DETEKSI TAJWID NUN MATI PADA AYAT AL-QURAN DENGAN METODE CONVOLUTIONAL NEURAL NETWORK MENGGUNAKAN MODEL TRAINING SSD MOBILENET

SKRIPSI

Diajukan Oleh:

MEGA ELLYADI
NIM. 180705020
Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI AR-RANIRY
BANDA ACEH
2022 M/1443 H

DETEKSI TAJWID NUN MATI PADA AYAT AL-QURAN DENGAN METODE CONVOLUTIONAL NEURAL NETWORK MENGGUNKAN MODEL TRAINING SSD MOBILENET

PERSETUJUAN PEMBIMBING

SKRIPSI

Diajukan kepada Fakultas Sains dan Teknologi
Universitas Islam Negeri Ar-Raniry Banda Aceh
Sebagai Beban Studi Memperoleh Gelar Sarjana Dalam Ilmu Teknologi Informasi

Oleh

MEGA ELLYADI
NIM. 180705020
Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

Disetujui Oleh:

Pembimbing I,	Pembimbing II,
Bustami, M.Sc	**Nazaruddin Ahmad, M.T**
NIDN: 2008048601	**NIDN: 0105068202**

DETEKSI TAJWID NUN MATI PADA AYAT AL-QURAN DENGAN METODE CONVOLUTIONAL NEURAL NETWORK MENGGUNKAN MODEL TRAINING SSD MOBILENET

PENGESAHAN SIDANG

SKRIPSI

Telah diuji oleh Panitia Ujian Munaqasyah Skripsi
Fakultas Sains dan Teknologi UIN Ar-Raniry dan dinyatakan Lulus
Serta diterima sebagai Salah Satu Beban Studi Program Sarjana (S-1)
Dalam Ilmu Teknologi Informasi

Pada Hari/Tanggal: Kamis 21 Juli 2022
22 Dzulhijah 1443 H

Panitia Ujian Munaqasyah Skripsi

Ketua,

Bustami, M.Sc
NIDN: 2008048601

Seketaris,

Nazaruddin Ahmad, M.T
NIDN: 0105068202

Penguji I,

Khairan AR, M.Kom
NIDN: 2004078602

Penguji II,

Mulkan Fadhli, M.T
NIDN: 1328118801

Mengetahui,
Dekan Fakultas Sains dan Teknologi
Universitas Islan Negeri Ar-Raniry Banda Aceh

Dr. Azhar Amsal, M.Pd
NIDN: 2001066802

LEMBAR PERNYATAAN KEASLIAN KARYA ILMIAH/SKRIPSI

Yang bertanda tangan di bawah ini:

Nama	: Mega Ellyadi
NIM	: 180705020
Program Studi	: Teknologi Informasi
Fakultas	: Sains Dan Teknologi
Judul Skripsi	: Deteksi Tajwid Nun Mati Pada Ayat Al-Quran Dengan Metode Convolutional Neural Network Menggunakan Model Training SSD Mobilenet

Dengan ini menyatakan bahwa dalam penulisan skripsi ini, saya:

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

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

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

Banda Aceh, 21 Agustus 2022

Yang Menyatakan,

Mega Ellyadi

ABSTRAK

Nama : Mega Ellyadi
NIM : 180705020
Program Studi : Teknologi Informasi
Fakultas : Sains dan Teknologi (FST)

Fakultas : Deteksi Tajwid Nun Mati Pada Ayat Al-Quran Dengan Metode Convolutional Neural Network Menggunakan Model Training SSD Mobilenet.

Tanggal Sidang : 21 Juli 2022 / 22 Dzulhijah 1443 H
Tebal Skripsi : 130 Halaman
Pembimbing I : Bustami, M.Sc
Pembimbing II : Nazaruddin Ahmad, M.T
Kata Kunci : Tajwid, CNN dan SSD Mobilenet

Al-Quran merupakan pedoman hidup yang penting untuk dipelajari kaum muslimin, adapun langkah awal mempelajari Al-Quran yaitu membaca dengan baik dan benar sesuai ketentuan *tajwid*, salah satunya hukum *nun mati* yang terbagi menjadi *izhar, ikhfah, iqlab dan idgham*. Algoritma yang populer pada *deep learning* yang juga termaksud kedalam bagian *machine learning* yaitu *Convolutional Neural Network* (CNN) yang mana model ini untuk melakukan klasifikasi pada gambar, suara, tesk dan video. Dalam penelitian ini menerapkan (CNN) dan *training SSD Mobilenet* dalam pembuatan sebuah model untuk mendeteksi *tajwid nun* mati yang mana kedepannya pendeteksian objek *tajwid nun* mati tersebut dapat dijadikan langkah awal dalam sistem pendeteksian objek khususnya objek *tajwid nun mati*. Jumlah dataset yang digunakan 350 gambar *tajwid* dengan metode *random sampling*. Implementasi CNN menggunakan *framework Tensorflow* dengan Bahasa pemograman *python* yang diaplikasikan di *website jupyter notebook*. Berdasarkan dari hasil penelitian didapatkan tingkat akurasi data *single* dengan nilai akurasi 84% dan data *multyple* nilai akurasi 80%. Data *single presisi* mendapatkan nilai 100% dan data *multiple* dengan nilai 100%, data *recall single* mendapatkan nilai 85% dan data *recall multiple* dengan nilai 84% dan terakhir presentase hasil *F1-Score* data *single* 92,5% dan presentase hasil *F1-Score* data *multyple* 92%. Dari penelitian ini disimpulkan bahwa kinerja dari pengujian yang telah dibuat berjalan dengan baik dan dapat mendeteksi hukum bacaan *tajwid* pada Al-Quran.

Kata kunci : Tajwid, CNN dan SSD Mobilenet

KATA PENGANTAR

بِسْـــــــــــــمِ اللهِ الرَّحْمَنِ الرَّحِيْمِ

Segala puji dan syukur selalu kita panjatkan atas kehadiran Allah SWT yang atas segala rahmat dan hidayah-Nya kita masih dapat melihat Alam semesta yang indah ini. Tak lupa pula shalawat beriring salam selalu kita panjatkan untuk tuntunan suri tauladan Baginda Rasulullah Shallauhu'alaihiwasalam dan beserta keluarga dan sahabat beliau yang senantiasa menjunjung tinggi nilai-nilai keislaman serta menggali ilmu yang tiada habisnya yang sampai saat ini masih bisa dinikmati oleh setiap manusia, sehingga penulis dapat menyelesaikan skripsi yang berjudul "**Deteksi Tajwid Nun Mati Pada Ayat Al-Quran Dengan Metode Convolutional Neural Network Menggunakan Model Training SSD Mobilenet".**

Penulisan skripsi ini adalah salah satu syarat untuk mendapatkan gelar sarjana pada Fakultas Sains dan Teknologi di UIN Ar-Raniry, Banda Aceh. Dalam penyusunan skripsi ini, penulis banyak sekali menghadapi kesulitan dalam teknik penulisan maupun dalam penguasaan bahan. Walaupun demikian, penulis tidak putus asa dalam menghadapi permasalahan, dan dengan adanya dukungan dari berbagai pihak, terutama sekali dosen pembimbing kesulitan yang dihadapi dapat teratasi. Pada kesempatan ini, penulis mengucapkan ribuan terima kasih kepada:

1. Kepada kedua orang tua yang penulis cintai karena Allah, Iswadi dan Erlinawati yang senantiasa mendoakan, membimbing, mendidik, serta memberikan semangat dan dukungan kebaikan tanpa batas, semoga Allah membalas segala jasa-jasanya dengan kebaikan yaitu SurgaNya.

2. Kepada kakak dan adik penulis tercinta, Nana Erdiana dan Fauzan terimakasih atas doa dan segala dukungan.
3. Kepada ustadzah saya Regina Fadilla Panjaitan S.Psi yang telah mendukung, memotivasi, dan mendoakan sehingga penulisan karya ilmiah ini dapat terselesaikan.
4. Segenap keluarga dan sahabat yang selalu menyemangati dan membantu penulis untuk menyelesaikan skripsi ini dari awal hingga akhir.
5. Bapak Dekan Fakultas Sains dan Teknologi Dr. Azhar Amsal,M.Pd yang selalu mendukung dan memberi motivasi untuk kami.
6. Bapak Bustami, M.Sc sebagai pembimbing pertama dan Bapak Nazaruddin Ahmad, M.T sebagai pembimbing kedua, yang telah meluangkan waktunya dan mencurahkan pemikirannya dalam membimbing penulis untuk menyelesaikan skripsi ini.
7. Ketua Prodi Teknologi Informasi Ibu Eriawati, S.Pd., M.Pd. Sekretaris Prodi Teknologi Informasi Bapak Hendri Ahmadian, S.Si.,M.IM, serta staf Prodi yang telah ikut membantu proses pelaksanaan penelitian.
8. Kepada Staf Prodi Ibu Cut Ida Rahmadiana S,Si. yang telah membantu membantu penulis dalam hal pengurusan administrasi dan surat-surat untuk keperluan pengelesaian tugas akhir.
9. Bapak dan Ibu dosen Program Studi Teknologi Informasi yang telah memberikan ilmu pengetahuan dalam bidang teknologi inofrmasi kepada penulis sehingga penulis mampu menyelesaikan tugas akhir karya ilmiah ini.

10. Sahabat dan teman-teman mahasiswa program studi Teknologi Informasi angkatan 2018 terkhusus nya kepada Sri Maulida, Adinda Gusnita dan Nura Nabilah, serta seluruh keluarga Teknologi Informasi yang telah memberikan dukungan dan semangat dalam penyelesaian tugas akhir ini.
11. Dan untuk semuanya yang tidak dapat penulis sebutkan satu persatu.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengaruh teknologi informasi telah masuk ke berbagai bidang aktivitas manusia dan memberikan manfaat yang besar dalam berbagai bidang yaitu dalam bidang pendidikan, kesehatan, politik, ekonomi dan bidang agama. Beberapa alat teknologi yang dimanfaatkan untuk belajar dalam bidang agama, baik mempelajari Al-Quran, hadist, fiqih dan ilmu agama lainnya seperti laptop, android, infokus, whatsapp.

Menurut Naharuddin et al.,2013 yang dikutip oleh (Hidayatullah, 2020) Al-Quran adalah firman Allah dan sekaligus petujuk bagi umat muslim. Hampir seluruh umat muslim tahu bagaimana cara membaca alquran, namun tidak semua paham mengenai *tajwid* yang benar. Pada penelitian Burhanudin terdapat 67% hasil data yang memyebabkan ketidakpahaman dan kesulitan pada hukum bacaan *tajwid mad far'i* (Burhanudin, 2018). Dalam mempelajari teknik membaca Al-Qur'an setiap orang harus mempelajari ilmu membaca Al-Qur'an dengan baik dan benar, jika tidak memiliki ilmu tersebut maka akan salah dalam membaca Al-Qur'an maka ilmu tersebut adalah ilmu tajwid.

Artificial Intelligence (AI) atau kecerdasan buatan, kecerdasan buatan merupakan mesin yang betindak dan meniru kecerdasan layaknya manusia. (Yakib, 2020). *Computer vision* bagian dari *Artificial Intelligence. Computer Vision* diartikan yaitu suatu disiplin ilmu tentang pembelajaran komputer agar *computer*

mampu mengenali pola pada objek yang diamati. *Computer vision* terdapat permasalahan yang terjadi diantaranya pendeteksian objek dan klasifikasi citra.

Adapun metode dalam melakukan pendeteksian atau menemukan objek pada suatu citra, yaitu salah satunya metode *Convolutional Neural Network (*CNN*)* yang mana sering dipergunakan pada data citra seperti citra sidik jari, wajah, masker, hijab dan bahkan Al-Quran dapat dijadikan objek atau citra. CNN adalah salah satu metode pada *Deep learning* yang sering di gunakan dalam menyelesaikan masalah tentang pendeteksian suatu objek dan klasifikasi citra. Maka dapat memungkinkan menguji algoritma CNN untuk pendeteksi hukum *tajwid* Al-Quran.

Model training yang digunakan dalam proses peningkatan keakuratan dan proses pendeteksian yaitu mengunakan training SSD *Mobilenet*. Seperti dalam penelitian Hendriyana & Maulana dengan menggunakan SSD *Mobilenet* dalam penelitiannya. Identifikasi Jenis Kayu menggunakan Convolutional Neural Network dengan Arsitektur *Mobilenet*. Hasil pengujian dalam penelitian ini didapatkan tingkat akurasi sebesar 98% training, 93,3 % testing, 28% untuk recall, dan 93% untuk presisinya. (Hendriyana & Maulana, 2020). Terdapat juga penelitian Dompeipen & Najoan dalam penelitiannya "*Computer Vision Implementation for Detection and Counting the Number of Humans"* menggunakan metode *Mobilenet*-SSD dan *Centroid Tracking***.** Hasil pengujian dalam penelitian ini didapatkan tingkat akurasi dan penghitungan objek manusia sebesar 93,75%. (Dompeipen & Najoan, 2021).

Fokus penelitian ini yaitu melakukan pengenalan *tajwid nun* mati terhadap *computer* dengan proses pendeteksian dan meningkatkan akurasi menggunakan

Training SSD Mobilenet. Dalam penelitian ini peneliti menerapkan *Convolutional Neural Network* (CNN) dan *training SSD Mobilenet* dalam pembuatan sebuah model untuk mendeteksi *tajwid nun* mati yang mana kedepannya pendeteksian objek *tajwid nun* mati tersebut dapat dijadikan langkah awal dalam sistem pendeteksian objek khususnya objek *tajwid nun* mati pada citra ayat Al-Quran sehingga dapat diterapkan dan dikembangkan oleh perusahaan atau penelitian yang lainnya.

Berdasarkan penelitian-penelitian sebelumnya menunjukan bahwa model training SSD *Mobilenet* memiliki daya komputasi yang lebih kecil dibandingkan dengan arsitektur yang lain. Maka demikian pada penelitian ini pendeteksian hukum bacaan *tajwid* nun mati pada Al-Quran dengan menggunakan *training* model SSD *Mobilenet*. *Framework* yang digunakan dalam penelitian ini adalah *TensorFlow object detection* API dengan bahasa pemograman *Python*. Klasifikasi hukum bacaan *tajwid* nun mati yang akan dideteksi diantara lain *izhar, iqlab, idhgam dan ikhfah.*

1.2 Rumusan Masalah

1. Bagaimana proses pendeteksian hukum bacaan *tajwid nun mati* dari citra Al-Quran dengan algoritma CNN menggunakan training model SSD *Mobilenet?*
2. Bagaimana tingkat akurasi pendeteksian *tajwid nun mati* dengan training model SSD *Mobilenet?*

1.3 Tujuan Penelitian

1. Mengetahui proses pendeteksian hukum bacaan *tajwid nun mati* dari citra Al-Quran dengan algoritma CNN menggunakan training model SSD *Mobilenet.*
2. Mengetahui tingkat akurasi pendeteksian *tajwid nun mati* dengan training model SSD *Mobilenet.*

1.4 Batasan Masalah

Batasan penelitian ini adalah:

1. Bahasa pemograman yang dipakai pada penelitian adalah *Python*.
2. Menggunakan *Framework TensorFlow*.
3. Menggunakan aplikasi edit teks bersifat *client-server* yaitu *Jupyter Notebook.*
4. *Komputasi paralel GPU dengan Compute Unified Device Architecture (CUDA)*
5. Perangkat keras yang digunakan dengan spesifikasi sebagai berikut:
   a) 1 Unit Laptop VivoBook 14_ASUS Laptop X441UB
   b) Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 1.99 GHz
   c) RAM 4.00 GB (3.88 GB *usable*).
   d) *Harddiks Drive* 1 *Tera.*
   e) *Untuk Pemograman CUDA menggunakan VGA Nvidia Geforce* MX110

f) *Sistem Operasi Microsoft Windows 10 Home Single language 64-bit Version 21H2*

6. Dataset gambar diambil menggunakan metode *snipping tools.*
7. Data yang digunakan adalah data gambar ayat Al-Quran yang memiliki hukum bacaan nun mati.
8. Training model menggunakan SSD *Mobilenet*.

1.5 Manfaat Penelitian

1. Penelitian ini diharapkan dapat memberikan pengetahuan mengenai implementasi *deep learning* menggunakan *Convolutional Neutral Network* dengan *training SSD Mobilenet* dengan bantuan sistem komputer dan teknologi, untuk mengatasi permasalahan khususnya pada bidang pendeteksian *tajwid nun* mati.
2. Mengetahui tingkat akurasi dari implementasi *Convolutioanl Neural Network* dengan *training SSD Mobilenet* untuk dijadikan acuan pengembangan penelitian berikutnya.
3. Proses dan hasil yang dipaparkan dapat berguna sebagai langkah awal penerapan *Artificial Intelligence* dalam bidang pendeteksian *tajwid nun* mati sehingga dapat dikembangakan untuk pembuatan aplikasi yang dapat bermanfaat untuk menunjukkan keberadaan *tajwid* pada ayat Al-Quran sehingga dapat berguna dalam pengetahuan kaum muslim terhadap tajwid sehingga memperbaiki bacaan Al-Quran.

BAB II
LANDASAN TEORI

2.1 Penelitian Terdahulu

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

Penelitian mengenai "*Deteksi Hukum tajwid Mad Lazim Harfi Musyba* pada Ayat Al-Quran Menggunakan *Deep Convolutional Neural Network*", (Burhanudin, 2018). Pengujian data diatas menggunakan sampel 9 ayat dari 8 surah dalam Al-Quran, total surah yang terdapat hukum *tajwid madd lazim harfi musyba* ada 27 surah dari 114 surah Al-Quran, namun dari 27 surah banyak yang memiliki pola yang sama sehingga bisa dikumpulkan dan direpresentasikan pada 8 surah, pada pengujian sistem ini menghasilkan akurasi pendeteksian rata-rata sebesar 93,25%.

Penelitian mengenai "*Deep Learning* Untuk Deteksi Wajah Yang Berhijab Menggunakan Algoritma *Convolutional Neural Network (*CNN*)* Dengan *TensorFlow*", (Wulan Angraini, 2020). Dalam penelitian ini menggunakan metode CNN dengan masukan *image size 80x80 pixel, filter 3x3, epoch* yang dipakai 20 dan *learning rate* 0,001. Dataset *training* sebnyak 250 dan *testing* 50 dengan *image face* yang memakai kerudung terdapat 300 *image*. Dengan *ekspresi face flat dan smile*. Hasil pada akurasi yang didapat pada penelitian ini pada data *training* 92% dan *testing* 87%.

Penelitian mengenai "*Deep learning Object Detection* Pada Video Menggunakan *TensorFlow* Dan *Convolutional Neural Network*" (Dewi, 2018). Dalam penelitian ini menggunakan CNN dapat bekerja dengan baik. Keakuratan yang diperoleh pada penelitian deteksi *image* pada meja dan kursi dengan motf jepara ini memiliki presentase 70% hingga 99%. Pada hasil penelitian ini menggunakan 250.000 jumlah *steps* dengan 2 *batch size* dan 3 file model-*ckpt.*

Penelitian mengenai "Identifikasi Jenis Kayu menggunakan *Convolutional Neural Network* dengan Arsitektur *Mobilenet*" (Hendriyana & Maulana, 2020). Dataset yang di pakai berjumlah 1000 image pada 10 jenis kayu. Hasil dari pengujian mengambil 30 image secara random sebagai dataset testing. Hasil yang diperoleh pada data training memiliki presentase 95%.

Penelitian mengenai " Implementasi *Deep Learning Object Detection* Rambu K3 Pada Video Menggunakan Metode *Convolutional Neural Network* (CNN) Dengan *Tensorflow* (Mashita, 2020). Metode yang dipakai yaitu CNN dalam mengenali dan mengelompokkan sebuah objek. Pada penelitian ini mendeteksi rambu K3 evakuasi dan mendeteksi alat pemadam pembakaran dengan framework tensorflow dan training model SSD. Dataset pada penelitian ini berjumlah 1500 image. Hasil tingkat akurasi yang didapatkan dengan presentase 50%-97%.

Penelitian mengenai "*Computer Vision Implementation for Detection and Counting the Number of Humans*" (Dompeipen & Najoan, 2021). Pada penelitian ini dapat mendeteksi objek *human* pada sebuah *frame* video dan memiliki hasil yang cukup baik. Telah diuji 5 kali sebanyak 12 video. Kondisi yang telah diuji

memiliki kondisi yang berbeda dari setiap video seperti cahaya, sudut kamera saat pengambilan video. Hasil yang diperoleh pada penelitian ini memiliki tingkat kaurasi 93,75%.

Penelitian mengenai "Implementasi Sistem Cerdas Pada Otomatisasi Pendeteksian Jenis Kendaraan Di Jalan Raya" (Budiarjo, 2020). Penelitian ini memakai *system* pendeteksi objek YOLO dengan metode CNN untuk mengelompokkan dan menghitung kendaraan lewat dengan otomatis. Dataset yang digunakan berjumlah 600 *image* dengan 4 item, yaitu mobil, seperda motor, bus dan juga truk. Hasil penelitian nya mendapati 80% tingkat akurasinya.

Tabel 2.1 Perbandingan Penelitian Sejenis

Peneliti	Metode	Kasus	Jumlah DataSet	Akurasi
(Burhanudin, 2018)	CNN	Deteksi Hukum Tajwid *Mad Lazim Harfi Musyba* pada Citra Ayat Alquran	9	93,25%
(Wulan Angraini, 2020)	CNN	Deteksi Wajah Yang Berhijab	300	92%
(Dewi, 2018)	CNN	*Deep learning Object Detection* Pada Video	500	99%

(Hendriyana & Maulana, 2020)	CNN	Identifikasi Jenis Kayu	1000	95%
(Mashita, 2020)	CNN	Implementasi Deep Learning Object Detection Rambu K3	1500	97%.
(Dompeipen & Najoan, 2021)	MobileNet-SSD	Computer Vision Implementation for Detection and Counting the Number of Humans	12	93,75%.
(Budiarjo, 2020)	CNN	Implementasi Sistem Cerdas Pada Otomatisasi Pendeteksian Jenis Kendaraan Di Jalan Raya	600	80%

Sumber: penulis

2.2 Definisi Al-Quran

Al-Quran adalah firman Allah yang diturunkan kepada Nabi Muhammad SAW sebagai wahyu dari perantara malaikat Jibril dan merupakan salah satu mukjizat yang diberikan oleh Allah kepada Nabi Muhammad SAW. Al-Quran adalah kitab suci berisi petunjuk-petunjuk bagi umat manusia untuk menjalani

kehidupan sesuai dengan ketentuan yang yang Allah tetapkan. Al-Quran berisi segala aspek aturan hukum untuk dipergunakan oleh manusia selama hidupnya. Disamping itu adalah amal ibadah bagi yang membacanya, mendakwahkan, mengamalkan, memperjuangkan, menghafalkan dan membelanya. Karena Al-Quran benar dari Allah SWT dan merupakan mukjizat yang mampu menundukan umat manusia dan tidak akan mungkin dapat ditiru (Iskandar, 2014).

قُلْ لَّىِٕنِ اجْتَمَعَتِ الْاِنْسُ وَالْجِنُّ عَلٰٓى اَنْ يَّأْتُوْا بِمِثْلِ هٰذَا الْقُرْاٰنِ لَا يَأْتُوْنَ بِمِثْلِهٖ وَلَوْ كَانَ بَعْضُهُمْ لِبَعْضٍ ظَهِيْرًا

Artinya *"Katakanlah, Sesungguhnya jika manusia dan jin berkumpul untuk membuat yang serupa (dengan) Al-Qur'an ini, mereka tidak akan dapat membuat yang serupa dengannya, sekalipun mereka saling membantu satu sama lain*." (QS. *Al-Isra*: 88)

Membaca Al-Quran adalah awal untuk dapat melaksanakan hak-hak Al-Quran lainnya. Terdapat enam hak hak Al-Quran yang harus dipenuhi oleh umat islam yaitu seperti membaca, menghafal, mengamalkan, memperjuangkan, membela, mendakwahkan Al-Quran. Hukum mempelajari Al-Quran adalah wajib, karna terdapat dalam firman Allah dalam Al-Quran surah *Al-Muzammil* ayat 4.

2.3 Ilmu Tajwid

Tajwid berasal dari kata *"Jawwada"* yang memiliki arti sesuatu yang indah, bagus dan membaguskan (Ashadiqi et al., 2020). *Tajwid* menurut bahasa adalah membaguskan. Menurut istilah *tajwid* adalah tempat keluarnya huruf sesuai dengan hak dan mustahak. Arti hak merupakan sifat asli yang bersama dengan huruf seperti *al-jahr, isti'la, istifal* dan yang lainnya. Sedangkan mustahak memiliki arti yaitu

sifat yang nampak sewaktu-waktu, seperti halnya *tarqiq, tafkhim, ikhfah* dan yang lainnya.

Maka *tajwid* merupakan ilmu yang mempelajari bagaimana mengeluarkan huruf sesuai dengan makhrajnya sesuai dengan kaidah-kaidahnya sehingga seseorang dapat mengetahui cara-cara membaca Al-Quran dengan baik dan indah (Sudiarjo et al., 2015). Dalam ilmu *tajwid* huruf huruf hijaiyah berjumlah 29 huruf dengan berbeda-beda haraqah atau baris dan macam-macam hubungannya.

2.4 Hukum Mempelajari Ilmu Tajwid

Hukum dalam memperlajari hukum *tajwid* yaitu *fardhu kifayah*, tetapi membaca Al-Quran sesuai dengan kaidah ilmu *tajwid* yaitu *fardhu 'ain*. Maka jika terdapat Qori membaca Al-Quran dengan benar dan bagus namun tidak mengetahui istilah hukum-hukum *tajwid* yang ia baca maka baginya telah cukup jika ada kaum muslimin yang lain telah banyak memperlajari ilmu tajwid.

Namun lain halnya dengan orang yang tidak dapat membaca Al-Quran dengan sesuai aturan-atuaran dalam ilmu tajwid, maka baginya menjadi wajib untuk terus berusaha membaguskan bacaan sesuai dengan standar yang telah ditetapkan oleh Baginda Rasullullah *Sholallohu'alaihi Wasallam.*

Terdapat beberapa dalil kewajiban membaca Al-Quran sesuai dengan aturan dalam membaca atau ilmu *tajwid* yaitu sebagai berikut:

1. Dalil Al-Quran Surah *Al-Muzammil*: 4

أَوْ زِدْ عَلَيْهِ وَرَتِّلِ الْقُرْآنَ تَرْتِيلًا

Artinya: "*Ataupun lebihkan (sedikit) daripadanya; dan bacalah Al-Quran dengan Tartil* (Surah *Al-Muzzammil*, 73:4)" Maka ini adalah sebuah firman Allah bagi kita untuk dapat membaca alquran sesuai dengan apa yang diturunkan olehNya.

2. Dalil Al-Quran Surah *Al-Baqarah* :121

ءَالَّذِيْنَ اٰتَيْنٰهُمُ الْكِتٰبَ يَتْلُوْنَهٗ حَقَّ تِلَاوَتِهٖۗ اُولٰۤىِٕكَ يُؤْمِنُوْنَ بِهٖۗ وَمَنْ يَّكْفُرْ بِهٖ فَاُولٰۤىِٕكَ هُمُ الْخٰسِرُوْنَ

Artinya: "*Orang-orang yang telah Kami beri Kitab, mereka membacanya sebagaimana mestinya, mereka itulah yang beriman kepadanya. Dan barangsiapa ingkar kepadanya, mereka itulah orang-orang yang rugi*. (Surah *Al-Baqarah* :121)

Membaca sebenarnya yaitu sesuai dengan kaidah dalam membaca Al-Quran dengan baik agar makna tidak berubah ketika dalam membaca Al-Quran (Sudiarjo et al., 2015).

2.5 Hukum Bacaan *Nun* Mati

Hukum *Nun* mati adalah salah satu hukum *tajwid* yang terdapat didalam Al-Quran, *nun* mati berlaku keseluruh huruf-huruf hijaiyah dan setiap huruf tertentu berbeda beda hukum bacaan nya dan mempengaruhi bunyi yang berlainan seperti *Izhar, Idgham, Iqlab* dan *Ikhfah* (S. Ariani & Realita, 2015).

Hukum nun mati terbagi menjadi 4 bagian yaitu:

2.5.1 Hukum bacaan *izhar*

Izhar bacaan nya jelas dan terang. Apabila nun mati bertemu dengan salah satu huruf *izhar* (ء, ه, غ, ع, خ, ح) maka dibaca jelas, dan panjang bacaanya satu harakat dengan tidak berdengung.

2.5.2 Hukum bacaan *ikhfah*

Ikhfah bacaannya samar-samar. Apabila nun mati bertemu dengan salah satu huruf ikhfah (ت ، ث ، ج ، د ، ذ ، ز ، س ، ش ، ص ، ض ، ط ، ظ ،ف ، ك ، ق) maka dibaca samar-samar dan berdengung dihidung, huruf *ikhfah* terdiri dari 15 huruf.

2.5.3 Hukum bacaan *Iqlab*

Iqlab bacaanya menukar atau mengganti bacaan *nun* mati dengan huruf *mim*. Apabila nun mati bertemu dengan salah stau huruf *iqlab* (ب) maka dibaca berdengung. (A. Ariani, 2014).

2.5.4 Hukum bacaan *Idgham*

Idgham terbagi menjadi 2, yaitu *idgham bighunnah* dan *idgham bilaghunnah.*

a) ***Idgham bigunnah*** adalah apabila *nun* mati berjumpa salah satu huruf *idgham bighunnah* (ي - و - ن - م) maka dibaca dengung.

b) ***Idgham bilaghunnah*** adalah apabila *nun* mati berjumpa dengan salah satu huruf *idgham bilaghunnah* (ر - ل) maka dibaca tidak dengung melainkan seperti tidak ada *nun* matinya (Siska & Fadillah, n.d.).

Tabel 2.2 Pola Bacaan Tajwid Hukum *Nun* Mati

Pola *Tajwid*	**Huruf *Izhar***
ن (Nun Mati)	ح (Ha Kecil)
ن (Nun Mati)	خ (Kha)
ن (Nun Mati)	ع (‘Ain)
ن (Nun Mati)	غ (Ghain)
ن (Nun Mati)	ه (Ha Besar)
ن (Nun Mati)	ء (Hamzah)
Pola *Tajwid*	**Huruf *Ikhfa***
ن (Nun Mati)	ت (Ta)
ن (Nun Mati)	ث (Tsa)
ن (Nun Mati)	ج (Jim)
ن (Nun Mati)	د (Dal)
ن (Nun Mati)	ذ (Dzal)
ن (Nun Mati)	ز (Zai)
ن (Nun Mati)	س (Sin)
ن (Nun Mati)	ش (Syin)
ن (Nun Mati)	ص (Shad)
ن (Nun Mati)	ض (Dhad)
ن (Nun Mati)	ط (Tha)
ن (Nun Mati)	ظ (Zha)
ن (Nun Mati)	ف (Fa)
ن (Nun Mati)	ق (Qaf)
ن (Nun Mati)	ك (Kaf)
Pola *Tajwid*	**Huruf *Iqlab***
ن (Nun Mati)	ب (Ba)
Pola *Tajwid*	**Huruf *Idgham Bighunnah***
ن (Nun Mati)	م (Mim)
ن (Nun Mati)	ن (Nun)
ن (Nun Mati)	و (Waw)

ن (Nun Mati)	ي (Yaa)
Pola *Tajwid*	**Huruf *Idgham Bilaghunnah***
ن (Nun Mati)	ر (Lam)
ن (Nun Mati)	ل (Ra)

Sumber: Penulis

2.6 Pengolahan Citra

Menurut dari (Ratna, 2020) pengolahan Citra *Digital* (*Digital Image Processing*) merupakan sebuah disiplin ilmu tentang mempelajari mengolah sebuah citra, citra tersebut yaitu *image* diam atau *image* yang bergerak (video). Pengolahan Citra Digital (*Digital Image Processing)* ialah ilmu yang dipelajari mengenai suatu citra yang di analisis, diproses, diolah hingga membuahkan sebuah informasi yang dapat di pahami oleh *human*. (Ratna, 2020).

Terdapat beberapa analisis dalam pengolahan citra yaitu, *Enhancement* adalah memperbaiki kualitas citra dengan memanipulasi parameter suatu citra. *Image Restoration* adalah untuk meminimumkan kecacatan yang terdapat pada citra. *Image Compression* adalah pemampatan citra atau representasi citra dalam bentuk yang lebih kompak. *Image Segmentation* adalah memecah citra sehingga menjadi dipecah menjadi beberapa segmen. *Analisis* Citra adalah menghitung besaran kuantitatif pada suatu citra sehingga menghasilkan pendeskripsian citra. *Image Reconstruction* adalah pembentukan ulang *object* pada beberapa citra (Pamungkas, 2020).

2.6.1 Definisi Citra

Citra yaitu representasi, kemiripan atau imitasi dari sebuah *object* yang merupakan tampilan dua dimensi yang di tangkap oleh kamera atau ditampilkan dilayar *monitor*, dicetak dan yang lainnya (Pamungkas, 2020). Citra yaitu suatu *output* (keluaran) dari sistem perekam data yang memiliki sifat *analog, optic* dan *digital*. Citra dapat digolongkan menjadi dua kelompok yaitu citra dapat dilihat dan citra tidak dapat terlihat. Citra tampak dapat dimisalkan di dalam kehidupan sehari hari yaitu seperti foto keluarga, lukisan, citra optis seperti *hologram*, dan apapun yang tampak pada layar *monitor* dan televisi (Wantania, 2020).

2.6.2 Definisi Citra *Digital*

Digital image merupakan ilmu yang memperlajari cara manipulasi, memodifikasi, memperbaiki ataupun mengubah kualitas suatu citra dengan proses *digital* atau dilakukan menggunkan komputer agar dapat menghasilkan hasil yang lebih baik. (Muliadi, 2020)

Citra *digital* merupakan representasi citra dua dimensi sebagai kumpulan dari nilai *digital* yang sering disebut yaitu elemen gambar atau *pixel*. *Pixel* merupakan elemen terkecil dari penyusunan citra yang memiliki kandungan nilai yang mewakili kecerahan warna pada sebuah titik tertentu yang terdapat pada citra. Ukuran pada citra biasanya dinyatakan dalam banyak *pixel* sehingga ukuran pada citra selalu bernilai bulat. Format citra *digital* yang sering digunakan yaitu citra *biner*. (Muliadi, 2020).

2.7 *Computer Vision*

Computer Vision adalah ilmu yang mempelajari tentang bagaimana *computer* dapat mengenali sebuah *object* yang disekelilingnya dan mampu dalam menganalisannya. *Computer visio*n adalah gabungan dari pengolahan citra, *mechine learning*, pengenalan pola dan komputer *graphics*. *Computer vision* juga merupakan kombinasi dari sistem pencahayaan dan analisa citra. *Computer vision* bersama dengan *Artificial Intelligence* akan dapat menghasilkan sesuatu sistem yang *intelligence visual.* (Manajang et al., 2020)

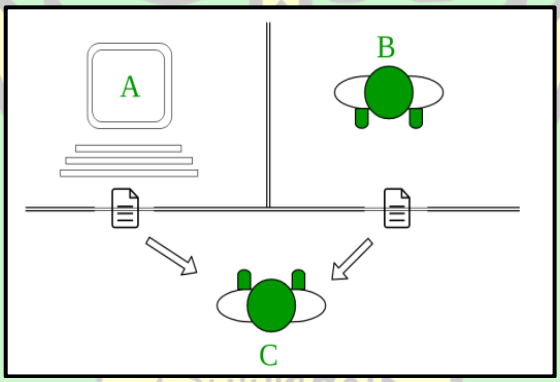
Computer vision adalah teknologi yang sering digunakan pada zaman ini. Teknologi ini juga salah satu bagian bidang dari teknologi *Artificial Intelligence*. Otak manusia dapat mengenali wajah manusia yang lain dengan cepat walaupun sekali bertemu, berbeda hal dengan komputer yaitu gambar hanya sekumpulan dari banyak *pixel*. Oleh karena itu inti dari teknologi *computer vision* adalah komputer dapat mengenali *object* dengan baik, sehingga dari *object* terdapat informasi-informasi penting. Dengan sebutan lain bahwa *computer vision* bertujuan untuk mengajarkan komputer dapat membuat *pixel-pixel object* yang diamati disekelilingnya menjadi terasa hidup seperti halnya dunia nyata dan juga mampu duplikasi kemampuan dari penglihatan manusia dalam teknologi sehingga dapat dipahami arti dari *object* yang dimasukkan (Manajang et al., 2020).

2.8 *Artificial Intelligence*

Artificial Intelligence (AI) atau kecerdasan buatan dimulai pada tahun 1950-an. AI diperkenalkan Alan Turing, pada saat itu Alan Turing melakukan tes

terhadap kecerdasan buatan dikenal dengan *turing test* yang tujuannya untuk dapat mendefinisikan sebuah mesin cerdas. Pengujian mesin cerdas itu dengan cara meletakkan mesin diruang A, dan manusia diletakkan pada ruang B, dan penguji terletak diruang C. Penguji mencoba melakukan komunikasi antara keduanya, apabila penguji tidak dapat membedakan antara ruang A dan ruang B maka *Turing Test* tersebut berhasil. Untuk ilustrasinya dilihat pada gambar (Yakib, 2020).

Kecerdasan buatan atau *Artificial Intelligence* merupakan mesin yang betindak dan meniru kecerdasan layaknya manusia. Kecerdasan buatan adalah komputer yang dibentuk untuk dapat mengetahui dan memodelkan serupa dengan proses befikirnya manusia.



Gambar 2.1 *Ilustrasi Turing Test* (Yakib, 2020)

Kecerdasan buatan telah banyak sekali membantu dan menyelesaikan kehidupan sehari-harinya manusia, dan bahkan dapat diartikan kecerdasan buatan akan menjadi masa depan (Yakib, 2020).

Terdapat beberapa metode yang dikembangkan dalam *Artificial Intelligence* (Yakib, 2020) yaitu:

1. *Fuzzy logic* (FL)

   Teknik ini adalah logika yang dipakai oleh mesin agar dapat memberi keputusan tanpa rasa tidak kaku dan dapat menyesuaikan kondisi atau beradaptasi. *Logika fuzzy* diterapkan salah satunya adalah sistem pengereman kereta api dijepang.

2. *Evolutionary Computing* (EC)

   Teknik ini adalah skema evolusi yang mana dapat menyeleksi individu terbaik yang akan menjadi generasi selanjutnya, skema ini dicontohkan yaitu Algoritma Genetika yang menggunakan mutasi dan kawin silang.

3. *Machine learning*

   Pembelajaran mesin saat ini sedang populer karna banyak yang menggunakan nya untuk dapat meniru segala prilaku pada manusia dan menurukan kecerdasan manusia (Rena, 2019).

2.9 *Machine Learning*

Machine learning didefinisikan pertama kali oleh *Athur Samuel* pada tahun 1959. Menurut *Athur Samuel* yang dikutip oleh (Fikriya et al., 2017) menyatakan bahwa *machine learning* merupakan bidang studi yang memberikan kemampuan program komputer untuk belajar jelas pada program, sehingga dalam proses pembelajaran yang dilakukan oleh komputer dengan program yang sederhana saja maka komputer sudah bisa melakukan proses pembelajaran menggunakan *machine learning*. Kemudian terdapat beberapa pendapat yang dijabarkan dari para ahli

seperti dalam buku Wahyono (2018) berjudul "*Fundamental of Python For Machine Learning*" sebagai berikut:

1. Komputer dapat belajar dari berbagai pengalaman terhadap tugasnya dan juga dapat meningkatkan kinerja (Mitchel 1997).
2. Kecerdasan buatan yang terdapat didalam komputer dapat memiliki kemampuan dalam belajar dari data tanpa dengan jelas harus sesuai dengan intruksi yang telah terprogram (Budiharto 2016).

Machine learning melakukan pendekatan pada kecerdasan buatan yang banyak dipakai dalam menirukan perilaku manusia dan dapat melakukan analisa dan penyelesaian masalah. *Machine learning* memerlukan data sehingga diistilahkan sebagai *learn from data*. *Machine learning* dalam informasi sebelumnya dapat sebagai metode komputasi untuk meningkatkan performa atau prediksi yang akurat (Rena, 2019).

Menurut (Roihan et al., 2020) terdapat tiga kategori kedalam pembelajaran *Machine learning*, yaitu:

1. *Supervised Learning*

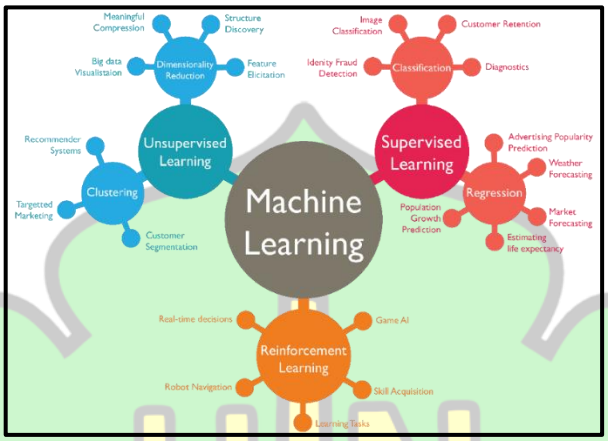
   Supervised learning merupakan kelompok data yang ditandai label pengelompokkan kelas yang tidak diketahuinya.

2. *Unsupervised Learning*

   Unsupervised learning tidak dibutuhkan untuk menandai sebuah label dan hasil nya tidak diketahui.

3. *Reinforcement Learning*

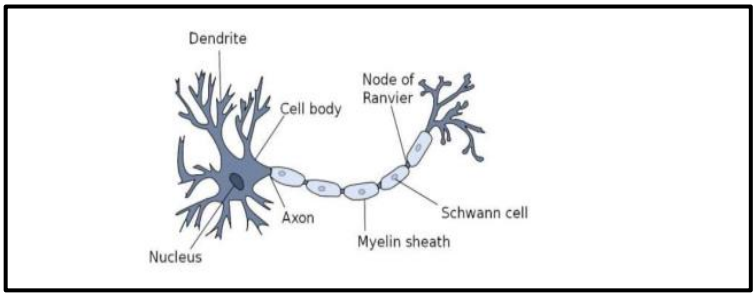
Teknik dapat berkerja dengan suasana yang dinamis namun harus menyelesaikan hingga akhir.



Gambar 2.2 *Machine Learning* (Rena, 2019)

2.10 *Artificial Neural Network* (Jaringan Saraf Tiruan)

Jaringan Saraf Tiruan (JST) atau *Artificial Neural Network* merupakan cabang ilmu dari AI, JST tersebut terinpirasi dari jaringan saraf manusia. JST adalah system pemrosesan informasi yang meniru dari saraf manusia yang terdiri atas *neuron* yang banyak. Kumpulan *neuron* saling berelasi atar neuron sehingga mempunyai sebagai pembelajari pada suatu mesin. *Neuron* merupakan unit pemrosesan terkecil yang terdapat pada otak manusia. Tersusun dari 1013 *neuron* dan 1015 *dendrit* yang saling terhubung satu sama lain (Muliadi, 2020). Seperti ilustrasi gambar berikut.



Gambar 2.3 *Stuktur Sederhana Sebuah Neuron* (Muliadi, 2020)

Fungsi *dendrit* sebagai penyampaian sinyal ke badan sel yang dikirimkan kejaringan lain dengan menggunakan *axon*. Pada suatu *neuron* tersusun atas 3 komponen yaitu:

1. ***Dendrites***, adalah saluran sinyal berfungsi sebagai pengirim informasi ke inti sel.
2. **Badan Sel (*Cell Body*),** adalah tempat proses komputasi pada sinyal input yang didapat pada dendrit yang akan menghasilkan *output* sinyal nantinya dikirim ke *neuron* yang lainnya.
3. ***Axon***, adalah befungsi sebagai pengirim sinyal *output* kepada *neuron* yang lain.

Jaringan Saraf Tiruan atau *Artificial Neural Network* (ANN) adalah metode yang dipakai dalam meramal dan mengenali pola. Dalam peramalan biasanya dilakukan untuk meramal harga saham, nilai tukar, peramalan cuaca dengan menggunakan metode CNN. Kemudian dalam pengenalan pola biasa nya digunakan dalam jaringan saraf tiruan yaitu tanda tangan, pola huruf maupun wajah dan suara. ANN adalah sistem yang adaptif sehingga dapat mengubah stuktur dan menyelesaikan permasalahan sesuai dengan informasi yang didapat pada internal

dan eksternal. ANN telah banyak dipakai dalam area yang luas. ANN bersifat fleksibel dan menghasilakan *output* yang konsisten (Rena, 2019).

2.11 *Deep Learning*

Deep learning sering dikenal dengan istilah *Deep Structured Learning)* atau dikenal dengan yang lain seperti pembelajaran hierarki (*Hierarchical Learning*) yang merupakan sebuah ilmu *machine learning* yang terdiri dari algoritma pemodelan dengan memiliki abstraksi tingkat tinggi pada sekumpulan data (Lorentius et al., 2019).

Deep learning merupakan bagian dari *machine learning* yang dapat memberikan pelajaran pada komputer agar melakukan sesuatu yang alami terhadap *human*. *Deep learning* adalah metode yang memanfaatkan *Artificial Neural Network* atau jaringan saraf tiruan, dimana *Artificial Neural Network* dibuat mirip dengan otak manusia. *Deep learning* pada komputer melakukan pembelajaran dengan cara mengklarifikasi langsung pada objek, seperti Gambar, suara, teks dan yang lainnya. Model *Deep learning* ini dapat menghasilkan akurat yang tinggi melebihi kinerja manusia. Karena model ini dilatih dengan menggunakan dataset berlabel dengan jumlah yang banyak dan *Neural Network* dapat melakukan penyelesaian masalah secara akurat dan otomatis. Metode *Deep learning* sebagian besarnya menggunakan *Neural Network Architecture*, oleh karena itu *Deep learning* sering disebut dengan *Deep Neural Network*. Dari istilah *"deep"* dapat mengacu pada jumlah lapisan yang tersembunyi di *Neural Network* atau dengan kata lain *hidden layer*. Pada *Neural Network* tradisonal hanya mengandung 2-3

lapisan saja, namun berbeda hal dengan *Deep Neural Network* dapat memiliki 150 lapisan (Yakib, 2020).

Deep learning dapat disebut juga dengan *Representation Leraning*. *Deep learning* melakukan model komputasi yang memiliki bebrapa layer pengolalahan agar mempelajari sebuah representasi data. Metode ini meningkatkan pengembangan pada berbagai bidang yaitu pengenalan suara (*voice recognition*), pengenalan *object* visual (*image recognition*), deteksi objek (*object detection*). *Deep learning* saat ini temasuk masuk kedalam terobosan berbagai bidang pada *artificial intelligence* (Manajang et al., 2020).

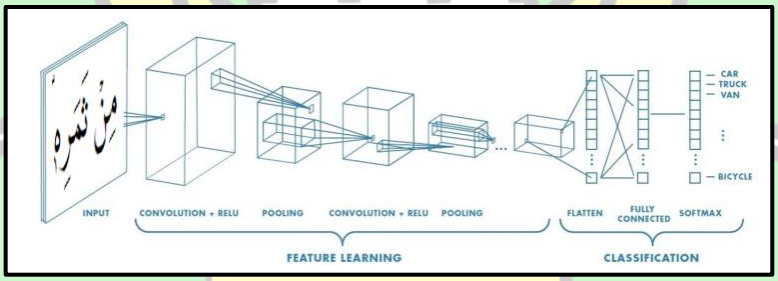
2.12 *Convolutional Neural Network* (CNN)

*Convolutional Neural Network (*CNN) merupakan pengembangan dari *Multilayer Perceptron* (MLP) yang dibuat sebagai memperoses data. CNN merupakan jenis dari *Deep Neural Network* dikarenakan memiliki dasar jaringan yang tinggi hingga sering digunakan pada citra (Pamungkas, 2020). CNN merupakan algoritma yang populer saat ini dalam *Deep learning* termaksuk kedalam bagian *machine learning* yang mana model untuk melakukan klarifikasi pada gambar, tesk, suara dan video. CNN sedikit berbeda dengan *Neural Network*. Biasanya CNN digunakan untuk dapat melakukan mengenali pola pada wajah atau *object* yang lain (Yakib, 2020).

2.12.1 Arsitektur Jaringan CNN

Arsitektur Convolutional Neural Network memiliki standar beberapa *layer konvolusional* (*layer* ReLU) lalu *pooling layer*, lalu beberapa dari layer konvolusi bagian dari yang lain ditambahkan ReLU, kemudian pooling layer yang lain dilakukan proses pengulangan yang sama.

kemudian beberapa dari lapisan *konvolusi* yang lain ditambahkan ReLU), kemudian *pooling layer* yang lain dan bergitu seterusnya. Pengambaran pada gambar semakin kecil dan terus semakin kecil gambar tersebut maka seiring banyak jaringan yang masuk maka semakin banyak layer. Hal tersebut dikerenakan dari lapisan konvolusional. Pada bagian dari CNN, terdapat *feed forward Neural Network regular* dan beberapa *fully connected layer* kemudian ditambahkan ReLUs, lalu pada lapisan akhir menampilkan prediksi (Yakib, 2020).



Gambar 2.4 *Arsitektur Convolutional Neural Network* (Yakib, 2020)

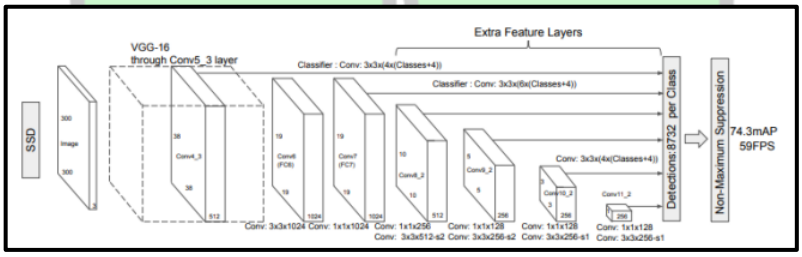
Terdapat beberapa Arsitektur CNN yaitu *AlexNet*, SSD menggunakan *Visual Geometry Group* (VGG) *16, Visual Geometry Group* (VGG) *19, Residual Network* (ResNet) *50, Residual Network (ResNet) 101, GoogleNet, Inception-V3, InceptionResNetV2* dan *Squeezenet* (Suartika E. P, I Wayan, Wijaya Arya Yudhi, 2016).

2.13 *Single Shot Multibox Derector* (SSD)

Proses yang ditempuh oleh metode ini yaitu dilakukannya pencocokan pada sebuah objek dengan *default bounding box* atas *skala* maupun *rasio* pada setiap dari *feature map location.* Saat mendeteksi objek, SSD melakukan perbandingan objek dengan *default bouding box* dengan *rasio* saat proses training. Metode SDD menggunakan ssejumlah *layer* pada berbagai skala yang memeberikan hasil terbaik pada objek yang dilakukan deteksi. Dalam penelitian penulis, arsitektur mobilenet dilakukan sebagai *feature extractor* pada metode SSD (Dompeipen & Najoan, 2021).

Penulis mengutip hasil penelitian dari Liu dkk (2016), SSD adalah salah satu metode yang dapat mendeteksi objek pada *image* dengan memakai *single deep Neural Network.* Dalam proses prediski, setiap kotak ia akan menyesuaikan dari sebuah objek sehingga lebih cocok. Hasil dari pengujian Liu dkk mendapatkan bahwa SSD memiliki keakuratan yang jauh lebih baik dengan berbagai ukuran, bahkan gambar yang *size* lebih kecil (Mashita, 2020).

SSD menerapkan model VGG-16 sebagai jaringan dasar dalam pembuatannya, adapun arsitektur SSD dapat dilihat dalam gambar 3.27 sebagai berikut.



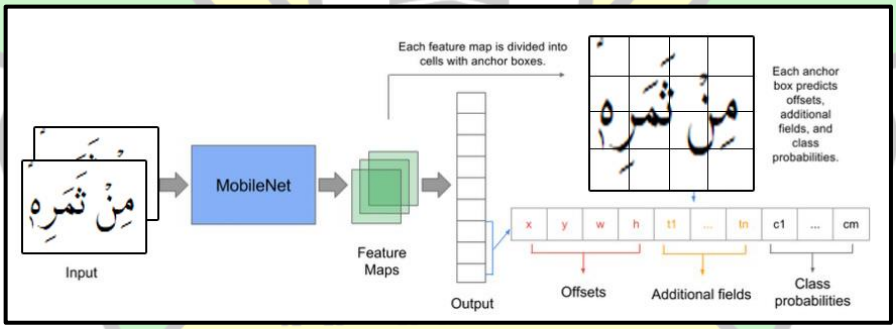
Gambar 2.5 *Arsitektur SSD* (Mashita, 2020)

Kepanjangan dari jaringan VGG-16 yaitu "*Visual Geometery Group*-16 *Network*", VGG-16 ia termaksuk ke dalam arsitekstur jaringan CNN, yang dirancang oleh sebuah grup dalam kompetisi *Image Net Large Scale Visual Recognition Challenge* (ILSVRC) yang terjadi pada tahun 2014 untuk dilakukannya lokalisasi dan klasifikasi pada sebuah *image*. Arsitektur CNN terdiri dari 16 layer yaitu: (Mashita, 2020)

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

2.14 *Mobile Network*

Mobilenet adalah salah satu metode dari bagian arsitektur *Convolutional Neural Network* yang dapat memenuhi keperluan komputasi yang berlebih. *Mobilenet* dibangun oleh para arsitektur *google* atas dasar memenuhi kebutuhan CNN dalam membangun arsitektur yang dapat digunakan pada sistem *mobile*. Perbedaan yang cukup mendasar di antara arsitektur *Mobilenet* dengan CNN adalah saat menggunakan *layer konvolusi* yang ketebalan *filter*nya dibuat sesuai dengan ketebalan input citra *tajwid* yang ada. *Mobilenet* dirancang di atas arsitektur jaringan yang efisien dengan menggunakan konvolusi yang dapat dipisahkan secara mendalam untuk menghasilkan *Deep Neural Network* yang ringan (Sindy, 2019). Pada gambar proses mobile network penulis sudah mengilustrasikan proses pengimputan gambar menggunakan image potongan ayat Al-Quran dengan ukuran 4x4 pixel, seperti yang terlihat pada gambar 2.6.



Gambar 2.6 *Proses Mobile Network*

Perbedaan dasar dari arsitektur mobilenet dan CNN yaitu pada penggunaan lapisan dengan ketebalan filter atau *layer konvolusi* yang disesuaikan pada ketebalan image tersebut (Hendriyana & Maulana, 2020). Adapun dengan memakai arsitektur yang benar maka pelatihan akan semakin cepat. *Mobilenet* salah satu arsitektur yang memiliki *performa* yang optimal. Pada *mobilenet* ini yang menjadi

focus utama yaitu *latensi* yang menghasilkan jaringan yang kecil dan dapat mengoptimalkan kecepatan. (Hendriyana & Maulana, 2020).

Adapun keisitimewaan pada *mobilenet* ini yaitu memiliki daya komputasi yang kecil sehingga dapat menjalankan maupun menerapkan *learning* ini. Oleh karena itu mobilenet ini sangat baik pada perangkat seluler, *embedded system* maupun *computer* tanpa *GPU* atau dapat dikatakan komputasi *computer* dengan kurang efisien hingga mengorbankan hasil akurasi yang kecil. *Mobilenet* menggunakan *layer konvolusi* dengan tebal *filter* yang dikondisikan pada sebuah input citra.

2.15 Tools

Dalam penelitian ini penulis menggunakan tools perangkat lunak berupa Bahasa pemograman python, *TensorFlow Object Detection API, LabelImg, Jupiter Notebook,* CUDA, *CuDNN, Visual Studio Code, Adobe Photoshop* dan *Anaconda. Tools* tersebut yang akan penulis gunakan untuk melakukan proses deteksi objek menggunakan citra penggalan ayat Al-Quran yang terdapat *tajwid nun* mati.

Perangkat lunak tersebut dapat difungsikan dengan mendownload dan melakukan instalasi masing-masing tools tersebut. *Tools* yang digunakan dapat menggunakan versi terbaru agar tidak terjadi kendala yang menghambat proses pendeteksian hukum bacaan *tajwid nun* mati.

2.15.1 *Python*

Python adalah bahasa *pemograman* dengan level tingkat tinggi. *Python* bersifat *open source*, dibuat agar mudah dipahami dan diterapkan. Bahasa *scripts*

python dapat dijalankan pada sistem operasi *Mac*, *Windosws*, *linux* dan yang lainnya. *Python* banyak juga digunakan dalam melakukan pengembangan *web*, pengembangan game, *pemograman* numerik, aplikasi pekantoran dan yang lain (Wantania, 2020).

Python yaitu bahasa *pemograman* yang dinamis yang memiliki manajemen *memory* otomatis dan dapat dipakai dalam berbagai jenis perangkat lunak. *Python* menyediakan dukungan untuk integrasi dengan *pemograman* yang lain. *Python* dikatakan sebagai bahasa *pemograman* yang memiliki sintak kode yang jelas (Nugroho et al., 2020).

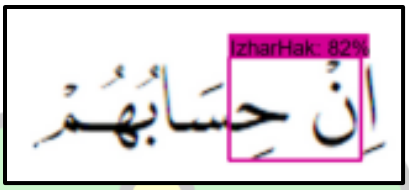
Kekurangan pada *python* yaitu salah satunya programnya harus diproses terlebih dahulu sebelum dijalankan pada komputer. Namun juga terdapat banyak sekali kelebihan pada *pemograman* tingkat tinggi ini yaitu mudah dalam mempelajarinya, mudah ditulis, mudah dalam membaca, mudah dalam mencari kesalahan dan juga yang lain (Rena, 2019).

2.15.2 Definisi *Object Detection*

Merupakan keberadaan suatu *object* serta lokasi pada sebuah gambar. Deteksi *object* dapat dibagi menjadi dua bagian yaitu deteksi lunak dan deteksi keras. deteksi lunak hanya mendeteksi adanya *object* dan deteksi keras mendeteksi adanya *object* dan juga lokasi dari *object* tersebut (Rachardi, 2020).

Menurut Sahasri, M., & Gireesh, C. (2017) yang dikutip dari (Nufus et al., 2021) menyatakan bahwa *Object Dection* merupakan teknik tujuan *computer* dalam menemukan objek pada *image* maupun video. Algoritma *object dection* memanfaatkan *machine learning* dan *deep learning* sehingga menghasilkan

sesuatu yang bermakna. Ketika manusia melihat *image* maupun video, ia dapat menemukan objek untuk beberapa saat. Bedahal nya dengan *computer* yang perluh komputasi yang komplek terlebih dahulu. Tujuan *object dectio*n yaitu untuk menerapkan kecerdasan manusia kepada *computer*.



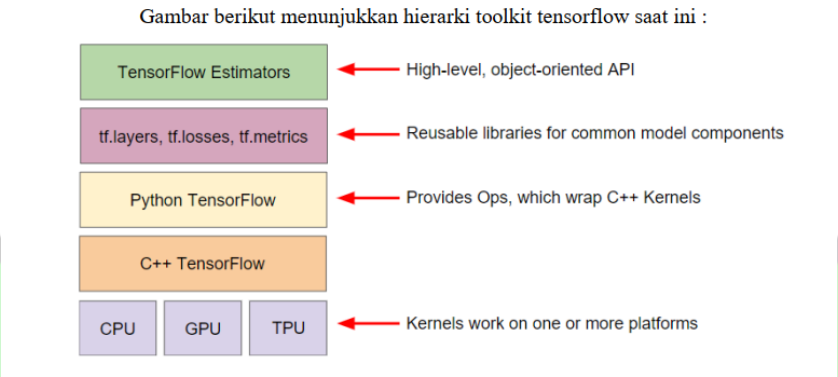
Gambar 2.7 *Deteksi Objek Tajwid*

Adapun cara kerja pada deteksi objek ini yaitu dengan ditandai keberadaan objek pada image dan memberi kotak pembatas pada sekeliling objek tersebut (Nufus et al., 2021). Penulis menggunakan objek berupa citra potongan ayat Al-Quran yang mengandung bacaan tajwid didalamnya. Pada citra tersebut penulis menggambarkan kotak pembatas disekitar objek bacaan yang terdapat *tajwid nun* mati, dapat dilihat pada gambar 2.7.

2.15.3 *TensorFlow Object Detection API*

TensorFlow merupakan sebuah *software library* yang dikembangkan oleh *google*. Tujuannya yaitu untuk melakukan *learning* pada mesin dan penelitian jaringan syaraf (Dewi, 2018). *TensorFlow* adalah salah satu *framework* dari *Deep learning* yang bersifat *free open source*. Dalam *deteksi objek* terdapat *framework TensorFlow object detection* API yaitu suatu alat yang dipakai agar mempermudah proses membangun, melatih, dan menyebarkan pada suatu model *object detection* (Manajang et al., 2020). *TensorFlow* telah banyak dipakai diberbagai produk seperti deteksi wajah, gambar, plat no kendaraan dan yang lainnya (Yakib, 2020).

TensorFlow mendukung berbagai bahasa *pemograman* sehingga dapat mempermudah *developer* dalam proses pembelajaran mesin. *User* dapat membuat model *machine learning* sendiri sesuai dengan kemauan yang dibutuhkan oleh *developer* (Muliadi, 2020). *TensorFlow* juga termaksud kedalam *library* yang popular saat ini dalam bidang data *science* (Rachardi, 2020).



Gambar 2.8 *TensorFlow Toolkit Hierarchy* (Rachardi, 2020)

Menurut (Yakib, 2020) *TensorFlow* memiliki beberapa keunggulan sebagai berikut:

1. Performa

Performa adalah aspek penting dalam mengembangkan sistem *machine learning*. Itu disebabkan *TensorFlow* menggunakan XLA. XLA pengkompilasi aljabar *linear* hingga membuat kode dari *TensorFlow* dapat secepat mungkin pada prosesornya seperti CPU, GPU, TPU.

2. Fleksibel

Framework TensorFlow tersedia API tingkat tinggi yang dapat mempermudah dalam pengembangan, melatih suatu model dan yang lainnya.

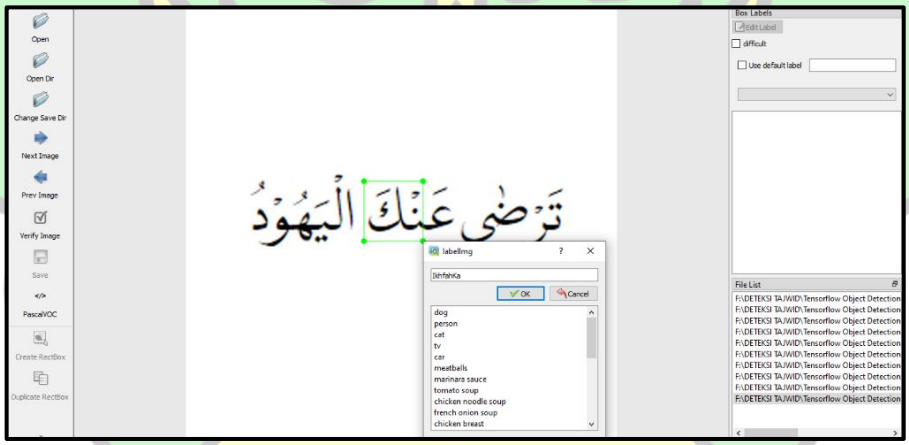
3. Siap Dipakai

Dalam penggunaan *framework TensorFlow* dapat meliputi riset penyelidikan hingga produksi dengan skala yang besar.

2.15.4 *LabelImg*

LabelImg merupakan program yang berfungsi untuk melabelkan sebuah *image.*

Program pelabelan ini dapat digunakan secara gratis dan *open souce. LabelImg* dirancang dengan menggunakan bahasa *pemrograman python* 3 dan Qt untuk *user interface*, oleh karena itu dibutuhkan *instalasi python 3* dan *library* PyQt di komputer untuk menjalankannya. *LabelImg* ditunjukan pada Gambar 2.5 berikut: (Budiarjo, 2020)



Gambar 2.9 *Labelimg*

Setelah dataset telah di label maka output dari labelimg yaitu data yang berformat XML. Pelabelan dataset secara manual di proses sesuai dengan jumlah data yang akan penulis teliti. Pelabelan data penting dilakukan untuk proses awal penanda objek mana yang akan di fokuskan atau dikenalkan oleh system.

2.15.2 *Jupiter Notebook*

Dokumen *notebook* menampilkan isi dokumen seperti berisi elemen teks, *image*, tautan dan persamaan. Campuran dari elemen *code* dan teks pada dokumen ini adalah tempat yang cocok untuk menyatukan antara deskripsi, analisis dan hasilnya, sehingga dapat dieksekusi dengan dilakukannya analisis data secara *real time*. *Jupiter Notebook* adalah aplikasi edit teks untuk *machine learning* dan yang sifatnya *client-server*. Aplikasi ini dapat menjalankan bahasa pemograman *python* (Mujilahwati et al., 2021).

Aplikasi ini dengan file yang berextensi ipynb tersebut ialah sebuah dokumen yang di hasilakn dari *Jupyter Notebook App* yang berisi sebuah code computer dan *rich text element* seperti paragraf, *image*, *links* dan persamaan tematik. Aplikasi ini sering dikenal dengan *python notebook* dan kana berkembang menjadi *jupyter lab*. (Setiabudidya, 2017).

2.15.5 CUDA

CUDA (*Compute Unified Device Architecture*) di kembangkan oleh *Nvidia* yang mana berfungsi untuk mempermudah dalam *utilitasi* GPU. Pada arsitektur CUDA ini digunakan oleh pengembang *software* agar mampu berjalan pada GPU buatan dari Nvidia. Nvidia menggunakan arsitektur CUDA ini untuk diterapkan pada sebuah GPU. (Budiarjo, 2020).

Perkembangan CUDA ini sangat pesat dan maju dalam teknologi. Dikarenakan banyak penelitian yang menggunakan CUDA hingga mendapati CUDA SDK banyak yang di *download*. (Budiarjo, 2020). CUDA ini sering difungsikan untuk pemograman grafis seperti pada gambar dan video precessing

secara digital seperti hal nya , perubahan kualitas citra, pengenalan pola dan juga *image* segmentation (Kurniawan, 2015).

2.15.6 *CuDNN*

CuDNN dan CUDA merupakan library yang dibesarkan oleh Nvidia diperuntukkan pada *deep neural network.* CuDNN bermanfaat dalam meningkatkan *performa GPU* pada *deep Neural Network* maupun pada jaringan saraf tiruan. Library CuDNN ini pula banyak difungsikan pada aplikasi untuk deep learning seperti *TensorFlow* , *Keras, Matlab, MxNet* dan *PyTorch*. Penulis menggunakan CuDNN sebagai library *system* pendeteksian.(Budiarjo, 2020).

2.15.7 *Visual Studio Code*

Visual Studio Code (*VS Code*) difungsikan untuk *operating system multiplatform* sebagai teks editor ringan dan handal yang di bangun oleh *Microsoft. VS Code* ini juga mendukung pada *system* operasi *linux* dan *mac*. Bahasa pemograman yang mendukung pada *VS Code* ini seperti *JavaScript, Typescript,* dan *Node.j*. dapat juga menggunakan bahasa pemograman yang lain dengan bantuan *plugin* yang dapat di pasang via *marketplace VS Code (seperti C++, C#, Python, Go, Java*, dst) (A. Yudi Permana, 2019).

Adapun fitur yang tersedia pada *VS Code yaitu Intellisense, Git Integration, Debugging*, dan juga fitur menambah kemampuan teks *editor*. Dengan bertambahnya *versi* pada *VS Code* ini maka bertambah pula lah *fitur-fitur* terbaru. Adapun yang membedakan *VS Code* dengan teks *editor* yang lain salah satunya pembaruan *versi* yang dilakukan setiap bulan sekali.(A. Yudi Permana, 2019).

2.15.8 *Adobe Photoshop*

Penulis menggunakan *Adobe Photoshop* versi CS6. *Adobe Photoshop* adalah perangkat lunak yang digunakann dalam mengedit atau sering dilakukan manipulasi pada sebuah *image*. Banyak sekali para *design* menggunakan aplikasi ini karna berbagai alasan salah satunya memiliki fitur yang lengkat dan mencakup *tools-tools* yang digunakan untuk mengedit *image, filter, efek,* manipulasi warna dan yang lainnya. (Kiki Firmantoro, Anton, 2016).

Adobe Photoshop ini sering digunakan sebagai software dalam pengolahan image. *Adobe Photoshop* telah memberikan kemudahan untuk mendapatkan hasil yang berkualitas hingga memuncak pada kesempurnaan (Kiki Firmantoro, Anton, 2016). Penulis menggunakan *Adobe Photoshop* untuk proses *convert size* dataset. Dataset yang telah di convert digunakan untuk mempermudah pengenalan pola pada *system.*

2.15.9 *Anaconda*

Anaconda sebuah *tools* yang di bangun untuk *data scientist. Anaconda* ini terdapat didalamnya sebuah *package* dan juga *environment manager* yang diperuntukkan pada *python*. Didalam *anaconda* terdapat *library* untuk *tensorflow*, *library* ini dapat diintsall dan digunakan.

Tools ini merupakan *platform* yang terbuka atau *open source* yang memiliki penggunaan bahasa pemograman *open source* seperti *R* dan *Python* yang dapat diproses dengan data berskala besar *predictive analytics,* dan *scientific computing. Anaconda* dapat menganalisis masalah sains data (Bagas, 2021)

2.16 Model Evaluasi

Model evaluasi yang digunakan oleh penulis yaitu *Confusion Matrix*. Evaluasi merupakan kesimpulan akhir dari proses yang telah dilakukan atau *output* yang dikeluarkan dari proses panjang yang telah dilakukan, maka metode yang digunakan yaitu *Confusion Matrix* untuk mengetahui keakuratan proses pendeteksian *tajwid nun* mati.

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

2.16.1 Confusion Matrix

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

Keterangan:

1. *True Positive (TP)* yaitu dimana *objek* tajwid telah berasil dideteksi oleh model *system*
2. *False Positive (FP)* yaitu data bukan objek *tajwid* namun dideteksi *tajwid* oleh *system*
3. *False Negative (FN)* yaitu data objek *tajwid* namun dideteksi bukan objek tajwid oleh *system*.

4. *True Negative (TN)* yaitu *system* tidak dapat mendeteksi objek *tajwid nun mati.*

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

1. *Accuracy,* merupakan hasil perhitungan tingkat keakuratan deteksi objek terhadap objek *tajwid nun mati* secara keseluruhan. Adapun persamaan *accuracy* dapat dilihat dipersamaan 2.1 (Manajang et al., 2020).

$$Accuracy = \frac{Jumlah\ Objek\ Yang\ Terdeteksi\ Benar}{Jumlah\ Keseluruhan\ Objek\ Yang\ Terdeteksi} \text{ x } 100\% \quad \text{Persamaan (2.1)}$$

2. *Precission,* merupakan persamaan mengenai jumlah prediksi yang benar dibandingkan dari keseluruhan hasil yang dapat di prediksi oleh system. Pada persamaan ini akan menghasilkan jumlah objek yang benar dari keseluruhan yang di deteksi oleh system. Adapun persamaan *precission* dapat dilihat dipersamaan 2.2 (Manajang et al., 2020).

$$Precision = \frac{TP}{(TP + FP)} \text{ x } 100\% \quad \text{Persamaan (2.2)}$$

3. *Recall,* merupakan jumlah prediksi yang benar dibandingkan dengaan keseluruhan hasil pendenteksian tajwid sebenarnya. Persamaan ini akan menghasilkan jumlah objek deteksi dengan benar dari keseluruhan jumlah objek yang terdeteksi oleh system. Adapun persamaan *recall* dapat dilihat dipersamaan 2.3 (Manajang et al., 2020).

$$Recall = \frac{TP}{(TP + FN)} \text{ x } 100\% \quad \text{Persamaan (2.3)}$$

BAB III
METODE PENELITIAN

Pada proses pengujian, penulis membuat kerangka kerja yang berbentuk skema untuk memudahkan penulis dalam melakukan penelitian sehingga skema tersebut penulis jadikan sebagai panduan dalam melakukan tahap-tahap pengujian. Adapun tahap-tahap yang disusun sebagai berikut:

3.1 Metode Pengumpulan Data

3.1.1 Studi Pustaka

Studi pustaka menjadi salah satu metode yang penulis gunakan untuk proses pengumpulan data yaitu dengan membaca, mengolah informasi, menulis catatan penting bersumber dari buku-buku pustaka, mengutip bahan-bahan yang mendukung dan berhubungan dengan pengujian penulis. Adapun referensi yang penulis kumpulkan berupa data dan informasi yang terdapat di berbagai buku, beberapa situs *online* terpecaya dan jurnal yang saling keterkaitan dengan penelitian dan pengujian *system*. Referensi tersebut penulis gunakan untuk menyelesaikan penulisan bab pendahuluan, landasan teori dan metode penelitian.

3.1.2 Observasi

Proses observasi yang penulis lakukan adalah mengumpulkan data gambar *tajwid nun mati* di *website* Kementerian Agama https://quran.kemenag.go.id/. Tahapan yang dilakukan adalah pencarian gambar menyeluruh pada surah Al-Quran. Kemudian gambar yang terdapat *tajwid nun mati* tersebut penulis ambil sebagian dan dikelompokkan sesuai dengan nama *tajwid* dan hurufnya yang sudah

ditetapkan. Data-data tersebut penulis kumpulkan satu persatu yang nantinya akan menjadi dataset untuk data *training* dan data *testing* pada pengujian sistem.

3.2 Metode Simulasi

Penulis menggunakan metode simulasi dalam penelitian ini, adapun tahap-tahap metode simulasi ini sebagai berikut:

3.2.1 Menemukan Formulasi Permasalahan

Identifikasi masalah yang dilakukan oleh penulis berdasarkan penelitian-penelitian yang berhubungan dengan metode *Convolutional Neural Network* dengan menggunakan training model SSD *Mobilenet*. Pada penelitian terdahulu belum pernah ada yang membahas pengujian sistem mengenai deteksi *tajwid nun mati* pada ayat Al-Quran dengan metode *Convolutional Neural Network* menggunakan model training SSD *Mobilenet.*

3.2.2 Konsep Penelitian

Penelitian ini menggunakan model untuk mendeteksi *tajwid nun mati* pada ayat Al-Quran dengan metode *Convolutional Neural Network* dengan model training SSD *Mobilenet.* Pada proses pengujian sistem ini nantinya terdapat proses *training* pada dataset gambar, membuat model pengujian untuk mengenali *tajwid nun mati* dan mendapatkan nilai akurasi tertinggi. Setelah itu hasil *output* yang didapat dari hasil pengujian *system* berupa *image* yang telah di *input* dan dikenali oleh sistem.

3.2.3 Pengumpulan Data *Input* Dan Data *Output*

Berikut tahap pengumpulan data *input* yang digunakan untuk pengujian sistem deteksi *tajwid nun mati* ini. Data yang dikumpulkan berupa *image* yang terdapat *tajwid nun mati* berdasarkan masing-masing huruf pada setiap *tajwid nun mati*. *Tajwid nun mati* terbagi menjadi 4 bagian yaitu *izhar, iqlab, ikhfah, idgham (idgham bilaghunnah* dan *idgham bighunnah*. Data tersebut penulis kumpulkan dari sumber internet di *website* Kementerian Agama *https://quran.kemenag.go.id/.* Proses mengambilan sebagian ayat pada suatu surah di dalam Al-Quran dari *website* tersebut menggunakan teknik *Snipping Tools*.

Data awal berupa surah Al-Quran yang masih utuh, kemudian penulis ambil sebagian menjadi penggalan kata yang terdapat *tajwid nun mati* sesuai dengan huruf masing-masing. Setelah seluruh data gambar berhasil dikumpulkan, data gambar tersebut nantinya akan dibagi menjadi data *train* dan data *test*. Dataset berjumlah 350 gambar penggalan ayat yang terdapat hukum bacaan *tajwid nun mati* dengan metode *random sampling*. Setelah data tersebut dikumpulkan maka akan dilakukan pelabelan *image* agar dapat dilakukan proses *training* pada dataset tersebut.

3.2.4 *Preprocesing*

Setelah pengumpulan data selesai, dilanjutkan pada tahap berikutnya yaitu tahap *preprocessing*. Data *image* akan di *resize* menggunakan aplikasi *photoshop*, untuk menyamakan ukuran dari *image* dengan ukuran 1080x1080 *pixel*. Data yang telah di *resize* kemudian dilakukan proses *labelimg*, konversi XML ke TFRecord dan konfigurasi *pipeline* yang bertujuan agar data gambar dapat dimasukan dalam proses *training* sehingga *system* dapat mengenali objek yang ada pada gambar.

3.2.5 Pemodelan Algoritma

Tahap selanjutnya yaitu melakukan pembuatan pengujian model dari dataset *tajwid* untuk mengenali pola gambar *tajwid nun mati* dan proses klasifikasi menggunakan metode *Convolutional Neural Network* dengan model training *SSD Mobilenet* agar sistem dapat mendeteksi *tajwid nun mati* beserta huruf pada setiap *tajwid.*

3.2.6 Training Dan Pengujian

Penulis menggunakan teknik training dan pengujian menentukan metode *Convolutional Neural Network* dengan model training *SSD Mobilenet*. Data *image* disatukan dalam bentuk citra, citra tersebut dibagi menjadi data *image train* dan data *image test*. Untuk mendapatkan model yang ingin dibuat dibutuhkan suatu analisis dari data *train*. Setelah data di training maka data *train* diuji coba ulang terlebih dahulu untuk melihat apakah *system* mampu mendeteksi *tajwid*. Data *test* berguna untuk mengetahui nilai keakuratan dari model yang telah ditraining dengan menggunakan *num-step* 50.000 hingga memiliki estimasi waktu *training* sebesar 48 jam.

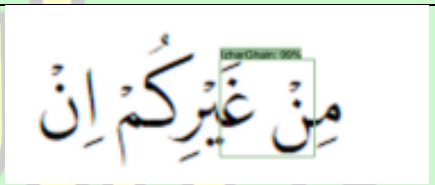
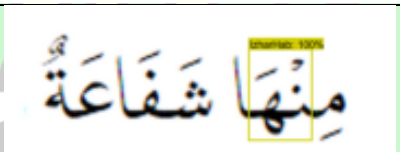
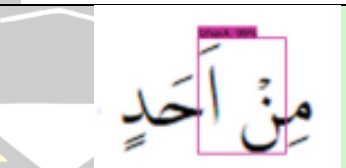
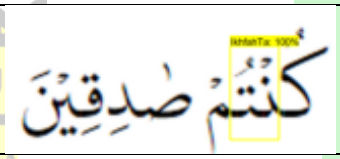
3.2.7 Menghubungkan ke API

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

3.2.8 Percobaan Keberhasilan

Kemudian tahap berikutnya yang penulis lakukan yaitu tahap pengujian. Tahap ini mencoba keberhasilan model yang dibangun. Penulis menginput *image tajwid nun mati* berektensi *(.jpg)* yang telah penulis siapkan, lalu sistem akan memproses *input* tersebut. Setelah diproses melalui model yang dibuat, keluarlah hasil dari *input* tersebut berupa *image* yang telah dideteksi beserta akurasi pendeteksian *tajwid nun mati* dengan setiap hurufnya. Terlihat seperti table 3.1.

Tabel 3.1 Keberhasilan Pendetesksian *Tajwid*

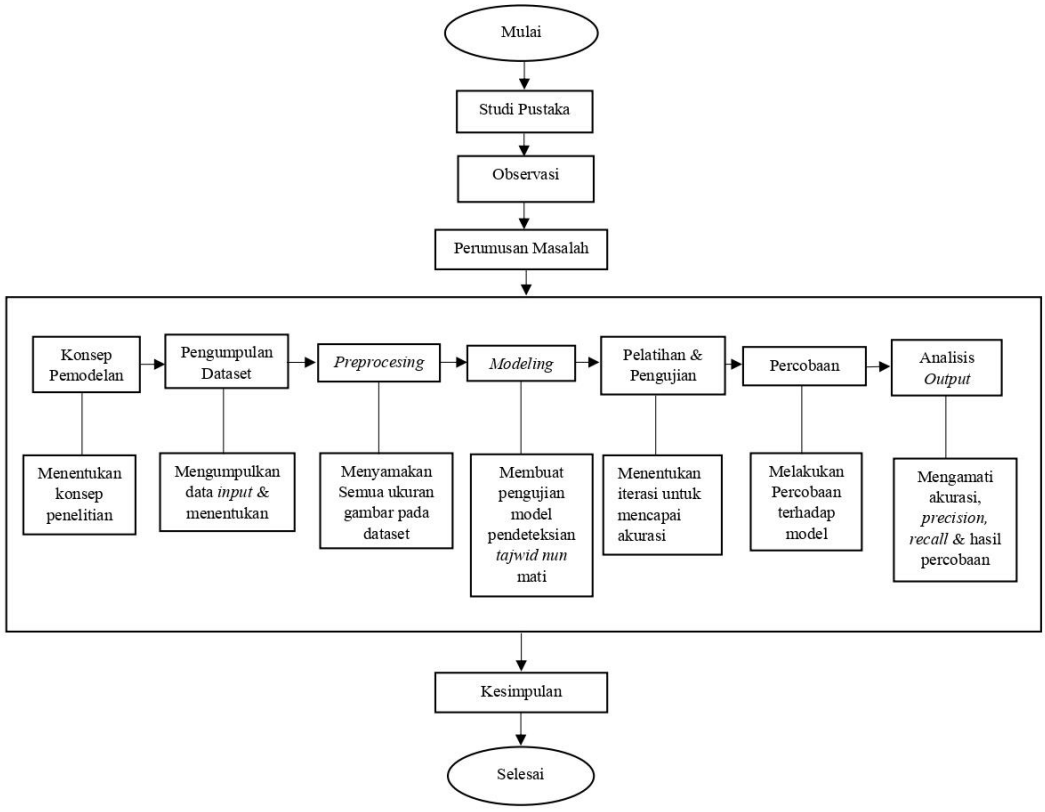
Tajwid Izhar ن berjumpa غ	
Tajwid Izhar ن berjumpa ه	
Tajwid Izhar ن berjumpa ء	
Tajwid Ikhfah ن berjumpa ت	
Tajwid Ikhfah ن berjumpa ث	

Sumber: penulis

3.2.9 Analisis *Output*

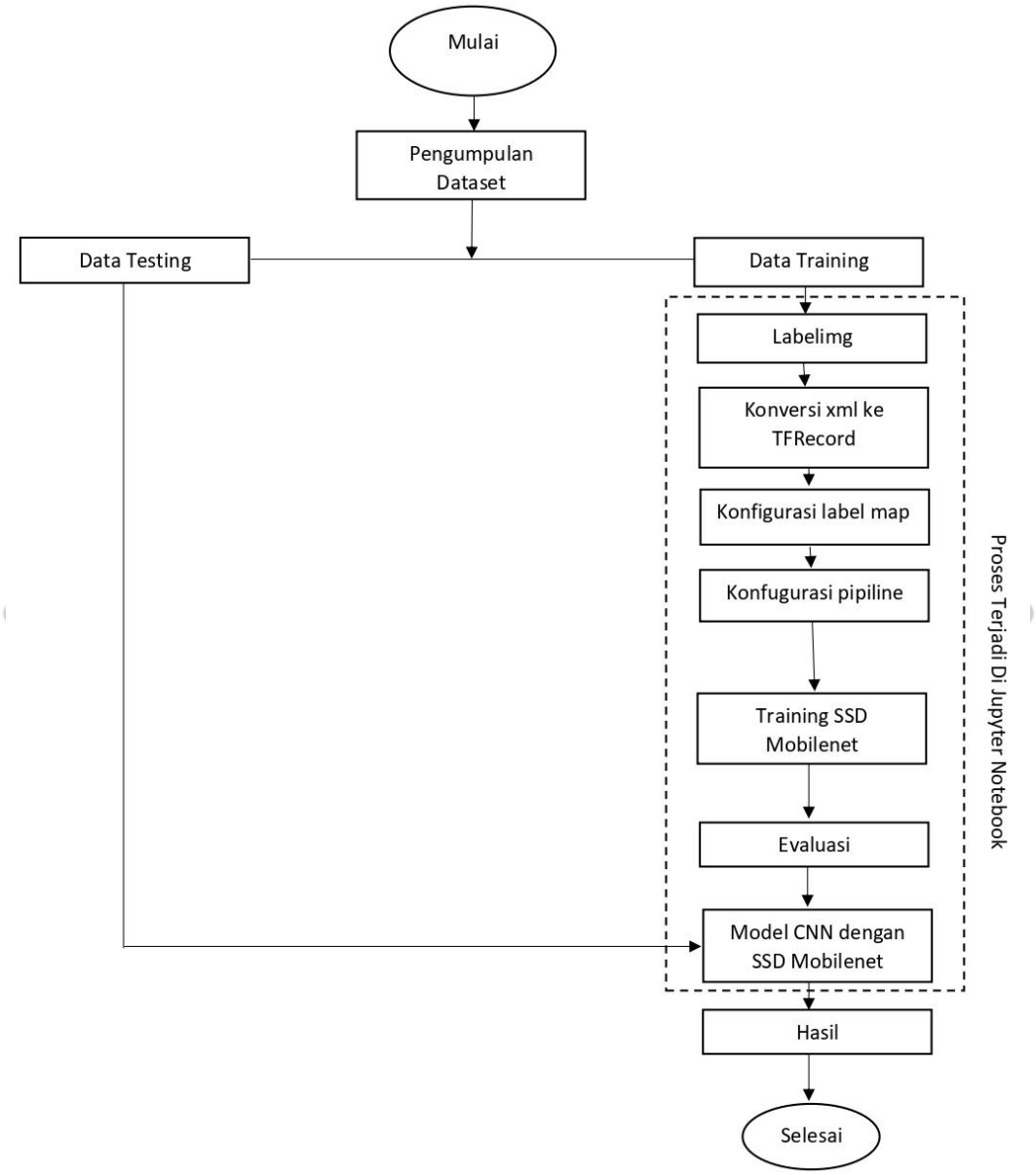
Tahap akhir yang penulis lakukan yaitu analisa terhadap *output* berdasarkan asumsi yang telah dilakukan apakah sesuai dengan *input*. Lalu, tahap ini akan ditampilkan hasil *output* dengan *input image.*

3.3 Kerangka Pemikiran



Gambar 3.1 Kerangka Pemikiran

3.4 Tahapan Penelitian



Gambar 3.2 Tahapan Penelitian

Berikut ini adalah penjelasan pada gambar tahapan penelitian:

a) Pengumpulan Dataset

Data yang digunakan dalam penelitian ini adalah gambar hukum bacaan *nun mati* yang terdapat dalam Al-Quran. *Nun mati* terbagi menjadi 4 bagian yaitu

meliputi *izhar, iqlab, ikhfah, idgham (idgham bilaghunnah dan idgham bighunnah)*. Pengambilan dataset berada diwebsite kemenag https://quran.kemenag.go.id/. Proses pengambilan data menggunakan dengan Teknik *Snipping Tools*. Dataset tersebut berupa penggalan ayat yang terdapat hukum bacaan nun mati dan multiple *tajwid* pada satu kalimat tersebut, maka dataset berjumlah 350 gambar penggalan ayat yang terdapat hukum bacaan *nun mati* dengan metode *random sampling*. Setelah data tersebut dikumpulakan maka akan dilakukan *pelabelan image* agar dapat dilakukan proses training pada dataset tersebut.

b) Data Training Dan Testing

Dataset yang berjumlah 350 telah dikumpulkan di bagi menjadi dua, yaitu data training sebesar 80% dan data testing 20% masing masing berjumlah 280 dan 70 data gambar. Data training dilakukan proses *procesing image* agar dapat digunakan untuk proses training.

c) Labelimg

Langkah selanjutnya setelah semua data terkumpul adalah pelabelan data gambar. Proses ini bertujuan untuk memberi label pada objek yang akan dideteksi secara satu per satu dengan nama yang sama. Pada penelitian ini nama atau label yang diberikan pada gambar objek seperti ikhfa, iqlab izhar, iqlab dan idgham (idgham bila ghunnah dan idgham bighunnah) disesuaikan dengan huruf masing masing *tajwid* tersebut. Pemberian label pada gambar menggunakan aplikasi LabelImg yang dilakukan dalam jupyter notebook. Setelah pelabelan, file yang disimpan akan tersimpan dalam format file ".xml"

d) Konversi File XML Ke TFRecord

Setelah *pelabelan image* selesai maka hasil file tersebut menjadi format xml. File xml dikonversikan dalam format TFRecord atau *TensorFlow* Record yang merupakan format penyimpanan dari *TensorFlow*. File TFRecord tersebut digunakan dalam proses membaca data input sehingga informasi dataset dapat diambil secara langsung.

e) Konfigurasi Label Map

Langkah selanjutnya adalah proses membuat label map. Label map dibuat untuk mendefinisikan numerikal kategori *tajwid* dalam kalkulasi. Pada penelitian ini menggunakan dua puluh delapan objek yang akan dideteksi sehingga label map yang akan dibuat berjumlah dua item yang mana terdiri dari id dan nama (name) yang harus sesuai dengan urutan dan nama saat dilakukannya proses labeling.

f) Konfigurasi Pipeline

Kemudian tahapan selanjutnya adalah konfigurasi pipeline hal ini dilakukan karena *TensorFlow* menggunakan ProtoBuf sehingga perlu dilakukannya konfigurasi pipeline yang berguna untuk mengkonfigurasi proses training dan evaluasi. Pada penelitian ini menggunakan konfigurasi model SSD *Mobilenet* V2.

g) Training SSD *Mobilenet*

Training dan testing menggunakan arsitektur CNN SSD Mobilenet. Pendekatan SSD berdasarkan pada feed-forward convolutional *Network* yang menghasilkan bounding box berukuran tetap dan nilai skor untuk masing-masing kelas objek dalam area bounding box tersebut.

h) Evaluasi

Evaluasi model train loss adalah mengevaluasi jumlah loss saat terjadinya proses training model dalam penelitian ini.

i) Deteksi Data Testing Dan Training

Data training maupun data testing di lakukan uji coba deteksi, dan data testing yang tidak dilakukan processing image juga dilakukan uji coba deteksi pada framework tensorflow dengan menggunakan metode SSD *Mobilenet*.

j) Hasil Deteksi

Model yang sudah melalui proses training akan siap digunakan untuk pendeteksian atau uji. Hasil deteksi berupa *tajwid* yang telah di deteksi telah terdapat *bouding box* yang sesuai dengan nama *tajwid* dan hurufnya.

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1 Dataset

Dataset atau himpunan data adalah kumpulan objek dari atributnya. Beberapa nama lain yang digunakan seperti *record, point, vector, pattern, case, sample, entitas.* Pengambaran objek dengan sejumlah atribut yang menjelaskan sifat atau karakteristik dari objek. Nama lain yang sering disebut seperti *variable, field, fitur,* atau *dimensi.* Atribut nilai nya bermacam-macam dari satu objek dengan objek yang lain (Radliya, 2015).

Data adalah fakta atau angka atau dapat dikatakan data mentah yang berhubungan dengan konteks permasalahan. Dataset penelitian ini berupa penggalan ayat yang terdapat hukum bacaan *nun* mati dan *multiple tajwid* pada satu kalimat tersebut, maka dataset berjumlah 350 gambar penggalan ayat yang terdapat hukum bacaan *nun mati* dengan metode *random sampling*.

4.1.1 Pengumpulan Dataset

Adapun dataset yang diterapkan dalam penelitian ini yang menggunakan metode SSD *Mobilenet* yaitu data *image*. Proses pembuatan dan pengumpulan pada dataset dilakukan dengan mengambil gambar *tajwid* nun mati meliputi *izhar, ikhfah, iqlab, idgham (idgham bighunnah dan idgham bilahgunnah)* beserta hurufnya menggunakan teknik *snipping tools*. Data yang dikumpulkan diambil dari *website* Kementrian Agama https://quran.kemenag.go.id/. Seluruh data *image* yang

telah di ambil berjumlah 350 data gambar. Gambar 4.2 di bawah ini merupakan dataset *tajwid nun* mati.

Gambar 4.1 Kumpulan Dataset

Kemudian setelah selesai pengumpulan dataset, selanjutnya seluruh data dibagi menjadi dua, yaitu data test dan data train. Agar pengujian deteksi nya baik dan memiliki akuratan yang tinggi dibutukan data train yang banyak untuk dilakukan proses training. agar system dapat mengenali dan mempelajari pola pada gambar tersebut. kedua dataset dibagi menjadi data testing 20% dan data training 80%. Pembagian kedua data dapat dilihat di tabel 4.1 dan tabel 4.2

Tabel 4.1 Data *Training*

Label	Jumlah Data
Ikhfah	150
Iqlab	10
Izhar	60
Idgham	60
Total	280 (80% data train)

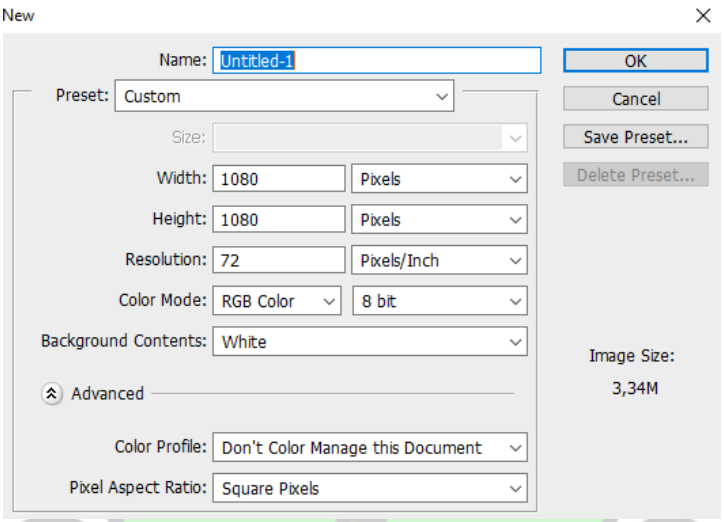
Tabel 4.2 Data *Testing*

Label	Jumlah Data single	Jumlah Data Multyple	Total
Ikhfah	23	7	30
Iqlab	5	3	8
Izhar	11	5	16
Idgham	11	5	16
Total	70 (20% data test)		

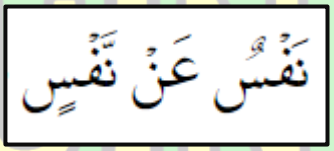
Tabel 4.1 pada tabel di atas merupakan data *training*, pada penelitian ini menggunakan 4 label *izhar, iqlab, ikhfah dan idgham* bergitu juga dengan tabel 4.2. Total data *training* yang berupa data gambar *ikhfah* berjumlah 150 gambar, *iqlab* berjumlah 10 gambar, *izhar* berjumlah 60 gambar dan *idgham* berjumlah 60 gambar dengan jumlah total data 280 gambar. Data testing dibagi menjadi dua, yaitu data *single* dan data *multyple*. Total data yang berupa data *ikhfah* berjumlah 30 gambar, *iqlab* berjumlah 8 gambar, *izhar* berjumlah 16 gambar dan *idgham* berjumlah 16 gambar dengan jumlah total data 70 gambar. Maka total data yang digunakan berjumlah 350 data gambar.

4.1.2 *Preprocessing Image*

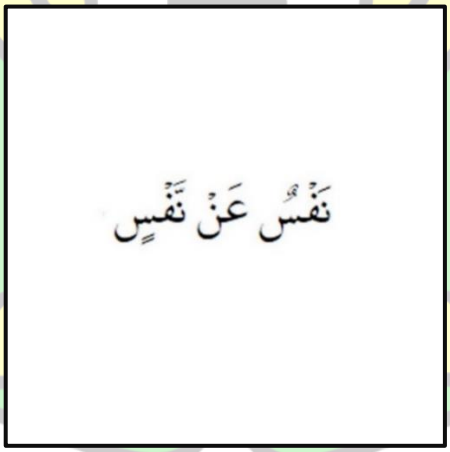
Setelah menyelesaikan tahapan pengumpulan dataset di *website* https://quran.kemenag.go.id/ yang berupa dataset gambar *tajwid nun* mati, maka selanjutnya akan dilakukan tahap *preprocessing image*. Pada proses *preprocessing image* hanya di lakukan pada data *train* saja. Tahapan *preprocessing image* dengan melakukan *convert size* gambar menggunakan aplikasi *Photoshop* dengan ukuran 1080x1080 *pixels*, fungsi *convert size* yaitu untuk memudahkan proses training dan mendeteksi gambar.



Gambar 4.2 *Setting Size Photoshop*



Gambar 4.4
Gambar 4.3 Sebelum *Diconvert Size 116 x 74*

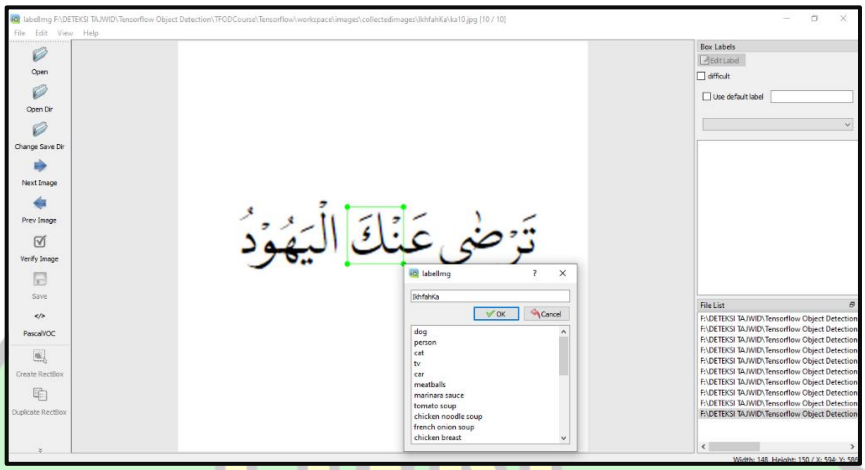


Gambar 4.4 Sesudah *Diconvert Size 1080x1080*

4.2 Pelabelan Citra

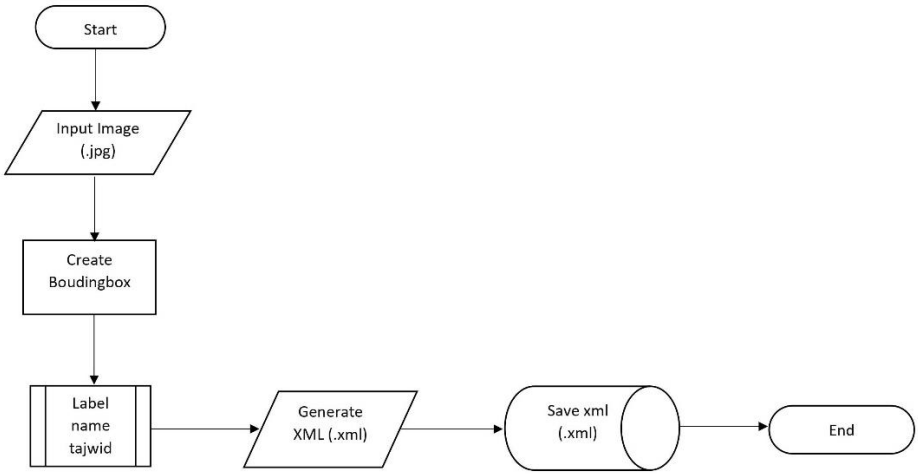
Pada tahap ini dilakukan dengan memberikan pelabelan pada sebuah citra untuk mendapatkan objek khusus yang mampu di baca oleh *system*. Pemberian

label pada citra dilakukan dengan menggunakan aplikasi *Labelimg*, yang berfungsi memberikan label pada citra *tajwid nun* mati dengan berdasarkan pembagian dan kategori yang telah ditentukan, dapat dilihat pada gambar 4.6.



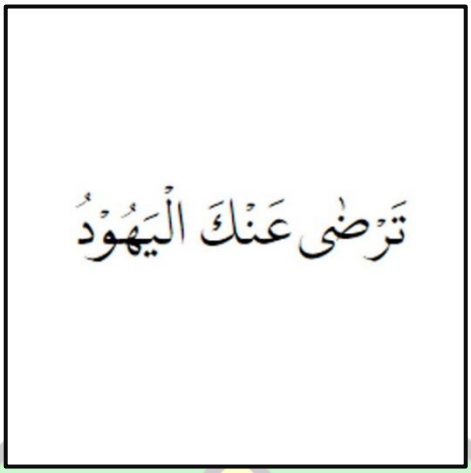
Gambar 4.5 Label citra berdasarkan kategori

Output yang di hasilkan berupa citra *tajwid* berserta nama *tajwid* nya yang sudah di label akan tersimpan dalam format (*.xml)*. Proses dalam pelabelan akan dilakukan secara manual untuk 280 data *training* citra *tajwid nun* mati menggunakan *labelImg.* Tujuan dari pelabelan image yaitu untuk mendapatlan sebuah karakter khusus yang tedapat pada masing-masing image tersebut. karekter image yang kita label tersebut akan dijadikan leraning dalam proses training. Adapun tahapan dalam melakukan *labelimg* dapat dilihat pada gambar 4.7.



Gambar 4.6 *Flowchart Image Labelimg*

Pada gambar 4.7 Proses pelabelan *image* dilakukan di aplikasi *labelimg* didalam *jupyter notebook* yang merupakan aplikasi *website*. Untuk membuka aplikasi *labelimg* tersebut dibutuhkan *syntax (!cd {LABELIMG_PATH} && python labelImg.py) syntax* tersebut di *copy paste* pada aplikasi website *jupyter notebook*. Setelah terbuka aplikasi *labelimg* kemudian *input image tajwid* agar dapat di lakukan proses membuat *boudingbox* dan memberikan *label name* pada *tajwid* agar *image* yang telah di *boudingbox* dapat di kenali pola oleh *system*. Setelah proses *label name* kemudian menghasilkan *output* dengan *format* XML (.xml) lalu *save* pada *database* kemudian proses pelabeling *image* telah selesai. Berikut gambar 4.8 sebelum dilakukan proses *label image* pada aplikasi *labelimg*.

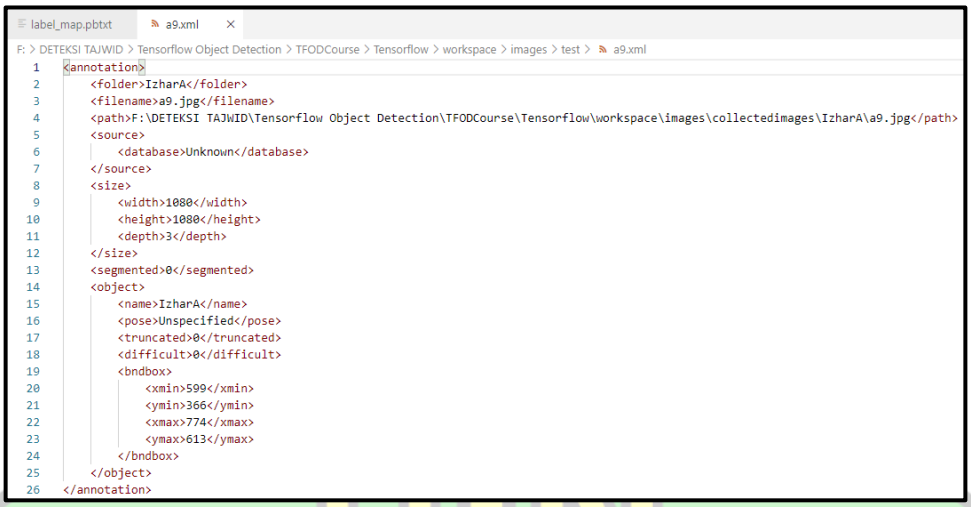


Gambar 4.7 Sebelum *Pelabelan Image*

Gambar 4.8 Sesudah *Pelabelan Image*

Pada gambar 4.9 merupakan proses yang terjadi untuk melakukan proses pelabelan yaitu dengan pemberian *rectbox* yang berguna untuk menandai citra *tajwid* yang sudah di *import* ke dalam aplikasi *labelimg*. Selanjutnya, citra yang objeknya sudah ditandai menggunakan *rectbox* akan diklasifikasi berdasarkan kategori nama *tajwid* yang ada. Pada contoh gambar adalah nama *tajwid ikhfah* dengan huruf *nun* mati berjumpa dengan *kaf* (ك). Selanjutnya, hasil citra yang sudah diklasifikasi akan ditandai dengan nama kategorinya di kolom sebelah kanan. Data akan di *export* dalam bentuk (*.xml)* yang sudah menyimpan informasi jenis

klasifikasi nama *tajwid*. Berikut hasil dari proses pelabelan *image* dari gambar menjadi dokumen dengan *format* (*.xml)* yang dapat di lihat pada gambar 4.10.



Gambar 4.9 Hasil Dari *Labelimg*

4.3 Pengolahan *Image* didalam *TensorFlow*

Berikut adalah struktur *file* yang akan dimasukkan kedalam *TensorFlow* dengan menggunakan *python*.

4.3.1 *Annotations*

Annotations adalah nama *folder* yang didalamnya terdapat sub *folder* yang *file* nya *berextention.xml*. File ini merupakan hasil dari *convert image* yang sudah diberi label. Untuk setiap data *image* dilakukan pelabelan, dengan ditandai *bounding box* dan label pada objek image. Dilakukan hal tersebut untuk *system* agar mengenali pola *tajwid* pada Al-Quran sehingga mampu mendeteksi *tajwid* pada *image*. Proses pemberian label menggunakan *labelImg* menghasilkan *file* dengan *format (.xml)*. kemudian file *(.xml)* dan *image* dilakukan konversikan ke file

TFrecord dengan format *(.record).* kemudian file dengan format *(.record*) dimasukan pada melatihan model objek deteksi.

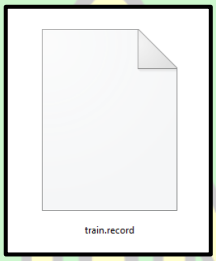
4.3.2 *Convert TFrecord*

Tahapan selanjutnya adalah *convert TFRecord*. Pada tahap ini, data pelabelan berupa (*.xml)* selanjutnya akan diubah ke dalam bentuk file *TensorFlow Record* (*TFRecord*). Hal ini dilakukan akibat data yang ada cukup banyak dan mengakibatkan ukuran yang cukup besar. Melalui *TFRecord* data yang ada akan disimpan ke dalam bentuk *string biner*, sehingga akan berdampak terhadap kecepatan proses training yang akan dilakukan selanjutnya. Data *xml* dan citra berupa nama *tajwid* akan di *convert* menjadi file *TFRecord*. Data yang sudah di *convert* menjadi file *TFRecord* ini akan menggunakan lebih sedikit ruang pada *disk* sehingga proses yang akan dilakukan sistem akan membutuhkan waktu yang lebih sedikit pula. Berikut merupakan *script convert* file ke *TFRecord* pada gambar 4.11. Adapun sebelum *convert TFRecord* dapat dilihat di gambar 4.12 dan hasil dari *convert TFRecord* dapat dilihat di gambar 4.13.

```
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'train')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'], 'train.record')}
```

Gambar 4.10 *Script Convert File Ke TFRecord*

Gambar 4.11 Sebelum *Convert TFRecord*

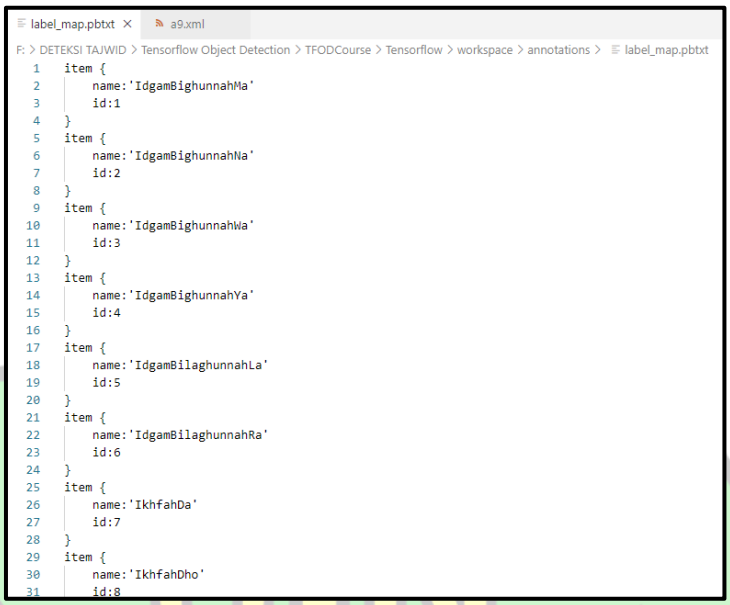


Gambar 4.12 Setelah *Convert TFRecord*

4.3.3 *Konfigurasi Label Map*

Pada tahap ini dilakukan proses pemetaan dengan diberi nama objek yang akan dideteksi. Pada penelitian ini terdapat dua delapan objek hingga label map juga terdapat dua puluh depalan item. Pada gambar 4.14 merupakan konfigurasi label map berisi name dan id sesuai dengan urutan pada saat dilakukan nya proses labeling. 28 item tersebut dapat dilihat pada tabel 4.3. *Label Map* tersebut disimpan

pada berkas dengan *format (.pbtxt)* yang nantinya akan dibutuhkan pada saat *konfigurasi pipeline.*



Gambar 4.13 *Script Label Map*

Tabel 4.3 Item Label

Id	Nama Item
1	IdgamBighunnahMa
2	IdgamBighunnahNa
3	IdgamBighunnahWa
4	IdgamBighunnahYa
5	IdgamBilaghunnahLa
6	IdgamBilaghunnahRa
7	IkhfahDa
8	IkhfahDho
9	IkhfahDza
10	IkhfahFa
11	IkhfahJa
12	IkhfahKa
13	IkhfahQa
14	IkhfahSa

15	IkhfahSho
16	IkhfahSya
17	IkhfahTa
18	IkhfahTho
19	IkhfahTsa
20	IkhfahZa
21	IkhfahZho
22	IqlabBa
23	IzharA
24	IzharAa
25	IzharGhain
26	IzharHab
27	IzharHak
28	IzharKho

4.3.4 *Konfigurasi Object Detection Training Pipeline*

Konfigurasi yang penulis gunakan yaitu model SSD *Mobilenet* pada penelitian ini. SSD memiliki algoritma relatif sederhana, sehingga model ini mudah dilatih dan lansung diintegrasikan kedalam *system*. *Tensorflow* menerapkan *protobuf* sehingga perlu dilakukan *konfigurasi pipeline*. *Konfigurasi* dilakukan untuk proses *training* dan *evaluasi*. Jumlah kelas yang digunakan oleh penulis pada *script num_classes*: 28 yang artinya jumlah kelas berjumlah 28 yang dapat dilihat di tabel 4.4. *batch_size* yang digunakan 28 yang artinya jumlah sampel yang dimasukkan kedalam *neural network* berjumlah 28 dan sampel tersebut diambil secara acak dari seluruh sampel dataset.

Tabel 4.4 Nama Huruf dan Kelas

Nama kelas	Huruf
IdgamBighunnah	م (Mim)
IdgamBighunnah	ن (Nun)
IdgamBighunnah	و (Waw)
IdgamBighunnah	ي (Yaa)
IdgamBilaghunnah	ر (Ra)
IdgamBilaghunnah	ل (Lam)
Ikhfah	ت (Ta)
Ikhfah	ث (Tsa)
Ikhfah	ج (Jim)
Ikhfah	د (Dal)
Ikhfah	ذ (Dzal)
Ikhfah	ز (Zai)
Ikhfah	س (Sin)
Ikhfah	ش (Syin)
Ikhfah	ص (Shad)
Ikhfah	ض (Dhad)
Ikhfah	ط (Tha)
Ikhfah	ظ (Zha)
Ikhfah	ف (Fa)
Ikhfah	ق (Qaf)
Ikhfah	ك (Kaf)
Iqlab	ب (Ba)
Izhar	ح (Ha Kecil)
Izhar	خ (Kha)
Izhar	ع (‘Ain)
Izhar	غ (Ghain)
Izhar	ه (Ha Besar)
Izhar	ء (Hamzah)

File konfigurasi pipeline disimpan atas nama pipeline_config seperti terlihat di gambar 4.15. file pipeline_config terssebut akan digunakan pada proses training data. Penelitian ini, penulis menggunakan num_steps: 50.00 maka maksimal proses training training model sebesar 50.000 step.

Gambar 4.14 *Pipiline Config*

4.4 *Training*

Training model merupakan tahapan awal dari *neural network*. Pada tahap ini seluruh dataset *train* dilakukan proses *training* agar *system* dapat mempelajari pola *image tajwid* bacaan Al-Quran yang terdiri dari pola *tajwid izhar, iqlab, ikhfah, idgam* beserta dengan huruf-hurufnya. Proses *training* tersebut terjadi di jaringan CNN dengan training model SSD *Moilenet* yang di aplikasikan pada *jupyter notebook.* Model training yang digunakan yaitu *pre-trained model SSD_Mobilenet_v2_coco* dengan *speed 22/millisecond*. Jumlah maksimal *step* yang digunakan 50.000 dengan estimasi waktu yang dibutuhkan 48 jam dalam proses *training* tersebut. Akhir dari proses training tersebut didapatkan tingkat akurasi yang tinggi pada pendeteksian objek tajwid.

```
python Tensorflow\models\research\object_detection\model_main_tf2.py --
model_dir=Tensorflow\workspace\models\my_ssd_mobnet_detec --
pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet_detec\pip
eline.config --num_train_steps=50000
```

Gambar 4.15 Perintah *Training*

Step 50.000 yang penulis lakukan belum tentu didapatkan model yang terbaik, oleh karena itu penulis menggunakan trial and error dengan model checkpoint. Proses training dilihat pada gambar 4.17.

Gambar 4.16 Proses *Training*

Model hasil *training* akan tersimpan dengan nama *folder training my_ssd_mobnet_detec.* File *pipeline_config* digunakan untuk proses *training object detection.* Proses *training* dijalankan dimulai dari *step* 0 sampai dengan maksimal *step* 50.000. Setiap *step*, output yang dikeluarkan sebuah *variable* yang disebut *loss*. *loss* memperlihatkan apakah model yang dilatih berkerja dengan baik atau tidak.

Didalam *libary object detection* telah terdapat *script* dari *export_inference_graph.py*. *script* tersebut untuk menjalankan *konfigurasi*

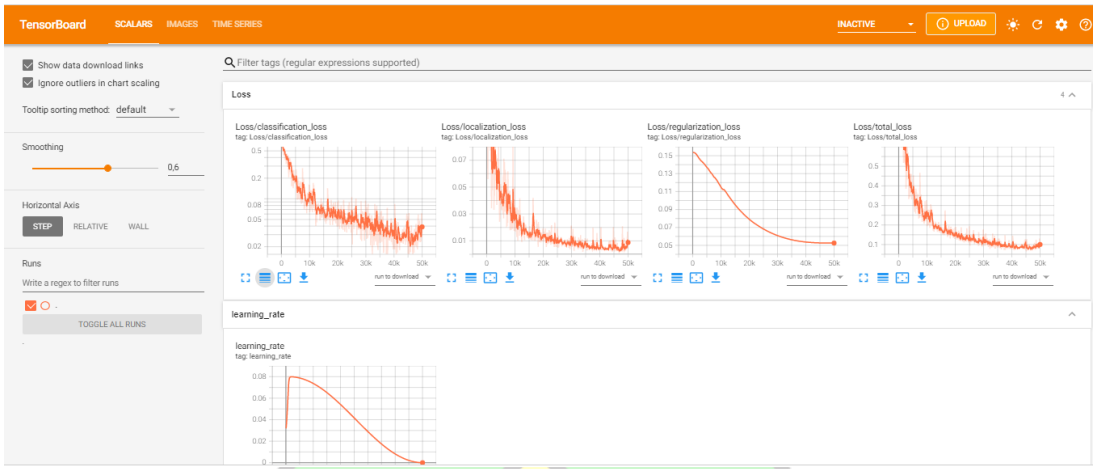
pipiline.config dan *model checkpoint*, yang nantinya menghasilkan berkas baru dengan nama *berkas my_ssd_mobnet_detec*. Setelah proses *training* selesai maka *model training* akan tersimpan pada berkas *my_ssd_mobnet_detec*. Hasil dari training model akan menghasilkan *output* file *checkpoint*.

Setelah proses *training* selesai, file yang tersimpan dan merekam *step* yang terakhir dikonversikan menjadi sebuah *model* hasil training dengan berformat *protobuf (.pb)* yang berada pada berkas *export*. Untuk mengetahui proses *training* dapat dipantau dengan menggunakan *module* yang telah *tensorflow* sediakan yaitu *tensorboard*. Adapun *syntax* dalam menjalankan *tensorboard* tersebut dapat dilihat pada gambar 4.18.

```
tensorborad --logdir=.
```

Gambar 4.17 *Script Tensorboard*

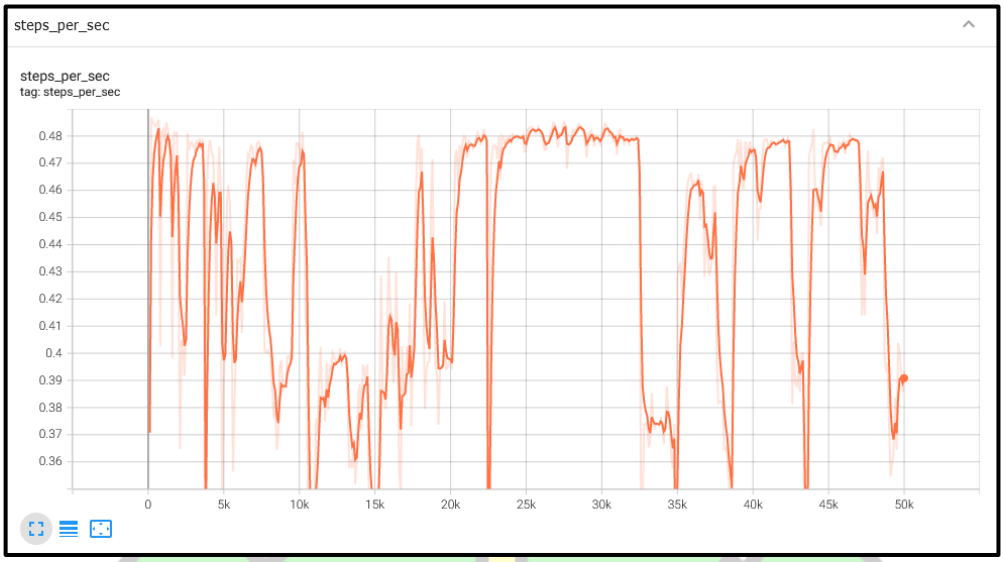
Setelah dilakukan tahap diatas, maka *output* yang dihasilkan berupa alamat untuk membuka *tensorboard* tersebut, dengan alamat *localhost: 6006* yang dibuka pada *browser*. Lalu *tensorboard* memanggil file *checkpoint*, yang mana file tersebut hasil dari proses *training*. penulis dapat melihat grafik-grafik seperti *total_loss* dan *global_step* pada *tensorboard* yang merupakan hasil dari proses *training*. adapun *dashboard* pada *tensorboard* dapat dilihat pada gambar 4.19.



Gambar 4.18 Tampilan Awal *Tensorbord*

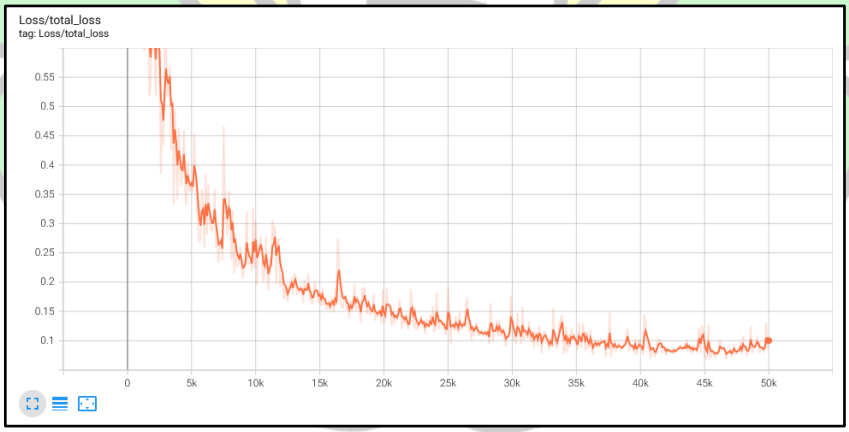
4.4.1 *Analisa Model*

Untuk mengetahui *performa* dari *model* yang telah dilatih, dilakukan analisa terhadap beberapa *parameter* yang dihasilkan dari proses *training* yang dapat dilihat melalui *tensorboard* yang menyediakan layanan yang dapat digunakan untuk berbagi visualisasi pembelajaran mesin *tensorflow*. *TensorBoard* menyediakan fitur *Visualizing Learning* yang memungkinkan pengguna untuk memvisualisasikan berbagai informasi tentang *model* yang dilatih, melacak data dari percobaan pembelajaran mesin, seperti grafik *model* untuk kerugian, akurasi, atau metrik khusus, untuk menyematkan proyeksi, gambar, dan *histogram* bobot dan bias.



Gambar 4.19 *Grafik Training Step*

Pada gambar 4.20 merupakan waktu yang dibutihkan pada setiap *step*. Pada gambar tersebut di hasilkan rata-rata waktu dibutukan pada saat *training* pada setiap *step* sekitar 0.48 detik. Kemudian pada gambar 4.21 yaitu hasil dari pencatatan *step* saat proses *training* dan juga dengan nilai *lossnya*.



Gambar 4.20 *Grafik Total Loss*

4.4.2 Total *Loss*

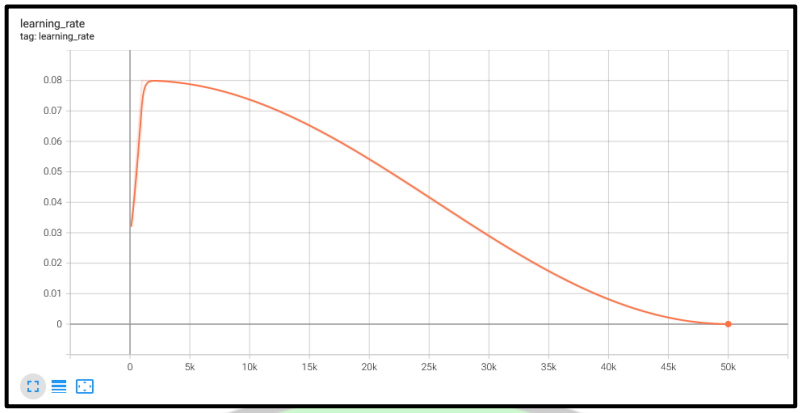
Loss berfungsi untuk memastikan seberapa jauh nilai prediksi yang menyimpang dengan nilai murni dalam data training. Saat training berlangsung,

semua proses terekam pada tensorboard. Aktivitas tang terekam meliputi *training step/second, loss/accuracy, validation loss/accuracy* dan divisualisasikan dengan grafik. Hasil dari totas loss yaitu nilai error yang berasal dari sisa keakuratan pada setiap step.

Jumlah error yang terhitung saat proses training merupakan total loss. Pada gambar 4.20 merupakan grafik dari total loss hasil dari proses training. gambar 4.20 dapat dilihat pada grafik adanya penurunan nilai total loss yang awal nya sangat tinggi, dikarenakan semakin banyak step yang di proses itu membuat nilai dari total loss semakin menurun. Maka hasil dari total loss tersebut semakin kecil loss maka model yang diiput akan semakin baik dalam dalam mendeteksi.

4.4.3 *Learing Rate*

Learing rate ialah yang memantau seberapa banyak perubahan model dalam menanggapi kesalahan yang diperkirakan setiap kali bobot model diperbarui seperti pada gambar 4.22. Menentukan learning rate merupakan tantangan karena nilai yang terlalu kecil dapat mengakibatkan proses training berjalan lama, sedangkan nilai yang terlalu besar dapat mengakibatkan pembelajaran rangkaian bobot yang kurang optimal karena terlalu cepat atau proses training yang tidak stabil.



Gambar 4.21 *Learning Rate*

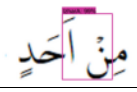
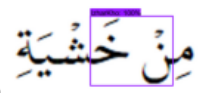
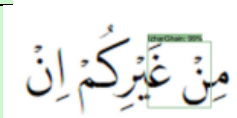
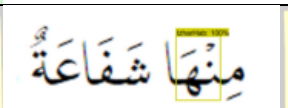
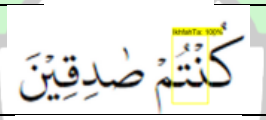
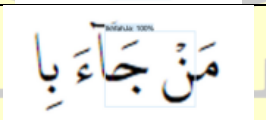
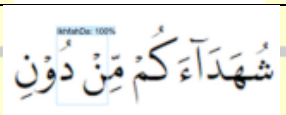
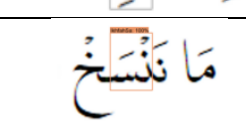
4.4.4 Proses *Training*

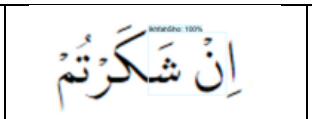
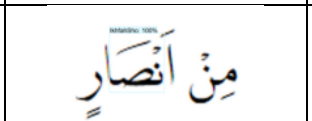
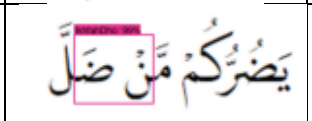
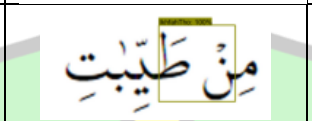
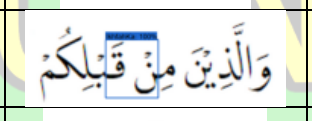
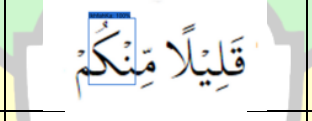
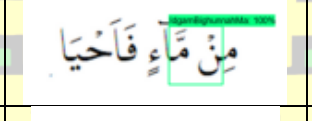
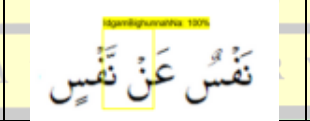
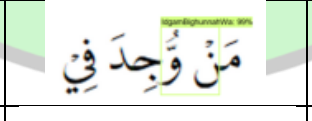
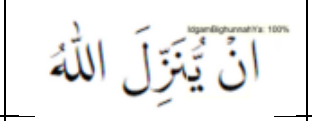
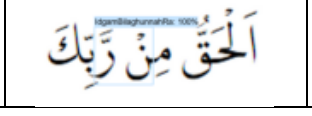
Selanjutnya adalah proses *training* yang merupakan tahapan pengujian dari model yang sudah terbentuk setelah proses *training* di tahap sebelumnya. Proses ini dilakukan untuk dapat mengetahui apakah model yang sudah ada dapat dibaca oleh sistem, apakah objek yang sudah dibuat berupa jenis *tajwid* dapat dikenali oleh sistem, dan apakah sistem dapat melakukan klasifikasi antara *izhar*, *ikhfah*, *iqlab*, *idgham* beserta dengan hurufnya, serta untuk menghitung tingkat keakuratan metode *Mobilenet*-SSD yang telah diterapkan pada sistem klasifikasi jenis *tajwid* di penelitian ini. Model yang sudah melalui proses training akan siap digunakan untuk pendeteksian atau uji coba terhadap jenis *tajwid* beserta huruf-hurufnya yang ditandai oleh kotak berwarna beserta *presentase* akurasinya.

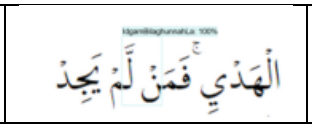
Adapun hasil pengujian *tajwid* pada *system* yang telah dibentuk. Pengujian tajwid terdapat 3 dataset yang akan di uji coba, yaitu pertama dataset *training* yang sudah dilakukan tahap *processing image*, dataset *testing*, dan dataset *multiple*.

Berikut hasil dataset *training* yang sudah dilakukan tahap *processing image* dilihat pada tabel 4.5.

Tabel 4.5 Hasil Dataset Yang Telah Di *Training*

NO	NAMA OBJEK	NAMA SPESIFIK	HASIL DETEKSI	KET	BENAR /SALAH	AKURASI (%)
1	**IZHAR**	ح (Ha Kecil)		Terdeteksi	Benar	99%
2		خ (Kha)		Terdeteksi	Benar	100%
3		ع (‘Ain)		Terdeteksi	Benar	100%
4		غ (Ghain)		Terdeteksi	Benar	99%
5		ه (Ha Besar)		Terdeteksi	Benar	100%
6		ء (Hamzah)		Terdeteksi	Benar	99%
7	**IKHFA**	ت (Ta)		Terdeteksi	Benar	100%
8		ث (Tsa)		Terdeteksi	Benar	100%
9		ج (Jim)		Terdeteksi	Benar	100%
10		د (Dal)		Terdeteksi	Benar	100%
11		ذ (Dzal)		Terdeteksi	Benar	100%
12		ز (Zai)		Terdeteksi	Benar	100%
13		س (Sin)		Terdeteksi	Benar	100%

14		ش (Syin)		Terdeteksi	Benar	100%
15		ص (Shad)		Terdeteksi	Benar	100%
16		ض (Dhad)		Terdeteksi	Benar	99%
17		ط (Tha)		Terdeteksi	Benar	100%
18		ظ (Zha)		Terdeteksi	Benar	100%
19		ف (Fa)		Terdeteksi	Benar	100%
20		ق (Qaf)		Terdeteksi	Benar	100%
21		ك (Kaf)		Terdeteksi	Benar	100%
22	**IQLAB**	ب (Ba)		Terdeteksi	Benar	100%
23		م (Mim)		Terdeteksi	Benar	100%
24	**IDGHAM BIGHUN NAH**	ن (Nun)		Terdeteksi	Benar	100%
25		و (Waw)		Terdeteksi	Benar	99%
26		ي (Yaa)		Terdeteksi	Benar	100%
27	**IDGHAM BILAGH UNNAH**	ر (Lam)		Terdeteksi	Benar	100%

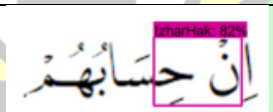
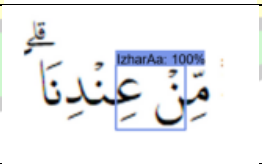
28		ل (Ra)		Terdeteksi	Benar	100%

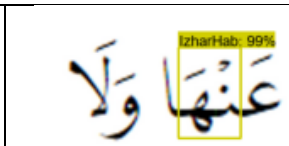
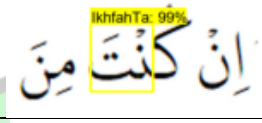
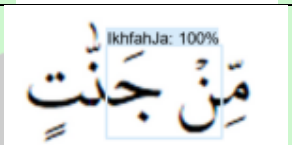
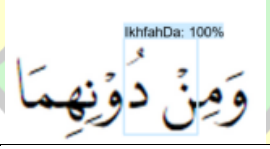
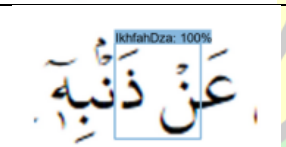
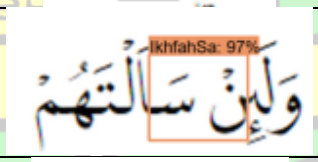
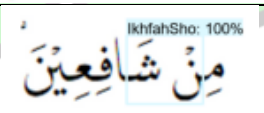
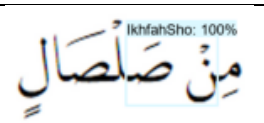
4.4.5 Proses *Testing*

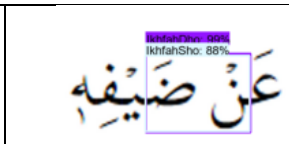
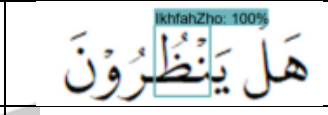
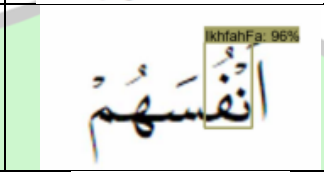
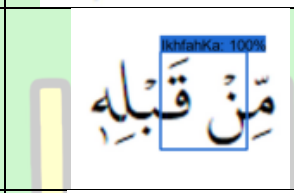
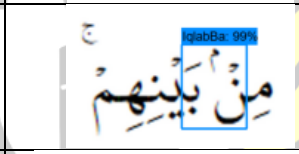
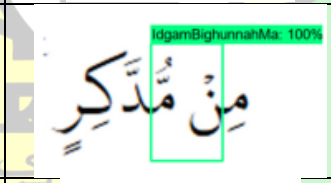
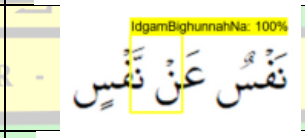
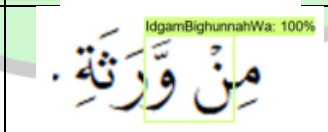
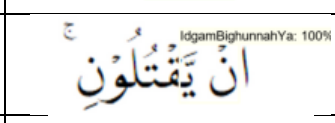
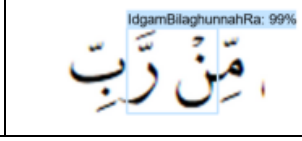
4.4.5.1 *Testing* Dengan Data *Single*

Berikut contoh dari hasil *testing* dapat dilihat di table 4.6 Data yang diuji coba berjumlah dua jenis data yaitu data *single* dan data *multiple*. Data *single* merupakan data yang memiliki satu hukum *tajwid* pada satu gambar, namun berbeda dengan data *multiple* menggunakan lebih dari satu hukum *tajwid* dalam satu gambar. Data dari kedua data tersebut berjumlah jenis 70 data gambar yang dimasukan dalam *system* yang telah dirancang agar dapat dilakukan pendeteksian. Data *single* berjumlah 40 data gambar dan data *multiple* berjumlah 20 data gambar.

Tabel 4.6 Contoh Hasil Testing Dengan Data Single

NO	NAMA OBJECK	NAMA SPESIFIK	HASIL DETEKSI	KET	BENAR /SALAH	AKURASI (%)
1	IZHAR	ح (Ha Kecil)		Terdeteksi	Benar	82%
2		خ (Kha)		Terdeteksi	Benar	100%
3		ع (‘Ain)		Terdeteksi	Benar	100%
4		غ (Ghain)		Terdeteksi	Benar	100%

5		ھ (Ha Besar)		Terdeteksi	Benar	99%
6		ء (Hamzah)		Terdeteksi	Benar	98%
7	**IKHFA**	ت (Ta)		Terdeteksi	Benar	99%
8		ث (Tsa)		Terdeteksi	Benar	100%
9		ج (Jim)		Terdeteksi	Benar	100%
10		د (Dal)		Terdeteksi	Benar	100%
11		ذ (Dzal)		Terdeteksi	Benar	100%
12		ز (Zai)		Terdeteksi	Benar	98%
13		س (Sin)		Terdeteksi	Benar	97%
14		ش (Syin)		Terdeteksi	Salah	100%
15		ص (Shad)		Terdeteksi	Benar	100%

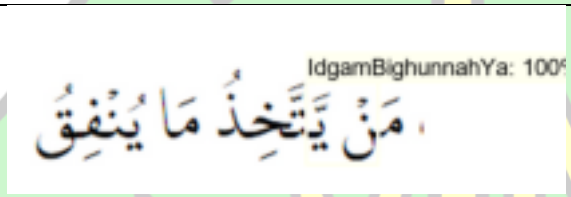
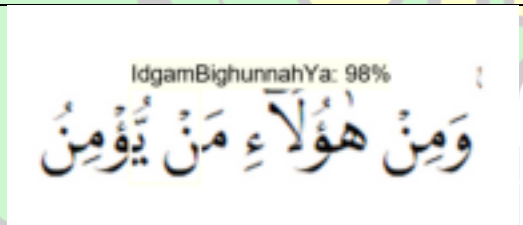
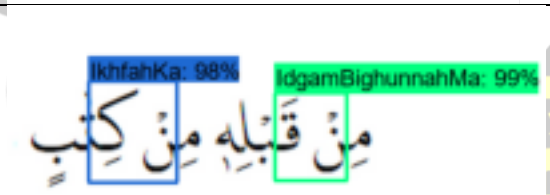
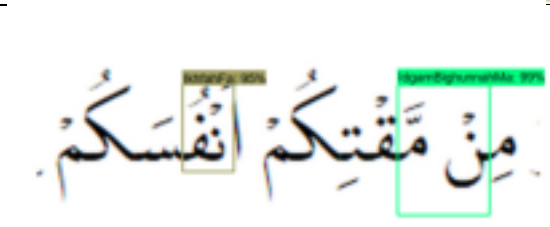
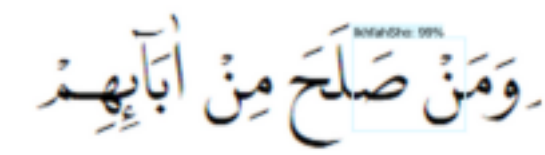
16		ض (Dhad)		Terdeteksi	Salah	88%
17		ط (Tha)		Terdeteksi	Benar	99%
18		ظ (Zha)		Terdeteksi	Benar	100%
19		ف (Fa)		Terdeteksi	Benar	96%
20		ق (Qaf)		Terdeteksi	Salah	100%
21		ك (Kaf)		Terdeteksi	Benar	100%
22	**IQLAB**	ب (Ba)		Terdeteksi	Benar	99%
23		م (Mim)		Terdeteksi	Benar	100%
24	**IDGHAM BIGHUNNAH**	ن (Nun)		Terdeteksi	Benar	100%
25		و (Waw)		Terdeteksi	Benar	100%
26		ي (Yaa)		Terdeteksi	Benar	100%
27	**IDGHAM BILAGHUNNAH**	ر (Ra)		Terdeteksi	Benar	99%

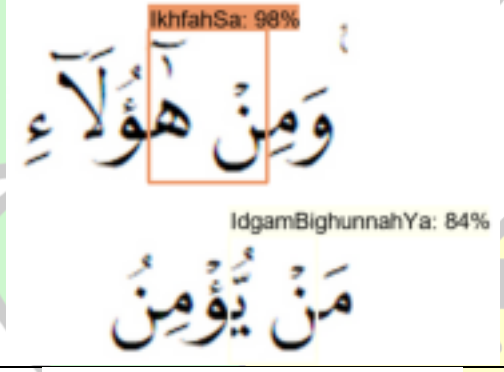
28		ل (Lam)		Terdeteksi	Benar	97%

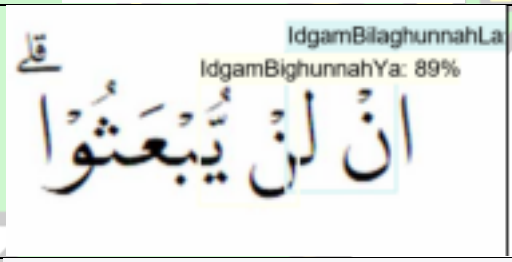
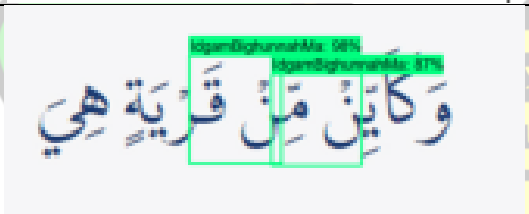
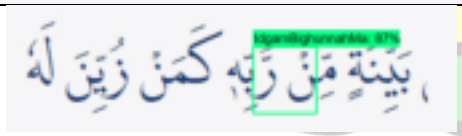
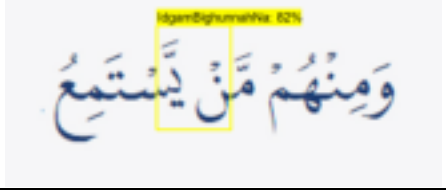
4.4.5.2 Testing Dengan Data *Multyple*

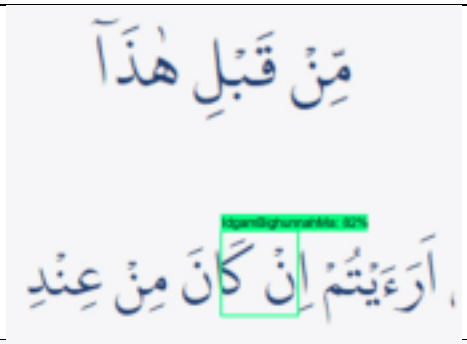
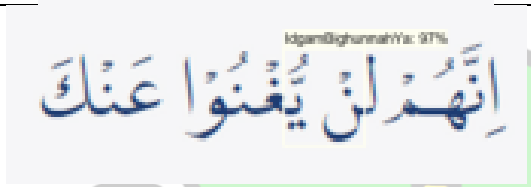
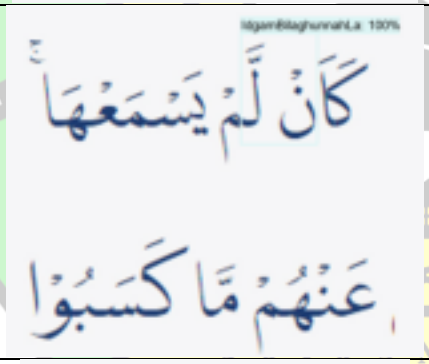
Data multyple merupakan data yang memiliki lenih dari satu hukum bacaan atjwid dalam satu gambar, jumlah data testing multyple berjumlah 20 gambar.

Tabel 4.7 Hasil Dataset *Multyple*

No	HASIL DETEKSI	KETERANGAN	BENAR/ SALAH	AKURASI
1		ي (Yaa) : Terdeteksi ف (Fa): tidak terdeteksi	ي (Yaa) : Benar ف (Fa): -	ي (Yaa) : 100% ف (Fa): -
2		ي (Yaa) : terdeteksi ه (Ha Besar): tidak terdeteksi	ي (Yaa) : benar ه (Ha Besar) : -	ي (Yaa) : 98% ه (Ha Besar) : -
3		ك (Kaf) : terdeteksi ق (Qaf) : terdeteksi	ك (Kaf) : benar ق (Qaf) : salah	ك (Kaf) :98% ق (Qaf) : 99%
4		ف (Fa) : terdeteksi م (Mim) : terdeteksi	ف (Fa) : benar م (Mim) : benar	ف (Fa) : 95% م (Mim): 99%
5		ء (Hamzah) : tidak terdeteksi ص (Shad): terdeteksi	ء (Hamzah): - ص (Shad) : benar	ء (Hamzah) - ص (Shad) : 99%

6		ء (Hamzah): terdeteksi ع (‘Ain) : terdeteksi ي (Yaa) : tidak terdeteksi	ء (Hamzah): benar ع (‘Ain) : benar ي (Yaa) : -	ء (Hamzah): 97% ع (‘Ain) : 100% Ya : -
7		ي (Yaa) : terdeteksi ف (Fa) : terdeteksi	ي (Yaa) : benar ف (Fa) : benar	ي (Yaa) : 90% ف (Fa) : 99%
8		ه (Ha Besar): terdeteksi ي (Yaa) : terdeteksi	ه (Ha Besar): salah ي (Yaa) : benar	ه (Ha Besar): 98% ي (Yaa) : 84%
9		س (Sin) : terdeteksi خ (Kha) : terdeteksi	س (Sin) : benar خ (Kha): benar	س (Sin) : 81% خ (Kha) 98%

10		ص (Shad) : terdeteksi ء (Hamzah): terdeteksi	ص (Shad) : benar ء (Hamzah): benar	ص (Shad) : 100% ء (Hamzah): 99%
11		ي (Yaa) : terdeteksi ب (Ba) : terdeteksi	ي (Yaa) : benar ب (Ba) : benar	ي (Yaa) : 93% ب (Ba) : 83%
12		ي (Yaa) : terdeteksi ل (Lam): terdeteksi	ي (Yaa) : benar ل (Lam): benar	ي (Yaa) : 89% ل (Lam): -
13		م (Mim) : terdeteksi ق (Qaf) : terdeteksi	م (Mim) : benar ق (Qaf) : salah	م (Mim) : 87% ق (Qaf) : 90%
14		ر (Ra) : terdeteksi ز (Zai): tidak terdeteksi	ر (Ra) : salah ز (Zai): -	ر (Ra) : 87% ز (Zai): -
15		ي (Yaa) : terdeteksi ه (Ha Besar): tidak terdeteksi	ي (Yaa) : benar ه (Ha Besar): -	ي (Yaa) : 82% ه (Ha Besar):

16		ق (Qaf) : tidak terdeteksi ع (‘Ain) : tidak terdeteksi ك (Kaf) : terdeteksi	ق (Qaf) : - ع (‘Ain): - ك (Kaf) : salah	ق (Qaf): - ع (‘Ain): - ك (Kaf) : 82%
17		ك (Kaf) : tidak terdeteksi ي (Yaa) : terdeteksi	ك (Kaf) : - ي (Yaa) : benar	ك (Kaf) : - ي (Yaa) : 97%
18		ر (Ra) : terdeteksi و (Waw) : terdeteksi	ر (Ra) : benar و (Waw) : benar	ر (Ra) : 81% و (Waw) : 100%
19		ه (Ha Besar): tidak terdeteksi ل (Lam): terdeteksi	ه (Ha Besar): - ل (Lam): benar	ه (Ha Besar): - ل (Lam): 100%
20		ر (Ra) : terdeteksi د (Dal): tidak terdeteksi	ر (Ra) : benar د (Dal): -	ر (Ra) : 100% د (Dal): -

Keberhasilan yang di dapat ketika dilakukan uji coba deteksi pada data multyple di dapatkan dari proses training data *single*, sehingga ia mampu untuk mendeteksi data *image multyple* yang mana dalam satu *image* terdapat beberapa *tajwid*. Ini didasarkan karna ia dapat mengenali pola yang di *input* kan didalam sistem. Namun terdapat juga kegagalan dalam mendeteksi, itu dikarenakan penulis lebih memfokuskan data *single* untuk proses *training*, sehingga *system* lebih cepat dalam mendeteksi data *single*.

4.5 *Confusion Matrix*

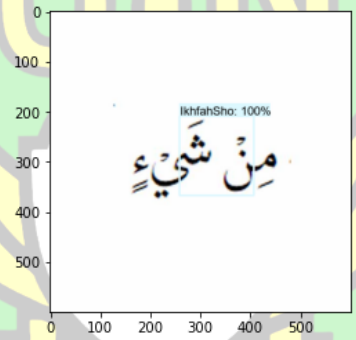
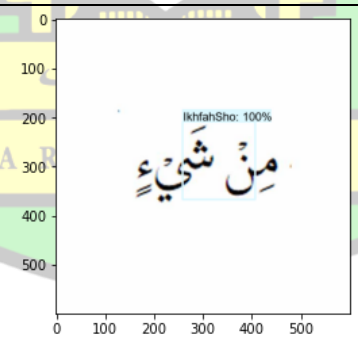
Confusion matrix biasanya digunakan untuk menghitung keakuratan pada sebuah objek deteksi. Adapun *Confusion matrix* ini terdapat 4 istilah sebagai hasil dari proses pendeteksian objek. Istilah tersebut yaitu *True Positive (TP), False Positive (FP), False Negative (FN)* dan *True Negative (TN)* (Sindy,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

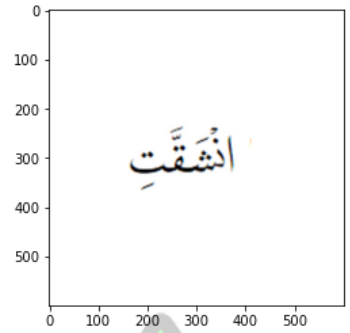
Keterangan:

1. *True Positive (TP)* yaitu dimana *objek* tajwid telah berasil dideteksi oleh model *system*
2. *False Positive (FP)* yaitu data bukan objek *tajwid* namun dideteksi *tajwid* oleh *system*
3. *False Negative (FN)* yaitu data objek *tajwid* namun dideteksi bukan objek tajwid oleh *system*.
4. *True Negative (TN)* yaitu *system* tidak dapat mendeteksi objek *tajwid nun mati*.

Berikut contoh gambar dari 4 istilah *confusion matrix* yaitu *True Positive*, *False Positive*, *False Negative* dan *True Negative* beserta keterangannya. Dapat dilihat di tabel 4.8.

Tabel 4.8 *Confusion Matrix*

Confusion Matrix	**Contoh Gambar**	**Keterangan**
True Positive (TP)		Dimana *objek* tajwid telah berasil dideteksi oleh model *system*
False Positive (FP)		Data bukan objek *tajwid* namun dideteksi *tajwid* oleh *system*
False Negative (FN)		Data objek *tajwid* namun dideteksi bukan objek tajwid oleh *system*.

True Negative (TN)		*system* tidak dapat mendeteksi objek *tajwid nun mati.*

Berikut perhitungan *confusion matrix* pada seluruh data single yang telah di uji coba deteksi menggunakan system dilihat di tabel 4.9.

Tabel 4.9 *Confusion Matrix Single*

No	Hasil Deteksi Image Single	TP	FP	FN	TN
1	ج (Jim)	1			
2	ش (Syin)			1	
3	ك (Kaf)				**1**
4	ط (Tha)	1			
5	ص (Shad)	1			
6	ز (Zai)	1			
7	ت (Ta)	1			
8	ث (Tsa)	1			
9	ج (Jim)	1			
10	د (Dal)	1			
11	ذ (Dzal)	1			
12	ز (Zai)	1			
13	س (Sin)			1	
14	ش (Syin)			1	
15	ص (Shad)	1			
16	ض (Dhad)	1			
17	ط (Tha)	1			
18	ظ (Zha)	1			
19	ف (Fa)			1	
20	ق (Qaf)			1	
21	ك (Kaf)			1	
22	ب (Ba)	1			
23	ح (Ha Kecil)	1			
24	خ (Kha)	1			
25	ع (‘Ain)	1			

26	غ (Ghain)	1			
27	ه (Ha Besar)	1			
28	ء (Hamzah)	1			
29	م (Mim)	1			
30	ن (Nun)	1			
31	و (Waw)	1			
32	ي (Yaa)	1			
33	ر (Lam)	1			
34	ل (Ra)	1			
35	ق (Qaf)			1	
36	ض (Dhad)	1			
37	ب (Ba)	1			
38	ح (Ha Kecil)	1			
39	ه (Ha Besar)	1			
40	ب (Ba)	1			
41	غ (Ghain)	1			
42	ب (Ba)	1			
43	خ (Kha)	1			
44	ب (Ba)	1			
45	ء (Hamzah)	1			
46	ن (Nun)	1			
47	و (Waw)	1			
48	ي (Yaa)	1			
49	ر (Lam)	1			
50	ل (Ra)	1			
Total		**42**	**0**	**7**	**1**

Tabel 4.10 *Hasil Confusion Matrix Single*

TP	**42**
FP	0
FN	7
TN	1
Total	50

Dari pengujian data pada tabel 4.10 didapatkan sebanyak 42 data TP yaitu *True Positive*, dimana data *true positive* terklasifikasi oleh sistem dengan benar. Selanjutnya, 7 data FN yaitu *False Negative* dimana data tajwid huruf yang sesuai namun dideteksi bukan tajwid sesuai dengan hurufnya, dan TN sebanyak 1 data

yaitu sistem yang tidak bisa mendeteksi *tajwid* sama sekali. Total data *testing single* yang digunakan untuk uji coba deteksi berjumlah 50 gambar *tajwid single*.

Berikut perhitungan *confusion matrix* pada seluruh data single yang telah di uji coba deteksi menggunakan system dilihat di tabel 4.11.

Tabel 4.11 *Confusion Matrix Multyple*

No	Hasil Deteksi Image Single	TP	FP	FN	TN
1	ج (Jim)	1			
2	ش (Syin)			1	
3	ك (Kaf)	1			
4	ص (Shad)	1			
5	ط (Tha)	1			
6	ف (Fa)				1
7	ق (Qaf)	1			
8	ب (Ba)			1	
9	ح (Ha Kecil)	1			
10	خ (Kha)	1			
11	ب (Ba)	1			
12	غ (Ghain)	1			
13	ه (Ha Besar)			1	
14	ء (Hamzah)	1			
15	م (Mim)	1			
16	ن (Nun)	1			
17	و (Waw)	1			
18	ي (Yaa)	1			
19	ر (Lam)	1			
20	ب (Ba)	1			
Total		**16**	**0**	**3**	**1**

Tabel 4.12 Hasil *dataset testing multyple*

TP	**16**
FP	0
FN	3
TN	1
Total	20

Dari pengujian data pada tabel 4.12 didapatkan sebanyak 16 data TP yaitu *True Positive,* dimana data *true positive* terklasifikasi oleh sistem dengan benar. Selanjutnya, 3 data FN yaitu *False Negative* dimana data tajwid huruf yang sesuai namun dideteksi bukan tajwid sesuai dengan hurufnya, dan TN sebanyak 1 data yaitu sistem yang tidak bisa mendeteksi *tajwid* sama sekali. Total data *testing multyple* yang digunakan untuk uji coba deteksi berjumlah 20 gambar *tajwid multyple.*

Setelah di lakukan uji coba pendeteksian data *single* dan data *multyple*, maka di hitung keakuratan yang di dapatkan dengan menggunakan persamaan dari akurasi, *precision*, dan *recall*. Berikut proses pencarian perhitungan akurasi, *presisi* dan *recall* pada dataset *testing single*.

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

$= \frac{42}{50}$ x 100 %

= 84 %

Persamaan *Precision* = $Precision = \frac{TP}{(TP + FP)}$ x 100%

$= \frac{42}{42+0}$ x 100 %

$= \frac{42}{42}$ x 100 %

= 100 %

Persamaan *Recall* = $Recall = \frac{TP}{(TP + FN)}$ x 100%

$= \frac{42}{42+7}$ x 100 %

$= \frac{42}{49}$ x 100 %

= 85 %

Persamaan *F1 Score* $= F1\ Score = \frac{1}{2}\ (\frac{1}{Presis} + \frac{1}{Recall})$ x 100%

$= \frac{1}{2}\ (\frac{1}{100} + \frac{1}{85})$ x 100%

$= \frac{1}{2}\ (\frac{2}{185})$ x 100%

$= \frac{1}{92,5}$ x 100%

$= \frac{1}{92,5}$ %

Berikut proses pencarian perhitungan akurasi, *presisi* dan *recall* pada dataset *testing multyple.*

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

$= \frac{16}{20}$ x 100 %

= 80 %

Persamaan *Precision* $= Precision = \frac{TP}{(TP + FP)}$ x 100%

$= \frac{16}{16 + 0}$ x 100 %

$= \frac{16}{16}$ x 100 %

= 100 %

Persamaan *Recall* $= Recall = \frac{TP}{(TP + FN)}$ x 100%

$= \frac{16}{16+3}$ x 100 %

$= \frac{16}{19}$ x 100 %

= 84 %

Persamaan *F1 Score* = $F1\ Score = \frac{1}{2}\ (\frac{1}{Presis} + \frac{1}{Recall})$ x 100%

= $\frac{1}{2}\ (\frac{1}{100} + \frac{1}{84})$ x 100%

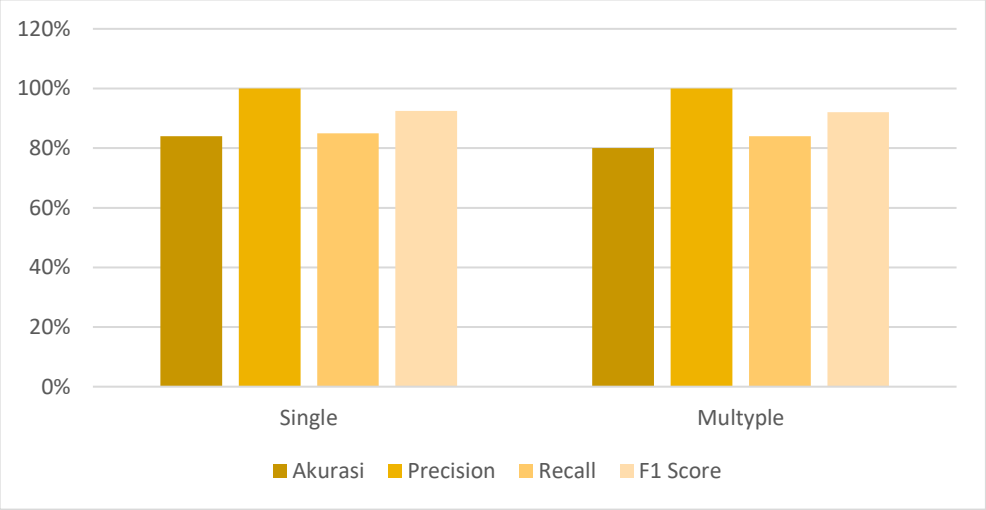
= $\frac{1}{2}\ (\frac{2}{184})$ x 100%

= $\frac{1}{92}$ x 100%

= $\frac{1}{92}$ %

Tabel 4.13 Perhitungan akurasi, Presisi dan Recall pada dataset testing single dan *testing multyple*

	Single	***Multyple***
Akurasi	84%	80%
Presisi	100%	100%
Recall	85%	84%
F1 Score	92,5%	92%



Gambar 4.22 Perhitungan akurasi, presisi dan recall pada *dataset testing single* dan *multyple*

Dari table 4.13 Tingkat akurasi dari dataset *testing single* dan *multyple* pendeteksian dan penghitungan 28 objek bacaan *tajwid nun* mati pada pengujian *testing* mendapatkan nilai yang cukup tinggi pada data *single* dengan nilai *recall* 85% dan data *multyple* nilai *recall* 84%. Data *single presisi* mendapatkan nilai 100% dan data *multiple* dengan nilai 100%, dan terakhir data akurasi *single* akurasinya mendapatkan nilai 84% dan data *multiple* akurasi dengan nilai 80%.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

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

1. Proses dalam implementasi *deep learning* untuk mendeteksi objek pada *tajwid nun* mati pada ayat Al-Quran menggunakan metode *Convolutional Neural Network* (CNN) dengan training data SSD *Mobilenet* berjalan dengan baik dan menghasilkan keberhasilan pendeteksian bacaan *tajwid nun mati*.
2. Dataset yang digunakan adalah sebanyak 350 citra untuk 5 *tajwid nun* mati pada ayat alquran. Dataset terbagi 2 yaitu data *training* dan data *testing*. Data *training* memiliki jumlah 280 citra, data *testing* berjumlah 70 citra. Data *testing* terbagi menjadi 2 bagian yaitu data *testing single* dengan jumlah 50 data, data *testing multiple* dengan 20 data. Data diambil dengan teknik *random* sampling. Arsitektur SSD *Mobilenet* pada penelitian ini menghasilkan Tingkat akurasi dari dataset *testing single* dan *multyple* pendeteksian dan penghitungan 28 objek bacaan *tajwid nun* mati pada pengujian *testing* mendapatkan nilai yang cukup tinggi pada data *single* dengan nilai *recall* 85% dan data *multyple* nilai *recall* 84%. Data *single presisi* mendapatkan nilai 100% dan data *multiple* dengan nilai 100%, dan terakhir data akurasi *single* akurasinya mendapatkan nilai 84% dan data *multiple* akurasi dengan nilai 80%.

5.2 Saran

Didalam penelitian ini peneliti mempunyai beberapa saran, yaitu:

1. Penulis mengharapkan saran perbaikan atau pengembangan oleh penulis seperti menambahkan jumlah dataset lebih dari 350 data.
2. menambahkan objek gambar lain dengan dilakukan secara realtime dan menggunakan metode yang lain seperti YOLO sebagai perbandingan pada hasil pendeteksian bacaan tajwid nun mati menggunakan metode Convolutional Neural Network dengan training model SSD Mobilenet.
3. Menambahkan num_step 100.000 agar tingkat akurasi yang didapat semakin baik dan tinggi.

DAFTAR KEPUSTAKAAN

A. Yudi Permana, P. R. (2019). *Perancangan Sistem Informasi Penjualan Perumahan Mengunakan Metode SDLC Pada PT. Mandiri land Prosperous Berbasis Mobile*. *10*, 153–167.

Abdullah, D., & Dewi, M. (n.d.). *Sistem Pendeteksian Pola Tajwid Al-Quran Hukum Idgham Mutaqaribain Pada Al-Quran menggunkan Metode Gower Dan Legendre*. 65–77.

Ariani, A. (2014). Peran dan Posisi Informasi Teknologi (IT) dalam Dakwah dan Komunikasi. *Alhadharah: Jurnal Ilmu Dakwah*, *13*(25), 27–36. http://jurnal.uin-antasari.ac.id/index.php/alhadharah/article/view/1714/1242

Ariani, S., & Realita. (2015). Program Bengkel Mengaji (Upaya Peningkatan Kemampuan Tahsin Al-Qur'an Mahasiswa PAI). *Jurnal Mudarrisuna - Media Kajian Pendidikan Agama Islam*, *5*(1), 113–144.

Ashadiqi, M. H., Erlansari, A., & Farady, F. (2020). Aplikasi Pembelajaran Ilmu Tajwid Berbasis Android. *Jurnal Rekursif*, *8*(1), 59–70. http://ejournal.unib.ac.id/index.php/rekursif/%0AAPLIKASI

Bagas, H. W. E. M. (2021). *Jurnal resti*. *1*(10), 476–481.

Budiarjo, D. D. (2020). *Implementasi Sistem Cerdas Pada Otomatisasi Pendeteksian Jenis Kendaraan Dijalan Raya*. universitas semarang.

Burhanudin, M. (2018). *Deteksi Hukum Tajwid Mad Lazim Harfi Musyba pada Ayat Alquran Menggunakan Deep Convolutional Neural Network*. Universitas Bhayangkara.

Dewi, S. R. (2018). *Deep Learning Object Detection Pada Video Menggunakan Tensorflow Dan Convolutional Neural Network* [Universitas islam indonesia]. https://dspace.uii.ac.id/bitstream/handle/123456789/7762/14611242_Syarifah Rosita Dewi_Statistika.pdf?sequence=1

Dompeipen, T. A., & Najoan, M. E. I. (2021). Computer Vision Implementation for Detection and Counting the Number of Humans. *Jurnal Teknik Informatika*, *16*(1), 65–76. https://doi.org/10.35793/jti.16.1.2021.31471

Fikriya, Z. A., Irawan, M. I., & Soetrisno., S. (2017). Implementasi Extreme Learning Machine untuk Pengenalan Objek Citra Digital. *Jurnal Sains Dan Seni ITS*, *6*(1). https://doi.org/10.12962/j23373520.v6i1.21754

Hendriyana, & Maulana, Y. H. (2020). Identifikasi Jenis Kayu Menggunakan Convolutional Neural Network Dengan Arsitektur Mobilenet. *Jurnal Resti*, *4*(1), 70–76. http://jurnal.iaii.or.id/index.php/RESTI/article/view/1445/203

Hidayatullah, M. A. I. (2020). *Pengenalan Makhraj Huruf Hijaiyah Menggunakan Convolutional Neural Network*. Universitas Sriwijaya.

Iskandar, B. A. (2014). *Materi Dasar Islam Dari Akar Sampai Daunnya*.

Kiki Firmantoro, Anton, E. R. N. (2016). *Animasi interaktif pengenalan hewan untuk pendidikan anak usia dini*. *XIII*(2), 14–22.

Kurniawan, B. (2015). *Analisis Perbandingan Komputasi GPU dengan CUDA dan Komputasi CPU untuk Image dan Video Processing*. 25–31.

Lorentius, C. A., Adipranata, R., & Tjondrowiguno, A. (2019). Pengenalan Aksara Jawa dengan Menggunakan Metode Convolutional Neural Network. *E-Proceeding of Engineering*.

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

Mashita, S. N. (2020). *Implementasi Deep Learning Object Detection Rambu K3 Pada Video Menggunakan Metode Convolutional Neural Network (CNN) Dengan Tensorflow (Studi Kasus: Rambu Kesehatan dan Keselamatan Kerja (K3) Jalur Evakuasi dan Alat Pemadam Api pada Gedung FMIPA UII)*. universitas islam indonesia.

Milatuchulwiyah. (2018). *Pengaruh Pemahaman Ilmu Tajwid Terhadap*. Institut Agama Islam Negeri (IAIN) Metro.

Mujilahwati, S., Sholihin, M., & Wardhani, R. (2021). Optimasi Hyperparameter TensorFlow dengan Menggunakan Optuna di Python: Study Kasus Klasifikasi Dokumen Abstrak Skripsi. *Jurnal Media Informatika Budidarma*, *5*(3), 1084. https://doi.org/10.30865/mib.v5i3.3090

Muliadi, D. (2020). *Aplikasi Pendeteksian Objek Buah-Buahan Yang Memiliki Kemiripan Menggunakan Algoritma Faster R-Cnn Berbasis Android*. Universitas Sumatera Utara.

Nufus, N., Ariffin, D. M., Satyawan, A. S., Nugraha, R. A. S., Asysyakuur, M. I., Marlina, N. N. A., Parangin, C. H., & Ema, E. (2021). Sistem Pendeteksi Pejalan Kaki Di Lingkungan Terbatas Berbasis SSD MobileNet V2 Dengan Menggunakan Gambar 360° Ternormalisasi. *Prosiding Seminar Nasional Sains Teknologi Dan Inovasi Indonesia (SENASTINDO)*, *3*(November), 123–134. https://doi.org/10.54706/senastindo.v3.2021.123

Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). Implementasi Deep Learning Menggunakan Convolutional Neural Network (Cnn) Pada Ekspresi Manusia. *Algor*, *2*, 12–21.

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

Rachardi, F. (2020). *Deteksi gambar gestur kosakata bahasa isyarat indonesia dengan convolutional neural network*. Universitas Islam Negeri Syarif Hidayatullah.

Radliya, N. R. (2015). *Data Set Dalam Pengelolaan Data Mining Algoritma Naive*

Bayes.

Ratna, S. (2020). Pengolahan Citra Digital Dan Histogram Dengan Phyton Dan Text Editor Phycharm. *Technologia: Jurnal Ilmiah*, *11*(3), 181. https://doi.org/10.31602/tji.v11i3.3294

Religia, Y. (2019). *Feature Extraction Untuk Klasifikasi Pengenalan Wajah Menggunakan Support Vector Machine Dan K-Nearest Neighbor*. *14*(September), 85–92.

Rena, P. N. (2019). *PeneraPan Metode Convolution Neural Network Pada Pendeteksian Gambar Notasi Balok*. Universitas Islam Negerii Syarif.

Roihan, A., Abas Sunarya, P., & Rafika, A. S. (2020). IJCIT (Indonesian Journal on Computer and Information Technology) Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper. *IJCIT (Indonesian Journal on Computer and Information Technology)*, *1*, 75–82.

Setiabudidya, D. (2017). *Penggunaan Piranti Lunak Jupyter Notebook Dalam Upaya Mensosialisasikan Open Science*. 1–4.

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

Siska, D., & Fadillah, C. (n.d.). Sistem Pendeteksi Pola Tajwid Al-Qur'an Hukum Idgram Bi-Ghunnah dan Bila Ghunnah pada Citra Menggunakan Metode Nei and Li. *TECHSI: Teknik Informatika*, 205–214. https://ojs.unimal.ac.id/index.php/techsi/article/view/127

Suartika E. P, I Wayan, Wijaya Arya Yudhi, S. R. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) Pada Caltech 101. *Jurnal Teknik ITS*, *5*(1), 76. http://repository.its.ac.id/48842/

Sudiarjo, A., Mariana, A. R., & Nurhidayat, W. (2015). Aplikasi Pembelajaran Ilmu Tajwid , Waqaf dan Makharijul Huruf Berbasis Android. *Jurnal Sisfotek Global*, *5*(2), 54–60. http://journal.stmikglobal.ac.id/index.php/sisfotek/article/view/80

Wantania, B. B. M. (2020). Penerapan Pendeteksian Manusia Dan Objek Dalam Keranjang Belanja Pada Antrian Di Kasir. *Jurnal Teknik Informatika*, *15*(2), 101–108. https://doi.org/10.35793/jti.15.2.2020.29004

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

Yakib. (2020). *Rancang Bangun Sistem Klasifikasi Mineral Dan Batuan Menggunakan Tensorflow.Js*. Universitas Hasanuddin.

LAMPIRAN

Lampiran 1 Instalasi

Langkah-langkah Persiapan *Software*

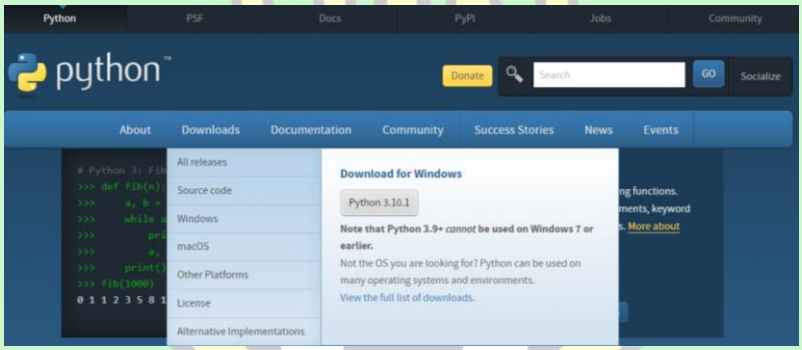
I. *Python*

Langkah-langkah penginstalan *python* sebagai berikut:

a) Mengunjungi halaman website untuk mendowload *Python :*

https://www.python .org/

b) Kemudian klik download dan pilih *windows*, lalu klik *python* untuk mendownload.



Gambar 1: python

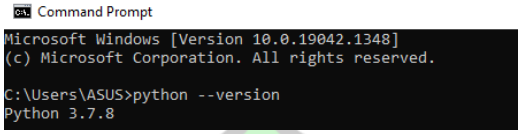
Sumber: *https://www.python .org/*

c) Setelah berhasil di download lalu di install, ketika pada proses penginstalan tempatkan pada folder *python* di C:\Program Files*Python* 37

Gambar 2: folder *python*

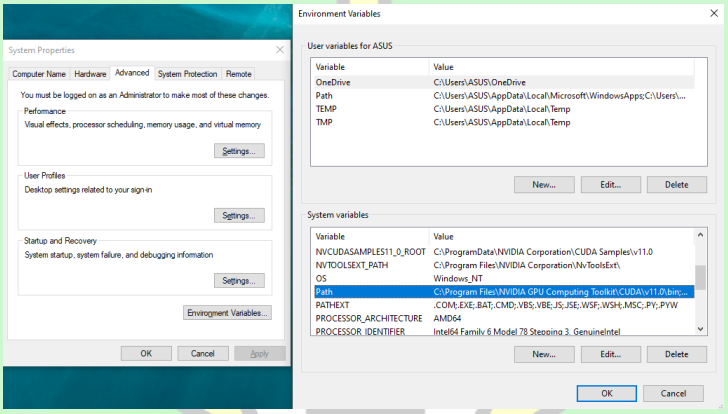
d) Setelah *Anaconda* berhasil di install, kemudian buka Command prompt untuk melihat versi *python* .

i. *python --version*



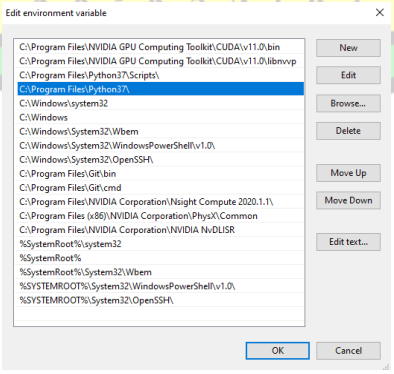
Gambar 3: versi *python*

e) Selanjutnya buat pengaturan pada setting path di environment variables, seperti pada gambar dibawah ini.



Gambar 4: *setting path di environment variables*

f) Pada setting environment tambahkan variables sesuai folder *Phython* diatas



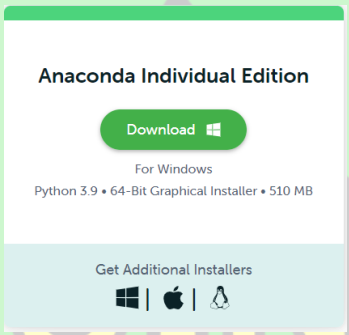
Gambar 5: *Setting Environment*

II. *Anaconda*

Langkah-langkah dalam penginstalan *Anaconda* sebagai berikut:

a) Mengunjungi halaman website untuk mendowload *Anaconda*: *https://www.Anaconda.com/products/individual*

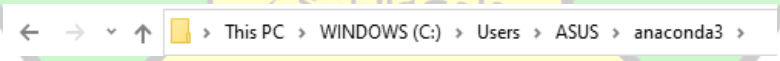
b) Kemudian klik *anaconda individual edition* dan pilih *download* dengan



Gambar 6: *download anaconda*

Sumber: https://www.Anaconda.com/products/individual

c) Setelah berhasil di download lalu di install, ketika pada proses penginstalan tempatkan pada folder *Anaconda* di C:\Users\ASUS*Anaconda*3

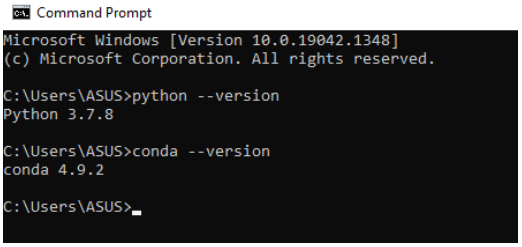


Gambar 7: folder *Anaconda*

d) Setelah *Anaconda* berhasil di install, kemudian buka Command prompt untuk melihat versi *Anaconda*.

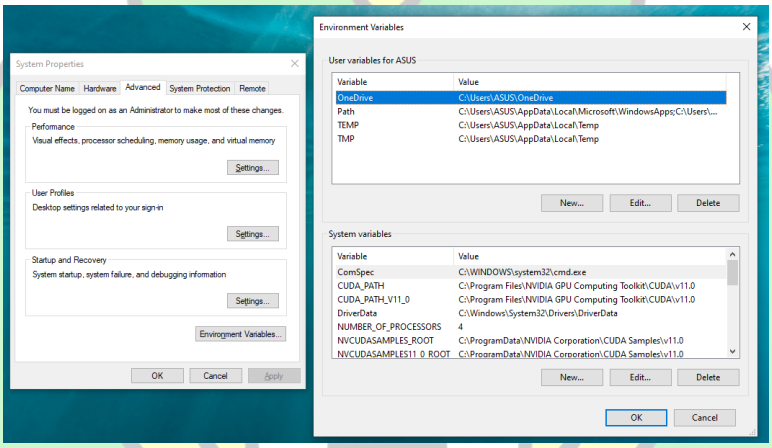
i. *python –version*

ii. *conda --version*



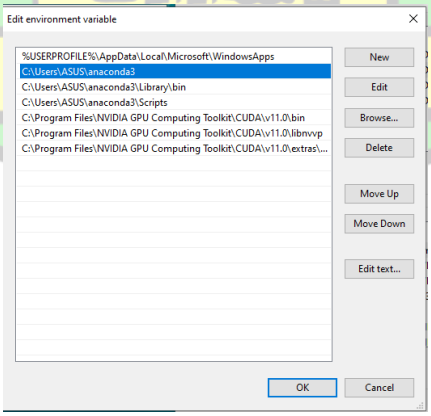
Gambar 8: melihat versi *Anaconda*

e) Selanjutnya buat pengaturan pada setting path di environment variables, seperti pada gambar dibawah ini.



Gambar 9: Setting Path Di Environment Variables

f) Pada setting environment tambahkan variables sesuai folder *Anaconda* diatas



Gambar *10: Setting Environment*

III. *CuDNN*

Langkah-langkah dalam penginstalan CUDA sebagai berikut:

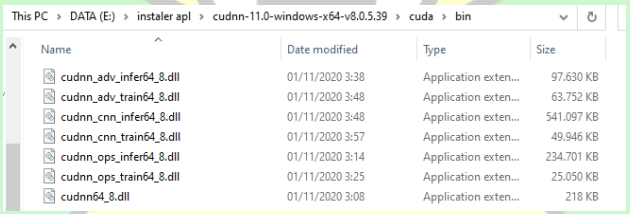
a) Mengunjungi halaman website untuk mendowload *CuDNN*: *https://developer.Nvidia.com/rdp/CuDNN-archive*

b) Kemudian pilih Download *CuDNN* v8.0.5 (November 9th, 2020), for CUDA 11.0 seperti pada gambar di bawah dan kemudian pilih *CuDNN Library* for Windows (x86)

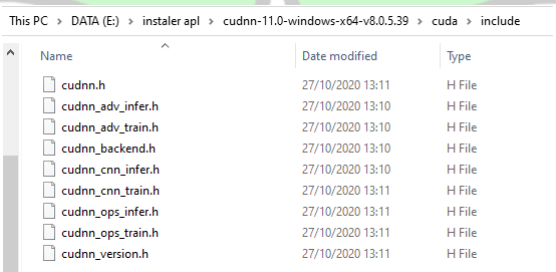
Gambar *11: Download CuDNN*

Sumber: https://developer.Nvidia.com/rdp/CuDNN-archive

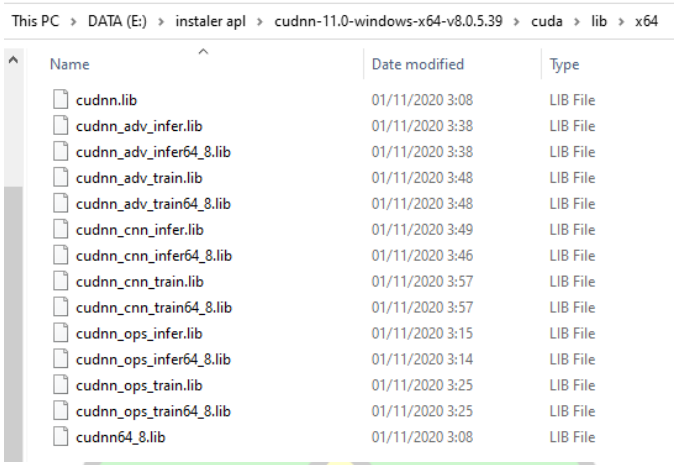
c) Setelah berhasil di download lalu di estrak file. Terdapat *bin*, *include*, *lib* yang akan dipindahkan pada file CUDA



Gambar *12: bin* CUDA



Gambar *13: include* CUDA



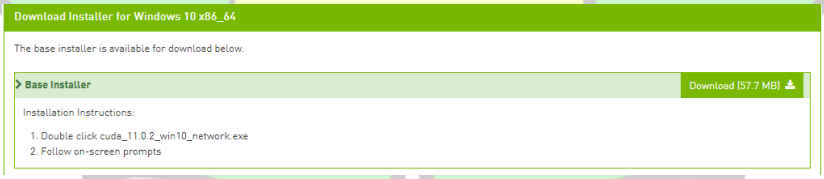
Gambar *14: lib* CUDA

IV. CUDA

Langkah-langkah dalam penginstalan CUDA sebagai berikut:

a) Mengunjungi halaman website untuk mendowload CUDA :https://*developer.Nvidia*.com/CUDA-11.0-download-archive?target_os=Windows&target_arch=x86_64

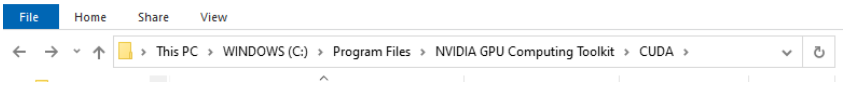
b) Kemudian klik *operating system window*, lalu klik *Architecture x86_64*, pilih version 10, lalu klik *Installer Type exe (Network).* Kemudian klik download.



Gambar 15: Download CUDA

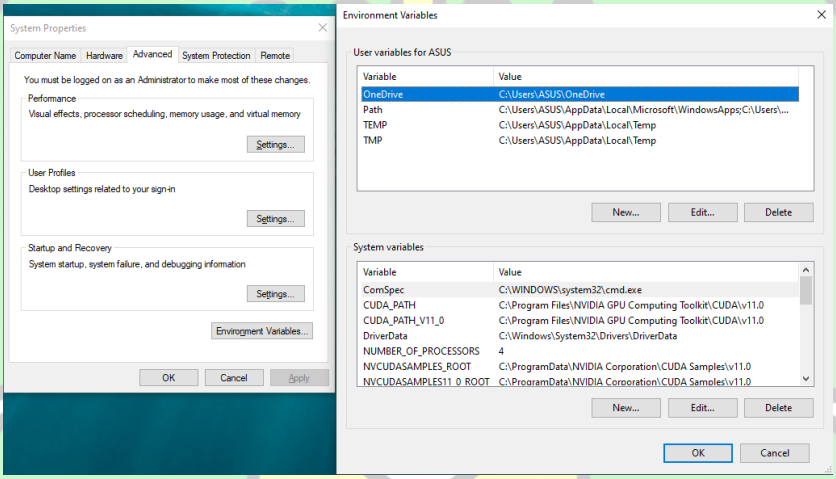
Sumber: :https://*developer.Nvidia*.com/CUDA-11.0-download archive?target_os=Windows&target_arch=x86_64

c) Setelah berhasil di download lalu di install, ketika pada proses penginstalan tempatkan pada folder *Anaconda* di C:\ProgramFiles*NVIDIAGPU*ComputingToolkit\CUDA



Gambar 16: folder *Anaconda*

d) Selanjutnya buat pengaturan pada setting path di environment variables, seperti pada Gambar dibawah ini.



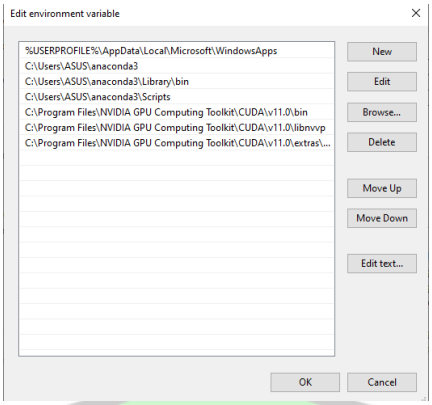
Gambar 17: *setting path di environment variables* CUDA

e) Pada setting environment tambahkan variables yaitu

C:\Program Files*NVIDIA GPU* Computing Toolkit\CUDA\v11.0\bin

C:\Program Files*NVIDIA GPU* Computing Toolkit\CUDA\v11.0\libnvvp

C:\Program Files*NVIDIA GPU* Computing Toolkit\CUDA\v11.0\extras\CUPTI\lib64



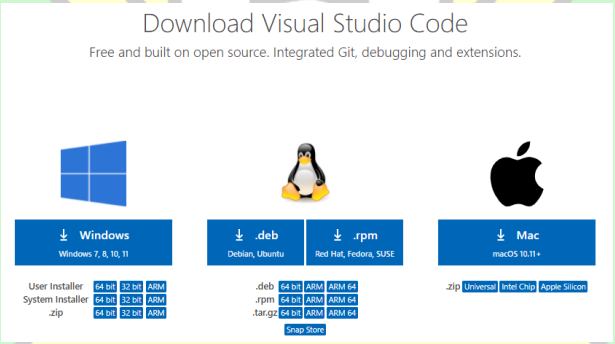
Gambar 18: *Setting Environment Tambahkan Variables* CUDA

V. *Visual Studio Code*

Langkah-langkah dalam penginstalan *Visual Studio Code* sebagai berikut:

a) Mengunjungi halaman website untuk mendowload *Visual Studio Code*: https://code.visualstudio.com/download

b) Kemudian pilih windows dan klik download



Gambar 19: Download Visual Studio Code

Sumber: https://code.visualstudio.com/download

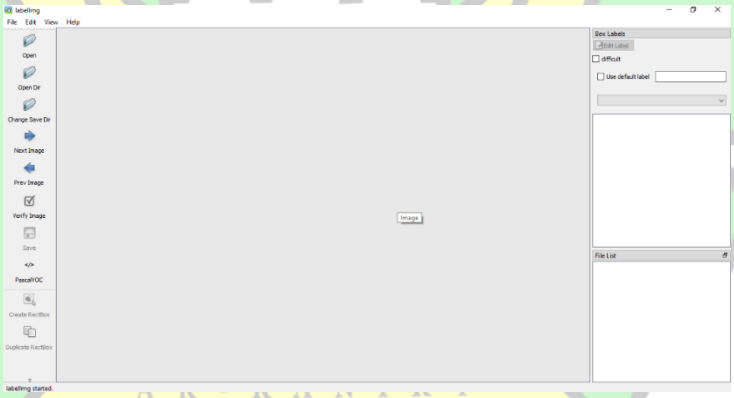
c) Setelah berhasil di download lalu di install

VI. *Labelimg*

Penulis menggunakan labelImg untuk *image* annotation atau pelabelan gambar. LabelImg dapat diunduh di repositori *Github* dengan alamat https://*Github*.com/tzutalin/labelImg. Agar dapat berjalan di jupyter notebook labelImg membutuhkan *Python* , PyQt5 dan lxml.

Langkah-langkah untuk instalasi labelImg adalah sebagai berikut: (Budiarjo, 2020)

1. Unduh labelImg dari https://*Github*.com/tzutalin/labelImg dijupyter
2. Ekstrak berkas yang telah diunduh
3. Masuk ke direktori labelImg kemudian buka Command Promt
4. Ketik *python* labelImg.py pada command promt
5. Kemudian akan muncul tampilan seperti



Gambar *20: Labelimg*

VII. *Photoshop*

a) Langkah-langkah dalam penginstalan *Photoshop* sebagai berikut:

b) Mengunjungi halaman website untuk mendowload *Photoshop*:

https://www.nesabamedia.com/download-adobe-*Photoshop*-cs6/

c) Kemudian pilih windows dan klik download

d) Setelah berhasil di download lalu di install

Lampiran 2 Coding

jupyter 1. Image Collection Last Checkpoint: 12/11/2021 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Kernel starting, please wait... Trusted tfodj

Run Markdown

1. Import Dependencies

In [1]:
```
!pip install opencv-python
```

```
Requirement already satisfied: opencv-python in f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\lib\site-pac
kages (3.4.2.17)
Requirement already satisfied: numpy>=1.14.5 in f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\lib\site-pac
kages (from opencv-python) (1.21.5)

WARNING: You are using pip version 21.3.1; however, version 22.0.4 is available.
You should consider upgrading via the 'f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\scripts\python.exe -m
pip install --upgrade pip' command.
```

In [2]:
```
# Import opencv
import cv2

# Import uuid
import uuid

# Import Operating System
import os

# Import time
import time
```

In [3]:
```
!pip install opencv-python==3.4.2.17 numpy==1.14.5
```

```
Requirement already satisfied: opencv-python==3.4.2.17 in f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\
lib\site-packages (3.4.2.17)
Collecting numpy==1.14.5
  Using cached numpy-1.14.5-cp37-none-win_amd64.whl (13.4 MB)
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.21.5
    Uninstalling numpy-1.21.5:
      Successfully uninstalled numpy-1.21.5
Successfully installed numpy-1.14.5

ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviou
r is the source of the following dependency conflicts.
object-detection 0.1 requires apache-beam, which is not installed.
object-detection 0.1 requires avro-python3, which is not installed.
object-detection 0.1 requires contextlib2, which is not installed.
object-detection 0.1 requires pycocotools, which is not installed.
tf-models-official 2.7.0 requires google-api-python-client>=1.6.7, which is not installed.
tf-models-official 2.7.0 requires kaggle>=1.3.9, which is not installed.
tf-models-official 2.7.0 requires oauth2client, which is not installed
```

2. Define Images to Collect

In [4]:
```
labels = ['IdgamBighunnahMa', 'IdgamBighunnahNa', 'IdgamBighunnahWa', 'IdgamBighunnahYa', 'IdgamBilaghunnahLa', 'IdgamBilagh
number_imgs = 10
```

In [5]:
```
number_imgs
```

Out[5]: 10

3. Setup Folders

In [6]:
```
IMAGES_PATH = os.path.join('Tensorflow', 'workspace', 'images', 'collectedimages')
```

In [7]:
```
print(IMAGES_PATH)
```

```
Tensorflow\workspace\images\collectedimages
```

In [8]:
```
os.name
```

Out[8]: 'nt'

In [9]:
```
if not os.path.exists(IMAGES_PATH):
    if os.name == 'posix':
        !mkdir -p {IMAGES_PATH}
    if os.name == 'nt':
         !mkdir {IMAGES_PATH}
for label in labels:
    path = os.path.join(IMAGES_PATH, label)
    if not os.path.exists(path):
        !mkdir {path}
```

4. Capture Images

In [10]:
```
IMAGES_PATH
```

Out[10]:
```
'Tensorflow\\workspace\\images\\collectedimages'
```

In [ ]:
```
for label in labels:
    cap = cv2.VideoCapture(0)
    print('Collecting images for {}'.format(label))
    time.sleep(5)
    for imgnum in range(number_imgs):
        print('Collecting image {}'.format(imgnum))
        ret, frame = cap.read()
        imgname = os.path.join(IMAGES_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
        cv2.imwrite(imgname, frame)
        cv2.imshow('frame', frame)
        time.sleep(2)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
```

5. Image Labelling

In [11]:
```
!pip install --upgrade pyqt5 lxml
```

```
Requirement already satisfied: pyqt5 in f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\lib\site-packages
(5.15.6)
Requirement already satisfied: lxml in f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\lib\site-packages (4.
8.0)
Requirement already satisfied: PyQt5-sip<13,>=12.8 in f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\lib\si
te-packages (from pyqt5) (12.9.0)
Requirement already satisfied: PyQt5-Qt5>=5.15.2 in f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\lib\site
-packages (from pyqt5) (5.15.2)

WARNING: You are using pip version 21.3.1; however, version 22.0.4 is available.
You should consider upgrading via the 'f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\scripts\python.exe -m
pip install --upgrade pip' command.
```

In [12]:
```
!pip list
```

...

In [13]:
```
LABELIMG_PATH = os.path.join('Tensorflow', 'labelimg')
```

In [14]:
```
if not os.path.exists(LABELIMG_PATH):
    !mkdir {LABELIMG_PATH}
    !git clone https://github.com/tzutalin/labelImg {LABELIMG_PATH}
```

In [15]:
```
LABELIMG_PATH
```

Out[15]:
```
'Tensorflow\\labelimg'
```

In [16]:
```
if os.name == 'posix':
    !cd {LABELIMG_PATH} && make qt5py3
if os.name =='nt':
    !cd {LABELIMG_PATH} && pyrcc5 -o libs/resources.py resources.qrc
```

In [ ]:
```
!cd {LABELIMG_PATH} && python labelImg.py
```

jupyter 2. Training and Detection Last Checkpoint: 12/11/2021 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Kernel starting, please wait... Trusted tfodj

Run Markdown

0. Setup Paths

In [48]:
```
import os
```

In [49]:
```
CUSTOM_MODEL_NAME = 'my_ssd_mobnet_detec'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x32
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'
```

In [50]:
```
paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow','models'),
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
    'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'export'),
    'TFJS_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH':os.path.join('Tensorflow','protoc')
 }
```

In [51]:
```
files = {
    'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM_MODEL_NAME, 'pipeline.config'),
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}
```

In [52]:
```
for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
            !mkdir {path}
```

In [53]:
```
files
```

Out[53]:
```
{'PIPELINE_CONFIG': 'Tensorflow\\workspace\\models\\my_ssd_mobnet_detec\\pipeline.config',
 'TF_RECORD_SCRIPT': 'Tensorflow\\scripts\\generate_tfrecord.py',
 'LABELMAP': 'Tensorflow\\workspace\\annotations\\label_map.pbtxt'}
```

1. Download TF Models Pretrained Models from Tensorflow Model Zoo and Install TFOD

In [54]:
```
# https://www.tensorflow.org/install/source_windows
```

In [55]:
```
if os.name=='nt':
    !pip install wget
    import wget
```

```
Requirement already satisfied: wget in f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\lib\site-packages (3.
2)

WARNING: You are using pip version 21.3.1; however, version 22.0.4 is available.
You should consider upgrading via the 'f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\scripts\python.exe -m
pip install --upgrade pip' command.
```

In [56]:
```
if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection')):
    !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
```

In [57]:
```
# Install Tensorflow Object Detection
if os.name=='posix':
    !apt-get install protobuf-compiler
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && cp object_detection/packages/

if os.name=='nt':
    url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
    os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && copy object_detection\\packag
    !cd Tensorflow/models/research/slim && pip install -e .
```

In [59]:
```
VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'builders', 'model_builder_tf2_te
# Verify Installation
!python {VERIFICATION_SCRIPT}
```

```
[  SKIPPED ] ModelBuilderTF2Test.test_session
[ RUN      ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
I0321 14:09:01.123618 12004 test_util.py:2309] time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extrac
tor): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
[ RUN      ] ModelBuilderTF2Test.test_unknown_meta_architecture
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
I0321 14:09:01.124828 12004 test_util.py:2309] time(__main__.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_meta_architecture
[ RUN      ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
I0321 14:09:01.126416 12004 test_util.py:2309] time(__main__.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.
0s
[       OK ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
----------------------------------------------------------------------
Ran 24 tests in 48.454s

OK (skipped=1)
```

In [ ]:
```
!pip install tensorflow --upgrade
```

In [ ]:
```
!pip install tensorflow --upgrade
```

In [ ]:
```
!pip install pyyaml
```

In [ ]:
```
!python -m pip install msvc-runtime
```

In [ ]:
```
!pip install scipy
```

In [ ]:
```
!pip install Pillow
```

In [58]:
```
!pip install numpy --upgrade
```

```
Requirement already satisfied: numpy in f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\lib\site-packages
(1.21.5)

WARNING: You are using pip version 21.3.1; however, version 22.0.4 is available.
You should consider upgrading via the 'f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\scripts\python.exe -m
pip install --upgrade pip' command.
```

In [ ]:
```
!pip uninstall protobuf matplotlib -y
!pip install protobuf matplotlib==3.2
```

In [60]:
```
import object_detection
```

In [38]:
```
!pip list
```

In [61]:
```
if os.name =='posix':
    !wget {PRETRAINED_MODEL_URL}
    !mv {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}
if os.name == 'nt':
    wget.download(PRETRAINED_MODEL_URL)
    !move {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}
```

```
100% [..............................................................................] 20515344 / 20515344        1 file(s) move
d.

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

2. Create Label Map

In [62]:
```
labels = [{'name':'IdgamBighunnahMa', 'id':1}, {'name':'IdgamBighunnahNa', 'id':2},
          {'name':'IdgamBighunnahWa', 'id':3}, {'name':'IdgamBighunnahYa', 'id':4},{'name':'IdgamBilaghunnahLa', 'id':5},
          {'name':'IdgamBilaghunnahRa', 'id':6}, {'name':'IkhfahDa', 'id':7}, {'name':'IkhfahDho', 'id':8}, {'name':'IkhfahD
          {'name':'IkhfahFa', 'id':10}, {'name':'IkhfahJa', 'id':11}, {'name':'IkhfahKa', 'id':12}, {'name':'IkhfahQa', 'id'
          {'name':'IkhfahSa', 'id':14}, {'name':'IkhfahSho', 'id':15}, {'name':'IkhfahSya', 'id':16}, {'name':'IkhfahTa', 'i
          {'name':'IkhfahTho', 'id':18}, {'name':'IkhfahTsa', 'id':19}, {'name':'IkhfahZa', 'id':20}, {'name':'IkhfahZho', '
          {'name':'IqlabBa', 'id':22}, {'name':'IzharA', 'id':23}, {'name':'IzharAa', 'id':24}, {'name':'IzharGhain', 'id':2
          {'name':'IzharHab', 'id':26}, {'name':'IzharHak', 'id':27}, {'name':'IzharKho', 'id':28}]

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

3. Create TF records

In [63]:
```
# OPTIONAL IF RUNNING ON COLAB
ARCHIVE_FILES = os.path.join(paths['IMAGE_PATH'], 'archive.tar.gz')
if os.path.exists(ARCHIVE_FILES):
  !tar -zxvf {ARCHIVE_FILES}
```

In [64]:
```
if not os.path.exists(files['TF_RECORD_SCRIPT']):
    !git clone https://github.com/nicknochnack/GenerateTFRecord {paths['SCRIPTS_PATH']}
```

In [65]:
```
!pip install pytz
```

```
Requirement already satisfied: pytz in f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\lib\site-packages (20
21.3)

WARNING: You are using pip version 21.3.1; however, version 22.0.4 is available.
You should consider upgrading via the 'f:\deteksi tajwid\tensorflow object detection\tfodcourse\tfod\scripts\python.exe -m
pip install --upgrade pip' command.
```

In [66]:
```
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'train')} -l {files['LABELMAP']} -o {os.path.join(
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'test')} -l {files['LABELMAP']} -o {os.path.join(p
```

```
Successfully created the TFRecord file: Tensorflow\workspace\annotations\train.record
Successfully created the TFRecord file: Tensorflow\workspace\annotations\test.record
```

4. Copy Model Config to Training Folder

In [67]:
```
if os.name =='posix':
    !cp {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKP
if os.name == 'nt':
    !copy {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHEC
```

```
        1 file(s) copied.
```

5. Update Config For Transfer Learning

In [68]:
```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
```

In [69]:
```
config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
```

In [70]:
```
config
```

Out[70]:
```
{'model': ssd {
   num_classes: 90
   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 {
```

In [71]:
```
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)
```

In [72]:
```
pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'che
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'train.rec
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'test.re
```

In [73]:
```
config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:
    f.write(config_text)
```

6. Train the model

In [74]:
```
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'model_main_tf2.py')
```

In [75]:
```
command = "python {} --model_dir={} --pipeline_config_path={} --num_train_steps=50000".format(TRAINING_SCRIPT, paths['CHECKP
```

In [76]:
```
print(command)
```

```
python Tensorflow\models\research\object_detection\model_main_tf2.py --model_dir=Tensorflow\workspace\models\my_ssd_mobnet
_detec --pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet_detec\pipeline.config --num_train_steps=50000
```

In [ ]:
```
!{command}
```

7. Evaluate the Model

In [77]:
```
command = "python {} --model_dir={} --pipeline_config_path={} --checkpoint_dir={}".format(TRAINING_SCRIPT, paths['CHECKPOINT
```

In [78]:
```
print(command)
```

```
python Tensorflow\models\research\object_detection\model_main_tf2.py --model_dir=Tensorflow\workspace\models\my_ssd_mobnet
_detec --pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet_detec\pipeline.config --checkpoint_dir=Tensorflow\
workspace\models\my_ssd_mobnet_detec
```

8. Load Train Model From Checkpoint

In [79]:
```
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
```

In [80]:
```
# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-51')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
```

9. Detect from an Image

In [81]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

In [82]:
```
category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
```

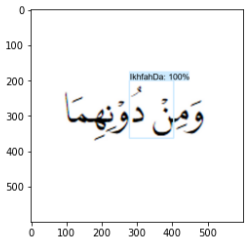
In [87]:
```
IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'dataset testing', '3.JPG')
```

In [88]:
```
IMAGE_PATH
```

Out[88]:
```
'Tensorflow\\workspace\\images\\dataset testing\\3.JPG'
```

In [89]:
```
img = cv2.imread(IMAGE_PATH)
image_np = np.array(img)

input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}
detections['num_detections'] = num_detections

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

label_id_offset = 1
image_np_with_detections = image_np.copy()
```

```
viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            detections['detection_boxes'],
            detections['detection_classes']+label_id_offset,
            detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=5,
            min_score_thresh=.8,
            agnostic_mode=False)

plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```



RIWAYAT HIDUP

MEGA ELLYADI, Dilahirkan di Banda Aceh Provinsi Aceh pada tanggal 02 November 2000. Anak kedua dari tiga bersaudara pasangan dari Iswadi dan Erlinawati. Penulis menyelesaikan Pendidikan di Sekolah Dasar di SD 10 Negeri Banda aceh dengan tahun lulus 2012, kemudian lanjut Pendidikan dijenjang Sekolah Menengah Pertama di SMP Negeri 14 Banda Aceh dengan tahun lulus 2015, kemudian melanjutkan pedidikan jenjang Sekolah Menengah Atas di SMA Negeri 11 Banda Aceh dengan tahun lulus 2018. Pada tahun 2018 penulis melanjutkan pendidikan Strata-1 (S1) di Perguruan Tinggi Negeri, tepatnya di Universitas Negeri Ar-Raniry Fakultas Sains dan Teknologi pada Program Studi Teknologi Informasi.