TUGAS AKHIR – EC184801

DETEKSI HURUF PADA TULISAN TANGAN LATIN MENGGUNAKAN METODE YOU ONLY LOOK ONCE (YOLO)

Aisyah Nurul Hidayah
NRP 07211640000053

Dosen Pembimbing
Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M.Sc.
Dr. Eko Mulyanto Yuniarno, ST., MT.

DEPATERMEN TEKNIK KOMPUTER
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – EC184801

DETEKSI HURUF PADA TULISAN TANGAN LATIN MENGGUNAKAN METODE YOU ONLY LOOK ONCE (YOLO)

Aisyah Nurul Hidayah
NRP 07211640000053

Dosen Pembimbing
Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M.Sc.
Dr. Eko Mulyanto Yuniarno, ST., MT.

DEPATERMEN TEKNIK KOMPUTER
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

FINAL PROJECT – EC184801

LETTERS DETECTION IN LATIN HANDWRITING USING YOU ONLY LOOK ONCE (YOLO)

Aisyah Nurul Hidayah
NRP 07211640000053

Advisors
Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M.Sc.
Dr. Eko Mulyanto Yuniarno, ST., MT.

Departement of Computer Engineering
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2020

Halaman ini sengaja dikosongkan

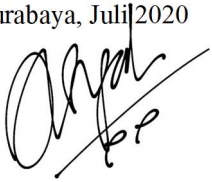
PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Deteksi Huruf pada Tulisan Tangan Latin Menggunakan Metode *You Only Look Once* (YOLO)**" adalah benar-benar hasil karya intelektual sendiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya orang lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, Juli 2020



Aisyah Nurul Hidayah
NRP. 07211640000053

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

DETEKSI HURUF PADA TULISAN TANGAN LATIN MENGGUNAKAN METODE *YOU ONLY LOOK ONCE* (YOLO)

Tugas Akhir ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Teknik di Institut Teknologi Sepuluh Nopember Surabaya

Oleh: Aisyah Nurul Hidayah (NRP: 07211640000053)

Tanggal Ujian : 8 Juli 2020 Periode Wisuda : September 2020

Disetujui oleh:

Prof. Dr. Ir. Yoyon K. Suprapto, M. Sc (Pembimbing I)
NIP. 195409251978031001

Dr. Eko Mulyanto Yuniarno, ST., MT. (Pembimbing II)
NIP. 196806011995121009

Dr. Diah Puspito Wulandari, ST., M. Sc (Penguji I)
NIP. 198012192005012001

Dr. Supeno Mardi Susiki Nugroho, ST., MT. (Penguji II)
NIP. 197003131995121001

Ahmad Zaini, ST., M. Sc. (Penguji III)
NIP. 197504192002121003

Mengetahui
Kepala Departemen Teknik Komputer

Dr. Supeno Mardi Susiki Nugroho, ST., MT.
NIP. 197003131995121001

Halaman ini sengaja dikosongkan

ABSTRAK

Nama Mahasiswa	:	Aisyah Nurul Hidayah
Judul Tugas Akhir	:	*Deteksi Huruf pada Tulisan Tangan Latin Menggunakan Metode You Only Look Once (YOLO)*
Pembimbing	:	1. Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M.Sc. 2. Dr. Eko Mulyanto Yuniarno, ST., MT.

Tulisan tangan saat ini masih terlibat dalam berbagai bidang, khususnya tulisan tangan latin. Tulisan tangan latin memiliki beberapa tantangan. Pertama, tulisan tangan memiliki banyak variasi. Variasi yang dimaksud adalah ukuran huruf, kemiringan huruf, *baseline* atau garis dasar tulisan, tekanan penulisan, bahkan jarak antarhuruf dan antarkata tulisan [1]. Kedua, setiap huruf pada tulisan tangan latin menyambung menjadi satu. Tantangan ini menyebabkan proses pengenalan tulisan latin memakan waktu yang lebih lama. *You Only Look Once* (YOLO) digunakan untuk mendeteksi huruf pada tulisan tangan latin sebagai cara agar proses pengenalan tulisan menjadi lebih cepat. Percobaan ini dilakukan dengan mengumpulkan sampel citra dari 4 jenis tulisan tangan yang berbeda. Hasil percobaan menggunakan GPU NVIDIA GeForce RTX 1050 menunjukkan bahwa *processing time* per *image* untuk deteksi huruf pada citra satu kata ialah 0.0776 detik dengan nilai *threshold confidence score* 0.3.

Kata Kunci : Tulisan, deteksi, yolo

Halaman ini sengaja dikosongkan

ABSTRACT

Name	:	Aisyah Nurul Hidayah
Title	:	*Letters Detection in Latin Handwriting Using You Only Look Once (YOLO)*
Advisors	:	1. Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M.Sc. 2. Dr. Eko Mulyanto Yuniarno, ST., MT.

Handwriting is currently still involved in various fields, specifically Latin handwriting. Latin handwriting has several challenges. First, handwriting has many variations. Variations discussed are letter size, the slope of the letter, baseline or the baseline of writing, even the different distance between letters and the words [1]. Second, each letter in Latin handwriting links together. This challenge led to a longer time for the recognition of Latin handwriting. You Only Look Once (YOLO) is used to detect letters in Latin handwriting as a way to make the recognition process of Latin handwriting faster. This experiment carried out by collecting sample images of 4 different handwritten types. The experimental result using GPU NVIDIA GeForce RTX 1050 shows processing time per image for the detection system in one-word images is 0.0776 seconds using a threshold score of 0.3.

Keywords : *Handwriting, detection, yolo*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur kehadirat Tuhan Yang Maha Esa atas segala karunia-Nya, penulis dapat menyelesaikan penelitian ini dengan judul **Deteksi Huruf pada Tulisan Tangan Latin Menggunakan Metode *You Only Look Once* (YOLO)**.

Penelitian ini disusun dalam rangka pemenuhan bidang riset di Departemen Teknik Komputer ITS, Bidang Studi Telematika, serta digunakan sebagai persyaratan menyelesaikan pendidikan S1. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa
2. Kedua orang tua, Bapak Sukaryani dan Ibu Mudrikah yang selalu memberikan dukungan baik material dan spiritual dalam mengerjakan penelitian ini
3. Bapak Kepala Departemen Teknik Komputer Dr. Supeno Mardi Susiki Nugroho, ST., MT
4. Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M.Sc. dan Dr. Eko Mulyanto Yuniarno, ST., MT. selaku dosen pembimbing, atas dukungan dan bimbingan selama mengerjakan penelitian ini.
5. Bapak dan Ibu dosen pengajar serta staff Departemen Teknik Komputer atas pengajaran, bimbingan, serta perhatian yang diberikan kepada penulis selama ini
6. Seluruh teman-teman Asisten Laboratorium B401, Tekkom 2016, dan Angkatan 18 Smudama (Antarctic) yang senantiasa memberikan dukungan dan motivasi dalam pelaksanaan penelitian tugas akhir ini
7. Teman-teman YOLO yang sudah bersama-sama belajar metode objek deteksi YOLO dari awal

Penulis memohon segenap kritik dan saran yang membangun. Harapannya penelitian ini dapat berguna sebagai acuan penelitian-penelitian selanjutnya dan bermanfaat bagi kita semua.

Surabaya, Juli 2020

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

NOMENKLATUR

S	: Ukuran *feature map*
B	: Jumlah *bounding box* yang sebenarnya
$\hat{B}$	: Jumlah *bounding box* yang terprediksi
C	: *Confidence score* suatu *bounding box* yang sebenarnya
$\hat{C}$	: *Confidence score* suatu *bounding box* yang terprediksi
c	: *class* yang sebenarnya
$\hat{c}$	: *class* yang terprediksi
$p(c)$	: Probabilitas bersyarat suatu *class* c yang sebenarnya
$\hat{p}(c)$	: Probabilitas bersyarat suatu *class* c yang terprediksi
w	: Lebar suatu *bounding box* yang sebenarnya
$\hat{w}$	: Lebar suatu *bounding box* yang terprediksi
h	: Tinggi suatu *bounding box* yang sebenarnya
$\hat{h}$	: Tinggi suatu *bounding box* yang terprediksi
x	: Nilai x suatu *bounding box* yang sebenarnya
$\hat{x}$	: Nilai x suatu *bounding box* yang terprediksi
y	: Nilai y suatu *bounding box* yang sebenarnya
$\hat{y}$	: Nilai y suatu *bounding box* yang terprediksi
$p_{interp}(r)$	: Nilai *interpolated precision* terhadap *recall*
$\tilde{r}$	: Nilai *recall* sebenarnya

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

Penelitian ini di latar belakangi oleh berbagai kondisi yang menjadi acuan. Selain itu juga terdapat beberapa permasalahan yang akan dijawab sebagai luaran dari penelitian.

1.1 Latar Belakang

Tulisan merupakan salah satu bentuk komunikasi. Dengan adanya tulisan, manusia jadi mampu untuk menyampaikan idenya tanpa harus berada di tempat yang sama dengan pembacanya. Dari situlah kemudian tulisan tangan semakin berkembang dan mengalami perkembangan yang unik di setiap budaya di dunia.

Sejarah perkembangan tulisan sendiri dimulai dari masyarakat primitif, dimana mereka mulai mengembangan tulisan dalam bentuk coretan di dinding-dinding goa, kayu, maupun batu. Lama kelamaan coretan-coretan tersebut berkembang menjadi simbol-simbol dan akhirnya menjadi alfabet/huruf. Huruf alfabet lalu menyebar ke seluruh negara sehingga memunculkan variasi huruf Alfabet yang berbeda, misalnya di alfabet Yunani, alfabet Rusia, hingga alfabet Latin. Khusus di Eropa juga berkembang tulisan sambung atau *cursive handwriting* yang saat ini dikenal tulisan tangan latin yang juga menyebar ke seluruh dunia.

Tulisan tangan hingga saat ini masih terlibat dalam berbagai bidang salah satunya pada jasa pengiriman. Menurut Badan Pusat Statistik (BPS), pengiriman surat oleh Pos Indonesia divisi Regional Jawa Timur pada tahun 2015 mencapai 22.200.517 [7] dan paket sebanyak 627.922 [8]. Angka tersebut diperkirakan akan meningkat pada tahun-tahun berikutnya.

Tulisan tangan khususnya tulisan tangan latin memiliki beberapa tantangan. Tantangan tersebut diantaranya banyaknya variasi yang ditulis oleh setiap orang. Variasi yang dimaksud adalah ukuran huruf yang besar atau kecil, kemiringan huruf dan kata yang cenderung ke kiri, kanan maupun tidak sama sekali, *baseline* atau garis dasar tulisan baik itu rata atau cenderung bergelombang, tekanan penulisan yang kuat atau lemah menghasilkan tebal tipisnya tulis-

an, bahkan jarak antarhuruf dan antarkata tulisan [1]. Tantangan lain dari tulisan tangan latin yakni setiap huruf pada tulisan tangan latin menyambung menjadi satu.

1.2 Perumusan Masalah

Kedua tantangan tersebut menyebabkan proses pengenalan tulisan tangan latin membutuhkan waktu yang lebih lama. Selain itu, *Optical Character Recognition* (OCR) yang sudah ada saat ini masih belum mendukung pengenalan tulisan tangan latin. Oleh karena itu, diperlukan sebuah sistem yang dapat mendeteksi huruf pada tulisan tangan latin sebagai cara agar proses pengenalan tulisan tangan latin menjadi lebih cepat.

1.3 Tujuan

Berdasarkan perumusan masalah yang telah dijabarkan, tujuan dari tugas akhir ini adalah untuk membuat sistem pendeteksi huruf pada tulisan tangan latin menggunakan *You Only Look Once* (YOLO). *You Only Look Once* (YOLO) digunakan karena YOLO dapat mendeteksi objek dengan cepat.

1.4 Batasan Masalah

Batasan masalah yang timbul dari permasalahan Tugas Akhir ini adalah tulisan tangan yang digunakan adalah tulisan tangan latin.

1.5 Sistematika Penulisan

Laporan penelitian Tugas akhir ini tersusun dalam sistematika dan terstruktur sehingga mudah dipahami dan dipelajari oleh pembaca maupun seseorang yang ingin melanjutkan penelitian ini. Alur sistematika penulisan laporan penelitian ini yaitu:

1. BAB I Pendahuluan
   Bab ini berisi uraian tentang latar belakang permasalahan, penegasan dan alasan pemilihan judul, sistematika laporan, tujuan dan metodologi penelitian.
2. BAB II Dasar Teori
   Pada bab ini berisi tentang uraian secara sistematis teori-teori yang berhubungan dengan permasalahan yang dibahas pada

pengerjaan tugas akhir ini. Teori-teori ini digunakan sebagai dasar dalam pengerjaan, yaitu teori yang berkaitan dengan *object detection* pada *image* dan *You Only Look Once* (YOLO) sebagai pengembangan dari *Convolutional Neural Network* (CNN).

3. BAB III Perancangan Sistem dan Impementasi
   Bab ini berisi tentang penjelasan-penjelasan terkait sistem yang akan dibuat. Guna mendukung itu digunakanlah blok diagram atau *work flow* agar sistem yang akan dibuat dapat terlihat dan mudah dibaca untuk implentasi pada pelaksanaan tugas akhir.
4. BAB IV Pengujian dan Analisa
   Bab ini menjelaskan tentang pengujian yang dilakukan terhadap proses pengambilan data dari *image* serta menganalisa sistem tersebut.
5. BAB V Penutup
   Bab ini merupakan penutup yang berisi kesimpulan yang diambil dari penelitian dan pengujian yang telah dilakukan. Saran dan kritik yang membangun untuk pengembangkan lebih lanjut juga dituliskan pada bab ini.

Halaman ini sengaja dikosongkan

BAB 2
TINJAUAN PUSTAKA

Demi mendukung penelitian ini, dibutuhkan beberapa teori penunjang sebagai bahan acuan dan referensi. Dengan demikian penelitian ini menjadi lebih terarah.

2.1 *Related Works*

Pada sub bab *related works* dijelaskan mengenai beberapa penelitian yang berelasi dengan Tugas Akhir ini.

2.1.1 *Related Works* Terkait dengan Tulisan Tangan

1. Penelitian yang berjudul "*Handwritten Text Segmentation using Pixel Based Approach*" [9] membahas metode *Pixel-Wise Approach* dan *Bounding Box Approach* dalam segmentasi tulisan tangan. Metode *Pixel-Wise Approach* digunakan untuk menyegmentasi tulisan tangan yang saling menyambung. Metode *Bounding Box Approach* digunakan untuk menyegmentasi tulisan tangan yang tidak menyambung. Fitur dari tulisan yang telah disegmentasi kemudian diekstrak (*feature extraction*) sehingga dapat diklasifikasi dan dikenali.
2. Penelitian yang berjudul "*Word Segmentation Method for Handwritten Documents based on Structured Learning*" [1]. Percobaan algoritma ini dilakukan pada database ICDAR 2009 dan ICDAR 2013 dimana pada database tersebut terdapat *script* tulisan Latin dan *script* tulisan India/Bangla. Dari percobaan yang diusulkan, akurasi rata-rata yang diperoleh adalah 92.82%.
3. Penelitian yang berjudul "Pengenalan Tulisan Tangan Naskah Aksara Jawa Aturan Penulisan Mardi Kawi Menjadi Huruf Alfabet Pada Media Kertas Menggunakan Metode *Convolutional Neural Network* (CNN) Arsitektur *You Only Look Once*" [10]. Penelitian ini menggunakan YOLO untuk mendeteksi tulisan tangan dari 5 judul dan penulis berbeda pada naskah aksara Jawa Mardikawi. Pengujian menggunakan 33 jenis aksara Jawa Mardikawi menghasilkan rata-rata IoU sebesar 46%, se-

dangkan pengujian menggunakan 44 jenis aksara Jawa Mardikawi menghasilkan 40% masing-masing menggunakan *dataset* ukuran asli kertas naskah.

4. Penelitian yang berjudul "*TextCaps : Handwritten Character Recognition with Very Small Datasets*" [11] membahas mengenai pengenalan tulisan tangan dengan menggunakan *dataset* yang relatif kecil. Sistem ini berguna dalam pengenalan karakter untuk bahasa lokal yang tidak memiliki banyak *dataset*. *Dataset* ini diaugmentasi sehingga didapatkan variasi yang lebih banyak. Augmentasi ini realistis, mencerminkan variasi aktual yang ada dalam tulisan tangan manusia dengan menambahkan *noise* yang dikendalikan secara acak. Pengenalan karakter menggunakan arsitektur yang terdiri dari *capsule network* dan *decoder network*. Hasil yang didapatkan melampaui hasil pengenalan karakter yang ada di *dataset* EMNIST-*letter*.

2.1.2 *Related Works* Terkait dengan YOLO *Object Detection*

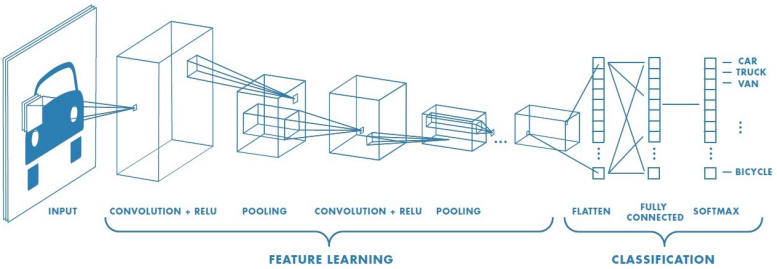
1. Penelitian yang berjudul "*Real-Time Face Detection based on YOLO*" [12] membahas pendeteksian wajah menggunakan *network* YOLOv3. Pengujian menggunakan WIDER FACE *dataset* menunjukkan hasil yang lebih cepat dengan waktu deteksi rata-rata 0.028 detik. Selain lebih cepat, hasil yang didapatkan juga lebih akurat. Kecepatan deteksi dapat memenuhi persyaratan *real-time* dan mendapatkan hasil yang baik.
2. Penelitian yang berjudul "*Thermal Object Detection in Difficult Weather Conditions Using YOLO*" [13] bertujuan untuk mendeteksi orang-orang dalam citra dan video termal yang diambil dalam cuaca berbeda (cuaca cerah, hujan lebat, dan kabut) dan kondisi perekaman selama pengawasan area yang dilindungi, menggunakan *real-time object detector*. *Object detector* yang digunakan dalam pengujian diantaranya Faster RCNN, SSD, FCOS, Cascade RCNN, dan YOLOv3. Berdasarkan hasil pengujian, YOLOv3 memiliki kecepatan pemrosesan tertinggi, yakni 27.472 FPS dengan AP sebesar 97.93%.
3. Penelitian yang berjudul "*A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector*" [14] membahas mengenai sistem ALPR (*Automatic License Plate*

Recognition) yang kuat dan efisien berdasarkan pada detektor objek YOLO yang canggih. Deteksi pada sistem ada dua jenis, yakni deteksi kendaraan dan nomor plat. Bagian kendaraan yang memiliki plat nomor dideteksi dan dipotong. Hasil berupa citra yang dipotong kemudian dideteksi nomor platnya. Nomor plat yang dihasilkan dipotong dan disegmentasi per karakter. Karakter hasil proses segmentasi kemudian dikenali. Hasil pengenalan nomor plat akhir merupakan nomor plat yang paling sering diprediksi untuk setiap posisi nomor plat (*majority votes*). Pada *dataset* SSIG didapatkan akurasi sebesar 93.53% dengan textittemporal redudancy dan 47 *Frames per Second* FPS. Sedangkan pada *dataset* UFPR-ALPR, *dataset* publik yang lebih besar, didapatkan akurasi sebesar 78.33% dengan *temporal redudancy* dan 35 FPS.

2.2 *Convolutional Neural Network* (CNN)

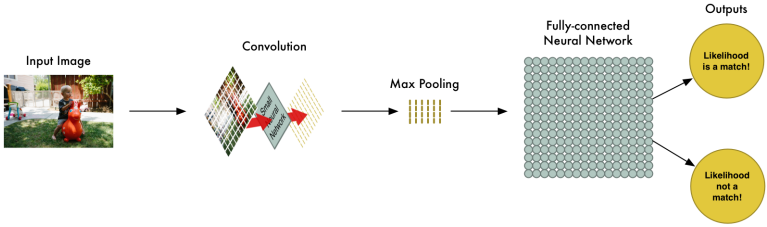
Convolutional Neural Network (ConvNet/CNN) adalah algoritma *deep learning* yang dapat mengambil masukan berupa citra, menetapkan kepentingan (bobot dan bias yang dapat dipelajari) untuk berbagai aspek/objek dalam citra dan dapat membedakan satu dari yang lain. Secara garis besar CNN tidak jauh beda dengan *neural network* biasanya. CNN terdiri dari neuron yang memiliki *weight*, *bias* dan *activation function*. CNN terdiri dari dua lapisan yakni *feature learning* dan *classification*. Lapisan yang terdapat pada CNN ditampilkan pada Gambar 2.1. Lapisan *feature learning* terdiri dari *convolutional layer* dan *pooling layer* sedangkan lapisan *classification* terdiri dari *flatten*, *fully-connected layer*, dan *softmax*.

Cara kerja CNN adalah sebagai berikut. Masukan berupa citra masuk ke dalam *convolutional layer* dan mengalami proses konvolusi dengan menggerakan sebuah *filter* berukuran tertentu ke sebuah citra sehingga didapatkan informasi representatif baru dari hasil perkalian bagian citra tersebut dengan *filter* yang digunakan. Hasil dari proses konvolusi adalah masukan sebelumnya yang telah dibagi-bagi menjadi citra yang lebih kecil. Setiap citra kecil dari hasil konvolusi tersebut kemudian dijadikan masukan untuk *neural network* yang lebih kecil. Proses ini dilakukan untuk semua bagian dari masing-masing citra kecilnya, dengan menggunakan *filter*



Gambar 2.1: Lapisan yang terdapat pada *Convolutional Neural Network* [2]

yang sama. Keluaran dari setiap *neural network* yang lebih kecil kemudian disimpan dalam *array* baru. Karena *array* masih terlalu besar, maka untuk mengecilkan ukuran *array* digunakan *downsampling* yang penggunaannya dinamakan *max pooling* atau mengambil nilai *pixel* terbesar di setiap *pooling kernel*. Dengan begitu, sekalipun mengurangi jumlah parameter, informasi terpenting dari bagian tersebut tetap diambil. *Array* kecil kemudian dimasukkan ke dalam jaringan saraf lain atau *fully-connected layer*. Jaringan ini akan memutuskan apakah citranya cocok atau tidak. Ilustrasi cara kerja *convolutional neural network* ditunjukkan pada Gambar 2.2.



Gambar 2.2: Lapisan yang terdapat pada *Convolutional Neural Network* [3]

2.3 *You Only Look Once* (YOLO)

You Only Look Once atau YOLO merupakan salah satu arsitektur dari *Convolutional Neural Network* untuk mendeteksi objek pada citra. Arsitektur YOLO sangat cepat dibandingkan arsitektur

object detector yang lain [4] dengan menerapkan sebuah jaringan syaraf tunggal pada keseluruhan citra. YOLO mendeteksi sebuah objek dalam beberapa tahap yaitu:

1. Membagi citra ke dalam *region/grid* berukuran $S \times S$ dengan S merupakan ukuran *feature map*. *Grid-grid* tersebut bertanggung jawab untuk mendeteksi objek dengan cara melihat apakah *center* sebuah objek jatuh ke dalam *grid cell*. Pada tiap *grid* juga akan diprediksi *bounding box* beserta *confidence score*. *Confidence score* ini menunjukkan seberapa yakin *bounding box* tersebut berisi objek dan seberapa akurat prediksinya. *Confidence score* diperoleh dengan fungsi yang ditunjukkan pada Persamaan 2.1.

$$\hat{C} = Pr(Object) \times IoU^{truth}_{predict} \tag{2.1}$$

   dimana $\hat{C}$ merupakan *confidence score* yang terprediksi, $Pr(Object)$ probabilitas objek yang muncul dalam suatu *region*, dan $IoU^{truth}_{predict}$ adalah rasio tumpang tindih (*Intersection Over Union*) antara *predicted box* dan *ground-truth box*.

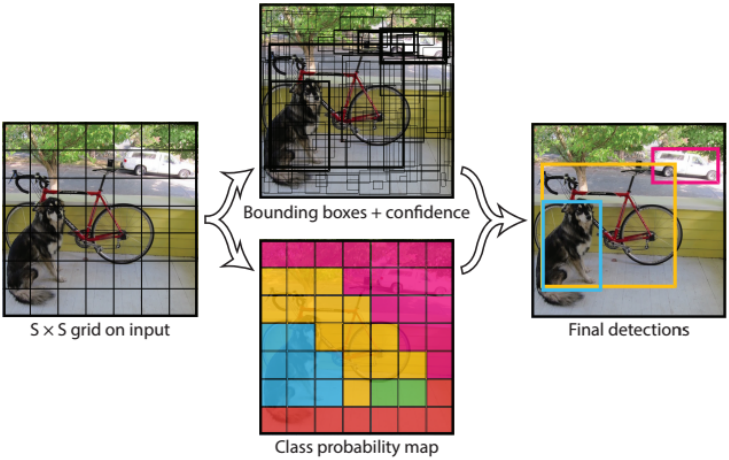
2. Setiap *bouding box* terdiri dari 5 parameter prediksi, yakni x, y, w, h, dan *confidence score*. Nilai x dan y adalah koordinat titik tengah *bounding box*, nilai w dan h adalah rasio ukuran lebar dan tinggi relatif terhadap *grid*, dan c adalah *confidence score* dari *bounding box* tersebut.

3. Pada algoritma YOLO, tiap *grid* akan memprediksi nilai *class probability* jika diprediksi terdapat objek di dalamnya. Saat pengujian, YOLO akan mengkalikan nilai *class probability* dengan *confidence score* dari *bounding box* seperti pada Persamaan 2.2.

$$Pr(Class_i|Object) \times Pr(Object) \times IoU^{truth}_{predict} = Pr(Class_i) \times IoU^{truth}_{predict} \tag{2.2}$$

   dimana $Pr(Class_i|Object)$ merupakan probabilitas objek milik $Class_i$ dengan syarat objek disajikan dan $Pr(Class_i)$ merupakan probabilitas objek milik $Class_i$. Sehingga mengha-

silkan *confidence score* suatu *class* secara spesifik pada tiap *bounding box*. Nilai ini menunjukkan *class probability* yang muncul pada *bounding box* dan seberapa akurat *bounding box* memprediksi sesuai dengan objek. Ilustrasi algoritma YOLO ditampilkan pada Gambar 2.3.

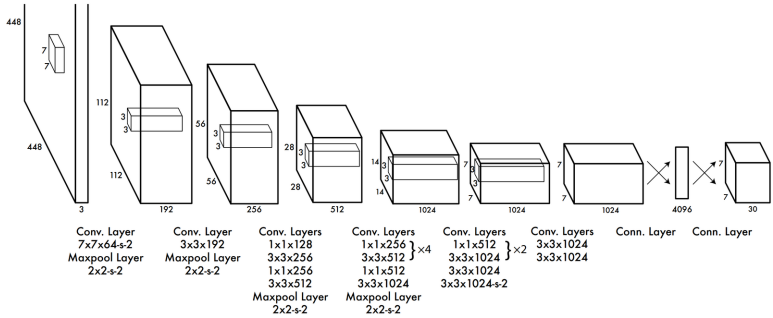


Gambar 2.3: Ilustrasi algoritma YOLO dalam mendeteksi objek [4]

Arsitektur YOLO sendiri terinspirasi dari model GoogleNet untuk klasifikasi citra. YOLO memiliki 24 *convolutional layers* yang diikuti dengan 2 *fully connected layers* seperti pada Gambar 2.4. YOLO mengunakan 1×1 *convolutional layer* (untuk integrasi antar channel) dan 3×3 *convolutional layer* sebagai pengganti permulaan modul.

Selain itu, YOLO menggunakan *linear activation function* untuk *final layer* dan *layer* lainnya menggunakan fungsi aktivasi Leaky ReLU seperti ditunjukkan pada Persamaan 2.3.

$$\phi(x) = \begin{cases} x, & \text{jika } x > 0 \\ 0.1x, & \textit{otherwise} \end{cases} \tag{2.3}$$



Gambar 2.4: Arsitektur jaringan YOLOv1 [4]

Proses paling utama dalam algoritma YOLO ada dua, yaitu *Convolution* dan *Max Pooling*. *Convolution* adalah sebuah proses dimana citra dimanipulasi dengan menggunakan eksternal *mask/subwindows* untuk menghasilkan citra yang baru. Proses konvolusi ini menggunakan kernel 3×3. *Max Pooling* adalah proses mereduksi masukan secara spasial (mengurangi jumlah parameter) dengan operasi *down-sampling* dengan mengambil nilai terbesar dari bagian tersebut. *Max Pooling* dalam YOLO menggunakan kernel 2×2 *filters* dan *stride* 2, artinya setiap matriks akan selalu terbagi menjadi setengahnya (416×416 menjadi 208×208 dst). Kedua proses ini akan terus terulang sampai menghasilkan keluaran berupa *grid cell*. Bentuk keluaran berupa *grid cell*/*tensor* sesuai persamaan $S \times S(B \times 5 + C)$ dimana S adalah jumlah *grid cell* atau ukuran *feature map*, B adalah jumlah *bounding box*, dan C adalah jumlah *class*. Misalnya, *class* yang digunakan sebanyak 20 *class*, jumlah *grid cell* sebanyak 13, dan setiap *grid cell* memprediksi 5 *bounding box* maka untuk prediksi akan digunakan $13 \times 13 \times 125$ *tensor*.

Pada proses prediksi, YOLO membagi citra menjadi *grid cell* menjadi ukuran 13×13. Untuk setiap *bounding box*, *cell* juga memprediksi *class*. Hal ini berfungsi sebagai klasifikasi. *Confidence score* untuk *bounding box* dan prediksi *class* digabungkan menjadi satu hasil akhir yang memberi tahu kemungkinan bahwa *bounding box* ini berisi jenis objek tertentu. Karena ada $13 \times 13 = 169$ *cell* dan masing-masing *cell* memprediksi 5 *bounding box*, jadi total ada 845 *bounding box*. Itu artinya ada sebagian kotak yang mempunyai

confidence score rendah. Oleh karena itu, YOLO hanya menyimpan dan menampilkan kotak yang mempunyai *confidence score* sama dengan atau lebih besar dari nilai *threshold*. Proses ini disebut *non-max suppression*.

Sum-squared error digunakan untuk mengoptimasi pada keluaran berupa model. *Sum-squared error* digunakan karena sangat mudah dioptimasi, namun tidak maksimal pada nilai *average precision*. Hal ini dapat menyebabkan ketidakstabilan model, menyebabkan pelatihan menyimpang sejak awal. Untuk memperbaiki hal ini, *loss* dari prediksi koordinat *bounding box* dinaikkan dan menurunkan *loss* dari nilai prediksi untuk *box* yang tidak mengandung objek. Digunakan $\lambda_{coord} = 5$ dan $\lambda_{noobj} = .5$ untuk menyelesaikan ini.

Dalam proses *training*, *multi-part loss function* dioptimasi sesuai fungsi pada Persamaan 2.4.

$$\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] + \\
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] + \\
&\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \\
&\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 + \\
&\sum_{i=0}^{S^2} 1_{i}^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned} \tag{2.4}$$

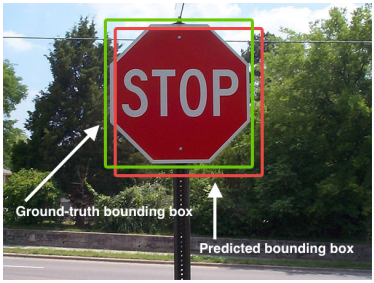
dimana 1_i^{obj} menunjukkan jika objek muncul pada *cell* i dan 1_{ij}^{obj} menunjukkan bahwa *bounding box predictor* ke-j pada *cell* i bertanggung jawab atas prediksi tersebut. (x_i, y_i) merupakan koordinat objek yang sebenarnya (*ground truth*) dan $(\hat{x}_i, \hat{y}_i)$ merupakan koordinat *bounding box* yang terprediksi. (w_i, h_i) merupakan lebar

dan tinggi *bounding box* yang sebenarnya (*ground truth*) dan $(\hat{w}_i, \hat{h}_i)$ merupakan lebar dan tinggi *bounding box* yang terprediksi. C_i merupakan *confidence score* suatu *bounding box* yang sebenarnya dan $\hat{C}_i$ merupakan *confidence score* suatu *bounding box* yang terprediksi. $p(c)$ merupakan probabilitas bersyarat suatu *class* c yang sebenarnya dan $\hat{p}(c)$ merupakan probabilitas bersyarat suatu *class* c yang terprediksi.

YOLO sendiri memiliki beberapa keuntungan diantaranya proses komputasinya sangat cepat. Hal ini dikarenakan YOLO hanya menjalankan satu *neural network* untuk memprediksi deteksi. Meskipun begitu, *mean average precision* (mAP) yang didapatkan YOLO jauh lebih baik dari *object detector* lainnya. Selain itu, YOLO menimbang-nimbang secara luas ketika membuat prediksi terhadap citra. YOLO melihat citra secara keseluruhan dan membuat informasi mengenai *class* yang sesuai dengan bentuknya. Keuntungan lainnya yakni YOLO mempelajari representasi objek yang dapat digeneralisasikan sehingga kecil kemungkinannya untuk *error* ketika diterapkan pada domain baru atau masukan yang tidak terduga.

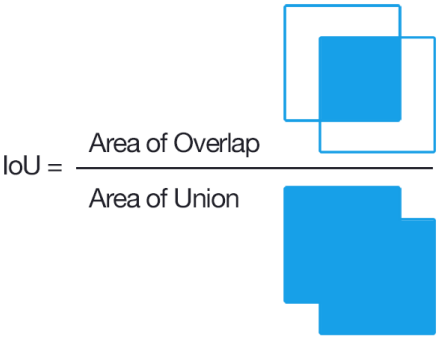
2.4 *Intersection over Union* (IoU)

Intersection over Union (IoU) didefinisikan sebagai rasio dari area yang tumpang tindih dan area yang berpotongan [5]. Dalam menentukan nilai IoU suatu *bounding box*, dibutuhkan *ground-truth* bounding box dan *predicted box*. Contoh visualisasi dari *ground-truth box* versus *predicted box* yang diprediksi dapat dilihat pada Gambar 2.5. Kotak hijau merupakan *ground-truth box* sedangkan kotak merah merupakan *predicted box*.



Gambar 2.5: Contoh deteksi tanda berhenti pada citra [5]

Persamaan IoU dapat dihitung seperti Gambar 2.6. Nilai IoU didapatkan dari pembagian nilai area yang *overlap* terhadap bagian union antara predicted bounding box dengan ground-truth-nya.



Gambar 2.6: Kalkulasi nilai IoU suatu *bounding box* [5]

Contoh perhitungan IoU untuk berbagai *bounding box* ditunjukkan pada Gambar 2.7. Nilai IoU yang lebih besar atau sama dengan 0.5 dianggap sebagai prediksi yang bagus. *Predicted box* yang sangat tumpah tindih dengan *ground-truth box* atau *heavily overlap* memiliki nilai IoU yang lebih besar dibandingkan *bounding box* yang *overlap*-nya kecil. Hal ini yang membuat *intersection over union* merupakan matriks yang baik untuk mengevaluasi *object detectors*.

Gambar 2.7: Contoh perhitungan IoU untuk berbagai *bounding box* [5]

2.5 *Confusion Matrix*

Confusion Matrix merupakan salah satu metrik yang populer untuk mengevaluasi dan mengukur *performance* dari suatu model *neural network* yang dihasilkan [15]. Matriks ini merepresentasikan prediksi dan kondisi sebenarnya dari data yang dihasilkan. Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat 4 (empat) istilah sebagai representasi hasil proses deteksi. Keempat istilah tersebut adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Tabel 2.1 merupakan tabel *confusion matrix*.

Tabel 2.1: Tabel penjelasan *Confusion Matrix*

		Predicted Value	
		Positive	Negative
Actual Value	True	TP	TN
	False	FP	FN

1. *True Positive*: IoU *predicted box* ≥ 0.5.
2. *True Negative*: Semua bagian pada citra yang tidak terprediksi sebagai objek. Matriks ini tidak berguna untuk *object detection*. Oleh karena itu, matriks ini diabaikan.
3. *False Positive*: IoU *predicted box* < 0.5.
4. *False Negative*: Objek yang memiliki *ground-truth box* dan model gagal dalam mendeteksi objek.

Pengukuran untuk mengevaluasi kinerja sistem yang terdapat pada *confusion matrix* diantaranya *accuracy, precision, recall, specificity, dan F-score)*. Pada tugas akhir ini, acuan performansi yang digunakan yakni rata-rata nilai *precision* dan *recall*.

1. *Accuracy*: Rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Nilai akurasi dapat dihitung sesuai Persamaan 2.5.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.5}$$

2. *Precision*: Rasio prediksi benar positif dibandingkan dengan

keseluruhan hasil yang diprediksi positif. Nilai presisi dapat dihitung sesuai Persamaan 2.6.

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

3. *Recall*: Rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Nilai *recall* dapat dihitung sesuai Persamaan 2.7.

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

4. *Specificity*: Kebenaran memprediksi negatif dibandingkan dengan keseluruhan data negatif. Nilai *specificity* dapat dihitung sesuai Persamaan 2.8.

$$Specificity = \frac{TN}{TN + FP} \tag{2.8}$$

5. *F-Score*: Perbandingan rata-rata presisi dan *recall* yang dibobotkan. Nilai *f-score* dapat dihitung sesuai Persamaan 2.9.

$$F - Score = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{2.9}$$

2.6 *mean Average Precision* (mAP)

Average precision (AP) adalah metrik populer yang digunakan dalam mengukur akurasi suatu *object detector* seperti *Faster* R-CNN, SSD, dll. *Average precision* akan menghitung nilai rata-rata nilai *precision* terhadap nilai *recall* dari 0 hingga 1. Nilai *precision* dan *recall* masing-masing dihitung sesuai Persamaan 2.6 dan 2.7. Kedua nilai ini nantinya digambarkan pada sebuah grafik. Grafik yang dibuat kemungkinan besar memiliki *noise*. Oleh karena itu, untuk mengurangi *noise* ini digunakan *interpolated precision* untuk mengurangi dampak *noise* yang disebabkan oleh variasi kecil.

Interpolated precision atau p_{interp}, dihitung pada setiap tingkat *recall* (r), dengan mengambil nilai *precision* maksimal yang di-

ukur untuk r. Fungsi *interpolated precision* ditunjukkan pada Persamaan 2.10.

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r}) \tag{2.10}$$

dimana $p(\tilde{r})$ adalah nilai *interpolated precision* yang telah dihitung terhadap *recall* $\tilde{r}$, $\tilde{r}$ merupakan nilai *recall* sebenarnya.

Misalkan, terdapat nilai *recall* dan nilai *precision* kemudian diurutkan berdasarkan nilai *recall*-nya seperti Tabel 2.2. Berdasarkan Tabel 2.2, nilai *recall* 0.33 memiliki nilai *precision* 1 dan 0.5. Pada kasus seperti ini, nilai *precision* tertinggi akan diambil sebagai nilai *interpolated precision*-nya. Oleh karena itu, nilai *interpolated precision* dari nilai *recall* 0.33 adalah 1. Nilai *precision* dan *interpolated precision* ditunjukkan pada Tabel 2.3.

Tabel 2.2: Tabel contoh nilai *precision* dan *recall*

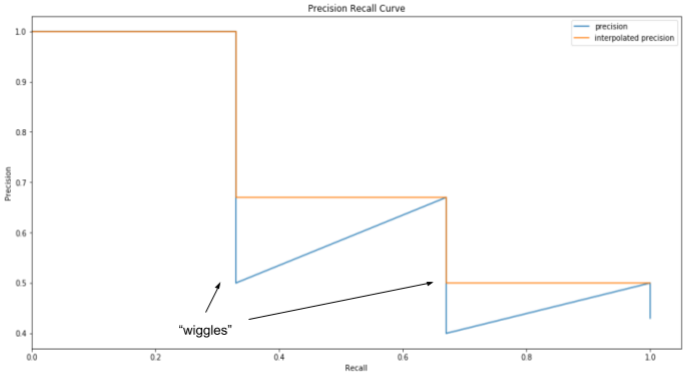
Recall	*Precision*
0.33	1
0.33	0.5
0.67	0.67
0.67	0.5
0.67	0.4
1	0.5
1	0.43

Tabel 2.3: Tabel contoh nilai *interpolated precision* dan *recall*

Recall	*Interpolated precision*
0.33	1
0.33	1
0.67	0.67
0.67	0.67
0.67	0.67
1	0.5
1	0.5

Nilai-nilai pada tabel digambarkan pada grafik seperti pada Gambar 2.8. Nilai AP kemudian ditentukan dengan menghitung area di bawah kurva. Perhitungan ini dilakukan dengan membagi nilai *recall* ke dalam 11 bagian (0, 0.1, 0.2,..., 0.9, 1) sesuai Persamaan 2.11.

$$
\begin{aligned}
AP &= \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots 0.9, 1\}} p_{interp}(r) \\
&= \frac{1}{11}(1 + 1 + 1 + 1 + 0.67 + 0.67 + 0.67 + 0.5 + 0.5 + 0.5 + 0.5) \\
&\approx 0.728
\end{aligned}
\tag{2.11}
$$



Gambar 2.8: Grafik *Precision-Recall* dari Tabel 2.2 dan 2.3 [6]

mAP untuk *object detection* merupakan rata-rata AP yang dihitung untuk semua *class*. Tools mAP yang dibuat oleh Cartucho [16] dapat dijadikan sebagai alternatif dalam menghitung mAP sebuah model.

BAB 3
DESAIN DAN IMPLEMENTASI SISTEM

Penelitian ini dilaksanakan sesuai dengan desain sistem juga dengan implementasinya. Desain sistem merupakan konsep dari pembuatan dan perancangan infrastruktur dan kemudian diwujudkan dalam bentuk blok-blok alur yang harus dikerjakan. Pada bagian implementasi merupakan pelaksanaan teknis untuk setiap blok pada desain sistem.

3.1 Desain Sistem

Tugas akhir ini merupakan penelitian dalam bidang visi komputer yang bertujuan untuk mendeteksi huruf pada tulisan tangan latin menggunakan *You Only Look Once* (YOLO). Diagram alur pada Gambar 3.1 merupakan gambaran dari sistem yang akan dibuat.

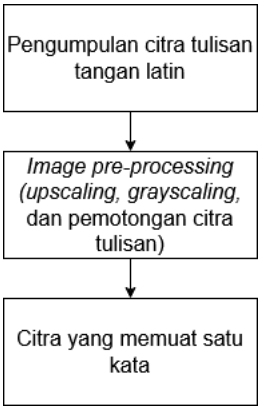
Gambar 3.1: Desain sistem keseluruhan

Berdasarkan Gambar 3.1, langkah awal yang dilakukan adalah akuisisi data, dengan mengumpulkan tulisan tangan latin berbagai kata dari berbagai penulis pada sebuah kertas. Kertas ini kemudian difoto sehingga didapatkan citra kertasnya. Citra kertas yang telah didapat kemudian di-*processing* berupa *upscaling* citra dan *grayscaling*. Citra yang memuat tulisan kemudian dipotong per kata sehingga satu citra memuat satu kata. Citra kata selanjutnya dibagi untuk *training sets* dan *testing sets* serta diberi anotasi sesuai dengan huruf. Ada 26 jenis huruf yang digunakan, mulai dari A sampai Z. Setelah *dataset* disiapkan, proses *training* dilakukan menggunakan *Darkflow*. Hasil *training* berupa sebuah model (*output weight*) dalam bentuk *meta file* dan *pb file* yang akan digunakan

dalam mendeteksi huruf pada *testing sets*. Tiap hasil deteksi huruf digabungkan sehingga kata pada citra dapat dikenali. Dari hasil deteksi maupun pembacaan dihitung akurasi dari model yang telah dihasilkan.

3.2 Akuisisi Data

Dalam tugas akhir ini, akuisisi data dilakukan dengan memfoto tulisan pada kertas menggunakan kamera. Pada Gambar 3.2 dijelaskan tahapan dalam proses akuisisi data.



Gambar 3.2: Alur akuisisi data

3.2.1 Pengumpulan Citra Tulisan Tangan Latin

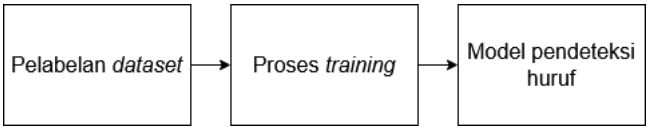
Proses pengumpulan tulisan tangan latin dimulai dari menyiapkan kombinasi kata yang akan ditulis. Kombinasi kata sangat berpengaruh terhadap variasi posisi huruf sehingga setiap huruf bisa berada pada posisi awal tengah, maupun akhir kata. Setiap orang menuliskan kembali kombinasi kata yang digunakan sebagai *dataset* pada sebuah kertas. Tulisan tangan latin dikumpulkan dari beberapa orang sehingga didapatkan beberapa lembar kertas dari penulis yang berbeda. Kertas yang berisi tulisan tangan latin kemudian diakuisisi dengan difoto menggunakan kamera atau di-*scan* menggunakan *printer scan* sehingga didapatkan citra kertasnya.

3.2.2 *Image Pre-processing*

Image pre-processing yang dilakukan yakni meningkatkan kualitas dan ukuran citra dengan *upscaling*, mengonversi citra menjadi *grayscale* dengan *grayscaling*, dan memotong citra tulisan menjadi citra yang memuat satu kata. Citra yang memiliki ukuran dibawah 2000 *pixel* akan di-*upscaling* sehingga ukurannya bisa diatas 4000 *pixel*. Citra yang telah di-*upscaling* kemudian di-*grayscaling*. Tujuan dari proses *grayscaling* ini untuk menyeragamkan warna tinta yang digunakan pada kertas sehingga dapat dibedakan fitur dan *background*. Proses ini dilakukan menggunakan *software photo editor* seperti Adobe Photoshop. Citra tulisan tangan latin kemudian dipotong-potong sehingga satu citra hanya memuat satu kata. Pemotongan citra dilakukan menggunakan *software* Microsoft Photos. Citra kata kemudian dibagi menjadi dua bagian, yakni *training sets* dan *testing sets* dengan rasio masing-masing sebesar 70% dan 30%.

3.3 *Training* Data

Training data dilakukan setelah proses pengumpulan citra tulisan tangan dan *image pre-processing*. Terdapat beberapa tahap dalam *proses* ini yang digambarkan dalam blok diagram pada Gambar 3.3.

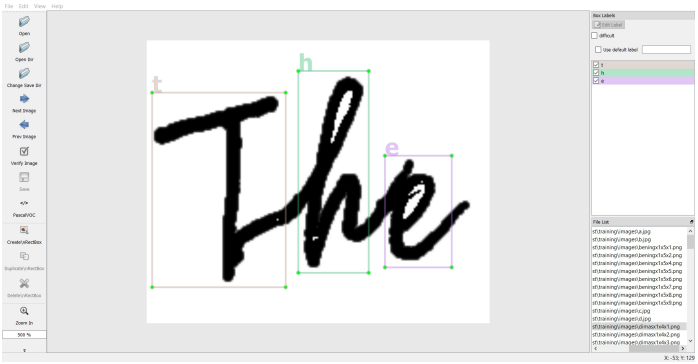


Gambar 3.3: Alur *training* data

3.3.1 Pelabelan *Dataset*

Citra pada *dataset* baik *training sets* maupun *testing sets* kemudian diberi anotasi/*labeling*. *Labeling* merupakan proses pemberian label pada citra dengan memberikan *bounding box* pada objek dan nama *class* objek tersebut [17]. Tujuan dari proses *labeling* yakni untuk mendapatkan *ground-truth bounding box*. Gambar 3.4 menunjukkan proses *labeling* pada citra kata.

Sebelum *labeling* dilakukan, *image* kata harus diletakkan pada satu folder yang sama. Karena *class* yang digunakan pada tugas



Gambar 3.4: Proses *labeling* pada citra

akhir ini sebanyak 26 *class*, huruf latin A sampai Z, *class* harus diedit pada file predefined_class.txt. Pada Gambar 3.4, *labeling* dilakukan secara manual menggunakan tools ***labelImg*** [18]. Setiap objek huruf diberi label sesuai *class*-nya. Untuk kata "The", maka huruf T pada citra diberi *bounding box* dan *class* 't'. Huruf h pada citra diberi *bounding box* dan *class* 'h'. Huruf e pada citra diberi *bounding box* dan *class* 'e'. Dalam *labeling* tulisan tangan latin, perlu diperhatikan batas antar huruf sehingga fitur setiap huruf dapat di-*extract* dengan akurat.

Hasil dari *labeling* adalah file yang berisi informasi koordinat *bounding box* objek pada citra. File ini tersimpan dalam bentuk .xml atau .txt sehingga setiap citra/*image* memiliki file anotasinya masing-masing. Dari proses ini, *dataset* yang dibutuhkan telah disiapkan yang berisi citra dan anotasi citranya.

3.3.2 Proses *Training*

Dalam proses *training*, dibutuhkan konfigurasi layer dalam bentuk .cfg dan *pre-trained weight*. Selain itu, versi *network* yang digunakan harus sama dengan versi *pre-trained weight*-nya. Pada tugas akhir ini, versi yolo yang digunakan adalah YOLOv1.

Sama seperti pada proses *labeling*, file labels.txt pada Darkflow [19] diubah *class* yang akan digunakan, yakni huruf latin a sampai z. *Network* perlu diatur sebelum memulai proses *training*. Konfigurasi

network tersimpan dalam bentuk file .cfg. Beberapa parameter yang diatur pada file cfg diantaranya:

1. Mengubah *classes* pada *region layer* sesuai dengan jumlah *class* yang akan di-*training*, *width&height*, *learning rate*, serta *filters convolutional layer* yang terletak pada dua *layer* terakhir sesuai Persamaan 3.1.

$$filters = num * (classes + 5) \tag{3.1}$$

   dengan $num = 5$ dan *classes* merupakan jumlah *class* yang digunakan.
2. *Uncomment* baris *batch* dan *subdivisions* pada bagian *training* dengan menghilangkan tanda #.

Detail untuk pengaturan parameter *network* untuk tugas akhir ini ditampilkan pada Tabel 3.1.

Tabel 3.1: Tabel detail untuk pengaturan parameter *network*

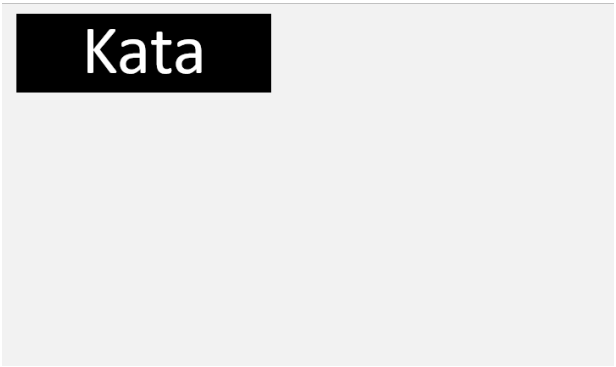
Parameter	***Value***
Width & Height	416x416
Batch	64
Subdivision	32
Epoch	1000
Learning rate	0.001
Decay	0.0005
Momentum	0.9
Class	26
Filters	155

Proses *training* menggunakan *framework* Darkflow pada Google Colaboratory. Selama *training*, Darkflow akan menyimpan hasil sementara ke dalam Tensorflow *checkpoint* atau folder ckpt/.

3.4 Model Pendeteksi Huruf

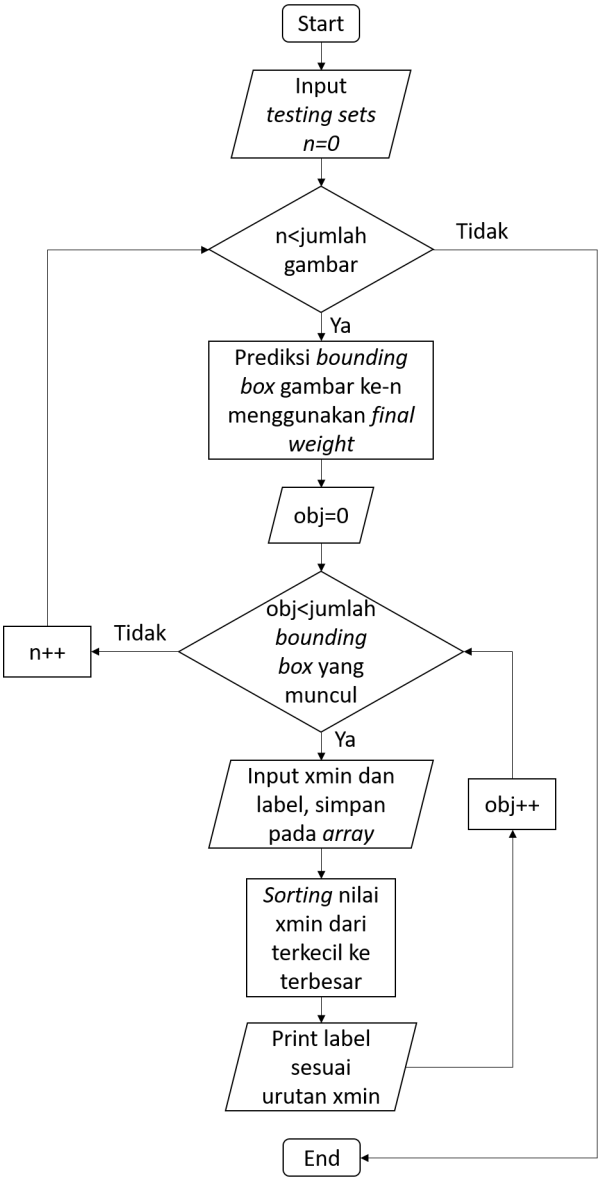
Hasil *training* berupa *final weight* akan tersimpan dalam bentuk file .meta dan .pb. pada folder built_graph/. *Final weight* akan digunakan sebagai *weight* pada model pendeteksi huruf. Pada *training* ini hasil *training* dari *checkpoint* terakhir disimpan sebagai *final weight*.

Selain deteksi, dibutuhkan juga pembacaan setiap huruf. Pembacaan yang dimaksud adalah pembacaan kata. Pembacaan ini berdasarkan pada *class* yang muncul pada prediksi. *Class* yang muncul kemudian digabungkan sehingga bisa dibaca layaknya sebuah kata. Hasil pembacaan ditampilkan pada pojok kiri atas setiap citra seperti pada Gambar 3.5.



Gambar 3.5: Desain *output* hasil pembacaan pada citra

Algoritma pembacaan dimulai dengan mengambil nilai xmin dan label setiap *bounding box* yang muncul setelah prediksi dan menaruhnya dalam sebuah *array* dua dimensi. Isi *array* ini akan diurutkan berdasarkan nilai xmin setiap *bounding box*. Nilai xmin diurutkan dari yang terkecil sampai terbesar. Kemudian label akan ditampilkan sesuai urutan pasangan nilai xmin-nya. Algoritma pembacaan lebih jelas digambarkan pada sebuah *flowchart* seperti pada Gambar 3.6.



Gambar 3.6: *Flowchart* Pembacaan Tulisan Tangan Latin

Halaman ini sengaja dikosongkan

BAB 4
PENGUJIAN DAN ANALISA

Pada bab ini dipaparkan hasil pengujian serta analisa dari desain sistem dan implementasi. Pengujian model deteksi dibagi menjadi 3 bagian antara lain:

1. Pengujian Deteksi Huruf dengan Ukuran Tulisan yang Bervariasi
2. Pengujian Deteksi Huruf pada Citra Satu Kata
3. Pengujian Pembacaan pada Citra Satu Kata

Dengan dilaksanakannya beberapa pengujian tersebut, sehingga dapat ditarik kesimpulan dari pelaksanaan tugas akhir ini.

4.1 Hasil Metodologi

Hasil dari tahapan yang dilakukan pada tugas akhir ini dijelaskan sesuai dengan metodologi yang telah dibuat. Tahapan yang dilakukan adalah sebagai berikut:

1. Hasil pengumpulan citra tulisan tangan latin
2. Hasil *upscaling* dan *grayscaling* citra
3. Hasil pemotongan citra tulisan
4. Hasil *labeling*
5. Hasil *training* data

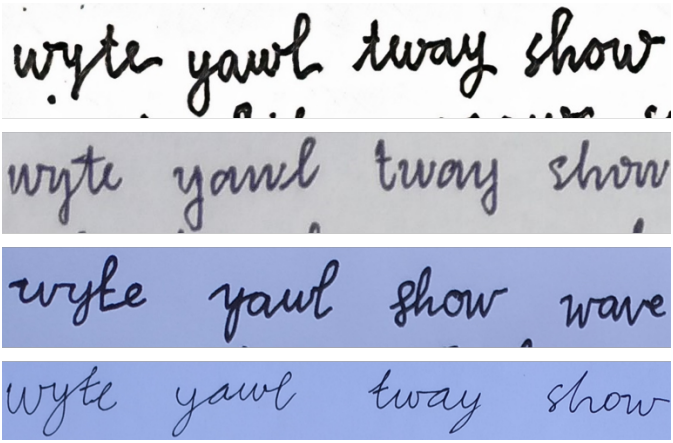
4.1.1 Hasil Pengumpulan Citra Tulisan Tangan Latin

Pengumpulan tulisan tangan latin dilakukan dengan menuliskan kombinasi kata yang telah ditentukan pada sebuah kertas. Jumlah kata yang akan digunakan sebanyak sebanyak 200 kata. Kata ini ditulis oleh empat penulis yang berbeda pada sebuah kertas. Kertas ini diakuisisi dengan cara memfoto kertas tersebut menggunakan kamera sehingga didapatkan citra kertasnya. Hasil akuisisi data ditunjukkan pada Gambar 4.1.

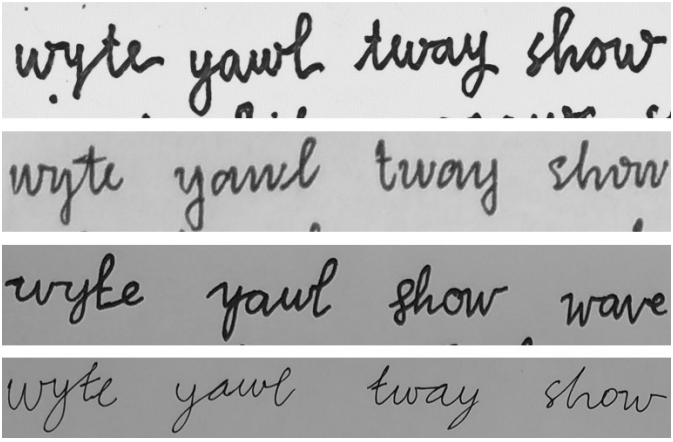
4.1.2 Hasil *Upscaling* dan *Grayscaling* Citra

Citra yang telah diakusisi menggunakan kamera mengalami *upscaling* dan *grayscaling* untuk meningkatkan kualitasnya. Pada

Gambar 4.2 ditunjukkan hasil *upscaling* dan *grayscaling* dari citra tulisan.



Gambar 4.1: Hasil akuisisi data dari berbagai penulis



Gambar 4.2: Hasil *upscaling* dan *grayscaling* dari citra tulisan

4.1.3 Hasil Pemotongan Citra Tulisan

Citra yang telah di-*pre-processing* kemudian dipotong. Pada *dataset* citra tulisan dipotong sehingga satu citra hanya memuat satu kata saja. Kemudian *dataset* dibagi menjadi *training sets* dan *testing sets* masing-masing sebanyak 70% dan 30%. Hasil pemotongan citra tulisan ditunjukkan pada Gambar 4.3.

Gambar 4.3: Hasil pemotongan citra tulisan

4.1.4 Hasil *Labeling*

Citra kata pada *data sets* kemudian diberi label untuk setiap huruf pada citra. Hasil anotasi citra/*labeling* yakni nilai xmin, ymin, xmax, ymax, dan *class* setiap objek yang tersimpan dalam file xml. Salah satu hasil anotasi sebuah objek ditunjukkan pada Gambar 4.4.

```
<object>
        <name>w</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
                <xmin>16</xmin>
                <ymin>126</ymin>
                <xmax>76</xmax>
                <ymax>194</ymax>
        </bndbox>
</object>
```

Gambar 4.4: Hasil anotasi citra kata

Setelah *labeling*, didapatkan jumlah *training sets* untuk setiap *class*. Jumlah *dataset* baik *training sets* dan *testing sets* untuk setiap *class* dan presentase-nya dapat dilihat pada Tabel 4.1.

Tabel 4.1: Tabel jumlah *dataset* untuk setiap *class*

class	***train sets***	***test sets***	***train sets* (dalam %)**	***test sets* (dalam %)**
a	278	160	0.6	0.4
b	104	31	0.8	0.2
c	177	40	0.8	0.2
d	113	44	0.7	0.3
e	306	136	0.7	0.3
f	100	40	0.7	0.3
g	103	37	0.7	0.3
h	120	36	0.8	0.2
i	230	96	0.7	0.3
j	112	32	0.8	0.2
k	99	32	0.8	0.2
l	109	48	0.7	0.3
m	107	33	0.8	0.2
n	107	45	0.7	0.3
o	212	56	0.8	0.2
p	95	28	0.8	0.2
q	98	31	0.8	0.2
r	115	40	0.7	0.3
s	128	35	0.8	0.2
t	102	40	0.7	0.3
u	172	56	0.8	0.2
v	100	32	0.8	0.2
w	103	31	0.8	0.2
x	99	36	0.7	0.3
y	121	40	0.8	0.2
z	107	33	0.8	0.2
Rata-rata			**0.7**	**0.3**

4.1.5 Hasil *Training* Data

Hasil dari proses *training* berupa *final weight* yang digunakan pada model pendeteksi huruf. *Final weight* berasal dari *checkpoint* 22000. *Final weight* yang telah didapatkan kemudian diuji untuk melihat seberapa tepat deteksi huruf yang dihasilkan. Pengujian ini menggunakan *testing sets* yang berupa citra dan diletakkan pada satu folder yang sama. Proses pengujian menggunakan perangkat laptop MSI GL63 8RC dengan spesifikasi perangkat seperti ditampilkan pada Tabel 4.2.

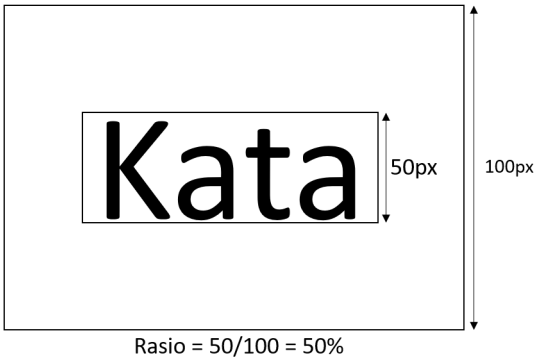
Tabel 4.2: Tabel spesifikasi laptop yang digunakan

Processor	*Intel Core i7-8750H @2.20 GHz*
RAM	8 GB DDR4
Storage	1 TB HDD
Graphic Card	NVIDIA GeForce GTX 1050
Operating System	Windows 10 Home

Pengujian dilakukan menggunakan GPU perangkat. Pada proses pengujian, nilai ambang batas atau *threshold* untuk *confidence score* perlu diatur. Nilai *threshold* untuk *confidence score* diatur antara 0 sampai 1. Apabila nilai *threshold* untuk *confidence score* terlalu kecil, model akan *overfitting*. Sedangkan bila nilai *threshold* untuk *confidence score* terlalu besar, model akan *underfitting*. Hasil pengujian model yang didapatkan kemudian dianalisis sebagai evaluasi. Analisis yang digunakan untuk mengevaluasi model yang dihasilkan yakni analisis *confusion matrix* dan analisis *mean average precision* (mAP).

4.2 Pengujian Deteksi Huruf dengan Ukuran Tulisan yang Bervariasi

Pengujian ini dilakukan untuk mengetahui pengaruh variasi ukuran tulisan terhadap hasil deteksi huruf. Ukuran tinggi tulisan berupa rasio ukuran tinggi tulisan dalam *pixel* terhadap tinggi citra dalam *pixel* secara keseluruhan. Misalkan terdapat tulisan dengan tinggi 50 *pixel* dan tinggi citra 100 *pixel*. Maka rasio ukuran tinggi tulisan sebesar 50%. Ilustrasi rasio ukuran tinggi tulisan terhadap tinggi citra secara keseluruhan ditampilkan pada Gambar 4.5.

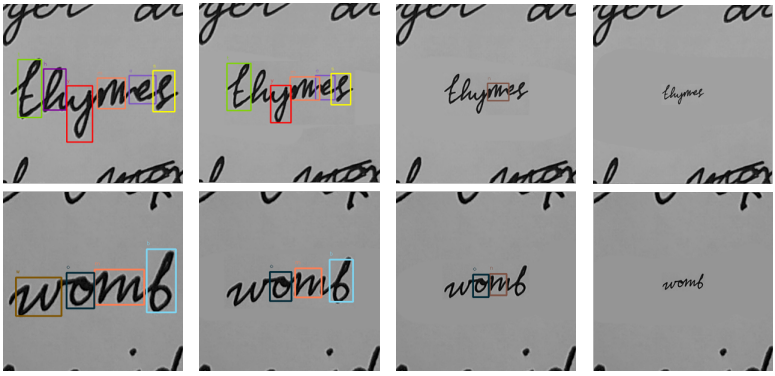


Gambar 4.5: Ilustrasi rasio ukuran tinggi tulisan terhadap tinggi citra secara keseluruhan

Pada pengujian ini terdapat 3 percobaan *training* dengan ukuran tulisan yang berbeda-beda. Percobaan 1 merupakan percobaan *training* menggunakan *dataset* dengan ukuran tinggi tulisan sebesar 50% dari tinggi citra secara keseluruhan. Percobaan 2 merupakan percobaan *training* menggunakan *dataset* dengan ukuran tinggi tulisan sebesar 20% dari tinggi citra secara keseluruhan. Percobaan 3 merupakan percobaan *training* menggunakan *dataset* gabungan antara *dataset* percobaan *training* 1 dan 2. Hasil ketiga percobaan ini berupa model yang akan dicobakan pada *dataset* dengan ukuran tinggi tulisan sebesar 50%, 40%, 20%, dan 10% dari tinggi citra secara keseluruhan.

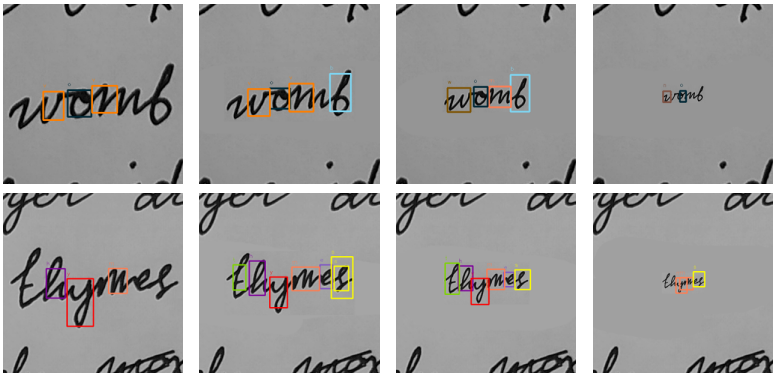
Pada percobaan 1, terlihat bahwa model tidak dapat mendeteksi ukuran tinggi tulisan 10%. Untuk ukuran tinggi tulisan 40% dan 20%, model cukup sulit mendeteksi hurufnya. Meskipun begitu, *bounding box* cukup presisi dalam mendeteksi huruf khususnya pada ukuran tinggi tulisan 50%. Hasil percobaan 1 ditunjukkan pada Gambar 4.6.

Pada percobaan 2, terlihat model dapat mendeteksi huruf pada semua variasi ukuran tulisan. Namun, hasil deteksi untuk ukuran tinggi tulisan 50% tidak sebaik pada ukuran tinggi tulisan 40%. *Bounding box* yang dihasilkan presisi hanya pada ukuran tinggi tulisan 20%. Hasil percobaan 2 ditunjukkan pada Gambar 4.7.



(a) Ukuran tinggi tulisan 50% **(b)** Ukuran tinggi tulisan 40% **(c)** Ukuran tinggi tulisan 20% **(d)** Ukuran tinggi tulisan 10%

Gambar 4.6: Hasil percobaan 1

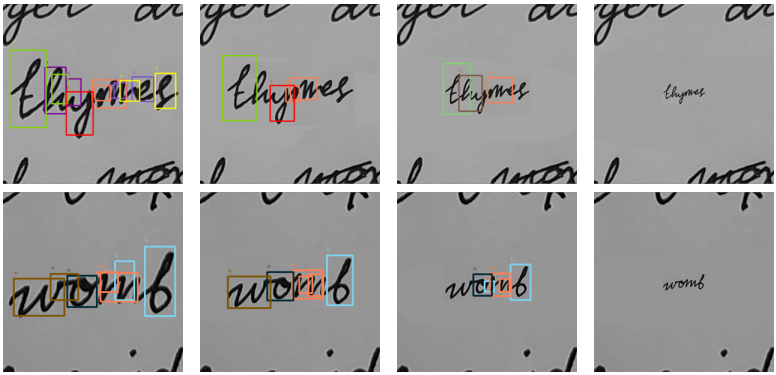


(a) Ukuran tinggi tulisan 50% **(b)** Ukuran tinggi tulisan 40% **(c)** Ukuran tinggi tulisan 20% **(d)** Ukuran tinggi tulisan 10%

Gambar 4.7: Hasil percobaan 2

Pada percobaan 3, terlihat model tidak dapat mendeteksi huruf pada ukuran tinggi tulisan 10%. Selain itu, banyak sekali *bounding box* yang tidak presisi atau *false positive*. Hal ini terjadi dikarenakan model menghasilkan cukup banyak prediksi yang sa-

lah. Meskipun begitu, cukup banyak juga *bounding box* yang presisi. Hasil percobaan 3 ditunjukkan pada Gambar 4.8.



(a) Ukuran tinggi tulisan 50% **(b)** Ukuran tinggi tulisan 40% **(c)** Ukuran tinggi tulisan 20% **(d)** Ukuran tinggi tulisan 10%

Gambar 4.8: Hasil percobaan 3

4.3 Pengujian Deteksi Huruf pada Citra Satu Kata

Model yang telah dibuat harus bisa mendeteksi huruf pada citra tulisan. Pengujian ini bertujuan untuk mengetahui hasil deteksi huruf dan *processing time*-nya pada citra satu kata untuk tiap nilai *threshold confidence score*-nya agar diketahui nilai *threshold confidence score* yang optimal. Pendeteksian dilakukan pada *test sets* dengan jumlah kata sebanyak 56 kata/1268 huruf yang ditulis oleh empat penulis berbeda. Hasil pendeteksian ditunjukkan pada Gambar 4.9.

Berdasarkan Gambar 4.9, terdapat 10 huruf yang ditulis oleh penulis. Terlihat bahwa model yang dibuat berhasil mendeteksi huruf pada citra tersebut saat nilai *threshold confidence score* 0.3. Hal ini dikarenakan warna tulisan sangat kontras dengan *background*-nya sehingga huruf dapat dideteksi dengan baik. Jumlah *predicted box* dan *processing time* per *image* untuk tiap nilai *threshold confidence score* ditampilkan pada Tabel 4.3.

Gambar 4.9: Hasil deteksi huruf pada citra satu kata (*threshold* 0.3)

Pada Tabel 4.3 terlihat saat *threshold confidence score* sama dengan 1 tidak ada *predicted box* yang muncul. Hal ini dikarenakan tidak ada *predicted box* yang memiliki *confidence score* sama dengan 1. *Threshold* akan mengeliminasi *predicted box* jika *predicted box* memiliki *confidence score* lebih rendah dari nilai *threshold*. Hasil prediksi kemudian dianalisis untuk mengevaluasi kinerja sebuah model.

Analisis *confusion matrix* hasil deteksi huruf pada citra satu kata ditunjukkan pada Tabel 4.4. Acuan nilai yang digunakan pada analisis *confusion matrix* yakni nilai *precision*. Nilai *precision*

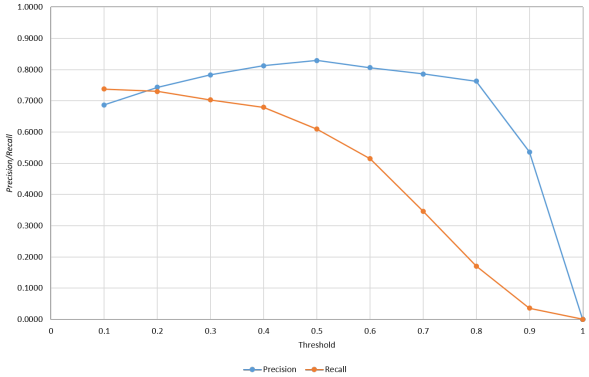
tertinggi saat nilai *threshold confidence score* sama dengan 0.5 yakni 81.63% dan nilai *recall* sebesar 70.66%. Grafik perubahan nilai *precision* dan *recall* pada tiap nilai *threshold confidence score* yang digunakan berdasarkan pada Tabel 4.4 ditampilkan pada Gambar 4.10.

Tabel 4.3: Tabel jumlah *predicted box* dan *processing time* untuk tiap nilai *threshold confidence score* citra satu kata

Threshold	$\hat{\mathbf{B}}$	***Processing time* per *image* (s)**
0.1	1463	0.0747
0.2	1315	0.0800
0.3	1241	0.0776
0.4	1167	0.0763
0.5	1074	0.0768
0.6	933	0.0748
0.7	660	0.0767
0.8	244	0.0750
0.9	33	0.0741
1.0	0	0.0744

Tabel 4.4: Tabel analisis *confusion matrix* hasil deteksi pada citra satu kata

Threshold	**TP**	**FP**	**FN**	***Precision***	***Recall***
0.1	1129	334	139	0.6997	0.8107
0.2	1107	208	161	0.7636	0.7945
0.3	1089	152	179	0.7692	0.7827
0.4	1053	114	215	0.7933	0.7536
0.5	999	75	269	0.8163	0.7066
0.6	886	47	382	0.7995	0.6185
0.7	634	26	634	0.7721	0.4323
0.8	240	4	1028	0.7653	0.1541
0.9	33	0	1235	0.2692	0.0244
1.0	0	0	1268	0.0000	0.0000

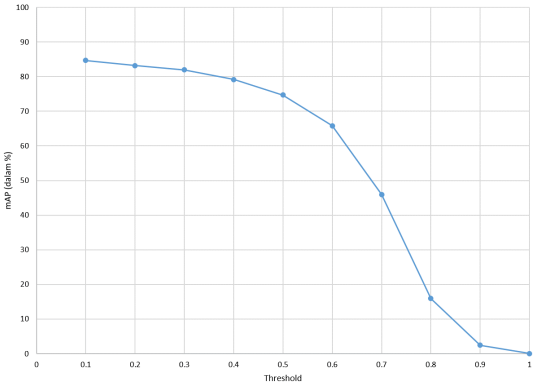


Gambar 4.10: Grafik perubahan nilai *precision* dan *recall* pada citra satu kata

Analisis *mean average precision* (mAP) hasil deteksi huruf pada citra satu kata ditunjukkan pada Tabel 4.5. Perhitungan mAP didapatkan dari menghitung nilai *precision* dan *recall*. Ketika nilai *threshold* sama dengan 1, mAP yang didapatkan yakni 0. Hal ini dikarenakan tidak ada *predicted box* yang muncul. Grafik perubahan nilai *mean average precision* pada citra satu kata ditampilkan pada Gambar 4.11.

Tabel 4.5: Tabel analisis *mean average precision* hasil deteksi pada citra satu kata

Threshold	**mAP (dalam %)**
0.1	84.70
0.2	83.16
0.3	81.93
0.4	79.14
0.5	74.68
0.6	65.81
0.7	45.97
0.8	15.95
0.9	2.44
1.0	0



Gambar 4.11: Grafik perubahan nilai *mean average precision* pada citra satu kata

4.4 Pengujian Pembacaan Tulisan pada Citra Satu Kata

Pengujian ini bertujuan untuk mengetahui pembacaan tulisan pada citra berdasarkan hasil prediksi yang telah dilakukan. Pengujian ini menggunakan 56 kata yang ditulis oleh empat penulis berbeda. Hasil pengujian sistem pembacaan pada citra satu kata ditunjukkan pada Tabel 4.6 dan Gambar 4.12.

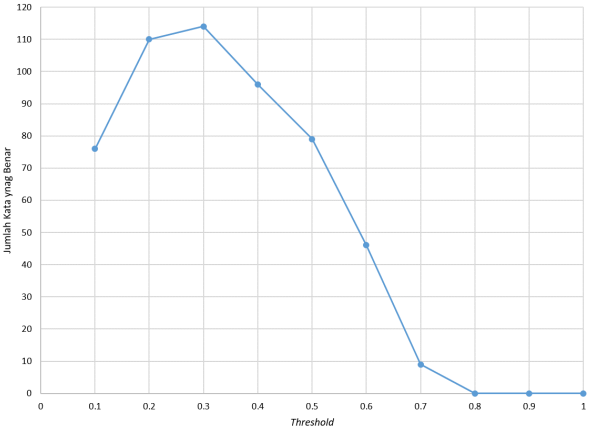
Tabel 4.6: Tabel hasil pembacaan tulisan pada citra satu kata

Threshold	**Jumlah kata yang benar**
0.1	76
0.2	110
0.3	114
0.4	96
0.5	79
0.6	46
0.7	9
0.8	0
0.9	0
1.0	0

Gambar 4.12: Hasil pengujian pembacaan pada citra satu kata (*threshold* 0.3)

Pembacaan tulisan yang paling banyak terjadi ketika nilai *threshold confidence score* sebesar 0.3 yakni 114 kata benar terbaca dari 224 kata. Ketika nilai *threshold* dinaikkan, jumlah kata yang terbaca benar semakin sedikit. Hal ini dikarenakan jumlah *bounding box*

yang muncul semakin sedikit. Grafik perubahan jumlah kata yang benar pada citra satu kata ditampilkan pada Gambar 4.13.



Gambar 4.13: Grafik perubahan jumlah kata yang benar pada citra satu kata

BAB 5
PENUTUP

5.1 Kesimpulan

Dari hasil pengujian yang sudah dilakukan dapat ditarik beberapa kesimpulan sebagai berikut:

1. Berdasarkan hasil pengujian deteksi huruf dengan ukuran tulisan yang bervariasi, model *training* yang cocok yakni menggunakan *dataset* dengan ukuran tinggi tulisan sebesar 50%.
2. Berdasarkan hasil pengujian deteksi huruf dan pembacaan pada citra satu kata, nilai *threshold confidence score* yang ideal untuk digunakan yakni 0.3, dengan *processing time* per *image* ialah 0.0776 detik, skor *precision* sebesar 76.92%, nilai *recall* sebesar 78.27%, dan skor *mean average precison* (mAP) sebesar 81.93%.
3. Sistem pembacaan tulisan tangan sudah dapat membaca huruf yang terdeteksi secara urut dari kiri ke kanan.

5.2 Saran

Untuk pengembangan penelitian selanjutnya terdapat beberapa saran sebagai berikut:

1. Dibutuhkan penambahan *dataset* agar akurasi yang didapatkan bisa meningkat.
2. Jumlah *epoch* yang tepat dapat membuat waktu dalam proses *training* data lebih cepat dan menghasilkan nilai *loss* yang optimal.
3. Sistem yang akan dikembangkan sebaiknya mampu mendeteksi lebih dari satu kata.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] J. Ryu, H. I. Koo, and N. I. Choo, "Word segmentation method for handwritten documents based on structured learning," IEEE Signal Processing Letters, vol. 22, no. 8, pp. 1161–1165, 2015. (Dikutip pada halaman i, iii, 2, 5).

[2] "A comprehensive guide to convolutional neural networks." https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Terakhir diakses pada 2 Juli 2020. (Dikutip pada halaman ix, 8).

[3] "Apa itu convolutional neural network?." https://mc.ai/apa-itu-convolutional-neural-network/. Terakhir diakses pada 2 Juli 2020. (Dikutip pada halaman ix, 8).

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once unified, real-time object detection," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016. (Dikutip pada halaman ix, 9, 10, 11).

[5] "Intersection over union (iou) for object detection." https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. Terakhir diakses pada 2 Juli 2020. (Dikutip pada halaman ix, 13, 14).

[6] "Breaking down mean average precision (map)." https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52/. Terakhir diakses pada 28 Juli 2020. (Dikutip pada halaman ix, 18).

[7] "Surat pos dalam dan luar negeri yang dikirim dan diterima, 2015 (update terakhir : 18 oct 2017)." https://jatim.bps.go.id/statictable/2017/10/18/701/surat-pos-dalam-dan-luar-negeri-yang-dikirim-dan-diterima-di-provinsi-jawa-timur-2015.html. Terakhir diakses pada 16 April 2020. (Dikutip pada halaman 1).

[8] "Paket pos dalam dan luar negeri yang dikirim dan diterima, 2015 (update terakhir : 18 oct 2017)." https://jatim.bps.go.id/statictable/2017/10/18/700/paket-pos-dalam-dan-luar-negeri-yang-dikirim-dan-diterima-di-provinsi-jawa-timur-2015.html. Terakhir diakses pada 16 April 2020. (Dikutip pada halaman 1).

[9] M. Arun, S. Arivazhagan, and D. Rathina, "Handwritten text segmentation using pixel based approach," in Third International Conference on Trends in Electronics and Informatics (ICOEI 2019), IEEE, 2019. (Dikutip pada halaman 5).

[10] L. E. Damayanti, Pengenalan Tulisan Tangan Naskah Aksara Jawa Aturan Penulisan Mardi Kawi Menjadi Huruf Alfabet Pada Media Kertas Menggunakan Metode Convolutonal Neural Network (CNN) Arsitektur You Only Look Once. Institut Teknologi Sepuluh Nopember, 2020. (Dikutip pada halaman 5).

[11] V. Jayasundara, S. Jayasekara, H. Jayasekara, J. Rajasegaran, S. Seneviratne, and R. Rodrigo, "Textcaps : Handwritten character recognition with very small datasets," 2019. (Dikutip pada halaman 6).

[12] W. Yang and Z. Jiachun, "Real-time face detection based on yolo," pp. 221–224, 2018. (Dikutip pada halaman 6).

[13] M. Krišto, M. Ivasic-Kos, and M. Pobar, "Thermal object detection in difficult weather conditions using yolo," IEEE Access, vol. 8, pp. 125459–125476, 2020. (Dikutip pada halaman 6).

[14] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the yolo detector," in 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–10, 2018. (Dikutip pada halaman 6).

[15] "Mengenal accuracy, precision, recall dan specificity serta yang diprioritaskan dalam machine learning," Terakhir diakses pada 28 April 2020. (Dikutip pada halaman 15).

[16] J. Cartucho, R. Ventura, and M. Veloso, "Robust object recognition through symbiotic deep learning in mobile robots," pp. 2336–2341, 2018. (Dikutip pada halaman 18).

[17] "Tutorial deteksi objek menggunakan yolo (you only look once)." https://medium.com/@andikirahyagara/tutorial-yolo-you-only-look-once-for-absolutely-noob-c4d5f3751e1f. Terakhir diakses pada 3 April 2020. (Dikutip pada halaman 21).

[18] "Labelimg." https://github.com/tzutalin/labelImg. Terakhir diakses pada 4 April 2020. (Dikutip pada halaman 22).

[19] Trieu and T. Hoang, "Darkflow," GitHub Repository. Available online: https://github. com/thtrieu/darkflow (Terakhir diakses pada 18 November 2019), 2018. (Dikutip pada halaman 22).

Halaman ini sengaja dikosongkan

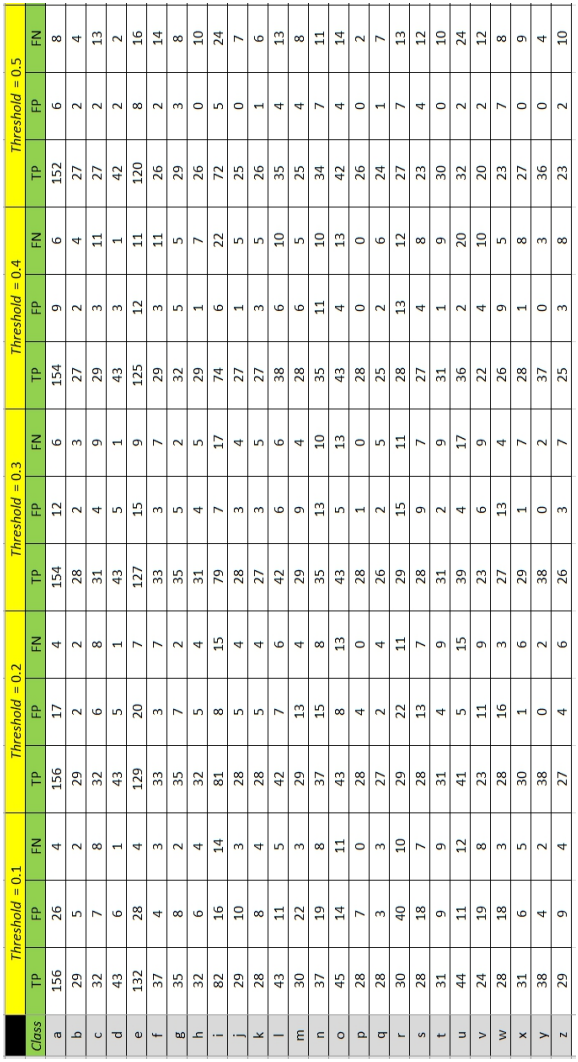
LAMPIRAN

Hasil anotasi salah satu citra kata

```
<annotation>
<folder>besarimg</folder>
<filename>ana_125.png</filename>
<path>D:\darkflow-master\test\training\besarimg\ana_125.png</
    path>
<source>
<database>Unknown</database>
</source>
<size>
<width>278</width>
<height>277</height>
<depth>1</depth>
</size>
<segmented>0</segmented>
<object>
<name>w</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<difficult>0</difficult>
<bndbox>
<xmin>16</xmin>
<ymin>126</ymin>
<xmax>76</xmax>
<ymax>194</ymax>
</bndbox>
</object>
<object>
<name>o</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<difficult>0</difficult>
<bndbox>
<xmin>80</xmin>
<ymin>132</ymin>
<xmax>115</xmax>
<ymax>193</ymax>
</bndbox>
</object>
<object>
<name>m</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<difficult>0</difficult>
```

```
<bndbox>
<xmin>123</xmin>
<ymin>134</ymin>
<xmax>186</xmax>
<ymax>189</ymax>
</bndbox>
</object>
<object>
<name>b</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<difficult>0</difficult>
<bndbox>
<xmin>201</xmin>
<ymin>79</ymin>
<xmax>251</xmax>
<ymax>183</ymax>
</bndbox>
</object>
</annotation>
```

Hasil analisis *confusion matrix* pada citra satu kata



	Threshold = 0.1			Threshold = 0.2			Threshold = 0.3			Threshold = 0.4			Threshold = 0.5		
Class	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
a	156	26	4	156	17	4	154	12	6	154	9	6	152	6	8
b	29	5	2	29	2	2	28	2	3	27	2	4	27	2	4
c	32	7	8	32	6	8	31	4	9	29	3	11	27	2	13
d	43	6	1	43	5	1	43	5	1	43	3	1	42	2	2
e	132	28	4	129	20	7	127	15	9	125	12	11	120	8	16
f	37	4	3	33	3	7	33	3	7	29	3	11	26	2	14
g	35	8	2	35	7	2	35	5	2	32	5	5	29	3	8
h	32	6	4	32	5	4	31	4	5	29	1	7	26	0	10
i	82	16	14	81	8	15	79	7	17	74	6	22	72	5	24
j	29	10	3	28	5	4	28	3	4	27	1	5	25	0	7
k	28	8	4	28	5	4	27	3	5	27	3	5	26	1	6
l	43	11	5	42	7	6	42	6	6	38	6	10	35	4	13
m	30	22	3	29	13	4	29	9	4	28	6	5	25	4	8
n	37	19	8	37	15	8	35	13	10	35	11	10	34	7	11
o	45	14	11	43	8	13	43	5	13	43	4	13	42	4	14
p	28	7	0	28	4	0	28	1	0	28	0	0	26	0	2
q	28	3	3	27	2	4	26	2	5	25	2	6	24	1	7
r	30	40	10	29	22	11	29	15	11	28	13	12	27	7	13
s	28	18	7	28	13	7	28	9	7	27	4	8	23	4	12
t	31	9	9	31	4	9	31	2	9	31	1	9	30	0	10
u	44	11	12	41	5	15	39	4	17	36	2	20	32	2	24
v	24	19	8	23	11	9	23	6	9	22	4	10	20	2	12
w	28	18	3	28	16	3	27	13	4	26	9	5	23	7	8
x	31	6	5	30	1	6	29	1	7	28	1	8	27	0	9
y	38	4	2	38	0	2	38	0	2	37	0	3	36	0	4
z	29	9	4	27	4	6	26	3	7	25	3	8	23	2	10

Gambar 1: Hasil analisis *confusion matrix* pada citra satu kata (Bagian 1)

	Threshold = 0.6			Threshold = 0.7			Threshold = 0.8			Threshold = 0.9			Threshold = 1.0		
Class	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
a	143	4	17	113	2	47	46	2	114	5	0	155	0	0	160
b	24	2	7	21	1	10	12	0	19	4	0	27	0	0	31
c	24	1	16	20	0	20	11	0	29	2	0	38	0	0	40
d	37	1	7	25	1	19	5	0	39	0	0	44	0	0	44
e	111	6	25	89	3	47	43	1	93	7	0	129	0	0	136
f	23	1	17	18	1	22	8	0	32	1	0	39	0	0	40
g	25	2	12	16	1	21	7	0	30	2	0	35	0	0	37
h	23	0	13	18	0	18	3	0	33	0	0	36	0	0	36
i	62	4	34	46	4	50	25	1	71	2	0	94	0	0	96
j	20	0	12	12	0	20	3	0	29	0	0	32	0	0	32
k	24	0	8	17	0	15	5	0	27	1	0	31	0	0	32
l	29	4	19	20	3	28	6	0	42	1	0	47	0	0	48
m	19	1	14	13	0	20	2	0	31	0	0	33	0	0	33
n	28	5	17	12	2	33	2	0	43	0	0	45	0	0	45
o	39	3	17	32	2	24	16	0	40	1	0	55	0	0	56
p	23	0	5	18	0	10	4	0	24	0	0	28	0	0	28
q	23	1	8	14	1	17	2	0	29	0	0	31	0	0	31
r	20	3	20	8	1	32	5	0	35	0	0	40	0	0	40
s	20	3	15	11	1	24	3	0	32	2	0	33	0	0	35
t	28	0	12	18	0	22	3	0	37	0	0	40	0	0	40
u	27	0	29	18	0	38	7	0	49	1	0	55	0	0	56
v	18	0	14	13	0	19	10	0	22	2	0	30	0	0	32
w	21	4	10	14	2	17	5	0	26	2	0	29	0	0	31
x	22	0	14	14	0	22	3	0	33	0	0	36	0	0	36
y	33	0	7	19	0	21	4	0	36	0	0	40	0	0	40
z	20	2	13	15	1	18	0	0	33	0	0	33	0	0	33

Gambar 2: Hasil analisis *confusion matrix* pada citra satu kata (Bagian 2)

	Threshold = 0.1			Threshold = 0.2			Threshold = 0.3			Threshold = 0.4			Threshold = 0.5		
Class	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
a	156	26	4	156	17	4	154	12	6	154	9	6	152	6	8
b	29	5	2	29	2	2	28	2	3	27	2	4	27	2	4
c	32	7	8	32	6	8	31	4	9	29	3	11	27	2	13
d	43	6	1	43	5	1	43	5	1	43	3	1	42	2	2
e	132	28	4	129	20	7	127	15	9	125	12	11	120	8	16
f	37	4	3	33	3	7	33	3	7	29	3	11	26	2	14
g	35	8	2	35	7	2	35	5	2	32	5	5	29	3	8
h	32	6	4	32	5	4	31	4	5	29	1	7	26	0	10
i	82	16	14	81	8	15	79	7	17	74	6	22	72	5	24
j	29	10	3	28	5	4	28	3	4	27	1	5	25	0	7
k	28	8	4	28	5	4	27	3	5	27	3	5	26	1	6
l	43	11	5	42	7	6	42	6	6	38	6	10	35	4	13
m	30	22	3	29	13	4	29	9	4	28	6	5	25	4	8
n	37	19	8	37	15	8	35	13	10	35	11	10	34	7	11
o	45	14	11	43	8	13	43	5	13	43	4	13	42	4	14
p	28	7	0	28	4	0	28	1	0	28	0	0	26	0	2
q	28	3	3	27	2	4	26	2	5	25	2	6	24	1	7
r	30	40	10	29	22	11	29	15	11	28	13	12	27	7	13
s	28	18	7	28	13	7	28	9	7	27	4	8	23	4	12
t	31	9	9	31	4	9	31	2	9	31	1	9	30	0	10
u	44	11	12	41	5	15	39	4	17	36	2	20	32	2	24
v	24	19	8	23	11	9	23	6	9	22	4	10	20	2	12
w	28	18	3	28	16	3	27	13	4	26	9	5	23	7	8
x	31	6	5	30	1	6	29	1	7	28	1	8	27	0	9
y	38	4	2	38	0	2	38	0	2	37	0	3	36	0	4
z	29	9	4	27	4	6	26	3	7	25	3	8	23	2	10

Gambar 3: Hasil analisis *confusion matrix* pada citra satu kata (Bagian 3)

	Threshold = 0.6		Threshold = 0.7		Threshold = 0.8		Threshold = 0.9		Threshold = 1.0	
class	precision	recall	precision	recall	precision	recall	precision	recall	precision	recall
a	0.9728	0.8938	0.9826	0.7063	0.9583	0.2875	1.0000	0.0313	0.0000	0.0000
b	0.9231	0.7742	0.9545	0.6774	1.0000	0.3871	1.0000	0.1290	0.0000	0.0000
c	0.9600	0.6000	1.0000	0.5000	1.0000	0.2750	0.0000	0.0500	0.0000	0.0000
d	0.9737	0.8409	0.9615	0.5682	1.0000	0.1136	0.0000	0.0000	0.0000	0.0000
e	0.9487	0.8162	0.9674	0.6544	0.9773	0.3162	1.0000	0.0515	0.0000	0.0000
f	0.9583	0.5750	0.0000	0.4500	0.0000	0.2000	0.0000	0.0250	0.0000	0.0000
g	0.9259	0.6757	0.9412	0.4324	1.0000	0.1892	0.0000	0.0541	0.0000	0.0000
h	1.0000	0.6389	1.0000	0.5000	1.0000	0.0833	0.0000	0.0000	0.0000	0.0000
i	0.9394	0.6458	0.9200	0.4792	0.9615	0.2604	0.0000	0.0208	0.0000	0.0000
j	1.0000	0.6250	1.0000	0.3750	1.0000	0.0938	0.0000	0.0000	0.0000	0.0000
k	1.0000	0.7500	1.0000	0.5313	1.0000	0.1563	1.0000	0.0313	0.0000	0.0000
l	0.8788	0.6042	0.8696	0.4167	1.0000	0.1250	0.0000	0.0208	0.0000	0.0000
m	0.9500	0.5758	1.0000	0.3939	1.0000	0.0606	0.0000	0.0000	0.0000	0.0000
n	0.8485	0.6222	0.8571	0.2667	1.0000	0.0444	0.0000	0.0000	0.0000	0.0000
o	0.9286	0.6964	0.9412	0.5714	1.0000	0.2857	1.0000	0.0179	0.0000	0.0000
p	1.0000	0.8214	1.0000	0.6429	1.0000	0.1429	0.0000	0.0000	0.0000	0.0000
q	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
r	0.8696	0.5000	0.8889	0.2000	1.0000	0.1250	0.0000	0.0000	0.0000	0.0000
s	0.8696	0.5714	0.9167	0.3143	1.0000	0.0857	1.0000	0.0571	0.0000	0.0000
t	1.0000	0.7000	1.0000	0.4500	1.0000	0.0750	0.0000	0.0000	0.0000	0.0000
u	1.0000	0.4821	1.0000	0.3214	1.0000	0.1250	1.0000	0.0179	0.0000	0.0000
v	0.0000	0.5625	0.0000	0.4063	0.0000	0.3125	0.0000	0.0625	0.0000	0.0000
w	0.8400	0.6774	0.8750	0.4516	0.0000	0.1613	0.0000	0.0645	0.0000	0.0000
x	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
y	1.0000	0.8250	1.0000	0.4750	1.0000	0.1000	0.0000	0.0000	0.0000	0.0000
z	0.0000	0.6061	0.0000	0.4545	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Rata-rata	**0.7995**	**0.6185**	**0.7721**	**0.4323**	**0.7653**	**0.1541**	**0.2692**	**0.0244**	**0.0000**	**0.0000**

Gambar 4: Hasil analisis *confusion matrix* pada citra satu kata (Bagian 4)

BIOGRAFI PENULIS

Aisyah Nurul Hidayah, lahir di Ujung Pandang pada tanggal 13 Juli 1998. Merupakan anak pertama dari dua bersaudara. Penulis telah menyelesaikan pendidikan di SDN Bontokamase (2004-2010), MTsN 1 Makassar (2010-2013), dan SMAN 5 Gowa (2013-2016). Penulis diterima di Program Studi S-1 Departemen Teknik Komputer Fakultas Teknologi Elektro dan Informatika Cerdas. Penulis memiliki minat di bidang keilmuan dan seni. Selama masa kuliah, penulis aktif berpatisipasi sebagai panitia *event* seperti ELECTRA, MAGE, dan PEKSIMITS, menjadi asisten Laboratorium Komputasi Multimedia B401, dan juga aktif dalam Internet of Things (IoT) *developing*. Pembaca yang memiliki kepentingan kepada penulis seperti memberikan kritik, saran, atau pertanyaan mengenai tugas akhir ini dapat menghubungi penulis melalui email **aisyahnrlh@gmail.com**.

Halaman ini sengaja dikosongkan