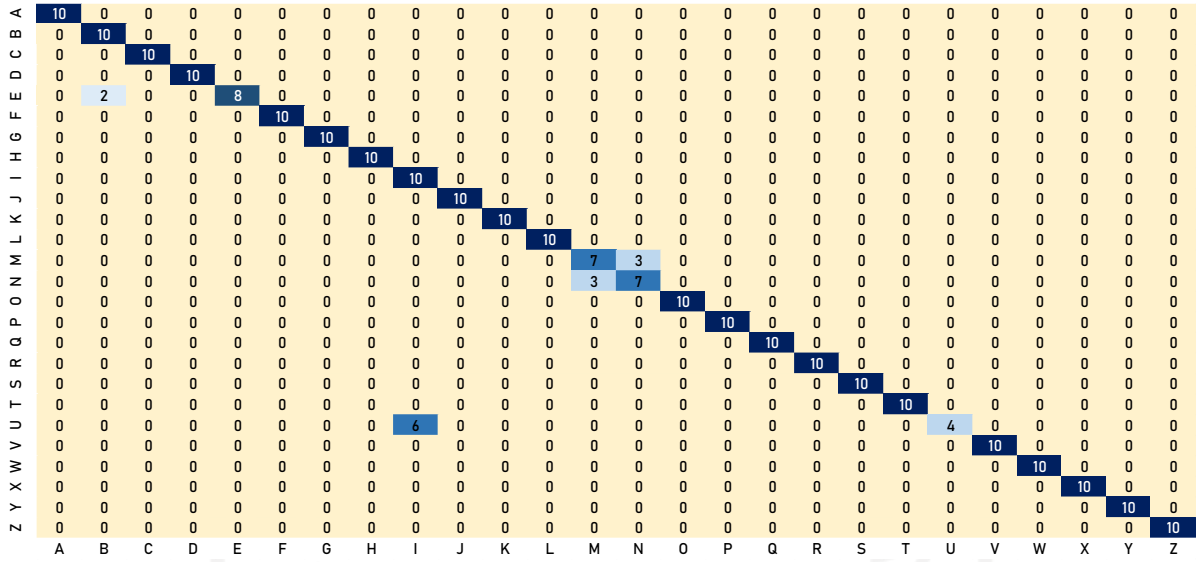


Gambar 4.17 *Confusion matrix* pengujian foto latar tembok

Pengujian Foto dengan Latar Kemeja

Pengujian pada kategori ini menggunakan kemeja bercorak sebagai latarnya. Contoh latar kemeja bercorak bisa dilihat pada Gambar 3.2 pada huruf E dan F. Sebelum menguji, perlu disiapkan dahulu citra alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.14. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.18, dari 26 alfabet hanya ada 1 huruf yang belum berhasil ditebak dengan benar yaitu huruf U. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 25 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 94.62% pada pengujian foto dengan latar kemeja.

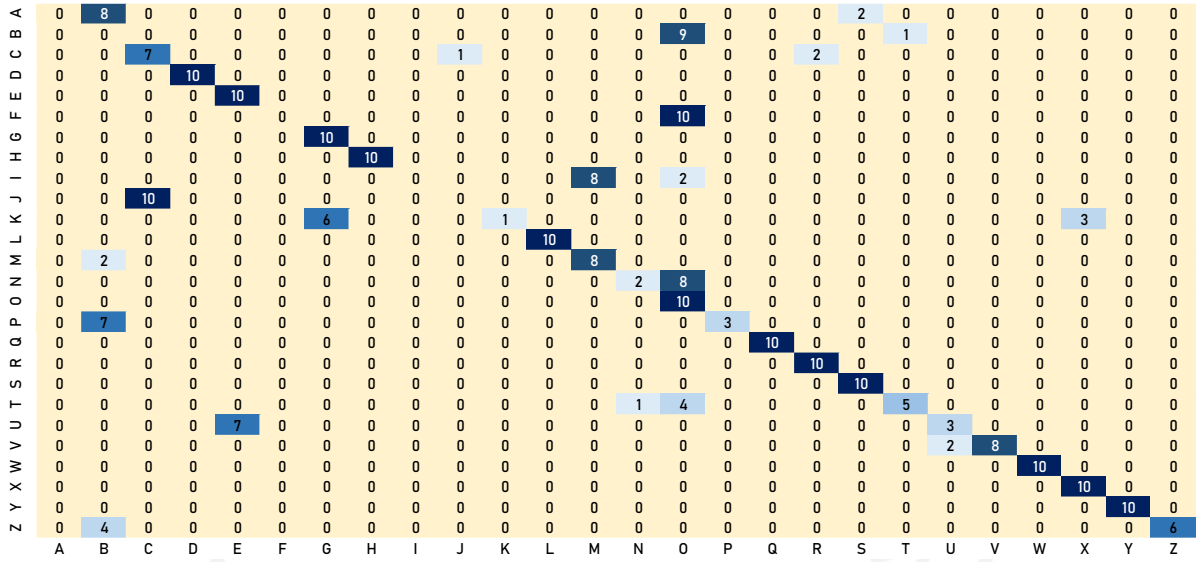


Gambar 4.18 *Confusion matrix* pengujian foto latar kemeja

Pengujian Foto dengan Latar Baju

Pengujian pada kategori ini menggunakan baju berwarna hitam sebagai latarnya. Latar baju belum pernah dipelajari oleh model sebelumnya yang berarti pengujian ini bertujuan untuk menguji kemampuan model terhadap data dengan latar baru. Contoh latar baju bisa dilihat pada Gambar 3.4 huruf A dan B. Sebelum menguji, perlu disiapkan dahulu citra alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.14. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.19, dari 26 alfabet ada 9 huruf yang belum berhasil ditebak dengan benar yaitu huruf A, huruf B, huruf F, huruf I, huruf J, huruf K, huruf N, huruf P, dan huruf U. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 17 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 62.69% pada pengujian foto dengan latar baju.

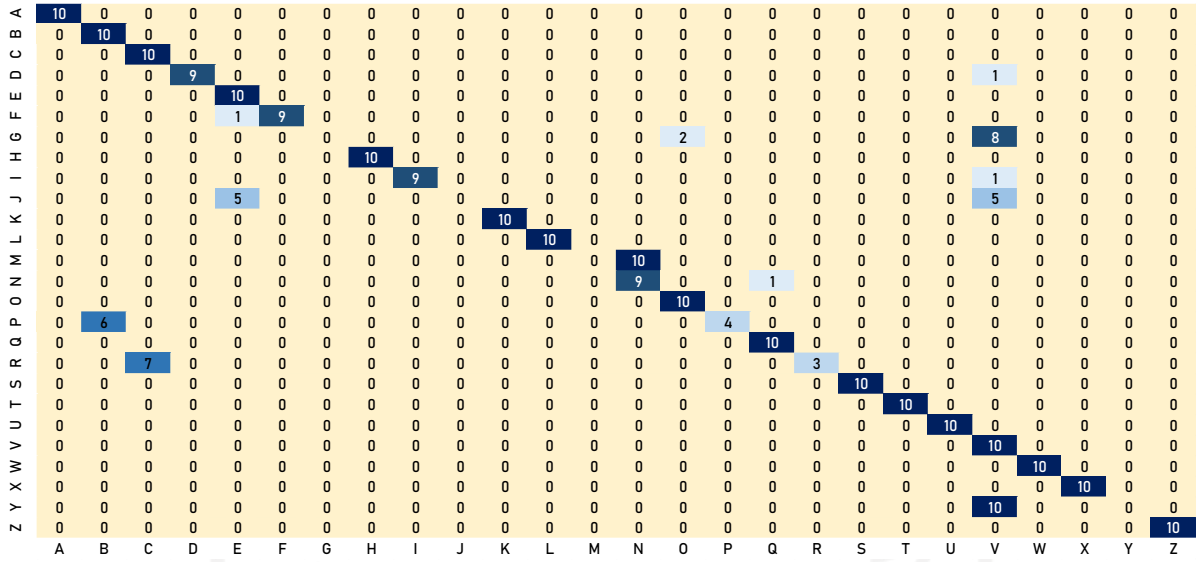


Gambar 4.19 *Confusion matrix* pengujian foto latar baju

Pengujian Foto dengan Latar Jubah

Pengujian pada kategori ini menggunakan jubah berwarna biru sebagai latarnya. Latar jubah belum pernah dipelajari oleh model sebelumnya yang berarti pengujian ini bertujuan untuk menguji kemampuan model terhadap data dengan latar baru. Contoh latar jubah bisa dilihat pada Gambar 3.4 huruf C dan D. Sebelum menguji, perlu disiapkan dahulu citra alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.14. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.20, dari 26 alfabet ada 6 huruf yang belum berhasil ditebak dengan benar yaitu huruf G, huruf J, huruf M, huruf P, huruf R, dan huruf Y. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 20 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 78.08% pada pengujian foto dengan latar jubah.

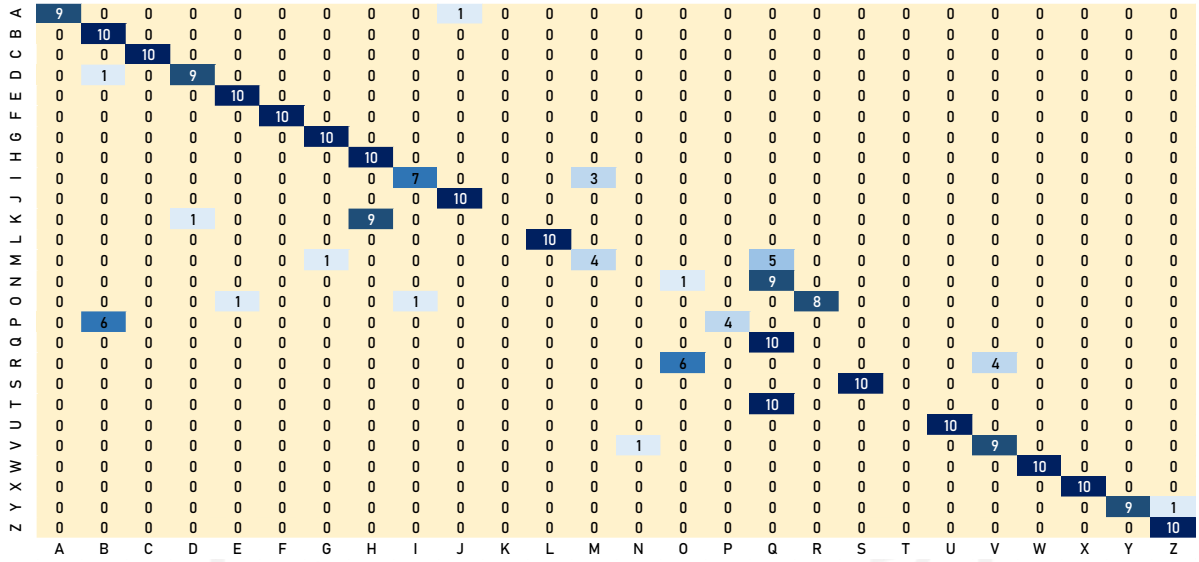


Gambar 4.20 *Confusion matrix* pengujian foto latar jubah

Pengujian Foto dengan Peraga Ahli

Pengujian pada kategori ini menggunakan kaos berwarna putih sebagai latarnya. Pengujian menggunakan peraga ahli bertujuan untuk melihat kemampuan model terhadap gerakan bahasa isyarat yang semestinya. Contoh gerakan dari peraga ahli bisa dilihat pada Gambar 3.5. Sebelum menguji, perlu disiapkan dahulu citra alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.14. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.21, dari 26 alfabet ada 7 huruf yang belum berhasil ditebak dengan benar yaitu huruf K, huruf M, huruf N, huruf O, huruf P, huruf R dan huruf T. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 19 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 73.46% pada pengujian foto dengan peraga ahli.



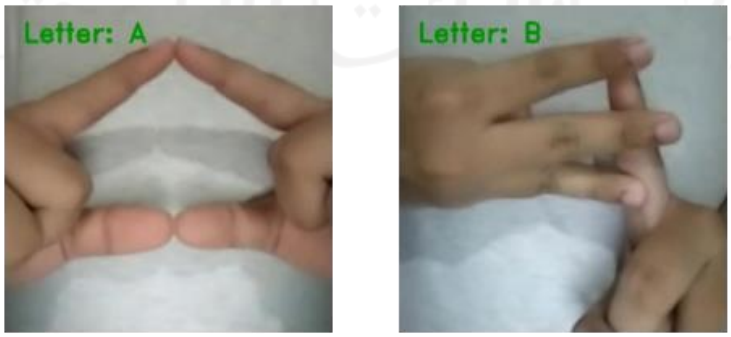
Gambar 4.21 *Confusion matrix* pengujian foto dengan peraga ahli

4.6.2 Pengujian Data Video

Video akan dipecah dengan mengambil tiap *frame* yang ada di video lalu menyimpannya ke dalam larik. Selanjutnya, *frame* dalam larik dipanggil secara berurutan lalu diubah ukuran citranya sesuai dengan yang sudah ditetapkan yaitu 64 x 64, lalu dimasukkan ke dalam fungsi *predict* untuk mendapatkan hasil prediksinya, begitupun yang terjadi pada *frame* selanjutnya.

Kurang lebih sama seperti yang terjadi pada pengujian citra, yang membedakannya adalah pada pengujian video hasil prediksinya dituliskan pada citra seperti pada Gambar 4.22. Prediksi ditempel pada citra lalu disimpan ke dalam folder *temporary* dengan nama yang berurutan sesuai dengan urutan *frame*nya.

Setelah semua *frame* diprediksi dan disimpan ke dalam satu folder dengan nama yang berurutan, langkah selanjutnya adalah menggabungkan semua citra itu kembali menjadi video dengan menggunakan fungsi *VideoWriter* dari *library opencv*.



Gambar 4.22 Sampel citra dari *folder temporary*

Untuk melihat kemampuan model untuk mendeteksi dengan baik, telah disiapkan video yang berisi bahasa isyarat dari huruf A sampai huruf Z. Video tersebut akan dimasukkan ke dalam pengujian lalu dihitung berapa banyak huruf yang berhasil ditebak dan berapa yang gagal untuk ditebak dengan benar. Contoh citra terprediksi dengan benar dan citra terprediksi dengan salah bisa dilihat pada Gambar 4.23.

Gambar 4.23 Tangkap layar pengujian video untuk (a) huruf C terprediksi C (b) huruf M terprediksi N

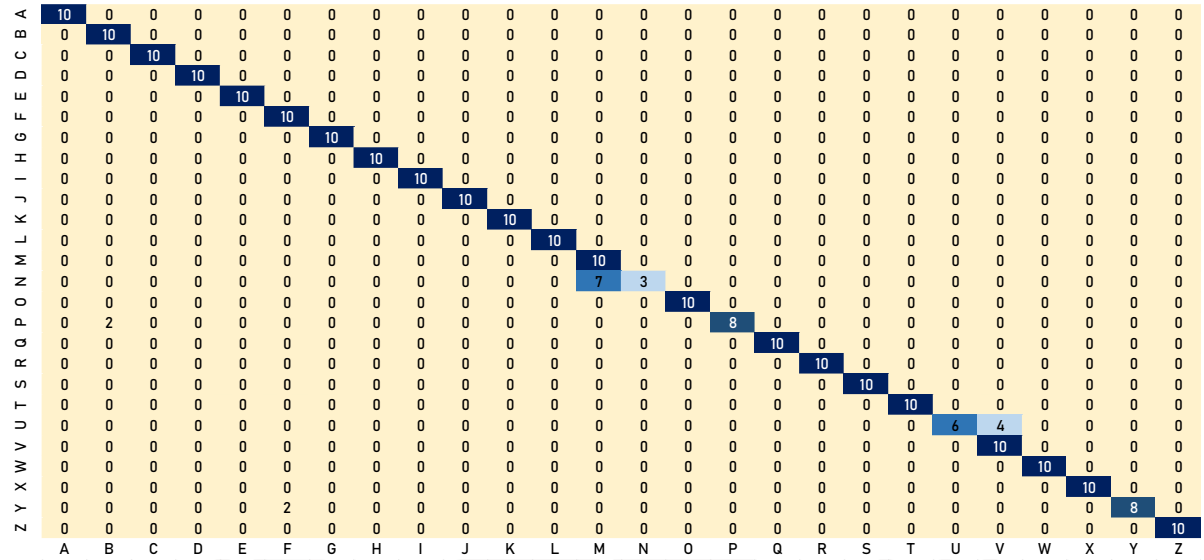
Dalam pengujian video ini, terbagi ke dalam enam kategori yang berbeda. Kategorinya adalah pengujian menggunakan latar tembok, pengujian menggunakan latar kaos, pengujian menggunakan latar kemeja, pengujian menggunakan latar baju, pengujian menggunakan latar jubah, dan pengujian dengan peraga ahli. Hasil pengujian menunjukan performa yang cukup bagus, walaupun ada beberapa huruf yang membutuhkan sedikit waktu untuk dapat ditebak dengan benar. Supaya lebih jelas, berikut penjelasan hasil pengujian pada masing-masing kategori.

Pengujian Video dengan Latar Kaos

Pengujian pada kategori ini menggunakan kaos berwarna putih sebagai latarnya. Contoh latar kaos putih bisa dilihat pada Gambar 3.2 pada huruf A dan B. Sebelum menguji, perlu disiapkan dahulu video berisi gerakan alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.22. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.24, dari 26 alfabet hanya ada 1 huruf yang belum berhasil ditebak dengan benar yaitu huruf N. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut

menandakan 25 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 94.23% pada pengujian video dengan latar kaos.

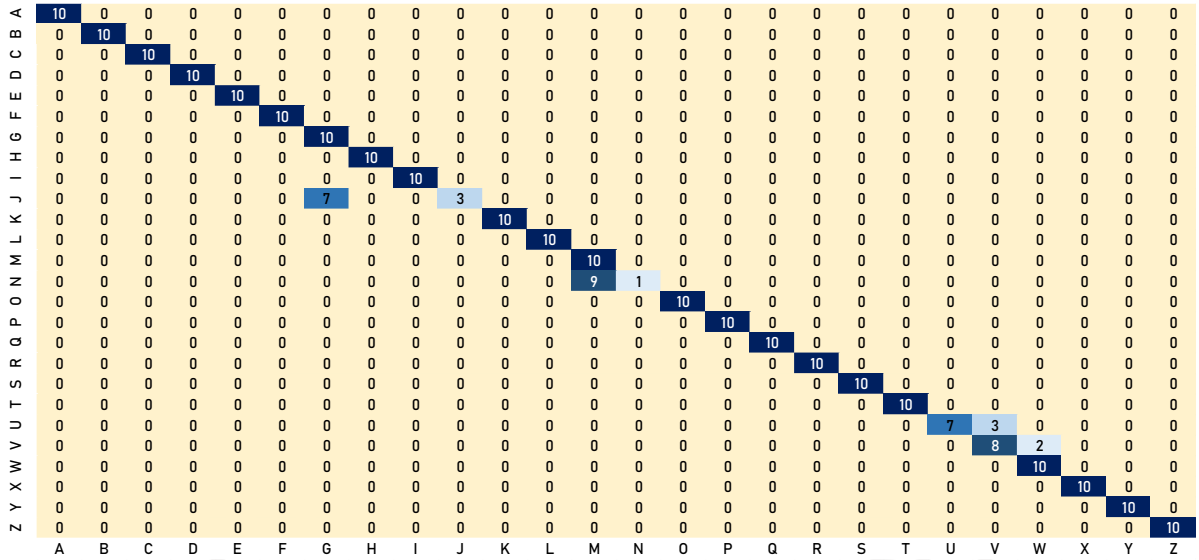


Gambar 4.24 *Confusion matrix* pengujian video latar kaos

Pengujian Video dengan Latar Tembok

Pengujian pada kategori ini menggunakan tembok berwarna putih sebagai latarnya. Contoh latar tembok putih bisa dilihat pada Gambar 3.2 pada huruf C dan D. Sebelum menguji, perlu disiapkan dahulu video berisi gerakan alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.22. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.25, dari 26 alfabet hanya ada 2 huruf yang belum berhasil ditebak dengan benar yaitu huruf J dan huruf N. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 24 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 91.92 pada pengujian video dengan latar tembok.

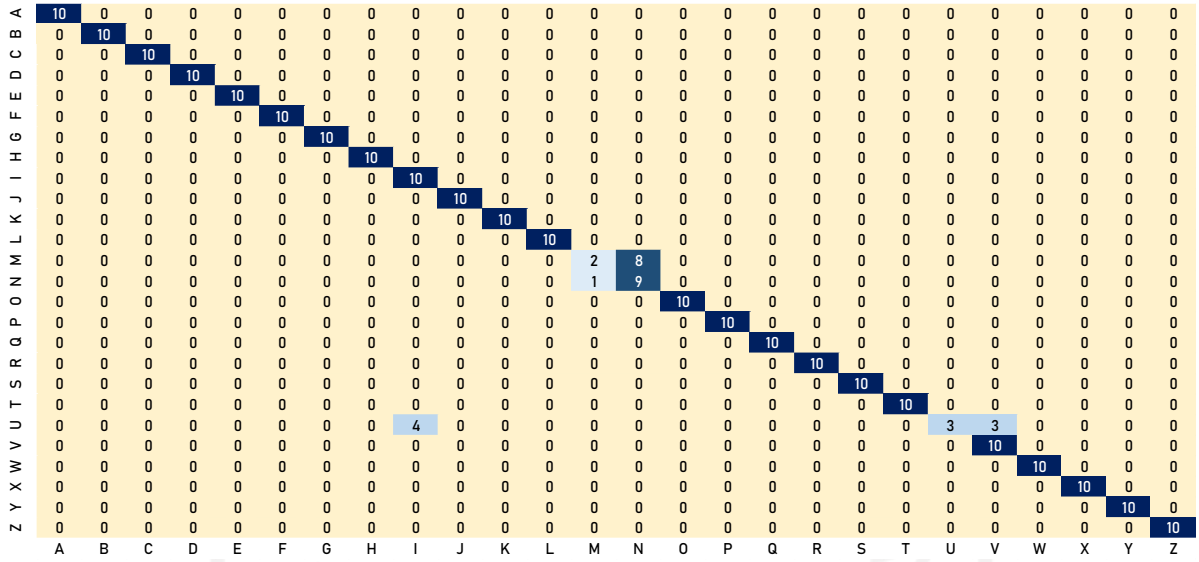


Gambar 4.25 *Confusion matrix* pengujian video latar tembok

Pengujian Video dengan Latar Kemeja

Pengujian pada kategori ini menggunakan kemeja bercorak sebagai latarnya. Contoh latar kemeja bercorak bisa dilihat pada Gambar 3.2 pada huruf E dan F. Sebelum menguji, perlu disiapkan dahulu video berisi gerakan alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.22. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.26, dari 26 alfabet hanya ada 2 huruf yang belum berhasil ditebak dengan benar yaitu huruf M, dan huruf U. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 24 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 93.85% pada pengujian video dengan latar kemeja.

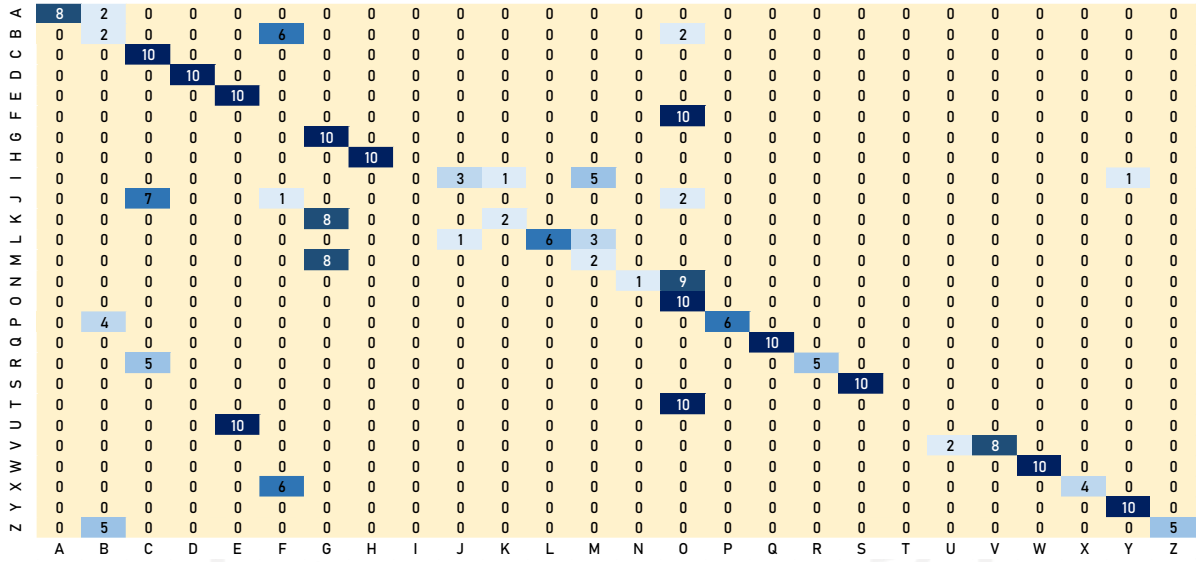


Gambar 4.26 *Confusion matrix* pengujian video latar kemeja

Pengujian Video dengan Latar Baju

Pengujian pada kategori ini menggunakan baju berwarna hitam sebagai latarnya. Latar baju belum pernah dipelajari oleh model sebelumnya yang berarti pengujian ini bertujuan untuk menguji kemampuan model terhadap data dengan latar baru. Contoh latar baju bisa dilihat pada Gambar 3.4 huruf A dan B. Sebelum menguji, perlu disiapkan dahulu video berisi gerakan alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.22. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.27, dari 26 alfabet ada 10 huruf yang belum berhasil ditebak dengan benar yaitu huruf B, huruf F, huruf I, huruf J, huruf K, huruf M, huruf N, huruf T, huruf U, dan huruf X. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 16 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 57.31% pada pengujian video dengan latar baju.

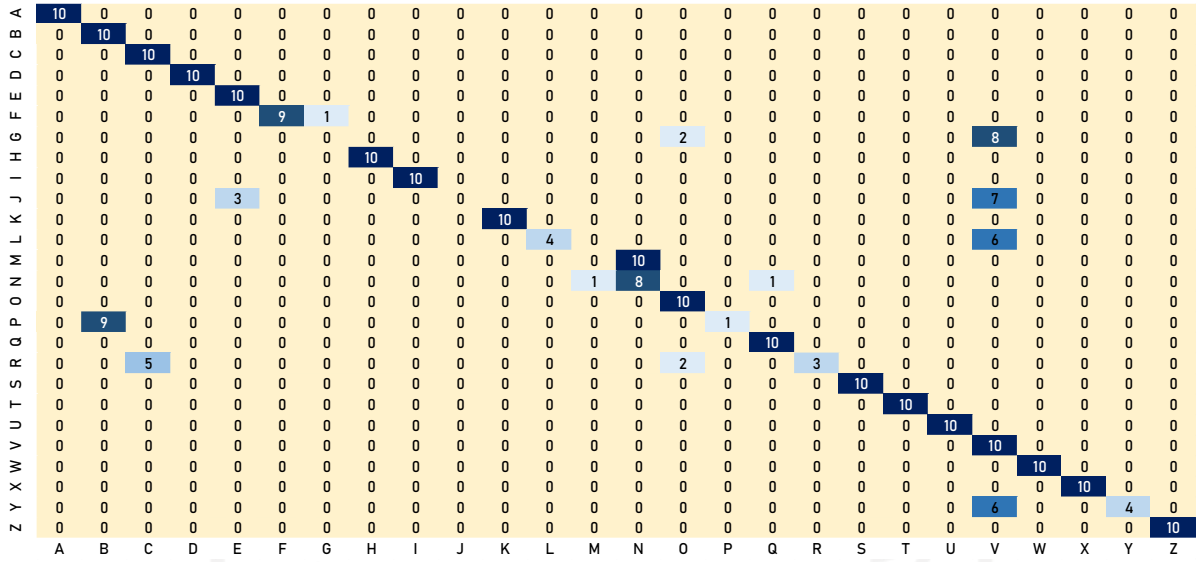


Gambar 4.27 *Confusion matrix* pengujian video latar baju

Pengujian Video dengan Latar Jubah

Pengujian pada kategori ini menggunakan jubah berwarna biru sebagai latarnya. Latar jubah belum pernah dipelajari oleh model sebelumnya yang berarti pengujian ini bertujuan untuk menguji kemampuan model terhadap data dengan latar baru. Contoh latar jubah bisa dilihat pada Gambar 3.4 huruf C dan D. Sebelum menguji, perlu disiapkan dahulu video berisi gerakan alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.22. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.28, dari 26 alfabet ada 7 huruf yang belum berhasil ditebak dengan benar yaitu huruf G, huruf J, huruf L, huruf M, huruf P, huruf R, dan huruf Y. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 19 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 76.54% pada pengujian video dengan latar baju.

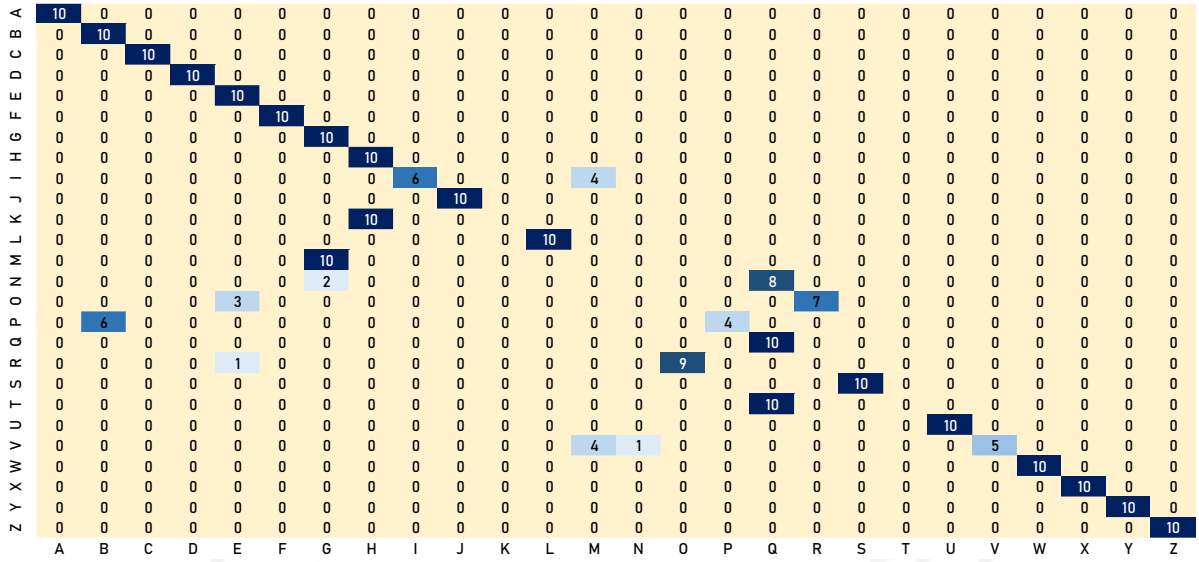


Gambar 4.28 *Confusion matrix* pengujian video latar jubah

Pengujian Video dengan Peraga Ahli

Pengujian pada kategori ini menggunakan kaos berwarna putih sebagai latarnya. Pengujian menggunakan peraga ahli bertujuan untuk melihat kemampuan model terhadap gerakan bahasa isyarat yang semestinya. Contoh gerakan dari peraga ahli bisa dilihat pada Gambar 3.5. Sebelum menguji, perlu disiapkan dahulu video berisi gerakan alfabet bahasa isyarat Indonesia untuk dimasukan ke dalam program. Hasil keluarannya kurang lebih sama seperti yang ada pada Gambar 4.22. Hasil pengujian dicatat satu per satu secara manual dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.29, dari 26 alfabet ada 7 huruf yang belum berhasil ditebak dengan benar yaitu huruf K, huruf M, huruf N, huruf O, huruf P, huruf R, dan huruf T. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 19 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 71.15% pada pengujian video dengan peraga ahli.

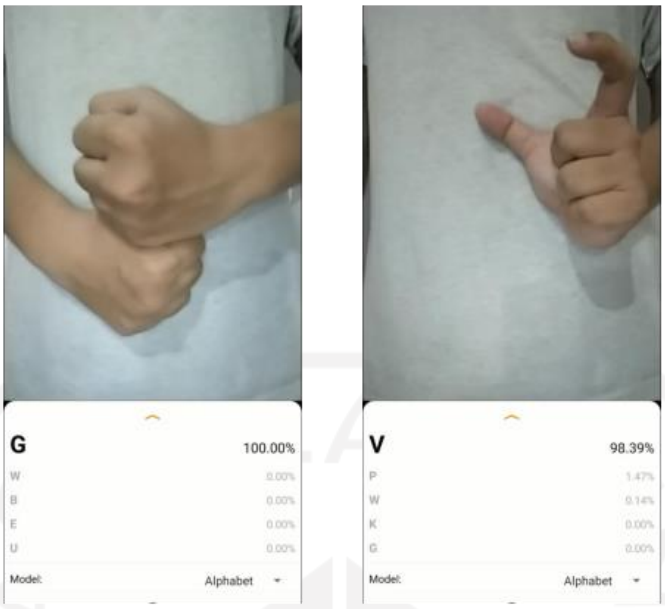


Gambar 4.29 *Confusion matrix* pengujian video dengan peraga ahli

4.6.3 Pengujian Data *Realtime*

Berbeda dengan pengujian foto dan pengujian video, pengujian *realtime* dilakukan di perangkat *smartphone* dengan model yang sudah disimpan dalam format tflite. Menggunakan aplikasi contoh yang ada di laman Tensorflow Lite, aplikasi berbasis android yang sudah siap pakai untuk proses pengujian *realtime*. Hanya saja butuh sedikit penyesuaian seperti mengganti model bawaan dengan model tflite yang dihasilkan dari pelatihan pada penelitian ini.

Pengujian *realtime* dilakukan dengan berdiri di depan kamera sambil menggerakan bahasa isyarat dan hasil prediksinya akan terlihat langsung secara *realtime*. Benar atau salahnya prediksi kembali lagi kepada manusia atau pengguna yang menilai. Contoh citra terprediksi dengan benar dan citra terprediksi dengan salah bisa dilihat pada Gambar 4.30.



Gambar 4.30 Tangkap layar pengujian *realtime* untuk (a) huruf G dengan hasil prediksi benar (b) huruf U dengan hasil prediksi salah

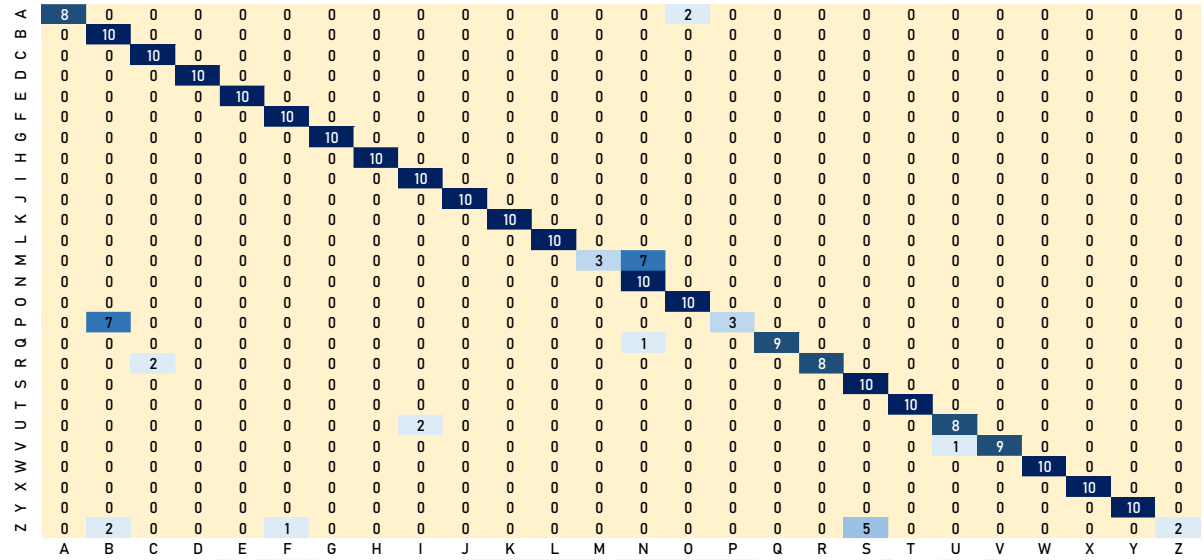
Dalam pengujian *realtime* ini, terbagi lagi ke dalam lima kategori yang berbeda. Kategorinya adalah pengujian menggunakan latar tembok, pengujian menggunakan latar kaos, pengujian menggunakan latar kemeja, pengujian menggunakan latar baju, dan pengujian menggunakan latar jubah. Terdapat beberapa kendala dalam pengujian *realtime*, yaitu pada beberapa huruf yang membutuhkan sedikit waktu dan posisi tangan yang tepat untuk dapat ditebak dengan benar.

Pengujian *Realtime* dengan Latar Kaos

Pengujian pada kategori ini menggunakan kaos berwarna putih sebagai latarnya. Contoh latar kaos bisa dilihat pada Gambar 3.2 pada huruf A dan B. Sebelum melakukan pengujian, pastikan dahulu posisi tangan dapat ditangkap kamera dengan baik. Setelah itu, melakukan pengujian menggunakan aplikasi dengan cara memperagakan gerakan huruf bahasa isyarat di depan lensa untuk dikenali oleh model. Hasil pengujian dicatat satu per satu dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.31, dari 26 alfabet hanya ada 3 huruf yang belum berhasil ditebak dengan benar yaitu huruf M, huruf P, dan huruf Z. Sedangkan yang lainnya berhasil ditebak

dengan baik. Hal tersebut menandakan 23 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 88.46% pada pengujian *realtime* dengan latar kaos.

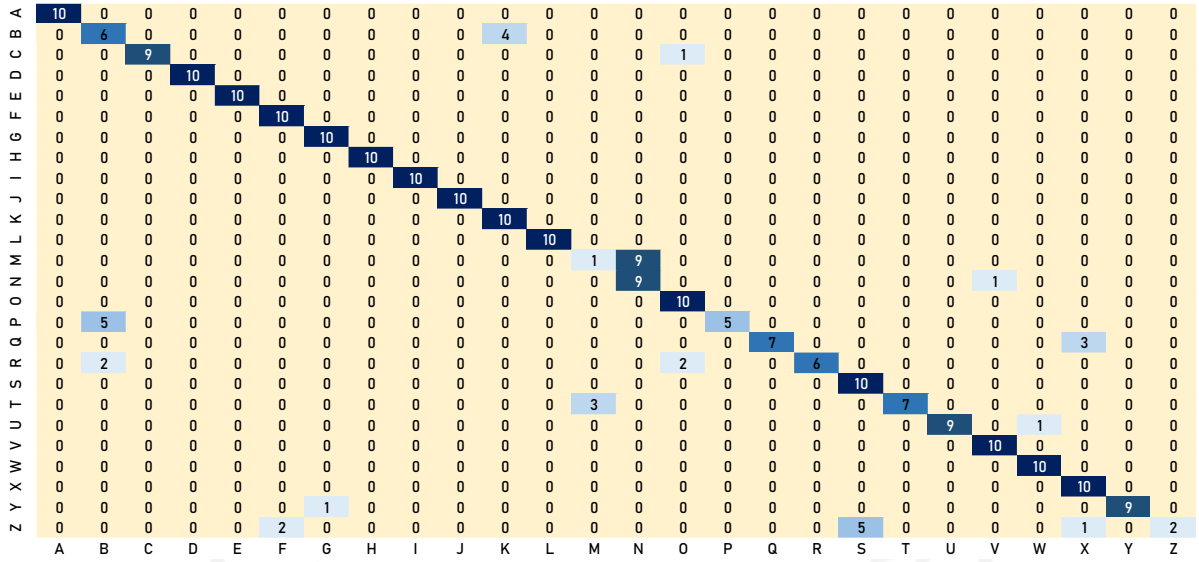


Gambar 4.31 *Confusion matrix* pengujian *realtime* latar kaos

Pengujian *Realtime* dengan Latar Tembok

Pengujian pada kategori ini menggunakan tembok berwarna putih sebagai latarnya. Contoh latar tembok bisa dilihat pada Gambar 3.2 pada huruf C dan D. Sebelum melakukan pengujian, pastikan dahulu posisi tangan dapat ditangkap kamera dengan baik. Setelah itu, melakukan pengujian menggunakan aplikasi dengan cara memperagakan gerakan huruf bahasa isyarat di depan lensa untuk dikenali oleh model. Hasil pengujian dicatat satu per satu dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.32, dari 26 alfabet hanya ada 2 huruf yang belum berhasil ditebak dengan benar yaitu huruf M dan huruf Z. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 24 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 84.62% pada pengujian *realtime* dengan latar tembok.

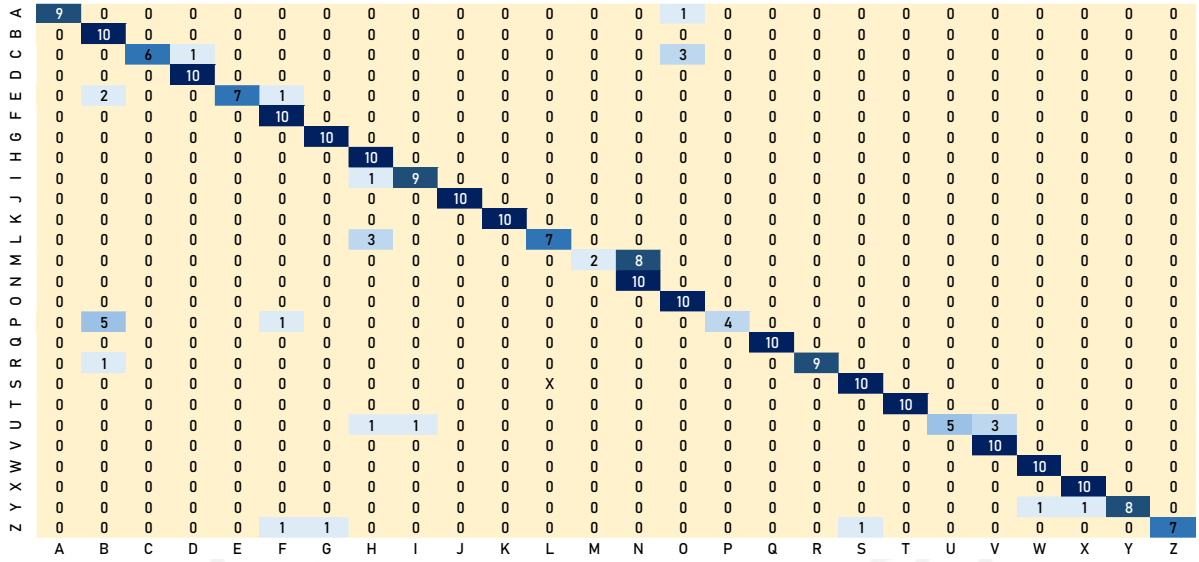


Gambar 4.32 *Confusion matrix* pengujian *realtime* latar tembok

Pengujian *Realtime* dengan Latar Kemeja

Pengujian pada kategori ini menggunakan kemeja bercorak sebagai latarnya. Contoh latar kemeja bisa dilihat pada Gambar 3.2 pada huruf E dan F. Sebelum melakukan pengujian, pastikan dahulu posisi tangan dapat ditangkap kamera dengan baik. Setelah itu, melakukan pengujian menggunakan aplikasi dengan cara memperagakan gerakan huruf bahasa isyarat di depan lensa untuk dikenali oleh model. Hasil pengujian dicatat satu per satu dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.33, dari 26 alfabet hanya ada 2 huruf yang belum berhasil ditebak dengan benar yaitu huruf M, dan huruf P. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 24 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 85.77% pada pengujian *realtime* dengan latar kemeja.

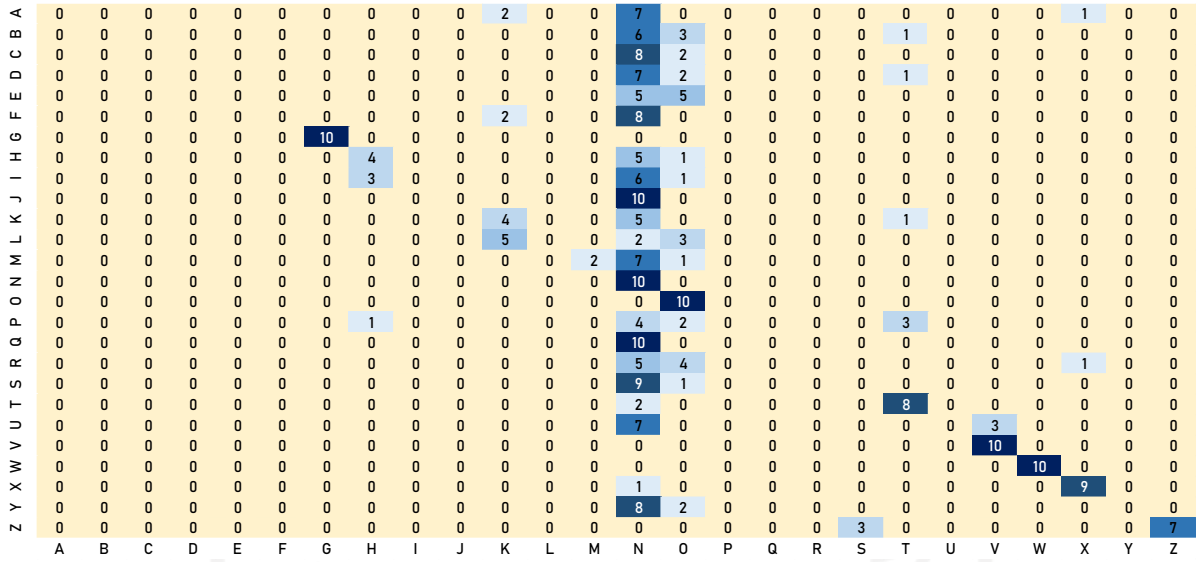


Gambar 4.33 *Confusion matrix* pengujian *realtime* latar kemeja

Pengujian *Realtime* dengan Latar Baju

Pengujian pada kategori ini menggunakan baju berwarna hitam sebagai latarnya. Latar baju belum pernah dipelajari oleh model sebelumnya yang berarti pengujian ini bertujuan untuk menguji kemampuan model terhadap data dengan latar baru. Contoh latar baju bisa dilihat pada Gambar 3.4 huruf A dan B. Sebelum melakukan pengujian, pastikan dahulu posisi tangan dapat ditangkap kamera dengan baik. Setelah itu, melakukan pengujian menggunakan aplikasi dengan cara memperagakan gerakan huruf bahasa isyarat di depan lensa untuk dikenali oleh model. Hasil pengujian dicatat satu per satu dari huruf A sampai huruf Z.

Berdasarkan Gambar 4.34, dari 26 alfabet ada 18 huruf yang belum berhasil ditebak dengan benar yaitu huruf A, huruf B, huruf C, huruf D, huruf E, huruf F, huruf H, huruf I, huruf J, huruf K, huruf L, huruf M, huruf P, huruf Q, huruf R, huruf S, huruf U, dan huruf Y. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 8 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 32.31% pada pengujian *realtime* dengan latar baju.

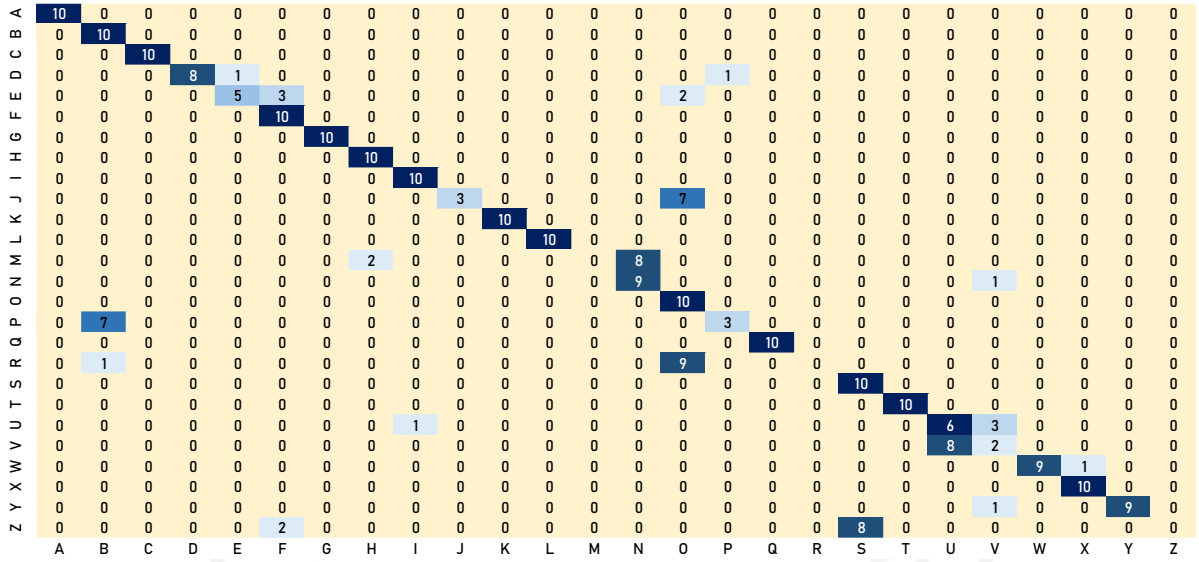


Gambar 4.34 *Confusion matrix* pengujian *realtime* latar baju

Pengujian *Realtime* dengan Latar Jubah

Pengujian pada kategori ini menggunakan jubah berwarna biru sebagai latarnya. Latar jubah belum pernah dipelajari oleh model sebelumnya yang berarti pengujian ini bertujuan untuk menguji kemampuan model terhadap data dengan latar baru. Contoh latar jubah bisa dilihat pada Gambar 3.4 huruf C dan D. Sebelum melakukan pengujian, pastikan dahulu posisi tangan dapat ditangkap kamera dengan baik. Setelah itu, melakukan pengujian menggunakan aplikasi dengan cara memperagakan gerakan huruf bahasa isyarat di depan lensa untuk dikenali oleh model. Hasil pengujian dicatat satu per satu dari huruf A sampai huruf Z.

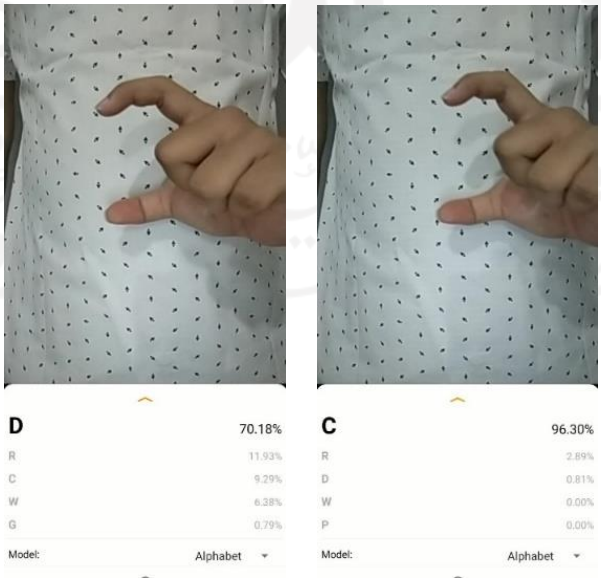
Berdasarkan Gambar 4.35, dari 26 alfabet ada 6 huruf yang belum berhasil ditebak dengan benar yaitu huruf J, huruf M, huruf P, huruf R, huruf V, dan huruf Z. Sedangkan yang lainnya berhasil ditebak dengan baik. Hal tersebut menandakan 20 dari 26 alfabet berhasil ditebak dengan benar. Tingkat akurasi kumulatifnya mencapai 74.62% pada pengujian *realtime* dengan latar jubah.



Gambar 4.35 *Confusion matrix* pengujian *realtime* latar jubah

4.7 Evaluasi Pengujian

Model yang dilatih sudah bisa menunjukan performa yang cukup baik dilihat dari kemampuannya dalam mendeteksi citra dengan tepat berdasarkan hasil pengujian model di sub-bab sebelumnya. Walaupun begitu, model membutuhkan sedikit waktu untuk mendeteksi dengan tepat. Terutama saat melakukan pengujian *realtime*, model tidak langsung bisa menebak bahasa isyarat dengan benar. Model baru bisa mendeteksi dengan tepat saat posisi tangan sudah berada dalam wilayah tangkapannya. Tidak semua alfabet mengalami hal ini melainkan hanya beberapa alfabet saja contohnya huruf C.



Gambar 4.36 Huruf C terdeteksi dengan benar saat posisinya sudah sesuai

Bisa dilihat hasil pengujian realtime dengan huruf C pada Gambar 4.36, kedua tangkapan layar tersebut memiliki hasil prediksi yang berbeda walaupun secara sekilas lokasi tangan pada citra tidak ada bedanya. Itulah yang dimaksud dengan model baru bisa mendeteksi dengan tepat saat posisi tangan sudah berada dalam posisi yang tepat. Sekaligus bukti dari keluhan yang sudah pernah disebutkan sebelumnya yaitu model membutuhkan sedikit waktu untuk dapat ditebak dengan benar.

Intensitas cahaya juga menjadi faktor penting dalam tahap pengujian. Perlu diperhatikan tempat pengujian supaya tidak terlalu gelap ataupun terlalu terang. Kendala dengan intensitas cahaya juga disebutkan oleh Nguyen (Nguyen et al., 2013) dalam penelitiannya.

Tabel 4.4 Hasil pengujian model dengan performanya masing-masing

Jenis Pengujian	Latar	Akurasi
Pengujian foto	Kaos putih	251/260 (96.54%)
Pengujian foto	Tembok putih	240/260 (92.31%)
Pengujian foto	Kemeja putih bercorak	246/260 (94.62%)
Pengujian foto	Baju hitam	163/260 (62.69%)
Pengujian foto	Jubah biru	203/260 (78.08%)
Pengujian foto	Kaos putih (peraga ahli)	191/260 (73.46%)
Pengujian video	Kaos putih	245/260 (94.23%)
Pengujian video	Tembok putih	239/260 (91.92%)
Pengujian video	Kemeja putih bercorak	244/260 (93.85%)
Pengujian video	Baju hitam	149/260 (57.31%)
Pengujian video	Jubah biru	199/260 (76.54%)
Pengujian video	Kaos putih (peraga ahli)	185/260 (71.15%)
Pengujian realtime	Kaos putih	230/260 (88.46%)
Pengujian realtime	Tembok putih	220/260 (84.62%)
Pengujian realtime	Kemeja putih bercorak	223/260 (85.77%)
Pengujian realtime	Baju hitam	84/260 (32.31%)
Pengujian realtime	Jubah biru	194/260 (74.62%)

Melihat hasil dari seluruh pengujian pada Tabel 4.4, hanya ada satu pengujian yang termasuk kategori belum berhasil ditebak dengan benar karena tidak mencapai *threshold* akurasi sebesar 46.8% yaitu pada pengujian *realtime* dengan latar baju hitam dengan nilai akurasinya sebesar 32.31%. Sedangkan pada pengujian lainnya termasuk ke dalam kategori berhasil ditebak dengan baik.

Berdasarkan latar pengujian, latar kaos berwarna putih menghasilkan akurasi terbaik pada pengujian foto, pengujian video, dan pengujian *realtime*. Sedangkan latar baju berwarna hitam menghasilkan akurasi yang paling rendah pada pengujian foto, pengujian video, dan pengujian *realtime*.

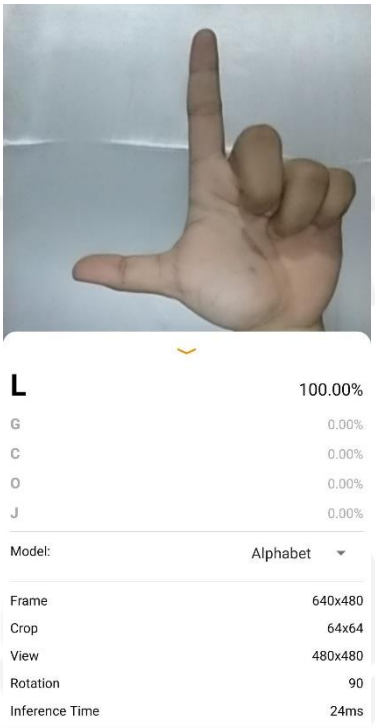
Di antara dua latar yang belum pernah dikenali oleh model, latar jubah berwarna biru menghasilkan akurasi yang lebih baik dibandingkan dengan latar baju berwarna hitam. Hal tersebut dirasa wajar karena latar berwarna hitam kontras dengan latar pada tahap pelatihan yaitu warna putih. Sedangkan warna biru masih memiliki unsur putih atau cerah sehingga masih bisa dikenali dengan benar oleh model.

Pengujian foto dengan peraga ahli dan pengujian video dengan peraga ahli termasuk dalam kategori berhasil karena melewati *threshold*. Namun, huruf K pada kedua pengujian tersebut tidak ada satupun yang berhasil ditebak dengan benar berdasarkan *confusion matrix* Gambar 4.21 dan Gambar 4.29. Ditemukan fakta bahwa citra huruf K yang diujikan sedikit berbeda dengan citra huruf K yang dilatih dari segi kemiringan gerakan tangan seperti pada Gambar 4.37. Hal tersebut diduga menjadi penyebab huruf K tidak berhasil ditebak dengan benar saat pengujian foto dengan peraga ahli dan pengujian video dengan peraga ahli. Masalah perbedaan gerakan tersebut mungkin juga terjadi pada huruf lainnya atau pada pengujian lainnya, tidak hanya terjadi pada pengujian dengan peraga ahli saja.

Gambar 4.37 Sampel gerakan huruf K pada (a) citra latih (b) citra uji

Secara keseluruhan, model mampu menunjukkan hasil yang cukup memuaskan. Hal itu didasari oleh hasil dari pengujian foto, pengujian video, dan pengujian *realtime* yang hampir seluruh latarnya termasuk ke dalam kategori berhasil ditebak dengan benar. Model juga sudah mampu digunakan pada data dengan latar baru yaitu baju dan jubah, walaupun tentu saja hasilnya masih belum bisa sebagus data yang sudah dipelajari oleh model.

Tidak sampai situ saja, model juga sudah bisa mengenali citra secara cepat dengan durasi pendeteksian rata-rata selama 24 *millisecond* pada pengujian *realtime* untuk huruf L. Hasil tangkapan layarnya bisa dilihat pada Gambar 4.38.



Gambar 4.38 Kecepatan deteksi citra pada pengujian *realtime*

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan terhadap topik pengenalan bahasa isyarat, diperoleh beberapa kesimpulan sebagai berikut.

a. Dalam menentukan kombinasi atau konfigurasi arsitektur CNN yang tepat, perlu melakukan berkali-kali percobaan hingga dirasa hasilnya sesuai atau lebih familiar disebut dengan *trial and error*. Walaupun terkesan coba-coba, tetapi harus ada beberapa faktor yang menjadi pertimbangan. Dalam penelitian ini, nilai *epoch*, model *optimizer*, dan arsitektur CNN lebih spesifiknya lapisan konvolusi dan lapisan *pooling* menjadi faktor utama penentu sebuah model dapat dihasilkan dengan performa yang baik. Ketiga faktor tersebut terus dikonfigurasi hingga mendapatkan hasil yang cukup baik. Menggunakan arsitektur pada yang sudah pernah digunakan oleh peneliti sebelumnya juga bisa menjadi jalan pintas untuk mendapatkan arsitektur CNN yang performanya cukup baik karena pastinya arsitektur tersebut sudah melalui rangkaian proses hingga terpilihlah arsitektur dengan performa paling baik menurut penelitian sebelumnya. Pada akhirnya, ditemukan nilai *epoch* sebanyak 100 untuk pelatihan, lalu model di*compile* dengan *RMSprop* model *optimizer*, dan menggunakan enam lapisan konvolusi dan tiga lapisan *maxpool* dengan *kernel pooling* 3x3 sebagai konfigurasi yang paling sesuai untuk digunakan pada penelitian ini.
b. Citra yang digunakan dalam tahap pelatihan semuanya memiliki latar dengan mayoritas berwarna putih yaitu, kaos putih, tembok putih, dan kemeja putih bercorak. Oleh karena itu, pengujian dengan ketiga latar tersebut juga menghasilkan akurasi yang cukup baik. Tidak cukup sampai situ, pengujian dengan latar selain warna putih juga dicoba dalam penelitian, yaitu dengan latar baju berwarna hitam dan jubah berwarna biru. Hasil yang buruk ditunjukan oleh latar baju berwarna hitam dengan menghasilkan akurasi terendah pada pengujian foto, pengujian video, dan pengujian *realtime*. Hal tersebut diduga karena warna hitam sangat kontras dengan warna putih yang menyebabkan model kesulitan untuk mengenali citra. Berbeda dengan latar jubah berwarna biru yang seluruh pengujiannya dapat melampaui nilai akurasi *threshold* yang telah ditentukan. Hal tersebut dirasa wajar karena warna biru masih memiliki unsur warna putih atau cerah. Dapat ditarik kesimpulan

bahwa dalam memilih latar pada tahap pengujian, usahakan untuk memilih latar dengan warna yang sama atau mendekati latar yang digunakan dalam tahap pelatihan model, dalam penelitian ini adalah latar dengan mayoritas berwarna putih atau cerah. Hindari pemilihan warna yang kontras dengan latar berwarna putih atau cerah, misalnya seperti warna hitam atau gelap supaya model tidak kesulitan untuk mengenali citra masukan.

c. Kemampuan model yang dihasilkan dalam penelitian sudah cukup kuat. Tidak hanya kemampuannya dalam mengenali *dataset* citra, model juga sudah bisa mendeteksi *dataset* video dan bahkan sudah bisa mendeteksi secara *realtime* yang dilakukan di perangkat *smartphone* dengan sistem operasi *Android*. Bicara mengenai performa dalam pengenalan, pengujian *realtime* berhasil mencapai akurasi terbaiknya sebesar 88.46% untuk citra dengan latar kaos putih. Tidak sampai situ saja, model juga sudah bisa mengenali citra secara cepat dengan durasi pendeteksian rata-rata selama 24 *millisecond* pada pengujian *realtime.* Hal tersebut membuktikan bahwa untuk melakukan pendeteksian bahasa isyarat tidak harus membutuhkan sumber daya yang besar.

5.2 Saran

Adapun beberapa saran yang bisa dijadikan referensi untuk penelitian kedepannya sebagai berikut.

a. Bahasa yang digunakan dalam penelitian ini adalah alfabet Bahasa Isyarat Indonesia (BISINDO). Supaya semakin lengkap, sangat mungkin untuk menambahkan kelas klasifikasi baru yaitu angka Bahasa Isyarat Indonesia (BISINDO).
b. Menambahkan variasi pada citra sangat memungkinkan. Seperti menggunakan warna latar yang berbeda-beda atau menambahkan latar yang tidak satu warna contohnya seperti di tempat terbuka.
c. Coba untuk menggunakan metode lainnya seperti MobileNet karena sudah mendukung algoritme deteksi objek.
d. Mencari pendekatan yang lain untuk melatih model. Menggunakan *dataset* video sebagai masukan bisa menjadi terobosan baru karena dengan menggunakan *dataset* video otomatis jangkauannya akan semakin luas. Apabila *dataset* citra hanya mendukung pengenalan huruf, dengan menggunakan *dataset* video mungkin saja bisa mendukung pengenalan kata atau bahkan kalimat.

DAFTAR PUSTAKA

Brownlee, J. (2016). Display Deep Learning Model Training History in Keras. Retrieved March 11, 2021, from https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/

Brownlee, J. (2018). A Gentle Introduction to Dropout for Regularizing Deep Neural Networks. Retrieved from https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

Chollet, F., & others. (2015). *MNIST CNN*. Retrieved from https://github.com/keras-team/keras/blob/keras-2/examples/mnist_cnn.py

Chollet, F., & Others. (2015a). *Adagrad*. Retrieved from https://keras.io/api/optimizers/adagrad/

Chollet, F., & Others. (2015b). *Adam*. Retrieved from https://keras.io/api/optimizers/adam/

Chollet, F., & Others. (2015c). *Adamax*. Retrieved from https://keras.io/api/optimizers/adamax/

Chollet, F., & Others. (2015d). *Batch Normalization*. Retrieved from https://keras.io/api/layers/normalization_layers/batch_normalization/

Chollet, F., & Others. (2015e). *Nadam*. Retrieved from https://keras.io/api/optimizers/Nadam/

Chollet, F., & Others. (2015f). *RMSprop*. Retrieved from https://keras.io/api/optimizers/rmsprop/

Chollet, F., & Others. (2015g). *SGD*. Retrieved from https://keras.io/api/optimizers/sgd/

Deaf, B. (2015). *ABJAD JARI BISINDO*. Retrieved from https://www.youtube.com/watch?v=Py6Ch1vBvL0&feature=youtu.be

Dertat, A. (2017). Applied Deep Learning - Part 4: Convolutional Neural Networks | by Arden Dertat | Towards Data Science. Retrieved March 11, 2021, from https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Dicoding Indonesia. (n.d.-a). Retrieved March 11, 2021, from https://www.dicoding.com/academies/184/tutorials/8502?hl=aktivasi

Dicoding Indonesia. (n.d.-b). Retrieved April 26, 2021, from https://www.dicoding.com/academies/184/tutorials/8477

Draelos, R. (2019). Best Use of Train/Val/Test Splits, with Tips for Medical Data. Retrieved April 22, 2021, from https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-

splits-with-tips-for-medical-data/#:~:text=Common ratios used are%3A,20%25 val%2C 20%25 test

Elsayed, R. A., Sayed, M. S., & Abdalla, M. I. (2018). Hand gesture recognition based on dimensionality reduction of histogram of oriented gradients. *2017 Proceedings of the Japan-Africa Conference on Electronics, Communications, and Computers, JAC-ECC 2017*, *2018-Janua*(1), 119–122. https://doi.org/10.1109/JEC-ECC.2017.8305792

Fadillah, R. Z. (2020). Model Penerjemah Bahasa Isyarat Indonesia (Bisindo) Menggunakan Convolutional Neural Network Model Penerjemah Bahasa Isyarat Indonesia (Bisindo) Menggunakan Convolutional Neural Network. *Fakultas Sains Dan Ilmu Komputer, Program Studi Ilmu Komputer, Universitas Pertamina*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Retrieved from https://books.google.co.id/books?id=Np9SDQAAQBAJ

Gumelar, G., Hafiar, H., & Subekti, P. (2018). Bahasa Isyarat Indonesia Sebagai Budaya Tuli Melalui Pemaknaan Anggota Gerakan Untuk Kesejahteraan Tuna Rungu. *Informasi*, *48*(1), 65.

Harpini, A. (2019). Disabilitas Rungu. Retrieved May 3, 2020, from https://pusdatin.kemkes.go.id/resources/download/pusdatin/infodatin/infodatin-tunarungu-2019.pdf

Ippolito, P. P. (n.d.). Hyperparameters Optimization. Retrieved November 21, 2020, from https://towardsdatascience.com/hyperparameters-optimization-526348bb8e2d

Jayaprakash, R., & Majumder, S. (2011). *Hand Gesture Recognition for Sign Language: A New Hybrid Approach*. *1*.

KLOBILITY. (n.d.). BISINDO dan SIBI: Apa Bedanya? Retrieved July 14, 2020, from https://www.klobility.id/post/perbedaan-bisindo-dan-sibi

Li, G., Tang, H., Sun, Y., Kong, J., Jiang, G., Jiang, D., … Liu, H. (2019). Hand gesture recognition based on convolution neural network. *Cluster Computing*, *22*, 2719–2729. https://doi.org/10.1007/s10586-017-1435-x

MATLAB. (n.d.). Convolutional Neural Network: 3 things you need to know. Retrieved June 24, 2020, from https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

Mishkin, D., Sergievskiy, N., & Matas, J. (2017). Systematic evaluation of convolution neural network advances on the Imagenet. *Computer Vision and Image Understanding*. https://doi.org/https://doi.org/10.1016/j.cviu.2017.05.007

Mufarroha, F. A., & Utaminingrum, F. (2017). Hand gesture recognition using adaptive network based fuzzy inference system and K-nearest neighbor. *International Journal of Technology*. https://doi.org/10.14716/ijtech.v8i3.3146

Nguyen, T.-N., Huynh, H.-H., & Meunier, J. (2013). Static Hand Gesture Recognition Using Artificial Neural Network. *Journal of Image and Graphics*, *1*(1), 34–38. https://doi.org/10.12720/joig.1.1.34-38

Oyedotun, O. K., & Khashman, A. (2017). Deep learning in vision-based static hand gesture recognition. *Neural Computing and Applications*, *28*(12), 3941–3951. https://doi.org/10.1007/s00521-016-2294-8

Patel, R., Dhakad, J., Desai, K., Gupta, T., & Correia, S. (2018). Hand Gesture Recognition System using Convolutional Neural Networks. *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, 1–6. https://doi.org/10.1109/CCAA.2018.8777621

Ritacheta. (2020). Guide To Optimizers For Machine Learning. Retrieved from https://analyticsindiamag.com/guide-to-optimizers-for-machine-learning/

Sistem Informasi Manajemen Penyandang Disabilitas Kementerian Sosial. (n.d.). Retrieved May 5, 2020, from https://simpd.kemsos.go.id/

Yang, J., & Yang, G. (2018). Modified Convolutional Neural Network Based on Dropout and the Stochastic Gradient Descent Optimizer. *Algorithms*, *11*(3), 28. https://doi.org/10.3390/a11030028

Yolanda, D., Gunadi, K., & Setyati, E. (2020). Pengenalan Alfabet Bahasa Isyarat Tangan Secara Real- Time dengan Menggunakan Metode Convolutional Neural Network dan Recurrent Neural Network. *Infra*, *8*(1).

LAMPIRAN

Tautan menuju *dataset*.

https://www.kaggle.com/achmadnoer/alfabet-bisindo