PENGENALAN HUKUM TAJWID PADA CITRA AL-QURAN MENGGUNAKAN SSD MOBILENET V2

Disusun Oleh:

N a m a	: Ihya Fathurrahman
NIM	: 17523126

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGENALAN HUKUM TAJWID PADA CITRA AL-QURAN MENGGUNAKAN SSD MOBILENET V2

TUGAS AKHIR

Disusun Oleh:

N a m a : Ihya Fathurrahman

NIM : 17523126

Yogyakarta, 21 Agustus 2021

Pembimbing,

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

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGENALAN HUKUM TAJWID PADA CITRA AL-QURAN MENGGUNAKAN SSD MOBILENET V2

TUGAS AKHIR

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

Yogyakarta, 21 Agustus 2021

Tim Penguji

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

Anggota 1

Aridhanyati Arifin, S.T., M.Cs. _____

Anggota 2

Kurniawan Dwi Irianto, S.T., M.Sc. _____

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

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

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Ihya Fathurrahman

NIM : 17523126

Tugas akhir dengan judul:

PENGENALAN HUKUM TAJWID PADA CITRA AL-QURAN MENGGUNAKAN SSD MOBILENET V2

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

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

Yogyakarta, 21 Agustus 2021

(Ihya Fathurrahman)

HALAMAN PERSEMBAHAN

Segala puji bagi Allah SWT tuhan semesta alam dan syukur saya panjatkan kepada-Nya. Atas kehendak dan karunia-Nya, saya dapat melewati masa-masa pengerjaan tugas akhir ini dan menyelesaikannya sebagai mahasiswa Program Studi Informatika Fakultas Teknologi Industri Universitas Islam Indonesia. Tugas akhir ini saya persembahkan kepada:

1. Ibu saya yang sayangi, ibu Lela Nurlaela yang senantiasa menyayangi saya, ibu Lela Nurlaela yang selalu mendukung dan melindungi saya dan ibu Lela Nurlaela yang selalu berkorban untuk kesuksesan saya saya.
2. Bapak saya yang sayangi, bapak Moh Iskandar yang setiap hari mengucurkan keringatnya untuk kebahagiaan keluarga, sosok yang tegas, bertanggung jawab, dan seorang penyayang.
3. Kedua adik saya Nuryafi Ilyasin dan Muhammad Irsyad Ulya yang selalu menghibur meskipun bukan setiap saat, menemani hari-hari libur ketika pulang ke rumah, dan menjadi contoh seseorang untuk saya sayangi.
4. Teman-teman di Informatika khususnya teman-teman seangkatan di PIXEL.
5. Ibu Arrie Kurniawardhani, S.Si., M.Kom. selalu dosen pembimbing, yang selalu membimbing, mengarahkan serta memberikan ilmu dan nasihatnya.
6. Keluarga besar Program Studi Informatika UII yang menjadi tempat saya mengembangkan diri dan menuntut ilmu.

HALAMAN MOTO

“Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya”

(QS. Al-Baqarah: 286)

“Maka, sesungguhnya bersama kesulitan ada kemudahan. Sesungguhnya, bersama kesulitan ada kemudahan.”

(QS. Al-Insyirah: 5-6)

"Betapa bodohnya manusia, Dia menghancurkan masa kini sambil mengkhawatirkan masa depan, tapi menangis di masa depan dengan mengingat masa lalunya"

- Ali bin Abi Thalib

"Seseorang yang tidak sanggup mengorbankan apapun, tidak akan bisa mengubah apa-apa"

- Armin Arlert

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh

Puji dan syukur saya panjatkan kepada Allah SWT, berkat kuasa, rahmat dan hidayah-Nya saya dapat menyelesaikan tugas akhir ini yang berjudul "Pengenalan Hukum Tajwid Pada Citra Al-Quran Menggunakan SSD MobileNet V2". Laporan ini dibuat dan disusun sebagai syarat untuk memperoleh gelar Sarjana atau Strata-1 (S1) dan juga bentuk penerapan ilmu yang telah didapatkan dan nilai keislaman sebagaimana visi Universitas Islam Indonesia. Tugas akhir ini tidak akan terselesaikan tanpa adanya faktor-faktor pendukung karena sadar akan kemampuan dan kuasa penulis bukanlah apa-apa. Dengan demikian, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Allah SWT berkat rahman rahim-Nya, rahmat, karunia, dan ampunan-Nya saya dapat menyelesaikan tugas akhir ini.
2. Kedua orang tua saya, Ibu Lela Nurlaela dan Bapak Moh Iskandar untuk doa, kepercayaan, motivasi, dukungan materi maupun emosional untuk menyelesaikan studi ini.
3. Kedua adik saya, Nuryafi Ilyasin dan Muhammad Irsyad Ulya atas doa, dukungan dan motivasinya.
4. Ibu Arrie Kurniawardhani, S.Si., M. Kom. yang selalu membimbing dan mengarahkan saya dalam menyelesaikan tugas akhir ini serta memberikan ilmu dan nasihatnya.
5. Keluarga besar Informatika Universitas Islam Indonesia yang telah memberikan kisah, cerita dan pengalaman.
6. Keluarga Himpunan Mahasiswa Teknik Informatika yang memberikan pengalaman dalam berinteraksi sosial, bertugas, kekeluargaan, bertanggung jawab, dan berorganisasi.
7. Teman-teman di PIXEL yang sudah menjadi tempat juga keluarga di kampus sekaligus perantauan.
8. Teman-teman satu konsentrasi di sistem cerdas khususnya teman-teman satu bimbingan yang sama-sama telah berjuang.
9. Teman-teman di VVIBU Grup tempat bermain, mengerjakan tugas, dan berbagi cerita.

10. Teman-teman di kontrakan Lugie yang dengan senang hati membiarkan saya datangi selama mengerjakan tugas akhir ini dan tempat paling tepat untuk bermain (gim) setelah banyak kelelahan mengerjakan tugas.
11. Teman-teman di kontrakan Andri (Syamil & Gozy) yang sudah bersedia untuk saya mampir dan tempati untuk waktu yang cukup lama selama mengerjakan tugas akhir ini.
12. Pihak lainnya yang dukungan dan kontribusinya mungkin tidak bisa saya lihat dengan kasatmata dan mungkin rasakan, tidak bisa saya ucapkan satu persatu rasa terima kasih saya pada semuanya, semoga kebaikan kalian dibalas Allah SWT.
13. Diri saya sendiri yang tidak menyerah dengan keadaan, yang sudah bangun karena urgensi-urgensi yang terjadi, dan tidak berlepas diri dari rahmat-Nya sehingga terselesaikannya tugas akhir ini.

Semoga segala bentuk dukungan, motivasi, pelajaran, yang diberikan pada penulis dihitung sebagai amal kebaikan dan dibalas Allah SWT. Dengan kesadaran kemampuan penulis, mohon maaf atas segala kekurangan yang ada pada proses ataupun pada penelitian ini. Semoga kekurangan-kekurangan yang ada dapat dijadikan pelajaran, adapun kelebihannya dapat dimanfaatkan dengan baik. Semoga siapapun yang membaca penelitian ini bisa mendapatkan manfaat sebagai apapun dan semoga dengan adanya penelitian ini harapan kedepannya akan bermanfaat bagi bangsa, agama dan negara.

Yogyakarta, 21 Agustus 2021

(Ihya Fathurrahman)

SARI

Al-Quran adalah kitab suci umat muslim yang menjadi pegangan dalam menjalankan kehidupannya. Membaca Al-Quran merupakan amalan yang luar biasa karena membaca satu huruf saja bernilai sebagai sepuluh kebaikan. Bagi seorang muslim membaca Al-Quran hukumnya *fardhu ain* yang artinya setiap individu muslim wajib membacanya. Di Indonesia sendiri angka tidak bisa mengaji masih cukup tinggi seperti yang dikatakan Wakil Ketua Dewan Masjid Indonesia (DMI) Syarifudin yaitu sebesar 65%, angka yang besar untuk tidak bisa membaca Al-Quran. Hal ini tentu saja menjadi sebuah keprihatinan. Untuk dapat membaca Al-Quran perlu dasar salah satunya adalah tajwid. Di zaman sekarang variasi pembelajaran sangatlah banyak, yang seharusnya dapat dimanfaatkan juga untuk pembelajaran tajwid. Dengan kecanggihan teknologi sekarang juga dapat disalurkan kepada pembelajaran, dalam hal ini khususnya pembelajaran tajwid. Dengan kecanggihan komputer sekarang dan serangkaian algortimanya terdapat istilah *deep learning* yang dapat digunakan untuk mendeteksi sebuah objek dan sudah banyak digunakan penelitian-penelitian dengan tema serupa. Kecanggihan ini dapat dimanfaatkan untuk pembelajaran tajwid. Dengan harapan dapat meningkatkan pembelajaran tajwid melalui deteksi objek, maka pada penelitian ini dilakukan percobaan untuk menerapkan salah satu algoritma dari *deep learning* yaitu SSD MobileNet V2. Tajwid yang akan menjadi objek deteksi pada penelitian ini adalah Mad Layyin dan Mad Arid Lissukun yang datanya didapatkan melalui tangkapan layar dari aplikasi Qur'an Kemenag dan didapat sebanyak 520 citra. Dengan bantuan *framework* TensorFlow akan dibuat tiga skema dengan konfigurasi yang menghasilkan tiga model yang berbeda pula untuk mengetahui perbandingan dari hasil ketiga model tersebut. Dari ketiga konfigurasi dihasilkan tiga model, setelah melewati pengujian model dari konfigurasi pertama menghasilkan akurasi sebesar 92,307%, model dari konfigurasi kedua sebesar 96,153%, dan model dari konfigurasi ketiga sebesar 90,384% dan. Diantara ketiga model tersebut, model dengan konfigurasi kedua memiliki kinerja yang lebih unggul dibanding model dengan konfigurasi pertama dan ketiga.

Kata kunci: Deteksi Objek, SSD MobileNet V2, Tajwid, TensorFlow

GLOSARIUM

AI	Cabang ilmu yang ada di bidang komputer, yang di dalamnya lebih menekankan pola pikir dan bekerja pada manusia, pengembangan dalam hal kecerdasan mesin, dan lain sebagainya.
API	Sekumpulan perintah, fungsi, serta protokol yang dapat digunakan oleh pemrogram saat membangun perangkat lunak.
Convergence	Kondisi ketika selama pelatihan nilai loss mengendap di sekitar rentang nilai akhir, dengan kata lain model tidak akan mengalami peningkatan lagi.
End-to-end	Metode komunikasi aman yang mencegah pihak ketiga mengakses data saat ditransfer dari satu sistem atau perangkat ke perangkat lainnya.
Fine Tuning	Menggunakan atau mengambil bobot model yang sudah terlatih untuk digunakan sebagai inisialisasi pada model yang akan dilatih.
Flowchart	Diagram yang mewakili algoritma, alir kerja atau proses, yang menampilkan langkah-langkah menggunakan simbol-simbol grafis yang dihubungkan dengan panah.
Framework	Sebuah kerangka kerja yang digunakan untuk mempermudah para pengembang perangkat lunak dalam membuat dan mengembangkan aplikasi.
Inference	Penggunaaan pada model *deep learning* terlatih untuk membuat prediksi terhadap data yang sebelumnya tidak terlihat.
Num steps	Jumlah langkah yang dilakukan dalam proses pelatihan sebuah model.
Overfitting	Kondisi sebuah model terlalu cocok/ahli dalam mengenali data latih, sehingga kurang dapat mengenali dari di luar data latih.
Stride	Parameter yang menentukan seberapa filter/kernel bergeser.
Tensorboard	*Tool* yang menyediakan pengukuran dan visualisasi yang diperlukan pada alur kerja pembelajaran mesin.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Al-Quran merupakan kitab suci yang menjadi pegangan umat islam dalam menjalankan kehidupannya. Membaca Al-Quran memiliki amalan yang luar biasa bukan sekali membaca dihitung sebagai kebaikan, akan tetapi satu huruf pada satu kata dihitung sepuluh kebaikan. Bahkan dalam masa atau proses belajar membacanya dan terbata-bata akan mendapatkan pahala dua kali, sesuai dalam satu hadits yang menyebutkan: "Orang yang ahli dalam Alquran akan bersama para malaikat pencatat yang mulia lagi benar. Dan orang-orang yang terbata-bata membaca Alquran serta bersusah payah (mempelajarinya), maka baginya pahala dua kali." (HR. Bukhari, Muslim, Abu Dawud). Membaca Al-Quran hukumnya wajib ain yang artinya setiap individu muslim wajib membaca Al-Quran dengan baik dan benar, dan agar bacaannya baik dan benar perlu mengetaui tajwid (Mistari, Asmara, & Hakkun, 2012). Tajwid yang menjadi kaidah atau cara dalam membaca Al-Quran yang tentunya sangat penting. Namun faktanya, belum semua yang muslim bisa mengaji. Di Indonesia sendiri yang penduduknya mayoritas muslim masih banyak yang belum bisa mengaji. Dalam sebuah kunjungannya ke kantor PBNU, Wakil Ketua Dewan Masjid Indonesia (DMI) Syarifudin, mengatakan hanya 35% dari muslim di Indonesia yang bisa membaca Al-Quran yang artinya 65% lainnya tidak bisa baca Al-Quran (Yusuf, 2021). Tentu saja hal ini memprihatinkan mengingat jumlah muslim di Indonesia adalah mayoritas. Ditambah lagi yang seharusnya dengan kemajuan media pembelajaran yang ada dapat membantu mempelajari Al-Quran beserta tajwidnya, sehingga angka tidak bisa mengaji khusunya di Indonesia tidak sebesar itu.

Dengan variasi pembelajaran yang dikolaborasikan dengan kemajuan teknologi saat ini seharusnya dapat dibuat sistem atau pun alat untuk membuat seseorang lebih terbantu dalam belajar. *Mathway* adalah sebuah aplikasi yang dapat menyelesaikan sebuah persamaan matematika hanya dengan memotret persamaan tersebut (Mathway | About Us, n.d.), hal ini sangat membuat terpukau dan tentu saja hal yang potensial jika diterapkan pada hal lain dalam konteks pembelajaran. Dari hal tersebut penulis terpikirkan bagaimana kalau hal serupa diterapkan untuk pembelajaran tajwid dan dalam penelitian ini penulis akan mencoba sebuah algoritma dalam *deep learning* yang disebut SSD MobileNet V2 untuk diterapkan dalam pendeteksian tajwid. Pada penelitian-penelitian sebelumnya tajwid yang dideteksi (secara dominan) adalah jenis-jenis tajwid hukum nun mati, mim mati, dan tanwin. Diantaranya seperti

Ikhfa Hakiki, Idgham, Ikhfa Syafawi, Izhar Syafawi, Idgham Mutajanisain, dan lainnya. Maka pada penelitian ini dipilih hukum tajwid jenis mad, Mad Layyin dan Mad Aridlissukun adalah untuk memperluas, melengkapi juga menambah pengetahuan dalam pengenalan tajwid.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini adalah sebagai berikut:

a. Bagaimana percobaan pendeteksian pada tajwid yang dilakukan dengan menggunakan SSD MobileNet V2?
b. Bagaimana akurasi dari percobaan pada pendeteksian pada hukum tajwid?
c. Bagaimana kinerja dari percobaan pada pendeteksian pada hukum tajwid?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut:

a. Sumber data (citra) yang digunakan hanya diambil dari aplikasi Qur'an Kemenag.
b. Font dalam citra teks Al-Qur'an hanya dari font LMPQ Isep Misbah yang menjadi font standar pada cetakan Al-Quran di Indonesia.
c. Hukum tajwid yang dideteksi hanya pada Mad Layyin dan Mad Arid Lissukun.
d. Hanya dapat mendeteksi tajwid Mad Layyin dan Mad Arid Lissukun dengan lebih baik pada satu kata saja atau paling banyak dua kata.

1.4 Tujuan Penelitian

Tujuan pada penelitian ini adalah sebagai berikut:

a. Mengetahui perbandingan hasil dari percobaan percobaan yang dilakukan pada pendeteksian tajwid dengan menggunakan SSD MobileNet V2.
b. Menguji model untuk mengetahui akurasi dalam mengenali tajwid yang dideteksi.
c. Mengetahui hasil kinerja dari model dalam mengenali tajwid yang dideteksi.

1.5 Manfaat Penelitian

Manfaat pada penelitian ini adalah sebagai berikut:

a. Dapat dikembangkan untuk pembelajaran Al-Qur'an melalui *AI*.
b. Menjadi pengetahuan baru dalam pendeteksian tajwid, khususnya di Indonesia.
c. Hasil penelitian diharapkan dapat dikembangkan dan menjadi pembelajaran untuk meningkatkan kemampuan baca Al-Qur'an, khususnya di Indonesia.

1.6 Metodologi Penilitian

Untuk mencapai tujuan dari penelitian ini maka dibuat langkah-langkah sebagai berikut:

a. Pengumpulan Data

Pengumpulan data diperlukan sebagai bahan dalam pengembangan program yang akan dibuat untuk pengenalan hukum tajwid. Tahap ini juga mencakup akuisisi data, *crop*, dan pembagian data dengan kelas-kelasnya juga pembagian menjadi data latih, data validasi dan data uji.

b. Perancangan Sistem

Tahap ini merupakan gambaran dari perancangan sistem agar lebih tersusun dengan rapi dan terstruktur. Gambaran dari perancangan yang ada dituangkan pada sebuh diagram alir *flowchart*.

c. Implementasi Sistem

Implementasi menjadi tahap penerapan studi pustaka pada sistem yang dibuat untuk menyelesaikan masalah yang ada. Langkah ini meliputi *preprocessing* data, pemrosesan data, dengan keluaran berupa citra yang berhasil dideteksi tajwidnya.

d. Pengujian Sistem

Pengujian sistem adalah untuk memastikan bahwa sistem memenuhi apa yang menjadi tujuan penelitian yaitu membandingkan hasil pendeteksi dari ketiga model dengan konfigurasi masing-masing pada citra tajwid Al-Quran.

1.7 Sistematika Laporan

Pada penyusunan laporan tugas akhir ini dibagi menjadi beberapa bab yaitu sebagai berikut:

Bab I Pendahuluan

Pada bab ini berisi latar belakang yang mendasari penilitian ini, urgensi mempelajari tajwid, tingkat literasi mengaji di Indonesia, dan bagaimana penyelesaiannya. Dari latar belakang kemudian dibuat rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika laporan.

Bab II Landasan Teori

Bab ini berisi pembahasan singkat penelitian-penelitian terdahulu yang membahas pendeteksian tajwid. Kemudian ada teori-teori yang menjadi landasan yang diambil dari jurnal, artikel, buku, dan penelitian lainnya yang masih berkaitan. Teori-teori

yang dibahas diantaranya tajwid, citra, model pra telatih, *dataset* dan alat/*tools* pendukung dalam penelitian ini.

Bab III Metodologi Penelitian

Bab ini berisi tahapan dari penelitian yang akan dilakukan yang secara singkat digambarkan melalui diagram alir. Bab ini juga memuat tentang data, *preprocessing*, pelatihan sampai evaluasi.

Bab IV Hasil dan Pembahasan

Bab ini berisi hasil juga pembahasan dari tahap penelitian yang telah dilakukan, pengujian model dengan data uji dan mengevaluasi model.

Bab V Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil penelitian yang telah dilakukan berdasarkan dari rumusan masalah dan juga saran untuk meningkatkan dari sistem yang telah dibuat ataupun hal lainnya yang nantinya mungkin akan digunakan peneliti berikutnya.

BAB II
LANDASAN TEORI

2.1 Landasan Teori

2.1.1 Tajwid

Tajwid secara bahasa berasal dari kata jawwada – yujawwidu – tajwiidan yang memiliki arti memperbagus atau memperindah. Sedangkan secara istilah tajwid berarti mengeluarkan setiap huruf dari tempat keluarnya (makhraj) atau cara membaca (membunyikan) huruf-huruf Al-Quran dengan baik dan benar (Triyasyid, 2015). Adapun tajwid yang terdapat pada penelitian ini adalah sebagai berikut.

1. Mad Layyin

   Mad Layyin yaitu apabila ada huruf mad yang berupa wau sukun atau ya sukun didahului oleh huruf yang berharakat fathah. Mad Layyin memiliki panjang dua harakat atau empat harakat atau enam harakat (Triyasyid, 2015). Contoh bacaan Mad Layyin:

(a) (b)

Gambar 2.1 Contoh bacaan Mad Layyin (a) Contoh 1 dan (b) Contoh 2

2. Mad Arid Lissukun

   Mad Arid Lissukun yaitu apabila ada huruf mad yang terletak sebelum huruf akhir pada kata, memiliki panjang dua harakat atau empat harakat atau enam harakat (Triyasyid, 2015). Contoh bacaan Mad Arid Lissukun:

(a) (b)

Gambar 2.2 Contoh bacaan hukum Mad Arid Lissukun, (a) Contoh 1 dan (b) Contoh 2

2.1.2 Citra

Citra adalah fungsi kontinu yang memiliki intensitas cahaya yang ada pada bidang dua dimensi. Citra agar dapat diolah dengan komputer, maka harus direpresentasikan secara numerik dengan nilai-nilai diskrit. Sebuah citra dapat direpresentasikan dengan matriks dua dimensi. Jika sebuah citra direpresentasikan sebagai matriks *f(x, y)* dan M sebagai kolom dan N sebagai baris (Kusumanto & Tompunu, 2011), maka bisa digambarkan seperti berikut.

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

Gambar 2.3 Representasi citra dalam matriks

Sumber: Kusumanto (2011)

$0 \leq x \leq M\text{-}1$

$0 \leq y \leq N\text{-}1$

$0 \leq f(x, y) \leq G\text{-}1$

Dengan keterangan:

M = jumlah piksel baris (*row)* pada array citra

N = jumlah piksel kolom (*column*) pada array citra

G = nilai skala keabuan (*grayscale*)

Pada umumnya dalam pengolahan citra, citra dibagi menjadi tiga jenis yaitu *color image, black and white image* dan *binary image.*

1. *Color Image*

   Color Image adalah citra digital yang mempunyai informasi warna di setiap piksel citra tersebut. Terdapat beberapa sistem warna dalam citra warna diantaranya RGB, CMYK, HSV dan lain-lain. RGB sendiri adalah model warna yang terdiri dari *channel* merah, hijau, biru yang dengan nilainya masing-masing jika digabungkan akan membentuk susunan warna yang luas. Masing-masing *channel* memiliki nilai yang berada pada rentang 0 sampai 255 (Dewi, 2018). Jika dihitung secara keseluruhan kombinasi warna menggunakan tiga *channel* totalnya adalah

sebanyak 255^3 warna atau senilai dengan 16.581.375 warna (Kusumanto & Tompunu, 2011)

2. *Black and White Image*

   Black and White Image atau juga bisa disebut *grayscale image* (citra keabuan) adalah citra yang setiap pikselnya menggandung satu *layer* yang nilai intensitasnya beradapa pada 0 (hitam) sampai 255 (putih) (Dewi, 2018).

3. *Binary Image*

   Binary Image atau citra biner adalah citra yang hanya mempunyai dua nilai untuk satu pikselnya. Nilai 0 atau 1 (Dewi, 2018). Dengan nilai 0 atau 1 pada satu pikselnya membuat citra biner sangat efisien untuk hal penyimpanan. Citra biner bisa didapatkan dari proses pengolahan citra keabuan (Kusumanto & Tompunu, 2011).

2.1.3 Deteksi Objek

Deteksi Objek (*Object Detection*) adalah menentukan/mengenali keberadaan keberadaan suatu objek tertentu dan lokasi pada sebuah citra. Menurut Jared (2016), Deteksi Objek dibagi menjadi dua yaitu *Soft Detection* dan *Hard Detection*. *Soft Detection* yaitu mendeteksi objek, sedangkan *Hard Detection* adalah mendeteksi objek juga lokasi objek.

2.1.4 *Deep Learning*

Deep Learning atau pembelajaran mendalam adalah bagian dari pembelajaran mesin yang mempelajari/menggunakan beberepa level dari banyak lapisan dari informasi non-linier untuk ekstraksi ciri, transformasi, analisis pola dan klasifikasi (Deng & Yu, 2014). *Deep Learning* juga telah terbukti bermanfaat di banyak bidang seperti perangkat lunak, termasuk visi komputer, pemrosesan ucapan dan audio, pemrosesan bahasa alami, robotika, bioinformatika dan kimia, *video game*, mesin pencarian, iklan daring dan keuangan (Kim, 2016).

2.1.5 Model Pra Terlatih

Sederhananya Model Pra Terlatih (*Pre-Trained Model*) adalah Model yang dibuat oleh orang lain untuk memecahkan masalah yang mirip. Dari pada membuat model dari awal, penggunaannya dapat digunakan sebagai titik awal untuk penyelesaian masalah yang sama, pendeteksian objek pada citra misalnya. Ketika akan membuat sebuah model *deep learning* bisa digunakan dua pilihan, pertama membuat model tersebut dari awal atau kedua menggunakan Model Pra Terlatih yang sudah ada. Model Pra Terlatih mungkin tidak akan

memberikan akurasi 100%, namun penggunaan Model Pra Terlatih dapat menghemat waktu dan usaha untuk menemukan model yang tepat atau bagus.

Pada Model Pra Terlatih yang sebelumnya telah dilatih pada *dataset* dengan skala yang besar, saat penggunaannya kita dapat secara langsung menggunakan bobot dan arsitektur model tersebut pada masalah yang ingin kita selesaikan. Inilah yang disebut dengan *Transfer Learning* (Gupta, 2017). Sederhananya yaitu ketika kita mentransfer pengetahuan yang ada (untuk menyelesaikan suatu masalah) pada sebuah model untuk digunakan dalam penyelesaian masalah lainnya. Cara untuk melakukan *Transfer Learning* ialah dengan cara yang disebut *Fine Tuning*. Terdapat beberapa teknik untuk melakukan *Fine Tuning* seperti menggunakan *Learning Rate* yang lebih rendah atau melakukan *freeze* pada layer-layernya (Yu, 2016).

Model Pra Terlatih bisa didapatkan dari beberapa sumber tergantung pada *framework* yang digunakan. Beberapa *framework* yang populer diantaranya seperti *Caffe, Keras, TensorFlow, Torch, MxNet* dan lain-lain. Untuk TensorFlow sendiri memiliki banyak sekali Model Pra Terlatih yang disediakan di TensorFlow 2 *Detection Model Zoo*, di dalamnya terdapat banyak Model Pra Terlatih yang sudah dilatih pada COCO 2017 *dataset* (models/tf2_detection_zoo.md at master tensorflow/models, n.d.).

Tabel 2.1 Beberapa contoh Model Pra Terlatih yang ada di TensorFlow 2 *Detection Model Zoo*

No	Nama Model
1	SSD MobileNet v2 320x320
2	SSD MobileNet V1 FPN 640x640
3	SSD MobileNet V2 FPNLite 320x320
4	SSD MobileNet V2 FPNLite 640x640
5	SSD ResNet50 V1 FPN 640x640 (RetinaNet50)
6	SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)
7	SSD ResNet101 V1 FPN 640x640 (RetinaNet101)
8	SSD ResNet101 V1 FPN 1024x1024 (RetinaNet101)
9	SSD ResNet152 V1 FPN 640x640 (RetinaNet152)
10	SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)
11	Faster R-CNN ResNet50 V1 640x640

12	Faster R-CNN ResNet50 V1 1024x1024
13	Faster R-CNN ResNet50 V1 800x1333
14	Faster R-CNN ResNet101 V1 640x640
15	Faster R-CNN ResNet101 V1 1024x1024
16	Faster R-CNN ResNet101 V1 800x1333
17	Faster R-CNN ResNet152 V1 640x640
18	Faster R-CNN ResNet152 V1 1024x1024
19	Faster R-CNN ResNet152 V1 800x1333
20	Faster R-CNN Inception ResNet V2 640x640
21	Faster R-CNN Inception ResNet V2 1024x1024

2.1.6 SSD MobileNet V2

. SSD MobileNet V2 merupakan model pra terlatih yang telah diujikan pada COCO *dataset*, sehingga bobot yang ada pada model ini sudah ada berdasarkan pelatihan sebelumnya. Ketika akan menggunakan model ini dengan objektif yang berbeda, maka hanya perlu melakukan *fine tuning* pada pendeteksian objek yang akan dibuat. SSD MobileNet V2 memiliki lapisan sebanyak 267 lapisan, dengan total parameter sebanyak 15,291,106 (Francis, n.d.)

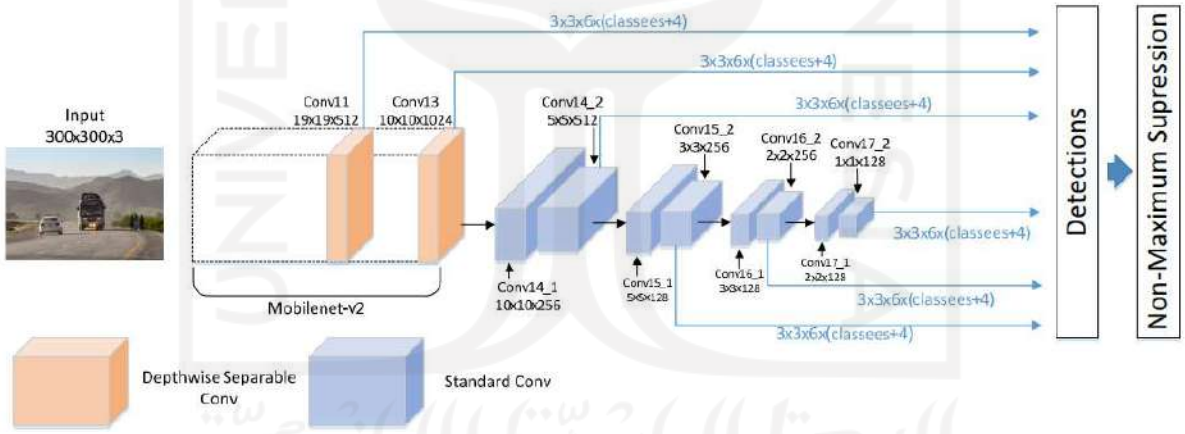
SSD atau *Single Shot Detector* merupakan salah satu metode dalam pendeteksian objek. Dari istilahnya *Single Shot* memiliki arti metode ini menyelesaikan *localization* dan *classification* dalam satu alur maju pada jaringan (Forson, 2017). SSD juga merupakan salah satu pendeteksi objek *one-stage* yang digunakan di beberapa aplikasi pendeteksian. Pada SSD, VGG16 adalah sebagai ekstraktor ciri default dan klasifikasi. MobileNet V2 sendiri merupakan pengembangan dari versi sebelumnya yang tadinya bernama VGG16-SSD. Dengan tujuan mengurangi kompleksitas komputasi, Google menggantikan VGG16 dengan MobileNet, hal ini juga meningkatkan kinerja *real-time* dari SSD lebih cepat dari pendeteksi yang pernah ada. (Chiu, Ruan, Shen, & Lie, 2020). SSD MobileNet V2 memiliki enam *feature map* yang diwariskan dari VGG16. Dibawah adalah perbandingan dari VGG16-SSD dengan MobileNet-SSD.

Detector Model	VGG16-SSD	Mobilenet-SSD
Layer 1	38x38x512	19x19x96
Layer 2	19x19x1024	10x10x1280
Layer 3	10x10x512	5x5x512
Layer 4	5x5x256	3x3x256
Layer 5	3x3x256	2x2x256
Layer 6	1x1x128	1x1x128

Gambar 2.4 Perbandingan dari VGG16-SSD dengan MobileNet-SSD

Sumber: Chiu, Ruan, Shen, & Lie (2020)

SSD MobileNet V2 ini pada arsitekturnya menggunakan *depth-wise separable convolution* untuk membangun *deep neural network* yang ringan. Proses konvolusi dari tiap *layer* yang ada membuat banyak parameter dari ke dalam *output* konvolusi, sehingga memaksa algoritma untuk menghafal daripada belajar yang artinya akan menyebabkan *overfitting* dan *depth-wise separable convolution* ini dapat menghindari itu terjadi. (Pandey, 2018).



Gambar 2.5 Arsitektur SSD MobileNet V2

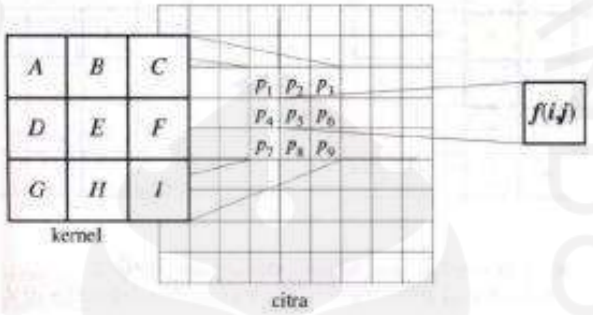
Sumber: Chiu, Ruan, Shen, & Lie (2020)

Pada MobileNet, lapisan konvolusional konvensional di modifikasi dengan *depth-wise convolution* dan *pointwise convolution. Depth-wise convolusion* melakukan operasi konvolusional dengan kernel 3 x 3 dan *pointwise convolution* adalah lapisan konvolusional umum yang menggunakan kernel 1 x 1. Kemudian dilanjutkan dengan *batch normalization* dan dan ReLU 6 pada tiap hasil konvolusional. Kombinasi *depth wise* dan *point wise* membuat

MobileNet mempercepat proses pembelajaran juga mengurangi waktu komputasi (Teng, et al., 2020).

2.1.7 Konvolusi

Konvolusi merupakan operator matemetika yang digunakan untuk melakukan kombinasi linear pada dua deret angka yang menghasil deret angka ketiga (Fahrani, 2020). Konvolusi juga diterapkan dalam pengolahan citra seperti untuk memperhalus atau memperjelas sebuah citra (Soeparno, Ghazali, & Ohliati, 2012).



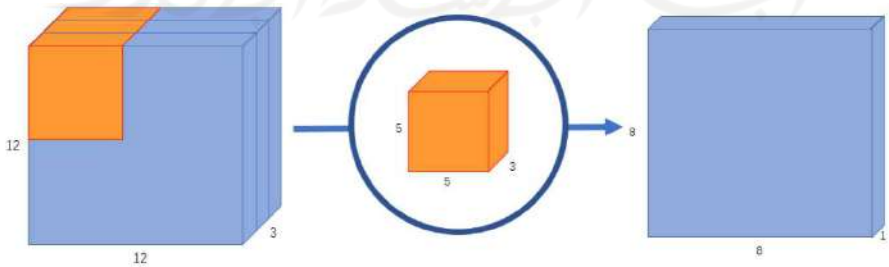
Gambar 2.6 Ilustrasi konvolusi

Sumber: Soeparno (2012)

2.1.8 Depth Wise Separable Convolution

Depth Wise Separable Convoltion merupakan jenis konvolusi yang bekerja bukan hanya pada dimensi spasial namun pada kedalaman juga (Wang, 2018), konvolusi ini memiliki gagasan bahwa kedalaman dan dimensi spasial dapat dipisahkan (Pandey, 2018).

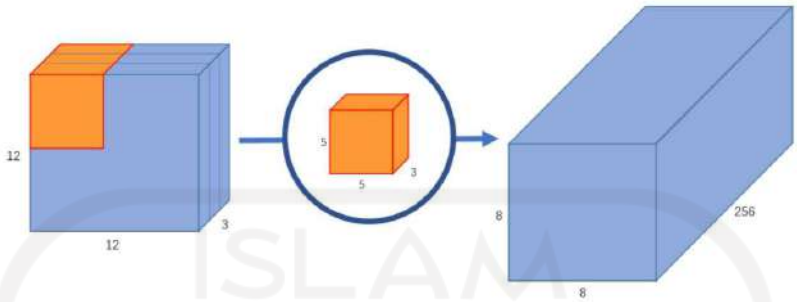
Pada konvolusi biasa, konvolusi dilakukan dengan mengalikan seluruh citra beserta *channel* nya dengan kernelnya. Misal dilakukan konvolusi pada citra 12x12 dengan *channel* 3 menggunakan kernel 5x5 dengan *channel* 3 juga. Konvolusi tersebut akan menghasilkan citra dengan ukuran 8x8x1.



Gambar 2.7 Konvolusi biasa dengan *output* 8x8x1

Sumber: Wang (2018)

Lalu bagaimana jika membutuhkan output dengan ukuran 8x8x256, maka perlu membuat kernelnya 256 lalu menumpuknya untuk mendapatkan ukuran 8x8x256.



Gambar 2.8 Konvolusi biasa dengan *output* 8x8x256

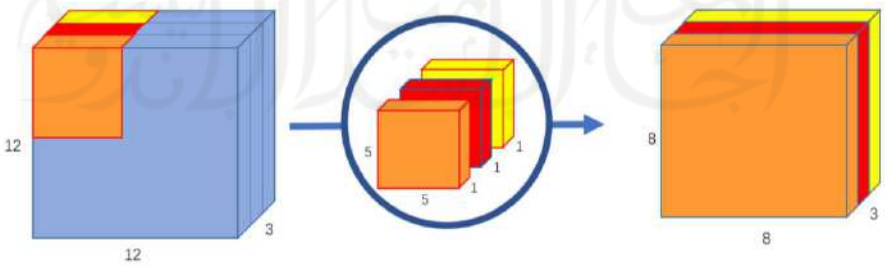
Sumber: Wang (2018)

Seperti itulah contoh konvolusi biasa, secara singkat bisa direpresentasikan dengan 12x12x3 → (5x5x3x256) →8x8x256 (5x5x3x256 merepresentasikan tinggi, lebar, *input channel*, dan jumlah *output* dari *channel* kernel).

Hal ini berbeda dengan *Depth Wise Separable Convolution* yang membagi konvolusi menjadi dua bagian untuk melakukan dua konvolusi yang disebut *Depth Wise Convolution* dan *Point Wise Convolution.*

2.1.8.1 Depth Wise Convolution

Depth Wise Convolution adalah konvolusi yang dilakukan pada tiap *channel* kedalaman pada citra. Masih dengan citra dengan ukuran 12x12x3 dilakukan konvolusi menggunakan kernel 5x5x3. Dengan *Depth Wise Convolution* konvolusi tersebut akan memiliki *output* citra dengan ukuran 8x8x3. Belum selesai pada tahap ini, karena pada hal ini dibutuhkan output dengan ukuran 8x8x256 seperti pada percobaan konvolusi biasa.

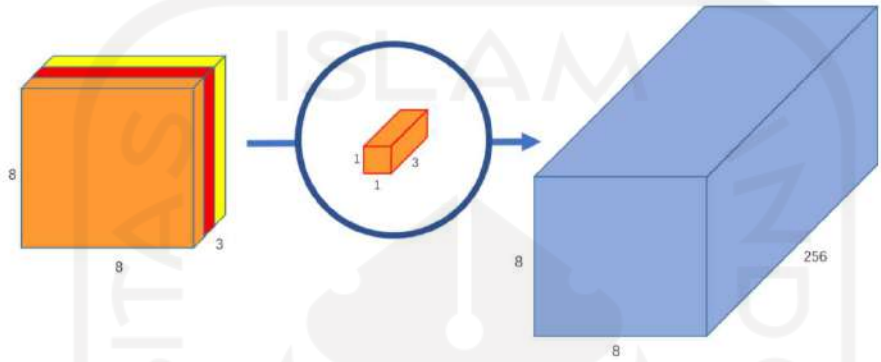


Gambar 2.9 *Depth Wise Convolution*, menggunakan 3 kernel mengubah citra 12x12x3 menjadi 8x8x3

Sumber: Wang (2018)

2.1.8.2 Point Wise Convolution

Dinamai *Point Wise Convolution* karena konvolusi ini menggunakan kernel ukuran 1x1 atau sebuah kernel yang mengiterasi setiap poin pada citra. Karena output sebelumnya berukuran 8x8x3 maka ukuran kernel pun disesuaikan yaitu menjadi 1x1x3. Melanjutkan pada tahap ini untuk mendapatkan *output* 8x8x256 maka dibutuhkan sebanyak 256 kernel 1x1x3.



Gambar 2.10 *Point Wise Convolution* dengan 256 kernel menghasilkan citra dengan 256 *channel*

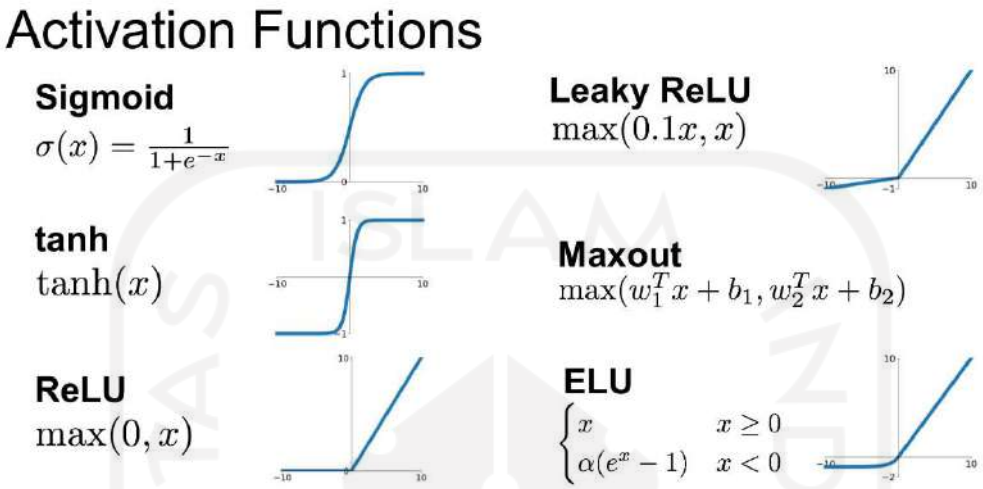
Sumber: Wang (2018)

Secara singkat konvolusi biasa direpresentasikan dengan 12x12x3 → (5x5x3x256) →8x8x256, lalu *Depth Wise Separable Convolution* direpresentasikan dengan 12x12x3 → (5x5x1x3) → (1x1x3x256) → 8x8x256. Dilihat dari komputasinya pada konvolusi biasa terdapat 256 kernel 5x5x3 yang bergeser 8x8 kali, jika dikalikan secara keseluruhan maka 256x3x5x5x8x8 = 1,228,800 perkalian. Pada *Depth Wise Convolution* terdapat 3 kernel 5x5x1 yang bergeser 8x8 kali yang keseluruhannya 3x5x5x1x8x8 = 4,800 perkalian. Lalu pada *Point Wise Convolution* terdapat 256 kernel 1x1x3 yang bergeser 8x8 kali, maka 256x1x1x3x8x8 = 49,152 perkalian. Kemudia ditotalkan antara *Depth Wise Convolution* dengan *Point Wise Convolution* yaitu 53,952 perkalian. Hasil ini lebih kecil dibandingkan konvolusi biasa yang berarti nilai komputasi *Depth Wise Separable Convolution* lebih ringan dibandingkan dengan konvolusi biasa. Tentunya hal ini juga memperkecil parameter yang ada sehingga model tidak terlalu kompleks (Wang, 2018).

2.1.9 Fungsi Aktivasi

Fungsi aktivasi merupakan fungsi yang digunakan dalam sebuah model untuk mendapatkan keluaran *node*. Keluaran dari fungsi aktivasi bermacam-macam seperti *yes* atau

no, nilai 0 sampai 1 ataupun nilai -1 sampai 1, semua itu bergantung pada jenis fungsi aktivasi yang digunakan (Fahrani, 2020). Diantaranya Sigmoid, ReLU dan Tangen Hiperbolik.

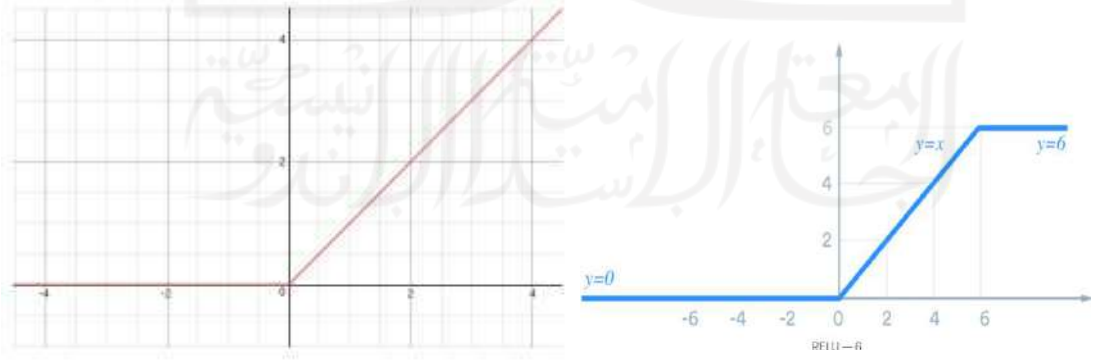


Gambar 2.11 Jenis fungsi aktivasi

Sumber: *kotakode.com*

2.1.10 ReLU 6

ReLU (Rectified Linear Units) adalah fungsi aktivasi yang digunakan secara konvensional untuk lapisan tersembunyi pada *deep neural network*. ReLU bekerja pada nilai ambang 0, dengan fungsi f(x) = max(0,x). ReLU akan menghasilkan nilai 0 ketika x < 0 dan menghasilkan nilai 1 ketika x > 0 (Agarap, 2019). Untuk ReLU 6 memiliki enam unit yang memiliki fungsi f(x) = min(max(0,x), 6) (Doshi, 2019).



(a) (b)

Gambar 2.12 (a) Fungsi Aktivasi ReLU (b) Fungsi Aktivasi ReLU 6

Sumber: Agarap (2019) dan Doshi (2019)

2.1.11 COCO (*Common Objects in Context*)

COCO adalah *dataset* untuk deteksi objek, segementasi dan *dataset* dengan keterangan yang berskala besar (COCO - Commont Object in Context, n.d.). COCO *dataset* berisi 91 kategori objek umum dengan 82 diantranya memiliki lebih dari 5000 contoh yang berlabel. Secara keseluruhan *dataset* ini memiliki 2.500.000 contoh berlabel pada 328.000 citra. Berbeda dengan ImageNet, COCO memiliki lebih sedikit kategori namun memiliki lebih banyak memiliki contoh untuk tiap kategorinya dan juga lebih besar dalam jumlah contoh per kategorinya dibandingkan PASCAL VOC dan SUN *dataset*. Dengan jumlah contoh berlabel per citra *dataset* ini dapat membantu untuk mempelajari informasi yang bersifat kontekstual (Lin, et al., 2015).

Gambar 2.13 Contoh citra berlabel pada COCO *dataset*

Sumber: (Lin, et al., 2015)

2.1.12 Hyperparameter

Hyperparamter adalah variabel yang menunjukan struktur jaringan dan variabel yang menentukan bagaimana jaringan dilatih (seperti: *Learning Rate*) (Radhakrishnan, 2017). Pengaturan *Hyperparamter* ini memiliki dampak pada akurasi dan kinerja model yang akan dibuat.

2.1.13 Learning Rate

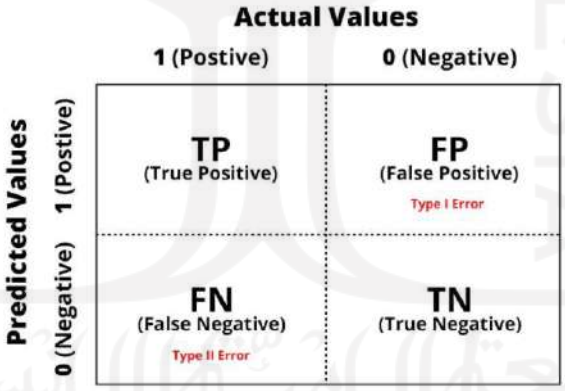
Learning Rate atau tingkat pembelajaran adalah *hyperparameter* yang mengontrol seberapa banyak kita mengatur bobot jaringan untuk penurunan gradien. Nilai *Learning Rate* perlu diperhatikan karena jika terlalu kecil penurunan gradien akan menjadi terlalu lambat yang artinya memakan waktu, tapi jika terlalu besar akan terjadi penurunan gradien secara berlebih yang berpeluang atau bahkan dapat menyebabkan tidak tercapainya *convergence* (Zulkifli, 2018).

2.1.14 Batch Size

Batch Size adalah jumlah atau banyaknya data yang digunakan dalam pelatihan model setiap terjadi perubahan parameter (Radhakrishnan, 2017). *Batch Size* memiliki dampak pada hasil yang akan dimunculkan sebagai *output. Batch Size* memiki andil dalam proses pelatihan, semakin besar *Batch Size* yang diatur semakin lama juga durasi pelatihan.

2.1.15 *Confusion Matrix*

Confusion Matrix adalah alat evaluasi visual yang digunakan untuk pengukuran kinerja dalam pembelajaran mesin (Xu, Zhang, & Miao, 2019) .



Gambar 2.14 *Confusion Matrix*

Sumber: *medium.com/@ksnugroho/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning*

Dalam *Confusion Matrix* terdapat empat nilai yang digunakan untuk penghitungan kinerja diantaranya yaitu *True Positive* (TP)*, False Positive* (FP)*, False Negative* (FN) dan *True Negative* (TN)*.* Lalu pada pengukuran kinerja sebuah model dapat dilihat dari nilai *Accuracy, Presicion, Recall* dan *F-1 Score* (Anggreany, n.d.). *Accuracy* adalah rasio prediksi benar (positif dan negatif) dengan keseluruhan data (Arthana, 2019).

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \quad (2.1)$$

Precision adalah rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positf. (Arthana, 2019).

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \quad (2.2)$$

Recall adalah rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif (Arthana, 2019).

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad (2.3)$$

F-1 Score adalah perbandingan rata-rata *precision* dan *recall* yang dibobotkan (Arthana, 2019).

$$F - 1\ Score = \frac{2\ x\ Recall\ x\ Precision}{Recall + \ Precision} \quad (2.4)$$

2.1.16 *Python*

Python adalah bahasa pemrograman interpretatif tingkat tinggi, berorientasi objek, dengan semantik dinamis. *Python* memiliki pembangunan struktur data yang tingkat tinggi, dikombinasikan dengan pengetikan dinamis (*dynamic typing*) dan pengikatan dinamis (*dynamic binding*), membuatnya sangat atraktif untuk Pengembangan Aplikasi Cepat (*Rapid Application Development*), serta untuk digunakan sebagai bahasa *scripting* atau *glue* untuk menghubungkan komponen yang ada secara bersamaan (What is Python? Executive Summary, n.d.).

2.1.17 *Tensorflow*

Tensorflow adalah platform sumber terbuka *end-to-end* untuk pembelajaran mesin. *Tensorflow* memiliki ekosistem alat, pustaka (*library*), dan sumber daya komunitas yang komprehensif dan fleksibel yang memungkinkan peneliti memutakhirkan, membangun dengan mudah dalam pengembangan yang aplikasi yang mendukung pembelajaran mesin

(TensorFlow, n.d.). *TensorFlow* awalnya dikembangkan oleh para peneliti dan insinyur yang bekerja di tim *Google Brain* dalam organisasi *Intelligent Research Google Machine* untuk melakukan *Machine Learning* dan *Deep Neural Network*. Sistem ini cukup umum untuk diterapkan di berbagai domain lain juga (tensorflow, n.d.).

2.1.18 *Google Colab*

Colaboratory, atau disingkat "*Colab*" adalah produk dari *Google Research*. Colab memungkinkan siapa saja untuk menulis dan mengeksekusi kode python arbitrer melalui browser, dan sangat cocok untuk pembelajaran mesin, analisis data, dan pendidikan. Secara lebih teknis, Colab adalah layanan *Jupyter notebook* yang dihosting yang tidak memerlukan persiapan untuk digunakan, sekaligus memberikan akses gratis ke sumber daya komputasi termasuk GPU (Colaboratory - Google, n.d.).

2.2 Tinjauan Pustaka

Tabel 2.2 Tabel penelitian-penelitian terdahulu tentang pendeteksian tajwid.

Judul	Metode	Akurasi	Penulis
Detection System Tajwid Al Quran on Image Using Bray Curtis Distance	Bray Curtis Distance	60% - 90%	Rizal, Fadlisyah, Muhathir, Al Muammar Akfal
Model Fuzzy K-Nearest Neighbor dengan Local Mean pada Pengenalan Pola Citra Tajwid.	Fuzzy K-Nearest Neighbor dengan Local Mean	96,43%	Hafizh Al Kautsar Aidilof
Automatic Tajweed Rules Recognition using k-Nearest Neighbour (k-NN)	k-Nearest Neighbour	84,44 %	Shafaf Ibrahim, Farah Afiqah Abdul Rahim, Zaaba Ahmad
Pendeteksian Tajwid Idgham Mutajanisain Pada Citra Al-Qur'an Menggunakan Fuzzy Associative Memory (FAM)	Fuzzy Associative Memory (FAM)	60% - 90%	Maryana, Fadlisyah, Sujacka Retno
Sistem Pendeteksian Pola Tajwid Al-Qur'an Hukum Ikhfa Syafawi Dan Idgham Mimi Pada Citra Menggunakan Metode Euclid Distance Dan Bray Curtis Distance	Euclid Distance Dan Bray Curtis Distance	70% - 90%	Defry Hamdhana, Fadlisyah, Safira Adani

Pada penelitian dengan judul "*Detection System Tajwid Al Quran on Image Using Bray Curtis Distance*" (Rizal, Fadliansyah, Muhathir, & Akfal, 2015) melakukan beberapa tahap pengolahan citra untuk memudahkan proses pendekteksian seperti *resizing, grayscale, convolution* sebelum menggunakan Bray Cusrtis Distance. Pada penelitian tersebut terdapat dua tajwid yang dideteksi diantaranya Tanwin Izhar, Qalqalah. Hasilnya menunjukan akurasi pendeteksian 60% - 90%. Hasil ini juga terjadi pada penelitian yang berjudul "Pendeteksian Tajwid Idgham Mutajanisain Pada Citra Al-Qur'an Menggunakan Fuzzy Associative Memory (FAM)" (Maryana, Fadlisyah, & Retno, 2017). Namun dengan metode yang berbeda. Jika dirata-ratakan kedua penelitian memiliki akurasi pendeteksian sebesar 75%. Selain itu proses *resizing, grayscale, convolution* sama-sama dilakukan kedua penelitian tersebut dan juga dilakukan oleh penelitian "Sistem Pendeteksian Pola Tajwid Al-Qur'an Hukum Ikhfa Syafawi Dan Idgham Mimi Pada Citra Menggunakan Metode Euclid Distance Dan Bray Curtis Distance" (Hamdhana, Fadliansyah, & Adani, 2018). Akan tetapi, penelitian ini memiliki hasil akurasi pendeteksian yang berbeda yaitu 70% - 90% atau jika dirata-ratakan 80%.

Pada penelitian yang dilakukan oleh Shafaf Ibrahim, dkk (2019) dengan judul "*Automatic Tajweed Rules Recognition using k-Nearest Neighbour (k-NN)*" melakukan tahap yang hampir mirip juga seperti tiga penelitian sebelumnya yang sudah disinggung diatas. Namun perbedaanya terdapat pada tahap-tahap tersebut dirangkum dalam sebuah tahap yang disebut *Pre-processing*. Pada penelitian ini, pada tahap *Pre-processing* memiliki perbedaan yaitu penelitian ini melakukan *thinning & flip* juga *word segmentation*. Tahap *thinning* dilakukan untuk memperkecil bentuk yang ada pada citra tanpa memecahnya dan *flip* dilakukan untuk membantu tahap segementation karena fungsinya hanya akan bekerja dengan baik untuk citra dengan urutan kiri ke kanan sedangkan tulisan arab diurutkan dari kanan ke kiri. Penelitian ini memiliki hasil akurasi sebesar 84,44%. Lalu terakhir dengan judul "Model Fuzzy K-Nearest Neighbor dengan Local Mean pada Pengenalan Pola Citra Tajwid" (Aidilof, 2018) memiliki akurasi paling tinggi diantara lainnya yaitu sebesar 96,43%. Dalam penelitiannya Aidilof, Hafizh Al Kautsar (2018) melakukan beberapa percobaan dengan mengubah nilai *k* yang terdapat dalam algortima K-Nearest Neighbor. Dengan nilai k=1 didapat akurasi 96,43%, dengan nilai k=2 didapat akurasi 96,30%. Nilai akurasi ini tidak mengalami perubahan ketika mengubah nilai k=3 sampai k=1 dengan akurasi 96,15%.

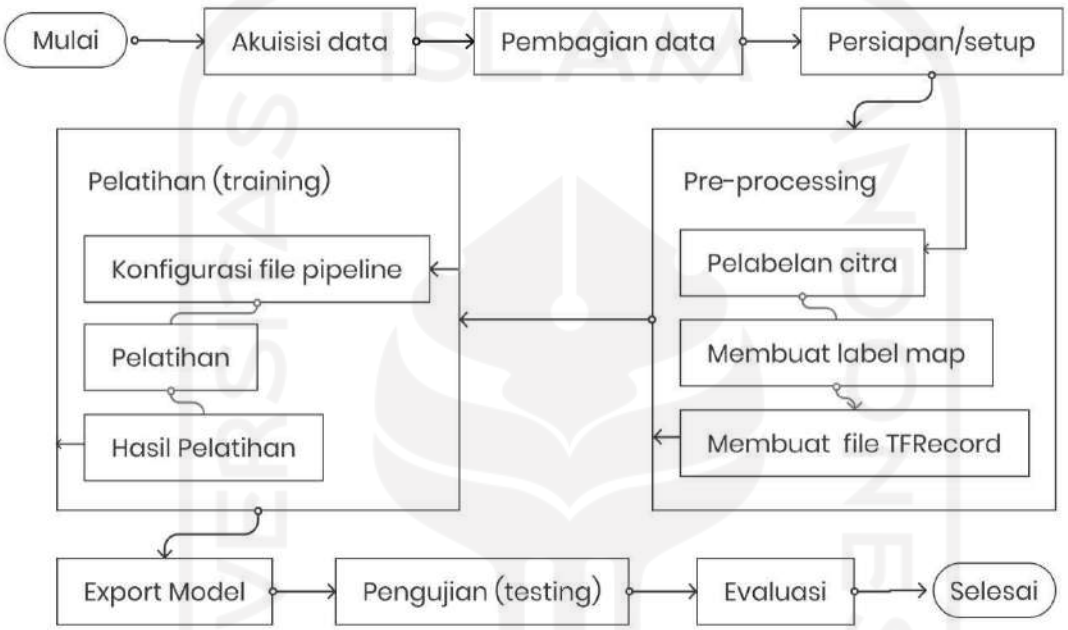
Pada penelitian-penelitian diatas diantaranya terdapat beberapa fenomena *gap* ataupun kekurangan yang menyebabkan hasil tidak maksimal. Seperti perbedaan font yang digunakan

sebagai data set yang menyebabkan hasil pendeteksian menjadi *false*. Lalu pada penelitian yang lainnya terdapat penyebaran data yang tidak merata karena salah satu tajwidnya memiliki jumlah huruf yang lebih banyak dari hukum tajwidnya. Pada penelitian ini mencoba menutupi *gap* tersebut dengan menggunakan font yang sejenis dan pemilihan tajwid yang tidak memiliki banyak huruf sehingga penyebaran data tidak merata dapat dihindari. Adapun penggunaan SSD MobileNet V2 pada penelitian ini juga untuk mendukung proses pendeteksian dengan arsitekturnya yang ringan karena menggunakan *Depth Wise Separable Convolution*. Berdasarkan TensorFlow 2 *Detection Model Zoo* SSD MobileNet V2 memiliki waktu pendeteksian (*Speed*) yang lebih cepat diantara yang lainnya. Selain itu, dengan harapan ke depannya dapat diterapkan untuk aplikasi telepon pintar model ini tepat digunakan untuk *mobile* (seluler).

BAB III
METODOLOGI PENELITIAN

3.1 Perancangan

Pada poin ini menggambarkan bagaimana alur percobaan atau eksperimen dalam penelitian ini yang divisualisasikan melalui *flowchart*.



Gambar 3.1 *Flowchart*/alur percobaan dalam penelitian

Pada penelitian ini akan dilakukan tiga percobaan dengan alur yang sama, untuk mengetahui perbandingan dari tiga percobaan tersebut.

3.2 Data

Data yang digunakan pada penelitian ini adalah data primer. Data pada penelitian ini berbentuk citra/gambar dua dimensi yang berisikan potongan bacaan Al-Quran yang mengandung tajwid, khususnya tajwid yang akan dideteksi pada penelitian ini yaitu Mad Layyin dan Mad Arid Lissukun.

3.2.1 Akuisisi Citra

Data yang digunakan merupakan gambar/citra yang berasal dari hasil tangkapan layar telepon pintar dari aplikasi Qur'an Kemenag yang secara resmi dirilis oleh Lajnah Pentahsinan Mushaf Al-Quran (Purnomo, n.d.).

Gambar 3.2 Tampilan Qur'an Kemenag

Sumber: (Purnomo, n.d.)

Cetakan font yang digunakan pada aplikasi ini sama dengan yang digunakan pada cetakan standar Al-Quran Indonesia yaitu menggunakan font "LPMQ Isep Misbah" (Intan, 2018), sehingga hal ini menjadi lingkup penelitian yang dikhususkan di Indonesia. Hasil tangkapan layar berupa citra halaman Al-Quran dengan resolusi 1080x2340 dengan ekstensi .jpg.

Gambar 3.3 Tangkapan layar dari halaman Qur'an Kemenag

Setelah gambar didapatkan, gambar dipindahkan ke perangkat komputer personal atau laptop kemudian dipotong atau *crop* untuk diambil bagian yang terdapat tajwid Mad Layyin dan Mad Arid Lissukun. Pemotongan dilakukan menggunakan aplikasi *dekstop* Figma, dengan ukuran 322x182 piksel. Kemudian disimpan dengan format .jpg berdasarkan kelas tajwid yang akan dideteksi yaitu dua kelas, Mad Layyin dan Mad Arid Lissukun. Secara keseluruhan didapat 520 citra.

3.2.2 Pembagian Data Citra

Pada penelitian yang dilakukan oleh Kurdthongmee (2020), pada penelitian tersebut membandingkan dua jenis *deep neural network* yaitu YOLO dan SSD MobileNet. Peneliti tersebut melakukan pembagian data yang berbeda pada masing-masing *deep neural network*. Pada YOLO membagi data dengan perbandingan data latih dan data uji sebesar 70:30 dan pada SSD MobileNet dengan perbandingan 80:20. Hasilnya SSD MobileNet memiliki *detection rate* dan *detection correctness* lebih tinggi dibandingan YOLO.

Dari hal tersebut, penulis akan mencoba melakukan hal yang sama dalam pembagian data yaitu sebesar 80% untuk data latih dan 20% untuk data uji. Namun penulis membutuhkan partisi data lagi untuk validasi, sehingga penulis akan membagi lagi persentasi data uji yang ada menjadi 10% untuk data uji dan 10% untuk data validasi. Secara keseluruhan pada penelitian ini perbandingan antara data latih, uji, dan validasi yaitu 80:10:10.

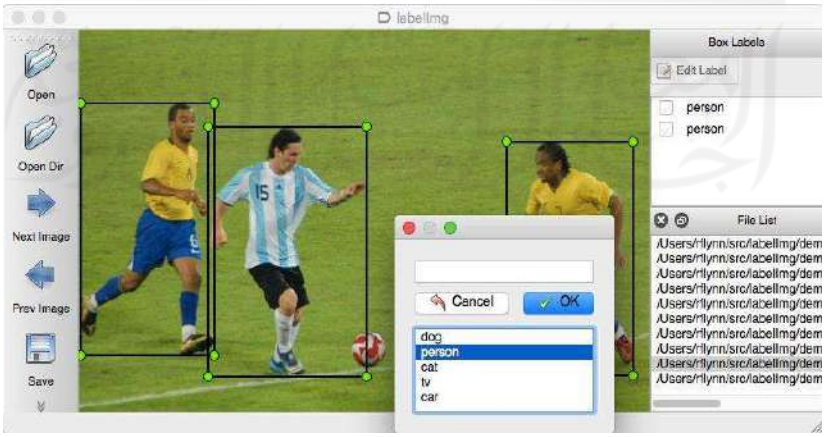
3.3 Persiapan (*Setup*)

Pada setiap percobaan diperlukan *setup* pada projek yang akan dibuat dengan menjalankan perintah (*command*) yang ada pada *colab notebook* secara sekuensial seperti instalasi *package* ataupun *API*, mengunduh model pra terlatih, ekstrak berkas, dan membuat struktur *folder* projek.

3.4 *Pre-processing*

3.4.1 Pelabelan Citra

Pada tahap ini dilakukan pelabelan pada citra untuk menandai area yang akan menjadi objek pendeteksian. Data citra yang diberi label merupakan dua kelas citra tajwid, Mad Layyin dan Mad Arid Lissukun. Pelabelan dilakukan menggunakan *tool* yang disebut *labelImg* (Training Custom Object Detector TensorFlow 2 Object Detection API tutorial, n.d.). *Tool* tersebut dipasang dan dijalankan menggunakan perintah Python. Dengan bantuan antarmukanya, proses pelabelan diatur agar proses pelabelan menjadi efisien. Seperti mengatur berkas *path directory* dari citra yang akan digunakan, *file path* untuk *save*, menggunakan *shortcut key* untuk membuat label, berpindah antar citra, dan pengaturan *auto save*. Hasil dari pelabelan atau *output* akan berupa berkas dengan ekstensi .XML.



Gambar 3.4 *Tool labelImg*

Sumber: github.com/tzutalin/labelImg

3.4.2 Membuat Label Map

Label Map dibuat untuk menjadi *reference* dalam algoritma untuk klasifikasi. Label Map berisi nama dari sebuah kelas beserta id nya. Berkas Label Map memiliki ekstensi .pbtxt.

3.4.3 Membuat Berkas TFRecord

Berkas TFRecord ini digunakan meyimpan urutan biner untuk berkas yang ada pada *folder* data uji dan data validasi untuk menjadi *input* dalam proses pelatihan yang hanya dapat dibaca oleh algoritma TensorFlow. Berkas ini dibentuk dengan menjalankan perintah yang men-*generate* berkas *generate_tfrecord.py* (berisi sekuen kode untuk membuat berkas TFRecord), *file path* data latih dan validasi, dan berkas Label Map. Dari perintah tersebutlah akan terbentuk berkas *train.record* dan *val.record* yang merupakan input untuk proses pelatihan.

3.5 Pelatihan (*Training*)

3.5.1 Konfigurasi Berkas Pipeline

Pada tahap ini dilakukan konfigurasi pada berkas yang disebut *pipeline.config* yang berisi konfigurasi pelatihan untuk mengatur *hyperparameter* dan beberapa *file path*. Pada penelitian ini ada beberapa hal yang diatur dalam berkas konfigurasinya diantaranya jumlah kelas yang akan dideteksi, *batch size, learning rate, file path* dari berkas *checkpoint* (berisi informasi seberapa jauh model dilatih, berkas yang menyimpan semua nilai parameter yang digunakan oleh model) yang sudah diunduh dari model pra terlatih, *file path* dari Label Map, dan *file path* dari *train.record* dan *val.record*.

3.5.2 Pelatihan Model

Pada model pra terlatih yang digunakan memiliki pembagian peran, MobileNet adalah arsitektur yang memiliki peran untuk ekstraksi ciri (Rahmaniar & Hermawan, 2021) dan SSD adalah arsitektur yang memiliki peran dalam mengklasifikasi dan lokalisasi objek deteksi pada citra (Forson, 2017). MobileNet V2 memiliki dua jenis blok, yaitu blok dengan *stride* 1 dan blok dengan *stride* 2. Pada tiap bloknya terdapat tiga lapisan. Pertama, lapisan *point wise convolution* ukuran 1x1 dengan *batch normalization* dan dan ReLU 6. Kedua, lapisan *depth wise convolution* ukuran 3x3 dengan *batch normalization* dan dan ReLU 6. Ketiga, lapisan konvolusi 1x1 dengan *batch normalization*. Pada SSD dilakukan konvolusi biasa dengan *stride* 1 dan 2 untuk mendapatkan objek yang dideteksi dengan *bounding box* dan nama kelasnya.

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	300 × 300 × 3
Conv dw / s1	3 × 3 × 32	150 × 150 × 32
Conv / s1	1 × 1 × 32 × 64	150 × 150 × 32
Conv dw / s2	3 × 3 × 64	150 × 150 × 64
Conv / s1	1 × 1 × 64 × 128	75 × 75 × 64
Conv dw / s1	3 × 3 × 128	75 × 75 × 128
Conv / s1	1 × 1 × 128 × 128	75 × 75 × 128
Conv dw / s2	3 × 3 × 128	75 × 75 × 128
Conv / s1	1 × 1 × 128 × 256	38 × 38 × 128
Conv dw / s1	3 × 3 × 256	38 × 38 × 256
Conv / s1	1 × 1 × 256 × 512	38 × 38 × 256
Conv dw / s1	3 × 3 × 512	38 × 38 × 512
Conv / s1	1 × 1 × 512 × 512	38 × 38 × 512
5× Conv dw / s1	3 × 3 × 512	38 × 38 × 512
5× Conv / s1	1 × 1 × 512 × 512	38 × 38 × 512
Conv / s2	3 × 3 × 512 × 1024	38 × 38 × 512
Conv / s1	1 × 1 × 1024 × 1024	19 × 19 × 1024
Conv / s1	1 × 1 × 1024 × 256	19 × 19 × 1024
Conv / s2	3 × 3 × 256 × 512	19 × 19 × 256
Conv / s1	1 × 1 × 512 × 128	10 × 10 × 512
Conv / s2	3 × 3 × 128 × 256	10 × 10 × 128
Conv / s1	1 × 1 × 256 × 128	5 × 5 × 256
Conv / s2	3 × 3 × 128 × 256	5 × 5 × 128
Conv / s1	1 × 1 × 256 × 128	3 × 3 × 256
Conv / s1	3 × 3 × 128 × 256	3 × 3 × 128
Conv / s1	1 × 1 × 256 × 128	1 × 1 × 256
Conv / s1	3 × 3 × 128 × 256	1 × 1 × 128

Gambar 3.5 Struktur layer pada model SSD MobileNet V2.

Sumber: (Arabi, Haghighat, & Sharma, 2019)

Pelatihan dapat dimulai dengan cara menjalankan perintah yang men-*generate* berkas *model_main_tf2.py* (berisi sekuen kode untuk membuat model, memulai pelatihan, dan menjalankan model) bersama dengan berkas *pipeline.config*.

3.5.3 Hasil Pelatihan

Setelah tahap pelatihan selesai maka akan terbentuk beberapa berkas yang diantaranya akan digunakan untuk evaluasi. Berkas yang terbentuk yaitu berkas *events* yang berisi informasi aktivitas pelatihan (seperti *loss* dan grafiknya) dan berkas *checkpoint* (hasil pelatihan tiap percobaan).

3.6 *Export* Model

Setelah hasil pelatihan didapatkan, berkas yang ada pada hasil pelatihan di-*export* untuk disimpan modelnya. Hasil *export* model tersebut akan digunakan untuk tahap pengujian. Tahap ini dilakukan dengan men-*generate* berkas *exporter_main_v2.py* (berisi sekuen kode untuk meng-*export* berkas hasil pelatihan), berkas *pipeline.config* yang digunakan sebelumnya, dan berkas yang ada pada *folder* hasil pelatihan.

3.7 Pengujian (*Testing*)

Tahap pengujian ini akan digunakan untuk menguji model (dalam mendeteksi) yang sudah dilatih menggunakan data uji yang telah disiapkan. Pengujian dilakukan dengan meng-*input* setiap data uji pada kode *Inference* dari TensorFlow. Pada kode *Inference* membutuhkan *file path* dari model yang telah di-*export* dan Label Map. Kemudian dibantu dengan *Confusion Matrix* untuk memetakan hasil pengujian deteksi yang hasilnya dapat digunakan untuk menghitung metrik kinerja pada pengujian.

3.8 Evaluasi

Evaluasi pada model dilakukan untuk mengukur kinerja model yang telah dilatih untuk melihat nilai yang menjadi tolak ukur kinerjanya, yaitu *Average Precision* dan *Average Recall*. Semakin besar nilai-nilai tersebut maka semakin baik pula sebuah model (Narkhede, 2018). Selain itu juga terdapat *Classification Loss* adalah *loss/*ketidakakuratan yang terjadi pada model dalam mengklasifikasikan yang diprediksi dengan yang sebenarnya. *Localization Loss* adalah *loss* pada *bounding box* yang diprediksi dengan yang sebenarnya. (Zanini, n.d.).

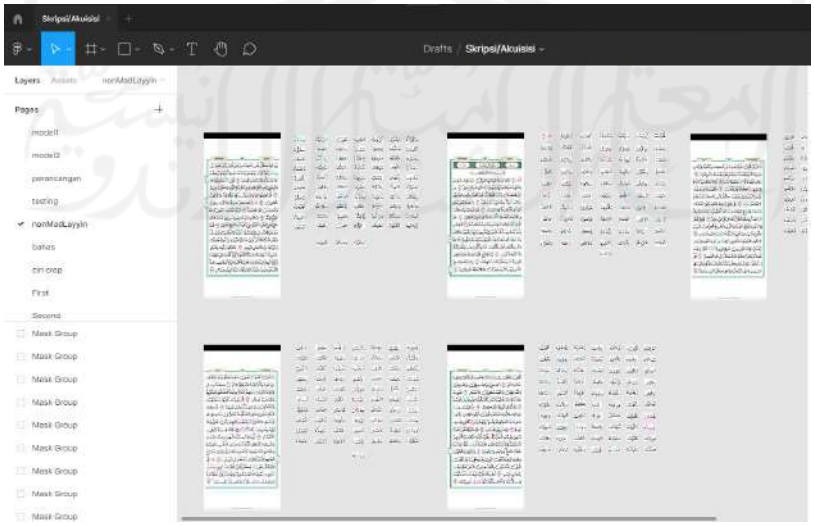
BAB IV
HASIL DAN PEMBAHASAN

4.1 Data

4.1.1 Akuisisi Citra

Pada akuisisi citra, pertama-tama dilakukan tangkapan layar dari aplikasi Qur'an Kemenag menggunakan telepon pintar yang kemudian dipindahkan ke komputer untuk dilakukan *crop*. Lalu dikelompokkan berdasarkan kelas tajwid sebelum akhirnya dilakukan pembagian untuk data latih, uji dan validasi.

Gambar 4.1 Hasil tangkapan layar pada aplikasi Qur'an Kemenag



Gambar 4.2 Proses *crop* citra

Gambar 4.3 Hasil *crop* citra dengan ukuran 322x182 piksel

Pada tiap hasil *crop* citra paling banyak terdapat dua kata yang salah satunya terdapat hukum tajwid yang akan dideteksi.

4.1.2 Pembagian Data Citra

Setelah di-*crop*, citra ditempatkan dalam *folder* berdasarkan kelasnya. Kemudian dibagi lagi untuk keperluan percobaan yang dibagi menjadi tiga yaitu data latih, uji, dan validasi dengan perbandingan 80:10:10. Dari pembagian data tersebut memiliki perbandingan Mad Layyin 50% dan Mad Arid Lissukun 50%.

4.2 *Setup*

Dalam tahap ini mencakup beberapa hal seperti instalasi TensorFlow, *Object Detection* API, mengunduh model pra terlatih SSD MobileNet V2 FPNLite 320x320 dari TensorFlow 2 *Detection Model Zoo*, dan membuat struktur folder.

```
!pip install tensorflow-gpu
```

Gambar 4.4 Perintah untuk instalasi tensorflow.

Pada Gambar 4.4 merupakan perintah untuk instalasi tensorflow, versi yang digunakan adalah TensorFlow 2.5.0. dan untuk versi Python nya adalah Python 3.7.10

```
1) !git clone https://github.com/tensorflow/models
2) cd models/research/
3) !protoc object_detection/protos/*.proto --python_out=.
4) !git clone https://github.com/cocodataset/cocoapi.git
5) cd cocoapi/PythonAPI
6) !make
7) cp -r pycocotools /content/models/research
8) cd /content/models/research
```

```
9) cp object_detection/packages/tf2/setup.py .
10) python -m pip install --use-feature=2020-resolver .
```

Gambar 4.5 Perintah untuk instalasi *Object Detection API*.

Pada Gambar 4.5 merupakan serangkaian perintah untuk instalasi *Object Detection API* yang berisi berkas-berkas seperti *model_main_tf2.py* dan *generate_tfrecord.py*. Baris pertama digunakan untuk mengunduh berkas *Object Detection* API, lalu perintah *cd* untuk membuka *path directory*. Lalu baris ketiga digunakan untuk meng-*compile* berkas-berkas protobuf dengan ekstensi .proto. Protobuf adalah *framework* untuk mengatur serialisasi model dan parameter pelatihan. Baris keempat digunakan untuk mengunduh *COCO API* dari github yang memuat *evaluation metrics* untuk evaluasi model, lalu perintah *cd* untuk membuka *path directory* dan perintah baris ke enam untuk meng-*compile* berkas di dalam *COCO API*. Baris berikutnya atau ketujuh berfungsi untuk menyalin berkas *pycocotools* ke *directory* yang telah ditentukan. Lalu menyalin *setup.py* pada baris kesembilam dan di baris terakhir di *run* menggunakan perintah untuk menginstal *Object Detection API* secara keseluruhan.

Setelah instalasi TensorFlow dan *Object Detection API* semuanya selesai, selanjutnya adalah membuat struktur folder. Struktur folder dibuat seperti yang ada pada tutorial resmi dari TensorFlow 2 *Object Detection API Tutorial.*

Gambar 4.6 Struktur folder projek

Berikutnya yang perlu dipersiapkan adalah mengunduh model pra terlatih dari TensorFlow 2 *Detection Model Zoo*. Pada penelitian ini menggunakan model SSD MobileNet V2 FPNLite 320x320 (models/tf2_detection_zoo.md at master tensorflow/models, n.d.).

```
cd /content/training_demo/pre-trained-models
!wget http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
!tar -xvf ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
```

Gambar 4.7 Perintah untuk mengunduh dan ekstrak model SSD MobileNet V2 FPNLite 320x320.

Pada Gambar 4.7 baris pertama digunakan untuk membuka *directory berkas* yang akan menjadi tempat unduhan model. Baris berikutnya dengan perintah !*wget* untuk mengunduh berkas model lalu perintah *!tar* untuk mengekstrak berkas model tersebut. Didalamnya terdapat berkas yang dibutuhkan untuk konfigurasi yaitu *pipeline.config.* Berkas tersebut disalin ke *directory /content/training_demo/models/my_ssd_mobnet.*

4.3 *Pre-processing*

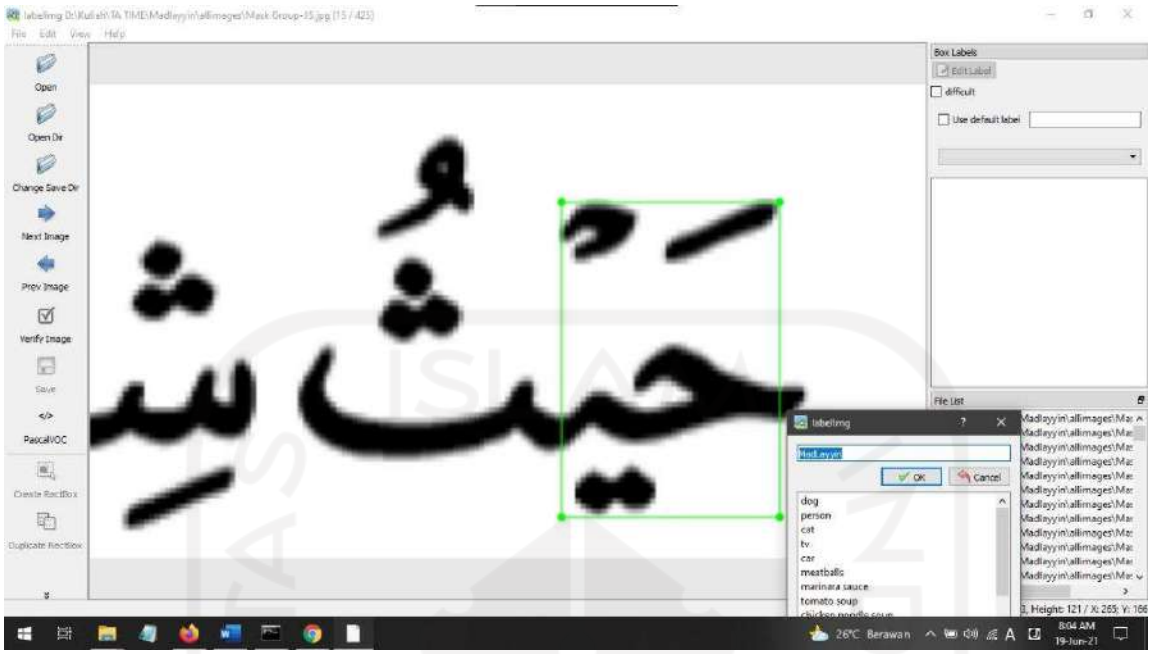
4.3.1 Pelabelan Citra

Pelabelan dilakukan untuk menandai daerah dari citra yang menjadi informasi untuk diolah dalam pelatihan model agar model dapat mengenali objek yang dideteksi. Berkas dari pelabelan citra ini berupa berkas dengan ekstensi .XML. Pelabelan menggunakan *tool* bernama *labelImg* dengan menjalankan perintah pada Gambar 4.8 maka akan terbuka tampilan dari *tool labelImg*. Pada penelitian ini *tool labelImg* dijalankan pada perangkat secara lokal terpisah dari sekuen yang lainnya yang ada pada *colab notebook*.

```
python labelImg.py
```

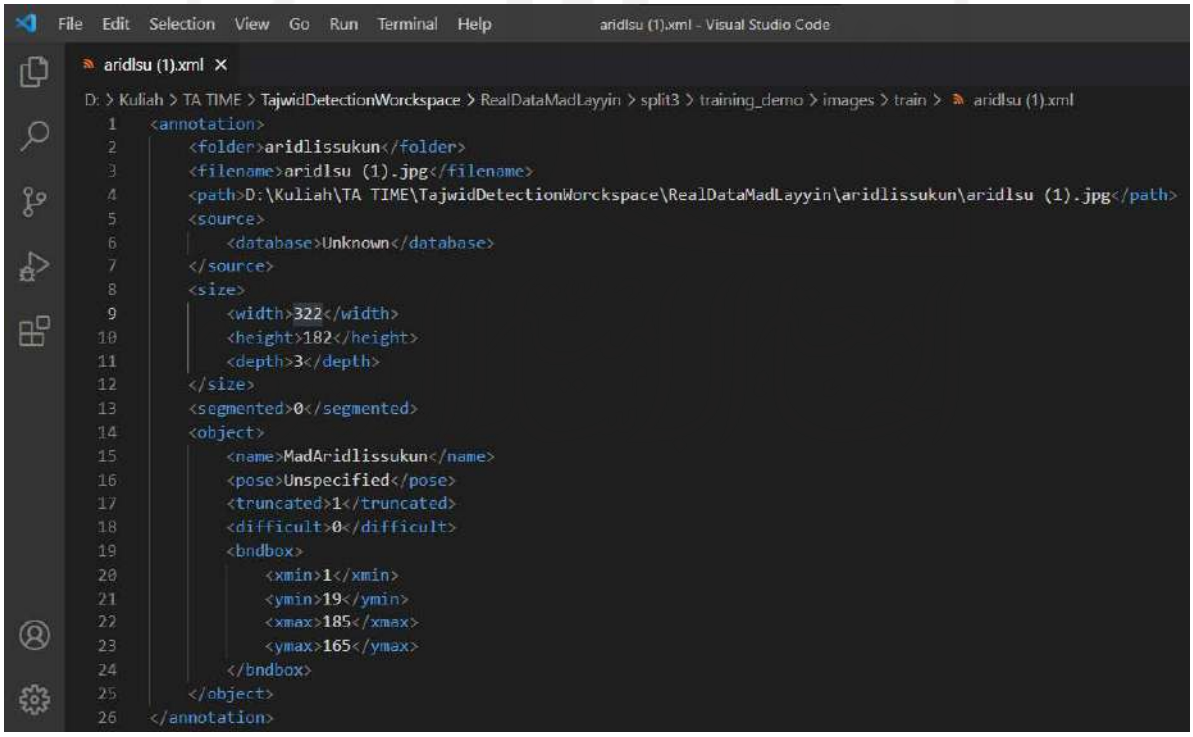
Gambar 4.8 Perintah menjalankan *tool labelImg*

Dengan menjalankan perintah pada Gambar 4.8 maka akan muncul tampilan dari *Tool labelImg*. Setelah *tool* ini muncul perlu diatur untuk *path directory* dari folder citra, tempat penyimpanan hasil pelabelan yang *path directory* nya sama dengan folder citra, dan mengatur untuk *auto save*.



Gambar 4.9 Proses pelabelan citra

Pelabelan diterapkan pada semua citra, untuk pelatihan, uji maupun validasi. Setelah pelabelan selesai berkas XML akan terbentuk dan tersimpan secara otomatis pada *path directory* yang telah diatur.



```xml
<annotation>
    <folder>aridlissukun</folder>
    <filename>aridlsu (1).jpg</filename>
    <path>D:\Kuliah\TA TIME\TajwidDetectionWorckspace\RealDataMadLayyin\aridlissukun\aridlsu (1).jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>322</width>
        <height>182</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>MadAridlissukun</name>
        <pose>Unspecified</pose>
        <truncated>1</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>1</xmin>
            <ymin>19</ymin>
            <xmax>185</xmax>
            <ymax>165</ymax>
        </bndbox>
    </object>
</annotation>
```

Gambar 4.10 Berkas XML hasil pelabelan citra

Karena pelabelan citra dilakukan secara lokal dan sekuen dari projek berada pada *colab notebook* secara *online*, maka setelah semua citra diberi label semuanya diunggah ke dalam *directory* citra yang ada pada projek *Google Colab*.

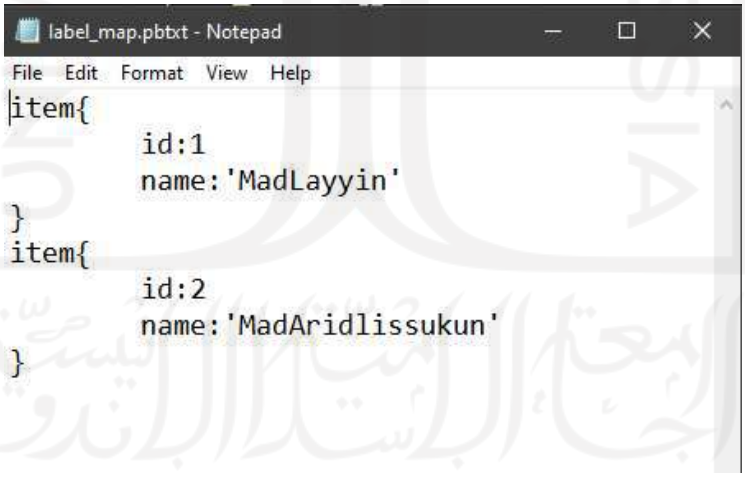
4.3.2 Membuat Label Map

Label map digunakan untuk penamaan kelas yang akan menjadi *reference* untuk model yang akan dilatih. Label Map berisi id dan nama kelas yang akan dideteksi. Pada penelitian ini terdapat dua kelas yaitu kelas Mad Layyin dengan id 1 dan Mad Arid Lissukun dengan id 2.

```
Labels = [{'name':'MadLayyin', 'id':1}, {'name':'MadArisLissukun',
'id':2}]

with open(ANNOTATION_PATH + '\label_map.pbtxt', 'w') as f:
    for label in labels:
        f.write('item { \n')
        f.write('\tid:{}\n'.format(label['id']))
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('}\n')
```

Gambar 4.11 Kode untuk membuat berkas label map



Gambar 4.12 Hasil/isi dari berkas Label Map

Berkas Label Map dibuat menggunakan kode yang ada pada Gambar 4.11 yang dilakukan secara lokal pada perangkat komputer dan hasilnya akan berupa berkas dengan ekstensi .pbtxt seperti yang ada pada Gambar 4.12.

4.3.3 Membuat Berkas TFRecord

Berkas TFRecord dibutuhkan dalam proses pelatihan untuk menjadi *input* pelatihan. Terdapat dua berkas yang akan dibentuk dalam pembuatan berkas TFRecord ini yaitu berkas *train.record* dan *val.record*. Untuk membentuk berkas ini, dibutuhkan berkas lainnya yaitu berkas *generate_tfrecord.py,* berkas Label Map dan citra pelatihan juga validasi.

```
!python /content/training_demo/generate_tfrecord.py -
x /content/training_demo/images/train -
l /content/training_demo/annotations/label_map.pbtxt -
o /content/training_demo/annotations/train.record

!python /content/training_demo/generate_tfrecord.py -
x /content/training_demo/images/validation -
l /content/training_demo/annotations/label_map.pbtxt -
o /content/training_demo/annotations/val.record
```

Gambar 4.13 Perintah untuk membentuk berkas TFRecord

4.4 Pelatihan

4.4.1 Konfigurasi Berkas Pipeline

Pada penelitian ini dilakukan tiga kali percobaan dengan konfigurasi yang berbeda juga yang diatur pada berkas konfigurasi masing-masing *pipeline.config.* Setiap percobaan konfigurasi pelatihan memiliki *num steps* sebanyak 10000 dengan *num class* (banyak kelas yang dapat dideteksi) sebanyak 2, perbedaannya terdapat pada *hyperparamter* lain pada berkas *pipeline.config*.

Tabel 4.1 Tabel perbandingan Konfigurasi 1, Konfigurasi 2, dan Konfigurasi 3

Hyperparameter	**Konfigurasi 1**	**Konfigurasi 2**	**Konfigurasi 3**
Learning Rate	0,001	0,01	0,1
Batch Size	2	4	8

Perbedaan konfigurasi ini dilakukan untuk mengetahui perbandingan performa model yang akan dihasilkan ketiganya dengan partisi data yang sama. Pada tahap ini selain mengatur berkas konfigurasi *pipeline.config*, perlu diatur juga *file path* dari Label Map, TFRecord (*train.record* dan *val.record*) dan berkas *checkpoint* dari model pra terlatih yang digunakan sebagai inisialisasi pelatihan model (*fine tuning*).

```
  fine_tune_checkpoint: "/content/training_demo/pre-trained-models/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0"
  num_steps: 10000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
  fine_tune_checkpoint_type: "detection"
  fine_tune_checkpoint_version: V2
}
train_input_reader {
  label_map_path: "/content/training_demo/annotations/label_map.pbtxt"
  tf_record_input_reader {
    input_path: "/content/training_demo/annotations/train.record"
  }
}
eval_config {
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
  label_map_path: "/content/training_demo/annotations/label_map.pbtxt"
  shuffle: false
  num_epochs: 1
  tf_record_input_reader {
    input_path: "/content/training_demo/annotations/val.record"
  }
}
```

Gambar 4.14 *File path* yang perlu diatur untuk pelatihan

4.4.2 Pelatihan Model

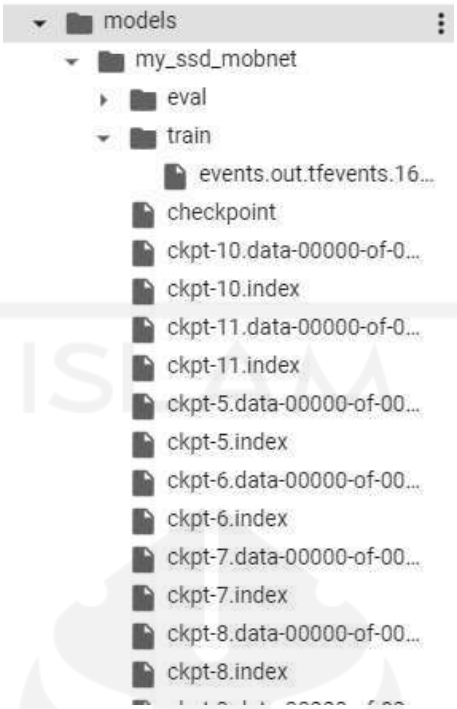
Selanjutnya untuk memulai pelatihan adalah dengan cara menggunakan sebuah perintah yang di dalamnya memanggil berkas *model_main_tf2.py, directory* tempat menyimpan model hasil pelatihan (untuk menyimpan hasil pleatihan), dan berkas *pipeline.config*.

```
!python /content/training_demo/model_main_tf2.py --
model_dir=/content/training_demo/models/my_ssd_mobnet --
pipeline_config_path=/content/training_demo/models/my_ssd_mobnet/pipe
line.config
```

Gambar 4.15 Perintah untuk memulai pelatihan

4.4.3 Hasil Pelatihan

Setelah pelatihan selesai akan terdapat beberapa berkas yang terbentuk diantaranya berkas *events* yang berisi informasi aktivitas pelatihan (seperti *loss* dan grafiknya) dan *checkpoint* yang berisi informasi seberapa jauh model dilatih.



Gambar 4.16 Struktur berkas dari model yang telah terbentuk dari hasil pelatihan

4.5 *Export* Model

Setelah hasil pelatihan didapatkan maka selanjutnya adalah meng-*export* hasil pelatihan. Dilakukan dengan cara menjalankan perintah yang men-*generate* berkas *exporter_main_v2.py*, berkas *pipeline.config* yang digunakan sebelumnya, dan berkas yang ada pada *folder* hasil pelatihan.

```
!python /content/training_demo/exporter_main_v2.py --
input_type image_tensor --
pipeline_config_path /content/training_demo/models/my_ssd_mobnet/pipe
line.config --
trained_checkpoint_dir /content/training_demo/models/my_ssd_mobnet --
output_directory /content/training_demo/exported_models/my_model
```

Gambar 4.17 Perintah untuk mendapatkan *export* model

4.6 Pengujian

Pengujian dilakukan menggunakan kode *Inference* TensorFlow dengan cara memuat model yang telah di-*export* sebelumnya, Label Map, dan data uji.

```
# PROVIDE PATH TO IMAGE DIRECTORY
IMAGE_PATHS = '/content/training_demo/images/test/aridlsu (61).jpg'


# PROVIDE PATH TO MODEL DIRECTORY
PATH_TO_MODEL_DIR = '/content/training_demo/exported_models/my_model'

# PROVIDE PATH TO LABEL MAP
PATH_TO_LABELS = '/content/training_demo/annotations/label_map.pbtxt'
```

Gambar 4.18 *File path* untuk model yang telah di-*export,* Label Map, dan data uji.

Lalu dilakukan *input* untuk setiap data uji pada kode *Inference* yang kemudian hasilnya dipetakan menggunakan *Confusion Matrix. Confusion Matrix* digunakan untuk membantu mendapatkan nilai TP, FP, FN, dan TN yang dari keempat nilai tersebut dapat dihitung *Accuracy, Precision, Recall,* dan *F-1 Score*.

Tabel 4.2 *Confusion Matrix* untuk model Konfigurasi 1

Aktual/Prediksi		Aktual		
		Mad Layyin	Mad Arid Lissukun	Tidak Terdeteksi
Prediksi	Mad Layyin	22	2	2
	Mad Arid Lissukun	0	25	1
	Tidak Terdeteksi	0	0	0

Dari table 4.2 didapatkan nilai:

TP = 22 FN = 0

FP = 2 + 2 = 4 TN = 25 + 1 = 26

Tabel 4.3 *Confusion Matrix* untuk model Konfigurasi 2

Aktual/Prediksi		Aktual	
		Mad Layyin	Mad Arid Lissukun
Prediksi	Mad Layyin	24	2
	Mad Arid Lissukun	0	26

Dari table 4.3 didapatkan nilai:

TP = 24 FN = 0 FP = 2 TN = 26

Tabel 4.4 *Confusion Matrix* untuk model Konfigurasi 3

Aktual/Prediksi		Aktual		
		Mad Layyin	Mad Arid Lissukun	Tidak Terdeteksi
Prediksi	Mad Layyin	21	4	1
	Mad Arid Lissukun	0	26	0
	Tidak Terdeteksi	0	0	0

Dari table 4.4 didapatkan nilai:

TP = 21 FN = 0

FP = 4 + 1 = 6 TN = 26

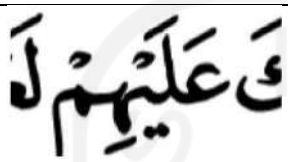
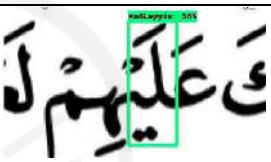
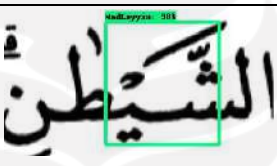
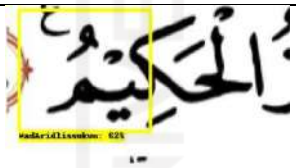
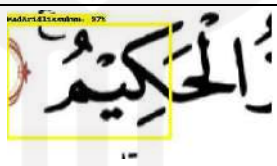
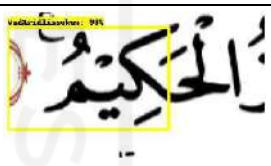
Setelah didapat nilai *Confusion Matrix* dari masing-masing model konfigurasi, maka dapat dihitung nilai dari *Accuracy, Precision, Recall,* dan *F-1 Score.*

Tabel 4.5 Hasil pengujian pendeteksian dari model Konfigurasi 1, Konfigurasi 2, dan Konfigurasi 3

Metrik	**Konfigurasi 1**	**Konfigurasi 2**	**Konfigurasi 3**
Accuracy	0.923076923	0.961538462	0.903846154
Precision	0.846153846	0.923076923	0.807692308
Recall	1	1	1
F-1 Score	0.916666667	0.96	0.893617021

Dapat dilihat dari Tabel 4.5 ketiga konfigurasi memiliki akurasi di atas 80%, jika melihat penelitian sebelumnya pada Tabel 2.1 ketiga hasil akurasi ini setidaknya berada di urutan kedua dari penelitian yang akurasinya paling tinggi yang akurasinya mencapai 96,43%. Meskipun ketiga model dari konfigurasi masing-masing berhasil menyentuh angka di atas 90% dengan nilai *Recall* yang sama pula, masih terdapat beberapa nilai kinerja lainnya yaitu *Precision* dan *F-1 Score*. Model dari Konfigurasi 2 memiliki nilai *Precision* dan *F-1 Score* lebih tinggi dibandingkan dengan Konfigurasi 1, namun nilai *Precision* dan *F-1 Score* Konfigurasi 1 masih lebih tinggi dibandingkan Konfigurasi 3.

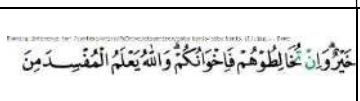
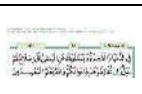
Tabel 4.6 Sampel hasil pengujian pendeteksian dari model Konfigurasi 1, Konfigurasi 2, dan Konfigurasi 3

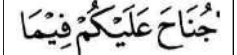
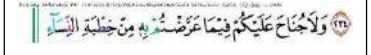
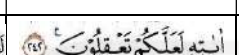
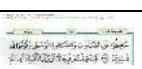
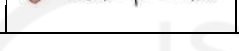
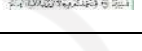
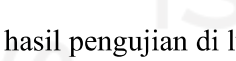
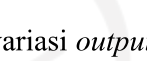
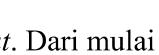
No	Konfigurasi 1	Konfigurasi 2	Konfigurasi 3
1			
2			
3			
4			
5			

Tabel 4.6 merupakan sampel hasil pendeteksian antara ketiga konfigurasi. Dapat dilihat dari kelima sampel dari masing-masing hasil pendeteksian tiga konfigurasi memiliki perbedaan-perbedaan, seperti *detection score* yang ditampilkan dan ukuran kotak yang mendeteksi tajwidnya.

Selain pengujian diatas, dilakukan juga pengujian pada data di luar data *set* yang ada dengan beberapa variasi seperti dua kata, tiga kata, satu baris, satu halaman, dan tanpa tajwid.

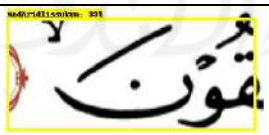
Tabel 4.7 Pengujian di luar data uji

No	Dua Kata	Tiga Kata	Satu Baris	Satu Halaman	Tanpa Tajwid
1					

2					
3					
4					
5					

Dari Tabel 4.7 hasil pengujian di luar data uji didapat beberapa variasi *output*. Dari mulai *output* yang *true, false* sampai yang seharusnya tidak terdeteksi tajwid tetapi terdeteksi. Pada percobaan menggunakan dua kata tiga diantranya terdeteksi tajwid dua bernilai *true* dan satu *false*. Sedangkan pada percobaan tiga kata, satu baris, dan satu halaman tidak ada yang terdeteksi sama sekali. Pada percobaan tanpa tajwid dari lima terdapat tiga yang terdeteksi tajwid, idealnya pada bagian percobaan ini tidak ada yang terdeteksi. Hal ini mungkin terjadi karena beberapa faktor seperti model yang tidak cukup dalam melakukan pelatihan sehingga tidak cukup handal untuk mendeteksi tajwid yang mirip, mengingat juga model ini awalnya dilatih pada COCO data *set* yang sangat berbeda dengan data tajwid sehingga pelatihan perlu dilakukan pelatihan lebih mendalam. Selain itu, jumlah data yang masih sedikit juga dapat menjadi penyebabnya yang menyebabkan model tidak terlalu banyak belajar, juga dari resolusi data citra yang digunakan ataupun pola-pola tajwid itu sendiri.

Tabel 4.8 Perbandingan pola tajwid yang terdeteksi

No	Tajwid yang dideteksi	Bukan tajwid yang dideteksi
1		
2		

Ciri yang dipelajari oleh model dalam mengenali pola tajwid bisa saja belum cukup baik untuk membedakan pola sejenis, sehingga dalam mengenali pola-pola yang mirip model belum bisa atau belum cukup baik membedakannya. Seperti pada Tabel 4.8 secara kasat mata diantara sampel tajwid yang dideteksi dan yang bukan memiliki kemiripan pola, pada nomor 1 mirip

dalam pola lam dan wau sedangkan pada nomor 2 memiliki kemiripan pada pola nun dan kaf yang memanjang.

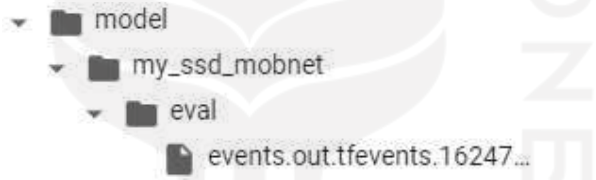
4.7 Evaluasi

Tahap ini menjadi tahap untuk mengevaluasi kinerja dari ketiga model. Evaluasi dilakukan menggunakan data validasi.

```
!python /content/training_demo/model_main_tf2.py --
model_dir=/content/training_demo/models/my_ssd_mobnet --
pipeline_config_path=/content/training_demo/models/my_ssd_mobnet/pipe
line.config --
checkpoint_dir=/content/training_demo/models/my_ssd_mobnet
```

Gambar 4.19 Perintah untuk menampilkan nilai evaluasi

Dengan menggunakan perintah yang ada pada Gambar 4.18 TensorFlow akan menampilkan hasil evaluasi pada *log* aktivitas dan membentuk berkas *events* yang berada pada *folder* eval.



Gambar 4.20 Perintah untuk menampilkan nilai evaluasi

Pada Gambar 4.18 memuat berkas *events* yang ada pada *folder* eval berisi informasi nilai evaluasi *Average Precision*, *Average Recall*, *Classification Loss,* dan *Localization Loss*.

Tabel 4.9 Tabel kinerja model Konfigurasi 1 dan Konfigurasi 2

Konfigurasi	***Average Precision***	***Average Recall***
Konfigurasi 1	0.683325	0.742308
Konfigurasi 2	0.808254	0.830769
Konfigurasi 3	0.803135	0.828846

Dari Tabel 4.7 dapat dilihat hasil model dari Konfigurasi 2 memiliki nilai *Average Precision* dan *Average Recall* yang lebih tinggi dibandingkan model dari Konfigurasi 1. *Average Recall* pada Konfigurasi 1 tampak sekali perbedaannya dibandingkan dengan nilai *Average Precision* nya. Konfigurasi 3 memiliki nilai nilai *Average Precision* dan *Average*

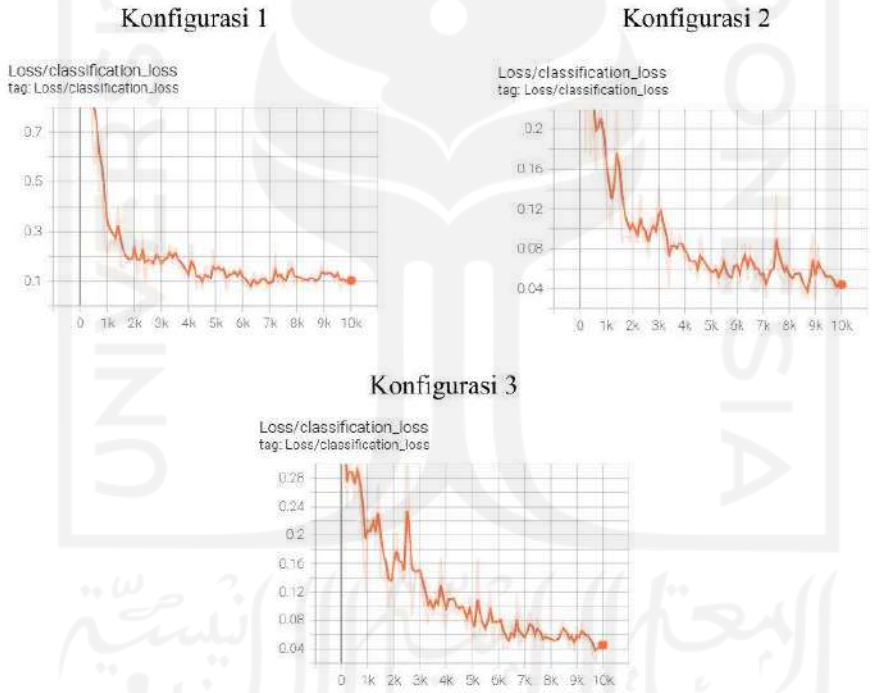
Recall yang sangat berdekatan dengan Konfigurasi 2, namun Konfigurasi 2 tetap memiliki nilai *Average Precision* dan *Average Recall* yang paling tinggi diantara dua konfiguasi lainnya.

Lalu ada *Classification Loss* dan *Localization Loss*. Dua nilai *loss* tersebut dapat divisualisasikan menggunakan grafik yang bisa didapatkan dengan menjalankan perintah untuk menampilkan grafik pada *Tensorboard.*

```
%load_ext tensorboard
tensorboard --logdir=.
```

Gambar 4.21 Perintah untuk membuka *Tensorboard*

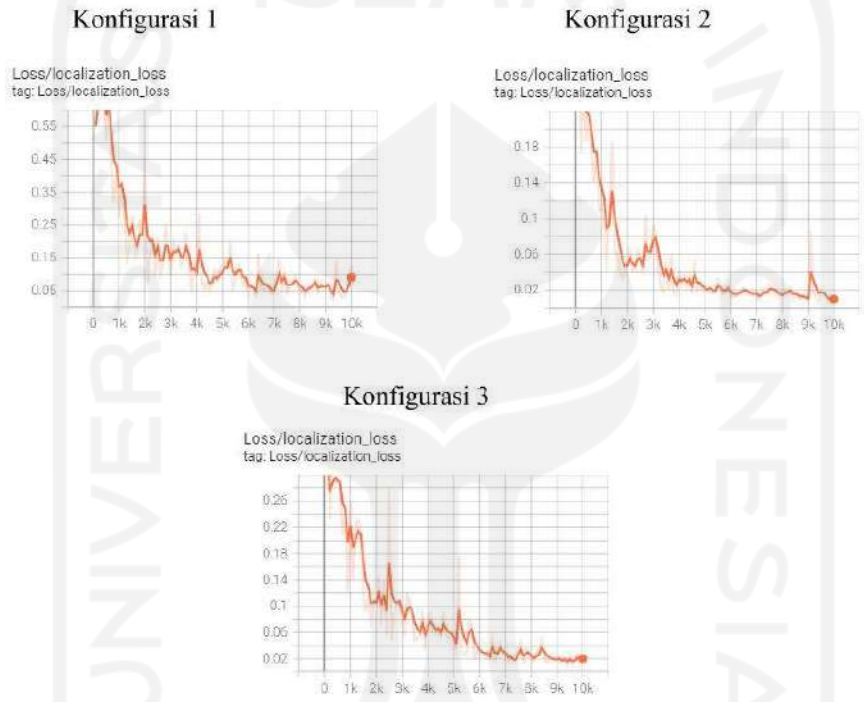
Pada Gambar 4.21 baris digunakan untuk memanggil *Tensorboard* dan baris kedua digunakan untuk menjalankan *Tensorboard* yang memuat grafik.



Gambar 4.22 Perbandingan *Classification Loss* Konfigurasi 1, Konfigurasi 2, dan Konfigurasi 3

Pada Gambar 4.22 merupakan grafik *Classification Loss* dari tiap konfigurasi. Pada Konfigurasi 1 tampak mengalami penurunan *loss* yang cukup ekstrem di langkah 1000 pertama. Penurunan *loss* tetap berlangsung sampai ke langkah terakhir meskipun sempat terjadi beberapa kenaikan. Jika dilihat dengan mata telanjang, *loss* paling kecil berhasil diraih sekitar

pada langkah 6000 sampai 7000. Lalu pada Konfigurasi 2 terjadi penurunan *loss* di awal namun tidak seekstrem pada Konfigurasi 1 jika dilihat secara grafik. Meskipun demikian, grafik Konfigurasi 2 berada pada rentang *loss* yang berbeda sehingga penurunan pada grafik Konfigurasi 1 tidak sebanding. *Loss* paling kecil pada Konfigurasi 2 berada pada langkah antara 8000 sampai 9000. Kemudian pada Konfigurasi 3 mengalami penurunan dan kenaikan *loss* pada sekitar langkah 1000 sampai 2000. Namun terjadi lagi kenaikan *loss* yang cukup signifikan pada langkah antara 2000 sampai 3000.



Gambar 4.23 Perbandingan *Localization Loss* Konfigurasi 1, Konfigurasi 2, dan Konfigurasi 3

Pada Gambar 4.23 merupakan grafik *Localization Loss* dari tiap konfigurasi. Dari ketiga grafik, ketiganya sama-sama memiliki *loss* yang tinggi diawal yang kemudian sempat terjadi beberapa kali kenaikan namun tidak sebesar di awal langkah seribuan. Pada Konfigurasi 1 tampak naik ketika sampai ke langkah terakhir berbeda dengan Konfigurasi 2 yang tampak turun ketika sampai ke langkah terakhir, berbeda juga dengan Konfigurasi 3 yang cenderung datar ketika menuju langkah terakhir.

Tabel 4.10 Perbandingan nilai *loss* Konfigurasi 1, Konfigurasi 2, dan Konfigurasi 3

Konfigurasi	***Classification Loss***	***Localization Loss***
Konfigurasi 1	0.088737	0.131142
Konfigurasi 2	0.044297	0.097607
Konfigurasi 3	0.044414	0.090325

Nilai-nilai *loss* pada Tabel 4.8 Merupakan nilai *loss* yang didapatkan pada langkah terakhir atau ke 10000. Nilai *loss* memiliki pengaruh pada model dalam mengklasifikasikan dan menemukan lokasi objek yang dideteksi. Dari keseluruhan nilai *loss*, Konfigurasi 2 memiliki *localization loss* paling kecil dibanding Konfigurasi 1 dan Konfigurasi 3. Dari hal tersebut bisa dikatakan bahwa model dari Konfigurasi 2 lebih baik dalam hal klasifikasi, akan tetapi model dari Konfigurasi 3 lebih baik dalam lokalisasi.

Perbedaan-perbedaan yang dihasilkan merupakan efek dari proses pelatihan dengan mengatur atau mengkonfigurasi *hyperparameter* pelatihan. Baik atau tidaknya model yang dihasilkan juga dari sudah pas atau belumnya pengaturan pada *hyperparameter*. Selain *learning rate* dan *batch size, num steps* juga dapat menjadi faktornya karena bisa jadi baik atau tidaknya model yang dihasilkan bukan hanya dari kombinasi pengaturan *learning rate* dan *batch size* saja melainkan seberapa banyak atau lama model dilatih yang hal ini diatur oleh *num steps* tersebut. Bisa saja dari hasil-hasil konfigurasi diatas bisa lebih baik jika model dilatih lebih jauh atau lama lagi.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari serangkaian percobaan dan analisis yang telah dilakukan pada penelitian ini, didapat kesimpulan sebagai berikut:

1. Percobaan pendeteksi tajwid menggunakan model pra terlatih SSD MobileNet V2 dilakukan dengan tiga skema yang berbeda. Perbedaan tersebut terdapat pada pengaturan *hyperparameter* yang diatur. *Hyperparameter* yang diatur pada tiap percobaan diantaranya yaitu *learning rate* dan *batch size.* Dari pengaturan *hyperparameter* tersebutlah dihasilkan model dengan kemampuan deteksi dan kinerja yang berbeda untuk dilihat perbandingan dari hasil percobaan pendeteksian tersebut.
2. Akurasi dari tiga percobaan dengan konfigurasi yang berbeda melalui pengujian menggunakan data uji didapat hasil, Konfigurasi 2 memiliki akurasi yang paling tinggi dengan nilai 96,153% lalu Konfigurasi 1 dengan nilai akurasi 92,307% disusul Konfigurasi 3 dengan nilai akurasi 90,384%.
3. Dilihat dari kinerjanya, model Konfigurasi 2 memiliki kinerja lebih baik dari model konfigurasi lainnya, hal ini dapat dilihat dari hasil pada tahap pengujian maupun nilai pada evaluasi pada model Konfigurasi 2 yang lebih tinggi, serta *loss* yang lebih kecil. Pada tahap pengujian Kofigurasi 1 memiliki hasil yang lebih baik dari Konfigurasi 3, namun Konfigurasi 3 memiliki hasil yang lebih baik dari Konfigurasi 1 pada tahap evaluasi.
4. Pada pengujian diluar data uji yang ada dan hasilnya, menunjukan bahwa pendeteksian yang dilakukan dengan model yang telah dilatih masih memiliki kekurangan-kekurangan (belum bisa mendeteksi data diluat data uji dengan baik) yang menjadi batasan. Sehingga selain nilai-nilai pada pengujian dan evaluasi, hal ini juga perlu menjadi pertimbangan.

5.2 Saran

Penelitian ini tidaklah sempurna, masih perlu improvisasi dari berbagai aspek. Penulis disini memiliki beberapa saran diantaranya sebagai berikut:

1. Dari segi data, jumlah data dapat ditambah untuk meningkatkan kemampuan model untuk lebih banyak belajar, dapat dicoba untuk menerapkannya pada data citra Al-Quran yang tidak hanya satu kata tetapi satu baris atau lebih bahkan satu halaman Al-Quran dengan variasi tajwid yang lebih banyak lagi.
2. SSD MobileNet V2 termasuk algortima yang bagus untuk pendeteksian. Patut dicoba untuk dilakukan beberapa percobaan yang lebih variatif dari segi pembagian data, konfigurasi *Learning Rate, Batch Size,* dan *Num Steps*.
3. Akan lebih bagus lagi jika dikembangkan lagi menjadi pendeteksian secara *real-time*. Sebelumnya, penulis sempat mencoba untuk melakukan hal tersebut namun terhalang *error*, spesifikasi laptop dan waktu.
4. Sangat direkomendasikan menggunakan spesifikasi PC atau laptop *gaming* yang tinggi, untuk mendukung proses pelatihan yang memakan waktu yang sangat lama.
5. Dengan hasil yang ada menggunakan model SSD MobileNet V2 ini tidak menjadi pembatas untuk mencoba dengan model yang lainnya yang mungkin akan lebih cocok sehingga dapat menghasilkan kinerja dan akurasi terbaik.

DAFTAR PUSTAKA

Agarap , A. F. (2019). Deep Learning using Rectified Linear Units (ReLU).

Anggreany, M. S. (t.thn.). *Confusion Matrix*. Diambil kembali dari Binus University: https://socs.binus.ac.id/2020/11/01/confusion-matrix/

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

Arthana, R. (2019, April 5). *Mengenal Accuracy, Precision, Recall dan Specificity serta yang diprioritaskan dalam Machine Learning*. Diambil kembali dari Medium: https://rey1024.medium.com/mengenal-accuracy-precission-recall-dan-specificity-serta-yang-diprioritaskan-b79ff4d77de8

Chiu, Y. C., Ruan, M. D., Shen, G. Y., & Lie, T. T. (2020). Mobilenet-SSDv2: An Improved Object Detection. *IEEE*.

COCO - Commont Object in Context. (t.thn.). Diambil kembali dari COCO - Commont Object in Context: https://cocodataset.org/

Colaboratory - Google. (t.thn.). Diambil kembali dari Google Colab: https://research.google.com/colaboratory/faq.html

Deng, L., & Yu, D. (2014). Deep Learning: Methods and Applications. *now*.

Dewi, S. R. (2018). *Deep Learning Object Detection Pada Video Menggunakan Tensorflow dan Convolutional Neural Network.* Yogyakarta.

Doshi, C. (2019, June 29). *Why Relu? Tips for using Relu. Comparison between Relu, Leaky Relu, and Relu-6.* Diambil kembali dari towards data science: https://medium.com/@chinesh4/why-relu-tips-for-using-relu-comparison-between-relu-leaky-relu-and-relu-6-969359e48310

Fahrani, F. S. (2020). *Algoritma Pendeteksi Dan Pengenalan Objek Pada Media Edukasi LKS Secara Otomatis.* Yogyakarta.

Forson, E. (2017, November 18). *Understanding SSD MultiBox — Real-Time Object Detection In Deep Learning*. Diambil kembali dari towardsdatasciences: https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

Francis, J. W. (t.thn.). *SSD-MobileNet V2 - Wolfram Neural Net Repository*. Diambil kembali dari Wolfram:

https://resources.wolframcloud.com/NeuralNetRepository/resources/SSD-MobileNet-V2-Trained-on-MS-COCO-Data/

Gupta, D. (2017, June 1). *Transfer Learning | Pretrained Models in Deep Learning*. Diambil kembali dari Analytics Vidhya: https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/

Intan, N. (2018, April 13). *LPMQ Luncurkan Font Isep Misbah untuk Penulisan Alquran*. Diambil kembali dari Republika: https://www.republika.co.id/berita/p73ucz396/lpmq-luncurkan-font-isep-misbah-untuk-penulisan-alquran

Jalled, F., & Voronkov, I. (2016). Object Detection Using Image Processing. *arXiv*.

Kim, K. G. (2016). *Deep Learning.* Cambridge: The MIT Press.

Kurdthongmee, W. (2020). A comparative study of the effectiveness of using popular DNN object. *Cell*.

Kusumanto, R., & Tompunu, A. N. (2011). Pengolahan Citra Digital Untuk Mendeteksi Obyek.

Lin, Y. T., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., . . . Dollar, P. (2015). Microsoft COCO: Common Objects in Context. *arXiv*.

Mathway | About Us. (t.thn.). Diambil kembali dari Mathway: https://www.mathway.com/about

Mistari, Asmara, R., & Hakkun, R. Y. (2012). Aplikasi Belajar Membaca Al-Quran dan Mengucapkan Huruf Hijaiyah Dengan Tajiwd Berbasis Android. *PENS*, 2.

models/tf2_detection_zoo.md at master tensorflow/models. (t.thn.). Diambil kembali dari github: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

Narkhede, S. (2018, May 9). *Understanding Confusion Matrix | By Sarang Narkhede | Towards Data Science*. Diambil kembali dari towards data science: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Pandey, A. (2018, September 9). *Depth-wise Convolution and Depth-wise Separable Convolution by Atul Pandey*. Diambil kembali dari Medium: https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Purnomo, B. (t.thn.). *Aplikasi Qur'an Kemenag Semakin Mengasyikan*. Diambil kembali dari Lajnah Pentahsin Mushaf Al-Quran: https://lajnah.kemenag.go.id/berita/604-aplikasi-qur-an-kemenag-semakin-mengasyikkan

Radhakrishnan, P. (2017, August 10). *What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network?* Diambil kembali dari towards data science: https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a

Rahmaniar, W., & Hermawan, A. (2021). Real-Time Human Detection Using Deep Learning on Embedded Platforms: A Review. *Journal of Robotics and Control (JRC)*.

Soeparno, H., Ghazali, W., & Ohliati, J. (2012). Penerapan Metode Konvolusi Dalam Pengolahan Citra Digital. *Jurnal Mat Sat*.

Teng, T. W., Veerajagadheswar, P., Ramalingam, B., Yin, J., Mohan, R. E., & Gómez, B. F. (2020). Vision Based Wall Following Framework: A Case Study With HSR Robot for Cleaning Application. *sensors*.

tensorflow. (t.thn.). Diambil kembali dari Github: https://github.com/tensorflow/

TensorFlow. (t.thn.). Diambil kembali dari Tensorflow: https://www.tensorflow.org/

Training Custom Object Detector TensorFlow 2 Object Detection API tutorial. (t.thn.). Diambil kembali dari TensorFlow 2 Object Detection API tutorial: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html#preparing-the-dataset

Triyasyid, N. (2015). *Pedoman Ilmu Tajwid: Mudah dan Aplikatif.* Sukoharjo: Penerbit Taujih.

Wang, F. C. (2018, August 14). *A Basic Introduction to Separable Convolutions*. Diambil kembali dari towards data science : https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

What is Python? Executive Summary. (t.thn.). Diambil kembali dari https://www.python.org/doc/essays/blurb/

Xu, J., Zhang, Y., & Miao, D. (2019). Three-way confusion matrix for classification: A measure driven view. *Elsevier*.

Yu, F. (2016, October 3). *A Comprehensive guide to Fine-tuning Deep Learning Models in Keras (Part I)*. Diambil kembali dari Felix Yu: https://flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html

Yusuf, A. (2021, Februari 26). *65 Persen Muslim RI Buta Alquran, Syafruddin Prihatin | Republika*. Diambil kembali dari Republika:

https://www.republika.co.id/berita/qp4v1p483/65-persen-muslim-ri-buta-alquran-syafruddin-prihatin

Zanini, H. (t.thn.). *Custom Real-Time Object Detection in the Browser Using TensorFlow.js*. Diambil kembali dari towards data science: https://towardsdatascience.com/custom-real-time-object-detection-in-the-browser-using-tensorflow-js-5ca90538eace

Zulkifli, H. (2018, January 22). *Understanding Learning Rates and How It Improves Performance in Deep Learning*. Diambil kembali dari towards data science: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10

LAMPIRAN

LAMPIRAN Berkas *generate_tfrecord.py*

```
import os
import glob
import pandas as pd
import io
import xml.etree.ElementTree as ET
import argparse

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow
logging (1)
import tensorflow.compat.v1 as tf
from PIL import Image
from object_detection.utils import dataset_util, label_map_util
from collections import namedtuple

# Initiate argument parser
parser = argparse.ArgumentParser(
    description="Sample TensorFlow XML-to-TFRecord converter")
parser.add_argument("-x",
                    "--xml_dir",
                    help="Path to the folder where the input .xml
berkass are stored.",
                    type=str)
parser.add_argument("-l",
                    "--labels_path",
                    help="Path to the labels (.pbtxt) berkas.",
type=str)
parser.add_argument("-o",
                    "--output_path",
                    help="Path of output TFRecord (.record) berkas.",
type=str)
parser.add_argument("-i",
                    "--image_dir",
                    help="Path to the folder where the input image
berkass are stored. "
                         "Defaults to the same directory as
XML_DIR.",
                    type=str, default=None)
parser.add_argument("-c",
                    "--csv_path",
                    help="Path of output .csv berkas. If none
provided, then no berkas will be "
                         "written.",
                    type=str, default=None)

args = parser.parse_args()
```

```
if args.image_dir is None:
    args.image_dir = args.xml_dir

label_map = label_map_util.load_labelmap(args.labels_path)
label_map_dict = label_map_util.get_label_map_dict(label_map)


def xml_to_csv(path):
    """Iterates through all .xml berkass (generated by labelImg) in a
given directory and combines
    them in a single Pandas dataframe.

    Parameters:
    ----------
    path : str
        The path containing the .xml berkass
    Returns
    -------
    Pandas DataFrame
        The produced dataframe
    """

    xml_list = []
    for xml_berkas in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_berkas)
        root = tree.getroot()
        berkasname = root.find('berkasname').text
        width = int(root.find('size').find('width').text)
        height = int(root.find('size').find('height').text)
        for member in root.findall('object'):
            bndbox = member.find('bndbox')
            value = (berkasname,
                     width,
                     height,
                     member.find('name').text,
                     int(bndbox.find('xmin').text),
                     int(bndbox.find('ymin').text),
                     int(bndbox.find('xmax').text),
                     int(bndbox.find('ymax').text),
                     )
            xml_list.append(value)
    column_name = ['berkasname', 'width', 'height',
                   'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df
```

```
def class_text_to_int(row_label):
    return label_map_dict[row_label]


def split(df, group):
    data = namedtuple('data', ['berkasname', 'object'])
    gb = df.groupby(group)
    return [data(berkasname, gb.get_group(x)) for berkasname, x in
zip(gb.groups.keys(), gb.groups)]


def create_tf_example(group, path):
    with tf.gberkas.GBerkas(os.path.join(path,
'{}'.format(group.berkasname)), 'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size

    berkasname = group.berkasname.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
    classes = []

    for index, row in group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)
        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))

    tf_example =
tf.train.Example(features=tf.train.Features(feature={
        'image/height': dataset_util.int64_feature(height),
        'image/width': dataset_util.int64_feature(width),
        'image/berkasname': dataset_util.bytes_feature(berkasname),
        'image/source_id': dataset_util.bytes_feature(berkasname),
        'image/encoded': dataset_util.bytes_feature(encoded_jpg),
        'image/format': dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin':
dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax':
dataset_util.float_list_feature(xmaxs),
```

```
        'image/object/bbox/ymin':
dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax':
dataset_util.float_list_feature(ymaxs),
        'image/object/class/text':
dataset_util.bytes_list_feature(classes_text),
        'image/object/class/label':
dataset_util.int64_list_feature(classes),
    }))
    return tf_example


def main(_):

    writer = tf.python_io.TFRecordWriter(args.output_path)
    path = os.path.join(args.image_dir)
    examples = xml_to_csv(args.xml_dir)
    grouped = split(examples, 'berkasname')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())
    writer.close()
    print('Successfully created the TFRecord berkas:
{}'.format(args.output_path))
    if args.csv_path is not None:
        examples.to_csv(args.csv_path, index=None)
        print('Successfully created the CSV berkas:
{}'.format(args.csv_path))


if __name__ == '__main__':
    tf.app.run()
```

LAMPIRAN Berkas *pipeline.config* (Konfigurasi 1)

```
model {
  ssd {
    num_classes: 2
    image_resizer {
      fixed_shape_resizer {
        height: 320
        width: 320
      }
    }
    feature_extractor {
      type: "ssd_mobilenet_v2_fpn_keras"
      depth_multiplier: 1.0
      min_depth: 16
```

```
    conv_hyperparams {
      regularizer {
        l2_regularizer {
          weight: 3.9999998989515007e-05
        }
      }
      initializer {
        random_normal_initializer {
          mean: 0.0
          stddev: 0.009999999776482582
        }
      }
      activation: RELU_6
      batch_norm {
        decay: 0.996999979019165
        scale: true
        epsilon: 0.0010000000474974513
      }
    }
    use_depthwise: true
    override_base_feature_extractor_hyperparams: true
    fpn {
      min_level: 3
      max_level: 7
      additional_layer_depth: 128
    }
  }
  box_coder {
    faster_rcnn_box_coder {
      y_scale: 10.0
      x_scale: 10.0
      height_scale: 5.0
      width_scale: 5.0
    }
  }
  matcher {
    argmax_matcher {
      matched_threshold: 0.5
      unmatched_threshold: 0.5
      ignore_thresholds: false
      negatives_lower_than_unmatched: true
      force_match_for_each_row: true
      use_matmul_gather: true
    }
  }
  similarity_calculator {
    iou_similarity {
    }
```

```
  }
  box_predictor {
    weight_shared_convolutional_box_predictor {
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 3.9999998989515007e-05
          }
        }
        initializer {
          random_normal_initializer {
            mean: 0.0
            stddev: 0.009999999776482582
          }
        }
        activation: RELU_6
        batch_norm {
          decay: 0.996999979019165
          scale: true
          epsilon: 0.0010000000474974513
        }
      }
      depth: 128
      num_layers_before_predictor: 4
      kernel_size: 3
      class_prediction_bias_init: -4.599999904632568
      share_prediction_tower: true
      use_depthwise: true
    }
  }
  anchor_generator {
    multiscale_anchor_generator {
      min_level: 3
      max_level: 7
      anchor_scale: 4.0
      aspect_ratios: 1.0
      aspect_ratios: 2.0
      aspect_ratios: 0.5
      scales_per_octave: 2
    }
  }
  post_processing {
    batch_non_max_suppression {
      score_threshold: 9.99999993922529e-09
      iou_threshold: 0.6000000238418579
      max_detections_per_class: 100
      max_total_detections: 100
      use_static_shapes: false
```

```
      }
      score_converter: SIGMOID
    }
    normalize_loss_by_num_matches: true
    loss {
      localization_loss {
        weighted_smooth_l1 {
        }
      }
      classification_loss {
        weighted_sigmoid_focal {
          gamma: 2.0
          alpha: 0.25
        }
      }
      classification_weight: 1.0
      localization_weight: 1.0
    }
    encode_background_as_zeros: true
    normalize_loc_loss_by_codesize: true
    inplace_batchnorm_update: true
    freeze_batchnorm: false
  }
}
train_config {
  batch_size: 2
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_crop_image {
      min_object_covered: 0.0
      min_aspect_ratio: 0.75
      max_aspect_ratio: 3.0
      min_area: 0.75
      max_area: 1.0
      overlap_thresh: 0.0
    }
  }
  sync_replicas: true
  optimizer {
    momentum_optimizer {
      learning_rate {
        cosine_decay_learning_rate {
          learning_rate_base: 0.001
          total_steps: 10000
          warmup_learning_rate: 0.001
```

```
          warmup_steps: 1000
        }
      }
      momentum_optimizer_value: 0.8999999761581421
    }
    use_moving_average: false
  }
  fine_tune_checkpoint: "/content/training_demo/pre-trained-
models/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-
0"
  num_steps: 10000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
  fine_tune_checkpoint_type: "detection"
  fine_tune_checkpoint_version: V2
}
train_input_reader {
  label_map_path:
"/content/training_demo/annotations/label_map.pbtxt"
  tf_record_input_reader {
    input_path: "/content/training_demo/annotations/train.record"
  }
}
eval_config {
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
  label_map_path:
"/content/training_demo/annotations/label_map.pbtxt"
  shuffle: false
  num_epochs: 1
  tf_record_input_reader {
    input_path: "/content/training_demo/annotations/val.record"
  }
}
```

LAMPIRAN Berkas *pipeline.config* (Konfigurasi 2)

```
model {
  ssd {
    num_classes: 2
    image_resizer {
      fixed_shape_resizer {
        height: 320
        width: 320
```

```
    }
  }
  feature_extractor {
    type: "ssd_mobilenet_v2_fpn_keras"
    depth_multiplier: 1.0
    min_depth: 16
    conv_hyperparams {
      regularizer {
        l2_regularizer {
          weight: 3.9999998989515007e-05
        }
      }
      initializer {
        random_normal_initializer {
          mean: 0.0
          stddev: 0.009999999776482582
        }
      }
      activation: RELU_6
      batch_norm {
        decay: 0.996999979019165
        scale: true
        epsilon: 0.0010000000474974513
      }
    }
    use_depthwise: true
    override_base_feature_extractor_hyperparams: true
    fpn {
      min_level: 3
      max_level: 7
      additional_layer_depth: 128
    }
  }
  box_coder {
    faster_rcnn_box_coder {
      y_scale: 10.0
      x_scale: 10.0
      height_scale: 5.0
      width_scale: 5.0
    }
  }
  matcher {
    argmax_matcher {
      matched_threshold: 0.5
      unmatched_threshold: 0.5
      ignore_thresholds: false
      negatives_lower_than_unmatched: true
      force_match_for_each_row: true
```

```
      use_matmul_gather: true
    }
  }
  similarity_calculator {
    iou_similarity {
    }
  }
  box_predictor {
    weight_shared_convolutional_box_predictor {
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 3.9999998989515007e-05
          }
        }
        initializer {
          random_normal_initializer {
            mean: 0.0
            stddev: 0.009999999776482582
          }
        }
        activation: RELU_6
        batch_norm {
          decay: 0.996999979019165
          scale: true
          epsilon: 0.0010000000474974513
        }
      }
      depth: 128
      num_layers_before_predictor: 4
      kernel_size: 3
      class_prediction_bias_init: -4.599999904632568
      share_prediction_tower: true
      use_depthwise: true
    }
  }
  anchor_generator {
    multiscale_anchor_generator {
      min_level: 3
      max_level: 7
      anchor_scale: 4.0
      aspect_ratios: 1.0
      aspect_ratios: 2.0
      aspect_ratios: 0.5
      scales_per_octave: 2
    }
  }
  post_processing {
```

```
      batch_non_max_suppression {
        score_threshold: 9.99999993922529e-09
        iou_threshold: 0.6000000238418579
        max_detections_per_class: 100
        max_total_detections: 100
        use_static_shapes: false
      }
      score_converter: SIGMOID
    }
    normalize_loss_by_num_matches: true
    loss {
      localization_loss {
        weighted_smooth_l1 {
        }
      }
      classification_loss {
        weighted_sigmoid_focal {
          gamma: 2.0
          alpha: 0.25
        }
      }
      classification_weight: 1.0
      localization_weight: 1.0
    }
    encode_background_as_zeros: true
    normalize_loc_loss_by_codesize: true
    inplace_batchnorm_update: true
    freeze_batchnorm: false
  }
}
train_config {
  batch_size: 4
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_crop_image {
      min_object_covered: 0.0
      min_aspect_ratio: 0.75
      max_aspect_ratio: 3.0
      min_area: 0.75
      max_area: 1.0
      overlap_thresh: 0.0
    }
  }
  sync_replicas: true
  optimizer {
```

```
    momentum_optimizer {
      learning_rate {
        cosine_decay_learning_rate {
          learning_rate_base: 0.01
          total_steps: 10000
          warmup_learning_rate: 0.01
          warmup_steps: 1000
        }
      }
      momentum_optimizer_value: 0.8999999761581421
    }
    use_moving_average: false
  }
  fine_tune_checkpoint: "/content/training_demo/pre-trained-
models/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-
0"
  num_steps: 10000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
  fine_tune_checkpoint_type: "detection"
  fine_tune_checkpoint_version: V2
}
train_input_reader {
  label_map_path:
"/content/training_demo/annotations/label_map.pbtxt"
  tf_record_input_reader {
    input_path: "/content/training_demo/annotations/train.record"
  }
}
eval_config {
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
  label_map_path:
"/content/training_demo/annotations/label_map.pbtxt"
  shuffle: false
  num_epochs: 1
  tf_record_input_reader {
    input_path: "/content/training_demo/annotations/val.record"
  }
}
```

LAMPIRAN Berkas *pipeline.config* (Konfigurasi 3)

```
model {
  ssd {
    num_classes: 2
    image_resizer {
      fixed_shape_resizer {
        height: 320
        width: 320
      }
    }
    feature_extractor {
      type: "ssd_mobilenet_v2_fpn_keras"
      depth_multiplier: 1.0
      min_depth: 16
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 3.9999998989515007e-05
          }
        }
        initializer {
          random_normal_initializer {
            mean: 0.0
            stddev: 0.009999999776482582
          }
        }
        activation: RELU_6
        batch_norm {
          decay: 0.996999979019165
          scale: true
          epsilon: 0.0010000000474974513
        }
      }
      use_depthwise: true
      override_base_feature_extractor_hyperparams: true
      fpn {
        min_level: 3
        max_level: 7
        additional_layer_depth: 128
      }
    }
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
        use_matmul_gather: true
      }
    }
```

```
    similarity_calculator {
      iou_similarity {
      }
    }
    box_predictor {
      weight_shared_convolutional_box_predictor {
        conv_hyperparams {
          regularizer {
            l2_regularizer {
              weight: 3.9999998989515007e-05
            }
          }
          initializer {
            random_normal_initializer {
              mean: 0.0
              stddev: 0.009999999776482582
            }
          }
          activation: RELU_6
          batch_norm {
            decay: 0.996999979019165
            scale: true
            epsilon: 0.0010000000474974513
          }
        }
        depth: 128
        num_layers_before_predictor: 4
        kernel_size: 3
        class_prediction_bias_init: -4.599999904632568
        share_prediction_tower: true
        use_depthwise: true
      }
    }
    anchor_generator {
      multiscale_anchor_generator {
        min_level: 3
        max_level: 7
        anchor_scale: 4.0
        aspect_ratios: 1.0
        aspect_ratios: 2.0
        aspect_ratios: 0.5
        scales_per_octave: 2
      }
    }
    post_processing {
      batch_non_max_suppression {
        score_threshold: 9.99999993922529e-09
        iou_threshold: 0.6000000238418579
        max_detections_per_class: 100
        max_total_detections: 100
        use_static_shapes: false
      }
      score_converter: SIGMOID
    }
    normalize_loss_by_num_matches: true
    loss {
      localization_loss {
        weighted_smooth_l1 {
        }
      }
```

```
      classification_loss {
        weighted_sigmoid_focal {
          gamma: 2.0
          alpha: 0.25
        }
      }
      classification_weight: 1.0
      localization_weight: 1.0
    }
    encode_background_as_zeros: true
    normalize_loc_loss_by_codesize: true
    inplace_batchnorm_update: true
    freeze_batchnorm: false
  }
}
train_config {
  batch_size: 8
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_crop_image {
      min_object_covered: 0.0
      min_aspect_ratio: 0.75
      max_aspect_ratio: 3.0
      min_area: 0.75
      max_area: 1.0
      overlap_thresh: 0.0
    }
  }
  sync_replicas: true
  optimizer {
    momentum_optimizer {
      learning_rate {
        cosine_decay_learning_rate {
          learning_rate_base: 0.1
          total_steps: 10000
          warmup_learning_rate: 0.1
          warmup_steps: 1000
        }
      }
      momentum_optimizer_value: 0.8999999761581421
    }
    use_moving_average: false
  }
  fine_tune_checkpoint: "/content/training_demo/pre-trained-
models/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint/ckpt-0"
  num_steps: 10000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
  fine_tune_checkpoint_type: "detection"
  fine_tune_checkpoint_version: V2
}
train_input_reader {
  label_map_path: "/content/training_demo/annotations/label_map.pbtxt"
  tf_record_input_reader {
    input_path: "/content/training_demo/annotations/train.record"
```

```
  }
}
eval_config {
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
  label_map_path: "/content/training_demo/annotations/label_map.pbtxt"
  shuffle: false
  num_epochs: 1
  tf_record_input_reader {
    input_path: "/content/training_demo/annotations/val.record"
  }
}
```

LAMPIRAN Kode *Inference* untuk pengujian

```
"""
Object Detection (On Image) From TF2 Saved Model
=====================================
"""

import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow log
ging (1)
import pathlib
import tensorflow as tf
import cv2
import argparse
from google.colab.patches import cv2_imshow

# Enable GPU dynamic memory allocation
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)

# PROVIDE PATH TO IMAGE DIRECTORY
IMAGE_PATHS = '/content/training_demo/images/test/aridlsu (14).jpg'


# PROVIDE PATH TO MODEL DIRECTORY
PATH_TO_MODEL_DIR = '/content/training_demo/exported_models/my_model'

# PROVIDE PATH TO LABEL MAP
PATH_TO_LABELS = '/content/training_demo/annotations/label_map.pbtxt'

# PROVIDE THE MINIMUM CONFIDENCE THRESHOLD
MIN_CONF_THRESH = float(0.60)

# LOAD THE MODEL
```

```
import time
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils

PATH_TO_SAVED_MODEL = PATH_TO_MODEL_DIR + "/saved_model"

print('Loading model...', end='')
start_time = time.time()

# LOAD SAVED MODEL AND BUILD DETECTION FUNCTION
detect_fn = tf.saved_model.load(PATH_TO_SAVED_MODEL)

end_time = time.time()
elapsed_time = end_time - start_time
print('Done! Took {} seconds'.format(elapsed_time))

# LOAD LABEL MAP DATA FOR PLOTTING

category_index = label_map_util.create_category_index_from_labelmap(P
ATH_TO_LABELS,
                                                                    u
se_display_name=True)

import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')   # Suppress Matplotlib warnings

def load_image_into_numpy_array(path):
    """Load an image from berkas into a numpy array.
    Puts image into numpy array to feed into tensorflow graph.
    Note that by convention we put it into a numpy array with shape
    (height, width, channels), where channels=3 for RGB.
    Args:
      path: the berkas path to the image
    Returns:
      uint8 numpy array with shape (img_height, img_width, 3)
    """
    return np.array(Image.open(path))

print('Running inference for {}... '.format(IMAGE_PATHS), end='')

image = cv2.imread(IMAGE_PATHS)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image_expanded = np.expand_dims(image_rgb, axis=0)
```

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

# input_tensor = np.expand_dims(image_np, 0)
detections = detect_fn(input_tensor)

# All outputs are batches tensors.
# Convert to numpy arrays, and take index [0] to remove the batch dim
ension.
# 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'].ast
ype(np.int64)

image_with_detections = image.copy()

# SET MIN_SCORE_THRESH BASED ON YOU MINIMUM THRESHOLD FOR DETECTIONS
viz_utils.visualize_boxes_and_labels_on_image_array(
      image_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=0.5,
      agnostic_mode=False)

print('Done')
# DISPLAYS OUTPUT IMAGE
cv2_imshow(image_with_detections)
# CLOSES WINDOW ONCE KEY IS PRESSED
```

LAMPIRAN *Log* aktivitas yang menampilkan nilai evaluasi

```
INFO:tensorflow:Performing evaluation on 52 images.
I0627 11:58:47.933328 139886884796288 coco_evaluation.py:293]
Performing evaluation on 52 images.
```

```
creating index...
index created!
INFO:tensorflow:Loading and preparing annotation results...
I0627 11:58:47.933736 139886884796288 coco_tools.py:116] Loading and
preparing annotation results...
INFO:tensorflow:DONE (t=0.00s)
I0627 11:58:47.936924 139886884796288 coco_tools.py:138] DONE
(t=0.00s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=0.17s).
Accumulating evaluation results...
DONE (t=0.05s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all |
maxDets=100 ] = 0.808
 Average Precision  (AP) @[ IoU=0.50      | area=   all |
maxDets=100 ] = 1.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all |
maxDets=100 ] = 1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small |
maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium |
maxDets=100 ] = 0.750
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large |
maxDets=100 ] = 0.809
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=
1 ] = 0.831
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=
10 ] = 0.831
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all |
maxDets=100 ] = 0.831
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small |
maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium |
maxDets=100 ] = 0.800
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large |
maxDets=100 ] = 0.830
INFO:tensorflow:Eval metrics at step 10000
I0627 11:58:48.162491 139886884796288 model_lib_v2.py:1007] Eval
metrics at step 10000
INFO:tensorflow:   + DetectionBoxes_Precision/mAP: 0.808254
I0627 11:58:48.164604 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Precision/mAP: 0.808254
INFO:tensorflow:   + DetectionBoxes_Precision/mAP@.50IOU: 1.000000
I0627 11:58:48.165417 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Precision/mAP@.50IOU: 1.000000
INFO:tensorflow:   + DetectionBoxes_Precision/mAP@.75IOU: 1.000000
I0627 11:58:48.166091 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Precision/mAP@.75IOU: 1.000000
INFO:tensorflow:   + DetectionBoxes_Precision/mAP (small): -1.000000
I0627 11:58:48.166735 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Precision/mAP (small): -1.000000
INFO:tensorflow:   + DetectionBoxes_Precision/mAP (medium): 0.750000
I0627 11:58:48.167369 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Precision/mAP (medium): 0.750000
```

```
INFO:tensorflow:   + DetectionBoxes_Precision/mAP (large): 0.808896
I0627 11:58:48.168012 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Precision/mAP (large): 0.808896
INFO:tensorflow:   + DetectionBoxes_Recall/AR@1: 0.830769
I0627 11:58:48.168672 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Recall/AR@1: 0.830769
INFO:tensorflow:   + DetectionBoxes_Recall/AR@10: 0.830769
I0627 11:58:48.169304 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Recall/AR@10: 0.830769
INFO:tensorflow:   + DetectionBoxes_Recall/AR@100: 0.830769
I0627 11:58:48.169956 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Recall/AR@100: 0.830769
INFO:tensorflow:   + DetectionBoxes_Recall/AR@100 (small): -1.000000
I0627 11:58:48.170587 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Recall/AR@100 (small): -1.000000
INFO:tensorflow:   + DetectionBoxes_Recall/AR@100 (medium): 0.800000
I0627 11:58:48.171233 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Recall/AR@100 (medium): 0.800000
INFO:tensorflow:   + DetectionBoxes_Recall/AR@100 (large): 0.829515
I0627 11:58:48.171957 139886884796288 model_lib_v2.py:1010]    +
DetectionBoxes_Recall/AR@100 (large): 0.829515
INFO:tensorflow:   + Loss/localization_loss: 0.044297
I0627 11:58:48.172528 139886884796288 model_lib_v2.py:1010]    +
Loss/localization_loss: 0.044297
INFO:tensorflow:   + Loss/classification_loss: 0.097607
I0627 11:58:48.173126 139886884796288 model_lib_v2.py:1010]    +
Loss/classification_loss: 0.097607
INFO:tensorflow:   + Loss/regularization_loss: 0.149588
I0627 11:58:48.173706 139886884796288 model_lib_v2.py:1010]    +
Loss/regularization_loss: 0.149588
INFO:tensorflow:   + Loss/total_loss: 0.291493
I0627 11:58:48.174335 139886884796288 model_lib_v2.py:1010]    +
Loss/total_loss: 0.291493
INFO:tensorflow:Waiting for new checkpoint at
/content/training_demo/models/my_ssd_mobnet
I0627 12:03:17.206951 139886884796288 checkpoint_utils.py:140]
Waiting for new checkpoint at
/content/training_demo/models/my_ssd_mobnet
```