ANALISIS AKURASI PADA SIMBOL ABJAD SISTEM ISYARAT BAHASA INDONESIA (SIBI) DENGAN MENGGUNAKAN METODE CNN DAN YOLO *(YOU ONLY LOOK ONCE)*

TUGAS AKHIR

Diajukan Oleh:

SRI MAULIDA
NIM. 180705009
Mahasiswa Fakultas Sains dan Teknologi
Prodi Teknologi Informasi

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI AR-RANIRY
BANDA ACEH
2023 M /1445 H

ANALISIS AKURASI PADA SIMBOL ABJAD SISTEM ISYARAT BAHASA INDONESIA (SIBI) DENGAN MENGGUNAKAN METODE CNN DAN YOLO *(YOU ONLY LOOK ONCE)*

Diajukan Kepada Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN) Ar-Raniry Banda Aceh
Sebagai Salah Satu Beban Studi Memperoleh Gelar Sarjana (S1)
dalam Ilmu/Prodi Teknologi Informasi

Oleh:
Sri Maulida
180705009
Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

Disetujui untuk Dimunaqasyahkan Oleh:

Pembimbing I,	Pembimbing II,
Hendri Ahmadian, S.Si., M.I.M	**Khairan AR, M.Kom**
NIP:198301042014031002	**NIP:198607042014031001**

Mengetahui,
Ketua Program Studi Teknologi Informasi

Ima Dwitawati, MBA
NIP:198210132014032002

LEMBAR PENGESAHAN

ANALISIS AKURASI PADA SIMBOL ABJAD SISTEM ISYARAT BAHASA INDONESIA (SIBI) DENGAN MENGGUNAKAN METODE CNN DAN YOLO (*YOU ONLY LOOK ONCE*)

TUGAS AKHIR/SKRIPSI

Telah Diuji Oleh Panitian Ujian Munaqasyah Tugas Akhir/Skripsi
Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh dan Dinyatakan Lulus
Serta Diterima Sebagai Salah Satu Beban Studi Program Sarjana (S-1)
Dalam Ilmu/Prodi Teknologi Informasi

Pada Hari/Tanggal: Kamis 21 Desember 2023
8 Jumadil Akhir 1445 H
di Darussalam, Banda Aceh

Panitia Ujian Munaqasyah Tugas Akhir/Skripsi:

Ketua,	Sekretaris,
Hendri Ahmadian, S.Si., M.IM	**Khairan AR, M.Kom**
NIP: 198301042014031002	**NIP:198607042014031001**
Penguji I,	Penguji II,
Nazaruddin Ahmad, MT	**Rika Yullana, MT**
NIP: 198206052014031002	**NIP: 198407132014032001**

Mengetahui:
Dekan Fakultas Sains dan Teknologi
UIN AR-Raniry Banda Aceh,

Dr. Ir.M.Dirhamsyah, M.T., IPU
NIP.196210021988111001

LEMBAR PERNYATAAN KEASLIAN TUGAS AKHIR/SKRIPSI

Yang bertanda tangan di bawah ini:

Nama	: **Sri Maulida**
NIM	: 180705009
Program Studi	: Teknologi Informasi
Fakultas	: **Sains dan Teknologi**
Judul	: Analisis Akurasi Pada Simbol ABJAD Sistem Isyarat Bahasa Indonesia (SIBI) dengan Menggunakan Metode CNN dan YOLO (*You Only Look Once*)

Dengan ini menyatakan bahwa dalam penulisan tugas akhir/skripsi ini, saya:

1. Tidak menggunakan ide orang lain tanpa mampu mengembangkan dan mempertanggungjawabkan;
2. Tidak melakukan plagiasi karya orang lain;
3. Tidak menggunakan karya orang lain tanpa menyebutkan sumber asli atau tanpa izin pemilik karya;
4. Tidak memanipulasi dan memalsukan data;
5. Mengerjakan sendiri karya ini dan mampu bertanggungjawab atas karya ini.

Bila dikemudian hari ada tuntutan dari pihak lain atas karya saya, dan telah melalui pembuktian yang dapat dipertanggungjawabkan dan ternyata memang ditemukan bukti bahwa saya telah melanggar pernyataan ini, maka saya siap dikenai sanksi berdasarkan aturan yang berlaku di Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh.

Demikian pernyataan ini saya buat dengan sesungguhnya dan tanpa paksaan dari pihak manapun.

Banda Aceh, 12 Desember 2023
Yang Menyatakan

(Sri Maulida)

ABSTRAK

Nama : Sri Maulida
NIM : 190705009
Program Studi : Teknologi Informasi

Judul : Analisis Akurasi Pada Simbol Abjad Sistem Isyarat Bahasa Indonesia (SIBI) Menggunakan Metode CNN dan YOLO (*You Only Look Once*)

Tanggal Sidang : 21 Desember 2023
Jumlah Halaman : 80
Pembimbing I : Hendri Ahmadian, S.Si., M.I.M
Pembimbing II : Khairan AR, M.Kom
Kata Kunci : SIBI, CNN, SSD *MobileNet* dan YOLO (*You Only Look Once*)

Sistem Isyarat Bahasa Indonesia (SIBI) merupakan sebuah sistem komunikasi yang digunakan oleh para penyandang tunarungu di Indonesia. SIBI menggunakan isyarat tangan, gerakan tubuh serta ekspresi wajah dalam mengkomunikan pesan-pesan tertentu. Dalam era digital saat ini teknologi pengenalan gambar dan pengolahan citra semakin berkembang, salah satunya adalah *Convolutional Neural Network* (CNN) yang merupakan salah satu arsitektur dari algoritma *Deep Learning*, yaitu Teknik pembelajaran mesin yang menggunakan jaringan syaraf tiruan yang terdiri dari banyak *layer*. Salah satu penerapan *deep learning* adalah pada bidang *computer vision* seperti deteksi objek, klasifikasi gambar dan pengenalan wajah. Pada penelitian ini menerapkan CNN dan *pre-trained* model SSD *MobileNet* dan YOLOv7 dalam melakukan deteksi pada simbol abjad SIBI dan menganalisis hasil deteksi yang didapatkan dari pengujian menggunakan *pre-trained* model SSD *MobileNet* dan YOLOv7. *Dataset* yang digunakan berupa simbol abjad SIBI dari A sampai Z dengan jumlah keseluruhan data sebanyak 260 data gambar. Implementasi model menggunakan *Google Colaboratory* dengan Bahasa pemrograman *Python*. Berdasarkan hasil penilitian yang didapatkan dari pengujian model menggunakan SSD *MobileNet* dan YOLOv7 diperoleh bahwa pendeteksian objek menggunakan YOLOv7 menghasilkan akurasi yang lebih tinggi dibandingkan SSD *MobileNet* yaitu 100% akurasi pada YOLOv7 dan 98,07% pada SSD *MobileNet.*

Kata Kunci : SIBI, CNN, SSD *MobileNet* dan YOLO (*You Only Look Once*)

KATA PENGANTAR

بِسْـــــــــــــمِ اللهِ الرَّحْمَنِ الرَّحِيْمِ

Segala puji dan syukur penulis panjatkan kepada Allah SWT Sang Maha Segalanya, atas seluruh curahan rahmat dan hidayahNya sehingga penulis mampu menyelesaikan skripsi yang berjudul **"Analisis Akurasi Pada Simbol Abjad Sistem Isyarat Bahasa Indonesia (SIBI) dengan Menggunakan Metode CNN dan YOLO *(You Only Look Once)".*** Penulisan skripsi ini adalah salah satu syarat untuk mendapatkan gelar sarjana pada Fakultas Sains dan Teknologi di UIN Ar-Raniry Banda Aceh.

Dalam penyusunan skrpsi ini, penulis banyak sekali menghadapi kesulitan dalam Teknik penulisan maupun dalam penguasaan bahan. Walaupun demikian, penulis tidak putus asa dalam menghadapi permasalahan dan dengan adanya dukungan dari berbagai pihak, terutama dosen pembimbing yang telah banyak memberikan arahan dan bimbingan kepada penulis selama proses penulisan skripsi. Pada kesempatan ini, penulis mengucapkan terima kasih yang tak terhingga kepada:

1. Sebagai ungkapan terima kasih, skripsi ini penulis prsembahkan kepada orang tua tercinta Ayahanda Ismail (Alm) dan Ibunda Alina, terima kasih telah berjuang untuk kehidupan penulis, beliau memang tidak sempat merasakan pendidikan sampai bangku perkuliahan, namun beliau mampu mendidik penulis, motivasi, memberikan dukungan serta do'a hingga penulis mampu menyelesaikan studinya sampai sarjana.
2. Kepada cinta kasih kedua saudara-saudara saya Isnawati, S.Pd dan Irwan Is, S.Pd. Terima kasih atas segala do'a, usaha, dan motivasi yang telah diberikan kepada adik terakhir ini.
3. Bapak Hendri Ahmadian, S.Si., M.I.M sebagai pembimbing pertama dan Bapak Khairan AR, M.Kom sebagai pembimbing kedua, yang telah meluangkan waktunya dan mencurahkan pemikirannya dalam membimbing penulis untuk menyelesaikan skripsi ini.
4. Ketua Prodi Teknologi Informasi Ibu Ima Dwitawati, MBA. Sekretaris Prodi Teknologi Informasi Bapak Khairan AR, M.Kom serta staf prodi Ibu

Cut Ida Rahmadiana S,Si. Yang telah membantu penulis dalam proses pelaksaan penelitian dan dalam hal pengurusan administrasi dan surat-surat untuk keperluan penyelesain skripsi ini.

5. Sahabat dan teman-teman penulis Mega Ellyadi, Nura Nabilah, dan Mulisida, serta seluruh keluarga Teknologi Informasi yang telah memberikan dukungan dan semangat dalam penyelesaian skripsi ini.
6. Dan semuanya yang tidak dapat penulis sebutkan satu persatu.

Banda Aceh, 21 Desember 2023
Penulis,

Sri Maulida

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

I.1 Latar Belakang

Sistem Isyarat Bahasa Indonesia (SIBI) merupakan sebuah sistem komunikasi yang digunakan oleh para penyandang tunarungu di Indonesia. SIBI menggunakan isyarat tangan, gerakan tubuh serta ekspresi wajah dalam mengkomunikan pesan-pesan tertentu. Sistem ini sangat penting bagi kumunitas tunarungu di Indonesia, kerena memungkinkan mereka untuk berkomunikasi dengan orang lain dan memperoleh akses informasi yang mereka butuhkan. Namun, meskipun SIBI telah digunakan selama beberapa dekade, masih banyak tantangan yang dihadapi oleh komunitas tunarungu di Indonesia. Salah satu tantangan terbesar adalah kurangnya pemahaman dan kesadaran tentang SIBI di kalangan masyarakat umum. Banyak orang yang tidak tahu bagaimana cara berkomunikasi dengan orang-orang penyandang tunarungu, dan ini dapat menyebabkan isolasi sosial dan kesulitan dalam memperoleh pekerjaan atau Pendidikan.(Rachardi, 2020)

Dalam era digital saat ini teknologi pengenalan gambar dan pengolahan citra semakin berkembang. Salah satunya adalah *Convolutional Neural Network* (CNN) yang merupakan salah satu arsitektur dari algoritma *Deep Learning*. *Deep Learning* adalah Teknik pembelajaran mesin yang menggunakan jaringan saraf tiruan (*Artificial Neural Network*) yang terdiri dari banyak lapisan atau *layer*.salah satu contoh penerapan *deep learning* adalah pada bidang *Computer Vision* seperti deteksi objek, klasifikasi gambar, dan pengenalan wajah. *Deep Learning* dapat digunakan untuk mengenali simbol-simbol abjad pada bahasa isyarat.

Penelitian tentang bahasa isyarat dengan menggunakan metode *deep learning* sebelumnya juga sudah pernah dilakukan oleh beberapa peneliti, seperti penelitian yang dilakukan oleh (Putra, 2021), melakukan pendeteksian simbol SIBI menggunakan *convolutional neural network*. Dalam penelitiannya tersebut menggunakan 6 kelas kata simbol SIBI (kamu, dia, maaf, cinta dan sedih) nilai akurasi yang didapatkan sebesar 90%. Menurut penelitian (Hidayahtullah, 2022),

mendeteksi simbol SIBI secara *Realtime* dengan menggunakan *MobileNet*-SSD, dengan jumlah *dataset* yang digunakan sebanyak 6 kelas dan akurasi tertinggi yang didapatkan adalah sebesar 86,6%.

Kemudian, penelitian oleh (Syahrul, 2021) melakukan rancang bangun penerjemah bahasa isyarat dengan menggunakan pengolahan citra. Pada penelitiannya tersebut metode yang digunakan adalah *You Only Look Once* (YOLO) dengan melakukan pendeteksian gerakan bahasa isyarat secara *realtime.* Proporsi keberhasilan yang didapatkan sebesar 90%, dengan akurasi 94%, *presisi* 99,9% dan *recall* 100%. Selanjutnya penelitian oleh (Dadang Iskandar, 2022) mendeteksi Bahasa isyarat dalam pengenalan huruf hijaiyah dengan menggunakan metode YOLO, dan akurasi yang dihasilkan cukup tinggi yaitu sebesar 95%. Dan penelitian oleh (Sani & Rahmadinni, 2022), melakukan pendeteksian pada gestur tangan berbasis pengolahan citra. Pada penelitiannya tersebut metode yang digunakan adalah YOLO-V3 yang mampu mendeteksi dan mengklasifikasi objek sekaligus tanpa dipengaruhi olen intensitas cahaya dan *background* dari objek, dengan tingkat akurasi diatas 90%.

Pada penelitian sebelumnya telah menunjukkan bahwa penggunaan metode CNN dengan arsitektur SSD *MobileNet* dan YOLO memberikan hasil yang sangat baik dalam tugas deteksi objek, dengan kecepatan dan efisien yang tinggi. Oleh karena itu pada penelitian ini akan dilakukan analisis terhadap akurasi pada simbol abjad Sistem Isyarat Bahasa Indonesia (SIBI) menggunakan metode CNN dengan arsitektur SSD *MobileNet* dan *You Only Look Once* (YOLO).

Fokus penelitian ini adalah melakukan pengenalan simbol abjad SIBI pada *computer* dengan menggunakan *pre-trained* model SSD M*obileNe*t dan YOLOv7. Kemudian akan diuji performa kedua model dengan menggunakan *dataset* yang telah diannotasi dengan benar dan dilakukan analisis terhadap hasil deteksi dari pengujian model SSD *MobileNet* dan YOLOv7 Dan *dataset* yang akan dipakai untuk pelatihan dan pengujian model adalah data yang diambil secara mandiri berdasarkan sistem isyarat Bahasa Indonesia (SIBI) yaitu huruf A sampai Z dengan jumlah seluruh *dataset* sebanyak 260 gambar dari 26 kelas.

I.2 Rumusan Masalah

Rumusan masalah yang menjadi dasar pemikiran pada penelitian ini adalah:

1. Bagaimana proses dari pendeteksian simbol abjad SIBI pada metode SSD *MobileNet* dan YOLO?
2. Bagaimana akurasi dari pendeteksian simbol SIBI menggunakan SSD *MobileNet* dan YOLO?

I.3 I.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah:

1. Mengetahui proses deteksi simbol SIBI pada SSD *MobileNet* dan YOLO
2. Mengetahui akurasi dari algoritma pendeteksian objek pada SSD *MobileNet* dan YOLO.

I.4 Batasan Masalah

Batasan pada penelitian ini adalah:

1. Data yang digunakan adalah data gambar dari gestur tangan simbol abjad SIBI
2. *Training* model menggunakan SSD *MobileNet* dan YOLO-V7
3. Menggunakan bahasa pemrograman *Python*.
4. Model diuji di *Google Collab*
5. Pelatihan model dilakukan di *Epoch* 1500
6. *Dataset* gambar diambil menggunkan kamera *Smartphone* Samsung A20s

I.5 Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah:

1. Dapat menambah pengetahuan mengenai deteksi SIBI menggunakan metode SSD *MobileNet* dan YOLOv7.
2. Mengetahui tingkat akurasi dari implementasian metode *deep learning* dengan *training* model SSD *MobileNet* dan YOLOv7
3. Dapat dijadikan acuan untuk pengukuran berbagai algoritma lainnya.

4. Dapat menjadi referensi bagi mahasiswa yang ingin melakukan penelitian mengenai metode SSD *MobileNet* dan YOLOv7.

BAB II
TINJAUAN PUSTAKA

II.1 Penelitian Terdahulu

Terkait dengan penelitian yang penulis lakukan menggunakan model pembelajaran *deep learning*, dibutuhkan referensi atau penelitian terkait guna untuk terhindar dari duplikasi dan plagiarisme, sehingga penulis dapat mengembangkan sesuatu hal yang berbeda pada penelitian ini. Berikut ini adalah beberapa penelitian terkait yang berhubungan dengan penelitian penulis.

Penelitian yang dilakukan oleh (Putra, 2021) dengan judul *"*Sistem Deteksi Simbol Pada SIBI (Sistem Isyarat Bahasa Indonesia) Menggunakan Metode *Convolutional Neural Network"*. Pada penelitian tersebut menggunakan 6 simbol bahasa isyarat, yaitu dengan kata Saya, Kamu, Dia, Maaf, Sedih dan Cinta. Hasil yang diperoleh dari pengujian *training* pada penelitian tersebut sebesar 90%, dan *validatio*n data yang diperoleh sebesar 91%. Pada prediksi diperoleh akurasi sebesar 80% pada simbol "Cinta", 70% pada simbol "Dia" dan "Saya", 90% pada simbol "Kamu", 100% pada simbol "Sedih" dan 0% pada simbol "Maaf".

Penelitian yang dilakukan (Hidayahtullah, 2022) tentang *"*Sistem Deteksi Simbol pada SIBI (Sistem Isyarat Bahasa Indonesia) Secara *Realtime* Menggunakan *Mobilenet*-SSD*"*. Pada penelitian tersebut menggunakan 6 kelas kata dari simbol SIBI yaitu Saya, Dia, Kamu, Cinta, Maaf dan Sedih. Proses *training* model yang digunakan pada penelitian tersebut terdapat 3 *step training* yaitu 5.000, 10.000 dan 20.000. Hasil *training* model dengan 5.000 *step* mempunyai nilai *loss* sebesar 0,2712 dan akurasinya 83,3%, sedangkan pada proses *training* 10.000 *step* mendapatkan nilai *loss* 0,1881 dan akurasinya 83,6%, kemudian pada proses *training* 20.000 *step* memperoleh nilai *loss* 0,1387 dengan akurasinya 86,6%.

Penelitian yang dilakukan (Syahrul, 2021) tentang "Rancang bangun penerjemah Bahasa isyarat menggunakan pengolahan citra dengan metode *You Only Look Once* (YOLO)". Penelitian tersebut melakukan pendeteksian pada

Bahasa isyarat secara *realtime* dan menghasilkan klasifikasi bahasa isyarat yang ditampilkan pada *interface* dengan *output* suara dan teks dari perangkat. Dari 20 kata percobaan, terdapat 19 kata yang terdeteksi secara sempurna, proporsi keberhasilan yang didapatkan sebesar 90%, akurasi 94%, presisi 99,9, dan *recall* 100%.

Penelitian yang dilakukan oleh (Rachardi, 2020) tentang "Deteksi gambar gestur kosakata Bahasa Isyarat Indonesia dengan *Convolutional Neural Network*". Pada penelitian tersebut menggunakan jenis Bahasa isyarat indonsia (Bisindo) dengan menggunakan arsitektur SSD *MobileNet* dalam pelatihan model. Objek yang digunakan adalah 32 kosakata isyarat dari lirik lagu "Bidadari tak bersayap" dengan total dataset berjumlah 17.600 citra, dan akurasi yang dihasilkan sebesar 93,75% dan 75%.

Penelitian yang dilakukan (Sani & Rahmadinni, 2022) tentang "Deteksi gestur tangan berbasis pengolahan citra". Pada penelitian tersebut dataset yang digunkan berupa huruf dan kosakata (satu, dua, tiga, empat, *hello, yes, no, ok, call*, dan diam) dengan jumlah kelas data sebanyak 10 kelas data. Pelatihan model dilakukan menggunakan metode Yolo-v3 yang mampu mendeteksi dan mengklasifikasi objek sekaligus tanpa dipengaruhi oleh intensitas cahaya dan *background* dari objek, tingkat akurasi yang dihasilkan diatas 90%.

Penelitian yang dilakukan (Mulyana et al., 2022) mengenai "Deteksi Bahasa isyarat dalam pengenalan huruf hijaiyah dengan metode YOLOv5". Penelitian tersebut melakukan pelatihan model dengan menggunaka metode YOLOv5, dan dataset yang digunakan adalah huruf hijaiyah (alif sampai ya) dengan jumlah dataset yang digunakan adalah 114 gambar huruf hijaiyah. Adapun nilai akurasi yang didapatkan pada penelitian tersebut cukup tinggi yaitu 95%.

Tabel II. 1 Perbandingan Penelitian Sejenis

Peneliti	Kasus	Model	Dataset	Hasil Penelitian
(Putra, 2021)	Mendeteksi simbol pada SIBI (Sistem isyarat Bahasa Indonesia)	CNN	Kosakata bahasa isyarat (kamu,saya,dia, maaf,sedih,dan cinta).	Menghasilkan nilai akurasi training sebesar 91% dan validasi menghasilkan nilai akurasi sebesar 90%.
(Hidayahtullah, 2022)	Mendeteksi simbol SIBI secara *Realtime*	SSD *MobileNet*	Kosakata Bahasa isyarat (kamu, saya, dia, maaf, sedih, dan cinta) dengan jumlah kelas data sebanyak 6 kelas.	Penelitian dilakukan menggunakan 3 steps: 1. Steps 5000 mengahsilkan akurasi sebesar 83,3% 2. Steps 10.000 menghasilkan akurasi sebesar 83,6% 3. Steps 20.000 menghasil

Peneliti	Kasus	Model	Dataset	Hasil Penelitian
				kan akurasi sebesar 86,6%.
(Syahrul, 2021)	Melakukan rancang bangun penerjemah Bahasa isyarat menggunakan pengolahan citra	YOLO	Menggunakan kosakat Bahasa isyarat Indonesia (BISINDO) dengan jumlah kelas data yang digunakan adalah 20 kelas.	Menghasilkan klasifikasi Bahasa isyarat yang ditampilkan pada interface dengan output berupa suara dan teks. Dari 20 kata percobaan hanya 19 kata yang terdeteksi secara sempurna dengan nilai akurasi sebesar 94%.
(Rachardi, 2020)	Mendeteksi gambar dari gestur kosakata Bahasa isyarat Indonesia dengan *convolutional neural network*	SSD *MobileNet*	Mengggunakan jenis Bahasa isyarat Bisindo dengan jumlah kosakata sebanyak 32 kosakata isyarat dari lirik lagu "Bidadari tak bersayap" yang berjumlah	Model dapat mendeteksi objek kosakata Bahasa isyarat dengan hasil akurasi yang didapatkan sebesar 93,75% dan 75%.

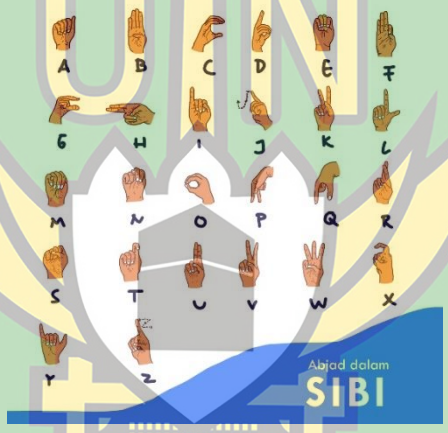
Peneliti	Kasus	Model	Dataset	Hasil Penelitian
			sebanyak 17.600 citra	
(Mulyana et al., 2022)	Mendeteksi Bahasa isyarat dalam pengenalan huruf hijaiyah	YOLO	Menggunakan dataset huruf hijaiyah (alif sampai ya) dengan total jumlah dataset yang digunakan sebanyak 114 gambar huruf hijaiyah.	Menghasilkan nilai akurasi yang cukup tinggi yaitu 95%.
(Sani & Rahmadinni, 2022)	Mendeteksi gestur tangan yang berbasis pengolahan citra	YOLO	Menggunakan dataset huruf dan kosakata (satu,dua,tiga,empat, hello,yes,no,ok,call ,dan diam) yang berjulah 10 kelas data.	Model mampu mendeteksi dan mengklasifikasiobjek Bahasa isyarat tanpa dipengaruhi intensitas cahaya dan *background* dari objek dan tingkat akurasi yang dihasilkan diatas 90%.

Sumber:Penulis

II.2 Sistem Isyarat Bahasa Indonesia (SIBI)

SIBI dengan kepanjangan Sistem Isyarat Bahasa Indonesia memiliki artian sebuah bahasa bagi seorang penderita tunarungu dan tunawicara yang digunakan untuk berkomunikasi. SIBI merupakan bahasa yang diperuntukkan kepada para penyandang tunarungu dan tunawicara. Tidak hanya dalam komunikasi sehari-

sehati SIBI juga telah digunakan dalam lingkungan Pendidikan yaitu sebagai bahasa isyarat formal. SIBI merupakan bahasa yang diadopsi dari bahasa isyarat luar negeri yang kemudian dicampur dengan bahasa isyarat lokal. Bahasa isyarat SIBI sudah dibakukan dan diresmikan oleh pemerintah untuk dipergunakan oleh para penyandang tunarungu dan tunawicara serta menciptakan kamus SIBI sebagai media pembelajaran. Bahasa isyarat SIBI memiliki aturan isyarat yang terstruktur yaitu dengan gerakan tangan, jari, dan bahasa tubuh (Simabura, 2019). Pada penerapannya SIBI mempunyai individual bahasa isyarat untuk abjad sebagaiamana diperhatikan pada gambar II.1.



Gambar II. 1 Huruf Abjad dalam SIBI (Putra, 2021)

II.2.1 Elemen Penentu Makna

a. Penampil, yakni bagian tangan yang dipergunakan buat menghasilkan isyarat, antara lain:
    1. Tangan kana, tangan kiri maupun kedua tangan.
    2. Telapak tangan dengan jari membuka, menggenggam maupun sebagian jari mencuat.
    3. Pose jari tangan membentuk huruf A, B, C maupun huruf lain.
    4. Jari-jari tangan merapat maupun renggang.
    5. Pose jari tangan membentuk angka 1, 2, 3 ataupun angka lain.

b. Posisi, yakni peran tangan terhadap pengisyarat saat berisyarat, antara lain:
    1. Tangan kanan ataupun kiri tegak, condong, mendatar, menghadap ke kanan, ke kiri, ke depan ataupun menyerong.
    2. Telapak tangan kanan ataupun kiri telentang, telengkung menghadap ke kanan, ke kiri, ke depan, ke pengisyarat.
    3. Kedua tangan berdampingan, berjajar, bersilang ataupun bersusun.
c. Tempat, yakni anggota tubuh yang menjadi tempat awal isyarat dibentuk ataupun arah akhir isyarat, antara lain:
    1. Kepala dengan seluruh bagiannya, semacam pelipis, dahi serta dagu.
    2. Leher.
    3. Dada kanan, kiri, tengah.
    4. Tangan.
d. Arah, yakni gerak penampil saat isyarat terbuat, antara lain:
    1. Menjahui ataupun mendekati pengisyarat.
    2. Ke samping kanan, kiri, ataupun bolak balik.
    3. Lurus, melengkung.
e. Frekuensi yakni jumlah gerak yang dilakukan pada waktu isyarat dibentuk. terdapat isyarat yang frekuensinya Cuma sekali, terdapat yang 2 kali ataupun lebih, ataupun terdapat pula gerakan kecil yang diulang-ulang.

II.2.2 Elemen Penunjang

a. Ekspresi wajah, menyampaikan maksud tambahan terhadap pesan isyarat yang disampaikan. Biasanya menyimbolkan intensitas atau keseriusan pesan yang disampaiakan. Contohnya pada saat mengisyaratkan rasa Bahagia, duka, ataupun marah.
b. Gerak batang tubuh misalnya bahu, menyampaikan kesan tambahan atas pesan, misalnya isyarat tidak tahu, ditambah naiknya kedua bahu diartikan *benar-benar tidak tahu atau tidak tahu sedikit pun.*
c. Kecepatan gerak berguna sebagai penambah pendalaman makna. Isyarat pergi yang dilakukan dengan cepat, dapat diartikan *pergilah dengan segera.*

d. Kelenturan gerak menandai keseriusan maksud isyarat yang disampaikan. Isyarat marah yang dilakukan dengan kaku bias didefinisikan sebagai *marah sekali*. Begitu juga isyarat berat yang dilakukan dengan kaku dapat ditafsir *berat sekali.*

II.3 *Deep Learning*

Deep learning adalah metode yang memanfaatkan *Artificial Neural Network* atau jaringan saraf tiruan, dimana *Artificial Neural Network* dibuat mirip dengan otak manusia. *Deep learning* pada komputer melakukan pembelajaran dengan cara mengklarifikasi langsung pada objek, seperti Gambar, suara, teks dan yang lainnya. Model *Deep learning* ini dapat menghasilkan akurat yang tinggi melebihi kinerja manusia. Karena model ini dilatih dengan menggunakan dataset berlabel dengan jumlah yang banyak dan *Neural Network* dapat melakukan penyelesaian masalah secara akurat dan otomatis. Metode *Deep learning* sebagian besarnya menggunakan *Neural Network Architecture*, oleh karena itu *Deep learning* sering disebut dengan *Deep Neural Network*. Dari istilah *"deep"* dapat mengacu pada jumlah lapisan yang tersembunyi di *Neural Network* atau dengan kata lain *hidden layer*. Pada *Neural Network* tradisional hanya mengandung 2-3 lapisan saja, namun berbeda hal dengan *Deep Neural Network* dapat memiliki 150 lapisan (Yakib, 2020).

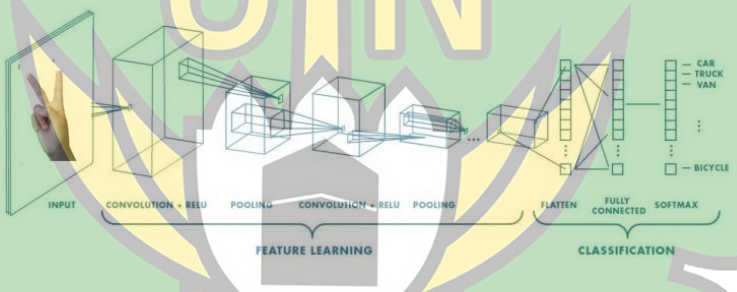
Deep learning melakukan model komputasi yang terdiri dari beberapa lapisan pengolahan agar dapat mempelajari represntasi data dengan beberapa tingkatan abstraksi. Metode ini secara signifikan dapat meningkatkan pengembangan dalam berbagai bidang seperti pengenalan suara (*voice recognition*), pengenalan *object* visual (*image recognition*), deteksi objek (*object detection*). *Deep learning* saat ini temasuk masuk kedalam terobosan berbagai bidang pada *artificial intelligence* (Manajang et al., 2020).

II.4 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan pengembangan dari *Multilayer Perceptron* (MLP) yang dibuat sebagai memperoses data. CNN merupakan jenis dari *Deep Neural Network* dikarenakan memiliki dasar jaringan

yang tinggi hingga sering digunakan pada citra (Pamungkas, 2020). CNN merupakan algoritma yang populer saat ini dalam *Deep learning* termasuk kedalam bagian *machine learning* yang mana model untuk melakukan klasifikasi pada gambar, tesk, suara dan video. Biasanya CNN digunakan untuk dapat melakukan mengenali pola pada wajah atau *object* yang lain (Yakib, 2020).

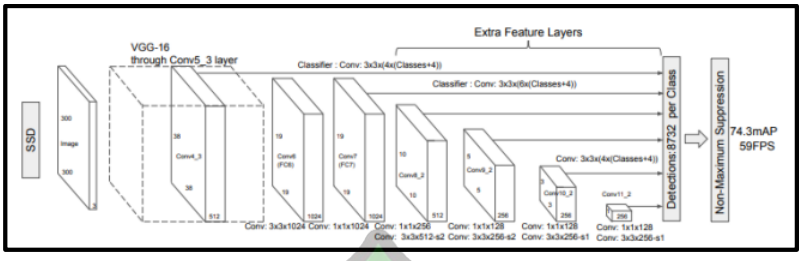
Terdapat dua Arsitektur yang digunakan pada CNN, yaitu *Featur Learning* dan *Classification.* Pada satu *layer feature learning* terdapat *Convolution, Relu* (sebagai *Activation Function*), setelah itu *Max Pooling*. kemudian pada *Classification* berisi *Flatten, Fully Connected,* dengan fungsi aktivasi *Softmax* untuk melakukan pengelompokkan (Putra, 2021).



***Gambar II. 2** Arsitektur* CNN (Putra, 2021)

II.5 *Single Shot Multibox Detector* (SSD)

Proses yang ditempuh oleh metode *Single Shot Multibox Detector* (SSD) yaitu melakukan pencocokan pada sebuah objek dengan *default bounding box* atau rasio maupun skala pada setiap dari *feature map location*. Pada saat pendeteksian objek, SSD melakukan perbandingan objek mengunakan *default bounding box* dengan rasio ketika proses *training.* Untuk menghasilkan hasil terbaik pada objek yang dideteksi metode SSD memakai beberapa *layer* pada berbagai macam skala (Dompeipin et al., 2021). SSD menerapkan model VGG-16 menjadi jaringan dasar dalam pembuatanya, adapun arsitektur SSD dapat diperhatikan pada gambar II.3.



Gambar II.3 Arsitektur SSD (Sutama et al., 2020)

VGG-16 merupakan singkatan dari "*Visual Geometery Group-16 Network",* VGG-16 termasuk kedalam arsitektur jaringan CNN, yang dirancang oleh sebuah grup dalam kompetisi *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) yang terjadi pada tahun 2014 untuk dilakukannya lokalisasi dan klasifikasi pada sebuah *image*. Arsitektur CNN terdiri dari 16 *layer* yaitu: (Mashita, 2020).

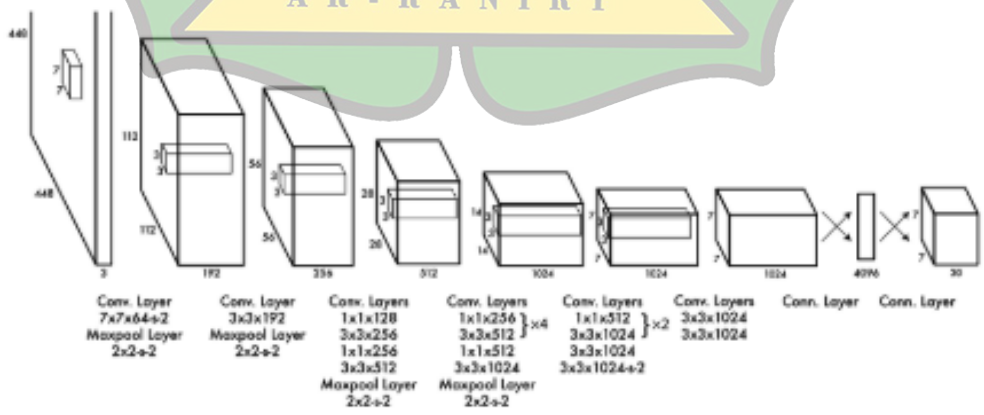
1. Konvolusi menggunakan 64 *filter.*
2. Konvolusi menggunakan 64 *filter* + *Max pooling.*
3. Konvolusi menggunakan 128 *filter.*
4. Konvolusi menggunakan 128 *filter* + *Max pooling.*
5. Konvolusi menggunakan 256 *filter.*
6. Konvolusi menggunakan 256 *filter.*
7. Konvolusi menggunakan 256 *filter* + *Max pooling.*
8. Konvolusi menggunakan 512 *filter.*
9. Konvolusi menggunakan 512 *filter.*
10. Konvolusi menggunakan 512 *filter* + *Max pooling.*
11. Konvolusi menggunakan 512 *filter.*
12. Konvolusi menggunakan 512 *filter.*
13. Konvolusi menggunakan 512 *filter* + *Max pooling.*
14. Terhubung sepenuhnya dengan 4096 *node*.
15. Terhubung sepenuhnya dengan 4096 *node*.
16. Lapisan keluaran dengan aktivasi *Softmax.*

II.6 *MobileNet*

Mobilenet adalah salah satu metode dari bagian arsitektur *Convolutional Neural Network* yang dapat memenuhi keperluan komputasi yang berlebih. *Mobilenet* dibangun oleh para arsitektur *google* atas dasar memenuhi kebutuhan CNN dalam membangun arsitektur yang dapat digunakan pada sistem *mobile*. Perbedaan yang cukup mendasar di antara arsitektur *Mobilenet* dengan CNN adalah saat menggunakan *layer konvolusi* yang ketebalan *filter*nya dibuat sesuai dengan ketebalan input citra yang ada. *Mobilenet* dirancang di atas arsitektur jaringan yang efisien dengan menggunakan konvolusi yang dapat dipisahkan secara mendalam untuk menghasilkan *Deep Neural Network* yang ringan (Sindy, 2019).

II.7 *You Only Look Once* (YOLO)

You Only Look Once (YOLO) merupakan pengembangan dari metode CNN (Convolutional Neural Network). YOLO dikenalkan secara *open source* oleh *University of Washington* untuk memprediksi *bounding boxes* dan klasifikasinya dari suatu gambar. Metode YOLO dikembangkan karena terinspirasi dari model *GoogleNet* untuk klasifikasi gambar. Metode YOLO memiliki 24 layer konvolusi yang diikuti dengan 2 layer yang terkoneksi (Budiyanta et al., 2021) Arsitektur dari metode YOLO dapat dilihat pada Gambar II.4 berikut.

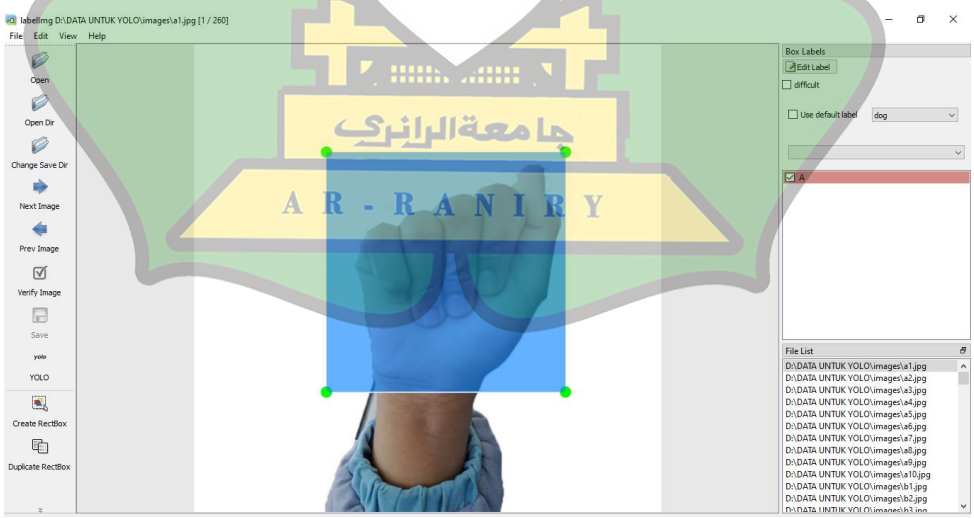


Gambar II.4 Arsitektur Metode YOLO

YOLO sendiri telah dikembangkan hingga beberapa versi, salah satunya adalah YOLOv7. YOLOv7 merupakan seri atau generasi ke-7 dari seri YOLO yang telah dikembangkan sebelumnya. YOLOv7 merupakan algoritma deteksi objek real-time yang dikembangkan pada tahun 2022. Sebagai objek detector YOLOv7 mampu mendeteksi objek dnegan cepat, presisi yang tingi, serta mudah untuk dilatih dan diterapkan. YLOv7 120% lebih cepat jika dibandingkan dengan YOLOv5. Kecepatan dan akurasi deteksi objek YOLOv7 semakin meningkat berdasarkan versi sebelumnya (Passa et al., 2023).

II.8 *LabelImg*

LabelImg merupakan program yang berfungsi untuk pelabelan gambar. *LabelImg* merupakan program *open source* yang *free* digunakan. *LabelImg* dirancang dengan menggunakan bahasa *pemrograman python* 3 dan Qt untuk *user interface*, sehingga diperluhkan *instalasi python 3* dan *library* PyQt di komputer untuk menjalankannya. *LabelImg* ditunjukan pada Gambar II.5 berikut: (Budiarjo, 2020)



Gambar II.5 *Labelimg*

II.9 *Google Colaboratory*

Google Colaboratory atau *google colab* merupakan *platform cloud computing* yang mirip dengan *Jupyter Notebook* dan *Google Research*. *Google colab* memungkinkan pengguna untuk menulis dan mengeksekusi kode program *python* secara acak hanya dengan menggunakan *browser web*. *Google colab* adalah sebuah IDE untuk pemrograman *Python* dimana pemrosesan akan dilakukan oleh *server Google* yang memiliki perangkat keras dengan performa yang tinggi . Dari sisi perangkat lunak, *Google Colab* telah menyediakan hampir sebagian besar pustaka *(library)* yang dibutuhkan. Dari sisi perangkat keras, *Google Colab* menyediakan layanan berupa media penyimpan yang terintegrasi dengan *Google Drive*, prosesor yang berupa CPU, GPU, dan TPU, serta RAM. Dengan jaminan kemampuan servernya yang stabil hampir keseluruhan pemrosesan tidak menemukan kendala dengan *Google Colab* selama koneksi jaringan internet lancar. Saat menggunakan *google colab* pengguna tidak perlu menginstal, menjalankan atau meng-*upgrade hardware* komputer untuk memenuhi persyaratan beban kerja intensif CPU/GPU *Python.*

II.10 *Tensorflow Object Detection* API

Tensorflow merupakan salah satu *framework* dari *Deep Learning* ynag besifat *free open source. Tensorflow Object Detection* API adalah pustaka perangkat luak yang dikembangkan oleh tim *Tensorflow* untuk memudahkan pengembang sistem deteksi objek menggunakan *framework tensorflow* dengan menyediakan berbagai algoritma dan fungsi yang siap pakai untuk membangun, melatih dan menerapkan model deteksi objek dengan mudah(Manajang et al., 2020).

Tensorflow telah banyak digunakan dalam pendeteksian objek seperti deteksi wajah, gambar, plat nomor kendaraan dan yang lainnya (Yakib, 2020). *Tensorflow* juga mendukung berbagai bahasa pemrograman sehingga mempermudah *developer* dalam proses pembelajaran mesin. Pengguna dapat membuat model *machine learning* sendiri sesuai dengan kemauan dan kebutuhan pengguna (Muliadi, 2020).

II.11 *Python*

Python merupakan Bahasa pemrograman yang popular dan mudah untuk dipelajari. Python banyak digunakan dalam pengembangan perangkat lunak, kecerdasan buatan, pengembangan web, machine learning, dan analisis data. Selain itu, Bahasa pemrograman python memiliki library yang bervariasi yang memiliki kegunaannya masing-masing yang bersifat open source. Python juga mudah diintegrasikan dengan teknologi lain, seperti database, big data tools, framework web, dan lain sebagainya yang berhubungan dengan analisis data dengan tujuan untuk mengakses dan mengelola data dari berbagai sumber (T I Sambi Ua et al., 2023)

II.12 Model Evaluasi

Pada penelitian ini akan menggunakan model evaluasi *Confusion Matrix*. Evaluasi merupakan kesimpulan akhir dari semua proses yang telah dilakukan atau *output* yang dikeluarkan dari proses yang telah dilakukan, adapun metode yang digunakan yaitu *Confusion Matrix* untuk mengetahui keakuratan pada proses pendeteksian huruf abjad Sistem Isyarat Bahasa Indonesia (SIBI).

Confusion Matrix memiliki nilai atau persamaan untuk mendapatkan nilai keakuratan *system* tersebut. Maka proses dilakukan *Confusion Matrix* sangat penting, sehingga *system* yang telah dibuat memiliki *system* yang baik untuk proses pendeteksian.

II.12.1 Confusion Matrix

Confusion matrix biasanya digunakan untuk menghitung keakuratan pada sebuah objek deteksi. Adapun *Confusion matrix* ini terdapat 4 istilah sebagai hasil dari proses pendeteksian objek. Istilah tersebut yaitu *True Positive (TP), False Positive (FP), False Negative (FN)* dan *True Negative (TN)* (Sindy, 2019).

Keterangan:

1. *True Positive (TP)* yaitu jumlah data yang diprediksi benar positif oleh model.

2. *False Positive (FP)* yaitu jumlah data yang diprediksi salah positif oleh model.
3. *False Negative (FN)* yaitu jumlah data yang diprediksi salah negatif oleh model.
4. *True Negative (TN)* yaitu jumlah data yang diprediksi benar negatif oleh model.

Menurut (Manajang et al., 2020) *confusion matrix*, data ditentukan 3 nilai yaitu:

1. *Accuracy*, merupakan hasil perhitungan tingkat keakuratan deteksi objek terhadap objek SIBI secara keseluruhan. Adapun persamaan *accuracy* dapat dilihat dipersamaan II.1 (Manajang et al., 2020).

   $$Accuracy = \frac{Jumlah\ Objek\ Yang\ Terdeteksi\ Benar}{Jumlah\ Keseluruhan\ Objek\ Yang\ Terdeteksi} \text{ x } 100\%$$ Persamaan (II.1)

2. *Precission*, merupakan persamaan mengenai jumlah prediksi yang benar dibandingkan dari keseluruhan hasil yang dapat di prediki oleh sistem. Pada persamaan ini akan menghasilkan jumlah objek yang benar dari keseluruhan yang di deteksi oleh sistem. Adapun persamaan *precission* dapat dilihat dipersamaan II.2 (Manajang et al., 2020).

   $$Precision = \frac{TP}{(TP + FP)} \text{ x } 100\%$$ Persamaan (II.2)

3. *Recall*, merupakan jumlah prediksi yang benar dibandingkan dengan keseluruhan hasil pendeteksian objek sebenarnya. Persamaan ini akan menghasilkan jumlah objek deteksi dengan benar dari keseluruhan jumlah objek yang terdeteksi oleh sistem. Adapun persamaan *recall* dapat dilihat dipersamaan II.3 (Manajang et al., 2020).

   $$Recall = \frac{TP}{(TP + FN)} \text{ x } 100\%$$ Persamaan (II.3)

BAB III
METODE PENELITIAN

Pada proses penelitian, peneliti membuat kerangka kerja yang berbentuk skema agar memudahkan peneliti dalam melakukan penelitian sehingga skema tersebut peneliti jadikan panduan dalam melakukan penelitian. Adapun tahapan penelitian yang disusun sebagai berikut:

III.1 Metode Pengumpulan Data

III.1.1 Studi Pustaka

Studi pustaka merupakan salah satu metode yang digunakan dalam proses pengumpulan data yaitu dengan membaca, mengola informasi, menulis catatan penting yang bersumber dari buku-buku pustaka dan mengumpulkan informasi yang mendukung dan berubungan dengan penelitian. Adapun referensi yang dikumpulkan berupa data dan informasi dari beberapa situs *online* terpercaya dan jurnal yang berhubungan dengan penelitian. Penulisan tersebut digunakan untuk menyelesaikan penulisan pada bab pendahuluan, landasan teori dan metode penelitian.

III.1.2 Observasi

Proses observasi yang peneliti lakukan adalah pada bagian pengumpulan *dataset* dengan cara mengamati dari setiap peragaan simbol abjad SIBI dari video pada *website* PMPK Kementerian Pendidikan dan Kebudayaan https://pmpk.kemdikbud.go.id/sibi/ dan gambar dalam abjad SIBI.

III.2 Metode Simulasi

III.2.1 Menentukan Perumusan Permasalahan

Identifikasi masalah yang dilakukan oleh peneliti berdasarkan beberapa referensi yang telah dibaca mengenai pendeteksian simbol SIBI belum ada yang membahas pengujian mengenai deteksi simbol abjad SIBI dengan menggunakan

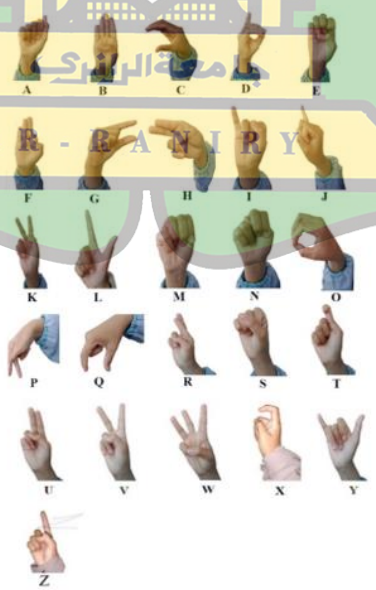
metode *Deep Learning* menggunakan *training* model SSD *MobileNet* dan YOLOv7.

III.2.2 Konsep Penelitian

Pada penelitian ini menggunakan metode *deep learning* untuk mendeteksi simbol abjad SIBI yaitu menggunakan model *training* SSD *MobileNet* dan YOLOv7. Pada prose pengujian sistem ini nantinya terdapat proses *training* pada *dataset* gambar, melakuakn pengujian untuk dapat mengenali simbol abajd SIBI dan mendapatkan nilai akurasi dari kedua model *training.* Hasil *output* yang didapatkan dari hasil pengujian sistem berupa gambar yang telah di *input* dan dikenali oleh sistem.

III.2.3 Pengumpulan Data

Data yang dikumpulkan berupa gambar dari simbol abjad SIBI (A-Z) dengan jumlah kelas sebanyak 26 kelas data. Data tersebut diambil menggunakan kamera *Smartphone Samsung* A20s. Pada tiap kelas data terdapat 10 gambar dengan total *dataset* sebanyak 260 gambar. Simbol abjad SIBI dalam dilihat pada III.1 dibawah ini.



Gambar III. 1 Simbol Abjad SIBI

III.2.4 Labeling Data

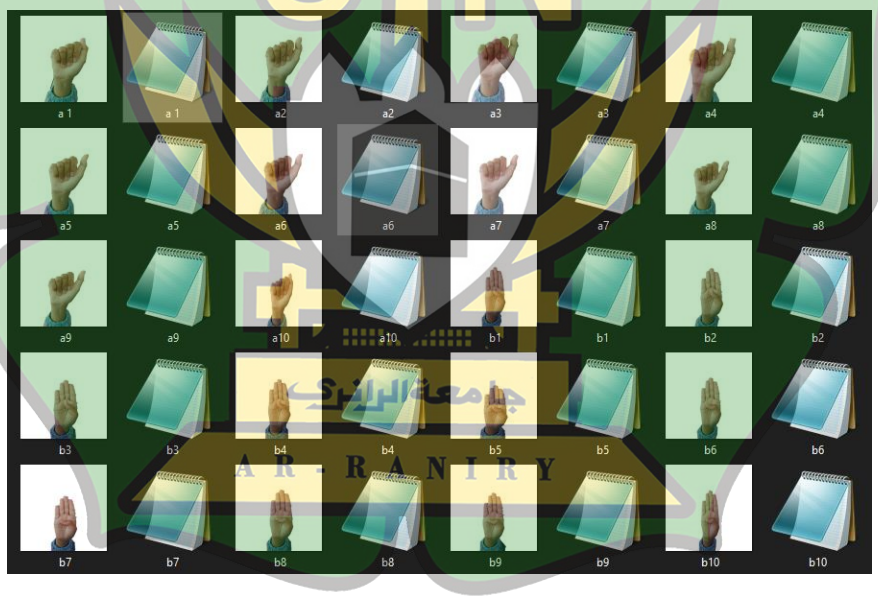
Data yang sudah dikumpulkan selanjutnya akan dilakukan labeling. Proses ini dilakukan untuk memberi label pada setiap simbol sehingga model *training* mengetahui nama simbol yang akan di prediksi. Pada proses pelabelan ini peneliti menggunakan *annotation tool* untuk memberi lokasi prediksi tiap simbol *class* seperti pada Gambar III.3, *annotation tool* yang digunakan adalah "LabelImg*".*

Dataset sebanyak 260 gambar yang telah dikumpulkan oleh peneliti dilakukan proses manual memberi kotak prediksi pada objek simbol dan memberi label satu per satu supaya hasil *training* yang didapatkan nanti memiliki hasil akurasi yang akurat.

Gambar III. 2 *Labeling* pada *LabelImg*

Setelah memberi kotak prediksi sebanyak 260 gambar maka akan didapatkan *file* XML yang digunakan untuk melatih model SSD *MobileNet* dan file txt digunakan untuk melatih model YOLO, isi *file* XML dan txt tersebut merupakan koordinat lokasi *box* yang telah diberi oleh peneliti pada gambar. Total data pada setiap gambar yang telah dilakukan pelabelan terdapat 520 *file dataset* yang terdiri dari 260 file *image* dengan format JPG dan 260 *file* XML dengan format PASCAL VOC dan 520 *file dataset* yang terdiri dari 260 *file image* dengan format JPG dan 260 *file* txt dengan format YOLO .

Gambar III. 3 *Dataset* dengan *file* XML



Gambar III. 4 *Dataset* dengan *file* txt

Data yang sudah deberi label, kemudian akan dibagi menjadi data *train* dan *test* dengan rasio pembagian 80% data *train* dan 20% data *test.*

III.2.5 Pemodelan Algoritma

Pada tahap ini melakukan pembuatan pengujian pada *dataset* simbol abjad SIBI untuk mengenali pola gambar abjad SIBI dan proses deteksi menggunakan model *training* SSD *MobileNet* dan YOLOv7 agar sistem dapat mendeteksi simbol pada abjad SIBI.

III.2.6 Training dan Pengujian

Training atau pelatihan merupakan proses pembelajaran jaringan agar jaringan mampu melakukan tugas yang diperintahkan. Data yang sudah dikumpulkan akan di *training* menggunakan *Google Colab* sampai selesai. Tahapan ini menggunakan metode CNN dengan *training* model SSD *MobileNet* dan YOLOv7 dengan menggunakan proses *Transfer Learning* dan *dataset* yang sudah diberi label. Setelah data di *training* maka data *train* akan di uji untuk melihat apakah sistem mampu mendeteksi simbol.

III.2.7 Menghubungkan ke API

Tahap selanjutnya yaitu menghubungkan model pengujian system dengan *interface* berupa API agar dapat melakukan proses pendeteksian dengan tampilan *interface* yag ada. *Framework* yang peneliti gunakan yaitu *Tensorflow Object Detection* API.

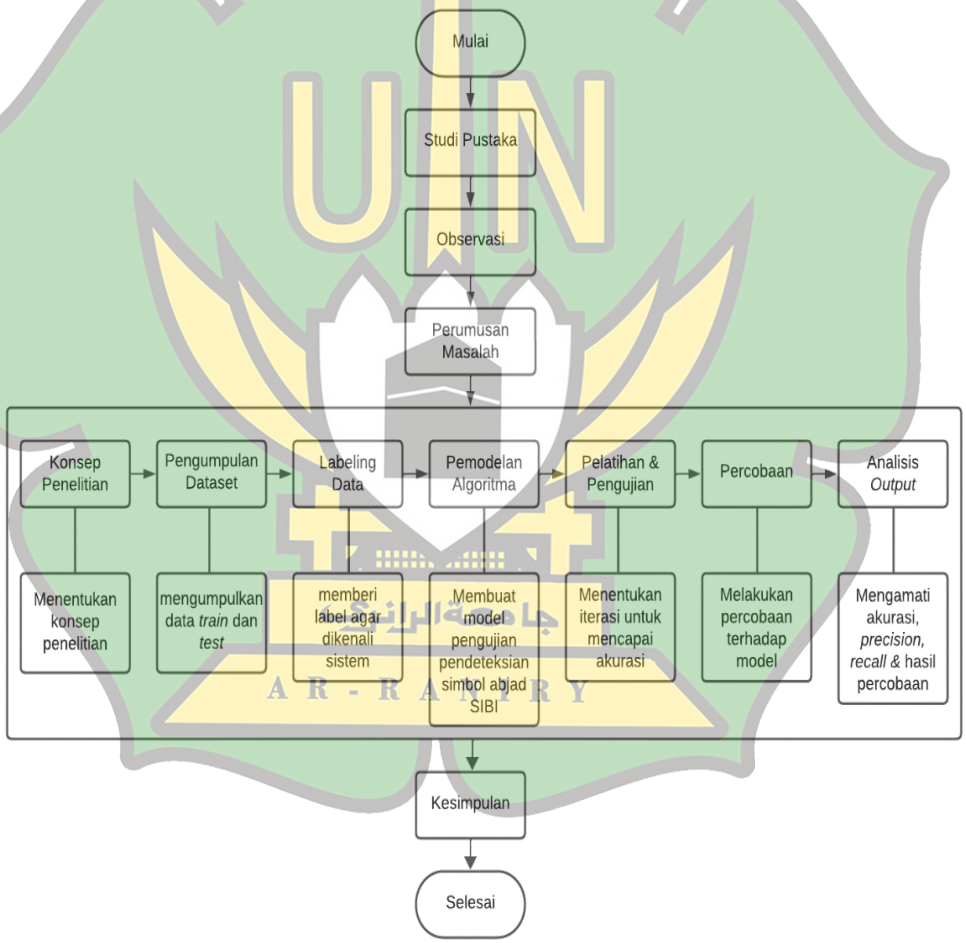
III.2.8 Percobaan Keberhasilan

Pada tahap ini peneliti mencoba melakukan pengujian terhadap model yang sudah dibangun, dengan memasukkan gambar simbol abjad SIBI yang berformat (.jpg) pada sistem tersebut. Setelah diproses melalui model yang sudah dibangun, maka keluarlah hasil dari masukan tersebut berupa gambar yang telah terdeteksi beserta akurasi pendeteksian simbol abjad SIBI dengan setiap hurufnya.

III.2.9 Analisis *Output*

Tahap akhir yang peneliti lakukan adalah analisis terhadap *output* berdasarkan asumsi yang telah silakukan apakah sesuai dengan *input.* Pada tahap ini akan ditampilkan hasil *output* dari *input image.*

III.3 Kerangka Pemikiran



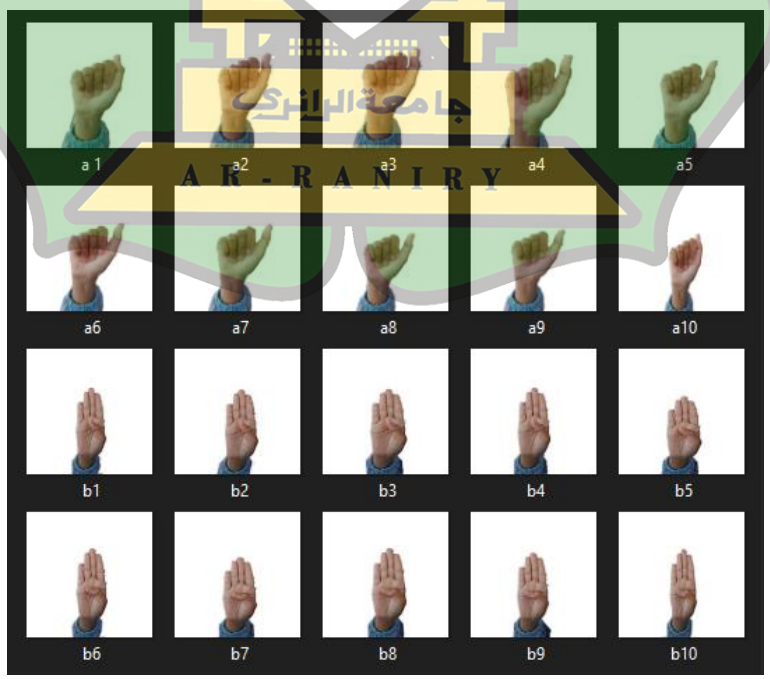
Gambar III. 5 Kerangka Pemikiran

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

Pada bab IV akan diberi pemaparan mengenai hasil penelitian dan pembahasan yang telah dilakukan oleh peneliti saat melakukan penelitian terhadap analisis akurasi pada simbol abjad SIBI yang uji dengan menggunakan model SSD *MobileNet* dan YOLOv7. Penelitian ini merupakan penelitian yang difokuskan pada deteksi gambar simbol abjad SIBI dari A sampai Z dengan total jumlah kelas sebanyak 26 kelas data yang setiap kelasnya berjumlah 10 gambar dan total keseluruhan data yaitu 260 data gambar.

IV.1 Pengumpulan *Dataset*

Proses pengumpulan pada *dataset* dilakukan dengan mengambil gambar simbol abjad SIBI dari A sampai Z menggunakan kamera *smartphone* Samsung A20s. Seluruh data gambar yang telah dikumpulkan berjumlah 260 data. Gambar IV.1 dibawah ini merupakan *dataset* abjad SIBI.



Gambar IV. 1 Kumpulan *Dataset*

Kemudian setelah semua data dikumpulkan seluruh data akan dibagi menjadi dua, yaitu data *test* 20% dan data *train* 80% agar saat pengujian memiliki nilai akurat yang tinggi. Pembagian kedua data dapat dilihat pada tabel IV.1.

Tabel IV. 1 Data *Training* dan *Testing*

Nama Kelas	Jumlah Data Training	Jumlah Data Testing
A	8	2
B	8	2
C	8	2
D	8	2
E	8	2
F	8	2
G	8	2
H	8	2
I	8	2
J	8	2
K	8	2
L	8	2
M	8	2
N	8	2
O	8	2
P	8	2
Q	8	2
R	8	2
S	8	2
T	8	2
U	8	2
V	8	2
W	8	2
X	8	2
Y	8	2
Z	8	2
Total	**208 (80% data train)**	**52 (20% data test)**

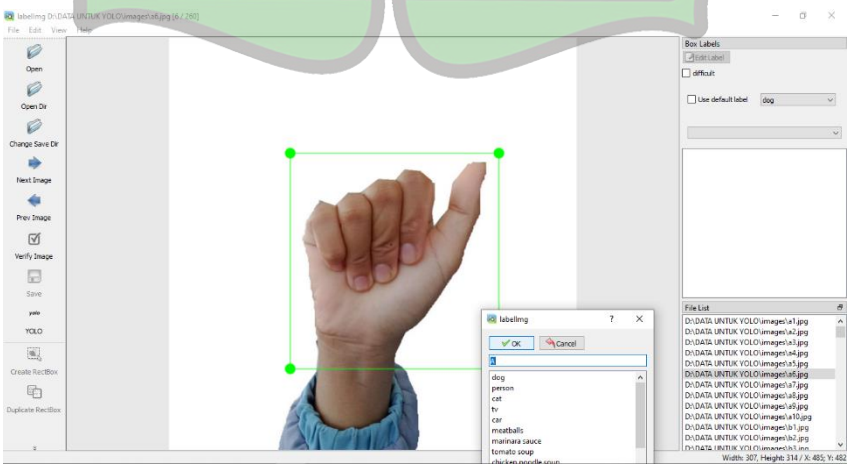
Tabel IV.1 merupakan data training dan data testing. Pada penelitian ini mengunakan sebanyak 26 label abjad A sampai Z. pada data training setiap kelas berjumlah 8 gambar dari setiap kelas dengan jumlah total data training sebanyak 208 gambar, dan untuk data testing setiap kelas berjumlah 2 gambar dari setiap

kelas dengan keseluruhan total data testing sebanyak 52 gambar. Maka total data yang digunakan pada penelitian ini berjumlah 260 data gambar.

IV.1.1 Pelabelan Gambar

Pada tahap ini dilakukan dengan memberi label pada sebuah gambar agar sistem dapat mengenali nama simbol yang akan dideteksi. Pelabelan gambar dilakukan dengan menggunakan aplikasi *Labelimg* yang berfungsi untuk memberikan label pada gambar simbol abjad SIBI berdasarkan pembagian dan kategori yang telah ditentukan, dapat dilihat pada gambar IV.2 dan gambar IV.3

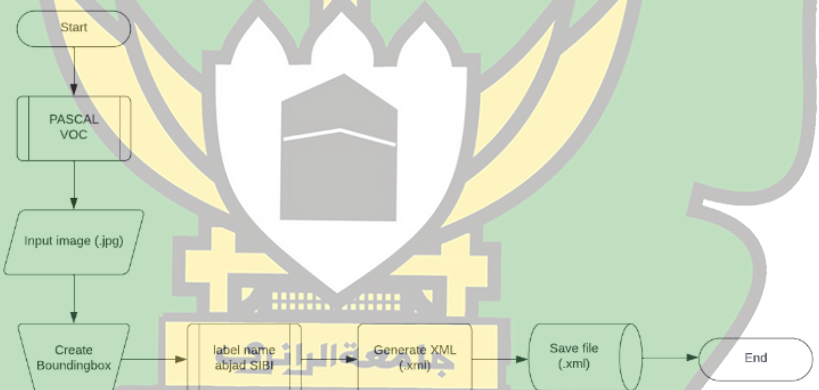
Gambar IV. 2 Label Gambar dari SSD



Gambar IV. 3 Label Gambar dari YOLO

Gambar IV.2 diatas merupakan pelabelan gambar simbol abjad SIBI dengan menggunakan PASCAL VOC dan gambar yang sudah diberi label akan tersimpan kedalam format (*.xml).* Pada gambar IV.3 merupakan pelabelan gambar simbol SIBI menggunakan YOLO dan gambar yang sudah diberi label akan tersimpan kedalam format (.txt).

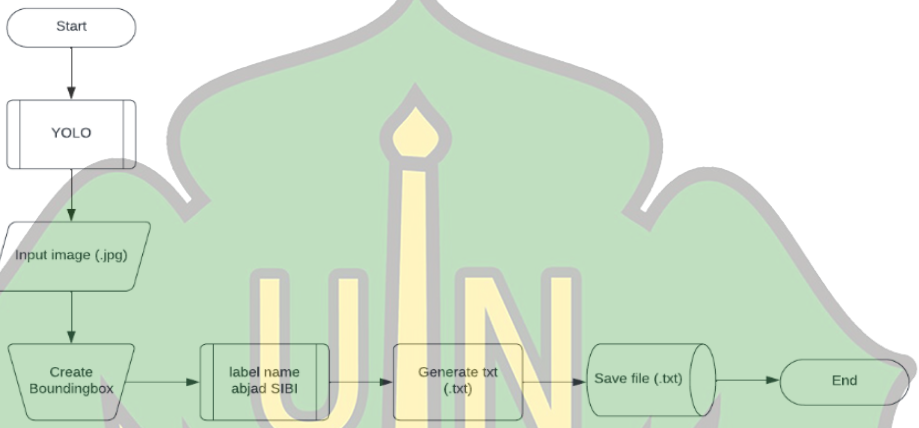
Proses pelabelan dilakukan secara manual untuk 260 data gambar abjad SIBI menggunakan *labelimg.* Tujuan dari pelabelan gambar yaitu untuk mendapatkan sebuah karakter khusus yang terdapat pada masing-masing gambar tersebut. Karakter gambar yang dilabel tersebut akan dijadikan *learning* dalam proses training. Adapun tahapan dalam melakukan pelabelan gambar dapat dilihat pada gambar IV.4 dan gambar IV.5.



Gambar IV. 4 *Flowchart labelimg* PASCAL VOC

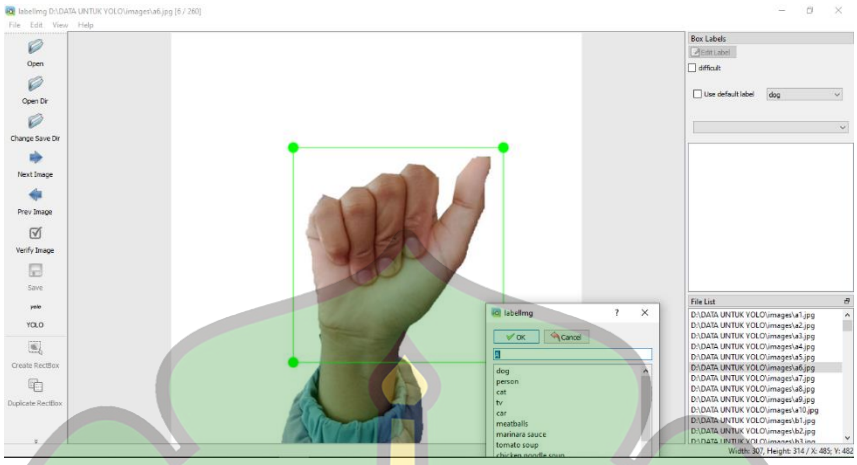
Pada gambar IV.4 merupakan *flowchart* pelabelan gambar data training model SSD *Mobilenet* yang dimulai dengan membuka aplikasi *labelimg* kemudian memilih Format PASCAL VOC , selanjutnya memasukkan gambar simbol abjad SIBI dengan format (.jpg). kemudian gambar ditandai dengan menggunakan *boudingbox* di sekeliling objek yang ingin diberi nama label, selanjutnya objek yang sudah dilabel akan disimpan anotasinya dengan memilih menu "*save"* atau

"*save as*" kemudian *labelimg* akan membuat *file* xml untuk menyimpan informasi anotasi objek pada gambar tersebut.



Gambar IV. 5 *Flowchart Labelimg* YOLO

Pada gambar IV.5 adalah *Flowchart* pelabelan gambar data training model YOLO, pertama buka aplikasi *labelimg* kemudiaan pilih format YOLO, lalu memasukkan gambar simbol abjad SIBI dengan format (.jpg). kemudian gambar ditandai dengan menggunakan *boundingbox* di sekeliling objek yang ingin diberi nama label, selanjutnya objek yang sudah dilabel akan disimpan anotasinya dengan memilih menu "*save*" atau "*save as*" *kemudian* labelimg akan membuat *file* txt untuk menyimpan informasi anotasi objek pada gambar tersebut.



Gambar IV. 6 Proses pelabelan

Gambar IV. 7 Sesudah pelabelan

Pada gambar IV.6 merupakan proses pelabelan gambar yang dilakukan dengan menandai objek menggunakan *boundingbox* atau kotak pembatas pada objek yang ingin diberi label dalam gambar. Setelah menandai objek dengan *boundingbox* gambar tersebut akan diberi label nama berdasakan klasifikasi nama simbol abjad SIBI seperti pada gambar IV.7 nama simbol abjad SIBI adalah (A).

IV.2 Hasil Penelitian

Hasil penelitian yang akan dibahas merupakan hasil dari pelabelan gambar, hasil pengujian model (training) dan hasil testing dari kedua model yang sudah di training.

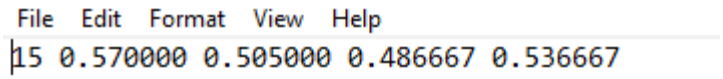
IV.2.1 Hasil Pelabelan Gambar

Objek gambar yang sudah dilabel akan menghasilkan *file* anotasi, yang mana file anotasi ini berisikan informasi tentang objek pada gambar tersebut dalam bentuk format (.xml) dan (.txt). Berikut file anotasi yang berisi informasi tentang objek dari gambar yang telah di label dapat dilihat pada gambar IV.8 dan gambar IV.9 dibawah ini.

```
<annotation>
        <folder>A</folder>
        <filename>a2.jpg</filename>
        <path>E:\DETEKSI SIBI\Tensorflow Object Detection\TFODCourse\Tensorflow\workspace\TRAIN\A\a2.jpg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>600</width>
                <height>600</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>A</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>186</xmin>
                        <ymin>143</ymin>
                        <xmax>509</xmax>
                        <ymax>541</ymax>
                </bndbox>
        </object>
</annotation>
```

Gambar IV. 8 *file* xml

Pada gambar IV.8 merupakan isi *file* anotasi dalam format xml. Pada *file* tersebut berisikan informasi tentang gambar serta detail setiap objek yang dilabel didalamnya. Beberapa elemen utama yang terdapat dalam *file* xml tersebut seperti *"filename"* yang merupakan nama berkas gambar yang dilabel, pada gambar diatas nama berkasnya adalah "a2.jpg". selanjutnya terdapat *"size"* yang berisikan informasi tentang lebar (*width*), tinggi (*height*) dan kedalaman (*depth*). Dan yang terakhir *"object"* yang mana elemen ini mewakili satu objek yang dilabel pada gambar, termasuk label, koordinat *boundingbox* dan pose.



Gambar IV. 9 *file* txt

Gambar IV.9 merupakan isi *file* anotasi dalam format txt. Setiap baris dalam *file* txt mewakili satu objek dengan format "label", "x", "y", *"width", "height".* Pada gambar diatas diketahui bahwa objek dengan label "15" memiliki bounding box yang dimulai dari titik (57% lebarnya) dan (50.5% tingginya), dengan lebar bounding box (48.7% *width*) dan tinggi bounding box (53.7% *height*).

IV.2.2 Hasil Training Model SSD *MobileNet*

Training model atau pelatihan model merupakan tahapan awal dari *neural network.* Pada tahap ini seluruh *dataset* akan dilakukan proses training agar sistem dapat mempelajari pola gambar dari simbol abjad SIBI (A-Z). Proses training tersebut terjadi pada jaringan CNN dengan training model SSD *MobileNet* yang di aplikasikan pada google colab. Model training yang digunakan yaitu *pre-trained* model *"ssd_mobilenet_v2_fpnlite_-320x320_coco17_tpu-8.config".* Jumlah step yang digunakan adalah 1500.

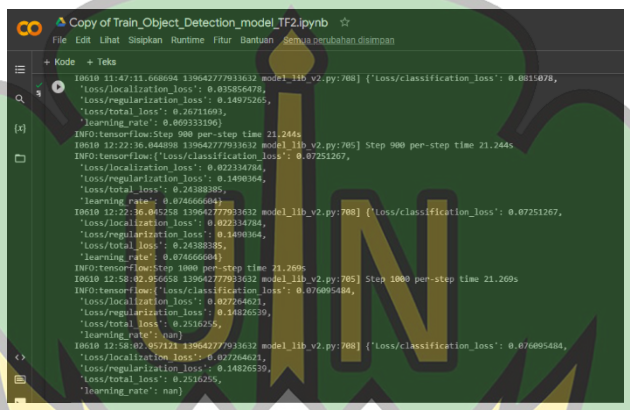
```
# Run the command below from the
content/models/research/object_detection directory
"""
PIPELINE_CONFIG_PATH=/content/gdrive/MyDrive/customTF2/data/ssd_mobilen
et_v2_fpnlite_320x320_coco17_tpu-8/pipeline.config
MODEL_DIR=/content/gdrive/MyDrive/customTF2/data/ssd_mobilenet_v2_fpnli
te_320x320_coco17_tpu-8/checkpoint
NUM_TRAIN_STEPS=1500
SAMPLE_1_OF_N_EVAL_EXAMPLES=1

python model_main_tf2.py -- \
  --model_dir=$MODEL_DIR --num_train_steps=$NUM_TRAIN_STEPS \
  --sample_1_of_n_eval_examples=$SAMPLE_1_OF_N_EVAL_EXAMPLES \
  --pipeline_config_path=$PIPELINE_CONFIG_PATH \
  --alsologtostderr
"""

!python model_main_tf2.py --
pipeline_config_path=/mydrive/customTF2/data/ssd_mobilenet_v2_fpnlite_3
20x320_coco17_tpu-8.config --model_dir=/mydrive/customTF2/training --
alsologtostderr
```

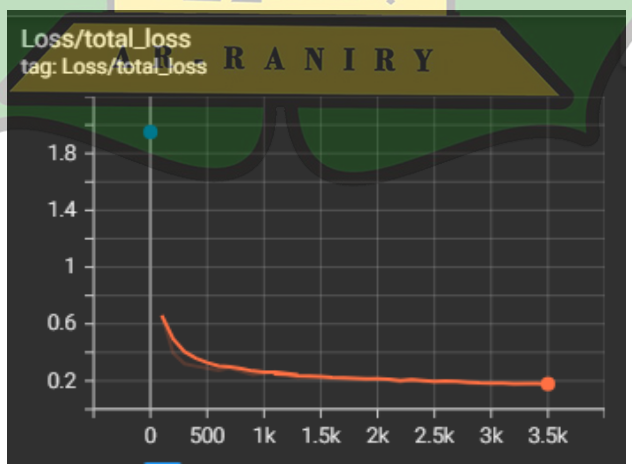
Gambar IV. 10 Skrip menjalankan "model_main_tf2.py"

Kode diatas merupakan perintah untuk menjalankan skrip "model_main_tf2.py" dengan menggunakan *file* konfigurasi *pipeline* yang berada pada *path* "/mydrive/CustomTF2/data/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.config" dan hasil pelatiha dan *checkpoint* akan disimpan dalam direktori "/mydrive/CustomTF2/training/".



Gambar IV. 11 Proses *Training*

Proses training dimulai dari 100 sampai dengan maksimal step 1500. Setiap step akan mengeluarkan sebuah variable yang disebut dengan *loss* yaitu ukuran yang digunakan untuk mengukur seberapa baik model dalam mempelajari pola dari data training. Dari training model deteksi objek didapatkan hasil sebagai berikut.



Gambar IV. 12 Grafik Total Loss

Pada gambar IV.12 merupakan grafik dari total *loss* hasil dari proses training. Pada gambar tersebut dapat dilihat bahwa adanya penurunan nilai total *loss* yang awalnya di 0.6 turun ke 0.2. semakin banyak step yang di proses maka akan membuat nilai dari total loss semakin menurun dan semakin kecil nilai *loss* maka model yang diinput akan semakin baik dalam mendeteksi.

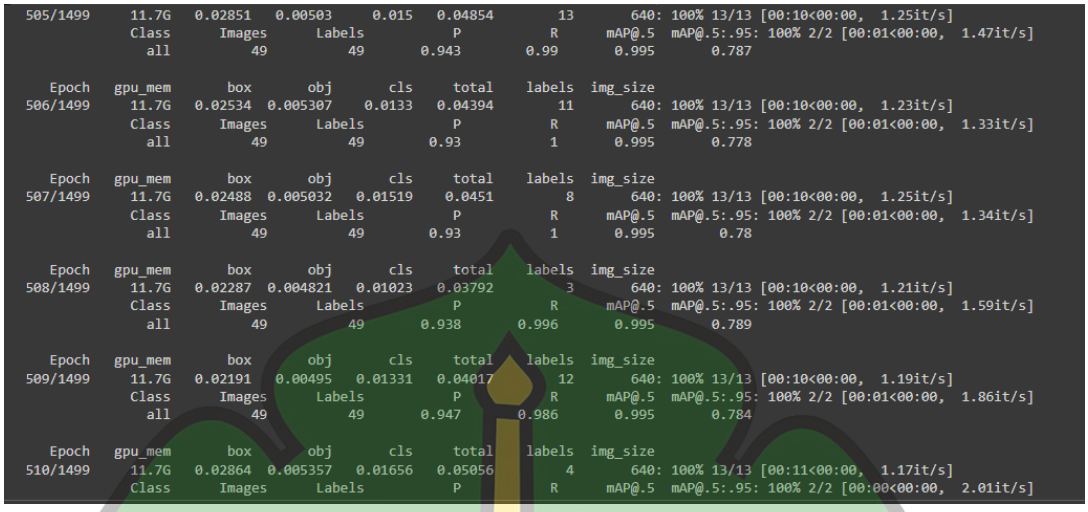
IV.2.3 Hasil Training Model YOLOv7

Training model merupakan proses dimana sebuah model pembelajaran mesin belajar dari kumpulan data yang telah dilabel untuk dapat mengenali pola dan dapat memprediksinya. Proses training tersebut dilakukan dengan menggunakan model YOLOv7 yang diaplikasikan pada *google colab*. Model training yang digunakan yaitu pre-trained model "yolov7.pt" dengan ukuran batc 16 dan menggunakan *epoch* 1500.

```
!python train.py --batch 16 --
cfg  /content/yolov7/cfg/training/custom_yolov7.yaml --epochs 1500 --
data /content/yolov7/data/custom.yaml --weights 'yolov7.pt' --device 0
```

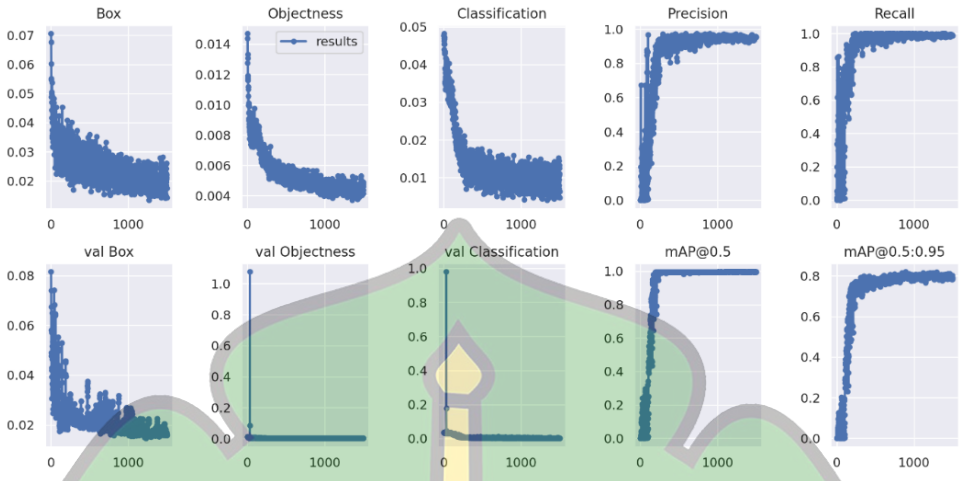
Gambar IV. 13 Skrip menjalankan "train.py"

Kode diatas adalah perintah yang digunakan untuk menjalankan training model YOLOv7. Perintah "--batch 16" adalah perintah untuk menentukan ukuran *batch* yang digunakan dalam pelatihan, pada penelitian ini ukuran *batch* yang digunakan adalah 16 yang bearti model akan melihat dan memproses 16 gambar pada setiap iterasi pelatihan. Kode "--cfg/content/yolov7/cfg/training/custom_yolov7.yaml" adalah untuk menentukan jalur file konfigurasi model yang digunakan selama pelatihan. Kode "--weights 'yolov7.pt'" adalah jalur file *pre-trained* model yang digunakan sebagai awal pelatihan. Dan "--epoch" adalah jumlah iterasi pelatihan yang dilakukan yaitu 1500 epoch.

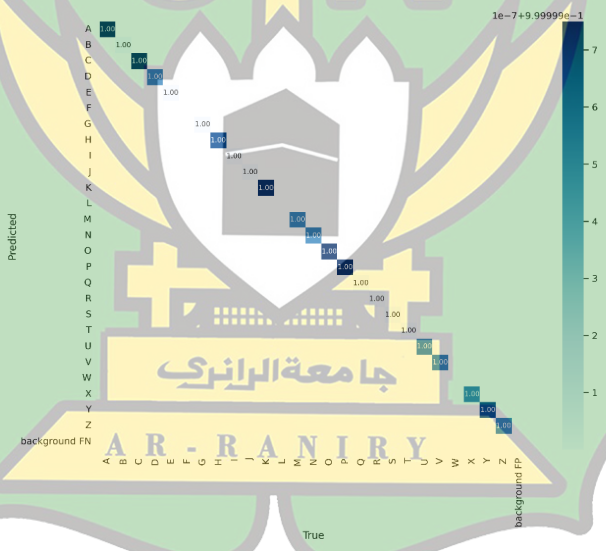


Gambar IV. 14 Proses *Training* YOLOv7

Pada gambar diatas merupakan proses pelatihan model dengan jumlah iterasinya adalah 1500. Setiap stepsnya akan mengeluarkan beberapa variabel seperti "epoch" menunjukkan jumlah *epoch* (iterasi) yang dilakukan dalam proses pelatihan, "gpu_mem" menunjukkan penggunaan memori GPU dalam satuan *GigaByte* (GB) selama pelatihan model, "box" menunjukkan nilai rata-rata dari *loss function* untuk prediksi pembatas objek pada setiap *batch* data, "obj" menunjukkan nilai rata-rata dari *loss function* untuk prediksi objek (apakah objek ada atau tidak) pada setiap batch data, "cls" menunjukka nilai rata-rata dari *loss function* untuk prediksi kelas objek pada setiap *batch* data, "total" menunjukkan nilai rata-rata dari total *loss function* (kombinasi dari *loss function* box, obj, dan cls) pada setiap batch data, "lables" menunjukkan jumlah total label yang ada pada pelatihan dataset, dan "img_size" menunjukkan ukuran gambar yang digunakan dalam pelatihan model deteksi objek dan didapatkan hasil seperti pada Gambar IV.15.



Gambar IV. 15 Hasil Evaluasi Data Training



Gambar IV. 16 *Confusion Matrix*

Dari gambar IV.16 *confusion matrix* diatas dapat dilihat tingkat akurasi pada semua kelas objek simbol abjad sibi mendapatkan nilai akurasi yang cukup baik yaitu 1.00 yang artinya sangat akurat.

IV.2.4 Hasil Testing

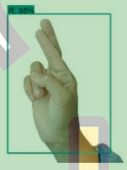
Pada proses testing data yang diuji berjumlah 26 kelas data gambar simbol abjad SIBI dan tiap kelasnya berjumlah 2 data gambar. Hasil testing dari deteksi model SSD *MobileNet* diperoleh nilai akurasi paling tinggi yaitu sebesar 99% dan paling rendahnya yaitu 80% denga n rata-rata 94% dari semua pengujian . Sedangkan hasil testing dari deteksi model YOLOv7 diperoleh nilai akurasi paling tinggi yaitu 0.97 dan paling rendahnya yaitu 0.77 dengan rata-rata 0.91 dari semua pengujian. Berikut contoh hasil testing yang dapat diliat pada tabel IV.2 dan tabel IV.3.

Tabel IV. 2 Hasil *Testing* model SSD MobileNet

No	Nama Kelas	Hasil Deteksi	Ket	Benar//Salah	Skor Prediksi
1	A		Terdeteksi	Benar	94%
2	B		Terdeteksi	Benar	93%
3	C		Terdeteksi	Benar	96%
4	D		Terdeteksi	Benar	96%

No	Nama Kelas	Hasil Deteksi	Ket	Benar//Salah	Skor Prediksi
5	E		Terdeteksi	Benar	97%
6	F		Terdeteksi	Benar	92%
7	G		Terdeteksi	Benar	97%
8	H		Terdeteksi	Benar	95%
9	I		Terdeteksi	Benar	96%
10	J		Terdeteksi	Benar	98%

No	Nama Kelas	Hasil Deteksi	Ket	Benar//Salah	Skor Prediksi
11	K		Terdeteksi	Benar	94%
12	L		Terdeteksi	Benar	96%
13	M		Terdeteksi	Benar	97%
14	N		Terdeteksi	Benar	97%
15	O		Terdeteksi	Benar	98%
16	P		Terdeteksi	Benar	98%

No	Nama Kelas	Hasil Deteksi	Ket	Benar//Salah	Skor Prediksi
17	Q		Terdeteksi	Benar	98%
18	R		Terdeteksi	Benar	95%
19	S		Terdeteksi	Benar	80%
20	T		Terdeteksi	Benar	95%
21	U		Terdeteksi	Benar	97%
22	V		Terdeteksi	Benar	99%

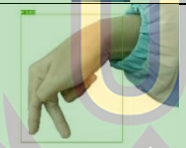
No	Nama Kelas	Hasil Deteksi	Ket	Benar//Salah	Skor Prediksi
23	W		Terdeteksi	Benar	98%
24	X		Terdeteksi	Benar	95%
25	Y		Terdeteksi	Benar	97%
26	Z		Terdeteksi	Benar	97%

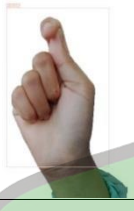
Tabel IV. 3 Hasil *Testing* model YOLOv7

No	Nama Kelas	Hasil Deteksi	Ket	Benar/Salah	Skor Prediksi
1	A		Terdeteksi	Benar	0.86

No	Nama Kelas	Hasil Deteksi	Ket	Benar/Salah	Skor Prediksi
2	B		Terdeteksi	Benar	0.92
3	C		Terdeteksi	Benar	0.84
4	D		Terdeteksi	Benar	0.92
5	E		Terdeteksi	Benar	0.91
6	F		Terdeteksi	Benar	0.91
7	G		Terdeteksi	Benar	0.91

No	Nama Kelas	Hasil Deteksi	Ket	Benar/Salah	Skor Prediksi
8	H		Terdeteksi	Benar	0.77
9	I		Terdeteksi	Benar	0.79
10	J		Terdeteksi	Benar	0.96
11	K		Terdeteksi	Benar	0.80
12	L		Terdeteksi	Benar	0.97
13	M		Terdeteksi	Benar	0.90

No	Nama Kelas	Hasil Deteksi	Ket	Benar/Salah	Skor Prediksi
14	N		Terdeteksi	Benar	0.93
15	O		Terdeteksi	Benar	0.89
16	P		Terdeteksi	Benar	0.93
17	Q		Terdeteksi	Benar	0.94
18	R		Terdeteksi	Benar	0.92
19	S		Terdeteksi	Benar	0.86

No	Nama Kelas	Hasil Deteksi	Ket	Benar/Salah	Skor Prediksi
20	T		Terdeteksi	Benar	0.95
21	U		Terdeteksi	Benar	0.93
22	V		Terdeteksi	Benar	0.78
23	W		Terdeteksi	Benar	0.88
24	X		Terdeteksi	Benar	0.90
25	Y		Terdeteksi	Benar	0.91

No	Nama Kelas	Hasil Deteksi	Ket	Benar/Salah	Skor Prediksi
26	Z		Terdeteksi	Benar	0.91

IV.2.5 *Confusion Matrix*

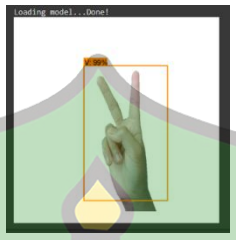
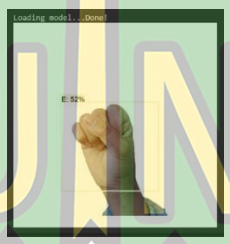
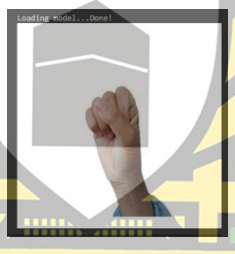
Confusion matrix merupakan sebuah metode evaluasi yang digunakan untuk mengukur kinerja deteksi dari model yang telah di latih. *Confusion matrix* menampilkan perbandingan antara prediksi yang dilakukan oleh model dengan nilai sebenarnya dari data yang di uji. Dalam *confusion matrix* terdapat empat *matrix* evaluasi utama, yaitu *True Positive (TP), False Positive (FP), False Negative (FN),* dan *True Negative (TN)* (Sindy, 2019).

Keterangan:

1. *True Positive (TP)* yaitu jumlah data yang diprediksi benar positif oleh model
2. *False Positive (FP)* yaitu jumlah data yang diprediksi salah positif oleh model
3. *False Negative (FN)* yaitu jumlah data yang diprediksi salah negatif oleh model
4. *True Negative (TN)* yaitu jumlah data yang diprediksi benar negatif oleh model.

Tabel dari empat matrix evaluasi *pada confusion matrix* dapat dilihat pada tabel IV.3 berikut.

Tabel IV. 4 *Confusion Matrix*

Confusion Matrix	**Contoh Gambar**	**Keterangan**
True Positive (TP)		Data merupakan objek simbol SIBI dan model memprediksi simbol SIBI
False Positive (FP)		Data bukan kelas dari objek simbol SIBI yang sebenarnya, namun model memprediksi simbol dari kelas objek tersebut
False Negative (FN)		Data merupakan objek simbol SIBI, tapi model tidak mampu memprediksi data tersebut.
True Negative (TN)		Data bukan simbol SIBI dan model memprediksi bukan simbol SIBI

IV.2.5.1 *Confusion Matrix* pada SSD *MobileNet*

Berikut perhitungan *confusion matrix* dari seluruh data yang telah di uji coba deteksi menggunakan model SSD *MobileNet.*

Tabel IV. 5 *Confusion Matrix* pada SSD *MobileNet*

No	Hasil Deteksi	TP	FP	FN	TN
1	A	2			
2	B	2			
3	C	2			
4	D	2			
5	E	2	1		
6	F	2			
7	G	2			
8	H	2			
9	I	2			
10	J	2			
11	K	2			
12	L	2			
13	M	2			
14	N	2			
15	O	2			
16	P	2			
17	Q	2			
18	R	2			
19	S	1			
20	T	2			
21	U	2			
22	V	2			
23	W	2			
24	X	2			
25	Y	2			
26	Z	2			
Total		51	1	0	0

Tabel IV. 6 Hasil *Confusion Matrix* pada SSD *MobileNet*

TP	**51**
FP	1
FN	0
TN	0
Total	52

Dari pengujian data pada tabel IV.5 didapatkan sebanyak 51 data *True Positive (TP)* yaitu data terdeteksi oleh model dengan benar, terdapat 1 data *False Positive (FP)* yaitu data terdeteksi oleh model, tetapi bukan dari kelas data yang sebenarnya. Total data testing yang digunakan sebanyak 52 data gambar simbol abjad SIBI.

Setelah dilakukan uji coba pendeteksian pada seluruh data simbol abjad SIBI, maka untuk mendapatkan keakuratan dapat dihitung dengan menggunakan persamaan dari akurasi, *presicision* , *recall* dan F1 *Score*. Berikut proses pencarian perhitungan akurasi, presisi dan recall dari dataset simbol abjad SIBI pada SSD *MobileNet.*

1. Persamaan Akurasi

$$Accuracy = \frac{\text{TP+TN}}{\text{(TP+FP+FN+TN)}} \times 100\%$$
$$= \frac{51+0}{51+1+0+0} \times 100\%$$
$$= \frac{51}{52} \times 100\% = 0{,}9807\%$$
$$= 98{,}07\%$$

2. Persamaan *Precision*

$$Precision = \frac{\text{TP}}{\text{(TP+FP)}} \times 100\%$$
$$= \frac{51}{(51+1)} \times 100\%$$
$$= \frac{51}{52} \times 100\% = 0{,}9807\%$$
$$= 98{,}07\%$$

3. Persamaan *Recall*

$$Recall = \frac{\text{TP}}{\text{(TP+FN)}} \times 100\%$$
$$= \frac{51}{(51+0)} \times 100\%$$
$$= \frac{51}{51} \times 100\%$$
$$= 100\%$$

4. Persamaan F1 *Score*

$$F1\ Score = 2\ \frac{precision\ .\ recall}{precision+recall}$$

$$= 2\ \frac{(98{,}07)\ .\ (100)}{(98{,}07)\ +(100)}$$

$$= 2\ \frac{9.807}{198{,}07}$$

$$= \frac{19.614}{198{,}07} = 99{,}02$$

IV.2.5.2 *Confusion Matrix* pada YOLOv7

Berikut perhitungan *confusion matrix* dari seluruh data yang telah di uji coba deteksi menggunkana model YOLOv7.

Tabel IV. 7 *Confusion Matrix* pada YOLOv7

No	Hasil Deteksi	TP	FP	FN	TN
1	A	2			
2	B	2			
3	C	2			
4	D	2			
5	E	2			
6	F	2			
7	G	2			
8	H	2			
9	I	2			
10	J	2			
11	K	2			
12	L	2			
13	M	2			
14	N	2			
15	O	2			
16	P	2			
17	Q	2			
18	R	2			
19	S	2			
20	T	2			
21	U	2			
22	V	2			
23	W	2			
24	X	2			

25	Y	2			
26	Z	2			
Total		52	0	0	0

Tabel IV. 8 Hasil *Confusion Matrix* pada YOLOv7

TP	**52**
FP	0
FN	0
TN	0
Total	52

Dari pengujian data pada objek simbol SIBI menggunakan model YOLOv7, mendapatkan hasil seperti pada tabel IV.8 bahwa hasil pengujian dari 52 data terprediksi *True Positive (TP)* semua oleh model.

Setelah dilakukan pengujian terhadap seluruh data simbol SIBI, maka untuk mendapatkan keakuratan dapat dihitung dengan menggunakan persamaan dari akurasi, *precision*, *recall* dan F1 *Scroe*. Berikut proses pencarian perhitungan akurasi, *precision*, *recall*, dan F1 *Score* dari data simbol SIBI pada model YOLOv7.

1. Persamaan Akurasi

$$Accuracy = \frac{TP+TN}{(TP+FP+FN+TN)} \times 100\%$$

$$= \frac{52+0}{52+0+0+0} \times 100\%$$

$$= \frac{52}{52} \times 100\% = 100\%$$

2. Persamaan *Precision*

$$Precision = \frac{TP}{(TP+FP)} \times 100\%$$

$$= \frac{52}{(52+0)} \times 100\%$$

$$= \frac{52}{52} \times 100\% = 100\%$$

3. Persamaan *Recall*

$$Recall = \frac{TP}{(TP+FN)} \times 100\%$$

$$= \frac{52}{(52+0)} \times 100\%$$

$$= \frac{52}{52} \times 100\% = 100\%$$

4. Persamaan F1 *Score*

$$F1\ Score = 2\ \frac{precision\ .\ recall}{precision+recall}$$

$$= 2\ \frac{100\ .\ 100}{100+100}$$

$$= 2\ \frac{10.000}{200} = 100\%$$

Tabel IV. 9 Perhitungan *Accuracy*, *Presicion*, *Recall*, dan F1 *Score* pada model SSD *MobileNet* dam YOLOv7.

	SSD MOBILENET	**YOLOv7**
Accuracy	98,07%	100%
Presicion	99,07%	100%
Recall	100%	100%
F1 *Score*	99,02%	100%

Dari tabel IV. 9 terdapat bahwa tingkat akurasi dari hasil deteksi pada objek simbol abjad SIBI menggunakan model SSD *MobileNet* dan YOLOv7 mendapatkan nilai yang cukup tinggi dengan nilai akurasi pada model SSD *MobileNet* yaitu 98,07% dan pada YOLOv7 100%. Kemudian untuk nilai *precision* pada SSD *MobileNet* sebanyak 99,07% dan pada YOLOv7 sebanyak 100% . Nilai *recall* pada SSD *MobileNet* sebanyak 100% dan pada YOLOv7 sebanyak 100%, Dan nilai F1 *Score* yang dihasilkan pada model SSD *MobileNet* sebanyak 99,02% dan pada YOLOv7 sebanyak 100%.

IV.3 Pembahasan

Analisis akurasi pada simbol Abjad Sistem Isyarat Bahasa Indonesia (SIBI) menggunakan metode CNN dengan arsitektur SSD *MobileNet* dan YOLO yang dilakukan pada 26 kelas gambar simbol abjad SIBI. Pendeteksian dari kedua model tersebut mendapatkan hasil akurasi bahwa model deteksi menggunakan YOLO mampu memberikan akurasi deteksi yang lebih tinggi dibandingkan dengan SSD *MobileNet*, hal ini didapatkan berdasarkan dari perhitungan menggunakan *Confusion Matrix.*

Confusion Matrix merupakan sebuah metode evaluai yang digunakan untuk mengukur kinerja deteksi dari model yang telah dilatih. *Confusion matrix* menampilkan perbandingan antara prediksi yang dilakukan oleh model dengan nilai sebenarnya dari data yang diuji. Dalam confusion matrix terdapat empat matrix evaluasi utama, yaitu *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), dan *True Negative* (TN) (Sindy, 2019).

Keterangan:

1. *True Positive (TP)* yaitu jumlah data yang diprediksi benar positif oleh model.
2. *False Positive (FP)* yaitu jumlah data yang diprediksi salah positif oleh model.
3. *False Negative (FN)* yaitu jumlah data yang diprediksi salah negatif oleh model.
4. *True Negative (TN)* yaitu jumlah data yang diprediksi benar negatif oleh model.

Menurut (Manajang et al., 2020) *confusion matrix*, data ditentukan 3 nilai yaitu:

1. *Accuracy*, merupakan hasil perhitungan tingkat keakuratan deteksi objek terhadap objek SIBI secara keseluruhan. Adapun persamaan *accuracy* dapat dilihat dipersamaan II.1 (Manajang et al., 2020).

$$Accuracy = \frac{Jumlah\ Objek\ Yang\ Terdeteksi\ Benar}{Jumlah\ Keseluruhan\ Objek\ Yang\ Terdeteksi} \text{ x } 100\%$$

Persamaan (II.1)

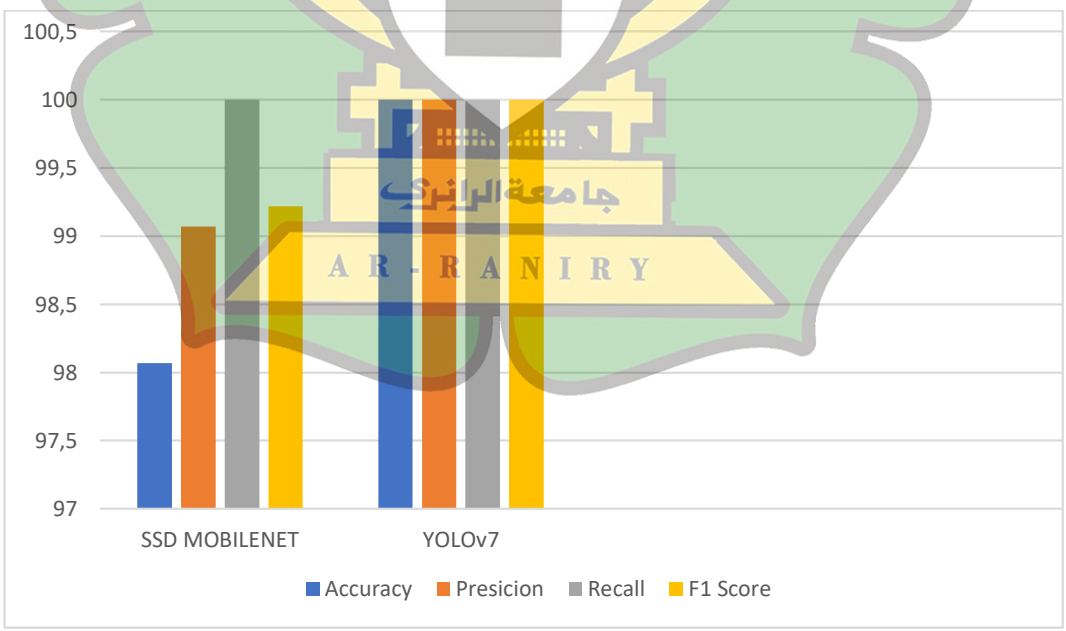
2. *Precission,* merupakan persamaan mengenai jumlah prediksi yang benar dibandingkan dari keseluruhan hasil yang dapat di prediki oleh sistem. Pada persamaan ini akan menghasilkan jumlah objek yang benar dari keseluruhan yang di deteksi oleh sistem. Adapun persamaan *precission* dapat dilihat dipersamaan II.2 (Manajang et al., 2020).

$$Precision = \frac{TP}{(TP + FP)} \text{ x } 100\% \qquad \text{Persamaan (II.2)}$$

3. *Recall,* merupakan jumlah prediksi yang benar dibandingkan dengan keseluruhan hasil pendeteksian objek sebenarnya. Persamaan ini akan menghasilkan jumlah objek deteksi dengan benar dari keseluruhan jumlah objek yang terdeteksi oleh sistem. Adapun persamaan *recall* dapat dilihat dipersamaan II.3 (Manajang et al., 2020).

$$Recall = \frac{TP}{(TP + FN)} \text{ x } 100\% \qquad \text{Persamaan (II.3)}$$

Diagram perhitungan *confusion matrix* dari kedua model dapat dilihat pada gambar IV.35 berikut.



Gambar IV. 17 Diagram Perhitungan *Accuracy, Presicion, Recall,* dan F1 *Score* pada model SSD *MobileNet* dam YOLOv7

BAB V
PENUTUP

V.1 Kesimpulan

Dari hasil penelitian yang telah penulis lakukan, penulis mengambil beberapa kesimpulan pada penelitian ini, yaitu:

1. Proses dalam implementasi metode *Convolutional Neural Network* (CNN) dengan menggunakan arsitektur SSD *MobileNet* dan *You Only Look Once* (YOLO) dalam melakukan deteksi objek pada simbol abjad Sistem Isyarat Bahasa Indonesia (SIBI) berjalan dengan baik dan menghasilkan keberhasilan pendeteksian simbol abjad SIBI.
2. *Dataset* yang digunakan sebanyak 260 data gambar simbol abjad SIBI dengan jumlah kelas sebanyak 26 kelas. *Dataset* terbagi 2 yaitu data train 80% dan data test 20%. Pendeteksian dengan model SSD *MobileNet* menghasilkan tingkat akurasi sebesar 98,07%, presisi 99,07%, *recall* 100% dan nilai F1 *Score* 99,02%. Sedangkan pendeteksian menggunakan model *You Only Look Once* (YOLO) menghasilkan nilai akurasi yang cukup tinggi yaitu 100% nilai akurasi, 100% nilai presisi, 100% nilai *recall*, dan 100% nilai F1 *Score*. Dari pelatihan model yang dilakukan dengan menggunakan model SSD *MobileNet* dan YOLOv7 pada penelitian ini diperoleh bahwa pendeteksian objek menggunaka model YOLOv7 menghasilkan nilai akurasi yang lebih tinggi dibandingkan deteksi objek pada model SSD *MobileNet* yaitu 100% akurasi pada YOLOv7dan 98,07% pada SSD *MobileNet.*

V.2 Saran

Adapun saran untuk pengembangan pada penelitian ini agar lebih baik, terdapat beberapa saran yaitu:

1. Menambahkan jumlah dataset dan jumlah data dari setiap kelas.
2. Menggunakan objek gambar lain dengan dilakukan secara *realtime.*

3. Pada model pelatihan ssd mobilenet dapat menambahkan jumlah steps diatas 2000 agar tingkat akurasi yang didapatkan semakin baik dan tinggi.
4. Jika pelatihan model diimplementasikan pada *Google Colab* dan dengan jumlah *steps* yang lebih dari 1500, disarankan menggunkan jenis *google colab pro* atau menggunakan *platform* komputasi lainnya.

DAFTAR PUSTAKA

Putra, ilham rizaldi widy. (2021). *Sistem deteksi simbol pada sibi (sistem isyarat bahasa indonesia) menggunakan Convolutional Neural Network.*

Wulan Angraini. (2020). *Deep Learning Untuk Deteksi Wajah Yang Berhijab Menggunakan Algoritma Convolutional Neural Network (CNN) Dengan Tensorflow*. Universitas Islam Negeri Ar-Raniry.

Rachardi, F. (2020). *Deteksi Gambar Gestur Kosakata Bahasa Isyarat Indonesia dengan Convolutional Neural Network*. 192.

Mulyana, D. I., Lazuardi, M. F., & Yel, M. B. (2022). Deteksi Bahasa Isyarat Dalam Pengenalan Huruf Hijaiyah Dengan Metode YOLOV5. *Jurnal Teknik Elektro Dan Komputasi (ELKOM)*, *4*(2), 145–151. http://jurnal.unmuhjember.ac.id/index.php/ELKOM/article/view/8145

Sani, A., & Rahmadinni, S. (2022). Deteksi Gestur Tangan Berbasis Pengolahan Citra. *Jurnal Rekayasa Elektrika*, *18*(2), 115–124. https://doi.org/10.17529/jre.v18i2.25147

Muqayyim, A. Al. (2021). *Pengenalan Isyarat Tangan Sistem Bahasa Isyarat Indonesia (SIBI) dengan Algoritme YOLO*. http://repository.unsoed.ac.id/id/eprint/11809%0Ahttp://repository.unsoed.ac.id/11809/9/DAFTAR PUSTAKA-Abu Bakar Ash Shidiq Al Muqayyim-H1A017085-Skripsi-2021.pdf

T I Sambi Ua, A. M., Lestriani, D. H., Sonia Kristanty Marpaung, E., Ong, J., Savinka, M., Nurhaliza, P., Yulia Ningsih, R., Kebun Jeruk Raya No, J., & Barat, J. (2023). Penggunaan Bahasa Pemrograman Python Dalam Analisis Faktor Penyebab Kanker Paru-Paru Universitas Bina Nusantara. *Jurnal Publikasi Teknik Informatika (JUPTI)*, *2*(2). https://doi.org/10.55606/jupti.v2i2.1742

Api, D. (2020). *IMPLEMENTASI ALGORITMA YOLO (YOU ONLY LOOK*. *1*(1), 81–91.

Budiyanta, N. E., Mulyadi, M., & Tanudjaja, H. (2021). Sistem Deteksi Kemurnian Beras berbasis Computer Vision dengan Pendekatan Algoritma YOLO. *Jurnal Informatika: Jurnal Pengembangan IT*, *6*(1), 51–55. https://doi.org/10.30591/jpit.v6i1.2309

Manajang, D., Dompie, S., & Jacobus, A. (2020). Implementasi Framework Tensorflow Object Detection Dalam Mengklasifikasi Jenis Kendaraan Bermotor. *Jurnal Teknik Informatika*, *15*(3), 171–178.

Pamungkas, N. H. (2020). *Deteksi Keaslian Mata Uang Rupiah Berbasis Android Menggunakan Algoritma Convolutional Neural Network Dengan Tensorflow*.

Sindy, F. (2019). *Pendeteksian Objek Manusia Secara Realtime Dengan Metode MobileNet-SSD Menggunakan Movidius Neural Stick pada Raspberry Pi*.

Universitas Sumatera Utara.

Hidayahtullah, A. M. (2022). Sistem *Deteksi Pada Simbol SIBI Secara Realtime Menggunakan SSD MobileNet.*

Passa, R. S., Nurmaini, S., Rini, D. P., Sriwijaya, U., Raya, J., Km, P.-P., & Ogan Ilir, I. (2023). DETEKSI TUMOR OTAK PADA MAGNETIC RESONANCE IMAGING MENGGUNAKAN YOLOv7. *Jurnal Ilmiah MATRIK*, *25*(2), 116–121.

Budiyanta, N. E., Mulyadi, M., & Tanudjaja, H. (2021). Sistem Deteksi Kemurnian Beras berbasis Computer Vision dengan Pendekatan Algoritma YOLO. *Jurnal Informatika: Jurnal Pengembangan IT*, *6*(1), 51–55. https://doi.org/10.30591/jpit.v6i1.2309

T I Sambi Ua, A. M., Lestriani, D. H., Sonia Kristanty Marpaung, E., Ong, J., Savinka, M., Nurhaliza, P., Yulia Ningsih, R., Kebun Jeruk Raya No, J., & Barat, J. (2023). Penggunaan Bahasa Pemrograman Python Dalam Analisis Faktor Penyebab Kanker Paru-Paru Universitas Bina Nusantara. *Jurnal Publikasi Teknik Informatika (JUPTI)*, *2*(2). https://doi.org/10.55606/jupti.v2i2.1742

Permana, D., & Sutopo, J. (2023). *APLIKASI PENGENALAN ABJAD SISTEM ISYARAT BAHASA INDONESIA (SIBI) DENGAN ALGORITMA YOLOv5 MOBILE APPLICATION ALPHABET RECOGNITION OF INDONESIAN LANGUAGE SIGN SYSTEM (SIBI) USING YOLOv5 ALGORITHM*. *11*(2), 231–240.

Sudana Putra, F., Kusrini, & Kurniawan, M. P. (2021). Deteksi Otomatis Jerawat Wajah Menggunakan Metode Convolutional Neural Network (CNN). *Journal of Information Technology*, *1*(2), 30–34. https://doi.org/10.46229/jifotech.v1i2.308

Gelar Guntara, R. (2023). Pemanfaatan Google Colab Untuk Aplikasi Pendeteksian Masker Wajah Menggunakan Algoritma Deep Learning YOLOv7. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, *5*(1), 55–60. https://doi.org/10.47233/jteksis.v5i1.750

Ma'ruf, A., & Hardjianto, M. (2023). Penerapan Algoritme You Only Look Once Version 8 Untuk Identifikasi Abjad Bahasa Isyarat Indonesia. *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, *2*(2), 567–576.

Apendi, S., Setianingsih, C., & Paryasto, M. (2023). Deteksi Bahasa Isyarat Sistem Isyarat Bahasa Indonesia Menggunakan Metode Single Shot Multibox Detector. *EProceedings …*, *10*(1), 249–255.https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/19322

LAMPIRAN

Lampiran 1 **Perintah Training Model Deteksi SSD *MobileNet***

1) Import Libraries

```
import os
import glob
import xml.etree.ElementTree as ET
import pandas as pd
import tensorflow as tf
print(tf.__version__)
```

```
2.14.0
```

2) Create *customTF2*, *training* and *data* folders in your google drive

Create a folder named ***customTF2*** in your google drive.

Create another folder named ***training*** inside the ***customTF2*** folder (***training*** folder is where the checkpoints will be saved during training)

Create another folder named ***data*** inside the ***customTF2*** folder.

3) Create and upload your image files and xml files.

Create a folder named ***images*** for your custom dataset images and create another folder named ***annotations*** for its corresponding xml files.

Next, create their zip files and upload them to the ***customTF2*** folder in your drive.

(Make sure all the image files have extension as ".jpg" only. Other formats like ".png" , ".jpeg" or even ".JPG" will give errors since the generate_tfrecord and xml_to_csv scripts here have only ".jpg" in them)

Collect Images Dataset and label them to get their PASCAL_VOC XML annotations

For Datasets, you can check out my Dataset Sources at the bottom of this article in the credits section. You can use any software for labeling like the labelImg tool.

Read this article to know more about collecting datasets and labeling process.

4) Upload the *generate_tfrecord.py* file to the *customTF2* folder on your drive.

You can find the generate_tfrecord.py file here

5) Mount drive and link your folder

```
from google.colab import drive
drive.mount('/content/gdrive')

# this creates a symbolic link so that now the path /content/gdrive/My\ Drive/ is equal to /mydrive
!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive
```

```
Mounted at /content/gdrive
 ABJAD.PNG                                          'SSD MOBILENET 222.pdf'
'Copy of Train_Object_Detection_model_TF2 (2).ipynb'  'ssd moilenet111.ipynb'
 customTF2                                           Untitled0.ipynb
 EPOCH1000.ipynb                                     Untitled1.ipynb
 EPOCH500.ipynb                                      Untitled2.ipynb
 EPOCH.ipynb                                         Untitled-2.jpg
'Folder tanpa nama'                                  Untitled3.ipynb
 images                                              Untitled4.ipynb
'My Drive'                                          'YOLO 1500 (2).ipynb'
 qobiltuamiin.ipynb                                  YOLO1RB.ipynb
 ssd1500.ipynb
```

6) Clone the tensorflow models git repository & Install TensorFlow Object Detection API

```
# clone the tensorflow models on the colab cloud vm
!git clone --q https://github.com/tensorflow/models.git

#navigate to /models/research folder to compile protos
%cd models/research

# Compile protos.
!protoc object_detection/protos/*.proto --python_out=.

# Install TensorFlow Object Detection API.
!cp object_detection/packages/tf2/setup.py .
!python -m pip install .
```

```
  Stored in directory: /tmp/pip-ephem-wheel-cache-p237okfg/wheels/53/dd/70/2de274d6c443c69d367bd6a5606f95e5a6df61aacf1435ec0d
  Building wheel for avro-python3 (setup.py) ... done
  Created wheel for avro-python3: filename=avro_python3-1.10.2-py3-none-any.whl size=43993 sha256=f244c0b040291ccaea4c63508c0be6c
  Stored in directory: /root/.cache/pip/wheels/bc/85/62/6cdd81c56f923946b401cecff38055b94c9b766927f7d8ca82
  Building wheel for crcmod (setup.py) ... done
  Created wheel for crcmod: filename=crcmod-1.7-cp310-cp310-linux_x86_64.whl size=31408 sha256=e7223013a73b29a9d70087f063217c4d43
  Stored in directory: /root/.cache/pip/wheels/85/4c/07/72215c529bd59d67e3dac29711d7aba1b692f543c808ba9e86
  Building wheel for dill (setup.py) ... done
  Created wheel for dill: filename=dill-0.3.1.1-py3-none-any.whl size=78542 sha256=9b53e48e48db11acbcbeee6a60f95a52c1d6d99f587060
  Stored in directory: /root/.cache/pip/wheels/ea/e2/86/64980d90e297e7bf2ce588c2b96e818f5399c515c4bb8a7e4f
  Building wheel for hdfs (setup.py) ... done
  Created wheel for hdfs: filename=hdfs-2.7.3-py3-none-any.whl size=34325 sha256=7812a364a237db73b0a390a8076e62735744638b29c31e74
  Stored in directory: /root/.cache/pip/wheels/e5/8d/b6/99c1c0a3ac5788c866b0ecd3f48b0134a5910e6ed26011800b
  Building wheel for seqeval (setup.py) ... done
  Created wheel for seqeval: filename=seqeval-1.2.2-py3-none-any.whl size=16161 sha256=011e4beca874a51095b9a7d080a4463cb26676de15
  Stored in directory: /root/.cache/pip/wheels/1a/67/4a/ad4082dd7dfc30f2abfe4d80a2ed5926a506eb8a972b4767fa
  Building wheel for pyjsparser (setup.py) ... done
  Created wheel for pyjsparser: filename=pyjsparser-2.7.1-py3-none-any.whl size=25985 sha256=b40a0bc2de39fc1c3ce687ad1485a5f933db
  Stored in directory: /root/.cache/pip/wheels/5e/81/26/5956478df303e2bf5a85a5df595bb307bd25948a4bab69f7c7
  Building wheel for docopt (setup.py) ... done
  Created wheel for docopt: filename=docopt-0.6.2-py2.py3-none-any.whl size=13706 sha256=f8699cd31ffb7f4883e30731fce7e794c6eb8d82
  Stored in directory: /root/.cache/pip/wheels/fc/ab/d4/5da2067ac95b36618c629a5f93f809425700506f72c9732fac
Successfully built object-detection avro-python3 crcmod dill hdfs seqeval pyjsparser docopt
Installing collected packages: sentencepiece, pyjsparser, docopt, crcmod, zstandard, tensorflow-model-optimization, tensorflow_io
  Attempting uninstall: pyparsing
    Found existing installation: pyparsing 3.1.1
    Uninstalling pyparsing-3.1.1:
      Successfully uninstalled pyparsing-3.1.1
Successfully installed apache-beam-2.51.0 avro-python3-1.10.2 colorama-0.4.6 crcmod-1.7 dill-0.3.1.1 dnspython-2.4.2 docopt-0.6.2
```

7) Test the model builder

```
# testing the model builder
!python object_detection/builders/model_builder_tf2_test.py
```

```
2023-11-09 07:04:31.556953: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: At
2023-11-09 07:04:31.557013: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Att
2023-11-09 07:04:31.557062: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory:
2023-11-09 07:04:32.843338: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Running tests under Python 3.10.12: /usr/bin/python3
[ RUN      ] ModelBuilderTF2Test.test_create_center_net_deepmac
2023-11-09 07:04:36.453519: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:47] Overriding orig_value setting because t
WARNING:tensorflow:`tf.keras.layers.experimental.SyncBatchNormalization` endpoint is deprecated and will be removed in a future r
W1109 07:04:36.483868 133398888275968 batch_normalization.py:1531] `tf.keras.layers.experimental.SyncBatchNormalization` endpoint
W1109 07:04:36.971744 133398888275968 model_builder.py:1112] Building experimental DeepMAC meta-arch. Some features may be omitte
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_create_center_net_deepmac): 1.95s
I1109 07:04:37.328484 133398888275968 test_util.py:2472] time(__main__.ModelBuilderTF2Test.test_create_center_net_deepmac): 1.95s
[       OK ] ModelBuilderTF2Test.test_create_center_net_deepmac
[ RUN      ] ModelBuilderTF2Test.test_create_center_net_model0 (customize_head_params=True)
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_create_center_net_model0 (customize_head_params=True)): 0.66s
I1109 07:04:37.985107 133398888275968 test_util.py:2472] time(__main__.ModelBuilderTF2Test.test_create_center_net_model0 (customi
[       OK ] ModelBuilderTF2Test.test_create_center_net_model0 (customize_head_params=True)
[ RUN      ] ModelBuilderTF2Test.test_create_center_net_model1 (customize_head_params=False)
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_create_center_net_model1 (customize_head_params=False)): 0.36s
I1109 07:04:38.347935 133398888275968 test_util.py:2472] time(__main__.ModelBuilderTF2Test.test_create_center_net_model1 (customi
[       OK ] ModelBuilderTF2Test.test_create_center_net_model1 (customize_head_params=False)
[ RUN      ] ModelBuilderTF2Test.test_create_center_net_model_from_keypoints
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_create_center_net_model_from_keypoints): 0.35s
I1109 07:04:38.694614 133398888275968 test_util.py:2472] time(__main__.ModelBuilderTF2Test.test_create_center_net_model_from_keyp
```

8) Navigate to /mydrive/customTF2/data/ and Unzip the *images.zip* and *annotations.zip* files into the *data* folder

```
%cd /mydrive/customTF2/data/

# unzip the datasets and their contents so that they are now in /mydrive/customTF2/data/ folder
!unzip /mydrive/customTF2/images.zip -d .
!unzip /mydrive/customTF2/annotations.zip -d .
```

```
/content/gdrive/My Drive/customTF2/data
Archive:  /mydrive/customTF2/images.zip
replace ./images/a 1.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
  inflating: ./images/a 1.jpg
  inflating: ./images/a10.jpg
  inflating: ./images/a2.jpg
  inflating: ./images/a3.jpg
  inflating: ./images/a4.jpg
  inflating: ./images/a5.jpg
  inflating: ./images/a6.jpg
  inflating: ./images/a7.jpg
  inflating: ./images/a8.jpg
  inflating: ./images/a9.jpg
  inflating: ./images/b1.jpg
  inflating: ./images/b10.jpg
  inflating: ./images/b2.jpg
  inflating: ./images/b3.jpg
  inflating: ./images/b4.jpg
  inflating: ./images/b5.jpg
  inflating: ./images/b6.jpg
  inflating: ./images/b7.jpg
  inflating: ./images/b8.jpg
  inflating: ./images/b9.jpg
  inflating: ./images/c1.jpg
  inflating: ./images/c10.jpg
  inflating: ./images/c2.jpg
  inflating: ./images/c3.jpg
  inflating: ./images/c4.jpg
  inflating: ./images/c5.jpg
  inflating: ./images/c6.jpg
  inflating: ./images/c7.jpg
  inflating: ./images/c8.jpg
  inflating: ./images/c9.jpg
  inflating: ./images/d1.jpg
  inflating: ./images/d10.jpg
  inflating: ./images/d2.jpg
  inflating: ./images/d3.jpg
  inflating: ./images/d4.jpg
  inflating: ./images/d5.jpg
  inflating: ./images/d6.jpg
  inflating: ./images/d7.jpg
  inflating: ./images/d8.jpg
  inflating: ./images/d9.jpg
  inflating: ./images/e1.jpg
  inflating: ./images/e10.jpg
  inflating: ./images/e2.jpg
  inflating: ./images/e3.jpg
  inflating: ./images/e4.jpg
```

9) Create test_labels & train_labels

Current working directory is /mydrive/customTF2/data/

Divide annotations into test_labels(20%) and train_labels(80%).

```
#creating two dir for training and testing
!mkdir test_labels train_labels

# lists the files inside 'annotations' in a random order (not really random, by their hash value instead)
# Moves the first 274/1370 labels (20% of the labels) to the testing dir: `test_labels`
!ls annotations/* | sort -R | head -274 | xargs -I{} mv {} test_labels/


# Moves the rest of the labels ( 1096 labels ) to the training dir: `train_labels`
!ls annotations/* | xargs -I{} mv {} train_labels/
```

```
mkdir: cannot create directory ‘test_labels’: File exists
mkdir: cannot create directory ‘train_labels’: File exists
ls: cannot access 'annotations/*': No such file or directory
```

The working directory at this point:

```
mydrive➔customTF2 /
        └── data (cwd)
            ├── images
            │   ├── image_1.jpg
            │   └── ...
            ├── annotations
            │   ├──
            │   └── ...
            ├── train_labels //contains the labels only
            │   ├── image_1.xml
            │   └── ...
            └── test_labels //contains the labels only
                ├── image_50.xml
                └── ...
```

10) Create the CSV files and the "label_map.pbtxt" file

Current working directory is /mydrive/customTF2/data/

Run xml_to_csv script below to create ***test_labels.csv*** and ***train_labels.csv***

This also creates the ***label_map.pbtxt*** file using the classes mentioned in the xml files.

```
#adjusted from: https://github.com/datitran/raccoon_dataset
def xml_to_csv(path):
  classes_names = []
  xml_list = []

  for xml_file in glob.glob(path + '/*.xml'):
    tree = ET.parse(xml_file)
    root = tree.getroot()
    for member in root.findall('object'):
      classes_names.append(member[0].text)
      value = (root.find('filename').text  ,
               int(root.find('size')[0].text),
               int(root.find('size')[1].text),
               member[0].text,
               int(member[4][0].text),
               int(member[4][1].text),
               int(member[4][2].text),
               int(member[4][3].text))
      xml_list.append(value)
  column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']
  xml_df = pd.DataFrame(xml_list, columns=column_name)
  classes_names = list(set(classes_names))
  classes_names.sort()
  return xml_df, classes_names

for label_path in ['train_labels', 'test_labels']:
  image_path = os.path.join(os.getcwd(), label_path)
  xml_df, classes = xml_to_csv(label_path)
  xml_df.to_csv(f'{label_path}.csv', index=None)
  print(f'Successfully converted {label_path} xml to csv.')

label_map_path = os.path.join("label_map.pbtxt")
pbtxt_content = ""

for i, class_name in enumerate(classes):
    pbtxt_content = (
        pbtxt_content
        + "item {{\n    id: {0}\n    name: '{1}'\n}}\n\n".format(i + 1, class_name)
    )
pbtxt_content = pbtxt_content.strip()
with open(label_map_path, "w") as f:
    f.write(pbtxt_content)
    print('Successfully created label_map.pbtxt ')
```

```
 Successfully converted train_labels xml to csv.
 Successfully converted test_labels xml to csv.
 Successfully created label_map.pbtxt
```

The working directory at this point:

```
mydrive → customTF2 /
          └── data/ (cwd)
              ├── images/
              │   └── ...
              ├── annotations/
              │   └── ...
              ├── train_labels/
              │   └── ...
              ├── test_labels/
              │   └── ...
              ├── label_map.pbtxt
              ├── test_labels.csv
              └── train_labels.csv
```

11) Create train.record & test.record files

Current working directory is /mydrive/customTF2/data/

Run the *generate_tfrecord.py* script to create *train.record* and *test.record* files

```
#Usage:
#!python generate_tfrecord.py output.csv output_pb.txt /path/to/images output.tfrecords

#For train.record
!python /mydrive/customTF2/generate_tfrecord.py train_labels.csv  label_map.pbtxt images/ train.record

#For test.record
!python /mydrive/customTF2/generate_tfrecord.py test_labels.csv  label_map.pbtxt images/ test.record
```

```
2023-10-31 03:01:55.935225: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Atte
2023-10-31 03:01:55.935278: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attem
2023-10-31 03:01:55.935316: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: At
2023-10-31 03:01:57.559093: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
groups: 100% 208/208 [00:00<00:00, 252.70it/s]
Successfully created the TFRecords: /content/gdrive/MyDrive/customTF2/data/train.record
2023-10-31 03:02:04.250825: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Atte
2023-10-31 03:02:04.250877: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attem
2023-10-31 03:02:04.250914: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: At
2023-10-31 03:02:05.903287: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
groups: 100% 260/260 [00:00<00:00, 488.55it/s]
Successfully created the TFRecords: /content/gdrive/My Drive/customTF2/data/test.record
```

12) Download pre-trained model checkpoint

Current working directory is /mydrive/customTF2/data/

Download **ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz** into the ***data*** folder & unzip it.

A list of detection checkpoints for tensorflow 2.x can be found here.

```
#Download the pre-trained model ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz into the data folder & unzip it.

!wget http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
!tar -xzvf ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
```

```
--2023-10-31 03:02:22--  http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco1
Resolving download.tensorflow.org (download.tensorflow.org)... 64.233.191.207, 173.194.74.207, 173.194.192.207, ...
Connecting to download.tensorflow.org (download.tensorflow.org)|64.233.191.207|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20515344 (20M) [application/x-tar]
Saving to: ‘ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz.2’

ssd_mobilenet_v2_fp 100%[===================>]  19.56M  84.6MB/s    in 0.2s

2023-10-31 03:02:23 (84.6 MB/s) - ‘ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz.2’ saved [20515344/20515344]

ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0.data-00000-of-00001
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/checkpoint
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0.index
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/pipeline.config
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/saved_model.pb
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/variables/
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/variables/variables.data-00000-of-00001
ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model/variables/variables.index
```

13) Get the model pipeline config file, make changes to it and put it inside the *data* folder

Current working directory is /mydrive/customTF2/data/

Download **ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.config** from */content/models/research/object_detection/configs/tf2*. Make the required changes to it and upload it to the */mydrive/custom/data* folder.

OR

Edit the config file from */content/models/research/object_detection/configs/tf2* in colab and copy the edited config file to the */mydrive/customTF2/data* folder.

You can also find the pipeline config file inside the model checkpoint folder we just downloaded in the previous step.

You need to make the following changes:

- change ***num_classes*** to number of your classes.
- change ***test.record*** path, ***train.record*** path & ***labelmap*** path to the paths where you have created these files (paths should be relative to your current working directory while training).
- change ***fine_tune_checkpoint*** to the path of the directory where the downloaded checkpoint from step 12 is.
- change ***fine_tune_checkpoint_type*** with value **classification** or **detection** depending on the type..
- change ***batch_size*** to any multiple of 8 depending upon the capability of your GPU. (eg:- 24,128,...,512).Mine is set to 64.
- change ***num_steps*** to number of steps you want the detector to train.

```
#copy the edited config file from the configs/tf2 directory to the data/ folder in your drive

!cp /content/models/research/object_detection/configs/tf2/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.config /mydrive/customTF2/data
```

15) Train the model

Navigate to the ***object_detection*** folder in colab vm

```
%cd /content/models/research/object_detection
```

```
/content/models/research/object_detection
```

15 (a) Training using model_main_tf2.py

Here **{PIPELINE_CONFIG_PATH}** points to the pipeline config and **{MODEL_DIR}** points to the directory in which training checkpoints and events will be written.

For best results, you should stop the training when the loss is less than 0.1 if possible, else train the model until the loss does not show any significant change for a while. The ideal loss should be below 0.05 (Try to get the loss as low as possible without overfitting the model. Don't go too high on training steps to try and lower the loss if the model has already converged viz. if it does not reduce loss significantly any further and takes a while to go down.)

```
# Run the command below from the content/models/research/object_detection directory
"""
PIPELINE_CONFIG_PATH=path/to/pipeline.config
MODEL_DIR=path to training checkpoints directory
NUM_TRAIN_STEPS=1500
SAMPLE_1_OF_N_EVAL_EXAMPLES=1

python model_main_tf2.py -- \
  --model_dir=$MODEL_DIR --num_train_steps=$NUM_TRAIN_STEPS \
  --sample_1_of_n_eval_examples=$SAMPLE_1_OF_N_EVAL_EXAMPLES \
  --pipeline_config_path=$PIPELINE_CONFIG_PATH \
  --alsologtostderr
"""

!python model_main_tf2.py --pipeline_config_path=/mydrive/customTF2/data/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.config --model_d:
```

```
2023-10-31 03:07:20.346862: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: At
2023-10-31 03:07:20.346912: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Att
2023-10-31 03:07:20.346952: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory:
2023-10-31 03:07:21.928467: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
2023-10-31 03:07:27.309633: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:47] Overriding orig_value setting because t
INFO:tensorflow:Using MirroredStrategy with devices ('/job:localhost/replica:0/task:0/device:GPU:0',)
I1031 03:07:27.311098 132153845276672 mirrored_strategy.py:423] Using MirroredStrategy with devices ('/job:localhost/replica:0/ta
INFO:tensorflow:Maybe overwriting train_steps: None
I1031 03:07:27.343191 132153845276672 config_util.py:552] Maybe overwriting train_steps: None
INFO:tensorflow:Maybe overwriting use_bfloat16: False
I1031 03:07:27.343372 132153845276672 config_util.py:552] Maybe overwriting use_bfloat16: False
WARNING:tensorflow:From /usr/local/lib/python3.10/dist-packages/object_detection/model_lib_v2.py:563: StrategyBase.experimental_d
Instructions for updating:
Use `tf.data.Dataset.interleave(map_func, cycle_length, block_length, num_parallel_calls=tf.data.AUTOTUNE)` instead. If sloppy ex
WARNING:tensorflow:From /usr/local/lib/python3.10/dist-packages/object_detection/builders/dataset_builder.py:235: DatasetV1.map_w
Instructions for updating:
Use `tf.data.Dataset.map()
W1031 03:07:27.442149 132153845276672 deprecation.py:50] From /usr/local/lib/python3.10/dist-packages/object_detection/builders/d
Instructions for updating:
Use `tf.data.Dataset.map()
Traceback (most recent call last):
  File "/content/models/research/object_detection/model_main_tf2.py", line 114, in <module>
    tf.compat.v1.app.run()
  File "/usr/local/lib/python3.10/dist-packages/tensorflow/python/platform/app.py", line 36, in run
    _run(main=main, argv=argv, flags_parser=_parse_flags_tolerate_undef)
  File "/usr/local/lib/python3.10/dist-packages/absl/app.py", line 308, in run
    _run_main(main, args)
  File "/usr/local/lib/python3.10/dist-packages/absl/app.py", line 254, in _run_main
    sys.exit(main(argv))
  File "/content/models/research/object_detection/model_main_tf2.py", line 105, in main
    model_lib_v2.train_loop(
  File "/usr/local/lib/python3.10/dist-packages/object_detection/model_lib_v2.py", line 563, in train_loop
    train_input = strategy.experimental_distribute_datasets_from_function(
  File "/usr/local/lib/python3.10/dist-packages/tensorflow/python/util/deprecation.py", line 383, in new_func
    return func(*args, **kwargs)
  File "/usr/local/lib/python3.10/dist-packages/tensorflow/python/distribute/distribute_lib.py", line 1561, in experimental_distr
    return self.distribute_datasets_from_function(dataset_fn, options)
  File "/usr/local/lib/python3.10/dist-packages/tensorflow/python/distribute/distribute_lib.py", line 1552, in distribute_dataset
    return self._extended._distribute_datasets_from_function(  # pylint: disable=protected-access
  File "/usr/local/lib/python3.10/dist-packages/tensorflow/python/distribute/mirrored_strategy.py", line 613, in _distribute_data
```

RETRAINING THE MODEL (in case you get disconnected)

If you get disconnected or lose your session on colab vm, you can start your training where you left off as the checkpoint is saved on your drive inside the ***training*** folder. To restart the training simply run **steps 1, 5, 6, 7, 14, and 15**

Note that since we have all the files required for training like the record files,our edited pipeline config file,the label_map file and the model checkpoint folder, therefore we do not need to create these again.

The model_main_tf2.py script saves the checkpoint every 1000 steps. The training automatically restarts from the last saved checkpoint itself.

However, if you see that it doesn't restart training from the last checkpoint you can make 1 change in the pipeline config file. Change **fine_tune_checkpoint** to where your latest trained checkpoints have been written and have it point to the latest checkpoint as shown below:

```
fine_tune_checkpoint: "/mydrive/customTF2/training/ckpt-X" (where ckpt-X is the latest checkpoint)
```

16) Test your trained model

Export inference graph

Current working directory is /content/models/research/object_detection

Klik dua kali (atau tekan Enter) untuk mengedit

```
##Export inference graph
!python exporter_main_v2.py --trained_checkpoint_dir=/mydrive/customTF2/training --pipeline_config_path=/content/gdrive/MyDrive/customTF
```

```
2023-10-01 11:49:56.739482: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
2023-10-01 11:50:00.228668: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:47] Overriding orig_value setting because t

results = tf.nest.map_structure(tf.stop_gradient, tf.map_fn(fn, elems))
I1001 11:50:05.148887 137212939849728 api.py:460] feature_map_spatial_dims: [(40, 40), (20, 20), (10, 10), (5, 5), (3, 3)]
I1001 11:50:20.763834 137212939849728 api.py:460] feature_map_spatial_dims: [(40, 40), (20, 20), (10, 10), (5, 5), (3, 3)]
I1001 11:50:28.048033 137212939849728 signature_serialization.py:148] Function `call_func` contains input name(s) resource with u
I1001 11:50:30.141257 137212939849728 api.py:460] feature_map_spatial_dims: [(40, 40), (20, 20), (10, 10), (5, 5), (3, 3)]
WARNING:tensorflow:Skipping full serialization of Keras layer <object_detection.meta_architectures.ssd_meta_arch.SSDMetaArch obje
W1001 11:50:36.957709 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <object_detection.meta_architec
WARNING:tensorflow:Skipping full serialization of Keras layer <keras.src.layers.convolutional.separable_conv2d.SeparableConv2D ob
W1001 11:50:37.236122 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <keras.src.layers.convolutional
WARNING:tensorflow:Skipping full serialization of Keras layer <object_detection.core.freezable_batch_norm.FreezableBatchNorm obje
W1001 11:50:37.236341 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <object_detection.core.freezabl
WARNING:tensorflow:Skipping full serialization of Keras layer <keras.src.layers.core.lambda_layer.Lambda object at 0x7ccacffd00d0
W1001 11:50:37.236450 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <keras.src.layers.core.lambda_l
WARNING:tensorflow:Skipping full serialization of Keras layer <keras.src.layers.convolutional.separable_conv2d.SeparableConv2D ob
W1001 11:50:37.236536 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <keras.src.layers.convolutional
WARNING:tensorflow:Skipping full serialization of Keras layer <object_detection.core.freezable_batch_norm.FreezableBatchNorm obje
W1001 11:50:37.236634 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <object_detection.core.freezabl
WARNING:tensorflow:Skipping full serialization of Keras layer <keras.src.layers.core.lambda_layer.Lambda object at 0x7ccae0791030
W1001 11:50:37.236710 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <keras.src.layers.core.lambda_l
WARNING:tensorflow:Skipping full serialization of Keras layer <keras.src.layers.convolutional.separable_conv2d.SeparableConv2D ob
W1001 11:50:37.236797 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <keras.src.layers.convolutional
WARNING:tensorflow:Skipping full serialization of Keras layer <object_detection.core.freezable_batch_norm.FreezableBatchNorm obje
W1001 11:50:37.236885 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <object_detection.core.freezabl
WARNING:tensorflow:Skipping full serialization of Keras layer <keras.src.layers.core.lambda_layer.Lambda object at 0x7cca68f13430
W1001 11:50:37.236960 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <keras.src.layers.core.lambda_l
WARNING:tensorflow:Skipping full serialization of Keras layer <keras.src.layers.convolutional.separable_conv2d.SeparableConv2D ob
W1001 11:50:37.237032 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <keras.src.layers.convolutional
WARNING:tensorflow:Skipping full serialization of Keras layer <object_detection.core.freezable_batch_norm.FreezableBatchNorm obje
W1001 11:50:37.237106 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <object_detection.core.freezabl
WARNING:tensorflow:Skipping full serialization of Keras layer <keras.src.layers.core.lambda_layer.Lambda object at 0x7cca68f125c0
W1001 11:50:37.237182 137212939849728 save_impl.py:66] Skipping full serialization of Keras layer <keras.src.layers.core.lambda_l
WARNING:tensorflow:Skipping full serialization of Keras layer <object_detection.core.freezable_batch_norm.FreezableBatchNorm obje
```

Test your trained Object Detection model on images

Current working directory is /content/models/research/object_detection

```
# Different font-type for labels text.(This step is optional)
!wget https://freefontsdownload.net/download/160187/arial.zip
!unzip arial.zip -d .

%cd utils/
!sed -i "s/font = ImageFont.truetype('arial.ttf', 24)/font = ImageFont.truetype('arial.ttf', 50)/" visualization_utils.py
%cd ..
```

```
--2023-10-01 11:51:40--  https://freefontsdownload.net/download/160187/arial.zip
Resolving freefontsdownload.net (freefontsdownload.net)... 172.67.180.27, 104.21.75.182, 2606:4700:3036::6815:4bb6, ...
Connecting to freefontsdownload.net (freefontsdownload.net)|172.67.180.27|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 172821 (169K) [application/force-download]
Saving to: ‘arial.zip’

arial.zip           100%[===================>] 168.77K  --.-KB/s    in 0.07s

2023-10-01 11:51:41 (2.28 MB/s) - ‘arial.zip’ saved [172821/172821]

Archive:  arial.zip
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@##################################@
@##        Downloaded from         #@
@##                                #@
@#   www.FreeFontsDownload.net    #@
@##                                #@
```

```
     @## ---------- More site -------- #@
     @##      https://funnytv.net      #@
     @##                               #@
     @##                               #@
     @###################################@
     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
       inflating: ./www.freefontsdownload.net.url
      extracting: ./arial.png
       inflating: ./arial.ttf
       inflating: ./freefontsdownload.txt
     /content/models/research/object_detection/utils
     /content/models/research/object_detection
```

```
#Loading the saved_model
import tensorflow as tf
import time
import numpy as np
import warnings
warnings.filterwarnings('ignore')
from PIL import Image
from google.colab.patches import cv2_imshow
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils

IMAGE_SIZE = (2, 2) # Output display size as you want
import matplotlib.pyplot as plt
PATH_TO_SAVED_MODEL="/mydrive/customTF2/data/inference_graph/saved_model"
print('Loading model...', end='')

# Load saved model and build the detection function
detect_fn=tf.saved_model.load(PATH_TO_SAVED_MODEL)
print('Done!')

#Loading the label_map
category_index=label_map_util.create_category_index_from_labelmap("/mydrive/customTF2/data/label_map.pbtxt",use_display_name=True)
#category_index=label_map_util.create_category_index_from_labelmap([path_to_label_map],use_display_name=True)
```

```
def load_image_into_numpy_array(path):

    return np.array(Image.open(path))

image_path = "/content/gdrive/MyDrive/fff.jpg"
#print('Running inference for {}... '.format(image_path), end='')

image_np = load_image_into_numpy_array(image_path)

# The input needs to be a tensor, convert it using `tf.convert_to_tensor`.
input_tensor = tf.convert_to_tensor(image_np)
# The model expects a batch of images, so add an axis with `tf.newaxis`.
input_tensor = input_tensor[tf.newaxis, ...]

detections = detect_fn(input_tensor)

# All outputs are batches tensors.
# Convert to numpy arrays, and take index [0] to remove the batch dimension.
# We're only interested in the first num_detections.
num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
               for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be ints.
detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

image_np_with_detections = image_np.copy()
```

```
viz_utils.visualize_boxes_and_labels_on_image_array(
      image_np_with_detections,
      detections['detection_boxes'],
      detections['detection_classes'],
      detections['detection_scores'],
      category_index,
      use_normalized_coordinates=True,
      max_boxes_to_draw=200,
      min_score_thresh=.4, # Adjust this value to set the minimum probability boxes to be classified as True
      agnostic_mode=False)
%matplotlib inline
plt.figure(figsize=IMAGE_SIZE, dpi=200)
plt.axis("off")
plt.imshow(image_np_with_detections)
plt.show()
```

Lampiran 2 Hasil Testing pada SSD *MobileNet*

lampiran 3 Perintah Training Medel Deteksi YOLOv7

```
#Download YOLOv7 repository and install requirements
!git clone https://github.com/WongKinYiu/yolov7
%cd yolov7
!pip install -r requirements.txt
```

```
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from -r requirements.
Requirement already satisfied: ipython in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (lin
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line
Collecting thop (from -r requirements.txt (line 36))
  Downloading thop-0.1.1.post2209072238-py3-none-any.whl (15 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from request
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->-
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.2
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.7.0->
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.7.0->r
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.7.0->
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.7.0->r
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->-r r
Collecting jedi>=0.16 (from ipython->-r requirements.txt (line 34))
  Downloading jedi-0.19.1-py2.py3-none-any.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 4.1 MB/s eta 0:00:00
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist-packages (from ipython->-r requiremen
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/dist-packages (from ipython->-r requirem
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.10/dist-packages (from ipython->-r requi
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pygments in /usr/local/lib/python3.10/dist-packages (from ipython->-r requirement
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-packages (from ipython->-r requirement
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.10/dist-packages (from ipython->-r re
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-packages (from ipython->-r requirem
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-au
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-aut
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-
Requirement already satisfied: parso<0.9.0,>=0.8.3 in /usr/local/lib/python3.10/dist-packages (from jedi>=0.16->
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.10/dist-packages (from pexpect>4.3->ipy
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/dist-packages (from prompt-toolkit!=3.0.0,!=
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch!=1.12.
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modu
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthli
Installing collected packages: jedi, thop
Successfully installed jedi-0.19.1 thop-0.1.1.post2209072238
```

▼ Default title text

```
#@title Default title text

%cd /content
!curl -L "https://github.com/madebymesm/bismillah/raw/main/dataRoboflow.zip"  > data_yolov7.zip; unzip data_yolov7.zip
```

```
inflating: train/images/h9_jpg.rf.eb65fa9dcf82a943de756d42cbf96a91.jpg
inflating: train/images/i1_jpg.rf.04bfd2e56c71eacb85faa19c483f6e29.jpg
inflating: train/images/i2_jpg.rf.ccbdcd44b5296e93060352c7918e0a24.jpg
inflating: train/images/i3_jpg.rf.3f11abf25557a905d891c99924459e86.jpg
inflating: train/images/i4_jpg.rf.d5030dd0e2ae2db673bfe24160c14502.jpg
inflating: train/images/i5_jpg.rf.07f1809487e82599df756128e5a49a86.jpg
inflating: train/images/i6_jpg.rf.20f005293a4d7a3f53f0b7312b1a9cbc.jpg
inflating: train/images/i7_jpg.rf.33a1c45dd4905915cd22e0ab232d3b79.jpg
inflating: train/images/i8_jpg.rf.8afbc6787d3e3f842c824c10b3f64328.jpg
inflating: train/images/i9_jpg.rf.ed875d3fcf5baee0561a5b4d4eb4239e.jpg
inflating: train/images/J1_jpg.rf.628e5f0d16baaa75b849b6274470840e.jpg
inflating: train/images/J10_jpg.rf.bb662fed57202138460b552fd9e9f972.jpg
inflating: train/images/J2_jpg.rf.a89374e2001fab66f9e27e74f7f22b6c.jpg
inflating: train/images/J3_jpg.rf.d44eee07a6097ae9c61d3d7e2797dc55.jpg
inflating: train/images/J4_jpg.rf.269a8e4d73af485098fba862a3fd6753.jpg
inflating: train/images/J5_jpg.rf.7467cc7ef056602408f40da0de8e09f0.jpg
inflating: train/images/J6_jpg.rf.83b925d5b9bed384a59ea82341f8ee95.jpg
inflating: train/images/J7_jpg.rf.2b5bf61f17ebcee57f2f4cc919fae1c8.jpg
inflating: train/images/J8_jpg.rf.cf7282fe7d4125673e7996eaf1718d36.jpg
inflating: train/images/k10_jpg.rf.881129830b9e455cff876d0effdb982e.jpg
inflating: train/images/k2_jpg.rf.68a1935aef1fabc96e23509be485828c.jpg
inflating: train/images/k3_jpg.rf.c914324ceee2a8168d63cdb5ade5b449.jpg
inflating: train/images/k5_jpg.rf.dc48ed5b4eb05f675eeaea46646e9037.jpg
inflating: train/images/k8_jpg.rf.e1a66f7b93b17754d9637a197e1a7895.jpg
inflating: train/images/k9_jpg.rf.c76171947f5cd5824148e027f7989b1c.jpg
inflating: train/images/L1_jpg.rf.6dbaefaa175f6a319508529254bc9bc8.jpg
inflating: train/images/M1_jpg.rf.9a88360a98e0f927a89fb970637b50c0.jpg
```

Prepare image path in txt

```
#@title Prepare image path in txt
import os


train_img_path ="/content/train/images"
val_img_path ="/content/valid/images"


%cd /content
```

```
/content
```

```
#Training images
with open('train.txt', "a+") as f:
  img_list = os.listdir(train_img_path)
  for img in img_list:
    f.write(os.path.join(train_img_path,img+'\n'))
  print("Done")
```

```
Done
```

```
#Validation images
with open('val.txt', "a+") as f:
  img_list = os.listdir(val_img_path)
  for img in img_list:
    f.write(os.path.join(val_img_path,img+'\n'))
  print("Done")
```

```
Done
```

```
%cp /content/yolov7/data/coco.yaml /content/yolov7/data/custom.yaml


#download COCO starting checkpoint
%cd /content/yolov7
!wget  "https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7.pt"
```

```
/content/yolov7
--2023-11-09 07:20:34--  https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7.pt
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/511187726/b0243edf-9fb0-433
--2023-11-09 07:20:34--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/511187726/b0
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.111.133, 185.199.109.133, 185.1
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 75587165 (72M) [application/octet-stream]
Saving to: 'yolov7.pt'

yolov7.pt           100%[===================>]  72.08M   277MB/s    in 0.3s

2023-11-09 07:20:35 (277 MB/s) - 'yolov7.pt' saved [75587165/75587165]
```

```
%cp /content/yolov7/cfg/training/yolov7.yaml /content/yolov7/cfg/training/custom_yolov7.yaml
```

```
!python train.py --batch 16 --cfg  /content/yolov7/cfg/training/custom_yolov7.yaml --epochs 1000 --data /content/yolov
```

```
     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
   989/999     11.7G   0.01927  0.004522  0.008086   0.03188        15       640: 100% 13/13 [00:11<00:00,  1.17
                Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00
                  all          49          49       0.934       0.994       0.996        0.81

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
   990/999     11.7G   0.02615  0.004575   0.01236   0.04309        16       640: 100% 13/13 [00:11<00:00,  1.14
                Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00
                  all          49          49       0.938       0.992       0.996        0.81

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
   991/999     11.7G   0.02332  0.004538   0.01178   0.03963        16       640: 100% 13/13 [00:11<00:00,  1.17
                Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 2/2 [00:01<00
                  all          49          49       0.941       0.992       0.996        0.81

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
   992/999     11.7G   0.01834  0.004827  0.007013   0.03018         9       640: 100% 13/13 [00:10<00:00,  1.18
                Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 2/2 [00:00<00
                  all          49          49       0.941       0.993       0.996        0.805

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
```

```
#Run
!python detect.py --weights /content/yolov7/runs/train/exp/weights/best.pt --source /content/drive/MyDrive/images/e4.jpg
```

```
Namespace(weights=['/content/yolov7/runs/train/exp/weights/best.pt'], source='/content/drive/MyDrive/images/f2.jpg
YOLOR 🚀 v0.1-126-g84932d7 torch 2.0.1+cu118 CUDA:0 (Tesla T4, 15101.8125MB)

Fusing layers...
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
IDetect.fuse
/usr/local/lib/python3.10/dist-packages/torch/functional.py:504: UserWarning: torch.meshgrid: in an upcoming relea
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
Model Summary: 314 layers, 36616622 parameters, 6194944 gradients, 103.6 GFLOPS
 Convert model to Traced-model...
 traced_script_module saved!
 model is traced!

1 F, Done. (22.1ms) Inference, (1.5ms) NMS
 The image with the result is saved in: runs/detect/exp72/f2.jpg
Done. (0.042s)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
#display inference on all test images

import glob
from IPython.display import Image, display
i=0
limit = 10000 #max images to print
for imageName in glob.glob('/content/yolov7/runs/detect/exp44/e4.jpg'):

  if i < limit:
    display(Image(filename=imageName))
    print("\n")
  i = i + 1
```

Lampiran 4 Hasil Testing pada model YOLOV7'